BASKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

MASTER OF SCIENCE IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FRACTAL COMPRESSION METHOD ON IMAGES FROM UNMANNED AERIAL VEHICLES

BY

MEHMET FATİH ÇELİK

MASTER OF SCIENCE THESIS

ANKARA – 2024

BASKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MASTER OF SCIENCE IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FRACTAL COMPRESSION METHOD ON IMAGES FROM UNMANNED AERIAL VEHICLES

BY

MEHMET FATİH ÇELİK

MASTER OF SCIENCE THESIS

ADVISOR

DR. DENİZ KARAÇOR

ANKARA – 2024

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by Mehmet Fatih ÇELİK for the program of Electrical and Electronics Engineering, has been approved in partial fulfillment of the requirements for the degree of Master of Science / Doctor of Philosophy in Electrical and Electronics Engineering Department by the following committee.

Date of Thesis Defense: 19 / 08 / 2024

Thesis Title: Fractal Compression Method on Images From Unmanned Aerial Vehicles

Examining Committee Members	**Signature**
Dr. Deniz KARAÇOR, Başkent University	……..
Assoc. Dr. Selda GÜNEY, Başkent University	……..
Dr. Aykut KALAYCIOĞLU, Ankara University	……..

APPROVAL

PROF. DR. DİLEK ÇÖKELİLER SERDAROĞLU
Director, Institute of Science and Engineering
Date: … / … / ……..

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
YÜKSEK LİSANS / DOKTORA TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 18 / 08 / 2024

Öğrencinin Adı, Soyadı: Mehmet Fatih ÇELİK

Öğrencinin Numarası : 22110041

Anabilim Dalı :Elektrik Elektronik Mühendisliği Anabilim Dalı

Programı : Elektrik Elektronik Mühendisliği Tezli Yüksek Lisans Programı

Danışmanın Unvanı/Adı, Soyadı : Dr. Deniz KARAÇOR

Tez Başlığı : İnsansız Hava Araçlarından Alınan Görüntüler Üzerinde Fraktal Sıkıştırma Yöntemi

Yukarıda başlığı belirtilen Yüksek Lisans/Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 64 sayfalık kısmına ilişkin, 18 / 08 / 2024 tarihinde şahsım/tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 10'dur. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:…….

ONAY

Öğrenci Danışmanı

Dr. Deniz KARAÇOR

Tarih: 18 / 08 / 2024

To my most valuable treasures

KAAN and EKİN

ACKNOWLEDGMENTS

I would like to convey my most meaningful thanks to my thesis advisor Dr. Deniz Karaçor for her support and most of all for the sense of trust she provided in this challenging process. I got to know her thanks to a course she gave and afterwards I could not help thinking that I wish I had taken that course earlier. With her always smiling face and calm conversations, she managed to be by my side even in the most stuck moments. I will not forget especially our thesis interviews. Again, I would like to express my gratitude to him for what he did at the beginning of my academic career.

I would like to thank Prof. Dr. Sedat Nazlıbilek for his endless support and his light that will guide all young people is an issue that cannot be passed without mentioning.

I would like to thank my family for always standing behind me, even when I had to spare time for them. Even if they do not accept it, my every success is thanks to them.

Finally, I would like to express my endless thanks to all my friends who have supported me and especially to my best friend Burçin Çalıkoğlu, who helped me in every aspect of the master's program we started together and contributed greatly to my graduation.

ABSTRACT

MEHMET FATIH CELIK

FRACTAL COMPRESSION METHOD ON IMAGES FROM UNMANNED AERIAL VEHICLES

Institute of Science and Engineering

Department of Electrical and Electronics Engineering

2024

Image compression, which is one of the most industry-leading topics of academic studies in image processing, is a subject that is constantly evolving today and in the future. Past and present studies have been developed to meet the needs such as the best compression ratio, the best time and the best image quality.

Compression methods have their own advantages and disadvantages. Lossy and lossless compression methods can be used for different purposes. One of the lossy compression methods, fractal compression, which is based on iteration theory and is assumed to consist of copies of each other, is a method open to development.

Fractal compression methods and algorithms have been created based on the idea of using fractal features that are frequently encountered in nature and extracting the intrinsic qualities in images.

Unmanned aerial vehicles, which are an indispensable technology of today, are likely to be in nature in terms of their general usage characteristics. In this study, the success of the fractal compression method in images taken from UAVs is tried to reveal the positive and negative aspects.

Keywords: Fractal, UAV, Compression, Image Analysis, DCT, DWT

ÖZET

MEHMET FATIH CELIK

İNSANSIZ HAVA ARAÇLARINDAN ALINAN GÖRÜNTÜLER ÜZERİNDE FRAKTAL SIKIŞTIRMA YÖNTEMİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı
2024

Görüntü işleme konularında akademik çalışmaların sektöre en çok yön veren konularından bir tanesi olan görüntü sıkıştırma, günümüzde ve gelecekte sürekli gelişen bir konu olarak karşımıza çıkmaktadır. Dünden bugüne yapılan çalışmalar en iyi sıkıştırma oranı, en iyi zaman ve en iyi görüntü kalitesi gibi ihtiyaçları karşılamak üzere geliştirilmiştir.

Sıkıştırma yöntemlerinin kendi içlerinde avantaj ve dezavantajları mevcuttur. Kayıplı ve kayıpsız sıkıştırma yöntemleri farklı amaçlar için kullanılabilmektedir. Kayıplı sıkıştırma yöntemlerinden biri olan ve birbirinin kopyalarından oluştuğu varsayılan ve yineleme teorisine dayana fraktal sıkıştırma yöntemi de gelişime açık bir yöntem olarak karşımıza çıkmaktadır.

Doğada sıkılıkla karşımıza çıkan fraktal özelliklerin kullanımı ve görüntülerdeki öz niteliklerin çıkartılması fikrine dayanarak fraktal sıkıştırma yöntemleri ve algoritmaları oluşturulmuştur.

Gündüzümün vazgeçilmez bir teknolojisi olan insansız hava araçları ise genel kullanım özellikleri bakımından doğa içerisinde olmaları muhtemeldir. Yapılan çalışmada İHA'lardan alınan görüntülerdeki fraktal sıkıştırma yönteminin başarımı olumlu ve olumsuz yönleri ortaya çıkartılmaya çalışılmıştır.

Anahtar Kelimeler: Fraktal, Sıkıştırma, İHA, Görüntü Analizi, DCT, DWT

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF ABBREVIATIONS

CC	Correlation Coefficient
CPU	Central Proccessing Unit
CR	Compression Ratio
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FPGA	Field Programmable Gate Array
FRC	Fractal
GHz	Giga Hertz
GPU	Graphical Processing Unit
IFS	Iterated Function Systems
JPEG	Joint Photographic Experts Group
KB	Kilo Byte
LZW	Lempel-Ziv-Welch
MAD	Mean Absolute Difference
MAE	Mean Absolute Error
Mbps	Mega Bit Per Second
MSE	Mean Squared Error
NCPR	Normalized Cross-Correlation Peak to Ratio
PIFS	Partitioned Iterated Function Systems
PSNR	Peak Signal-to-Noise Ratio
RAM	Random Acces Memory
RGB	Red Green Blue
RLE	Run-Length Encoding
RMSE	Root Mean Square Error
SC	Structural Content
SSIM	Structural Similarity Index Measure
VSI	Visual Saliency-Induced Index
VTOL	Vertical Take Off
W	Watt
WLT	Wavelet

1. INTRODUCTION

As technology, data science and artificial intelligence have progressed, the problems of storing, retrieving, extracting, extracting and sorting, transmitting and maintaining the quality of vast amounts of data have arisen [1]. Storing and transmitting image files, especially those containing large data, presents significant challenges due to limitations in bandwidth, storage capacity and write speeds [2], [3]. Image compression is a solution to overcome these challenges. Because these techniques help to reduce the required resources for processing, transmission and storage [4]. In this case, fractal image compression techniques are notable for their potential to reduce the image data footprint under certain conditions, low quality loss and high compression ratios.

Especially in next-generation applications such as Unmanned Aerial Vehicles (UAVs), where high technology is mandatory, image compression plays a much more critical role. [3]. UAVs have the potential to be used in many applications in our daily lives. So far, their use has become widespread in military and civilian areas, urban planning, search and rescue operations, traffic monitoring, agricultural applications, environmental surveillance and mapping activities as they provide great convenience [5]. Although the applications vary, it is an inevitable result that the data is stored, transferred to the control center and displayed to the user in real time [6]. Reducing the use of limited resources during these operations is of great importance for the success of the application. Image compression techniques are one of the most important elements that strengthen our hand in these issues. Although the development of fast data processing and transmission technologies reduces the problem, the increase in the quantity and quality of the data used necessitates the use of data compression techniques.

Fractal compression stands out here for its capacity to compress data, primarily image data, using repetitive features to a large extent [7], [8], [9]. This method, which uses the fractal qualities of the mathematical forms of images to fit into a storage space well below the original image size, offers very high compression rates, especially for images with repetitive areas [7], [10]. In UAV applications, fractal qualities are encountered quite intensively in areas such as forests, meadows, urban and traffic areas where the details of the earth are similar to each other.

This thesis aims to investigate the performance of fractal image compression technique, its theoretical foundations, and its comparison with existing methods on real and unique data, since the data in UAV applications have the characteristics described above.

Finally, fractal image compression plays an important role in making data processing pathways more effective in the development and operation of advanced technology platforms such as UAVs. In addition, this work will be a resource for the academic community as a source of knowledge on fractal image compression.

This thesis examines the outcomes of fractal compression, hypothesizing that images captured by unmanned aerial vehicles exhibit significant similarity. Various transformations and inversions applied in fractal compression will be evaluated, and their results compared.

1.1. Importance of Image Compression

Image compression is a technology that reduces the data processing resources required to store and transfer digitized images to different media. This technology provides performance improvements by reducing the storage footprint and bandwidth utilization of digital images and video files consisting of sequentially displayed digital images. Image compression techniques play a critical role, especially when it comes to processing and transmitting images with high data capacity due to high resolution [4]. The importance of image compression, while needed in many different applications, is most evident for systems such as Unmanned Aerial Vehicles (UAVs), which have limited resources and whose performance criterion depends on the best use of these resources [11].

1.2. Importance of Image Compression in UAV Applications

1.2.1. Data transmission speed and efficiency

UAVs are systems that are usually remotely controlled by a user or used for autonomous missions. These vehicles are used for observation, surveillance, reconnaissance and search and rescue purposes and can monitor and image very large areas. The images provided by UAVs are transmitted to the user in real time or to the control center via wireless communication methods. Image compression methods allow for real-time or quasi-real-time data streaming, whether the images are sent to the user or transmitted to the data center for storage. There is an inverse relationship between the size reduction of raw images through compression and the bandwidth used. As the compression ratio increases, the bandwidth

used decreases. This avoids the energy resources needed for data transmission and potential communication problems [11].

1.2.2. Optimization of bandwidth utilization

Bandwidth is used to express the data transmission capacity of a transmission medium or communication channel. In other words, the signal with the maximum frequency that can be carried on a channel is the bandwidth of the channel. The greater the bandwidth, the greater the volume of data that can be transmitted in a given period of time.

In data transmission; bandwidth or digital bandwidth is the measurement of the data rate in bits per second in data communication sources that are used or can be used [6].

Raw images captured by high-resolution cameras used in UAVs have large data sizes. For example, the resolution of an image from a 1080p Full HD camera is 1920 x 1080 pixels. Assuming that the color depth of this image is 8 bits per channel, the total raw image size will be 49,766,400 bits, which is approximately 6 MB in size excluding the title and exif information. With the advancement of technology, camera resolutions have reached 7680 x 4320 and color depths have reached 32-bit levels. As a result, the storage footprint of the raw image has increased 64 times.

Since UAVs generally have low-power data transmission systems, they have to communicate data using a limited bandwidth. For example, Iwave's FD-61MN model numbered data link can transmit 30Mbps data while consuming 8W power [12]. This data transmission refers to the total bandwidth from bottom to top and top to bottom.

Image compression allows the transmission of the image data to be transmitted in accordance with the bandwidth characteristics. Reducing the need for bandwidth helps to communicate both more stably and over longer distances [3].

1.2.3. Optimization of storage capacity

Depending on their mission and usage scenarios, UAVs locally store the data collected during flight on their own recording systems. The JPEG method, which is widely used for compression, reduces the size of an image from 6 MB to 124 KB, as shown in Figure 1.1. This allows UAVs to take up less space in their limited storage devices. This allows more data to be stored in the same storage space and helps to optimize capacity utilization for long-term missions [13].

