fractal and fractional

MDPI

Article

GPU-Accelerated Fractal Compression Dimension Estimation

Ángel Díaz-Herrezuelo [1] and Pedro Chamorro-Posada [1,2,*]

[1] Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, Universidad de Valladolid, ETSI Telecomunicación, Paseo de Belén, 15, 47011 Valladolid, Spain; angel.diaz.herrezuelo@uva.es
[2] Laboratory for Disruptive Interdisciplinary Science (LaDIS), Universidad de Valladolid, 47011 Valladolid, Spain
* Correspondence: pedro.chamorro@uva.es

Abstract

Fractal dimension is widely used as a quantitative descriptor of structural complexity in digital images. However, its practical implementation often involves methodological and computational trade-offs. The compression-based estimator provides an information-theoretic formulation that operates directly on grayscale images without mandatory binarization. Although the method is theoretically grounded and has been applied in real-world scenarios, its implementation-level behavior and computational characteristics have not been systematically analyzed under controlled conditions. To address this gap, this work presents a structured GPU-enabled validation framework for this estimator using synthetic Julia sets with known theoretical fractal dimensions. By focusing on their planar boundaries, which enable direct ground-truth comparison across multiple resolutions, numerical accuracy, statistical stability, and execution time are jointly evaluated across CPU and GPU implementations. Furthermore, additional experiments assess sensitivity to progressive Gaussian blur and exploratory behavior on grayscale textures from the Brodatz dataset, revealing that boundary-dominated fractals consistently yield dimensions between 1 and 2, whereas volumetric textures produce values greater than 2 without modifying the estimation framework. Performance profiling identifies distinct computational regimes and highlights a trade-off between robustness and execution time in the double-compression GPU configuration. This approach establishes a reproducible evaluation framework that supports the practical deployment of compression-based fractal dimension estimation in large-scale and time-constrained image analysis systems.

Keywords: fractal dimension; compression-dimension; real-time processing; image compression; Julia sets; Brodatz album; GPU computing; CUDA

Check for updates

Academic Editor: Carlo Cattani

Received: 16 January 2026
Revised: 3 March 2026
Accepted: 4 March 2026
Published: 6 March 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) license.

1. Introduction

Fractal dimension is a mathematical descriptor that characterizes the scaling behavior of self-similar structures and scale-invariant patterns, as originally formalized by Mandelbrot [1]. In digital image analysis, it has been widely used for texture classification [2–4], shape analysis [5–7], structural pattern recognition [8–10], and, most prominently, medical diagnosis [11,12]. Beyond its scientific and mathematical interest, fractal dimension has also been linked to human visual preferences [13]. These diverse applications illustrate the practical relevance of fractal-based descriptors in real-world image analysis scenarios.

Among the multiple approaches available for estimating fractal dimension in digital images, geometric techniques such as box-counting and its variants have long served as reference methods due to their conceptual clarity and direct interpretation in terms of

spatial coverage across scales [14]. In digital settings, however, these methods typically require image binarization and repeated multi-scale grid evaluations, making their numerical behavior sensitive to discretization choices, scale selection, and resolution. Furthermore, the evaluation of multiple scales over high-resolution images may introduce non-negligible computational cost, particularly when large datasets or repeated measurements are required. This tension between theoretical relevance and practical implementation efficiency has been noted in applied contexts such as industrial inspection [15], video analysis [15], and large-scale scientific imaging [16].

From an information-theoretic perspective, image compression has been proposed as an alternative mechanism for estimating fractal dimension. Chamorro-Posada [17] introduced the concept of compression dimension, in which the information dimension D_I is approximated from the slope of a log–log relationship between losslessly compressed image size and the image resolution. This formulation provides a quantitative link between entropy scaling and structural complexity by reformulating fractal dimension estimation in terms of redundancy exploitation by compression algorithms. Beyond its theoretical foundation, compression-based fractal descriptors have demonstrated practical utility in real-world structural analysis tasks, including water detection and smoke recognition [9,10], supporting the applicability of the approach beyond synthetic examples. Despite this theoretical grounding and empirical use, the compression-dimension estimator has not yet been subjected to a systematic implementation-level validation under controlled ground-truth conditions. In particular, there is a lack of studies that jointly analyze (i) numerical accuracy against known theoretical dimensions, (ii) statistical stability across repeated executions and resolutions, (iii) computational cost under different architectural configurations, and (iv) the impact of specific design decisions such as resizing strategy or recompression. Moreover, although GPU acceleration has been widely explored for classical counting-based fractal estimators, the structured integration of compression-driven scaling analysis within a fully GPU-enabled pipeline remains comparatively underexamined.

The objective of this work is therefore not to introduce a new estimator, but to provide a GPU-enabled structured validation framework for the compression-dimension approach. The study evaluates five implementations—four compression-based pipelines and a classical box-counting baseline—under controlled experimental conditions. Validation is conducted using a dataset of synthetic Julia sets whose fractal dimension is theoretically defined in [18]. Since the fractal dimension of a Julia set is associated with the geometry of its boundary, the analysis focuses on planar boundary structures, allowing direct comparison between estimated and theoretical values across multiple spatial resolutions.

Beyond controlled boundary validation, the study extends the analysis in two additional directions. First, a systematic blur experiment evaluates the estimator's response to progressively increasing structural degradation. By applying increasing levels of Gaussian smoothing, fine-scale details are gradually attenuated while global structure is preserved. The observed monotonic evolution of estimation error across blur levels provides insight into the estimator's sensitivity to multiscale structural information under controlled perturbations.

Second, an exploratory evaluation on grayscale textures from the Brodatz album [19] examines the behavior of the estimator outside strictly planar boundary scenarios. Without modifying the estimation framework, textures exhibiting distributed volumetric structure yield dimension values greater than 2, whereas Julia boundary images consistently produce values between 1 and 2. This distinction emerges naturally from the compression-based scaling behavior, suggesting that the estimator responds coherently to different structural regimes. While no formal claim is made regarding volumetric fractal modeling, the results

indicate that the same framework can differentiate boundary-dominated patterns from images with spatially distributed structural variation.

In addition to numerical validation, the study presents a detailed characterization of execution times and architectural behavior. Five implementations—CPU and GPU variants with and without recompression, together with a classical box-counting baseline—are systematically evaluated. Particular attention is given to the two-compression GPU configuration (GPU_2C), which demonstrates improved statistical stability at higher resolutions with only moderate computational overhead, illustrating an interesting trade-off between robustness and execution time.

Together, these analyses establish a reproducible evaluation framework that combines controlled ground-truth validation, structural sensitivity assessment, exploratory regime differentiation, and computational profiling within a coherent GPU-enabled pipeline. By situating compression-based fractal dimension estimation within a rigorously characterized computational context, this work contributes practical insight into its stability, scalability, and structural responsiveness, thereby facilitating its integration into large-scale or time-constrained image analysis systems while preserving its theoretical foundations.

To contextualize this contribution, it is necessary to examine how fractal dimension calculation in digital images has traditionally been addressed in the literature. The problem has been extensively studied through a range of methodologies, which can be broadly categorized into four groups: classical geometric techniques, entropy- and compression-based approaches, GPU-accelerated implementations and machine learning-based estimators.

1.1. Classical Methods

Traditionally, fractal dimension estimation has relied on geometric scaling properties. Methods such as box-counting (Minkowski–Bouligand) remain among the most widely used due to their simplicity. The method estimates fractal dimension by counting the number of occupied boxes across multiple spatial scales and fitting a linear model in log–log space. Despite its popularity, classical box-counting typically requires prior image binarization and is known to be sensitive to image resolution, grid alignment, and the selection of the scaling range used for regression [8,20].

To address the limitations of binarization in grayscale images, variants based on differential box-counting (DBC) were introduced [14,21]. These methods extend traditional box-counting by incorporating intensity variations within each spatial cell by partitioning the image into three-dimensional grids, enabling direct calculation of fractal dimension without requiring prior binarization. Multiple refinements have been proposed over time, addressing issues such as discretization strategies, noise sensitivity, and resolution dependence [22–24]. DBC-based approaches have been successfully applied in diverse domains, including geological image analysis and microstructural characterization of materials [25,26], where combining fractal metrics with texture descriptors enhances discriminative performance. While DBC variants mitigate the need for binarization, they introduce additional discretization parameters related to vertical partitioning, which may affect comparability across implementations. For this reason, in the present study, classical box-counting is considered as a standardized geometric baseline for benchmarking purposes.

Recent studies have further examined methodological sensitivities inherent to box-counting frameworks. Issues such as dependence on box-height discretization, grid-size selection, partial cell coverage at image borders, and variability induced by the counting origin have been systematically analyzed [27–30]. These works highlight that scale parameter choice and grid configuration can significantly influence the estimated dimension, motivating improved discretization schemes, weighted counting strategies, and automated scale-selection procedures to enhance reproducibility and robustness.

Spectral, variogram-based, and temporal techniques, such as the Higuchi method [8,31,32], infer scaling behavior from frequency, intensity, or time-domain patterns. In parallel, alternative mathematical definitions of fractal dimension grounded in metric or topological formalisms have been rigorously developed [20]. Despite their popularity, both traditional approaches and their optimizations are also affected by resolution, alignment, or scale-selection issues.

1.2. Entropy- and Compression-Based Methods

Instead of analyzing spatial distribution, these techniques leverage the relationship between complexity and data compressibility. The foundational work by Barnsley [33] introduced fractal image compression as a way to exploit self-similarity in natural images. Later, Chamorro-Posada [17] introduced the concept of compression dimension, using compressed image size across scales to estimate information dimension D_I. A practical advantage of this approach over others is the fact that it does not require image binarization, allowing the method to operate directly on grayscale images without additional preprocessing steps that would otherwise increase the overall computational cost. Subsequent studies have supported this connection, linking normalized spatial entropy and Kolmogorov complexity with fractal dimension across diverse datasets [3,34].

