

Article

Fast Fractal Image Compression Using Non-Uniform Partition

ManLong Li ^{1,2}  and KinTak U ^{1,*} ¹ Faculty of Innovation Engineering, Macau University of Science and Technology, Taipa, Macau SAR 999078, China; li_ml@bit.edu.cn² School of Computing, Beijing Institute of Technology, Zhuhai 519085, China

* Correspondence: ktu@must.edu.mo

Abstract

Fractal image compression achieves high compression ratios but suffers from prohibitively long encoding times and limited reconstruction quality. To address these limitations, we propose fast fractal image compression using non-uniform partition (FFICNUP), a hybrid algorithm that adaptively partitions range blocks (R-blocks) and domain blocks (D-blocks) based on local texture and edge content. Smaller R-blocks are employed in texture-rich regions or edge-dense areas to preserve fine details, while larger R-blocks are adopted in smooth regions to accelerate encoding. By integrating a Task-Serial Workflow with Data-Parallel Vectorization and adaptive block partitioning, FFICNUP substantially accelerates both encoding and decoding processes while enhancing reconstruction fidelity and compression ratios. Experimental results demonstrate that the proposed FFICNUP method significantly outperforms conventional fractal image compression (FIC) approaches. By leveraging vectorized parallelization, the proposed FFICNUP achieves state-of-the-art (SOTA) decoding speed with a $14\times$ acceleration, reduces encoding latency by three orders of magnitude, improves the Peak Signal-to-Noise Ratio (PSNR) by up to 5.19 dB, and attains a compression ratio 2.21 times higher than that of conventional FIC. Validated across both CPU and GPU platforms, FFICNUP dynamically balances encoding speed, reconstruction quality, compression ratio, and latency across varying image sizes, demonstrating its suitability for practical engineering applications.

Keywords: fractal image compression; non-uniform partition; data-parallel vectorization; adaptive block partition

1. Introduction

Fractal Image Compression (FIC) is a promising image coding technique renowned for its resolution-independent decoding capability and high compression potential. Owing to these properties, FIC has emerged as a practical approach for reducing the storage requirements and transmission bandwidth demands of digital images. It leverages fractal self-similarity to achieve high compression ratios (CRs) while preserving image quality. Unlike traditional image compression methods, it uses mathematical transformations to represent images, making it particularly effective for natural photos with repetitive patterns, such as landscapes or textures. Its application is particularly prominent in fields that require efficient storage and transmission of complex visual data, where high-resolution images must be compactly represented in limited-bandwidth environments. Fractal image compression is also utilized for digital archiving, allowing museums and libraries to store high-resolution images of artifacts or manuscripts efficiently. It also allows high-fidelity,



Academic Editor: José Carlos Castillo

Received: 14 January 2026

Revised: 7 February 2026

Accepted: 19 February 2026

Published: 25 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

memory-efficient text rendering in computer graphics and gaming, improving performance on resource-constrained systems.

Fractal image compression originated in 1988 with Barnsley's Iterated Function System (IFS) [1] and was subsequently advanced by Jacquin's Partitioned Iterated Function System (PIFS) [2]. This advancement forms the basis of what is commonly referred to as basic fractal image compression (BFIC). Due to its high compression ratio and cross-scale reconstruction capability, it has significant potential in medical imaging, satellite imaging, and multimedia processing. However, the BFIC method has been known to have relatively high encoding time during the optimal block-matching period since its proposal. Since then, researchers have proposed and developed numerous practical, fast fractal image compression algorithms, focusing on block segmentation schemes, block-matching searches, computational frameworks, and related areas. The block segmentation method determines how to divide an image into R-blocks and D-blocks, significantly impacting compression quality, computational complexity, and encoding time. Fisher [3] introduced fixed block partitioning (e.g., 8×8 pixels) in 1995, which is valued for its simplicity and low computational cost, making it suitable for basic image storage but less effective for heterogeneous regions. Nandi et al. [4] proposed adaptive quadtree partitioning, which dynamically adjusts block shapes via edge detection or texture analysis, achieving better CR. Dwivedi and Mishra [5] integrated the Discrete Wavelet Transform (DWT) with a fixed-block partition to enhance compression ratios and optimize storage applications. Smriti et al. [6] employed a PSO-ALO hybrid metaheuristic to refine partition, thereby improving compression ratios and PSNR, especially suitable for high-resolution image analysis. Tang et al. [7] combined adaptive partitioning with wavelet transforms to improve super-resolution reconstruction of complex images. Panigrahy et al. [8] developed a space-constrained quadtree partition method that controls segmentation levels through distortion tolerance, resulting in superior PSNR compared to non-search methods, making it apt for nearly real-time applications. Yu et al. [9] utilized an 8×8 fixed partition combined with FIC, Arnold transform, and Frobenius canonical form to enhance security in DICOM medical image encryption. Li et al. [10] proposed the FICANRP algorithm, which employs an adaptive non-uniform rectangular partition scheme. Their method dynamically segments R-blocks and non-overlapping D-blocks based on local texture and structural features, significantly reducing computational complexity and encoding time while improving the CR.

Range-domain block matching and its computational mechanisms form the foundation of fractal image compression (FIC), allowing the exploitation of self-similarities between R-blocks and D-blocks to improve compression ratios and reconstruction quality. Methods in this category include search-based and non-search approaches, as well as feature-driven computation strategies. Jacquin's seminal work introduced search-based matching through exhaustive or heuristic searches, yielding high-fidelity reconstructions ideally suited for archival purposes, albeit at the cost of substantial computational overhead. Fisher [3] introduced non-search fixed mapping, reducing encoding time in resource-limited scenarios. Wohlberg and de Jager [11] developed a block-matching search method for fractal image compression that utilizes DCT coefficients and K-D tree-based nearest-neighbor searches, thereby reducing computational complexity from $O(N^2)$ to $O(N \times \log 2N)$ while maintaining reconstruction quality. Wang and Zheng [12] enhanced search efficiency by utilizing block classification and sorting, resulting in a higher PSNR for the Lena image. Jaferzadeh et al. [13] employed Local Binary Patterns to accelerate searches, increasing encoding speed by $2\text{--}3\times$. Cao et al. [14] proposed orthogonal sparse grayscale transforms to match complex textures better. Abedellatif et al. [15] combined a non-exhaustive search with local features, reducing encoding time by $2\text{--}3\times$. Li et al. [16] employed generalized Mandelbrot sets and escape-time algorithms to generate feature blocks, achieving a $2\text{--}3$ dB increase in PSNR.

Li et al. [17] refined the centroid radius method, boosting encoding speed by $4.68\times$. Panigrahy et al. [8] introduced a space-constrained five-domain search, accelerating encoding by $1000\times$. Li and KinTak U [10] applied local search with non-overlapping D-blocks, achieving encoding speeds $54.14\text{--}1448.73\times$ faster with a PSNR improvement of $0.25\text{--}4.8$ dB than BFC. These advancements, encompassing exhaustive, heuristic, and non-search strategies, have significantly enhanced FIC's efficiency and quality, supporting applications in real-time video streaming, medical imaging, and satellite imagery processing.

The computational framework of fractal image compression (FIC) leverages both serial and parallel processing strategies to optimize the encoding pipeline. While serial execution ensures logical consistency, the parallel architecture is specifically engineered to maximize throughput and accelerate execution, making it highly effective for latency-sensitive applications—ranging from high-resolution remote sensing to real-time video streaming. Serial computing, as described by Barnsley [1], relies on sequential R-block matching on single-loop processing. It offers simplicity but suffers from high encoding times, making it suitable for low-complexity systems. To address this, parallel computing strategies have emerged that leverage multi-core CPUs, GPUs, or FPGAs to distribute block-matching tasks. Varghese and S. [18] explored GPU thread scheduling to accelerate encoding, achieving significant speedups but noting potential error accumulation in high-precision tasks. Saad and Abdullah [19] introduced a deep data pipelining approach using an FPGA, which reduces encoding time by up to 50% for real-time embedded systems, such as those used in medical imaging. Panigrahy et al. [8] demonstrated GPU-based parallelization for high-resolution satellite imagery, achieving a $1000\times$ increase in encoding speed while maintaining the same PSNR. These frameworks, which combine serial simplicity with parallel efficiency, have evolved to support a diverse range of applications. Future research is likely to explore hybrid models that integrate deep learning for further optimization.

Building upon prior research, scholars have leveraged the self-similarity and local repeatability inherent in images by substituting pixel values with descriptors of repetitive patterns, with a primary emphasis on reducing the encoding time (ET) in fractal image compression. However, most existing FIC studies employ fixed-block partitioning schemes that do not adequately account for the internal textures and features of images. This results in degraded reconstruction quality for large block sizes and limited CR for too small block sizes. Consequently, non-uniform partition emerges as a pivotal strategy, dynamically adjusting the sizes and shapes of R-blocks and D-blocks based on local textures, edges, and image complexity to enhance compression efficiency. Compared to a fixed uniform partition, a non-uniform partition better accommodates image heterogeneity, reduces computational redundancy in smooth regions, and enhances self-similarity detection in detailed areas, thereby simultaneously decreasing ET while improving PSNR and CR.

Moreover, previous FIC research has ignored the processing of high-resolution, large images—despite their growing use in fields such as remote sensing, medical imaging, and virtual reality. While traditional FIC methods work well for medium-sized photos, they become computationally prohibitive during decoding and reconstruction for massive datasets. For example, decoding images larger than 2048×2048 pixels can take tens of seconds, making them impractical for real-time applications such as video streaming or interactive systems. This challenge primarily stems from the iterative process of FIC decoding, which reconstructs images from compact fractal codes via repeated affine transformations and fixed-point iterations. As image resolution increases, the number of necessary iterations and computational effort increase significantly, resulting in nearly exponential growth in decoding complexity.

This paper aims to bridge the gap between the theoretical compression efficiency of fractal image compression (FIC) and its practical deployment limitations. To address these

challenges, we propose a fast fractal image compression algorithm that simultaneously reduces encoding and decoding times, enhances reconstruction quality, and improves overall compression efficiency. The proposed fast fractal image compression using non-uniform partition (FFICNUP) integrates innovative non-uniform block partitioning, a Task-Serial Workflow enhanced by Data-Parallel Vectorization, and adaptive hybrid fractal compression algorithms, thereby enabling real-time encoding and decoding across diverse applications. Building on prior methods that seek to minimize encoding and decoding times while boosting Peak Signal-to-Noise Ratio (PSNR) and compression ratio (CR), this work advances FIC for high-efficiency compression without compromising fidelity, facilitating seamless integration into resource-constrained environments. The main contributions of this research are summarized as follows:

1. We propose a local texture- and feature-guided adaptive non-uniform partition scheme for R-blocks and D-blocks. R-blocks undergo non-overlapping segmentation in a content-adaptive manner, while D-blocks undergo a two-stage refinement process—initial uniform non-overlapping segmentation followed by non-uniform subdivision. This approach improves matching accuracy in textured regions, reduces redundancy in smooth areas, and enhances PSNR without compromising CR.
2. We propose a Task-Serial Workflow integrated with Data-Parallel Vectorization, in which all computing-intensive operations (including downsampling, geometric transformation, and block similarity matching) are reformulated as highly optimized batched SIMD (Single Instruction Multiple Data) kernels while maintaining strict task-level sequential ordering. This design eliminates redundant data reading, writing, and computation computations, dramatically reduces encoding and decoding time, and achieves state-of-the-art (SOTA) decoding speed for fractal image compression while simultaneously improving PSNR and compression ratio.
3. We present a hybrid fractal compression algorithm that adaptively selects among CPU-serial, multi-threaded CPU, and GPU-accelerated execution modes based on image resolution and hardware constraints, achieving an optimal trade-off among encoding and decoding speed, reconstruction fidelity, compression ratio, and latency across diverse platforms for practical applications.

The remainder of this paper is organized as follows. Section 2 provides a brief review of the fundamentals of fractal image compression and non-uniform partition. Section 3 presents a comprehensive survey of fractal image compression techniques based on non-uniform partition, including detailed descriptions and analyses of representative algorithms. Section 4 conducts a thorough performance evaluation, examining key compression metrics and experimental results under diverse conditions. Finally, Section 5 concludes the paper and outlines directions for future work research.

2. Related Work

As noted earlier, fractal image compression significantly reduces data size by exploiting self-similarity and local repetition within an image, replacing storage of pixel values with descriptions of recurring patterns. However, traditional FIC relies heavily on fixed block segmentation. It does not fully adapt to the local textures and structural features of the image, leading to a trade-off between compression efficiency and reconstruction quality. The image non-uniform partition dynamically divides blocks into various sizes and shapes based on local complex textures or irregular structures. It combines matching calculation strategies for block-size classification to markedly reduce ET while also optimizing PSNR and maintaining a high CR. The following provides a detailed overview of the research progress and key technologies in fractal image compression and non-uniform partition.

2.1. The Fractal Image Compression

Fractal image compression is a lossy compression method based on fractal geometry that exploits the self-similarity of natural images, where similar structures recur across different regions or scales. Unlike traditional pixel-based techniques, such as JPEG, which store raw pixel data and can produce artifacts at high compression ratios, FIC encodes images using mathematical rules that significantly reduce storage requirements. The approach is grounded on three theoretical foundations: the Collage Theorem, the Iterated Function System (IFS), and affine transformations. During encoding, the image is divided into non-overlapping R-blocks, and for each R-block, a larger, visually similar D-block is searched within the same image. A contractive affine transformation—encompassing geometric operations such as scaling, rotation, and reflection, along with grayscale intensity adjustments—is then used to map the D-block onto the R-block, with the resulting parameters forming the fractal code.

The decoding process is driven by the IFS, which iteratively applies the encoded affine transformations to reconstruct the image. According to the Collage Theorem, if the transformed image is sufficiently close to the original (i.e., the collage error is small), this iterative procedure converges to a fixed-point image that closely approximates the original. This framework not only captures intrinsic image redundancy efficiently but also inherently supports resolution-independent decoding, enabling high-quality reconstruction at any scale.