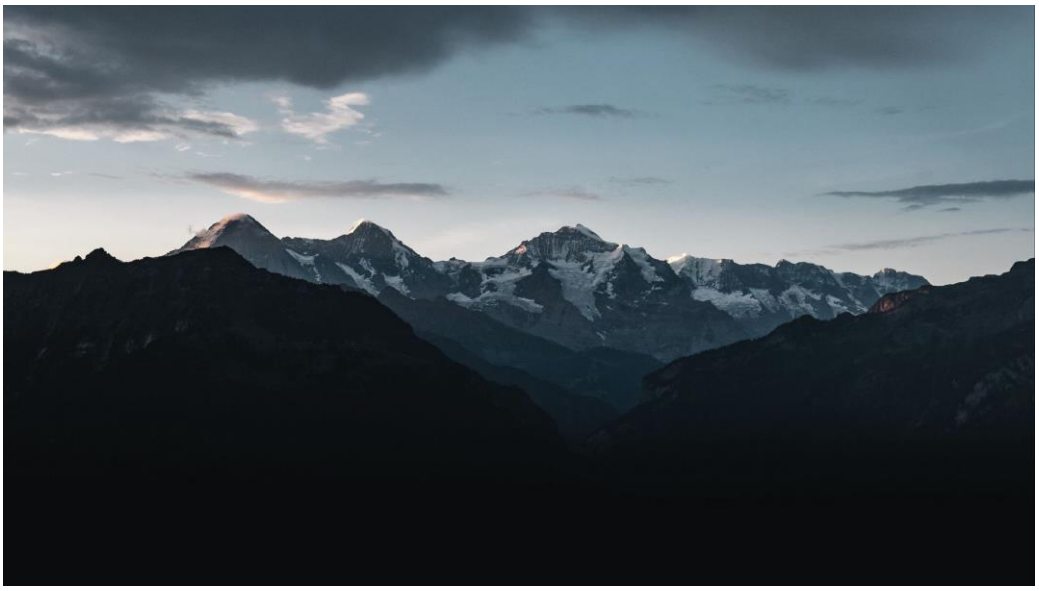

Figure 1.1. 1920 x 1080 Pixel Image Example

1.2.4. Energy efficiency

For UAVs, energy consumption is a crucial factor to improve performance, especially in long-duration missions [14]. For longer tenure, all efficiency needs must be met.

Data compression reduces the energy required for data transmission and storage. If the time required for self-storage is calculated in terms of the time taken to write each bit, it is similar to the calculation for storage. In addition, in terms of reducing the amount of energy used for data communication, the reduction in bandwidth requirements and transmission time also reduces the amount of energy consumed [11], [14]. Compression methods optimize the UAV's energy consumption and allow it to achieve longer flight times.

On the other hand, inappropriate compression methods also lead to increased data consumption and losses.

1.2.5. Advanced analysis and processing

As mentioned above, compressed images require less processing power, resources and bandwidth, so they can be processed and analyzed faster in the tasks to be performed. In this way, it enables fast and effective information extraction from the images taken from UAVs. It provides a great advantage especially in applications such as military operations, emergency action plans and disaster management where quick decisions need to be made [6].

In summary, the use of image compression has become a cornerstone of technology in all areas where UAVs are used. Its use is essential for UAVs to fulfill their missions efficiently and effectively, even in harsh conditions.

1.3. Historical Development of Fractal Compression and Basic Concepts

Fractal compression is an image compression method based on the theory of fractals, a mathematical concept that exploits the ability to represent complex geometric shapes with simple codes. This method was developed in the late 1980s by Michael Barnsley and Alan Sloan [15, p. 351]. Fractals are defined as geometric shapes that repeat infinitely and contain transformations. This quality of fractals provides an ideal basis for image compression, where large sets of data can be characterized by smaller, repeating structures.

Fractal compression has its roots in Benoit Mandelbrot's work on fractals. Mandelbrot showed that complex structures in nature can in fact be modeled by mathematical fractal structures [16]. In the 1970s and 1980s, Mandelbrot's work and books made fractal geometry more widely recognized. Building on Mandelbrot's work, Michael Barnsley worked on using fractal geometry for digital image compression [17].

Barnsley and Sloan developed a method for compressing images using mathematical modeling called "Iterated Function Systems" (IFS) [18]. This technique made it possible to model smaller parts of an image as self-similar parts and to reproduce these parts using mathematical functions.

The basic working principle of fractal compression is based on the idea that each part of an image is "self-similar" to another part of the image. By applying a specific set of transformations (inversion, scaling, mirroring) to each fragment, it allows the identification of different fragments within the image. The original size of the image becomes almost irrelevant for this method thanks to the use of repetitive structures. This means that even very large images can be compressed with a very low error rate.

Another important aspect of fractal compression is that it can achieve extremely high levels of compression. However, high compression is not without its drawbacks, such as high computational power and compression time. Optimization methods may need to balance between compression ratio and resource utilization.

2. THEORY

2.1. Fractal

Fractals are defined as structures that repeat themselves with certain transformations. These structures can continue to mathematically infinitesimal sizes, maintaining a specific pattern in each dimension [19]. Fractals, unlike classical geometric shapes, can have fractional dimensions. This allows them to be used to model complex natural systems [20].

Fractals, with their complex and self-similar parts, are an important mathematical structure both in theory and in practice. Fractal geometry provides a better understanding of the seemingly complex parts and processes found in nature and allows for innovative solutions in various scientific disciplines.

2.2. Mathematical Basis

Fractal compression theory is based on expressing high-dimensional images in simpler fractal codes. This chapter explores the mathematical structures and basic concepts that make fractal compression feasible.

Basic mathematical principles used in fractal compression, including concepts such as similarity transforms, dimensioning terms and iteration [18]. Each part of the image is subjected to a certain amount of scaling, rotation and displacement to achieve the best fractal convergence. Different methods can also be used to measure convergence. This process is applied iteratively on the image to get a better convergence to the original image.

Fractal compression theory is recognized as an innovative and promising approach in computer graphics and image processing, characterized by high compression ratios and high quality of reconstructed images [21].

2.2.1. Self-similarity

The most basic characteristic of a fractal is that it repeats itself as similar structures. This means that any part of the fractal is a copy of the whole, scaled down to different sizes within it. Mathematically, this property is usually expressed using a recurrence function [22]. Mathematically, the self-similarity for a fractal $(f(x))$ can be expressed as follows:

$$[f(x) = f(ax) + b] \tag{2.1}$$

In this formula (a) and (b) are scale and displacement parameters.

2.2.2. Fractional dimension

Fractals, unlike classical geometry, have fractional dimensions. This dimension determines the complexity of the fractal and the space it fills within the whole. The fractal dimension can be calculated by various methods, such as the box counting method or the Hausdorff dimension [23].

Hausdorff dimension is a concept that measures the complexity or size of a space. Unlike Euclidean dimensions, this dimension can also be defined for fractal geometries.

In order to calculate the Hausdorff dimension of a cluster, the cluster is covered with balls with an initial value. The diameters of these balls are gradually reduced and the coverage is continued.

The Hausdorff dimension (D) is the most widely used fractal dimension measure and is calculated by the formula:

$$[D = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}] \tag{2.2}$$

In this formula, $N(\epsilon)$, is the minimum number of parts of the fractal that must be covered by the scale (ϵ).

2.2.3. Iteration

Fractals are usually created by iterative processes. In these processes, an initial shape (usually a line or point) is subjected to repeated transformations. Each iteration increases the complexity of the shape and after a certain number of iterations the fractal structure emerges [24].

The Mandelbrot set is defined in the complex number plane by a certain iterative process [25]. An example of a Mandelbrot set is given in Figure 2.1. A complex number (c), is determined whether it belongs to the Mandelbrot set by the following iterations.

$$[z_{n+1} = z_n^2 + c] \tag{2.3}$$

Here it starts as$(z_0 = 0)$. If $(|z_n|)$ remains bounded in infinitesimal iterations, then (c) is part of the Mandelbrot set.

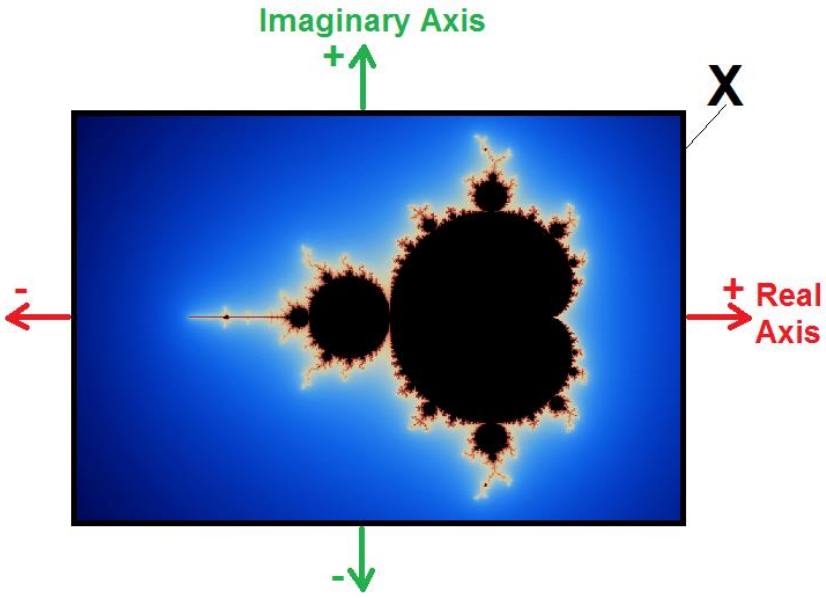

Figure 2.1. Mandelbrot Cluster Example

Julia sets are defined similarly to the Mandelbrot set, but different patterns are created using a fixed value of (c) [26].

$$[z_{n+1} = z_n^2 + c] \tag{2.4}$$

Julia sets represent the values of (z_0) that remain bounded at the end of the iteration for a fixed value of (c).

The Sierpinski triangle is formed by repeatedly subtracting the subdivisions of a triangle. Initially, an equilateral triangle is taken and the middle sub triangle is subtracted. When this process is repeated infinitely, the Sierpinski triangle given by Figure 2.2 is obtained [27]. This process can be described by the following iterative rule.

a) Draw a triangle.

b) Divide the triangle into four equal sub-triangles.

c) Subtract the middle sub-triangle.

d) Repeat for the remaining triangles.

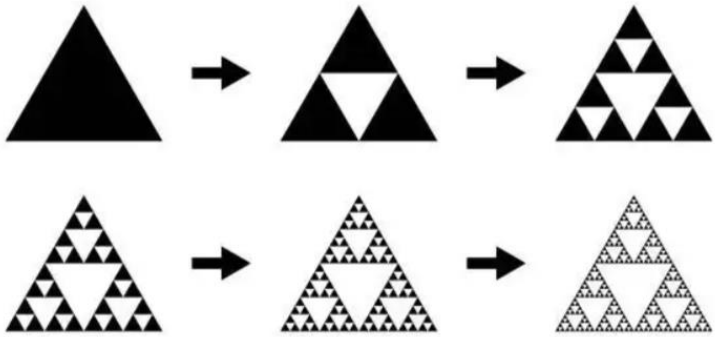

Figure 2.2. Sierpinski Triangle

The Koch snowflake is formed by repeatedly dividing a line segment into triangles. Initially, a line segment is taken and divided into three pieces at a time, then the middle piece is replaced by two sides of an equilateral triangle. Repeating this process infinitely results in the Koch snowflake, which is also illustrated in Figure 2.3 [28]. This process can be described as follows:

a) Draw a line segment.

b) Divide the line segment into three equal parts.

c) Replace the middle segment with two sides of an equilateral triangle.

d) Repeat the same process for each new segment.

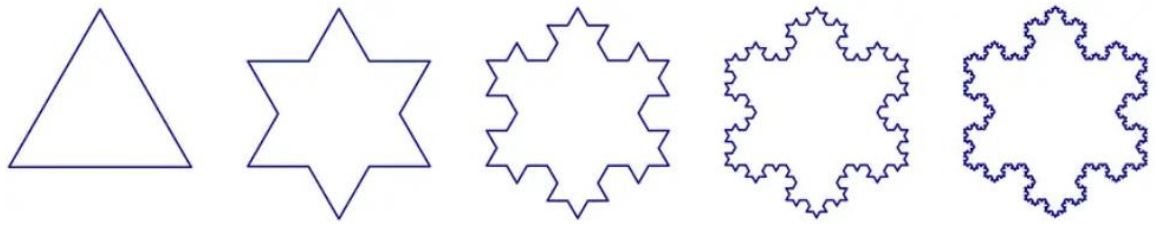

Figure 2.3. Koch Snowflake

2.2.4. Recursive function systems

Fractal geometry is used to describe self-repeating structures that show similar properties at any scale. Fractal compression is based on the principle of compressing images by identifying these repeating structures. Iterated Function Systems (IFS) are used to model an image or shape with mathematical functions through these repetitions [18], [29].

Metric Space $((X))$ is the space where the fractal structure is defined. Usually, two-dimensional Euclidean space $((R^2))$ is used.

Contraction Functions $((w_i))$ each (w_i), is a contraction function applied to a subset of the metric space. These functions create a new shape by shrinking and moving a certain part of the space. A contraction function satisfies the following condition:

$$[d\big(w_i(x), w_i(y)\big) \leq s_i \cdot d(x, y) \quad (\text{for all } x, y \in X \text{ and } 0 \leq s_i < 1)] \qquad (2.5)$$

Here (d), presents the metric (distance) function and (s_i), represents the contraction rate.

The Set of Functions$(\{w_1, w_2, \dots, w_n\})$, forms the entire IFS. Each function is used to create a part of the fractal [18].