1.3. Parallel Implementations

To address the demands of real-time processing, recent literature has shifted towards hardware-level optimizations. Specifically, GPU-accelerated frameworks have been designed to meet the significant computational demands of multi-scale fractal analysis, particularly for high-resolution and multidimensional data. CUDA-based versions of box-counting have achieved significant speedups by exploiting massive parallelism and minimizing CPU–GPU data transfers. For instance, optimized GPU strategies based on binary operations have enabled acceleration factors exceeding one order of magnitude compared to sequential CPU implementations, with demonstrated applicability to large 2D, 3D, and even 4D datasets such as EEG recordings [35].

More recent developments have extended differential box-counting to color images by modeling pixels in higher-dimensional feature spaces, enabling fractal estimation in RGB domains while maintaining efficient GPU utilization. Highly parallel CUDA implementations have reported extreme acceleration factors relative to CPU baselines, particularly for large-scale images, while preserving estimation accuracy comparable to classical approaches [36]. These advances confirm that state-of-the-art fractal dimension computation can be effectively scaled through GPU-based parallelism.

In addition to dedicated implementations, GPU support has been incorporated into existing analysis toolkits such as the FDI Toolbox [37], facilitating broader adoption in applied settings. Furthermore, general-purpose GPU compression libraries such as nvcomp [38] provide infrastructure for high-throughput data processing directly on device memory. Despite these advances, most parallel efforts focus on accelerating discrete counting procedures derived from classical formulations. The systematic integration of compression-driven scaling analysis into a fully GPU-resident pipeline remains comparatively underexplored [3].

1.4. ML-Based Estimators

Recently, machine learning has been introduced as an alternative strategy to enhance precision. These models, including convolutional neural networks and transfer-learning architectures, have been trained to predict fractal-related quantities such as the Hurst exponent or fractal dimension directly from image data [39,40]. These data-driven estimators have demonstrated competitive accuracy and reduced inference time once trained, particularly on synthetically generated datasets designed to span controlled parameter ranges. In addition to direct estimation, fractal descriptors have been integrated within broader

machine learning pipelines for classification and pattern recognition tasks, enhancing discriminative performance in application-specific contexts [41,42].

Unlike classical geometric or entropy-based techniques, machine learning estimators do not explicitly rely on multi-scale counting or regression in log–log space. Instead, they learn a mapping between image features and fractal descriptors through supervised training. While this strategy can reduce estimation error under controlled conditions, the performance is inherently dependent on the representativeness of the training data, the dimensionality of the parameter space, and the chosen model architecture.

In contrast, unsupervised analytical approaches based on geometric scaling or information-theoretic principles preserve an explicit relationship between image structure and scaling behavior without requiring prior training. These methodological differences highlight complementary perspectives in fractal analysis, with data-driven estimators emphasizing predictive performance and analytical methods retaining interpretability and direct scalability across heterogeneous datasets.

1.5. Applications

Beyond theoretical estimation, fractal analysis serves as a robust descriptor in several fields. Key applications include texture classification [8,43], shape retrieval [6] and biomedical diagnostics [11]. However, classical techniques are often too slow for real-time use and, even in faster variants, typically require an initial image binarization stage. The compression-based estimator [17], which operates directly on grayscale images, while promising in terms of robustness, still requires broader validation. The present work addresses this gap by providing a systematic analysis of competitive accuracy and computational efficiency, supporting the feasibility of real-time-oriented fractal analysis and its integration with advanced descriptors such as lacunarity, multiscale entropy, or multifractal analysis [3,11].

2. Materials and Methods

2.1. Theoretical Background and Fractal Dimension Calculation

The estimation approach adopted in this work relies on fractal dimension as a quantitative descriptor of the scaling behavior of self-similar structures, both in geometric and informational terms. In the context of digital images, fractal dimension characterizes how image detail or information content varies as a function of observation scale, and can be formalized through different but closely related definitions grounded in fractal geometry and information theory.

Among these, the information dimension D_I plays a central role when the object of interest is represented as a discrete distribution, such as a digital image. The information dimension describes the rate at which the Shannon entropy of a probability distribution grows with increasing resolution. For an image interpreted as a measure distributed over a two-dimensional support, this concept links fractal dimension to the scaling properties of information content.

The estimator employed in this work follows the formulation proposed in [17], in which the information dimension is approximated through the scaling behavior of losslessly compressed image representations. Under this framework, the compressed size $S(s)$ of an image rescaled by a factor s is assumed to be proportional to the amount of information required to encode the image at that scale. This assumption is grounded in the principle that lossless compression algorithms exploit structural redundancy within the data. Regions exhibiting regularity, self-similarity, or repeated geometric patterns can be encoded more efficiently, whereas highly irregular or complex regions require more bits.

Consequently, the amount of redundancy detected by the compressor is directly related to the geometrical regularity of the underlying structure. As the observation scale changes, the structural organization of the image evolves, modifying the balance between redundancy and unpredictability. The variation of the compressed size across scales therefore reflects changes in structural complexity. In this sense, the scaling behavior of $S(s)$ provides an indirect measurement of how informational content, and thus entropy, grows with resolution. This establishes a conceptual bridge between geometric fractal analysis and information theory: the estimator interprets fractal dimension as a scaling exponent governing the growth of entropy under multiscale observation.

Formally, the estimator is based on the fact that, for self-similar structures, the relationship between the compressed size S and the scale factor s follows an approximate power law. Taking logarithms, this relationship can be expressed as a linear model in the $(\log s, \log S)$ plane, whose slope provides an estimate of the fractal dimension. The fractal dimension is therefore obtained as the slope of a linear regression fitted to these log–log pairs.

For comparison, the classical box-counting (BC) dimension is also computed as a geometric baseline. In the standard formulation, the image is covered with square boxes of side length ϵ, and the number of boxes $N(\epsilon)$ intersecting the structure is counted. The box-counting dimension is defined as the scaling exponent relating $N(\epsilon)$ and ϵ in a log–log representation. In digital implementations, this requires binarization of the image prior to counting.

In this work, a fixed threshold of 0.2 (equivalently 51 in uint8 representation) is adopted for binarization. This choice is motivated by the intensity distribution of the analyzed fractal images. Non-white gray levels arise primarily from the convergence speed used in the numerical generation of the fractal. These values are concentrated at the lowest intensity levels of the scale. Empirically, it was observed that a threshold at 0.2 is sufficient to remove these residual grayscale variations while preserving the geometrical support of the fractal without removing relevant structural components.

The box sizes follow dyadic scales of the form 2^k. The use of dyadic box sizes is consistent with established digital box-counting implementations, where sequential divisions by powers of two are recommended to ensure coherent multiscale coverage and stable log–log linearity [44]. This choice guarantees nested grid refinement across scales and avoids irregular sampling artifacts. In practice, the log–log curves obtained using these dyadic scales exhibit a clear linear regime, supporting the appropriateness of this discretization strategy.

Regarding implementation, a vectorized approach based on NumPy is adopted. Specifically, the function `numpy.add.reduceat` is employed to aggregate contiguous memory blocks corresponding to box partitions. Instead of explicitly re-sampling the image or iterating through pixels using nested loops, this method performs block summations over contiguous memory segments. This significantly improves computational efficiency, enabling a fast and scalable implementation while preserving the standard geometric definition of the box-counting estimator.

2.2. Overview of the Proposed Pipeline

Each image undergoes autocropping and normalization, and is then downscaled using nine scale factors ranging from 10% to 90%. In the case of the box-counting method, a preliminary binarization step is implicitly assumed as part of the standard preprocessing procedure. In the compression dimension approach, the rescaled images are then compressed, and optionally recompressed, to analyze how the compressed size S scales with the image scale s. The dimension estimate is computed from the slope of the linear regression on the resulting $(\log s, \log S)$ pairs.

2.3. Dataset Description and Structure

2.3.1. Synthetic Julia Fractal Dataset

The experimental dataset consists exclusively of synthetic images generated from Julia sets, whose fractal structure is characterized by the geometry of their boundary. This choice is motivated by their well-known theoretical properties. Julia sets exhibit a fractal dimension that is analytically defined and can be continuously controlled within the interval between 1 and 2 through the selection of the complex parameter c. This enables the construction of a reproducible dataset covering a broad and continuous range of structural complexities, which is sufficient to evaluate the computational behavior of the estimator.

A total of 16 distinct Julia set configurations were generated, each corresponding to a different configuration of the complex parameter c. To analyze scalability effects and resolution-dependent behavior, each configuration was generated at six different spatial resolutions. This results in a total of 96 images. The following naming convention was adopted to denote image resolution:

- _low: 1000 pixels per side.
- _med: 2000 pixels per side.
- _high: 5000 pixels per side.
- _highx2: 8000 pixels per side.
- _highx20: 20,000 pixels per side.
- _highx40: 40,000 pixels per side.

To illustrate the diversity of the dataset, representative examples of the generated Julia sets are shown in Figure 1. These images are displayed using a logarithmic grayscale to enhance subtle structural differences and highlight the variation in complexity across the dataset.