2.1.1. Collage Theorem

The Collage Theorem [1], proposed by Barnsley, serves as the foundational mathematical principle in FIC and is a corollary of the Banach Fixed-point Theorem [20]. It quantifies the approximation error between the fractal attractor and the target image, ensuring reliable reconstruction by bounding global discrepancies based on local collage fidelity. The assurance of a unique attractor derives from the Banach Fixed-point Theorem, which applies to complete metric spaces and contractive mappings. Given a mapping W with contractivity factor $k < 1$, such that

$$d(W(x), W(y)) \leq kd(x, y) \quad (1)$$

for all $x, y \in X$, there exists a unique fixed point A^* satisfying $W(A^*) = A^*$. Iterative application from any initial x_0 , via $x_{n+1} = W(x_n)$, converges to A^* with an error bounded by

$$d(x_n, A^*) \leq \frac{k^n}{1-k} d(x_0, x_1) \quad (2)$$

As a direct corollary, the Collage Theorem extends this to approximation: for a target image $f \in X$, if the collage error satisfies $d_h(f, W(f)) \leq \epsilon$, then the attractor A^* approximates f with

$$d_h(f, A^*) \leq \frac{\epsilon}{1-k} = \frac{d_h(f, W(f))}{1-k} \quad (3)$$

This inequality bounds the global reconstruction error based on local collage fidelity, ensuring that the self-similar attractor $A^* = \bigcup w_i(A^*)$ aligns closely with f under controlled contractivity.

In fractal image compression (FIC), the Collage Theorem enables efficient reconstruction by minimizing local approximation errors to yield compact transformation parameters. As demonstrated in Figure 1a, the reconstruction fidelity (PSNR) typically reaches a stable plateau within 8 to 10 iterations across different images. Since Figure 1b shows that the decoding time (DT) increases linearly with the iteration count, exceeding 10 iterations provides negligible quality gains while incurring unnecessary computational overhead.

Consequently, an iteration range of 8 to 10 represents the optimal balance between image quality and temporal efficiency.

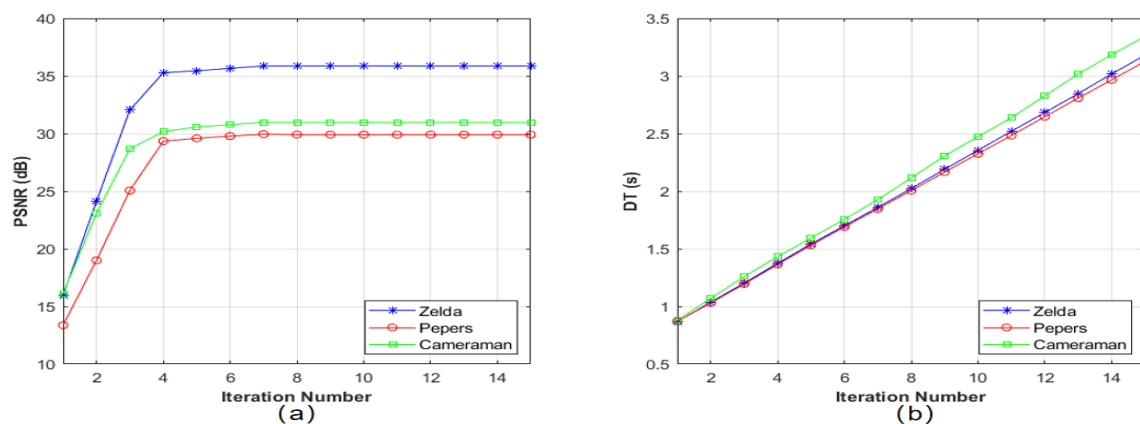


Figure 1. The impact of the iteration number on the quality of reconstructed images and decoding time cost when using the BFIC algorithm. (a) The relationship between iteration number and Peak Signal-to-Noise Ratio (PSNR). (b) The relationship between iteration number and decoding time (DT).

2.1.2. Partitioned Iterated Function System (PIFS)

The Partitioned Iterated Function System (PIFS) [2], introduced by Jacquin in 1992, extends the Iterated Function System (IFS) to automate fractal compression of natural images, as shown in Figure 2. It partitions the image into blocks to exploit local self-similarity, thereby achieving a practical implementation.



Figure 2. The self-similarity of images in the Iterated Function System (IFS) is demonstrated by the Koch curve. (a) First iteration; (b) Second iteration; (c) Third iteration; (d) Fourth iteration.

The transition from IFS to PIFS addresses the limitations of IFS, which relies on manual transformation design suitable primarily for synthetic fractals. IFS comprises a set of contractive affine mappings

$$w_i(x) = A_i x + b_i \quad (4)$$

where the spectral radius of A_i is less than 1, and the collective mapping

$$W(A) = \bigcup_{i=1}^N w_i(A) \quad (5)$$

generates attractors in a complete metric space via the Banach Fixed-point Theorem. For grayscale images, IFS extends to a three-dimensional form:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & F_s \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ F_o \end{pmatrix} \quad (6)$$

decomposing into geometric transformation g and grayscale mapping m ($w = g \circ m$), with $F_s = 0.5$ and A selected from eight symmetry transformations for efficiency. PIFS adapts this by decomposing global mappings into local block-level approximations: segmenting

the image into non-overlapping R-blocks (e.g., 8×8 pixels) and overlapping D-blocks (e.g., 16×16 pixels), where each R_j is approximated as $R_j \approx w_k(D_k)$.

$$R \approx F_s \cdot D + F_o \cdot I \quad (7)$$

The principle decomposes global self-similarity into localized matches while maintaining contractivity; the method includes (1) image partition, (2) searching for optimal matches that minimize mean squared error (MSE)

$$E(R_j, D_k) = \min \|R_j - w_k(D_k)\|^2 \quad (8)$$

This involves solving for F_s and F_o via linear regression:

$$F_s = \frac{\sum (R_j - \bar{R} \cdot I)(D_k - \bar{D} \cdot I)}{\sum (D_k - \bar{D} \cdot I)^2} \quad (9)$$

$$F_o = \bar{R} - F_s \cdot \bar{D} \quad (10)$$

$\bar{R}$ and $\bar{D}$ represent the pixel mean of R-blocks and D-blocks, respectively, and (3) storing block-specific parameters, utilizing Collage Theorem bounds to ensure reconstruction accuracy. PIFS incorporates optimization strategies, such as quadtree partitioning, texture classification, and hashing, to mitigate computational intensity during encoding while supporting high compression ratios and resolution-independent decoding.

2.1.3. Affine Transformation

Affine transformations are core spatial mappings in FIC, supporting both IFS and PIFS to model self-similar structures through scalable, rotatable, and translatable operations on image blocks. In PIFS, each transformation is defined as

$$w_k(x) = F_s(A_k x + b_k) + F_o \cdot I \quad (11)$$

where $|F_s| < 1$ ensures contractivity; A_k is selected from eight isometries for scaling, rotation, and shearing; b_k handles translation; and F_o adjusts brightness offset. For grayscale processing, intensity is incorporated as the z-coordinate to maintain dimensional consistency. This formulation minimizes MSE during block matching, aligning with the Collage Theorem to provide error-bounded approximations. The affine approach enables a compact parameter representation, replacing pixel storage with transformation coefficients, thereby facilitating efficient compression and preserving fidelity in diverse applications.

Fractal image compression relies on an affine transformation-based encoding process, which includes spatial compression, isometric transformation, and grayscale matching searches. In spatial compression, to allow a fair comparison between R-blocks and D-blocks, the D-blocks are resized to match the R-blocks' dimensions, typically via techniques such as pixel averaging or subsampling, as shown in Figure 3.

Isometric transformation is a critical step in the fractal image compression encoding process. It ensures that the fundamental geometric properties are preserved after resizing the D-block, thus maintaining the self-similarity features required by FIC. To further enhance image reconstruction quality, the virtual codebook is expanded by applying eight equally spaced transformations t_i ($i = 0, 1, \dots, 7$), including rotations of 0° , 90° , 180° , and 270° , as well as flips horizontally, vertically, diagonally, and anti-diagonally.

The selection of these eight transformations is grounded in the Dihedral Group D_4 [2], which represents an optimal configuration for image blocks matching. By comprehensively encompassing all isometries of a square region, this set eliminates interpolation artifacts

while providing sufficient spatial diversity to enhance both matching accuracy and reconstructed image quality (PSNR). Crucially, empirical evidence indicates that expanding the transformation set beyond these eight yields negligible PSNR gains while substantially escalating computational complexity and encoding time. Therefore, this eight-transformation framework achieves a Pareto-optimal balance, maximizing reconstruction fidelity while maintaining high computational efficiency.

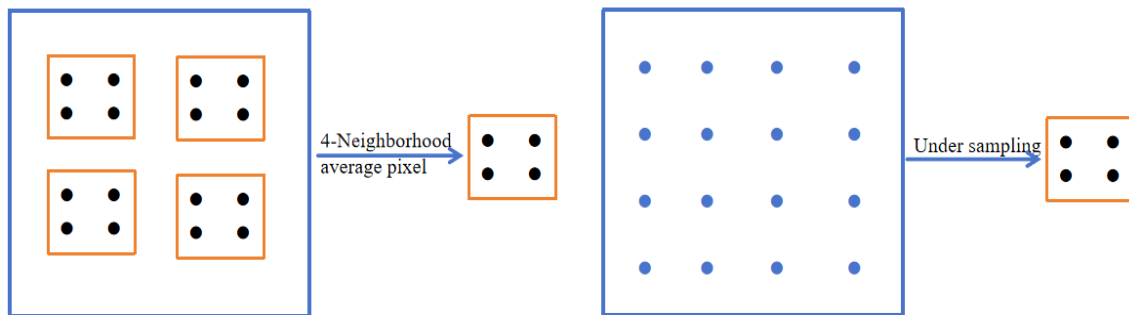


Figure 3. Airspace contraction method of D-blocks.

As detailed in Table 1, which summarizes recent advances in isometric transformations, each operation is encoded via a three-bit descriptor. This compact representation minimizes redundancy and facilitates parallel execution through linear matrices and translation vectors. Consistent with the Collage Theorem, this design ensures iterative convergence during decoding to reconstruct a high-fidelity approximation of the original image. While traditional fractal compression (e.g., Jacquin’s method) is often hindered by the computational bottleneck of exhaustive searches, our approach leverages optimizations, such as classifying search based on block size, to reduce complexity. Ultimately, the D_4 framework strikes an optimal balance between compression ratio and encoding efficiency by exploiting inherent image redundancies without inflating parameter storage requirements.

Table 1. Eight spatial transformation matrices.

t_i ($i = 0, 1, \dots, 7$)	Transformation	Matrix A
t_0	Identity transformation	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
t_1	Rotate 90 degrees clockwise	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
t_2	Rotate 180 degrees clockwise	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
t_3	Rotate 270 degrees clockwise	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
t_4	Symmetric reflection on x	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
t_5	Symmetric reflection on $y = x$	$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$
t_6	Symmetric reflection on y	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
t_7	Symmetric reflection on $y = -x$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

2.2. The Non-Uniform Partition

The non-uniform partition (NUP) methods represent a significant advancement in digital image processing, allowing for adaptive block segmentation based on local textures and features. Traditional uniform partition techniques, such as those used in JPEG, divide images into fixed 8×8 blocks, often resulting in inefficiencies like block artifacts, especially at low bit rates. Non-uniform partition methods address this issue by dividing images into regions of different sizes and shapes, guided by local signal characteristics. These methods have developed from early non-uniform signal transforms and are widely used in image reconstruction, watermarking, compression, denoising, steganography, and encryption.

The concept of non-uniform partition in signal processing emerged in response to the limitations of uniform sampling and division techniques. Early foundations can be traced back to the Non-Uniform Discrete Fourier Transform (NDFT) [21], proposed in the late 20th century, which introduced non-equidistant sampling in the frequency domain to handle non-stationary signals better. This was followed by the Warped Discrete Fourier Transform (WDFT) [22], which warps the frequency axis to enable adaptive analysis. A pivotal advancement occurred with the Wedgelet Decomposition [23] in the early 2000s. This method divides image sub-regions into wedge-shaped areas using lines to approximate piecewise constant functions, effectively addressing edge distortions in wavelet-based reconstructions. Building on this, KinTak et al. [24] introduced the Non-Uniform Rectangular Partition (NURP) for one-dimensional signals, extending it to images by adaptively partitioning based on least squares approximation with polynomials. This method partitions signals into regions of varying lengths, using one-variable polynomials for approximation, and demonstrates superior denoising effects. Subsequent developments include the Non-Uniform Triangular Partition (NUTP) [25], which utilizes triangular meshes to capture curved boundaries more effectively and reduce redundant coding. Zhang et al. [26] proposed the Image Texture Adaptive Non-Uniform Rectangular Partition (ITANRP), integrating U-system orthogonal bases and all-phase digital filters to enhance the coding of blocks with varying texture complexity. More recent innovations, such as those by Zhao et al. [27], which incorporate Linear Canonical Transform (LCT) into adaptive partition, and Zhao et al. [28], which use Gaussian functions for approximation, further expand the scope to include signal denoising and watermarking. Liu and KinTak [29] applied NURP to multi-focus image fusion with morphological operations, while KinTak et al. [30] focused on denoising via interpolation-enhanced partitioning. Chen et al. [31] proposed the Scale-Time Channel Cross-Coder (STCCE), a framework that optimizes the trade-off between prediction accuracy and computational efficiency through adaptive non-uniform representation, thereby enhancing model performance and processing throughput. These evolutions highlight a shift from uniform to adaptive, texture-aware methods, driven by needs in multimedia and the era of significant data.