IFS is based on the Banach Fixed Point Theorem. This theorem guarantees that the iterative application of contraction functions will lead to a fixed point (or a fixed set) [30]. IFS uses this fixed point as a fractal.

The Fractal Creation Process with IFS proceeds as follows.

a) An initial shape (usually a point or a line) is selected.

b) Contraction functions are applied sequentially to the selected initial shape. At each iteration, the shape shrinks and a new pattern is formed.

c) The fractal structure is formed by repeatedly applying the functions. With each iteration, the shape becomes more complex and self-similar.

The Sierpinski Triangle can be easily created using IFS. This fractal is defined using three contraction functions. Each function shrinks and moves the initial triangle into one of three sub-triangles.

Contraction functions are:

a. $(w_1(x, y) = \left(\frac{x}{2}, \frac{y}{2}\right))$

b. $(w_2(x,y) = \left(\frac{x}{2} + \frac{1}{2}, \frac{y}{2}\right))$

c. $(w_3(x,y) = \left(\frac{x}{2} + \frac{1}{4}, \frac{y}{2} + \frac{\sqrt{3}}{4}\right))$

When these functions are applied repeatedly, the Sierpinski triangle is formed.

2.2.5. Similarity transformations

Similarity transformations are operations that involve scaling, rotating, mirroring and displacing an object [31]. Fractal compression uses these transformations to reconstruct a large part of an image using smaller parts. The basic mathematical expression is based on finding the optimal similarity transformation between different parts of an image [32], [33].

A similarity transformation (S), is usually defined by the following formula:

$$[S(x,y) = \left(sx\cos\theta - sy\sin\theta + t_x, sx\sin\theta + sy\cos\theta + t_y\right)] \qquad (2.6)$$

Here:

$((x,y))$: Coordinates of the point to be transformed.

(s): Scaling factor (0 < s < 1).

(θ): Rotation angle.

(t_x) and (t_y): Displacement components (in x and y direction).

2.2.6. Attractor and Hutchinson operator

In fractal compression, the attractor is the fixed point reached after iteration [34]. The Hutchinson operator is a tool used to identify these attractors and defines how to reconstruct the image by applying a specific set of rules [35].

The fractal itself produced by an Iterated Function System (IFS) is called the attractor of the system. The attractor is a fixed fractal structure obtained by repeated application of the contraction functions of the IFS, achieved by finite or infinite iterations [36].

a) Self-Similarity: The attractor is characterized by the emergence of similar structures in each iteration in which the IFS is applied.

b) Fixed Point: The attractor is the fixed point of the IFS functions, i.e., the point to be reached at the end of the iterations.

c) Equilibrated State: When iterations continue indefinitely, the attractor is reached and further iterations do not change the attractor.

The Hutchinson operator is a mathematical tool used to describe the attractor of an IFS. The Hutchinson operator represents the union of all contraction functions in the IFS and the fixed point of this operator constitutes the attractor of the IFS [37].

An IFS is defined as a set of contraction functions $(\{w_1, w_2, \dots, w_n\})$. The Hutchinson operator $((W))$, is defined as the union of these functions:

$$[W(A) = \bigcup_{i=1}^{n} w_i(A)] \tag{2.7}$$

Here (A), is a subset of the space. The fixed point of the Hutchinson operator (A^*) satisfies the following condition:

$$[W(A^*) = A^*] \tag{2.8}$$

This shows that (A^*) is the attractor of the IFS.

The properties of the Hutchinson operator can be listed as follows:

a. Contraction Property: The Hutchinson operator has contraction properties, i.e., it reaches a certain fixed point in iterations.

b. Fixed Point Theorem: According to the Banach Fixed Point Theorem, iterative application of the Hutchinson operator guarantees reaching a fixed point (attractor).

c. Union of Sets: The Hutchinson operator expresses the union of sets resulting from the application of all functions in the IFS.

The IFS functions and the Hutchinson operator for the Sierpinski triangle are as follows:

Functions:

$$w_1(x,y) = \left(\frac{x}{2}, \frac{y}{2}\right)$$
$$w_2(x,y) = \left(\frac{x}{2} + \frac{1}{2}, \frac{y}{2}\right)$$
$$w_3(x,y) = \left(\frac{x}{2} + \frac{1}{4}, \frac{y}{2} + \frac{\sqrt{3}}{4}\right)$$

Hutchinson Operator:

$$[W(A) = w_1(A) \cup w_2(A) \cup w_3(A)]$$

Iterative application of this operator leads to the Sierpinski triangle attractor.
The Barnsley Fern is defined using four different contraction functions:

Functions:

$$w_1(x,y) = (0.85x + 0.04y, -0.04x + 0.85y + 1.6)$$
$$w_2(x,y) = (0.2x - 0.26y, 0.23x + 0.22y + 1.6)$$
$$w_3(x,y) = (-0.15x + 0.28y, 0.26x + 0.24y + 0.44)$$
$$w_4(x,y) = (0, 0.16y)$$

Hutchinson Operator:

$$[W(A) = w_1(A) \cup w_2(A) \cup w_3(A) \cup w_4(A)]$$

Iterative application of this operator leads to the Barnsley Fern attractor.

If the steps defined below are applied, the Barnsley Fern given in Figure 2.4 is obtained. The Hutchinson operator is used in the fractal creation process in the following steps:

a. Initial Set Selection: Initially a cluster $((A_0))$ is selected (usually a single point is used initially).

b. b. Iterations: The Hutchinson operator is iteratively applied to the initial set:

$$[A_{n+1} = W(A_n)]$$

c. Conclusion: When iterations continue to infinity, the sets (A_n) converge to the attractor $((A^*))$.

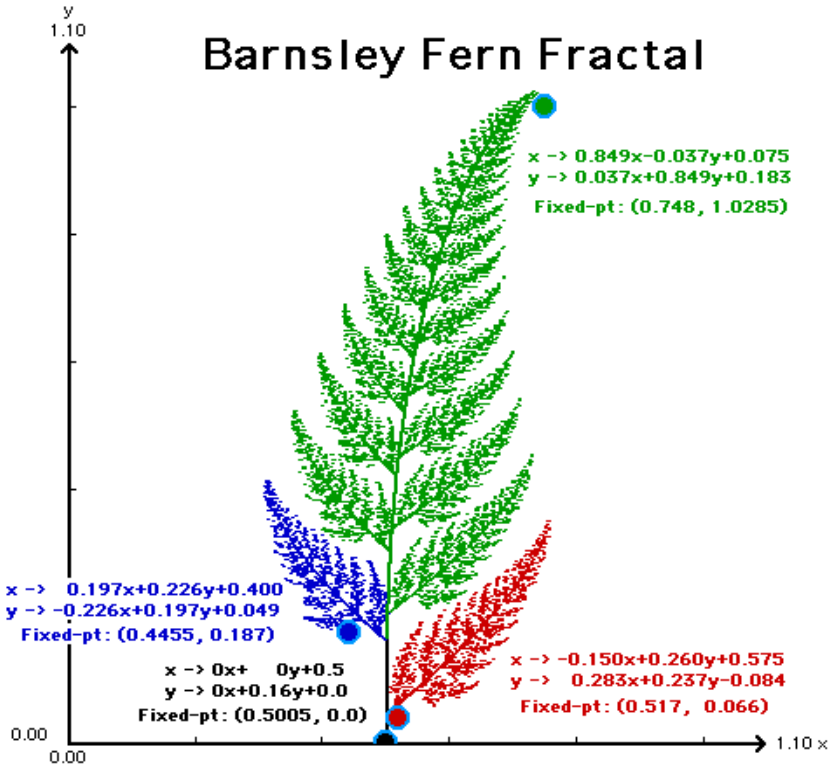

Figure 2.4. Barnsley Fern

Attractor and Hutchinson Operator are important concepts in the context of Iterated Function Systems (IFS) and fractal geometry [34], [35]. Attractor refers to the fixed fractal structure obtained as a result of the iterative application of the IFS, while the Hutchinson Operator describes this process mathematically and allows to obtain the attractor. These concepts are fundamental tools in the process of creating and analyzing fractal structures.

2.3. Fractal Types

Fractals are divided into different types according to their geometric properties and methods of creation [38].

2.3.1. Geometric fractals

Such fractals are created using a specific rule or formula. For example, the Sierpinski triangle and the Koch snowflake are classic examples of geometric fractals.

2.3.2. Natural fractals

Many structures found in nature show fractal properties. Tree branches, river deltas, mountain ranges and lung bronchi are examples of natural fractals.

2.3.3. Chaotic fractals

In dynamic systems, there are fractals that exhibit chaotic behaviors. These types of fractals are extremely sensitive to specific initial conditions, and the Lorenz attractor is a well-known example of this type. An example of the fracral obtained with the help of the Lorenz attractor is given in Figure 2.5.

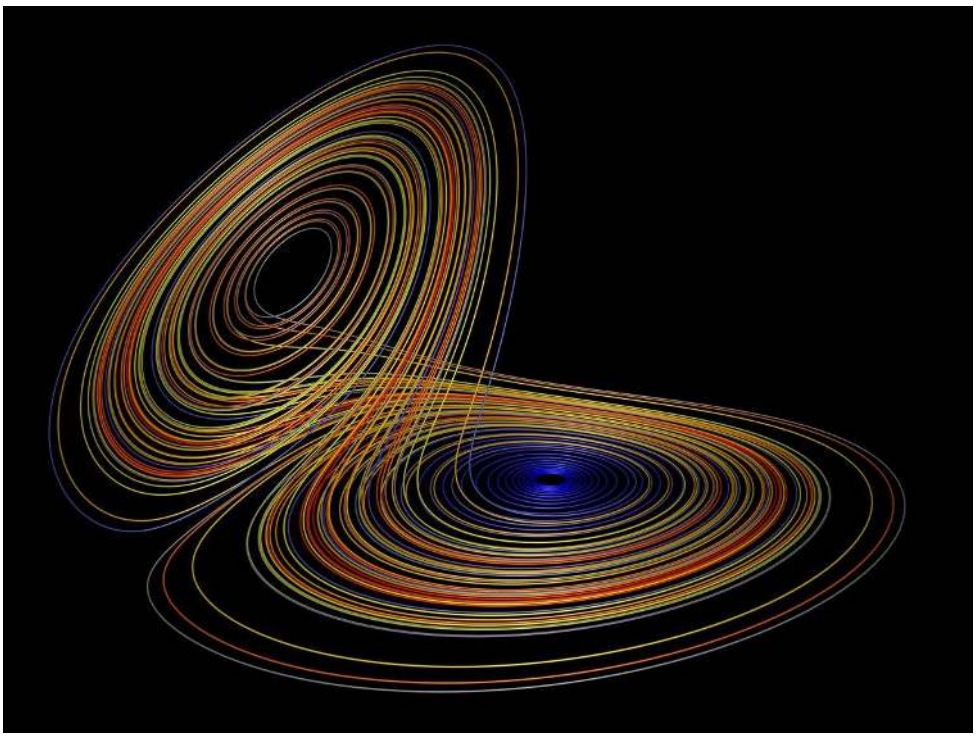

Figure 2.5. Lorenz Attractor Fractal

Fractals are a powerful tool to help us understand the mathematical pattern underlying seemingly complex and irregular structures. With properties such as self-similarity, fractional dimension and iterative processes, fractals find a wide range of applications from nature to economics [17], [38].

2.4. Fractals and Image Processing and Compression

Fractals are geometric shapes that can model complex image data with simple mathematical iterations. This chapter explores the unique properties of fractals and how they are used in digital image processing and compression.

Fractal compression theory is a method of compressing images using the repetitive qualities of fractals, which are mathematical patterns that repeat themselves. This theory aims to optimize data storage and transmission by assuming the existence of repeating patterns, no matter how small in scale.

Fractal image compression works by using Iterated Function Systems (IFS) and similar fractal generation techniques. Images are reorganized so that they are expressed in terms of their smaller content, and these smaller copies are expressed by mathematical transformations [39]. During the compression process, the optimal fractal code is searched for each image region (Range Block) and this code is used to reproduce that region of the original image.

2.4.1. Applications in image processing

Image processing Fractals have a wide range of applications and are used in a variety of techniques. The ability to model complex structures is fractals' greatest strength in image applications and analysis [9], [16], [40]. Fractal analysis opens doors to innovative solutions in many fields, as it enables better analysis and modeling of complex and seemingly irregular structures.

These unique properties of fractals make them a valuable tool in digital image processing and offer researchers new perspectives on data representation and analysis. Applications of fractal geometry can revolutionize the processing of complex and detailed image data and pave the way for the development of new methods in this field.

2.4.1.1. Compression

Fractal Compression: It is advantageous for compression because it describes small images or pieces of data in terms of mathematical transformation rules.

Fractal compression is a method for achieving smaller file sizes by exploiting the self-similarity properties of images. Fractal compression is based on the principle that small parts of images are expressed by transformations of larger parts [41].

The fractal compression process can be described as follows.

a. Splitting the Image into Blocks: The image is divided into small square blocks.
b. Transformations of Blocks: For each block, there is a similarity transformation (e.g., rotation, scaling, displacement) from a larger block.
c. Encoding: The similarity transformations found for each block are stored as a code.