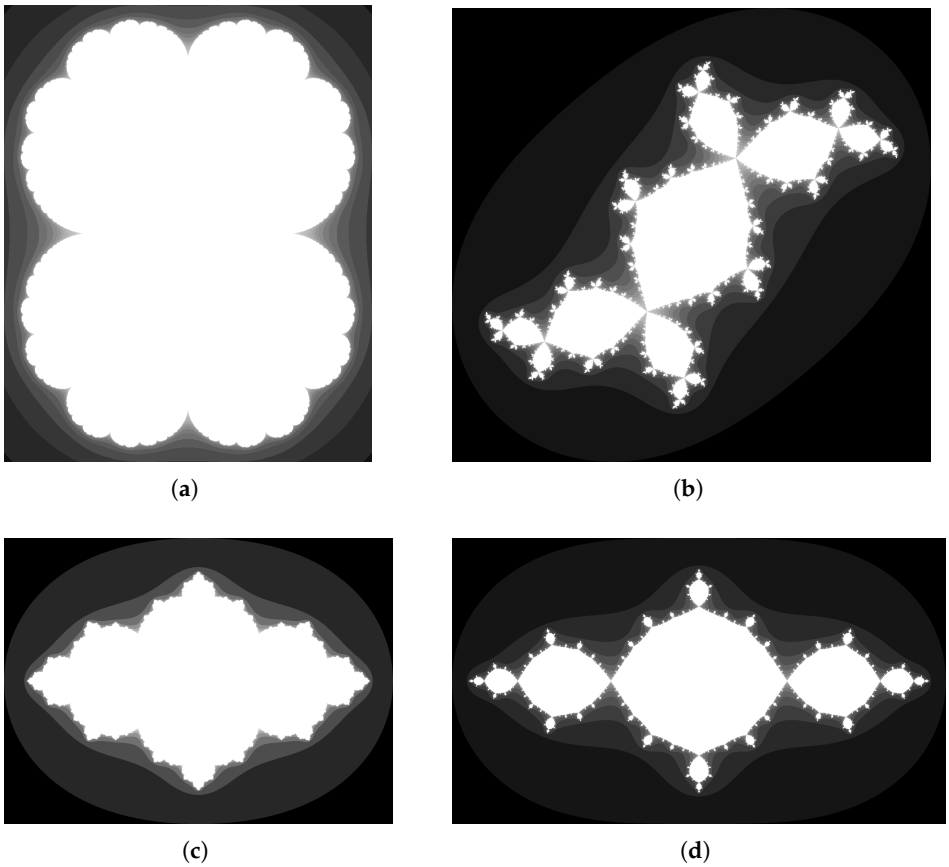


Figure 1. Representative Julia sets with known theoretical fractal dimensions used for validation of the estimator [18]. Panels correspond to: (**a**) $D \sim 1.0812$, (**b**) $D \sim 1.3934$, (**c**) $D \sim 1.1131$, and (**d**) $D \sim 1.2683$.

2.3.2. Blur Stability Analysis

To evaluate the robustness of the estimator under progressive structural degradation, a blur stability analysis was conducted on the complete synthetic Julia dataset described in Section 2.3.1. A Gaussian smoothing filter was applied to each image prior to processing. This gradually attenuates fine-scale structural details while preserving the global geometry of the fractal boundary.

The blur operation was implemented using the `GaussianBlur` function from OpenCV v4.13.0. The following σ values were evaluated: 0.25, 0.5, 1, 1.5, 2, 3, 4, and 5.

For each σ level, the entire set of 96 Julia images was processed through the same compression-dimension pipeline used in the main experiments. The estimation error was then computed following the same protocol used for the baseline dataset. This enables a direct comparison between the original and blurred conditions. This procedure allows the assessment of the consistency of the estimated fractal dimension and the evolution of the error as a function of increasing structural smoothing.

The experiment was conducted exclusively using the GPU-based two-compression configuration, as this implementation corresponds to the primary accelerated variant analyzed in this work.

To illustrate the diversity of the blur dataset, representative examples of the generated Julia sets are shown in Figure 2.

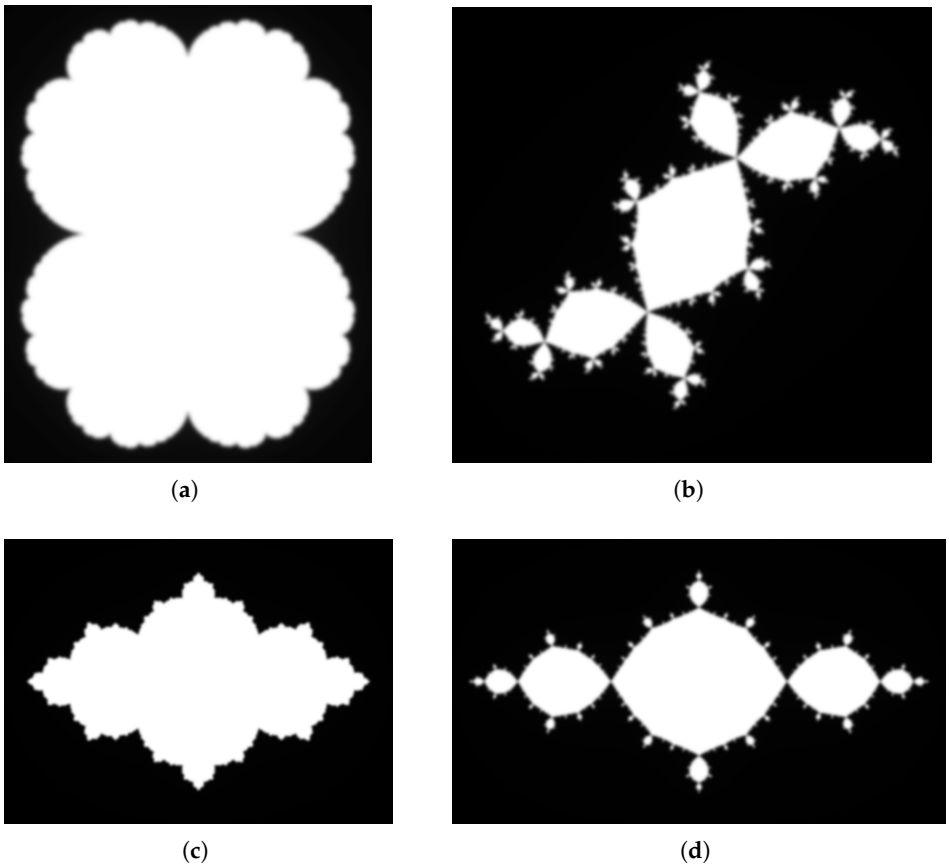


Figure 2. Blurred versions of the representative Julia sets shown in Figure 1. An isotropic Gaussian blur with standard deviation $\sigma = 5.0$ was applied. Panels correspond to: (**a**) $D \sim 1.0812$, (**b**) $D \sim 1.3934$, (**c**) $D \sim 1.1131$, and (**d**) $D \sim 1.2683$.

2.3.3. Brodatz Texture Dataset

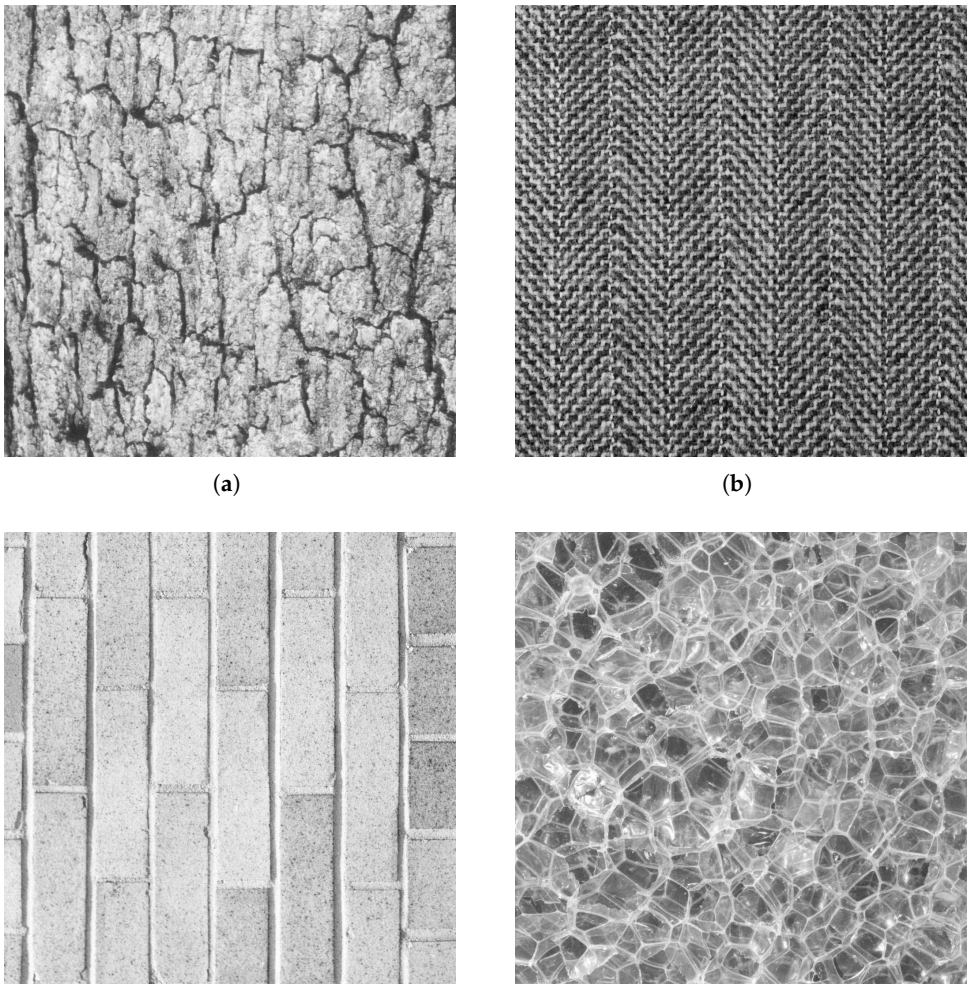
In addition to the synthetic fractal dataset, a subset of real-world texture images from the Brodatz album [19] was incorporated to analyze inter-texture stability and dis-

criminative behavior. The Brodatz dataset is a widely used benchmark in texture analysis, consisting of grayscale images representing a broad variety of natural and artificial surface patterns.

A total of 13 texture images were selected, each with an original spatial resolution of 1024×1024 pixels. To assess intra-texture consistency and spatial stability, five non-overlapping tiles were extracted from each image: the four quadrants and one central region. Each tile had a resolution of 512×512 pixels.