At its core, non-uniform partition relies on least squares approximation to fit polynomials or other functions to local image regions, adaptively subdividing until a predefined error threshold is met. Suppose an image G as a two-dimensional function over a domain, with grayscale values $Q(x_j, y_j)$ at pixels (x_j, y_j) . The process starts with an initial partition (e.g., dividing into four quadrants for rectangular methods). For each sub-region G_m , a polynomial $f(x, y)$

$$f(x, y) = ax + by + cxy + d \quad (12)$$

is fitted using least squares minimization: $\min_{a,b,c,d} \sum_{(x_j, y_j) \in G_m} [Q(x_j, y_j) - f(x_j, y_j)]^2$. The mean square error (MSE) is then computed:

$$\text{MSE} = \frac{1}{N} \sum_{(x_j, y_j) \in G_m} [Q(x_j, y_j) - f(x_j, y_j)]^2 \quad (13)$$

where N is the number of pixels. If MSE exceeds a control threshold ϵ , the region is subdivided (e.g., into four sub-rectangles or triangles) recursively, following self-similar rules. Regions are encoded using a quaternary tree structure for non-uniform partition. Similarly, the sub-regions obtained by non-uniform rectangular partition, according to the principle of the quadtree, assign numbers $m = m_k 4^k + m_{k-1} 4^{k-1} + \dots + m_1 4^1 + m_0 4^0$.

$$m = (m_k m_{k-1} \dots m_1 m_0)_4, m_j \in \{0, 1, 2, 3\}, j = 0, 1, 2, \dots \quad (14)$$

The positions and sizes of the required sub-regions can be quickly determined from the numbers, facilitating least-squares approximation calculations and image reconstruction. As shown in Figure 4, the block $S = 313$, which can be traced from $G_3 \rightarrow G_{31} \rightarrow G_{313}$ to its specific location.

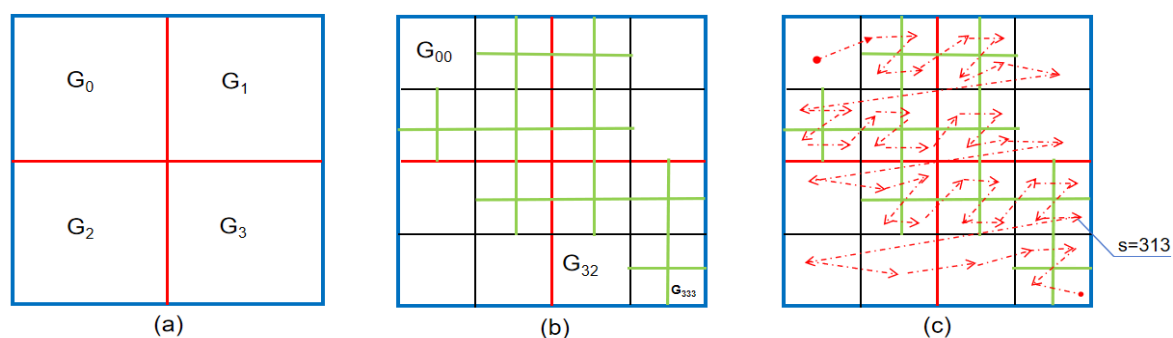


Figure 4. Process of non-uniform rectangular partition. (a) Initial rectangular partition region. (b) A non-uniform rectangular partition is performed based on (a). (c) Block tracing based on (b) and G_{333} represents reaching the control threshold stop partition.

Non-uniform partition methods have evolved from early frequency-domain transforms to advanced adaptive algorithms, primarily based on principles of least-squares approximation. These methods excel at capturing local image textures and discontinuities, as illustrated in Figure 5. They are widely used for denoising, multi-focus image fusion, and security tasks (e.g., robust watermarking and steganography via textured region embedding). Their smooth integration with image compression techniques, such as hybrid frameworks like ITANRP-U-system or APUBT3-NUP [26], offers significant benefits, including higher compression ratios, better Peak Signal-to-Noise Ratios at low bit rates, fewer blocking artifacts with variable block sizes, and increased efficiency for non-stationary signals—paving the way for advanced multimedia transmission and storage solutions.

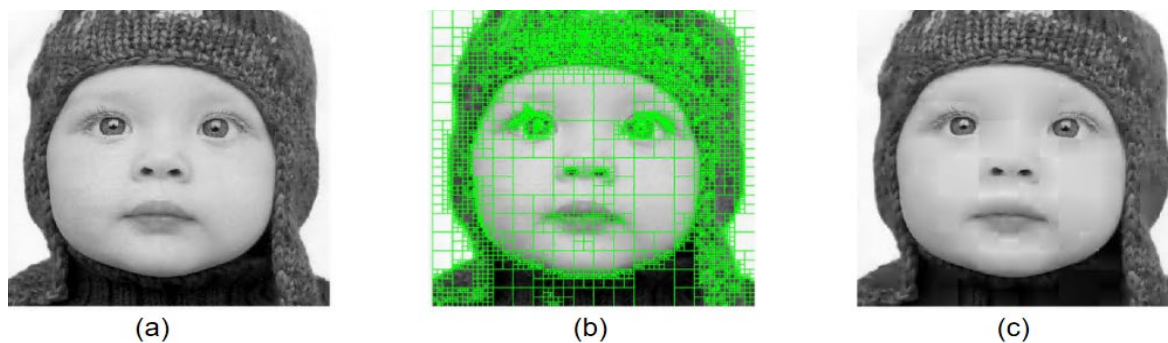


Figure 5. Image reconstruction using non-uniform rectangular partitioning: (a) original image, (b) partition overlay on the original image, and (c) reconstructed image.

3. Methodology and Scheme Design

The fractal image compression method exploits the self-similarity inherent in natural images by mapping D-blocks to R-blocks through affine transformations, thereby achieving a high CR with a compact set of parameters in the iterated function system (IFS). However, its underlying assumption of uniform block partitioning introduces several limitations. In highly textured regions, fixed-size blocks fail to capture fine-scale similarities, resulting in large matching residuals and a lower PSNR. Conversely, in the smooth areas, the rigid uniform partitioning produces redundant parameters, unnecessarily consumes coding bits, and ultimately constrains the achievable compression ratio. In addition, a global exhaustive search unfolds on a uniform grid, with a computational complexity of $O(n^4)$, resulting in excessively high ET, often up to tens of seconds to process a 512×512 image, making it challenging to support real-time applications. These problems stem from BFIC's neglect of image non-stationarity, which is characterized by uneven texture distributions in natural images. The inability of uniform grids to adapt to a one-size-fits-all approach results in a dual loss of quality and efficiency.

Non-uniform partitioning (NUP) as a content-adaptive strategy that leverages least-squares approximation and self-similar recursion to dynamically adjust block dimensions based on local texture and structural characteristics, thereby surmounting the rigid constraints of traditional quadtree-based “equal-quadrant” segmentation. While conventional quadtree methods provide a degree of adaptability, they are fundamentally limited by the computational interweaving of partitioning and matching that precludes efficient parallelization, a restricted focus on local features that neglects broader global correlations, and an unoptimized, redundant domain block pool that compromises compression efficiency. The proposed FFICNUP framework addresses these deficiencies by deeply integrating NUP into the fractal encoding pipeline—a process requiring a comprehensive architectural redesign involving texture-aware similarity metrics, optimized bitstream syntax, and enforced Iterated Function System (IFS) convergence conditions—to effectively balance high-precision matching in complex regions with redundancy reduction in smooth areas. By embedding NUP into the core of the partitioning and matching logic, FFICNUP systematically overcomes traditional quadtree bottlenecks, ensuring superior detail preservation and compression accuracy through an optimized, hierarchical tree structure.

3.1. Adaptive Non-Uniform Block Partition

To address the limitations of BFIC, which always relies on a fixed uniform partition scheme, the proposed FFICNUP framework adopts an adaptive non-uniform partition strategy that accounts for image spatial heterogeneity. Specifically, block sizes for both R-blocks and D-blocks are dynamically adjusted based on local texture. Smaller R-blocks are assigned to high-frequency regions (e.g., edges and intricate textures) to preserve fine structural details. In comparison, larger R-blocks are used in smooth or homogeneous areas to improve coding efficiency. Inspired by wedgelet subdivision [23] and non-uniform rectangular partition, this method enables more precise block matching while preserving fractal self-similarity. Furthermore, by categorizing candidate matches by block size, the search space is effectively limited, significantly reducing computational complexity and encoding time. As a result, FFICNUP achieves notable reductions in both encoding and decoding time while enhancing PSNR without sacrificing compression efficiency.

In contrast to the adaptive non-uniform partition of the R-block, the D-block segmentation employs a hybrid uniform-to-adaptive approach, as illustrated in Figure 6. The process begins with a regular, non-overlapping tiling of the source image into large rectangular D-blocks (typically 64×64 or 32×32 pixels, selected based on the image resolution and desired reconstruction quality). Each initial D-block is then fitted with a least-squares bilinear

polynomial to its intensity surface. If the resulting approximation error exceeds a set planarity tolerance—indicating significant deviation from a planar intensity distribution—the D-block is divided further into four equal sub-blocks. This error-based criterion ensures subdivision occurs only in areas with notable curvature or high-frequency details, while smooth or low-texture regions retain larger blocks.

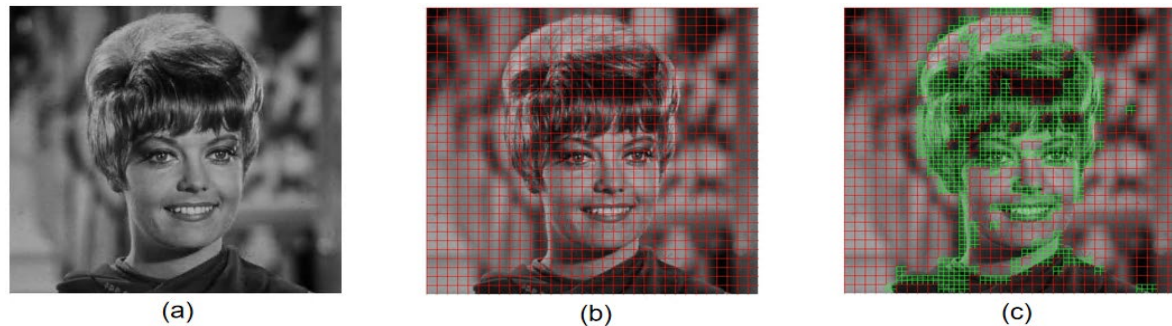


Figure 6. Non-uniform partition scheme of D-blocks. (a) Original image of Zelda. (b) Zelda divided into 32×32 D-blocks with no overlap. (c) Non-uniform partition performed on each D-block.

The resulting non-uniform rectangular partition-based decomposition creates a variable-size D-block pool, with individual blocks exhibiting significantly higher geometric and photometric similarity to their corresponding R-blocks. As a result, this method effectively captures fine-scale self-similarities often missed by uniform partitioning schemes. Due to the substantial reduction in local approximation errors, the proposed adaptive D-block segmentation provides noticeably better reconstruction quality, with improved edge preservation, better texture recovery, and reduced blocking artifacts and over-smoothing that are common in traditional fixed-partition FIC.

3.2. Task-Serial Workflow with Data-Parallel Vectorization (TSW-DPV)

Conventional fractal image compression algorithms generally rely on deeply nested loops to process each R-block sequentially, followed by a comprehensive search over the entire D-block pool to find the best-matching D-block. For each candidate D-block, various affine transformations (including rotations, reflections, and contrast scaling) are explicitly applied, and optimal transformation parameters are estimated by minimizing the MSE. Although conceptually simple, this method results in significant computational overhead due to redundant pixel-wise operations, repeated memory accesses, and frequent function calls. These inefficiencies become more pronounced in high-resolution settings, where the exponential increase in block candidates results in extremely long encoding times.

Building upon the adaptive non-uniform partitioning scheme established in the preceding stage, we re-engineer the computationally intensive core operations—specifically affine transformation estimation and block matching—into a Task-Serial Workflow with Data-Parallel Vectorization (TSW-DPV). To overcome the prohibitive computational burden inherent in conventional FIC, the proposed TSW-DPV framework achieves a substantial reduction in execution latency through two synergistic strategies. First, we implement a size-constrained searching strategy: rather than a global search, each R-block is matched only against a D-block pool of the corresponding scale within a localized neighborhood. By restricting the search to a constant number of candidates. Second, TSW-DPV harnesses fine-grained data-level parallelism by recasting loop-heavy operations (e.g., downsampling, geometric transformations, and similarity scoring) as batched matrix and vector operations. By leveraging heavily optimized linear algebra primitives, our approach eliminates redundant per-block iterations and substantially reduces the constant factor of the encoding time.

As illustrated in Figure 7, the proposed FFICNUP scheme implements a Task-Serial Workflow with Data-Parallel Vectorization and enhances encoding efficiency through the following key stages:

- a. Average Pooling (downsampling): Each D-block is downsampled by calculating the mean intensity over non-overlapping 2×2 local neighborhoods. Importantly, this operation is fully vectorized: all blocks of the same scale are batched into a higher-dimensional tensor, and spatial averaging is performed through successive array reshaping and axis-aligned mean reductions—avoiding per-block loops. This approach maintains photometric fidelity while halving spatial resolution, thus enabling efficient multi-scale matching with reduced computational and memory demands.
- b. Geometric Transformation: The eight isometric transformations of D-blocks—namely identity, horizontal flips, vertical flips, 90° -incremental rotations, and transposition—are encoded as lightweight, vectorization-ready mapping functions that convert a 2D block directly into a column vector. These functions are designed to operate seamlessly within batched workflows, enabling simultaneous application across all D-blocks of a given scale without iterative per-instance calls.
- c. Block Similarity Calculation and Matching: Inner products, squared norms, and global sums between the R-block and all transformed D-block candidates are computed in batch using matrix operations. Optimal scaling F_s and offset F_o parameters are obtained analytically in closed form for all candidates simultaneously, and the best match is selected by minimizing the resulting MSE vector—completely removing per-candidate iterative optimization.