The similarity transformation for each block (S_i), is defined as follows:

$$[S_i(x, y) = (a_i x + b_i y + e_i, c_i x + d_i y + f_i)] \quad (2.9)$$

Here $(a_i, b_i, c_i, d_i, e_i, f_i)$ are the transformation parameters. These parameters define the relationship of blocks to larger blocks.

2.4.1.2. Enhancement

Image Sharpening: Fractal algorithms can be used to sharpen the edges and details of images. It can be used for edge modelling to find edges [42]. This is particularly useful in medical imaging and satellite imagery.

Noise Reduction: Fractal based methods can also be used for modelling random noises for noise removal. It can effectively reduce the modelled noise in the images, and cleaner and clearer images can be obtained.

2.4.1.3. Image recognition and classification

Pattern Recognition: Fractal dimension and fractal features can be used for the recognition and classification of complex patterns. This is common in biometric identification, face recognition and fingerprint analysis [9].

Surface Analysis: Fractal properties of material surfaces can be used to analyze the roughness and other physical properties of the surface [43].

2.4.1.4. Analyzing natural images

Landscape and Geographic Information Systems: Satellite imagery and aerial photographs can be used for fractal analysis, land classification, vegetation analysis and monitoring of environmental changes [44].

Meteorology and Oceanography: Fractal patterns in air and water surfaces can be analyzed for climate change and weather forecasting.

2.5. Advantages and Disadvantages of Fractal Compression

Fractal compression techniques offer a unique approach to digital image processing, but as with any application, there are advantages and disadvantages.

Although fractal compression techniques offer significant advantages in digital image processing, the applicability of these methods may face some practical limitations. Advantages such as high compression ratios and resolution independence must be balanced against disadvantages such as high computational requirements and long processing times.

2.5.1. Advantages

2.5.1.1. High compression rates

Fractal compression can achieve much higher compression ratios than other compression methods, especially in images with an abundance of repeating patterns. This saves storage space and speeds up data transmission. Due to the high altitude and intended use in many UAV applications, there is a high probability that similar patterns will occur [45].

2.5.1.2. Resolution independence

Fractal compression theoretically offers resolution independence; that is, the compressed image can be reconstructed at any resolution. This is particularly useful for applications that need to be displayed at various resolutions. Images transferred to disk or sent to the control center can be obtained at a higher resolution using multipliers of the mathematical model that will be used during file decompression [46].

2.5.2. Disadvantages

2.5.2.1. High computation requirement

Fractal compression algorithms may require more computational power than other compression methods, especially when applied on large images [47]. This can be a disadvantage, especially in time-critical applications. Although the larger the image, the higher the compression performance, an exponential increase in the number of blocks results in an exponential increase in computational cost.

2.5.2.2. Compression and decompression time

Fractal compression may need long compression and decompression times, especially as it requires an iterative process [48], [49]. This may not be suitable for real-time applications. Finding the similarities of each part and applying transformations while finding these similarities causes loss in compression and decompression time. Parallel computing methods can be used for future applications.

3. MOTIVATION

This thesis analyses the results of fractal compression based on the prediction that the images obtained from unmanned aerial vehicles contain high similarity. Different transformations and inversions used for fractal compression will be tested and their results will be compared.

All of the images used as a database are original images taken from real flight images.

The results obtained will be comparatively compared with traditional methods and outputs for future studies will be discussed.

3.1. Literature Review

3.1.1. Traditional image compression methods

Traditional image compression techniques aim to reduce the data size during storage and transmission of digital images. These methods can be analyzed under two main categories: lossless and lossy. Lossless compression methods are techniques in which no information from the original data is lost, while lossy compression methods are techniques in which higher compression ratios are achieved by sacrificing some original data details. It should be decided which method will be used for the specific application.

Both methods have advantages and disadvantages. Lossless compression methods increase the amount of time and resources, while lossy compression methods result in an acceptable level of detail loss.

Traditional compression algorithms have been continuously improved over the years and optimized for various application needs. Each of these methods offers advantages and disadvantages in specific usage scenarios. For example, lossless methods preserve data integrity, while lossy methods offer greater data compression but may result in degraded image quality.

3.1.1.1. Lossless compression methods

Lossless compression is particularly favored in applications where data integrity and detail are critical, such as medical imaging, technical drawings, space applications and important archive images. These methods aim to ensure that the original image can be fully reconstructed from the compressed data. The most widely used lossless image compression formats include PNG and GIF.

Lossless compression works by finding identical blocks or detecting predictable patterns in image data. These methods allow data to be represented in a more compact format while preserving data integrity. Lossless compression methods usually consist of two main stages: prediction and coding.

3.1.1.1.1. Run-Length encoding (RLE)

RLE represents consecutive repeating data values with a single value and a number indicating how many times this value is repeated. Finding repeating patterns and obtaining the number of repetitions is the most important steps. This method is particularly effective for medical imaging data with large, homogeneous color spaces [50].

The RLE algorithm can be expressed mathematically as follows:

Let it be a data array $(\mathrm{S} = s_1, s_2, \ldots, s_n)$.

(s_i) If the character repeats consecutively, this repetition $((R_i, C_i))$ Let us express it in pairs, where (R_i) the number of repetitions and (C_i) represents the character.

$$[S = \{(R_1, C_1), (R_2, C_2), \ldots, (R_k, C_k)\}] \tag{3.1}$$

Here (k) represents the number of different character pairs in the compressed sequence.

The RLE encoding process can be defined by the following formulas:

(S) Original data array,

(C) Compressed data array.

$$[C = \bigcup_{i=1}^{k} (R_i, C_i)] \tag{3.2}$$

Here (R_i) number of repetitions, (C_i) represents the character.

The RLE decoding process can be described by the following formulas:

(C) Compressed data array,

(S) Original data array.

$$[S = \bigcup_{i=1}^{k} C_i^{R_i}] \tag{3.3}$$

Here $(C_i^{R_i})$, (C_i) character (R_i) refers to its repetition.

3.1.1.1.2. Huffman coding

Huffman Coding is an algorithm for lossless data compression developed by David A. Huffman in 1952. The algorithm assigns shorter codes to frequently used characters and longer codes to rarely used characters. This minimizes the average code length and compresses the data more efficiently [51].

Huffman coding creates variable length codes based on a frequency analysis of the data. Shorter codes are used for more frequently used data values, resulting in efficient compression.

Huffman coding is performed in the following steps:

a. Calculation of Character Frequencies: The frequency of each character in the data is calculated.

b. Creating a Min-Heap: A min-heap is created using each character and its frequency as a node.

c. Tree Building: Two nodes with the smallest frequency are selected and these two nodes are merged to form a new internal node. The frequency of this new node is the sum of the frequencies of the merged nodes. This process is repeated until all nodes are merged.

d. Code Assignment: In the generated Huffman tree, for each branch starting from the root node, 0 is assigned to the left branch and 1 to the right branch. Each leaf node represents the Huffman code of that character.

According to the frequencies of the characters in the data set, the entropy is calculated as follows:

$$[H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p\,(x_i)] \tag{3.4}$$

Here $(p(x_i))$, (i) is the probability of the -th character.

Huffman coding minimizes the average code length. Average code length $((L))$ is calculated as follows:

$$[L = \sum_{i=1}^{n} p(x_i) l(x_i)] \tag{3.5}$$

Here $(l(x_i))$, is the length of the Huffman code of the (i)-th character.

Coding efficiency $((\eta))$ expresses the relationship between entropy and average code length:

$$[\eta = \frac{H(X)}{L}] \tag{3.6}$$

3.1.1.1.3. Lempel-Ziv welch (LZW) algorithm

LZW is an algorithm that represents frequently used data sequences with fixed-length codes. This method achieves high compression ratios by recognizing repetitive patterns and structures in the image [52], [53].

The coding steps of the LWZ algorithm are as described below.

a. Creating an Initial Dictionary: An initial dictionary is created containing all single-character sequences.
b. Read Array Read a character from the data stream and add it to the current array.
c. Check Dictionary: If the current sequence exists in the dictionary, read the next character and add it to this sequence.
d. Add New Entry: If the current array is not in the dictionary, add it to the dictionary and write the code of the previous version of the current array to the output.
e. Repetition: Repeat steps until all characters have been processed.

The decoding steps of the LWZ algorithm are as described below.

a. Creating an Initial Dictionary: Create an initial dictionary containing all single-character sequences.

b. Read Code Read the first code and write the corresponding array to the output.

c. Add New Entry: Each time a new code is read, add the character inserted in the previous sequence to the dictionary.

d. Repetition: Repeat steps until all codes have been processed.

The LZW algorithm can be described by the following mathematical models and formulas:

Dictionary at the beginning (D)contains all single-character arrays:

$$[D = \{(i, c_i) \mid i = 0,1,2, \dots, n-1\}] \quad (3.7)$$

Here (c_i) the one-character show and (i) represents the code of this array.

Data array $(S = s_1, s_2, \dots, s_m)$ Let it be. The coding process is defined by the following steps:

Coding steps:

1. $(w \leftarrow \epsilon)$ (empty array)
2. Loop each: $(k \in S)$ for

   $(wc \leftarrow w + k)$

   If (wc) if it is in the dictionary:

   $(w \leftarrow wc)$

   If not:

   - $(D[w])$ code to the output.
   - (wc) Add the sequence to the dictionary.
   - $(w \leftarrow k)$
3. (w) if not empty $(D[w])$ code to the output.

Decoding steps:

Code sequence ($C = c_1, c_2, \dots, c_p$) Let it be. The decoding process is defined by the following steps:

1. Read the first code and write the corresponding sequence to the output.

Cycle 2: each ($k \in C$) for

($entry \leftarrow$) The corresponding sequence in the code dictionary.

($entry$) Write on the exit.

As a new entry in the dictionary ($w +$) first character ($entry$) Add.

($w \leftarrow entry$)

3.1.1.1.4. Predictive coding

Predictive coding predicts the value of a pixel based on the values of neighboring pixels and stores only the prediction error. This method is particularly effective in images with high correlation between consecutive pixels [54].

Predictive Coding expresses each element of a data array with a value predicted from previous elements. The difference between the predicted value and the actual value is called prediction error and this error is encoded and the data is compressed.

Predictive Coding is basically realized with the following steps:

a. Forecast Model: A forecast model is created using current and previous data.

b. Prediction: The prediction value is calculated for each element of the data.

c. Error Calculation: The difference between the actual value and the predicted value is calculated.

d. Coding: Prediction errors (differences) are encoded and compressed.

Predicted data value ($\widehat{x_n}$)is calculated based on previous data values. In the simplest form, ($\widehat{x_n}$), (x_{n-1}) may be equal to the previous value:

$$[\widehat{x_n} = x_{n-1}] \tag{3.8}$$

In more complex models, a weighted average of several previous values can be used:

$$[\widehat{x_n} = \sum_{i=1}^{k} a_i x_{n-i}] \tag{3.9}$$

Here (a_i) weight coefficients and (k) represents the number of times to look at past values.

Failure of foresight (e_n), actual data value (x_n) the data value estimated with $(\widehat{x_n})$ is the difference between:

$$[e_n = x_n - \widehat{x_n}] \tag{3.10}$$

Error signals (e_n) compressed by encoding. Since these errors are usually small values, they can be represented using fewer bits, resulting in data compression.

3.1.1.2. Lossy compression methods

Lossy compression methods are often used in areas such as web use, video broadcasting and consumer electronics, where details are not noticeable to the end-user eye because these techniques provide much higher compression ratios. JPEG is the most popular lossy compression standard and is commonly used for photography and internet graphics. JPEG compression works by reducing certain frequency components in the image, which can result in some loss of detail.

Lossy compression aims to remove imperceptible details from image data by exploiting the limitations of the human visual system. These methods reduce the size of the data using frequency transforms, quantization and entropy coding techniques.

3.1.1.2.1. JPEG (Joint photographic experts group)

JPEG is the most widely used lossy compression standard, especially for images. JPEG compression significantly reduces the size of images, making them suitable for storage and transmission [55].

JPEG compression starts with a conversion from RGB (red, green, blue) color space to YCbCr color space. YCbCr considers that the human eye is more sensitive to brightness and less aware of color details. The Y component contains luminance information, while the Cb and Cr components contain color information.

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cb = 128 - 0.168736R - 0.331264G + 0.5B$$
$$Cr = 128 + 0.5R - 0.418688G - 0.081312B$$

The image is then divided into 8x8 pixel segments to facilitate the transformation and quantization process.

Each 8x8 pixel fragment is separated into frequency components using the Discrete Cosine Transform (DCT). The DCT separates the low and high frequency components by transforming the data into frequency space [56]. With DCT, the low frequency components are usually located in the upper left corner of the block and the high frequency components in the lower right corner.

$$[F(u,v) = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]] \quad (3.11)$$

Here $(\alpha(u))$ and $(\alpha(v))$ coefficients, (u) and (v) according to their values:

$$[\alpha(u), \alpha(v) = \begin{cases} \frac{1}{\sqrt{2}} & if \quad u, v = 0 \\ 1 & Other \end{cases} \quad (3.12)$$

It reduces the amount of data by erasing small frequency components, and the resulting frequency data is quantized. During this process, high frequency components are further quantized, resulting in more information loss.