The complete compression-dimension pipeline with two compression stages was applied independently to each tile, yielding a single fractal dimension estimate per tile. For each texture image, the mean fractal dimension and the corresponding standard deviation were then computed across the five tiles. These aggregated statistics were used to evaluate the stability of the estimator within homogeneous texture regions and to examine its ability to differentiate between structurally distinct volumetric textures that are not strictly fractal in the mathematical sense.

To illustrate the diversity of the textures dataset, representative examples of the Brodatz album images used in this experiment are shown in Figure 3.



(**a**) (**b**)

(**c**) (**d**)

Figure 3. Representative grayscale textures from the Brodatz dataset [19] used for exploratory evaluation of the compression-based fractal dimension estimator. Panels correspond to: (**a**) Bark texture, (**b**) Woolen cloth texture, (**c**) Brick wall texture, and (**d**) Plastic bubbles texture. Each image was divided into five tiles for independent estimation, and the mean, standard deviation, and coefficient of variation were computed per texture.

2.4. Image Processing and Estimation Pipeline

2.4.1. Resizing Strategy

In this study, resizing is a crucial preprocessing step in the estimation pipeline. Images must be scaled across multiple factors before compression. The selected algorithm directly affects the resulting entropy and, consequently, the compressed size used in the log–log regression to estimate the fractal dimension.

After evaluating several options, block averaging was selected: In this method, each region of the original image is reduced to a single pixel whose value is the mean of the corresponding block. This approach preserves the statistical structure of the image without introducing artifacts or smoothing effects that may alter its complexity. It prioritizes the preservation of informational content over visual fidelity, maintaining a stable relationship between resolution and redundancy. Both CPU and GPU implementations follow the same logic to ensure methodological consistency. In the GPU approach, the resizing operation is implemented using a CUDA kernel (the implementation details of which can be found in Supplementary Materials, Listing S1), whereas the CPU version uses a compiled C module to improve the performance.

2.4.2. Compression Procedure

The approach follows the compression dimension formulation proposed in [17], where the information dimension D_I is calculated through the scaling behavior of losslessly compressed images. In accordance with the original formulation, the compression algorithm must be strictly lossless to preserve the full informational content of the image and maintain the theoretical link between compressed size and entropy scaling.

In this work, compression is performed using the Zstandard (Zstd [45]) algorithm: the reference Zstd library [46] is used for the CPU implementation, while the GPU-accelerated version relies on the NVIDIA nvCOMP library [38] (the implementation details of which can be found in Supplementary Materials, Listing S2).

2.4.3. Regression Strategy

The fractal dimension is obtained from a linear regression in the log–log domain between the image scale s and the corresponding compressed size S. For each image, nine $(\log s, \log S)$ pairs are obtained from the predefined set of scale factors.

For self-similar fractal structures with a sufficiently wide dynamic range, the relationship between scale and informational content is expected to follow an approximately linear behavior in the log–log domain. In digital implementations, however, deviations from linearity typically arise at the extreme scales. At very large scales, structural details are partially lost due to coarse representation. At very small scales, discretization effects and information saturation lead to a flattening of the curve.

This behavior has been consistently observed in the Julia dataset analyzed in this work. The log–log curves consistently exhibit mild flattening at the smallest and largest scales, while maintaining a stable linear regime across the intermediate scales. A similar effect is described in the foundational formulation of the compression-dimension method [17], where the loss of linearity at extreme scales is explicitly discussed.

Since the theoretical model underlying the estimator assumes a power-law relationship between compressed size and scale, the log–log representation is expected to follow a linear model whose slope directly estimates the fractal dimension [17]. For this reason, ordinary least-squares linear regression is employed, as the quantity of interest is the slope of this linear relationship. The use of alternative nonlinear regression models would not be consistent with the theoretical formulation of the estimator.

Rather than enforcing the use of all available scales, a predefined set of regression configurations is evaluated. Specifically, four cases are considered: removing the first point,

removing the last point, removing both the first and the last points, and removing the first two points and the last one. The selection of these particular configurations is motivated by the empirical observation that the flattening effects in the analyzed dataset consistently occur at those extreme scales. The removal of intermediate points is not permitted, ensuring preservation of a contiguous and physically meaningful multiscale interval.

For each configuration, a least-squares regression is computed and the mean squared error (MSE) is evaluated. The final fractal dimension estimate corresponds to the slope of the configuration providing the lowest MSE, thereby retaining the dominant linear regime while discarding extreme-scale deviations in a deterministic manner.

2.5. Implementation Details and Data Availability

The overall control flow and pipeline orchestration are implemented in Python 3.10.12 [47], which manages data loading, preprocessing, and coordination of the different computational stages. Performance-critical operations, specifically image resizing and compression, are offloaded to the GPU through custom C/CUDA modules. This separation allows the computationally intensive components to benefit from GPU parallelism while keeping the control logic simple and transparent.

For the GPU implementation, memory transfers between host and device are explicitly minimized. Each image is transferred once from host to device memory, after which all resizing and compression operations are performed entirely within device memory. Only the final compressed size (a scalar value per scale factor) is transferred back to the host for regression. The complete host–device interaction workflow is summarized in Algorithm 1.

Algorithm 1 Host–Device Execution Model of the GPU Resize–Compression Pipeline

Load CUDA kernel and initialize nvCOMP codec
$h_img \leftarrow$ Read image from disk
$(H, W) \leftarrow$ Image dimensions
$d_img \leftarrow$ Copy image to device memory ▷ Single H2D transfer
Preallocate device buffers:
  $d_resized^{max}, d_comp^{max}, d_comp2^{max}$
for $f = 1$ to 9 **do**
  $(H_f, W_f) \leftarrow$ Compute resized dimensions
  Resize kernel on GPU:
    $d_resized^{max} \leftarrow \text{Resize}(d_img)$
  Compress on GPU:
    $d_comp^{max} \leftarrow \text{Encode}(d_resized^{max})$
  if recompression enabled **then**
    $d_comp2^{max} \leftarrow \text{Encode}(d_comp^{max})$
    $s_d \leftarrow \text{GetSize}(d_comp2^{max})$
  else
    $s_d \leftarrow \text{GetSize}(d_comp^{max})$
  end if
  $s \leftarrow$ Copy s_d to host ▷ Scalar D2H
  Append s to size list
end for
Perform linear regression on CPU
return slope, MSE

GPU memory is explicitly managed to ensure clean execution and to avoid interference between successive runs. Execution times for each functional block are measured using straightforward timestamping before and after each operation. The CUDA implementation deliberately relies on simple kernels and a direct execution architecture, which is sufficient

for the objectives of this work and consistent with the focus on evaluating computational acceleration rather than exploring advanced GPU optimization strategies.

Due to the size of the generated image dataset, only representative data samples and the scripts required to generate and process the dataset are made available. This allows the experiments to be reproduced following the methodology described in this section. Further details are provided in the Data Availability Statement.

2.6. Experimental Setup

All computations were performed on Genoa nodes at SCAYLE [48], equipped with AMD EPYC 9454P CPUs (Advanced Micro Devices, Santa Clara, CA, USA) (96 threads) and NVIDIA H100 GPUs (80GB HBM3, NVIDIA Corporation, Santa Clara, CA, USA). To ensure temporal consistency in shared environments, each configuration was executed 30 times and results were averaged. A warm-up phase mimics data loading at different scales to reduce variability in the initial executions.

In addition to the high-throughput computing environment, a scalability experiment was conducted on a consumer-grade GPU (NVIDIA RTX 4050 Ti). This experiment evaluates performance behavior beyond data-center-class accelerators. The complete Julia dataset described in Section 2.3.1 was processed under identical experimental conditions, using the GPU-based two-compression configuration (GPU_2C). As in the primary setup, each image was executed 30 times and execution times were averaged. This complementary evaluation enables the assessment of computational scalability across different hardware tiers while maintaining a consistent experimental protocol.

2.7. Methods Compared

Five implementations are considered in this study. Four correspond to variants of the proposed compression-based pipeline, while a classical box-counting method is included as a reference approach:

- GPU: GPU pipeline without recompression.
- GPU_2C: GPU pipeline with recompression.
- CPU: CPU pipeline without recompression.
- CPU_2C: CPU pipeline with recompression.
- BC: Classical box-counting method deployed on CPU, used as a methodological reference.

The four pipeline-based implementations are evaluated to analyze trade-offs between execution time, accuracy, and resource usage under different computational configurations. The box-counting method is included to provide a classical geometric reference for the estimated scaling behavior, rather than as a direct performance baseline. Only the box-counting baseline operates on the fractal boundary (counting only boxes that are neither empty nor full), in order to emphasize the geometrically relevant structures; the proposed compression-based pipelines are applied directly to the input grayscale images. The regression is always performed on CPU due to its negligible cost and higher efficiency for small-scale operations.

2.8. Estimation Performance Metrics

The proposed framework was evaluated using complementary accuracy and stability metrics, depending on the availability of a theoretical fractal dimension. For synthetic fractals (Julia sets), where the theoretical dimension is known, estimation accuracy was quantified through the mean relative error. For each image, the relative error was computed as the absolute difference between the estimated and theoretical fractal dimensions, normalized by the theoretical value. These individual errors were then averaged across

images for each resolution and method. To characterize dispersion and robustness, the standard deviation (STD), coefficient of variation (CV), and 95% confidence intervals were computed over repeated runs under identical conditions. Confidence intervals are reported through their lower and upper bounds, as well as their half-width.

For natural textures from the Brodatz dataset, no ground-truth fractal dimension is available. In this case, the evaluation focuses on descriptive statistics of the estimated dimensions. For each of the 13 textures, the mean estimated fractal dimension, standard deviation, and coefficient of variation are reported. This analysis enables the assessment of the internal consistency and stability of the estimator when applied to real-world grayscale images. Additional details are provided in Supplementary Materials Tables S1–S9.