Compared with conventional FIC, TSW-DPV provides three key benefits: First, it delivers a speedup of one to two orders of magnitude, especially for large-scale images, by removing deeply nested loops and reducing per-call function overhead. Second, it improves numerical stability by consolidating iterative operations into a small set of highly optimized matrix calculations, thus reducing cumulative floating-point rounding errors. Third, its modular design allows easy integration with hardware accelerators (e.g., GPUs) and supports scalable deployment in real-time systems. Importantly, these improvements do not come at the expense of encoding accuracy, as shown by various benchmarks, including remote sensing imagery and the Kodak dataset.

During the decoding phase of fractal image compression, the proposed Task-Serial Workflow with Data-Parallel Vectorization (TSW-DPV) framework demonstrates significant advantages by fully utilizing vectorized computation. As shown in the flowchart in Figure 8, traditional decoding repeatedly performs D-block extraction, downsampling, geometric transformations, affine mapping, and R-block updates via deeply nested loops, resulting in considerable overhead. In contrast, TSW-DPV systematically reformulates these operations as batched matrix–vector operations, enabling the simultaneous processing of all blocks at the data level.

The TSW-DPV framework leverages comprehensive vectorization to batch-process D-blocks, encompassing downsampling, geometric transformations, and affine mapping $R = F_s \cdot D + F_o$ via broadcast matrix-vector operations. By streamlining these processes with advanced indexing for block write-back, the framework eliminates scalar per-pixel updates and mitigates random memory access penalties, thereby substantially reducing per-iteration execution latency. This data-parallel strategy strictly preserves task-level sequential dependencies while maximizing data-level parallelism, ensuring bit-exact equivalence and numerical consistency with traditional scalar implementations. Consequently, the reconstruction converges to the identical fixed-point attractor as conventional methods without fidelity degradation. While dynamic stopping criteria are theoretically sound,

experimental evidence confirms that the Peak Signal-to-Noise Ratio (PSNR) typically stabilizes within 8–10 iterations.

In the high-throughput parallel environment of TSW-DPV, the overhead of calculating global residual errors per iteration would offset net acceleration gains. Therefore, adopting a fixed eight-iteration limit (as illustrated in Figure 1) serves as a robust heuristic that maximizes reconstruction fidelity while minimizing computational overhead. This design strikes an optimal balance between accuracy and efficiency, delivering the deterministic execution timing critical for resource-constrained embedded systems.

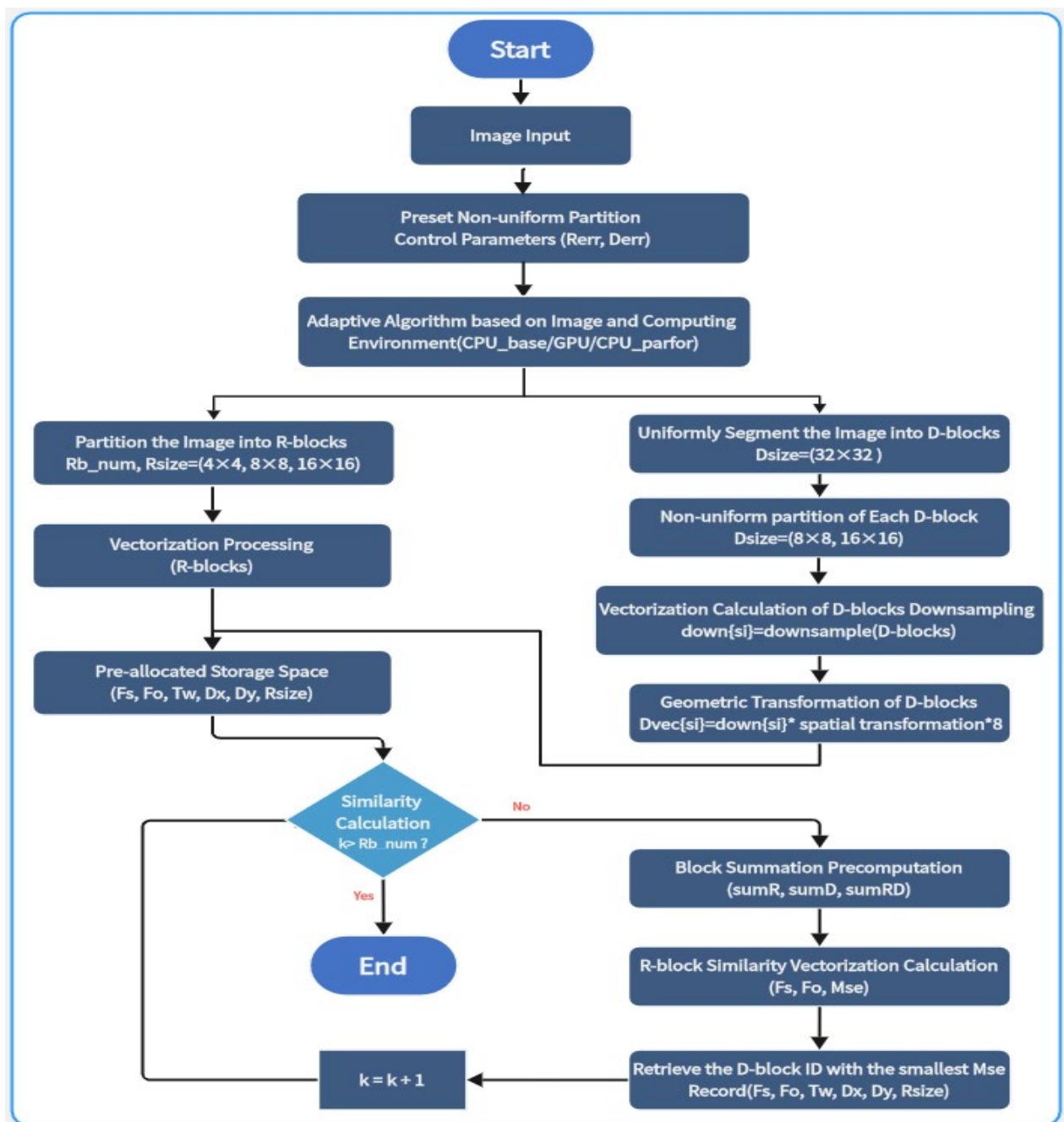


Figure 7. The encoding process of FFICNUP.

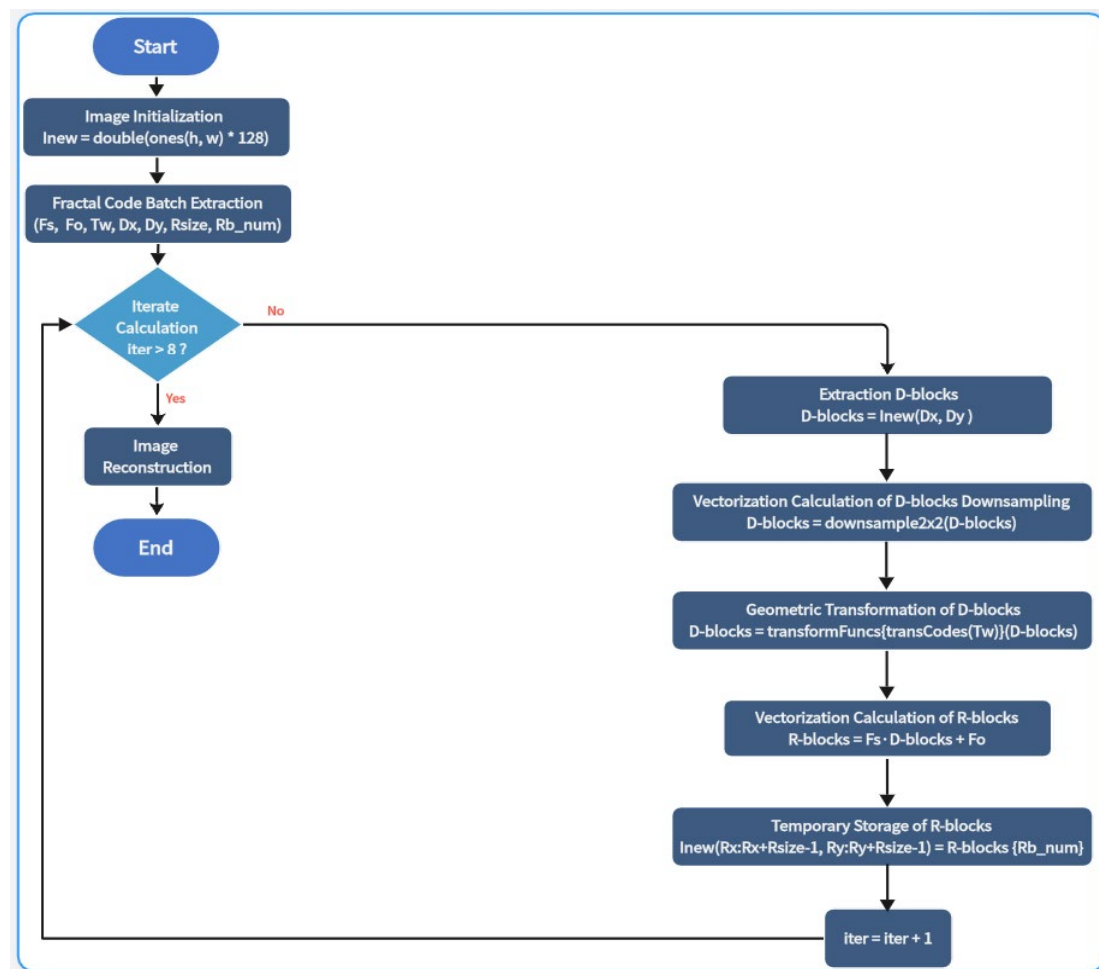


Figure 8. The decoding process of FFICNUP.

3.3. Adaptive Hybrid Fractal Compression

To alleviate the prohibitive computational burden of encoding and decoding in fractal image compression (FIC) and to better adapt to diverse real-world deployment environments, as shown in Figure 9, prior studies have examined various optimization methods, including CPU-based multithreading and GPU-accelerated parallel computing. CPU multithreading improves performance on general-purpose hardware but is often limited by thread synchronization overhead. GPU acceleration offers high throughput for large-scale images, but it requires specialized hardware and complex memory management. In contrast, this paper presents Task-Serial Workflow with Data-Parallel Vectorization (TSW-DPV), a new approach that leverages data-level parallelism through SIMD (Single Instruction, Multiple Data) instructions.

Furthermore, to overcome the limited flexibility of traditional fractal compression algorithms across various image sizes and hardware platforms, we introduce the FFICNUP. This hybrid fractal compression framework adaptively chooses or combines specialized encoding methods based on image sizes and hardware capabilities (including CPU cache hierarchy, SIMD support, GPU presence, and memory bandwidth). This content- and platform-aware approach allows for joint optimization of image reconstruction quality and processing speed, ensuring reliable performance in real-world applications. By integrating algorithmic adaptation with hardware awareness, FFICNUP offers stable and efficient compression across a wide range of systems, from resource-limited embedded devices to high-performance workstations.

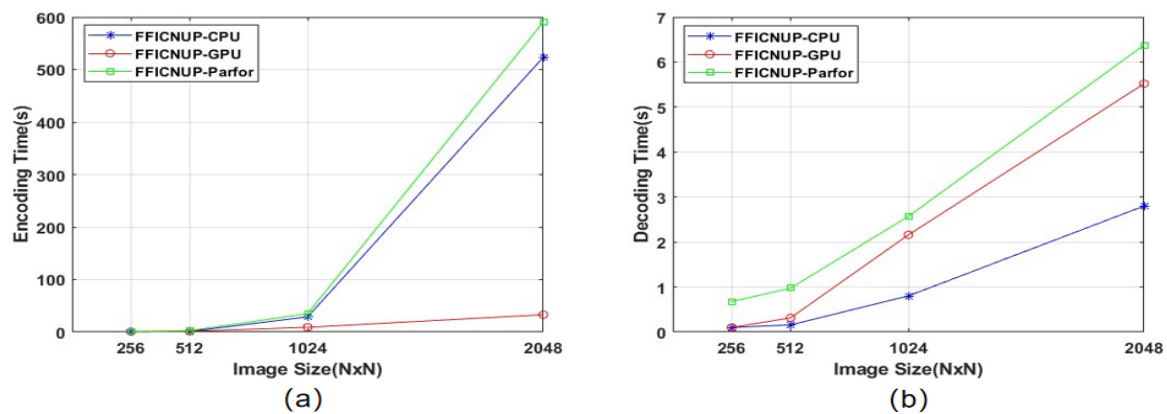


Figure 9. The impact of FFICNUP across different environments on the encoding and decoding times of the Plane image of various sizes; (a) The relationship between image size and encoding time (ET); (b) The relationship between image size and decoding time (DT).

Using the TSW-DPV framework introduced in this article, we assessed the encoding and decoding performance on the Plane test image at multiple resolutions across three typical computing environments. The PSNR achieved by the BFIC method, using uniform 8×8 R-block segmentation, is used as the quality benchmark. Under this fixed PSNR constraint, we present the CR achieved by the proposed FFICNUP hybrid algorithm, along with its respective encoding and decoding times in each environment.

Table 2 presents the three parallel computing environments of FFICNUP, showing the encoding and decoding times for fractal compression. FFICNUP_CPU employs CPU-based multithreading with vectorized operations; FFICNUP_GPU leverages the GPU's large-scale parallelism; and FFICNUP_Parfor uses MATLAB's parfor construct for high-level parallel prototyping.

Table 2. Comparison of encoding and decoding times for different-sized Plane images across different conditions.