Quantization is performed by dividing each DCT coefficient by the quantization matrix:

$$[Q(u,v) = \left\lfloor \frac{F(u,v)}{Q_{table}(u,v)} \right\rfloor] \quad (3.13)$$

Here $(Q_{table}(u,v))$ represents the elements of the quantization matrix.

The resulting coefficients are compressed by coding with entropy-based methods, either Huffman coding or Run-Length coding.

JPEG compression was also applied in this study as a comparison.

3.1.1.2.2. Wavelet compression

This method works with a mathematical transformation that decomposes information into frequency components. The Wavelet transform increases the compressibility of data by decomposing it into different frequency components [57], [58].

The Wavelet transform divides the image into at least two levels, low-frequency components and high-frequency components.

The low-frequency components provide information about the overall structure of the image, while the high-frequency components represent the details in the image.

This decomposition is usually done using the Discrete Wavelet Transform (DWT). The DWT provides a multi-scale analysis by decomposing the data into different levels of resolution.

DWT involves the separation of a signal using low pass and high pass filters [59]. [59]. A signal ($x[n]$) DWT coefficients are calculated as follows:

$$[A_j[k] = \sum_n x[n]g[2k - n]] \tag{3.14}$$

$$[D_j[k] = \sum_n x[n]h[2k - n]] \tag{3.15}$$

Here:

($A_j[k]$): (j)-approximation coefficients at the first level.

($D_j[k]$): (j)-first level detail coefficients.

($g[n]$): Low-pass filter.

($h[n]$): High-pass filter.

The resulting wavelet coefficients are then quantized. This is the process of rounding the data to certain intervals and this is why losses in the compression method occur. A certain level of data loss has to be accepted.

$$[Q_{A_j[k]} = \left\lfloor \frac{A_j[k]}{Q_{step}} \right\rfloor] \tag{3.16}$$

$$[Q_{D_j[k]} = \left\lfloor \frac{D_j[k]}{Q_{step}} \right\rfloor] \tag{3.17}$$

Here:

$(Q_{A_j[k]})$ and $(Q_{D_j[k]})$: Quantized coefficients.

(Q_{step}): Quantization step size.

The quantized wavelet coefficients are encoded in Huffman or run-length, which are entropy-based coding methods to provide greater compression [50], [51]. This step is added to make the compressed image suitable for storage.

The inverse transformation process is performed by reversing the steps described above. Due to the reduction of high-frequency components, the level of detail on the image is reduced, but the data obtained as a result of the inverse transform can be very close to the original data.

In this study, wavelet compression was also applied as a comparison.

3.1.1.2.3. JPEG 2000

JPEG 2000 is another lossy compression standard that provides higher compression ratios and better image quality than JPEG. This method analyzes image data using a discrete wavelet transform (DWT) and offers more flexible compression options.

3.2. Existing Fractal Compression Algorithms

Fractal compression has significant potential in the field of image processing due to its unique mathematical approaches and efficient compression ratios. This chapter discusses existing algorithms developed in the field of fractal compression and their practical applications.

3.2.1. Fractal compression algorithms

Fractal compression algorithms are usually developed using "Iterated Function Systems" (IFS). These algorithms detect similar parts of the image, express these parts with mathematical functions and reconstruct the image with the help of these functions [7], [8], [41]. The differences that make up the diversity of functions consist of methods for finding similarities and methods for expressing these similarities mathematically.

Jacquin's algorithm, PIFS and quadtree-based methods have made significant contributions to the use of fractal compression [33], [60], [61]. Modern applications improve

the performance and efficiency of fractal compression with hybrid methods, parallel processing techniques and deep learning [62], [63], [64]. These developments suggest that fractal compression will find wider use in the future.

3.2.1.1. Jacquin's algorithm

The algorithm proposed by Jacquin (1992) is one of the most well-known and widely used methods of fractal compression [24]. This algorithm divides the image into smaller parts (domain blocks) and larger blocks (range blocks) and finds the best matching domain block for each range block.

The detailed steps of the algorithm are as follows.

a. Splitting the Image

Image, ($N \times N$) into small square blocks of pixel size. Each ($N \times N$) block, called a range block.

b. 2. Defining Domain and Range Blocks

Domain blocks, ($2N \times 2N$) pixels in size and each ($2N \times 2N$) domain block, four ($N \times N$) is divided into sub-domain blocks. These sub-domain blocks are ($N \times N$) is reduced to size.

c. Finding Affine Transformations

For each range block, the optimal domain block and affine transformation are found by the following steps:

Shrinking Domain Blocks with Down sampling: Each ($2N \times 2N$) domain block, ($N \times N$) is reduced to size.

Affine Transform Calculation: For each range block, an affine transformation is applied to all minimized domain blocks and the transformation with the lowest error is selected among these transformations.

Affine transformation parameters are calculated as follows:

$$[\alpha_i = \frac{\mathrm{Cov}(R, D)}{\mathrm{Var}(D)}] \tag{3.18}$$

$$[\beta_i = \mu_R - \alpha_i \mu_D] \tag{3.19}$$

Here:

$(\mathrm{Cov}(R, D))$: Covariance between range block (R) and domain block (D).

$(\mathrm{Var}(D))$: Variance of the domain block.

(μ_R): Range is the average of the block.

(μ_D): Domain block average.

d. Encoding and Storage

Affine transformation parameters $((\alpha_i, \beta_i))$ and the position of the domain block is stored. These parameters form the compressed image data.

3.2.1.2. Partitioned iterated function system (PIFS)

PIFS is an extension of Jacquin's algorithm and aims to achieve higher compression ratios by using multiple domain blocks for each range block in the image [65]. This method allows for better representation, especially of complex and irregular patterns.

PIFS divides an image into sub-blocks, representing each sub-block as a similarity of another part of the image. PIFS provides compression using an affine transformation for each sub-block [66].

A PIFS consists of affine transformations. Each transformation is a block (B_i) another blocks (B_j) as the affine transformation.

Affine transformation is defined as follows:

$$[w_i(x, y) = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} (xy) + (e_i f_i)] \tag{3.20}$$

Here:

$(w_i(x, y))$: Affine transformation.

(a_i, b_i, c_i, d_i): Components of the transformation matrix.

(e_i, f_i): Displacement components.

The PIFS compression steps are:

a. Partitioning: The image is divided into smaller blocks.

b. Finding Affine Transform: Find the most appropriate affine transformation for each sub-block.

c. Coding: Affine transformation parameters and block information are stored for each block.

3.2.1.3. Quadtree based fractal compression

This method hierarchically divides the image according to a quadtree structure and calculates the fractal transform of each region. This structure enables efficient compression, especially at different resolution levels [67].

A quadtree is a tree structure where each node is subdivided into four sub-nodes. The image is subdivided with a certain error threshold and this process continues until a certain stopping criterion is met. Quadtree provides efficient compression, especially for images containing homogeneous regions [68].

Root Node The top node representing the whole image.

Internal Nodes: Nodes that are subdivided into smaller subdivisions.

Leaf Nodes: Nodes that do not split further and are considered sufficiently homogeneous.

a. Initializing and Splitting the Image

The image is initially considered as a single root node and then recursively split into four sub-nodes. These subdivisions form the internal nodes of the Quadtree structure.

b. Homogeneity Check and Splitting

Each node is evaluated according to a certain homogeneity criterion. If the node is not homogeneous enough, it is split into four more sub-nodes. This process continues until the homogeneity criterion is met.

c. Error Calculation and Stopping Criteria

For each child node, the error is calculated and compared to a certain error threshold. When the error threshold is met, the splitting process is stopped and the node is considered as a leaf node.

Error calculation is usually done using the mean deviation or variance of the pixel values. For example, (σ) The error calculated using variance can be as follows:

$$[\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2] \tag{3.21}$$

Here:

(σ^2): Variance of pixels within a node.

(N): Number of pixels in the node.

(x_i): The value of each pixel.

(μ): The average value of the pixels within the node.

d. Encoding and Storage

Each leaf node represents the region considered homogeneous and the average value of this region or some other representative value is stored. In this way, the image is compressed using less data.

The mathematical basis of the Quadtree model is based on the recursive splitting process and the evaluation of the homogeneity of each split. The homogeneity criterion is usually determined by statistical measures such as variance or mean deviation.

The homogeneity of a node can be checked as follows:

e. Average Value Calculation

$$[\mu = \frac{1}{N}\sum_{i=1}^{N} x_i] \tag{3.22}$$

f. Variance Calculation:

$$[\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2] \tag{3.23}$$

g. Threshold Comparison:

If (σ^2) is smaller than a certain threshold, the node is considered homogeneous and is not further split.

3.2.2. Fractal compression applications

Fractal compression techniques are used in areas such as the management of large image databases, image transmission over the Internet and image storage on mobile devices. These techniques are favored for their high compression ratios and their ability to preserve image quality. Below, we list the works that address these applications. Fractal compression has evolved in recent years with various improvements and new applications.

3.2.2.1. Hybrid methods

Fractal compression can be combined with other compression methods (e.g. wavelet transforms used in JPEG-200) to achieve higher performance [69], [70], [71], [72]. These hybrid methods combine the advantages of fractal and wavelet transforms (DWT).

3.2.2.2. Parallel processing techniques

The computational intensity of fractal compression algorithms can be reduced using parallel processing techniques [62], [73], [74]. Parallel computing with GPU-based, FPGA and CPU pipeline methods will significantly increase the speed of fractal compression.

3.2.2.3. Deep learning and fractal compression

Recently, deep learning techniques have been used to optimize fractal compression processes [72], [75], [76], [77], [78]. In particular, deep neural networks are being studied to provide faster and more accurate detection of fractal codes of images.

3.3. Objective

The main objective of this thesis is to investigate fractal image compression techniques in detail and to evaluate their potential in digital image processing. In particular, the scope of this research is to develop fractal compression methods, analyze their performance and provide innovative solutions to overcome the challenges in this field. This thesis aims to contribute to the body of knowledge in this field by providing a comprehensive review of fractal compression theory.

Research Objectives:

- Understanding Fractal Compression Techniques: To examine the mathematical and technical details of compression algorithms developed based on fractal theory.

- Evaluation of Algorithm Performance: Applying different fractal compression algorithms on various image types and evaluating their performance in terms of compression ratio, processing time and output image quality.
- Comparative Analysis: Comparing fractal compression methods with traditional compression methods (e.g., JPEG, Wavelet) and identifying their advantages and limitations.
- Identifying Application Areas: Identify potential applications of fractal compression and determine which image processing applications these techniques are particularly advantageous.

Developing Innovative Solutions: Provide innovative approaches and improvement proposals to overcome the limitations of existing fractal compression methods.

3.4. Scope

The research will cover both lossy and lossless compression techniques, but the focus will be more on fractal compression, which is a lossy compression method.

The analyzed images will include natural scene photographs from UAVs and will be studied especially on large datasets.

Performance evaluation will be done on the computer with the help of Matlab program.

The study will be limited to current technological possibilities and software tools, but the theoretical proposals will be designed to shed light on future technological developments.

This thesis aims to contribute to the development of fractal compression techniques and develop strategies on how these techniques can be used more effectively in digital image processing. It also aims to bring an innovative perspective to the field by evaluating the impact of these techniques on current and future digital media applications.

4. METHOD

4.1. Database

Within the scope of a project, we are conducting at STM, various raw recordings were taken by performing flight tests. The raw recordings were created by saving the video images taken by the camera used, using the lossless storage format BMP extension. During this study, a Baby Shark 260 VTOL with vertical take-off and landing capability was used [79]. The technical specifications and dimensions of the UAV are given in **Error! Reference source not found.** and Figure 4.1.

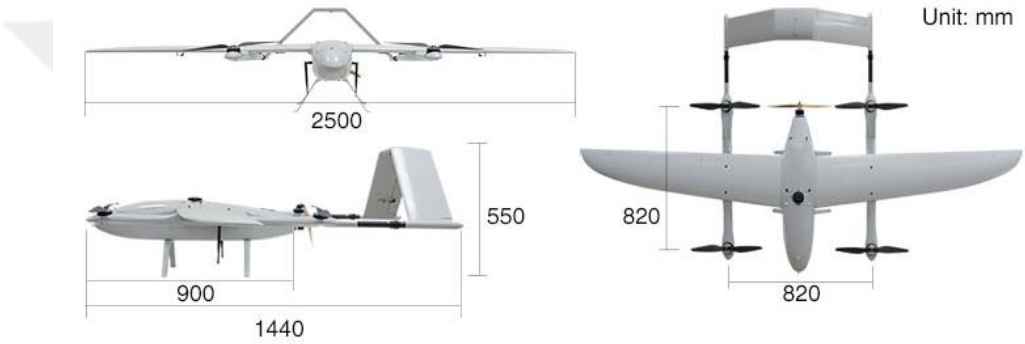

Figure 4.1. Baby Shark 260 VTOL Dimensions

Table 4.1. Baby Shark 260 VTOL Technical Specifications

BABY SHARK 260 VTOL Folding Version	
Material	Composite
Frame Weight	3.5 kg
Basic Empty Weight	6.7 kg (without battery)
Max. Take-off Weight	0~1500m/13kg; 1500m~3000m/11kg
Maximum Cruising Speed	110km/h
Recommended Cruising Speed	21-22m/s
Stall Speed	16m/s
Maximum Bank Rate	30°
Recommended Takeoff Height	0~3000m
Takeoff / Landing	VTOL
Operating Voltage	48V
Unfolded Wingspan/Folded Size	2500mm/900mm
Unfolded Length/Folded Size	1400mm/1000mm

During the flights, raw recordings were made with the Sony FCB-EV7520 camera integrated into the VTOL [80]. Video recording continued from the start of the flight until the moment of re-landing. Video recording was done at 30 frames per second.