In some configurations, the coefficient of variation reaches relatively high values, occasionally exceeding 100%. This effect results from the very small mean relative errors observed in the synthetic experiments (typically around 1–2%). Since the CV measures dispersion normalized by the mean, small expected values can inflate the ratio even when absolute variability remains low. Importantly, the corresponding standard deviations remain small in absolute terms, indicating that the estimation errors are tightly clustered. Therefore, CV values should be interpreted together with the absolute error and standard deviation.

3. Results

3.1. Accuracy of Estimated Dimension

The accuracy of the estimated fractal dimension was assessed using the relative error metric, computed with respect to the theoretical dimension of each Julia set image. The reported errors are method-dependent. In particular, the classical box-counting approach operates on a binarized representation of the image, whereas the compression-based estimator processes the original grayscale image. These methodological differences lead to variations in the numerical error values across implementations.

The complete numerical results, indicating the mean relative error and standard deviation for each configuration and resolution, along with other relevant metrics, are reported in Supplementary Materials Tables S1–S5, and the corresponding resolution-wise comparisons are illustrated in Figure 4.

The relative error remains within a moderate range across all implementations and resolutions, indicating that the compression-based estimator achieves accuracy comparable to the classical box-counting approach. While differences between methods are observable, they remain limited in magnitude, and no systematic instability is observed across the evaluated configurations.

At lower resolutions, the GPU-based implementations show competitive or slightly improved performance relative to CPU-based variants. As resolution increases, both CPU implementations and box-counting tend to show marginally lower mean errors, although the differences remain moderate. The two-compression configuration demonstrates improved stability at higher resolutions compared to the single-compression variant, highlighting the benefit of the second compression stage in maintaining consistency.

The standard deviation of the relative error remains controlled for most configurations, with larger dispersion observed primarily at the extreme resolutions.

3.2. Blur Robustness Evaluation

The blur robustness evaluation shows a consistent monotonic increase in relative error as the Gaussian smoothing parameter σ increases. A negligible blur level ($\sigma = 0.25$) produces results that are indistinguishable from the original images, indicating that minimal smoothing does not significantly alter the multiscale structure captured by the estimator.

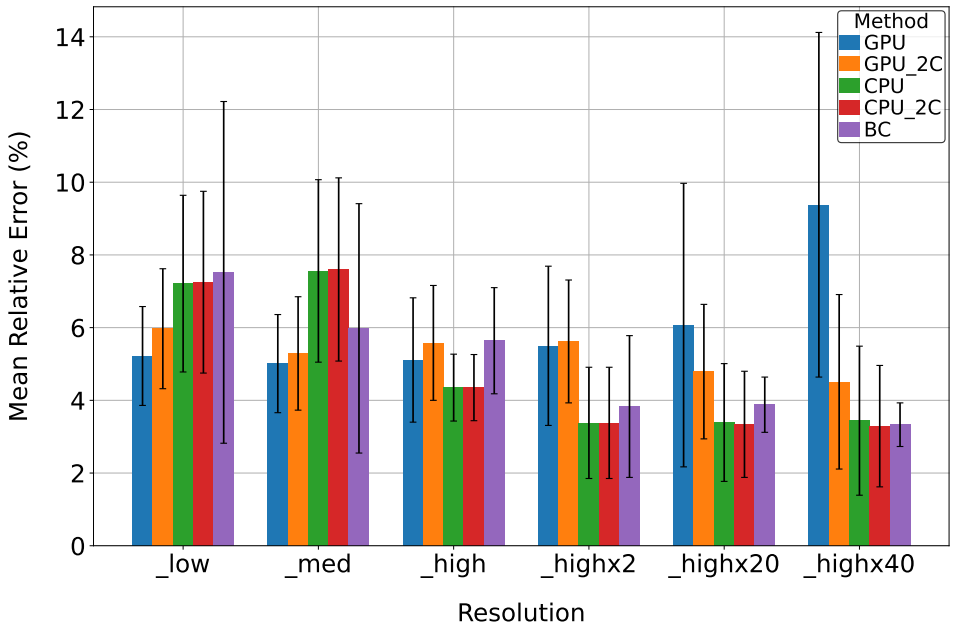


Figure 4. Mean relative error (%) of fractal dimension estimation for the synthetic Julia dataset across resolutions. Error bars represent the standard deviation over repeated runs. Results are shown for all evaluated implementations, including classical box-counting (BC) and compression-based methods (CPU, CPU_2C, GPU, and GPU_2C).

From $\sigma = 0.5$ onward, the relative error increases progressively across all resolutions, reaching a plateau at higher σ values. This behavior is systematically observed for all evaluated resolutions, with both the mean error and its dispersion increasing as structural smoothing intensifies, as illustrated in Figure 5.

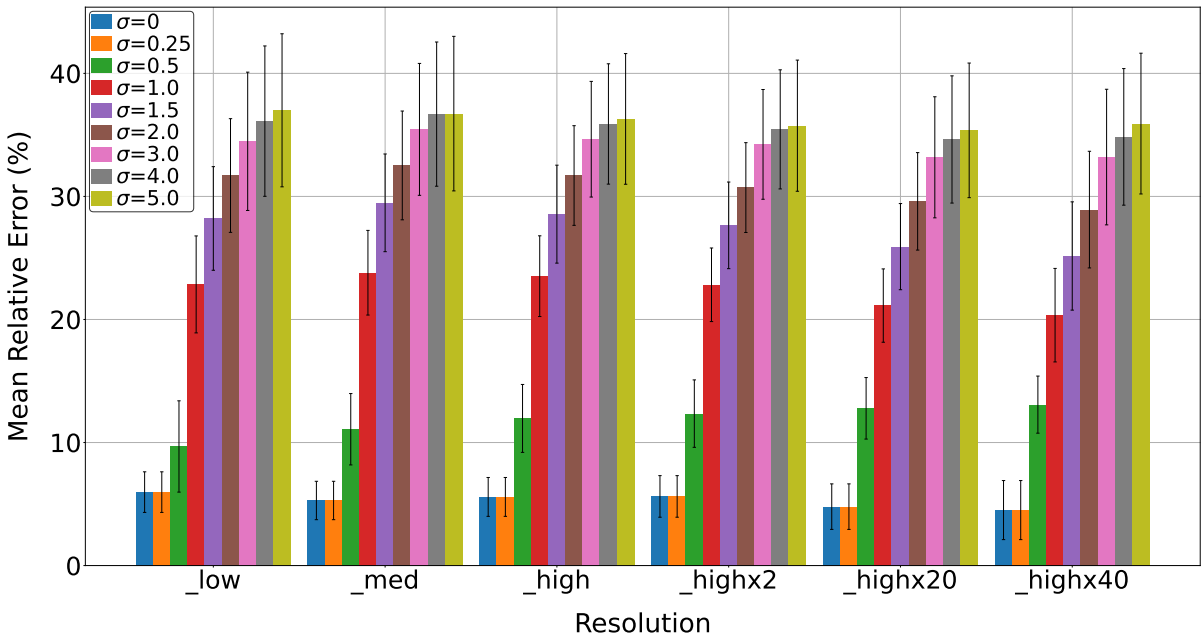


Figure 5. Mean relative error (%) of the GPU_2C implementation under increasing Gaussian blur levels ($\sigma = 0$–5) across image resolutions. For each resolution, bars correspond to the original image (no blur) followed by progressively increasing σ values. Error bars represent the standard deviation.

3.3. Inter-Texture Stability on the Brodatz Dataset

The compression-dimension estimator was applied to 13 texture images from the Brodatz dataset, each subdivided into five non-overlapping tiles. For each texture, the

mean fractal dimension and its dispersion were computed across tiles. Representative results are shown in Figure 6, and the complete numerical values are reported in Supplementary Materials Table S6.

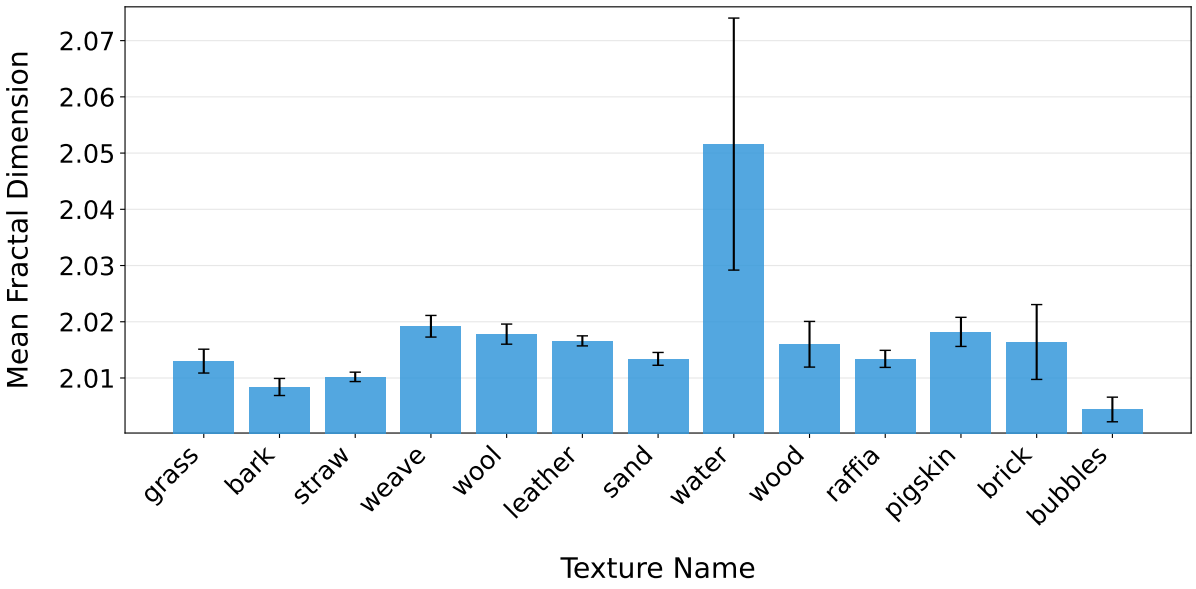


Figure 6. Fractal dimension estimation on selected Brodatz textures. Bars represent mean values across five tiles per image, and error bars indicate standard deviation.