Plane			Encoding Time (s)			Decoding Time (s)		
Image Size	PSNR	CR	FFICN UP_CPU	FFICN UP_GPU	FFICN PU_Parfor	FFICN UP_CPU	FFICN UP_GPU	FFICN PU_Parfor
256×256	21.27	27.27	0.23	0.57	1.14	0.10	0.83	0.67
512×512	24.54	31.21	1.19	1.54	2.63	0.15	1.12	0.97
1024×1024	36.44	18.61	28.90	8.97	35.13	0.80	2.16	2.57
2048×2048	45.39	17.27	523.32	32.99	590.31	2.80	5.52	6.37

As illustrated in Table 2, FFICNUP significantly reduces encoding time by approximately an order of magnitude compared to BFIC while maintaining competitive reconstruction fidelity as evidenced by comparable PSNR and CR metrics. Additionally, the execution time of FFICNUP varies across computing environments for the Plane test image, with resolutions ranging from 256×256 to 2048×2048 . Based on these findings, we have designed a hybrid automatic scheduling strategy for FFICNUP, prioritized as follows (in descending order of priority):

- Large images with sufficient GPU memory ($\geq 1024 \times 1024$): For image resolutions of 1024×1024 and higher, provided at least 16 GB of GPU memory is available, the system favors the FFICNUP_GPU vectorized encoding method. This method dramatically decreases the encoding time for a 2048×2048 image from about 523 s on CPU to roughly 33 s, achieving a speedup of approximately 16 times.

- b. Small images ($<1024 \times 1024$): Small images involve limited data size and lower computational demands. In such cases, the fixed overhead of transferring data from system memory to GPU memory and launching GPU kernels outweighs the benefits of parallel acceleration. Therefore, the system consistently uses the FFICNUP_CPU vectorized encoding method. Compared to the GPU path, this approach delivers $2\text{--}5\times$ better performance on 256×256 and 512×512 images.
- c. Fallback when GPU is unavailable or has insufficient memory: If there is no compatible GPU or the available GPU memory is less than 16 GB, the algorithm automatically switches to the FFICNUP_CPU computation path to ensure efficient encoding.

In summary, the main advantage of the non-uniform partition is its content-adaptive nature. By dividing the image into blocks of varying sizes (and optionally shapes), it assigns finer blocks to regions with complex textures or irregular structures while using larger blocks in smooth areas. This approach effectively captures local variations in detail, significantly improving both reconstruction quality and overall compression efficiency. During encoding, the non-uniform rectangular partition strategy divides R-blocks and D-blocks, allowing more precise matching of different image content types. This is especially evident in the handling of feature vectors from R-blocks compared to those from D-blocks, as measured by distance metrics and least-squares approximation. Such a method not only aids in finding the best-matching D-block and reducing redundancy but also potentially shortens encoding time and improves the quality of reconstructed images. By combining fractal image compression with non-uniform partition, the proposed approach strikes an excellent balance between compression ratio and reconstruction quality, delivering superior performance and broader applicability in practical scenarios.

4. Simulation Experiment and Result Analysis

A significant limitation of BFIC is its prohibitive computational intensity, which results in excessively prolonged encoding durations that hinder real-time implementation. To address this critical limitation, we propose fast fractal image compression using non-uniform partition (FFICNUP), which dramatically reduces both encoding and decoding times while simultaneously improving CR and PSNR. This section provides a comprehensive experimental evaluation of FFICNUP against BFIC and other representative fractal compression techniques. The following subsections describe the experimental setup, evaluation metrics, and detailed comparative results, demonstrating FFICNUP's superior speedup, reconstruction quality, and robustness across a wide range of image resolutions and hardware platforms.

4.1. Experimental Conditions and Key Parameters

4.1.1. Experimental Conditions and Datasets

All CPU-based experiments were conducted in MATLAB R2020a on a system equipped with an Intel Core i5-9500 CPU @3.00 GHz, running Windows 10 Professional 64-bit. GPU-accelerated experiments used the same MATLAB version and CPU setup but were performed on a Linux platform with an NVIDIA Tesla T4 GPU and Ubuntu 20.04.1 LTS. The proposed FFICNUP algorithm was fully implemented in MATLAB using the Parallel Computing Toolbox and the GpuArray interface to offload computationally intensive tasks—such as batched matrix multiplication, vectorized block transformations, and similarity evaluations—to the GPU. No custom CUDA kernels were created; instead, the implementation depends on MATLAB's built-in, highly optimized GPU-enabled linear algebra routines.

The experimental images employed in this work include classical grayscale test images and remote sensing images. Specifically, Cameraman, Zelda, Peppers, and Plane are

obtained from the USC SIPI Image Database [32], a public benchmark dataset widely used for image processing research. Furthermore, the remote sensing images (RSI343, RSI3515) are selected from the UC Merced Land Use Dataset [32], and additional test images (Kodim21, Kodim24) are sourced from the Kodak Dataset [33]. All images are tested at three resolutions: 512×512 , 1024×1024 , and 2048×2048 pixels. This heterogeneous test suite ensures a robust evaluation of the algorithm's scalability, efficiency, and reconstruction quality over diverse image content and sizes.

4.1.2. Control Parameters of Non-Uniform Blocks Partition

In the proposed FFICNUP algorithm, the partition control threshold (R_{err}/D_{err}) serves as the decisive parameter governing the granularity and distribution of R-block and D-block segmentation. This threshold fundamentally dictates key performance metrics, including Peak Signal-to-Noise Ratio encoding time (PSNR), encoding time (ET), and Compression Ratio (CR). As illustrated in Figure 10, reducing the R_{err} threshold facilitates a more refined decomposition into a larger number of R-blocks, thereby enhancing the fidelity of the reconstructed image. However, this gain in quality inherently increases the volume of transformation parameters, leading to a diminished compression ratio and prolonged encoding time—a classic trade-off between reconstruction accuracy and computational efficiency. Consequently, the judicious configuration of R_{err} is paramount for achieving a dynamic equilibrium between compression performance and visual quality.

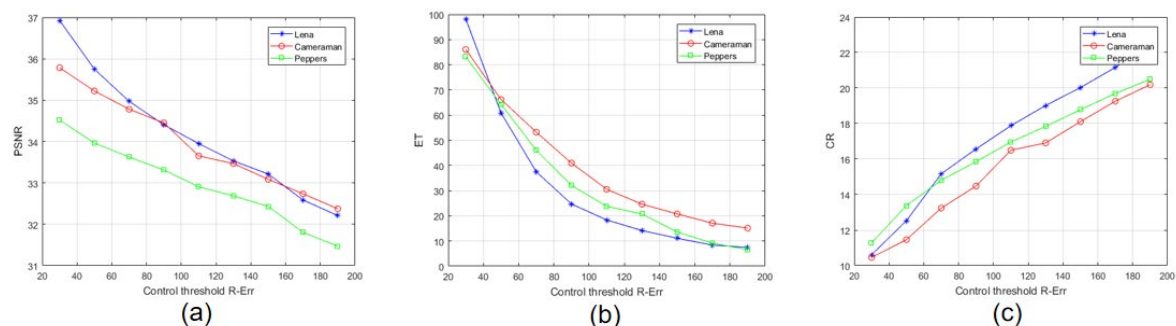


Figure 10. The non-uniform partition control threshold parameter R_{err} exerts an influence on image reconstruction quality (PSNR), compression ratio (CR), and encoding time (ET). (a) Correlation between R_{err} and PSNR; (b) Correlation between R_{err} and ET; (c) Correlation between R_{err} and CR.

To address the challenges of R_{err} calculation in resource-constrained scenarios, where computational power is limited or standard deviation is difficult to obtain, this paper proposes a streamlined estimation model. By leveraging two fundamental statistical features, namely mean intensity (Me) and mean gradient (Mg), we construct a linear regression framework for R_{err} estimation. Experimental validation demonstrates a robust linear correlation between R_{err} , Me , and Mg , confirming the efficacy of the simplified model. This approach significantly reduces computational overhead while maintaining high estimation accuracy, providing an efficient solution for fractal image compression in restricted environments and establishing a theoretical basis for the adaptive selection of partition control parameters.

$$R_{err} = 2.38Me + 2.95Mg - 280 \quad (15)$$

4.2. Experimental Evaluation Metrics

The evaluation of fractal image compression algorithms is typically divided into subjective and objective assessments. Subjective evaluation emphasizes human perceptual quality, particularly by identifying artifacts such as blocking effects in detailed regions.

Objective evaluation depends on measurable metrics, including CR, PSNR, Structural Similarity Index (SSIM), Multi-Scale SSIM (MS-SSIM), encoding time (ET), and decoding time (DT).

4.2.1. Subjective Evaluation

Subjective evaluation involves human observers assessing the visual quality of reconstructed images, which is crucial for capturing perceptual nuances that objective metrics may overlook. These include blocking effects (image block artifacts) in textured or high-frequency regions, where self-similar block-matching may introduce visible discontinuities. It also involves texture preservation, ensuring that fine details—such as patterns or surfaces—are retained without unnatural smoothing. Additionally, it considers edge sharpness, boundary clarity, and contour clarity to avoid blurring or ringing artifacts.

4.2.2. Objective Evaluation

Objective evaluation yields quantitative and reproducible metrics to rigorously assess algorithmic performance. The primary benchmarks include Compression Ratio (CR), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Multi-Scale SSIM (MS-SSIM), encoding time (ET), and decoding time (DT). Within this evaluative framework, CR serves as the definitive measure of compression efficiency, while PSNR and SSIM variants quantify reconstruction fidelity; concurrently, ET and DT characterize the computational latency, which is a critical determinant of system responsiveness in practical deployments.

The first objective evaluation metric is the CR, defined as the ratio of the original image data size S_o to the compressed data size S_c . A key advantage of fractal compression is its potential for high compression ratios, making CR a core metric. Ideally, algorithms should achieve ratios ranging from 10:1 to 100:1 or higher while maintaining acceptable distortion levels. As mentioned above, our proposed image compression and reconstruction structure of the fractal code is $Fs, Fo, Tw, R_size, Dx, Dy$, and its elements are quantified as five-bit, seven-bit, three-bit, two-bit, p -bit, and q -bit, where p and q represent the coordinate positions of the D-block, which will be dynamically quantized according to the number of D-block segments. Therefore, the CR of $M \times N$ image is defined as

$$CR = \frac{S_o}{S_c} = \frac{M \times N \times 8}{R_num \times (Fs + Fo + Tw + R_size + p + q)} \quad (16)$$

Here, R_num denotes the number of R-blocks. Note that under an adaptive non-uniform partition scheme, the numbers of both R-blocks and D-blocks are not fixed. The number of image blocks obtained will be dynamically adjusted based on different image textures and partition control thresholds.

Another objective evaluation metric is the PSNR of the reconstructed image. In the general calculation method, the mathematical formula for PSNR is

$$PSNR = 10 \times \log_{10} \left(\frac{255^2}{MSE} \right) \quad (17)$$

where MSE (Mean Squared Error) is defined as

$$MSE = \frac{1}{M \times N} \sum_{i=1}^M \sum_{j=1}^N (z(i, j) - \hat{z}(i, j))^2 \quad (18)$$

Here, $M \times N$ denotes the total number of pixels in the image, $z(x_j, y_j)$ denotes the grayscale value of pixel within the original image, and $\hat{z}(x_j, y_j)$ denotes the grayscale value of pixel within the reconstructed image.

The third evaluation metric is SSIM, which evaluates changes in structural information, luminance, and contrast, the mathematical formula for SSIM is

$$SSIM(z, \hat{z}) = \frac{(2\mu_z\mu_{\hat{z}} + c_1)(2\sigma_{z\hat{z}} + c_2)}{(\mu_z^2 + \mu_{\hat{z}}^2 + c_1)(\sigma_z^2 + \sigma_{\hat{z}}^2 + c_2)} \quad (19)$$

Here, μ_z , $\mu_{\hat{z}}$ denote the local means (average intensities) of the pixels in images z and $\hat{z}$, respectively, which represent the luminance components. σ_z^2 , $\sigma_{\hat{z}}^2$ denote the local variances of images z and $\hat{z}$, quantifying the contrast of the signals. c_1 , and c_2 are stabilization constants introduced to prevent the denominator from approaching zero in regions with extremely low mean or variance. Furthermore, to account for image variations across multiple scales (i.e., resolutions), we adopt Multi-Scale SSIM (MS-SSIM). By iteratively downsampling the images and applying the SSIM framework at each scale, MS-SSIM enables a more robust evaluation of image quality—particularly for images with complex textures and under varying viewing distances.

The final key evaluation metrics are ET and DT, which quantify the temporal efficiency of the algorithm during actual execution. In fractal image compression, these two metrics serve as definitive indicators of computational performance; they directly determine the practical feasibility and real-world applicability of the proposed method, particularly for time-sensitive tasks where low-latency processing is essential.

The overall performance of the FFICNUP algorithm is evaluated based on its capability to balance the CR, PSNR, SSIM, MS-SSIM and computational time. The goal is to identify a Pareto-optimal operating point that maximizes both compression efficiency and reconstruction quality while keeping encoding complexity within practical limits and reducing decoding time.

4.3. Experimental Results and Analysis

To rigorously evaluate the proposed FFICNUP, we compare it with two representative fractal image compression algorithms: the classical basic fractal image compression (BFIC), serving as the baseline, and FICANRP [10]. This recent advanced approach integrates the non-uniform partition with fractal coding optimization. Experiments are conducted on a diverse dataset encompassing standard test images and high-resolution remote sensing images with varying texture complexity and resolutions. Performance is quantitatively assessed using three key metrics: reconstruction quality (PSNR, in dB), compression ratio (CR), and computational efficiency (encoding time ET and decoding time DT, both in seconds).