There were 4 flights for the captured videos and the total duration was approximately 40 minutes. Since no compression was applied, the RAW recordings were sequentially created as BMP files. In order to better understand the properties of the files, it will be sufficient to read the descriptions below.

The BMP file format or bitmap is a raster graphics image file format used for storing bitmap digital images regardless of the display device (such as a graphics adapter), especially on Microsoft Windows and OS/2 operating systems.

The BMP file format can store two-dimensional digital images in various color depths and optionally with data compression, alpha channels and color profiles. The simplest format of the BMP image file is shown in Figure 4.2.

The 24 bits per pixel (24bpp) format supports 16,777,216 different colors and stores 1 pixel value in 3 bytes. Each pixel value defines the red, green and blue samples of the pixel (8.8.8.8.0.0 in RGBAX notation) [81].

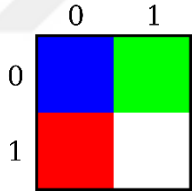

Figure 4.2. BMP Image Format

Table 4.2. BMP File Structure

Offset	Size	Hex Value	Value	Definition
BMP Header				
0h	2	42 4D	"UN"	ID Field (42h, 4Dh)
2h	4	46 00 00 00	70 bytes (54+16)	BMP file size (54 bytes header + 16 bytes data)
6h	2	00 00	Unused	App specific
8h	2	00 00	Unused	App specific
Ah	4	36 00 00 00	54 bytes (14+40)	Offset where the pixel array (bitmap data) can be located
DIB Header				
Eh	4	28 00 00 00	40 bytes	Number of bytes in the DIB header (from this point)
12h	4	02 00 00 00	2 pixels (left to right order)	Bitmap width in pixels

16h	4	02 00 00 00	2 pixels (bottom to top order)	Height of the bitmap in pixels. Positive for pixel order from bottom to top.
1Ah	2	01 00	1 plane	Number of color planes used
1Ch	2	18 00	24 bits	Bits per pixel
1Eh	4	00 00 00 00	0	BI_RGB, pixel array compression is not used
22h	4	10 00 00 00	16 bytes	Size of raw bitmap data (including fill)
26h	4	13 0B 00 00	2835 pixels/meter horizontal	Print resolution of the image,
2Ah	4	13 0B 00 00	2835 pixels/meter vertical	72 DPI × 39.3701 inches per meter gives 2834.6472 × 39.3701 inches per meter
2Eh	4	00 00 00 00	0 colors	Number of colors in the palette
32h	4	00 00 00 00	0 important colors	0 means that all colors are important
Start of pixel array (bitmap data)				
36h	3	00 00 FF	0 0 255	Red, Pixel (x=0, y=1)
39h	3	FF FF FF FF	255 255 255	White, Pixel (x=1, y=1)
3Ch	2	00 00	0 0	4 bytes of padding for alignment (can be a non-zero value)
3Eh	3	FF 00 00 00	255 0 0	Blue, Pixel (x=0, y=0)
41h	3	00 FF 00	0 255 0	Green, Pixel (x=1, y=0)
44h	2	00 00	0 0	4 bytes of padding for alignment (can be a non-zero value)

73001 BMP files were received during the flights. The total size of our images is 188 GB (202,132,176,896 bytes). The properties of our BMP files are given in Table 4.3.

Table 4.3. Characteristics of Each Image

Name	Value
File Size	2.6 MB
File Permissions	-rw-------
File Type	BMP
File Type Extension	bmp
MIME Type	image/bmp
BMP Version	Windows V3
Image Width	1280
Image Length	720
Plane	1
Bit Depth	24

Name	Value
Compression	None
Image Total Length	2764800
Pixels Per Meter X	3780
Pixels Per Meter	3780
Number of Colors	Use Color Depth
Number of Important Colors	All
Image Size	1280x720
Megapixel	0.922

File properties extracted with help from //exif.tools/

An example of the beginning of the flight from the pictures we used is given in Figure 4.3.

Figure 4.3. Sample Database Image

In our application, our algorithm was applied to a total of 500 BMP images to obtain the results. The total size of these images is 1382427000 Bytes. We tried to select images that were taken at different moments of the flight and had little similarity to each other.

Since the processing time per image is quite high, the application is limited to 500 images.

The average image size is 2764854 Bytes.

4.2. Algorithm

The compression algorithms we used for comparison in our study are given in Table 4.4, Table 4.5 and Table 4.6.

Table 4.4. Fractal Compression Algorithm

1: **Start** Read Image Files

Define Range Block and Domain Block sizes.

Generate Official Range and Domain blocks and segment them according to their size.

Find the number of pieces according to the Block Size.

Repeat for the number of pieces and cut the picture into pieces.

Generate Domain blocks Reduce to Range block size.

Her ($2N \times 2N$) domain block, ($N \times N$) is reduced in size.

2: **Repeat** for all Range blocks created.

Compare each Range Block with Domain blocks according to Frobenius Norma.

Save Result Save the Domain block information with the best result.

Apply Conversion Apply Affine conversions to each Domain Block.

Translation: $(x, y) \rightarrow (x + t_x, y + t_y)$

Scaling $(x, y) \rightarrow (sx \cdot x, sy \cdot y)$

Rotation $(x, y) \rightarrow (x \cos\theta - y \sin\theta\,, x \sin\theta + y \cos\theta)$

Cutting $(x, y) \rightarrow (x + ky, y)$

3: **Save** Save the Domain block and conversion information that gives the best result.

4: **Finalize** Create data file from the saved information and used Domain blocks.

5: **Rebuild**: Retrieve saved DBlocks.

Apply Inverse Transform Apply transformations of Dblocks in reverse.

Merge Place the new blocks at the saved coordinate.

Table 4.5. JPEG Compression Algorithm

1: **Start** Read Image File

2: **Convert YCbCr**: Convert the image file to YCbCr color space.

Produce $Y = 0.299R + 0.587G + 0.114B$

Produce $Cb = 128 - 0.168736R - 0.331264G + 0.5B$

Produce $Cr = 128 + 0.5R - 0.418688G - 0.081312B$

3: **Divide into Blocks** Divide the image into 8x8 pixel pieces.

4: **Apply Transform** Apply Discrete Cosine Transform (DCT).

5: **Reduce data** Delete small frequency components.

6: **Quantize** Divide by the quantization matrix of each DCT coefficient.

7: **Code** Huffman encoding or Run-Length encoding.

8: **Save** Save the generated codes.

Table 4.6. Wavelet Compression Algorithm

1: **Start** Read Image File.

2: **Apply Transform** Apply Discrete Wavelet Transform (DWT).

3: **Find Coefficients** Find DWT coefficients with Low Pass and High Pass Filter.

4: **Quantize** Quantization Parameters, Divide by Step size.

5: **Code** Huffman encoding or Run-Length encoding.

6: **Finalize** Create data file from the saved information and used Domain blocks.

7: **Save** Save the generated codes.

4.3. Metrics of the Method

Our algorithm was applied to a total of 500 BMP images. The total size of these images is 1382427000 Bytes. The average image size is 2764854 Bytes.

When we saved the images as JPG, the total size decreased to 31640066 Bytes. In order to equalize the quality and file size ratio, the JPG quality factor was set to 50. The average image size is 63280.132 Bytes.

The total size when the images are subjected to fractal compression and saved is 27734622 Bytes. The average image size in this case was found to be 55469.244 Bytes.

In the compression, while the RBlock size was 4x4, the Dblocks were tested to be 16x16 with 4 times the size.

Image, ($n \times n$) into small blocks of size.

$$[I = \cup_{i=1}^{k} B_i] \tag{4.1}$$

Here (I)original image and (B_i), $(i)$is the 'th block.

Since the resolution is equal in the example image and all processed images, there are 57600 Domain blocks. The repetition amount of these is 54751. In other words, the amount of Domain blocks that make up the image is 2550. Domain blocks consisting of 768 bits (16x16x3) total 1958400 bits. Another 180x320x16 (two 8 Bit) data (921,600 bits) must be added to store the indexes of the Domain blocks to which the Range blocks correspond. The total compressed file size is 2,880,000 bits. The size of the original BMP file is 22,118,832 bits. This results in 7.68015 times compressed data.

The total compression (encode) time of the images is 118814.0174 (1980 minutes, 33 Hours) seconds. The average compression time per image is 237.6280347 seconds.

The time to decode the images is 72.9376911 seconds. This time per image was measured as 0.145875382 seconds.

$$[\text{Coding} = \{(a_i, b_i, c_i, d_i, e_i, f_i) \mid i = 1,2, \dots, k\}]$$

Here (k)is the number of blocks.

Decoding

$$[B_i = S_i(B_i)]$$

This process is applied iteratively for all blocks, resulting in a reconstructed image.

The computer on which the algorithm is run is Windows 11 Pro configuration, a 64-bit operating system with an Intel I7 8750H @2.2 GHz processor and 64GB RAM.

All coding and testing were done in Matlab version 2023a.

4.4. Evaluation Metrics

4.4.1. Compression ratio (CR)

CR (Compression Ratio) is a metric used in image processing and data storage, particularly to evaluate the efficiency of data compression techniques [82]. Compression

Ratio is defined as the ratio of the original data size to the compressed data size and is an indicator of the size savings achieved as a result of the compression process.

Compression Ratio is used to evaluate the performance of compression algorithms. In particular, for applications such as image compression, video compression and audio file compression, CR provides important information about compression quality and efficiency. CR also helps to understand whether data is lost after compression and, if so, the extent of that loss [83].

Compression Ratio (CR) is usually expressed by the formula:

$$\mathrm{CR} = \frac{\text{Original Data Size}}{\text{Compressed Data Size}} \tag{4.2}$$

In this formula, "Original Data Size" refers to the size of the original image or file and "Compressed Data Size" refers to the size obtained after compression. The higher the CR value, the more effective the compression.

The CR results of our study are given in Table 4.7. The results obtained for each image are given in Figure 4.4 and Figure 4.5. The horizontal axis is the number of images and the vertical axis is the CR value.

Table 4.7. CR Results

CR			
	Average	**Standard Deviation**	**Variance**
FRC	55.16058807	18.22624484	332.196001
JPG	48.71643673	17.43236998	3.04E+02
WLT	271.968422	277.046325	76754.665961