All evaluated textures yielded fractal dimension values greater than 2, indicating that the estimator captures structural characteristics consistent with volumetric texture behavior rather than purely boundary-like patterns. Mean values ranged from approximately 2.01 to 2.05 across the analyzed samples.

Intra-texture variability remained low in all cases. Most textures exhibited standard deviations on the order of 10^{-3}, with coefficients of variation below 0.2%. Even the texture with the largest dispersion showed a coefficient of variation close to 1%, indicating stable behavior across spatially separated regions of the same pattern.

Overall, the results demonstrate consistent intra-texture estimation and clear differentiation between distinct texture classes.

3.4. Computational Performance Analysis

3.4.1. Execution Time on the Julia Dataset

Execution time was used as a practical metric to evaluate the computational efficiency of the different implementations. The total runtime of each configuration, averaged over all images and repeated runs, is summarized in Figure 7, showing the performance of each implementation across resolutions. Error bars indicate the standard deviation across repetitions.

GPU-based implementations exhibit competitive execution times across all resolutions, reflecting the benefits of parallelized resizing and compression. The introduction of a second compression stage in GPU_2C and CPU_2C introduces a measurable but moderate overhead relative to the total runtime. The BC implementation shows higher execution times compared to GPU-based pipelines, primarily due to repeated structural operations across multiple scales and its role as a non-optimized geometric reference implementation.

Time variability across runs remains limited, as indicated by the relatively small standard deviations. Averaging over multiple repetitions and the use of a warm-up phase ensure stable and reproducible timing measurements. A detailed decomposition of execution times across functional blocks is provided in Supplementary Materials Tables S7–S9.

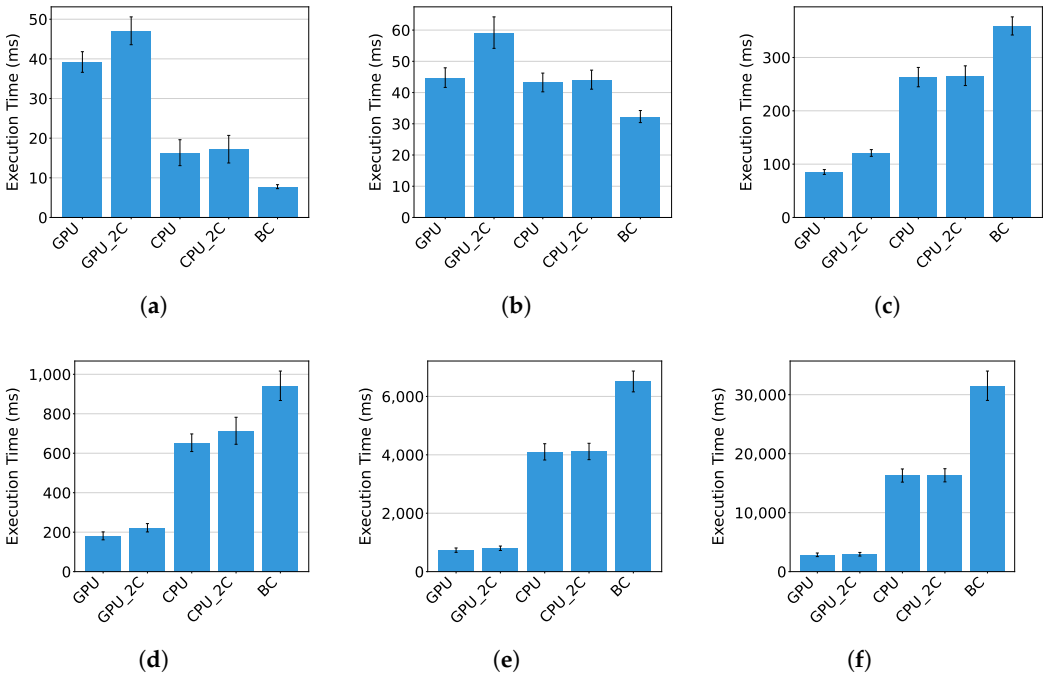


Figure 7. Mean execution time (ms) for all evaluated implementations at different input resolutions. Panels correspond to the follow resolutions: (**a**) _low, (**b**) _med, (**c**) _high, (**d**) _highx2, (**e**) _highx20, and (**f**) _highx40. Bars indicate the average runtime per image over repeated executions, and error bars represent the standard deviation. Results correspond to the SCAYLE environment [48] equipped with NVIDIA H100 GPU boards.

3.4.2. Performance Reproducibility on Consumer-Grade GPU

In addition to the high-throughput computing environment at SCAYLE, the GPU-based pipeline was reproduced on a consumer-grade laptop equipped with an NVIDIA GeForce RTX 4050 Ti GPU. One of the practical advantages of GPU computing lies in the ability to provide high-throughput parallel processing, enabling computational workflows to be migrated from supercomputing infrastructures to more modest hardware setups, provided that the problem size fits within the available resources.

Despite substantial differences in hardware specifications—approximately 3000 CUDA cores and 8 GB of VRAM in the RTX 4050 versus over 16,000 CUDA cores and 80 GB of HBM3 memory in the NVIDIA H100 SXM5, as well as significantly lower power consumption (100 W vs. 700 W), both architectures share comparable compute capabilities (CC 8.9 vs. CC 9.0). The results indicate comparable performance for the specific problem sizes addressed in this work.

Due to the reduced memory capacity of the RTX 4050, a minor implementation adjustment was required to ensure stable execution. In the original H100 setup, GPU memory blocks were released after processing each batch of 96 images. On the lower-memory device, memory fragmentation within the CuPy memory pool led to progressive exhaustion of contiguous VRAM. To address this, explicit synchronization and memory release (`Device().synchronize()` and `free_all_blocks()`) were performed after processing each image, ensuring that memory was physically freed and that each iteration had access to the maximum contiguous allocation available on the device.

3.4.3. Bottleneck Analysis

A detailed inspection of execution times reveals two distinct performance regimes across resolutions. For lower and medium image sizes, the dominant computational cost corresponds to the first GPU compression stage, which remains approximately constant across

resolutions and represents the primary contributor to total runtime. In this regime, resizing and host-to-device transfers have comparatively minor impact on overall performance.

As image resolution increases, CPU-based image loading becomes the dominant bottleneck, exceeding the cost of both GPU compression stages by a large margin. At the highest resolutions, disk I/O and host-side memory handling clearly outweigh GPU computation, limiting the achievable acceleration regardless of the parallel processing capabilities of the device.

These observations explain the moderate differences in total execution time across implementations and highlight that for large problem sizes, the pipeline is primarily constrained by host-side data loading rather than GPU computation.

4. Discussion

4.1. Overview of the Contribution

This work provides a structured computational validation and performance characterization of the compression-dimension estimator [17] under controlled ground-truth conditions. The study systematically evaluates the numerical behavior, statistical stability, and computational cost of the estimator across multiple implementations and resolutions. The analysis combines accuracy assessment against known theoretical dimensions (Julia fractal boundaries), execution-time profiling, architectural bottleneck inspection, cross-device reproducibility, and exploratory structural sensitivity experiments. Together, these elements establish a reproducible evaluation framework for assessing both numerical and computational properties of compression-based fractal dimension estimation.

4.2. Validation on Controlled Planar Fractal Boundaries

The primary validation was conducted using a dataset of Julia sets with known theoretical fractal dimension [18], enabling direct comparison between estimated and reference values. Across 16 images and 6 resolutions, with 30 repeated executions per configuration, the relative error remained below 10% in nearly all cases. No systematic instability was observed across implementations or resolutions.

The two-compression configuration consistently exhibited improved stability, particularly at higher resolutions, indicating that the additional compression stage contributes to reducing variability in the estimation process. While differences between implementations were observable, they remained moderate in magnitude. The GPU-based variants achieved accuracy comparable to CPU-based implementations and to classical box-counting (BC), indicating that acceleration does not alter the statistical behavior of the estimator.

The comparison with BC serves as a geometric baseline within the planar boundary framework considered here. Since the validation dataset consists of planar fractal boundaries, BC with prior binarization represents a natural reference method. The results indicate that the compression-based estimator achieves comparable accuracy across resolutions, with neither method exhibiting systematic dominance across all conditions.

It is worth noting that the Julia images processed in this study are not strictly binary. Due to the intrinsic grayscale variations associated with convergence speed in fractal generation, intensity levels are inherently present. Despite this, the compression-dimension estimator consistently produces values between 1 and 2, consistent with boundary-like fractal behavior. This confirms that the method captures structural scaling characteristics even in the presence of grayscale variations.

Observed differences between CPU and GPU implementations are attributable to internal differences between the standard Zstd compressor [46] and the GPU-based nvCOMP counterpart [38], rather than to conceptual differences in the estimator itself. Importantly, the acceleration process does not introduce systematic bias.

4.3. Computational Characterization and Performance Regimes

Beyond accuracy, a detailed analysis of execution times reveals two distinct computational regimes. For low and medium resolutions, the dominant cost corresponds to the first GPU compression stage, which remains approximately constant across resolutions and constitutes the primary contributor to total runtime. In this regime, resizing and host-to-device transfers have limited impact on overall performance.

As resolution increases, however, CPU-based image loading and host-side memory handling become the dominant bottlenecks. At the highest resolutions, disk I/O and memory operations clearly exceed GPU computation time, limiting achievable acceleration independently of device parallelism. These findings explain the moderate differences observed between implementations at large scales and highlight that the pipeline is ultimately constrained by host-side data movement rather than GPU arithmetic capacity.