4.3.1. Comprehensive Performance Evaluation

The BFIC results in this study are obtained under the following configuration: the test image is divided into non-overlapping R-blocks of size 8×8 . D-blocks are generated using an overlapping segmentation scheme with a block size of 16×16 and a step size of 8. For each R-block, a full search is conducted over the virtual codebook, which includes all transformed D-blocks, to find the best match under affine transformations. The fractal codes of F_s , F_o , T_w , D_x , and D_y are quantized as five-bit, seven-bit, three-bit, p -bit, and q -bit, respectively. The values of p and q are adaptively determined based on the D-block size and the total number. The CR of an $M \times N$ image under BFIC is defined as follows:

$$CR = \frac{M \times N \times 8}{R_num \times (F_s + F_o + T_w + p + q)} \quad (20)$$

In our comparative analysis condition, the FICANRP results are obtained using a multi-scale adaptive segmentation scheme. The input image is divided into non-overlapping

D-blocks of sizes 8×8 , 16×16 , and 32×32 pixels, with corresponding R-blocks set to half the linear dimension—that is, 4×4 , 8×8 , and 16×16 pixels. Block sizes are selected based on local texture complexity: finer partitions are applied in high-detail regions (such as edges and textured areas), while coarser partitions are used in smooth regions to enhance coding efficiency. Additionally, candidate D-blocks are grouped and searched by block size, thereby limiting the search space and significantly reducing computational complexity during the encoding process.

Table 3 demonstrates that the proposed FFICNUP consistently achieves the best reconstruction quality. On standard images such as Peppers and Zelda, it attains PSNR values of 31.43 dB and 36.31 dB, respectively, outperforming BFIC (29.73 dB, 35.86 dB) and FICANRP (31.12 dB, 36.10 dB). Notably, on high-resolution remote sensing images like RSI343, FFICNUP achieves a PSNR of 39.13 dB, significantly higher than the alternatives—highlighting its robustness in preserving fine details in complex, large-scale imagery. Regarding compression efficiency, FFICNUP either matches or slightly surpasses FICANRP and consistently outperforms BFIC. For example, on the Cameraman image, FFICNUP achieves a CR of 20.29, compared to 20.19 for FICANRP and 18.96 for BFIC. On RSI343, FFICNUP maintains a CR of 18.72, while BFIC drops to 16.51, demonstrating better adaptability to different image content.

Most importantly, FFICNUP demonstrates better computational efficiency, with significantly lower encoding time and decoding time than BFIC and FICANRP. For the 2048×2048 RSI343 image, FFICNUP reduces ET from 1.0×10^5 to 559.4 s and DT from 34.1 to 2.8 s. On the 512×512 Zelda image, ET decreases from 749.5 to 2.2 s. FFICNUP also outperforms FICANRP, which takes 13.5 s to encode Zelda and up to 32.7 s to decode RSI343.

Furthermore, the FFICNUP incorporates an adaptive hybrid execution mechanism that dynamically selects the optimal vectorization strategy—CPU-serial, multi-threaded CPU parallel (parfor), or GPU-accelerated—based on input image resolution and available computational resources. Extensive experiments were conducted on multi-scale images (512×512 to 2048×2048 pixels) across heterogeneous computing platforms (including desktop CPUs, server-grade processors, and GPUs). As shown in Table 4, FFICNUP consistently achieves an excellent trade-off between computational efficiency and reconstruction quality across all tested scenarios, demonstrating strong robustness and practical deployability.

For small-scale images (512×512), such as Zelda, Peppers, Plane, and Cameraman, the CPU-based implementation achieves the lowest ET, ranging from 2.0 to 2.2 s, outperforming the GPU (2.3–2.4 s) and MATLAB Parfor (2.2–2.7 s). This advantage results from minimal data transfer and synchronization overhead on standard hardware, where the computational load is too low to justify the cost of GPU or multi-core parallelization. For medium-resolution images (1024×1024), including Kodim21 and Kodim24, the GPU variant shows clear superiority, reducing ET to 8.9 s—significantly lower than CPU (28.6–31.2 s) and MATLAB Parfor (32.0–33.1 s). This highlights the GPU's efficiency in handling highly parallelizable tasks such as range-domain block search and affine coefficient estimation in fractal encoding. For ultra-high-resolution images (2048×2048), like remote sensing images RSI343 and RSI3515, the GPU implementation delivers dramatic acceleration performance, with an ET of only 37.2–38.8 s, far surpassing CPU (559.4–584.6 s) and MATLAB Parfor (582.3–582.5 s), achieving a 15–16 $\times$ speedup and confirming its scalability for large-scale data processing. Moreover, leveraging the FFICNUP_GPU framework, the encoding time for the RSI343 image was reduced by up to 2631.5 $\times$ compared to the BFIC method.

Table 3. Performance comparison of test images under different algorithms.

Image	BFIC						FICANRP [10]						FFICNUP					
	PSNR	CR	ET (s)	DT (s)	SSIM	MS-SSIM	PSNR	CR	ET (s)	DT (s)	SSIM	MS-SSIM	PSNR	CR	ET (s)	DT (s)	SSIM	MS-SSIM
Zelda (512 × 512)	35.86	18.96	749.5	2.8	0.9110	0.9806	36.1	20.05	13.5	2.0	0.9105	0.9802	36.31	20.08	2.2	0.2	0.9153	0.9816
Peppers (512 × 512)	29.73	18.96	751.9	2.8	0.8544	0.9556	31.12	21.10	6.2	2.0	0.8560	0.9694	31.43	21.10	2.1	0.2	0.8647	0.9698
Plane (512 × 512)	24.54	18.96	762.6	2.8	0.8137	0.9402	29.35	19.94	10.5	2.0	0.8326	0.9562	29.73	19.96	2.2	0.2	0.8650	0.9702
Cameraman (512 × 512)	31.0	18.96	733.1	2.8	0.9046	0.9765	32.34	20.19	14.2	2.0	0.9102	0.9790	32.53	20.29	2.0	0.2	0.9223	0.9860
COVIDxCT (512 × 512)	28.97	18.96	746.3	2.8	0.8938	0.9341	29.29	20.01	12.3	2.0	0.8978	0.9690	29.76	20.01	2.0	0.2	0.9023	0.9763
Kodim21 (1024 × 1024)	31.06	17.66	1.2×10^4	11.4	0.9132	0.9850	32.26	18.50	112.2	8.5	0.9185	0.9867	32.45	18.50	28.6	0.7	0.9220	0.9870
Kodim24 (1024 × 1024)	29.49	17.66	1.2×10^4	11.0	0.8951	0.9810	30.17	19.08	117.3	8.0	0.9178	0.9445	30.38	19.08	31.2	0.8	0.9227	0.9558
RSI343 (2048 × 2048)	38.89	16.51	1.0×10^5	34.1	0.9454	0.9739	38.90	18.72	3.3×10^3	32.7	0.9478	0.9768	39.13	18.72	559.4	2.8	0.9517	0.9796
RSI3515 (2048 × 2048)	43.36	16.51	1.0×10^5	33.0	0.9729	0.9912	43.51	18.07	4.3×10^3	31.4	0.9792	0.9923	43.92	18.07	584.6	2.7	0.9805	0.9987

Table 4. Encoding and decoding time comparison of FFICNUP images across CPU and GPU parallel implementations.

Image	PSNR	CR	FFICNUP_CPU		FFICNUP_GPU		FFICNUP_Parfor	
			ET (s)	DT (s)	ET (s)	DT (s)	ET (s)	DT (s)
Zelda (512 × 512)	36.31	20.08	2.2	0.20	2.3	0.45	2.5	1.28
Peppers (512 × 512)	31.43	21.10	2.1	0.21	2.3	0.45	2.7	1.06
Plane (512 × 512)	29.73	19.96	2.2	0.22	2.4	0.44	2.7	0.91
Cameraman (512 × 512)	32.53	20.29	2.0	0.21	2.4	0.45	2.2	0.88
Kodim21 (1024 × 1024)	32.45	18.50	28.6	0.74	8.9	1.77	32.0	2.33
Kodim24 (1024 × 1024)	30.38	19.08	31.2	0.81	8.9	1.79	33.1	2.14
RSI343 (2048 × 2048)	39.13	18.72	559.4	2.81	37.2	5.87	582.5	6.37
RSI3515 (2048 × 2048)	43.92	18.07	584.6	2.72	38.8	5.96	582.3	6.47

It is worth noting that the proposed FFICNUP achieves state-of-the-art decoding times across various computing conditions and image resolutions, consistently delivering ultra-low latency (0.20–6.47 s). This performance stems from the TSW-DPV framework that eliminates the redundant loop-based operations inherent in traditional fractal decoding, thereby enabling unprecedented efficiency in fractal-based image reconstruction and establishing a new SOTA in decoding times within the fractal compression domain. Crucially, these gains are attained without compromising reconstruction fidelity: under all tested conditions, including CPU-based, GPU-accelerated, and parallel-for (MATLAB parfor) execution modes—the method yields identical PSNR and CR, surpassing the quality and compactness of conventional FIC.

4.3.2. Performance of the TSW-DPV

To assess the effectiveness of the proposed Task-Serial Workflow with Data-Parallel Vectorization, we compare the performance of the original BFIC and FICANUP [10] implementations with their TSW-DPV-augmented versions. As shown in Tables 5 and 6, integrating TSW-DPV consistently and significantly alleviates the computational burden for both algorithms, resulting in substantial speedups in encoding and decoding times across all tested images while preserving or improving reconstruction quality.

Table 5. Encoding time of test images with or without the TSW-DPV.

Image	Without TSW-DPV				With TSW-DPV			
	BFIC		FICANUP [10]		BFIC		FICANUP [10]	
	PSNR	ET (s)	PSNR	ET (s)	ET (s)	Speedup	ET (s)	Speedup
Zelda (512 × 512)	35.86	749.47	36.1	13.45	3.20	234.21×	1.24	10.85×

Table 5. Cont.

Image	Without TSW-DPV				With TSW-DPV			
	BFIC		FICANUP [10]		BFIC		FICANUP [10]	
	PSNR	ET (s)	PSNR	ET (s)	ET (s)	Speedup	ET (s)	Speedup
Peppers (512 × 512)	29.73	751.91	31.12	6.15	2.74	274.42×	1.36	4.52×
Plane (512 × 512)	24.54	762.56	29.35	10.51	2.91	262.05×	1.24	8.48×
Cameraman (512 × 512)	31.0	733.08	32.34	14.17	2.76	265.61×	1.34	10.57×
Kodim21 (1024 × 1024)	31.06	12,575	32.26	112.2	46.21	272.11×	13.64	8.22×
Kodim24 (1024 × 1024)	29.49	12,466	30.17	117.34	44.23	281.87×	18.02	6.51×
RSI343 (2048 × 2048)	38.89	1.04×10^5	38.90	3.30×10^3	541.77	191.98×	266.51	12.38×
RSI3515 (2048 × 2048)	43.36	1.02×10^5	43.51	4.36×10^3	561.79	181.51×	294.22	14.61×

Table 6. Decoding time of test images with or without the TSW-DPV.

Image	Without TSW-DPV		With TSW-DPV			
	BFIC	FICANUP [10]	BFIC		FICANUP [10]	
	DT (s)	DT (s)	DT (s)	Speedup	DT (s)	Speedup
Zelda (512 × 512)	2.80	2.01	0.10	28.00×	0.22	9.14×
Peppers (512 × 512)	2.83	2.03	0.09	31.44×	0.21	9.67×
Plane (512 × 512)	2.79	2.03	0.10	27.90×	0.22	9.23×
Cameraman (512 × 512)	2.78	2.06	0.09	30.89×	0.21	9.81×
Kodim21 (1024 × 1024)	11.42	8.46	0.41	27.85×	0.74	11.43×
Kodim24 (1024 × 1024)	10.97	8.05	0.39	28.13×	0.82	9.82×
RSI343 (2048 × 2048)	34.12	32.76	1.70	20.07×	2.83	11.58×
RSI3515 (2048 × 2048)	33.06	31.76	1.67	19.79×	2.98	10.66×

To ensure a rigorous and equitable comparison, the BFIC algorithm serves as the baseline for reconstruction fidelity. All subsequent evaluations were conducted under

the constraint that the Peak Signal-to-Noise Ratio (PSNR) of the proposed method is benchmarked to be equal to or greater than that of the BFIC baseline. All experiments were executed using MATLAB R2020a on a workstation equipped with an Intel Core i5-9500 CPU @ 3.00 GHz.

For 512×512 grayscale images, the encoding time of BFIC drops from 749–763 s to 2.74–3.20 s (speedup 234–274 $\times$), while FICANUP shortens from 6.15–14.17 s to 1.24–1.36 s (accelerate 4.5–10.9 $\times$). On 1024×1024 test images, BFIC encoding time decreases from 12,466–12,575 s to 44.23–46.21 s (speedup 272–282 $\times$), and FICANUP from 112.2–117.34 s to 13.64–18.02 s (accelerate 6.5–8.2 $\times$), demonstrating strong scalability for larger and more complex images. For 2048×2048 synthetic fractal images, BFIC accelerates from 1.03×10^5 to 550 s (speedup 186 $\times$) and FICANUP from 3.8×10^3 to 280 s (acceleration 13 $\times$). The reduced speedup at the 2048×2048 scale primarily stems from GPU memory capacity constraints, which necessitate batch-wise processing and introduce scheduling overhead. In contrast, a single-pass vectorized computation—feasible under sufficient memory—would complete encoding in approximately 10 s.

It is noteworthy that while the majority of the existing FIC literature prioritizes encoding optimization, the computational overhead of decoding, particularly for high-resolution applications, remains largely unaddressed. The results in Table 6 bridge this gap by providing a direct comparison between conventional serial decoding and the proposed TSW-DPV parallel architecture. This framework demonstrates superior efficiency; as evidenced in the experimental data, its integration into the decoding pipeline substantially accelerates reconstruction across various image types and resolutions. By leveraging data locality and instruction-level parallelism to enhance iterative mapping computations, the TSW-DPV method delivers consistent speedup without increasing memory overhead, a benefit that is particularly pronounced for high-resolution, texture-rich images.