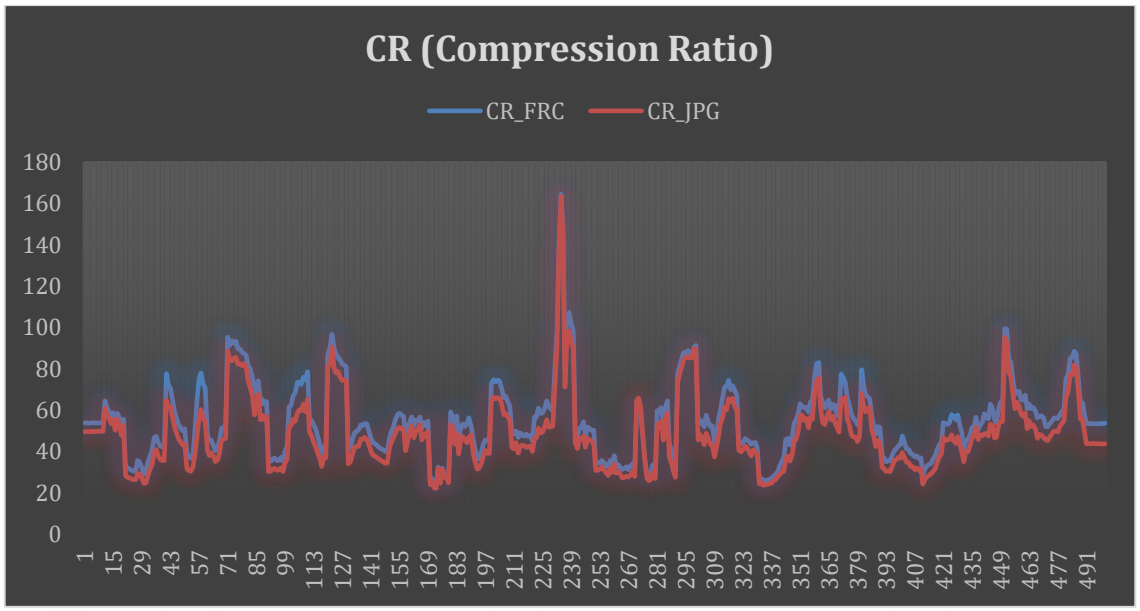

Figure 4.4. Value of CR Results in Images

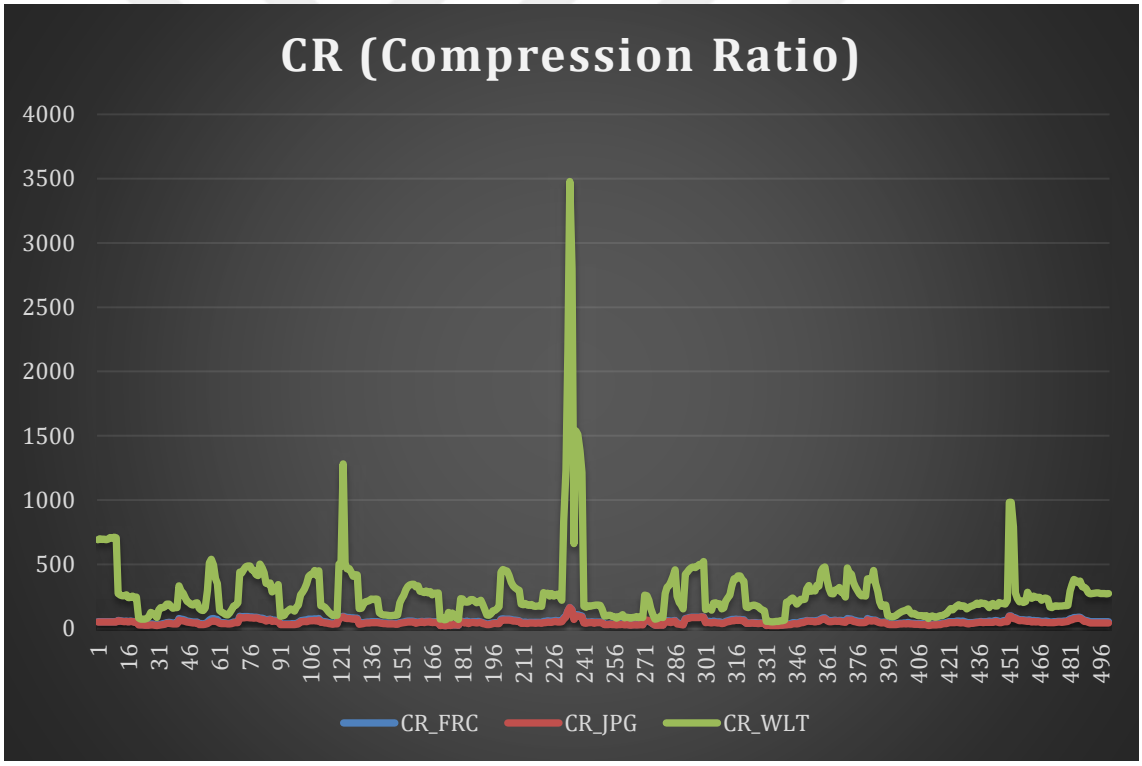

Figure 4.5. Value of CR Results in Images Wavelet

4.4.2. Peak Signal-to-Noise Ratio (PSNR)

PSNR (Peak Signal-to-Noise Ratio) is a widely used measurement metric in image processing, signal processing and video compression. It is used to quantitatively assess the difference in quality between the original and the processed content, i.e. the image degraded by compression or other processing [82], [84]. PSNR measures the logarithm of the ratio

between the maximum possible power of the signal and the power of the error signal and is usually expressed in decibels (dB).

A higher PSNR value means better quality results are likely to be obtained.

PSNR is used specifically to evaluate the quality of image and video compression. The higher the PSNR value, the lower the level of error compared to the original image, indicating a better image quality, closer to the original [85]. However, PSNR may not always accurately reflect the image quality perceived by the human eye. In some cases, even images with low PSNR can be visually satisfying. In particular, images that contain sharpness may be perceived as higher quality to the human eye.

PSNR is usually calculated using the following formula:

$$PSNR = 20 \times \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{4.3}$$

Here (MAX_I) is usually the maximum possible value of the pixel values (e.g., 255 for an 8-bit grayscale image). (MSE) (Mean Squared Error) refers to the mean squared error between the original image and the degraded image and is calculated as follows:

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\big(I(i,j) - K(i,j)\big)^2 \tag{4.4}$$

Here (m) and (n) are the dimensions of the image, (I) original image and (K) is a distorted image.

The PSNR results of our study are given in Table 4.8. The results obtained for each image are given in Figure 4.6. The horizontal axis is the number of images and the vertical axis is the PSNR value.

Table 4.8. PSNR Results

PSNR			
	Average	**Standard Deviation**	**Variance**
FRC	33.27179826	3.240132219	10.4984568
JPG	37.8863464	1.614825394	2.607661054
WLT	30.05011453	2.505022	6.275135

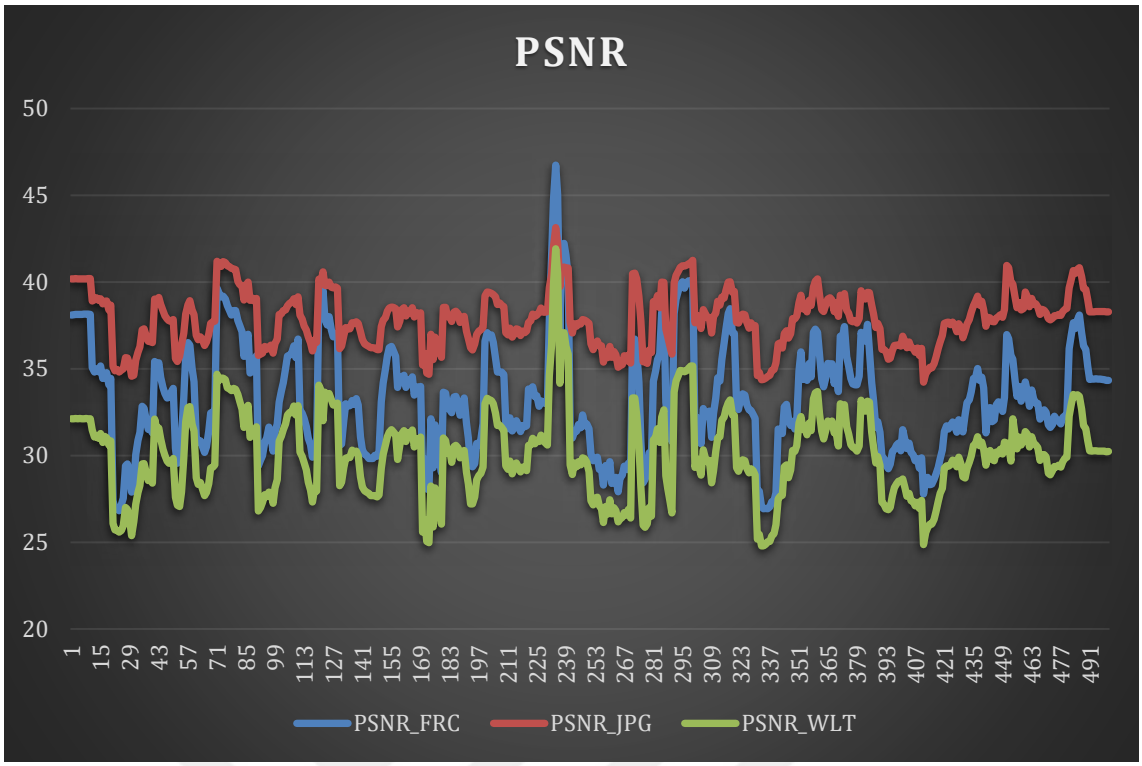

Figure 4.6. Value of PSNR Results in Images

4.4.3. Mean squared error (MSE)

MSE (Mean Squared Error) or Mean Squared Error is one of the most widely used error measurement metrics and is often used to assess the accuracy of predictions in fields such as image processing, statistics, machine learning and signal processing [82], [86]. MSE is the mean of the squares of the differences between actual values and predicted values. This measure quantitatively expresses how much the predictions deviate from the actual values.

MSE is used in many different fields such as regression analysis, image processing and signal processing. MSE is used to evaluate the performance of regression models, to measure the quality of image or signal reconstruction, and to see the level of error in training machine learning models.

One of the major advantages of the MSE is that it is easy to calculate and clearly expresses the amount of error. However, MSE is very sensitive to outliers, meaning that a few high errors can significantly increase the overall MSE value. For this reason, it is sometimes recommended to use it in combination with other metrics or to consider alternative error measures.

A lower MSE means that image quality is preserved.

MSE is calculated using the following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{4.5}$$

Here (Y_i) real values, ($\hat{Y}_i$) refers to the predicted values and (n) specifies the number of samples. Since each difference is squared, large errors are penalized more severely, making the MSE sensitive to extreme values.

The MSE results of our study are given in Table 4.9. The high amount of error in a few pixels causes the MSE results in fractal compression to be high in some images. The results obtained for each image are given in Figure 4.7. The horizontal axis is the number of images and the vertical axis is the MSE value.

Table 4.9. MSE Results

MSE			
	Average	**Standard Deviation**	**Variance**
FRC	39.17223565	27.07094834	732.8362438
JPG	11.32557087	4.236544461	17.94830897
WLT	75.010876	42.009894	1764.831229

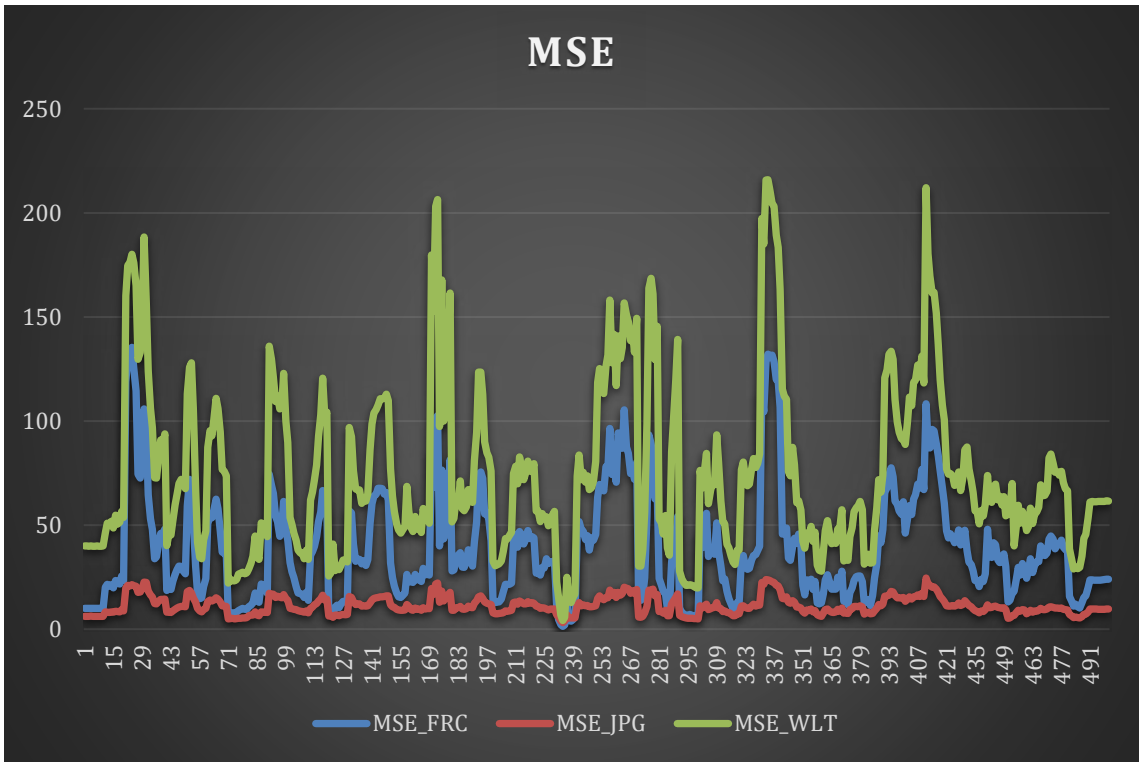

Figure 4.7. Value of MSE Results in Images

4.4.4. Structural similarity index measure (SSIM)

SSIM (Structural Similarity Index Measure) is a method for measuring the similarity between images. Developed by Zhou Wang, Alan C. Bovik, Hamid R. Sheikh and Eero P. Simoncelli, this metric evaluates image quality by comparing the structural similarity of two images. SSIM is designed to produce results closer to the human eye's ability to perceive image quality and is used primarily in image processing, video quality assessment and other media processing applications [87].

Unlike traditional metrics such as Mean Squared Error (MSE) or Peak Signal to Noise Ratio (PSNR), SSIM better reflects the structure and perceived quality of the image. With these features, SSIM is often used to evaluate the performance of compression, noise reduction, super-resolution and other image processing techniques.

SSIM uses mean, variance and covariance to measure the structural similarity between two images. SSIM index (x) and (y) represents two images, the SSIM is calculated according to the following formula:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4.6}$$

Here, (μ_x) and (μ_y) average values of the images, (σ_x^2) and (σ_y^2) variances and (σ_{xy}) is covariance. (c_1) and (c_2) are constants added to ensure stability.