Time variability across repeated runs remains limited, supported by the use of a warm-up phase and averaging over multiple executions. This confirms that the reported measurements reflect stable and reproducible performance characteristics.

4.4. Portability Across GPU Architectures

To assess architectural portability, the GPU-based pipeline was reproduced on a consumer-grade system equipped with an NVIDIA GeForce RTX 4050 Ti GPU, in addition to the high-performance NVIDIA H100 environment. Despite substantial differences in hardware specifications—including core count, memory bandwidth, and available VRAM—both devices share similar computation capabilities.

For the images evaluated in this study, comparable performance trends were observed across both architectures. The primary limitation on the lower-memory device was VRAM capacity rather than computation capability. Memory fragmentation within the CuPy memory pool required explicit synchronization and block release to ensure stable allocation across iterations. Once addressed, the pipeline executed consistently across runs, indicating that the implementation does not depend on supercomputing-class hardware for practical deployment.

4.5. Sensitivity to Structural Degradation (Blur Study)

The blur experiment provides insight into the estimator's response to progressive structural smoothing. Increasing the Gaussian blur parameter σ produces a consistent monotonic increase in relative error across all evaluated resolutions. Minimal smoothing ($\sigma = 0.25$) yields results indistinguishable from the original images, indicating that small perturbations do not significantly alter the captured multiscale structure.

From $\sigma = 0.5$ onward, both the mean error and dispersion increase progressively, reaching a plateau at higher blur levels. This behavior is systematically observed across resolutions and reflects the estimator's sensitivity to fine-scale structural information. As Gaussian smoothing attenuates high-frequency components and reduces local structural redundancy, the compression-based scaling relationship is correspondingly affected. These results demonstrate coherent structural sensitivity under controlled perturbations.

4.6. Behavior on Grayscale Textures (Brodatz Study)

An exploratory evaluation was conducted using 13 grayscale textures from the Brodatz dataset [19], each subdivided into five non-overlapping 512×512 tiles. The compression-dimension estimator with two compression stages was applied independently to each tile, and mean and dispersion were computed per texture.

All evaluated textures yielded fractal dimension values greater than 2, with mean values ranging approximately between 2.01 and 2.05. This indicates that the estimator

captures structural characteristics consistent with volumetric texture behavior rather than purely boundary-like patterns.

Intra-texture variability remained very low across all cases. Most textures exhibited standard deviations on the order of 10^{-3}, with coefficients of variation below 0.2%. Even the highest observed dispersion remained close to 1%. This demonstrates strong spatial consistency within homogeneous texture regions and measurable differentiation between structurally distinct texture classes.

Together with the Julia boundary results, these findings suggest that the compression-dimension estimator responds coherently to different structural regimes. Images exhibiting boundary-dominated patterns yield dimensions between 1 and 2, while grayscale textures with distributed structural variation yield values above 2. This behavior emerges without modification of the estimator, reflecting the sensitivity to multiscale structural organization.

4.7. Limitations and Scope

The present approach assumes monofractality and reports a single global dimension value per image. However, many natural structures exhibit multifractal behavior [11,49,50], characterized by local variations in scaling properties across spatial regions. The current framework does not explicitly account for such spatial heterogeneity, and its interpretation should therefore be restricted to global structural characterization within the evaluated conditions.

The observed differences between implementations partially reflect differences in underlying compression libraries. Additionally, performance scaling at very large resolutions is ultimately constrained by host-side data movement rather than by GPU computation resources. These factors define the scope of the present study to computational validation and characterization within the evaluated experimental framework.

4.8. Practical Implications and Future Directions

The results demonstrate that compression-based fractal dimension estimation can be implemented efficiently on GPU hardware while preserving numerical behavior comparable to classical geometric baselines in controlled planar boundary settings. The architectural analysis indicates that moderate consumer-grade GPUs are sufficient for practical workloads, expanding accessibility beyond high-performance computing environments.

Future work may extend the evaluation to broader classes of natural images and investigate formal statistical properties of the estimator. In particular, extending the framework beyond a monofractal assumption toward strategies capable of capturing local scaling variations represents a promising direction. Compression-based methodologies could potentially be adapted to estimate spatially localized complexity measures, enabling multifractal-like analyses within a computationally efficient pipeline.

Broader validation across diverse application domains, as well as architectural improvements aimed at reducing host-side bottlenecks, would further improve the practical relevance of the approach. Deployment in real-time, embedded, or distributed environments also represents a natural continuation of the present study, especially in scenarios where structural complexity must be assessed under computational constraints.

5. Conclusions

This study provides a structured computational validation of the compression-dimension estimator [17] under controlled ground-truth conditions. Using a dataset of Julia fractal boundaries with known theoretical dimensions, the estimator was evaluated in terms of numerical accuracy, statistical stability, and execution time across multiple implementations. The results indicate that calculating the fractal dimension of digital

images through GPU-accelerated compression is feasible and computationally efficient. The achieved accuracy is comparable to classical box-counting within the evaluated planar boundary framework.

The proposed pipeline enables direct processing of grayscale images without prior binarization, which represents a methodological advantage over classical geometric approaches such as box-counting, where binarization is a mandatory preprocessing step. The framework is based on average-pooling resizing and lossless Zstd compression [45], and yields robust and scalable behavior, particularly for medium- and high-resolution synthetic fractals such as Julia sets.

Although the box-counting baseline achieves the lowest mean error in several configurations, the higher computational cost and variability limit its suitability for high-throughput scenarios. In contrast, the GPU implementation with two compression stages offers a balanced trade-off between accuracy, stability, and execution time, especially at higher resolutions, with only moderate computational overhead.

Beyond accuracy, the detailed performance analysis clarifies the architectural factors governing scalability, identifying distinct computational regimes and host-side bottlenecks. Reproducibility across both high-performance and consumer-grade GPUs further supports the practical portability of the approach.

Overall, this work establishes a reproducible evaluation framework for compression-based fractal dimension estimation, combining numerical validation with computational insight and demonstrating its potential integration into real-time and large-scale image analysis systems.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/fractalfract10030174/s1, Table S1: GPU (single compression): Relative error statistics across resolutions. See Section S1 for details. Table S2: GPU (double compression): Relative error statistics across resolutions. See Section S1 for details. Table S3: CPU (single compression): Relative error statistics across resolutions. See Section S1 for details. Table S4: CPU (double compression): Relative error statistics across resolutions. See Section S1 for details.Table S5: Box Counting: Relative error statistics across resolutions. See Section S1 for details. Table S6: Estimated fractal dimension statistics for Brodatz grayscale textures. Each image was divided into five tiles, and statistics are computed across tiles (see Section S2). Table S7: Time Breakdowns (ms) in GPU Approach. Table S8: Time Breakdowns (ms) in CPU Approach. Table S9: Time Breakdowns (ms) in BC Approach. Listing S1: CUDA kernel implementing GPU-based image resizing via area-based block averaging. Each thread computes one output pixel by averaging the corresponding region in the original image defined by the scaling factors. Listing S2: Python routine performing GPU-based image compression using NVIDIA's nvcomp library (Zstandard backend). The resized image remains in device memory and is compressed directly on the GPU.

Author Contributions: Conceptualization, P.C.-P.; formal analysis, Á.D.-H.; software, Á.D.-H.; writing—original draft preparation, Á.D.-H.; writing—review and editing, P.C.-P.; and supervision, P.C.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Department of Education, Junta de Castilla y León, grant number VA184P24 and FEDER Funds (Reference: CLU-2023-1-05). The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Data Availability Statement: Image files of the Julia sets used in the manuscript are openly available in the Díaz-Herrezuelo, A. and Chamorro-Posada, P. image dataset for "GPU-accelerated fractal compression dimension estimation", UVaDOC Repository, 2026. https://uvadoc.uva.es/handle/10324/81641 (accessed on 20 February 2026).

Acknowledgments: The authors acknowledge the use of the supercomputing infrastructure of the Supercomputing Center of Castilla y León (SCAYLE) for the computational resources and technical support provided during this work.