Quantitatively, for 512×512 grayscale images, average decoding time decreases from 2.82 to 0.095 s for BFIC (speedup 28–31 $\times$) and from 2.04 to 0.215 s for FICANUP (speeding up 9.1–9.8 $\times$). On 1024×1024 images, BFIC's decoding time drops from 10.97–11.42 s to 0.39–0.41 s (accelerating 28 $\times$), while FICANUP decreases from 8.05–8.46 s to 0.74–0.82 s (accelerating 9.8–11.4 $\times$). For 2048×2048 synthetic fractal images, BFIC's decoding time decreases from 33.06–34.12 s to 1.67–1.70 s (speeding up 19.8–20.1 $\times$) and FICANUP's from 31.76–32.76 s to 2.83–2.98 s (accelerating 10.7–11.6 $\times$).

4.4. Visual Quality Assessment

In addition to objective metrics, perceptual evaluation is crucial for assessing the visual fidelity of reconstructed images. As demonstrated in Figures 11–19, images reconstructed using BFIC exhibit notable blocking artifacts, texture distortion, and edge blurring, particularly in texture-rich and high-frequency regions. For example, in the Plane image (Figure 11), BFIC reconstruction results in excessive smoothing on both the aircraft fuselage surface and mountain textures. Furthermore, significant text blurring and character distortion are observed in the reconstructed lettering (e.g., “U.S. AIR FORCE”). Moreover, the Peppers image (Figure 13) displays significant blocky distortions in smooth areas. Similarly, in the Cameraman image (Figure 14), BFIC fails to preserve fine facial details and introduces visible distortions around the camera lens. In the COVIDxCT image (Figure 15), the BFIC reconstruction suffers from prominent block artifacts and visible grid-like discontinuities, which significantly degrade the visual fidelity of the image and obscure subtle pathological textures.

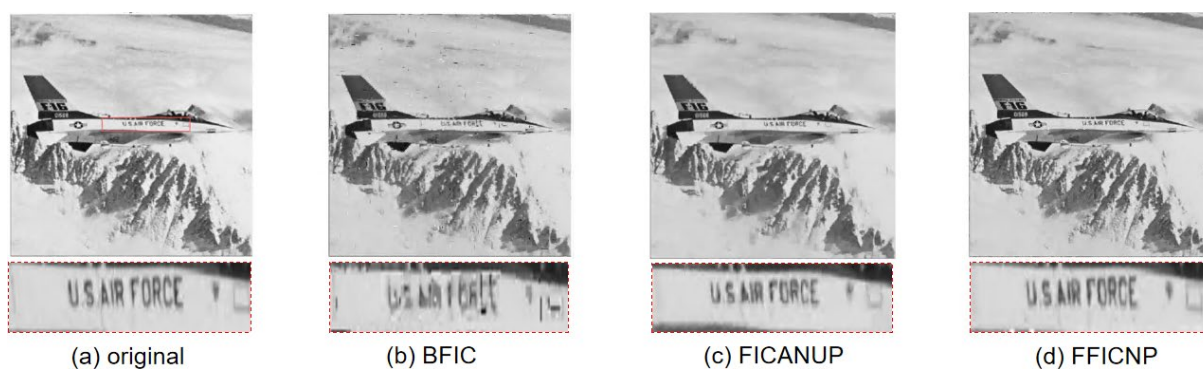


Figure 11. Comparison of reconstruction results for the Plane image using different algorithms.

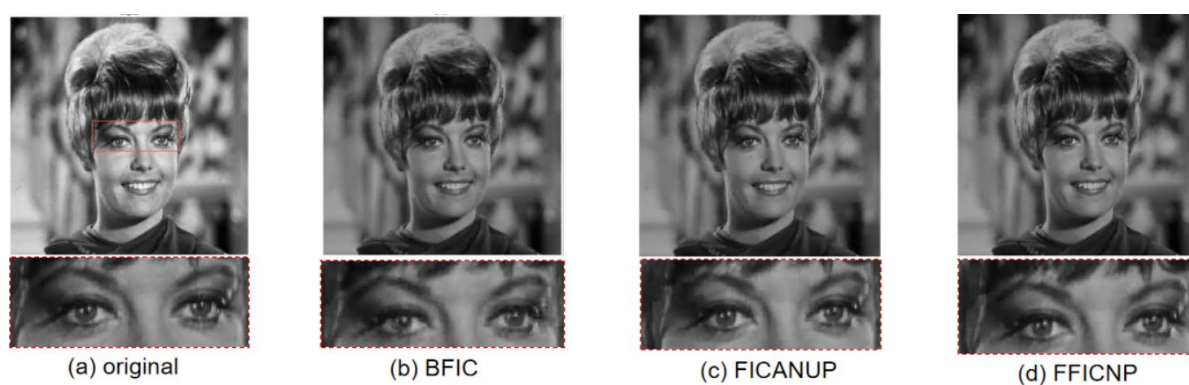


Figure 12. Comparison of reconstruction results for the Zelda image using different algorithms.

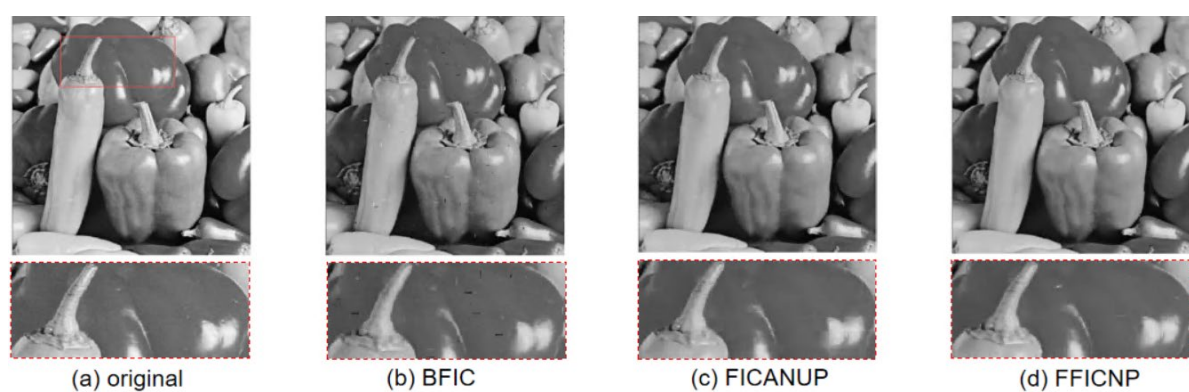


Figure 13. Comparison of reconstruction results for the Peppers image using different algorithms.

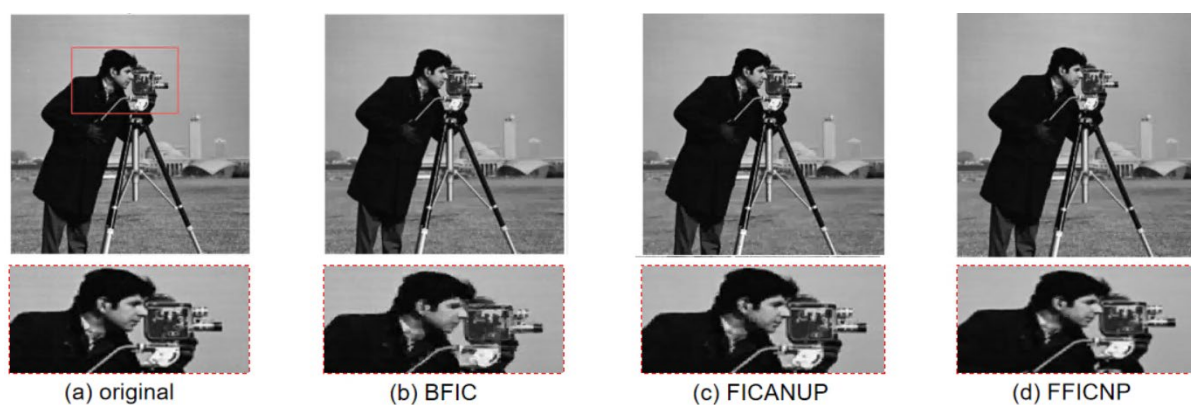


Figure 14. Comparison of reconstruction results for the Cameraman image using different algorithms.

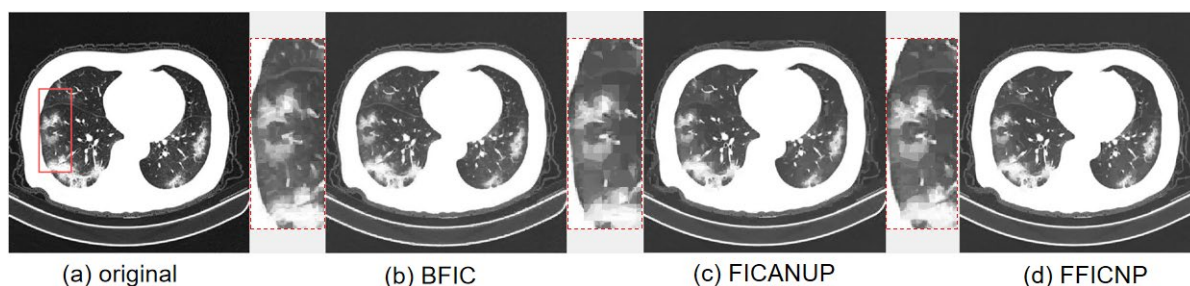


Figure 15. Comparison of reconstruction results for the COVIDxCT image using different algorithms.



Figure 16. Comparison of reconstruction results for the Kodim21 image using different algorithms.

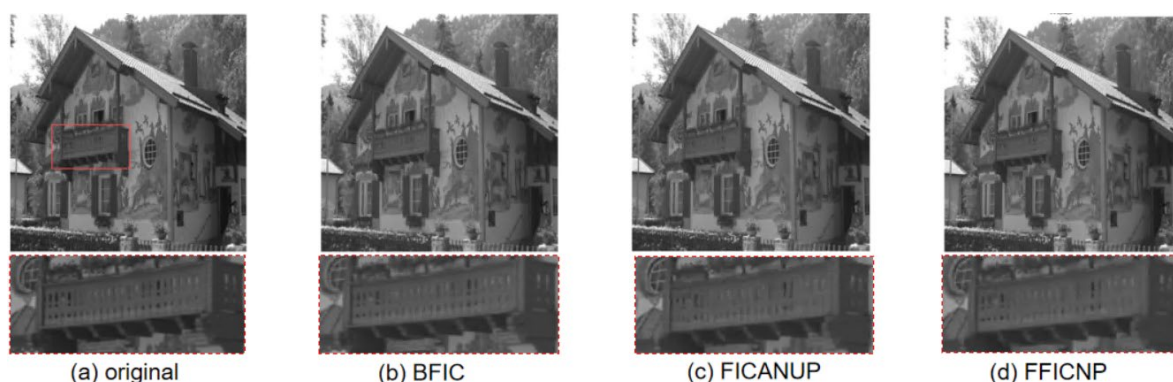


Figure 17. Comparison of reconstruction results for the Kodim24 image using different algorithms.

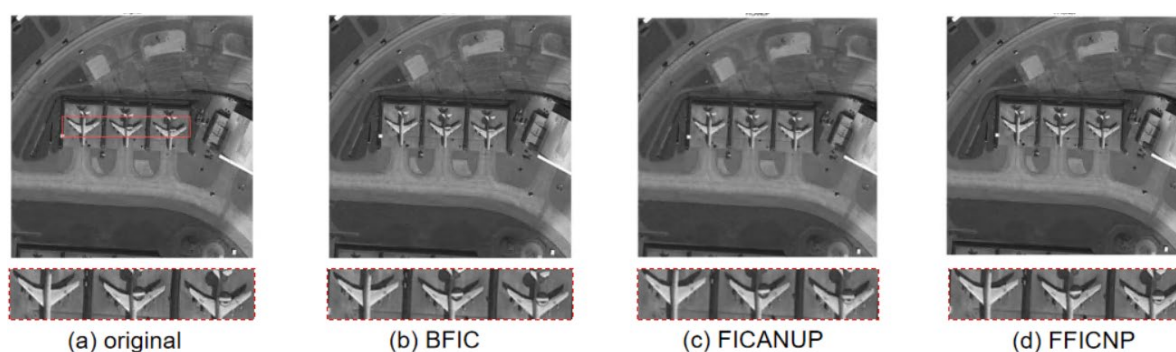


Figure 18. Comparison of reconstruction results for the RSI343 image using different algorithms.

The FICANUP method improves upon BFIC by reducing some artifacts through adaptive partitioning; however, it still suffers from residual distortions. In the Zelda image (Figure 12), FICANUP exhibits jagged edges and texture blurring in the aircraft wings and fuselage. In the Kodim21 and Kodim24 images (Figures 16 and 17), which contain complex natural textures, FICANUP generates ghosting effects and unnatural color transitions near object boundaries. Moreover, in remote sensing images such as RSI343 and RSI3515

(Figures 18 and 19), FICANUP introduces false patterns in homogeneous areas, degrading spatial consistency.

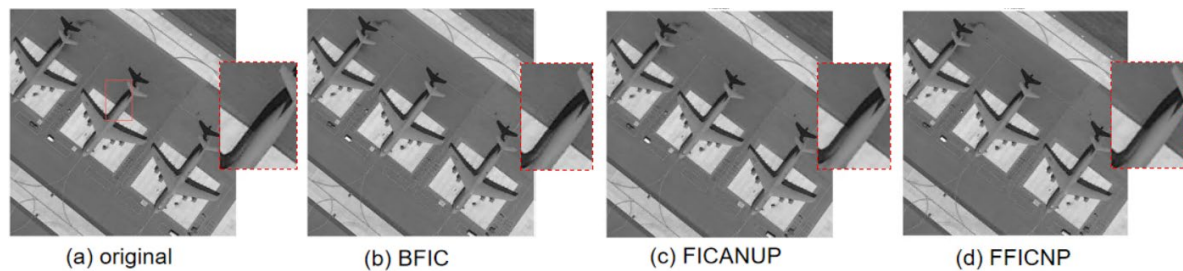


Figure 19. Comparison of reconstruction results for the RSI3515 image using different algorithms.