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

References

1. Mandelbrot, B.B. *The Fractal Geometry of Nature*; W. H. Freeman and Company: New York, NY, USA, 1983.
2. Babič, M.; Šturm, R.; Gălățanu, T.F.; Száva, I.R.; Száva, I. Modeling Porosity Surface of 3D Selective Laser Melting Metal Materials. *Fractal Fract.* **2025**, *9*, 331. [CrossRef]
3. Chen, Y. Equivalent relation between normalized spatial entropy and fractal dimension. *Phys. A Stat. Mech. Its Appl.* **2020**, *553*, 124627. [CrossRef]
4. Sharma, N.; Nawn, D.; Pratiher, S.; Shome, S.; Chatterjee, R.; Biswas, K.; Pal, M.; Paul, R.R.; Dutta, S.; Chatterjee, J. Multifractal Texture Analysis of Salivary Fern Pattern for Oral Pre-Cancers and Cancer Assessment. *IEEE Sens. J.* **2021**, *21*, 9333–9340. [CrossRef]
5. Plotze, R.d.O.; Falvo, M.; Pádua, J.G.; Bernacci, L.C.; Vieira, M.L.C.; Oliveira, G.C.X.; Bruno, O.M. Leaf shape analysis using the multiscale Minkowski fractal dimension, a new morphometric method: A study with Passiflora (Passifloraceae). *Can. J. Bot.* **2005**, *83*, 287–301. [CrossRef]
6. Forsythe, A.; Nadal, M.; Sheehy, N.; Cela-Conde, C.J.; Sawey, M. Predicting beauty: Fractal dimension and visual complexity in art. *Br. J. Psychol.* **2011**, *102*, 49–70. [CrossRef] [PubMed]
7. Atupelage, C.; Nagahashi, H.; Yamaguchi, M.; Abe, T.; Hashiguchi, A.; Sakamoto, M. Classification of prostate histopathology images based on multifractal analysis. *IEICE Trans. Inf. Syst.* **2012**, *95*, 3037–3045. [CrossRef]
8. Pentland, A.P. Fractal-Based Description of Natural Scenes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1984**, *PAMI-6*, 661–674. [CrossRef]
9. Del-Pozo-Velázquez, J.; Chamorro-Posada, P.; Aguiar-Pérez, J.M.; Pérez-Juárez, M.A.; Casaseca-De-La-Higuera, P. Water Detection in Satellite Images Based on Fractal Dimension. *Fractal Fract.* **2022**, *6*, 657. [CrossRef]
10. Del-Pozo-Velázquez, J.; Aguiar-Pérez, J.M.; Chamorro-Posada, P.; Ángeles Pérez-Juárez, M.; Wang, X.; de-la Higuera, P.C. Smoke detection in images through fractal dimension-based binary classification. *Digit. Signal Process.* **2025**, *166*, 105346. [CrossRef]
11. Lopes, R.; Betrouni, N. Fractal and multifractal analysis: A review. *Med. Image Anal.* **2009**, *13*, 634–649. [CrossRef]
12. Kuikka, J.T. Fractal analysis in medical imaging. *Int. J. Nonlinear Sci. Numer. Simul.* **2002**, *3*, 81–88. [CrossRef]
13. Spehar, B.; Taylor, R.P. Fractals in art and nature: Why do we like them? In *Proceedings of the Human Vision and Electronic Imaging XVIII*; SPIE: Bellingham, WA, USA, 2013; Volume 8651, pp. 298–309.
14. Sarkar, N.; Chaudhuri, B. An Efficient Differential Box Counting Approach to Compute Fractal Dimension of Image. *IEEE Trans. Syst. Man Cybern.* **1994**, *24*, 115–120. [CrossRef]
15. Conci, A.; Proença, C.B. A fractal image analysis system for fabric inspection based on a box-counting method. *Comput. Netw. ISDN Syst.* **1998**, *30*, 1887–1895. [CrossRef]
16. Sun, W.; Xu, G.; Gong, P.; Liang, S. Fractal analysis of remotely sensed images: A review of methods and applications. *Int. J. Remote Sens.* **2006**, *27*, 4963–4990. [CrossRef]
17. Chamorro-Posada, P. A simple method for estimating the fractal dimension from digital images: The compression dimension. *Chaos Solitons Fractals* **2016**, *91*, 562–572. [CrossRef]
18. McMullen, C.T. Hausdorff dimension and conformal dynamics, III: Computation of dimension. *Am. J. Math.* **1998**, *120*, 691–721. [CrossRef]
19. Brodatz, P. *Textures: A Photographic Album for Artists and Designers*; Dover Publications: New York, NY, USA, 1966.
20. Tricot, C. *Curves and Fractal Dimension*; Springer Science & Business Media: Dordrecht, The Netherlands, 1994.
21. Jin, X.; Ong, S.; Jayasooriah. A practical method for estimating fractal dimension. *Pattern Recognit. Lett.* **1995**, *16*, 457–464. [CrossRef]
22. Panigrahy, C.; Seal, A.; Mahato, N.K. Image texture surface analysis using an improved differential box counting based fractal dimension. *Powder Technol.* **2020**, *364*, 276–299. [CrossRef]
23. Panigrahy, C.; Seal, A.; Mahato, N.K. Quantitative texture measurement of gray-scale images: Fractal dimension using an improved differential box counting method. *Measurement* **2019**, *147*, 106859. [CrossRef]
24. Li, J.; Du, Q.; Sun, C. An improved box-counting method for image fractal dimension estimation. *Pattern Recognit.* **2009**, *42*, 2460–2469. [CrossRef]
25. Li, S.; Wang, Z.; Mou, D. Fractal Analysis of Volcanic Rock Image Based on Difference Box-Counting Dimension and Gray-Level Co-Occurrence Matrix: A Case Study in the Liaohe Basin, China. *Fractal Fract.* **2025**, *9*, 99. [CrossRef]
26. Pan, T.; Guo, R.; Yan, Y.; Fu, C.; Lin, R. Texture Feature Analysis of the Microstructure of Cement-Based Materials During Hydration. *Fractal Fract.* **2025**, *9*, 543. [CrossRef]

27. Wang, J.; Yang, G.; Yuan, Y.; Sun, J.; Pu, G. Effects of Scale Parameters and Counting Origins on Box-Counting Fractal Dimension and Engineering Application in Concrete Beam Crack Analysis. *Fractal Fract.* **2025**, *9*, 549. [CrossRef]
28. Malleswar, S.D.; Isoda, Y.; Nakaya, T. Box Height-Independent Differential Bar Cumulation (DBC) for 3D Raster Surface Fractal Dimension Analysis. *J. Geovis. Spat. Anal.* **2025**, *9*, 13. [CrossRef]
29. Jiang, W.; Liu, Y.; Wang, J.; Li, R.; Liu, X.; Zhang, J. Problems of the Grid Size Selection in Differential Box-Counting (DBC) Methods and an Improvement Strategy. *Entropy* **2022**, *24*, 977. [CrossRef] [PubMed]
30. Douglass, R.W. Automated Box-Counting Fractal Dimension Analysis: Sliding Window Optimization and Multi-Fractal Validation. *Fractal Fract.* **2025**, *9*, 633. [CrossRef]
31. Ahammer, H. Higuchi Dimension of Digital Images. *PLoS ONE* **2011**, *6*, e24796. [CrossRef]
32. Higuchi, T. Approach to an irregular time series on the basis of the fractal theory. *Phys. D Nonlinear Phenom.* **1988**, *31*, 277–283. [CrossRef]
33. Barnsley, M.F.; Hurd, L.P. *Fractal Image Compression*; AK Peters, Ltd.: Wellesley, MA, USA, 1993.
34. Peptenatu, D.; Andronache, I.; Ahammer, H.; Taylor, R.; Liritzis, I.; Radulovic, M.; Ciobanu, B.; Burcea, M.; Perc, M.; Pham, T.; et al. Kolmogorov compression complexity may differentiate different schools of Orthodox iconography. *Sci. Rep.* **2022**, *12*, 10743. [CrossRef]
35. Ruiz de Miras, J.; Posadas, M.; Ibáñez-Molina, A.; Soriano, M.; Iglesias-Parro, S. Fast computation of fractal dimension for 2D, 3D and 4D data. *J. Comput. Sci.* **2023**, *66*, 101908. [CrossRef]
36. Ruiz de Miras, J.; Li, Y.; León Salas, A.; Arroyo, G.; Lopez, L.; Torres, J.; Martín, D. Ultra-fast computation of fractal dimension for RGB images. *Pattern Anal. Appl.* **2025**, *28*, 36. [CrossRef]
37. Ruiz de Miras, J.; Casali, A.G.; Massimini, M.; Ibáñez-Molina, A.J.; Soriano, M.F.; Iglesias-Parro, S. FDI: A MATLAB tool for computing the fractal dimension index of sources reconstructed from EEG data. *Comput. Biol. Med.* **2024**, *179*, 108871. [CrossRef]
38. NVIDIA Corporation. nvCOMP—GPU-Accelerated Data Compression Library. 2024. Available online: https://developer.nvidia.com/nvcomp (accessed on 23 May 2025).
39. Chang, Y.C. Deep-Learning Estimators for the Hurst Exponent of Two-Dimensional Fractional Brownian Motion. *Fractal Fract.* **2024**, *8*, 50. [CrossRef]
40. Jimenez, A.D.; Quitl, C.T.; Vallejo, A.A.; Ortega, R.C.; Vivanco, A.P.; Carbone, A. Estimation of Urban Fractal Dimension Using a Convolutional Neural Network. In *Proceedings of the Advances in Soft Computing: 24th Mexican International Conference on Artificial Intelligence, MICAI 2025, Guanajuato, Mexico, 3 November 2025*; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2025; pp. 410–421. [CrossRef]
41. Romero-Morelos, P.; Herrera-López, E.; González-Yebra, B. Development, Application and Utility of a Machine Learning Approach for Melanoma and Non-Melanoma Lesion Classification Using Counting Box Fractal Dimension. *Diagnostics* **2024**, *14*, 1132. [CrossRef]
42. Lee, M.; Lee, S. Box-Counting Dimension Sequences of Level Sets in AI-Generated Fractals. *Fractal Fract.* **2024**, *8*, 730. [CrossRef]
43. Backes, A.R. Upper and lower volumetric fractal descriptors for texture classification. *Pattern Recognit. Lett.* **2017**, *92*, 9–16. [CrossRef]
44. Foroutan-Pour, K.; Dutilleul, P.; Smith, D. Advances in the implementation of the box-counting method of fractal dimension estimation. *Appl. Math. Comput.* **1999**, *105*, 195–210. [CrossRef]
45. Facebook. Zstandard—Fast Real-Time Compression Algorithm. 2025. Available online: http://facebook.github.io/zstd/ (accessed on 23 May 2025).
46. Pakhomov, S. Zstandard: Zstandard Bindings for Python. Python Package, Version 0.22.0. 2025. Available online: https://pypi.org/project/zstandard/ (accessed on 23 May 2025).
47. Python Core Team. *Python: A Dynamic, Open Source Programming Language*; Python Software Foundation: Wilmington, DE, USA, 2019.
48. Supercomputación Castilla y León (SCAYLE). SCAYLE Supercomputing Center. 2026. Available online: https://www.scayle.es/ (accessed on 15 January 2026).
49. Andjelkovic, J.; Zivic, N.; Reljin, B.; Celebic, V.; Salom, I. Application of multifractal analysis on medical images. *Wseas Trans. Inf. Sci. Appl.* **2008**, *5*, 1561–1572.
50. Nilsson, E. *Multifractal-Based Image Analysis with Applications in Medical Imaging*; Department of Computing Science, Umeå University: Umeå, Sweden, 2007; pp. 33–70.

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.