In contrast, the proposed FFICNUP yields visually smoother results with significantly fewer artifacts and better preservation of structural details, including edges, textures, and fine features. Specifically, in the Plane image, FFICNUP effectively mitigates blocking artifacts and edge discontinuities while maintaining sharpness along aircraft contours. In the Peppers image, the reconstruction preserves subtle surface gradients and avoids blockiness in smooth regions. In the Cameraman image, facial features such as eyes, hair, and clothing textures are restored with higher fidelity and without noticeable boundary distortions.

These improvements arise from an adaptive non-uniform block segmentation scheme that aligns partition boundaries with intrinsic image structures, thereby reducing mismatches in range-domain block mapping—a common source of speckle-like noise in conventional fractal coding. The qualitative observations are consistent with the quantitative gains in PSNR, demonstrating that FFICNUP not only achieves higher numerical accuracy but also delivers superior perceptual quality across diverse image types, including natural scenes, remote sensing data, and synthetic test images.

In summary, experimental results on a diverse set of standard and remote sensing images (512×512 to 2048×2048) clearly demonstrate that the proposed FFICNUP substantially outperforms both the classical BFIC and the recent FICANRP [10] across all evaluated metrics. Specifically, FFICNUP delivers consistently higher PSNR, comparable or superior compression ratios, and dramatically reduced encoding and decoding times, establishing state-of-the-art performance in decoding speed while maintaining or improving reconstruction fidelity and compression efficiency.

5. Conclusions and Future Work

In this study, we introduce fast fractal image compression with non-uniform partition. This innovative hybrid framework addresses the two main challenges of traditional fractal image compression: excessively long encoding times and limited reconstruction quality. By replacing the conventional single-threaded exhaustive search with a Task-Serial Workflow that incorporates intensive data-parallel vectorization and adopting a content-aware, non-uniform partition strategy, FFICNUP dynamically creates variable-sized R-blocks and D-blocks based on local texture and structural complexity. Finer blocks are allocated to complex, edge-rich regions to preserve visual details. In comparison, larger blocks are used in smooth areas to significantly accelerate encoding, resulting in notable improvements in computational efficiency, compression ratio, and reconstruction accuracy.

Experimental results demonstrate that FFICNUP establishes a new state of the art in decoding efficiency for fractal image compression. Specifically, it achieves decoding times of merely 0.2 s for 512×512 images and 2.8 s for high-resolution 2048×2048 remote sensing images, representing a remarkable $14\times$ speedup over conventional FIC methods. In addition to this efficiency gain, FFICNUP significantly improves encoding

performance, reducing encoding time by up to $2631.5\times$ relative to BFIC. Importantly, these computational advantages are achieved without sacrificing reconstruction fidelity or compression efficiency; the proposed method yields a PSNR improvement of up to 5.19 dB and a $2.21\times$ higher compression ratio on representative test images. These results indicate that FFICNUP effectively reconciles the traditionally conflicting objectives of speed, reconstruction quality, and compactness in fractal coding, offering a scalable, practical solution for real-time image compression in resource-constrained environments, such as edge computing and remote sensing platforms. Furthermore, FFICNUP's adaptive hybrid execution strategy dynamically selects the most suitable computation path—CPU-serial, multi-threaded CPU, or GPU acceleration—based on image resolution and available hardware resources. Validated across heterogeneous platforms ranging from resource-constrained edge devices to high-performance servers, it consistently achieves an optimal balance of encoding and decoding speed, reconstruction quality, and latency.

Despite recent advancements, fractal image compression (FIC) for ultra-high-resolution imagery, such as satellite remote sensing, digital pathology, and immersive media exceeding 4096×4096 pixels, remains severely constrained by prohibitive computational intensity and memory overhead. While adaptive partitioning improves reconstruction fidelity, it significantly expands the D-block search space, leading to excessive memory consumption. Furthermore, the task-serial characteristics of existing matching pipelines result in a superlinear increase in execution time relative to image size. This lack of linear scalability fundamentally restricts the potential for parallel acceleration when processing gigapixel-scale datasets, creating a critical bottleneck for real-time applications.

In future work, we plan to develop fully parallel and out-of-core architectures that leverage modern GPU memory hierarchies in conjunction with hierarchical D-block pool organizations, aiming to achieve substantial gains in both memory efficiency and computational scalability. A pivotal component of this framework is a feature-guided block pre-screening mechanism, which utilizes lightweight learning-based classifiers to rapidly categorize image blocks by their structural or semantic attributes, thereby aggressively pruning the search space before fine-grained, vectorized matching. Furthermore, we intend to integrate a Learnable Affine Transformer into our non-uniform partitioning model to optimize the scaling (F_s) and offset (F_o) parameters by replacing the traditional least squares method with a learned inference model, which can potentially achieve more precise parameter estimation with significantly reduced computational latency. This approach is expected to yield orders-of-magnitude acceleration in encoding speed while maintaining high Peak Signal-to-Noise Ratio (PSNR), making fractal compression viable for real-time, large-scale imaging applications.

Author Contributions: Conceptualization, M.L. and K.U.; methodology, M.L. and K.U.; software, M.L.; validation, M.L.; resources, M.L.; writing—original draft preparation, M.L.; writing—review and editing, M.L. and K.U.; visualization, M.L.; supervision, K.U.; project administration, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analyzed in this study. The UC Merced Land Use Dataset can be found here: [32]. The Kodak Lossless True Color Image Suite can be found here: <http://r0k.us/graphics/kodak/> (accessed on 20 June 2025) or in reference [33].

Acknowledgments: The authors are deeply grateful to the editor and reviewers for their careful review and suggestions, which have greatly enhanced the quality and rigor of this work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Barnsley, M.F. *Fractals Everywhere*, 2nd ed.; Morgan Kaufmann: San Diego, CA, USA, 1993; ISBN 978-0-12-079069-2.
2. Jacquin, A.E. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. Image Process.* **1992**, *1*, 18–30. [[CrossRef](#)] [[PubMed](#)]
3. Fisher, Y. *Fractal Image Compression: Theory and Application*; Springer Science & Business Media: New York, NY, USA, 1995; ISBN 978-1-4612-2472-3.
4. Nandi, U.; Mandal, J.K. Fractal Image Compression with Adaptive Quadtree Partitioning. In *Computer Science & Information Technology (CS & IT)*; Academy & Industry Research Collaboration Center (AIRCC): Tamil Nadu, India, 2013; pp. 289–296. [[CrossRef](#)]
5. Dwivedi, P.; Mishra, A. Fractal Image Compression based on Discrete Wavelet Transform. In Proceedings of the 2022 OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 14–16 December 2022; pp. 307–311. [[CrossRef](#)]
6. Smriti, S.; Laxmi, A.; Hima, B.M. Image Compression using PSO-ALO Hybrid Metaheuristic Technique. *Int. J. Perform. Eng.* **2021**, *17*, 998. [[CrossRef](#)]
7. Tang, Z.; Yan, S.; Xu, C. Adaptive super-resolution image reconstruction based on fractal theory. *Displays* **2023**, *80*, 102544. [[CrossRef](#)]
8. Panigrahy, M.; Chakrabarti, I.; Dhar, A.S. Space Limited Quadtree Based Fractal Image Encoder. In Proceedings of the 2024 IEEE 4th International Conference on Applied Electromagnetics, Signal Processing, & Communication (AESPC), Bhubaneswar, India, 29–30 November 2024; pp. 1–6. [[CrossRef](#)]
9. Yu, J. Image Encryption Technology Based on Fractal Image Compression Algorithm. *J. Cyber Secur. Mobil.* **2024**, *13*, 283–304. [[CrossRef](#)]
10. Li, M.; Tak U, K. An Enhanced Fractal Image Compression Algorithm Based on Adaptive Non-Uniform Rectangular Partition. *Electronics* **2025**, *14*, 2550. [[CrossRef](#)]
11. Wohlberg, B.E.; De Jager, G. Fast image domain fractal compression by DCTdomain block matching. *Electron. Lett.* **1995**, *31*, 869–870. [[CrossRef](#)]
12. Wang, J.; Zheng, N. A Novel Fractal Image Compression Scheme with Block Classification and Sorting Based on Pearson's Correlation Coefficient. *IEEE Trans. Image Process.* **2013**, *22*, 3690–3702. [[CrossRef](#)] [[PubMed](#)]
13. Jaferzadeh, K.; Moon, I.; Gholami, S. Enhancing fractal image compression speed using local features for reducing search space. *Pattern Anal. Appl.* **2017**, *20*, 1119–1128. [[CrossRef](#)]
14. Cao, J.; Zhang, A.; Shi, L. Orthogonal sparse fractal coding algorithm based on image texture feature. *IET Image Process.* **2019**, *13*, 1872–1879. [[CrossRef](#)]
15. Abedellatif, H.; Taha, T.E.; El-Shanawany, R.; El-Samie, F.E.A.; Zahran, O.F. Non-Exhaustive Search Method for Fractal Image Compression. In Proceedings of the 2021 International Conference on Electronic Engineering (ICEEM), Menouf, Egypt, 3–4 July 2021; pp. 1–5. [[CrossRef](#)]
16. Li, P.; Guan, Z.; Cao, J.; Hang, F. Research on Fractal Image Compression Algorithm Based on Generalized Mandelbrot Set. In Proceedings of the 2023 5th International Conference on Communications, Information System and Computer Engineering (CISCE), Guangzhou, China, 14–16 April 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 337–340. [[CrossRef](#)]
17. Li, L.-F.; Hua, Y.; Liu, Y.-H.; Huang, F.-H. Study on fast fractal image compression algorithm based on centroid radius. *Syst. Sci. Control Eng.* **2024**, *12*, 2269183. [[CrossRef](#)]
18. Varghese, B.; S., K. Parallel Computation strategies for Fractal Compression. In Proceedings of the 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 18–19 December 2020; pp. 1024–1027. [[CrossRef](#)]
19. Saad, A.-M.H.Y.; Abdullah, M.Z. High-Speed Fractal Image Compression Featuring Deep Data Pipelining Strategy. *IEEE Access* **2018**, *6*, 71389–71403. [[CrossRef](#)]
20. Sur les Opérations dans les Ensembles Abstraits et leur Application aux équations Intégrales. Available online: <http://matwbn.icm.edu.pl/ksiazki/or/or2/or215.pdf> (accessed on 16 August 2025).
21. Bagchi, S.; Mitra, S.K. The nonuniform discrete Fourier transform and its applications in filter design. I. 1-D. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **1996**, *43*, 422–433. [[CrossRef](#)]
22. Makur, A.; Mitra, S.K. Warped discrete-Fourier transform: Theory and applications. *IEEE Trans. Circuits Syst. Fundam. Theory Appl.* **2001**, *48*, 1086–1093. [[CrossRef](#)]
23. Donoho, D.L. Wedgelets: Nearly minimax estimation of edges. *Ann. Stat.* **1999**, *27*, 859–897. [[CrossRef](#)]
24. Tak, U.K.; Tang, Z.; Qi, D. A non-uniform rectangular partition coding of digital image and its application. In Proceedings of the 2009 International Conference on Information and Automation, Zhuhai/Macau, China, 22–24 June 2009; pp. 995–999. [[CrossRef](#)]
25. Yuan, X.; Cai, Z. An Adaptive Triangular Partition Algorithm for Digital Images. *IEEE Trans. Multimed.* **2019**, *21*, 1372–1383. [[CrossRef](#)]

26. Zhang, Y.; Cai, Z.; Xiong, G. A New Image Compression Algorithm Based on Non-Uniform Partition and U-System. *IEEE Trans. Multimed.* **2021**, *23*, 1069–1082. [[CrossRef](#)]
27. Zhao, W.; U, K.; Luo, H. Adaptive non-uniform partition algorithm based on linear canonical transform. *Chaos Solitons Fractals* **2022**, *163*, 112561. [[CrossRef](#)]
28. Zhao, W.; U, K.; Luo, H. Image representation method based on Gaussian function and non-uniform partition. *Multimed. Tools Appl.* **2023**, *82*, 839–861. [[CrossRef](#)]
29. Liu, X.; Kintak, U. A Multi-Focus Image Fusion Algorithm Based on Non-Uniform Rectangular Partition with Morphology Operation. In Proceedings of the 2018 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR), Chengdu, China, 15–18 July 2018; pp. 238–243. [[CrossRef](#)]
30. U, K.; Ji, N.; Qi, D.; Tang, Z.; Song, R. A Novel Image Denoising Algorithm Based on Non-Uniform Triangular Partition and Interpolation. In Proceedings of the 2010 International Conference on Future Power and Energy Engineering, Shenzhen, China, 26–27 June 2010; pp. 67–70. [[CrossRef](#)]
31. Han, X.; Chen, H.; Cheng, X.; Zhao, P. Continuous Estimation of sEMG-Based Upper-Limb Joint Angles in the Time–Frequency Domain Using a Scale Temporal–Channel Cross-Encoder. *Actuators* **2025**, *14*, 378. [[CrossRef](#)]
32. Yang, Y.; Newsam, S. Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, New York, NY, USA, 2–5 November 2010. pp. 270–279. [[CrossRef](#)]
33. Eastman Kodak Company. Kodak Lossless True Color Image Suite. Available online: <http://r0k.us/graphics/kodak/> (accessed on 2 June 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.