The SSIM results of our study are given in Table 4.10. In this metric, which provides an evaluation according to the human eye, there is a very small difference between traditional methods and fractal compression. The results for each image are given in Figure 4.8. The horizontal axis is the number of images and the vertical axis is the SSIM value.

Table 4.10. SSIM Results

SSIM			
	Average	**Standard Deviation**	**Variance**
FRC	0.92587131	0.028609521	0.000818505
JPG	0.960393775	0.010773127	0.00011606
WLT	0.845842	0.069287	0.004801

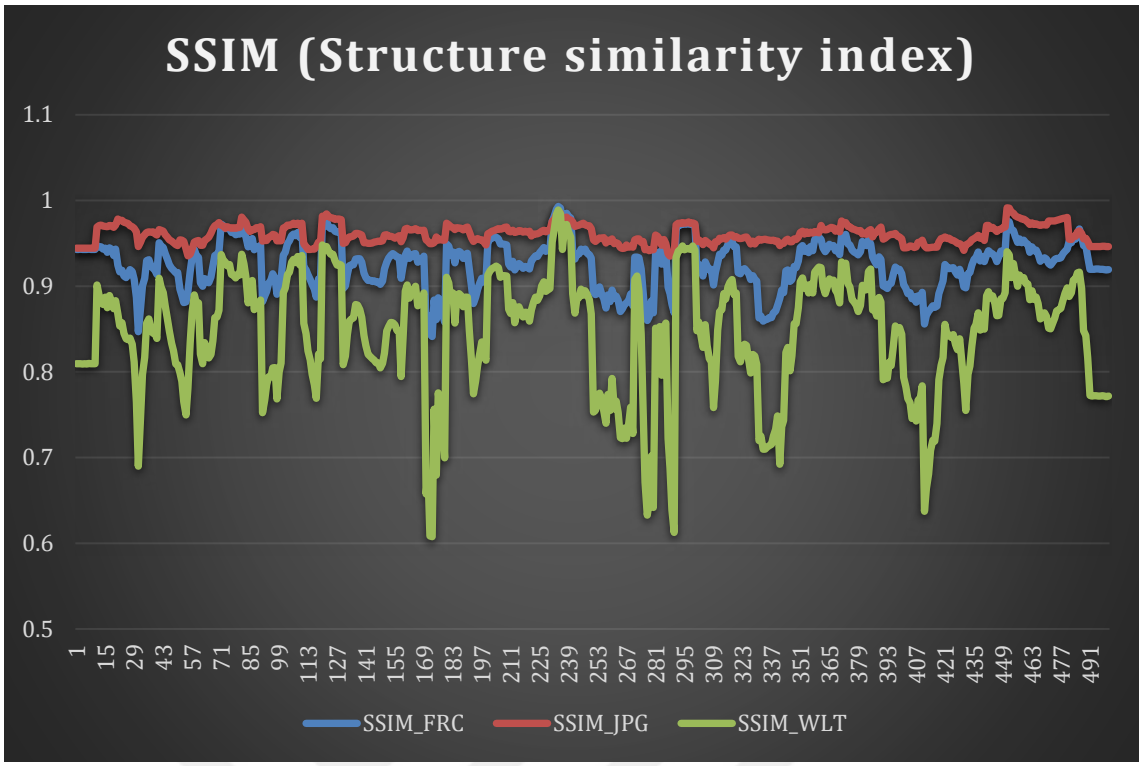

Figure 4.8. Value of SSIM Results in Images

4.4.5. Mean absolute difference (MAD)

MAD (Mean Absolute Difference) or Mean Absolute Difference is a metric used to measure the difference between two images or signals, especially in image processing and video analysis [82]. This metric calculates the average absolute differences between pixel values in a test image relative to a reference image. MAD is often used in image quality assessments and image similarity detection. The lower the difference values, the more similar two images are considered.

MAD, two images or signals (A) and (B) is calculated as follows:

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|A_i - B_i| \tag{4.7}$$

Here (A_i) and (B_i), respectively (A) and (B) of the footage (i)'th pixels and (N) indicates the total number of pixels. $(|A_i - B_i|)$ represents the absolute difference between two-pixel values.

MAD expresses the difference between images in a quantitative way and this has an important role in automated image analysis systems. For example, it can be used in medical imaging, satellite imagery or to evaluate the performance of video compression techniques [88]. MAD is particularly useful in assessing the quality of images that are degraded by noise or other distortions.

The MAD results of our study are given in Table 4.11. The results obtained for each image are given in Figure 4.9. The horizontal axis is the number of images and the vertical axis is the MAD value.

Table 4.11. MAD Results

MAD			
	Average	**Standard Deviation**	**Variance**
FRC	3.459894168	1.2956894	1.678811021
JPG	2.458246798	0.466134147	0.217281043
WLT	5.489784	1.710988	2.927482

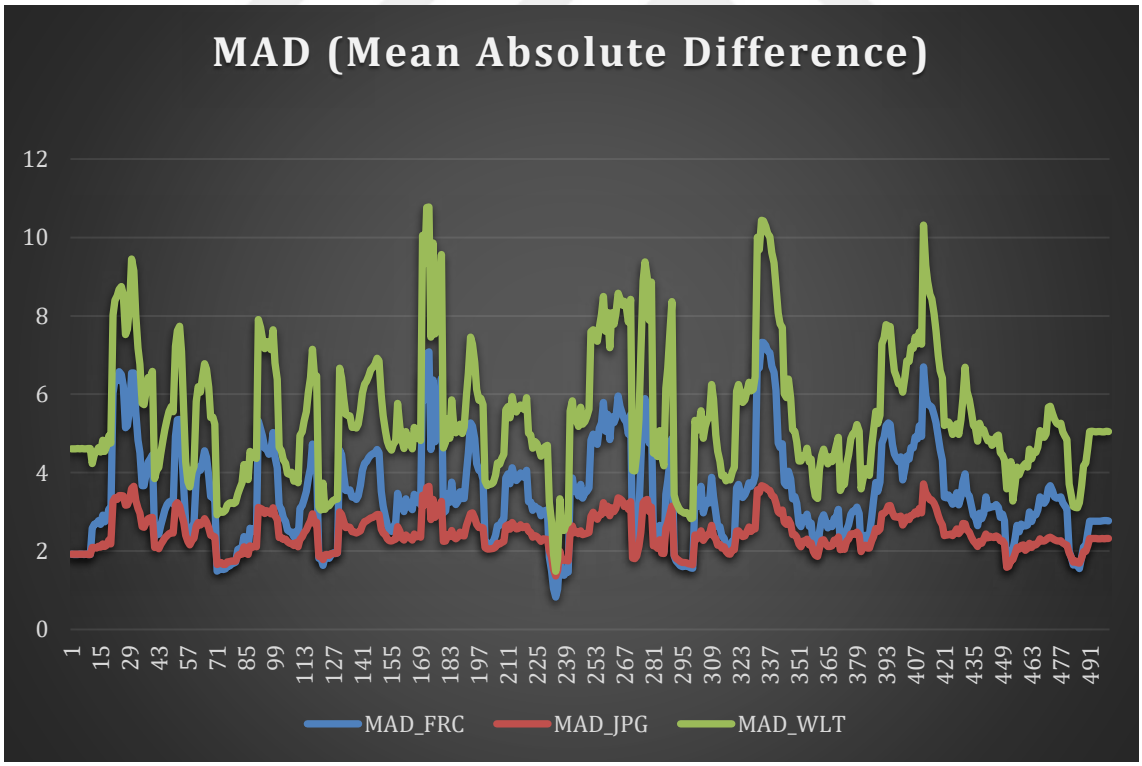

Figure 4.9. Value of MAD Results in Images

4.4.6. Structural content (SC)

SC (Structural Content) is a metric used as an image quality assessment metric. It is specifically designed to assess the structural similarity of two images and measures the degree of preservation of structural content between a reference image and a degraded image [82]. SC is often used in conjunction with other image quality metrics to assess the quality of images more comprehensively.

SC is used in image processing, especially in evaluating the effectiveness of image compression and restoration techniques. It is also used in medical imaging, satellite image analysis and other professional image processing applications. When evaluating image quality, SC is an important metric to understand whether the structural properties of the image are preserved.

SC, reference image (R) and distorted image (D) the preservation of the structural content between the two with the following formula:

$$SC = \frac{\sum_{i=1}^{N} R_i^2}{\sum_{i=1}^{N} D_i^2} \tag{4.8}$$

Here (R_i) and (D_i), respectively (R) and (D) of the footage (i)'th pixel values. This ratio indicates how much structural content the degraded image contains compared to the reference image. The higher the SC value, the closer the degraded image is structurally to the reference.

The SC results of our study are given in Table 4.12. The results obtained for each image are given in Figure 4.10. The horizontal axis is the number of images and the vertical axis is the SC value.

Table 4.12. SC Results

SC			
	Average	**Standard Deviation**	**Variance**
FRC	0.113096406	0.024778881	0.000613993
JPG	0.118179286	0.025821842	0.000666768
WLT	0.057215	0.017776	0.000316

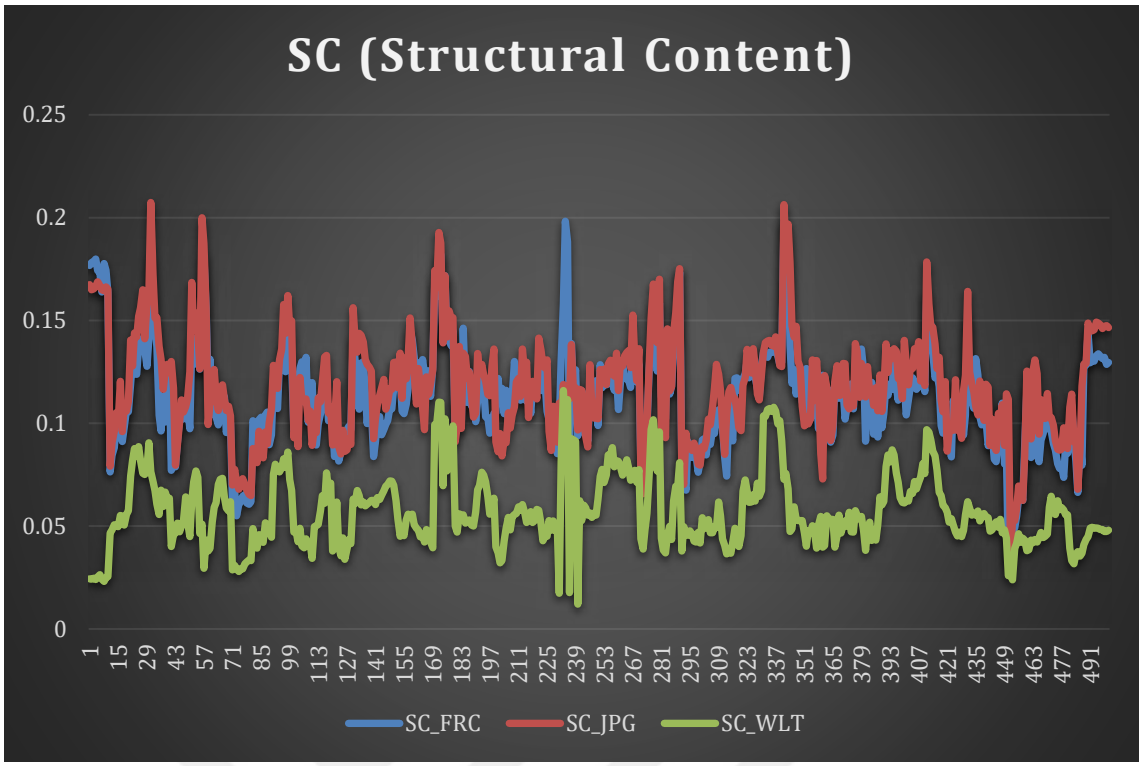

Figure 4.10. Value of SC Results in Images

4.4.7. Correlation coefficient (CC)

CC (Correlation Coefficient) is a widely used metric in statistics and data analysis. This metric expresses the strength and direction of the linear relationship between two variables [89]. The correlation coefficient takes values between -1 and +1, where +1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship and 0 indicates that there is no linear relationship between them.

The correlation coefficient is used to assess the relationship between two variables in economics, psychology, sociology, environmental science and many other fields. It measures only the linearity of the relationship between variables and does not explain the cause of the relationship or other potential types of relationships between variables.

Two variables (X) and (Y) Pearson Correlation Coefficient is usually calculated by the following formula:

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{4.9}$$

Here $(\overline{X})$ and $(\overline{Y})$ respectively (X) and (Y) represents the averages of the variables.

The CC results of our study are given in Table 4.13. The results obtained for each image are given in Figure 4.11. The horizontal axis is the number of images and the vertical axis is the CC value.

Table 4.13. CC Results

CC			
	Average	**Standard Deviation**	**Variance**
FRC	0.978120114	0.018168837	0.000330107
JPG	0.993942265	0.006187232	3.83E-05
WLT	0.951052	0.047825	0.002287

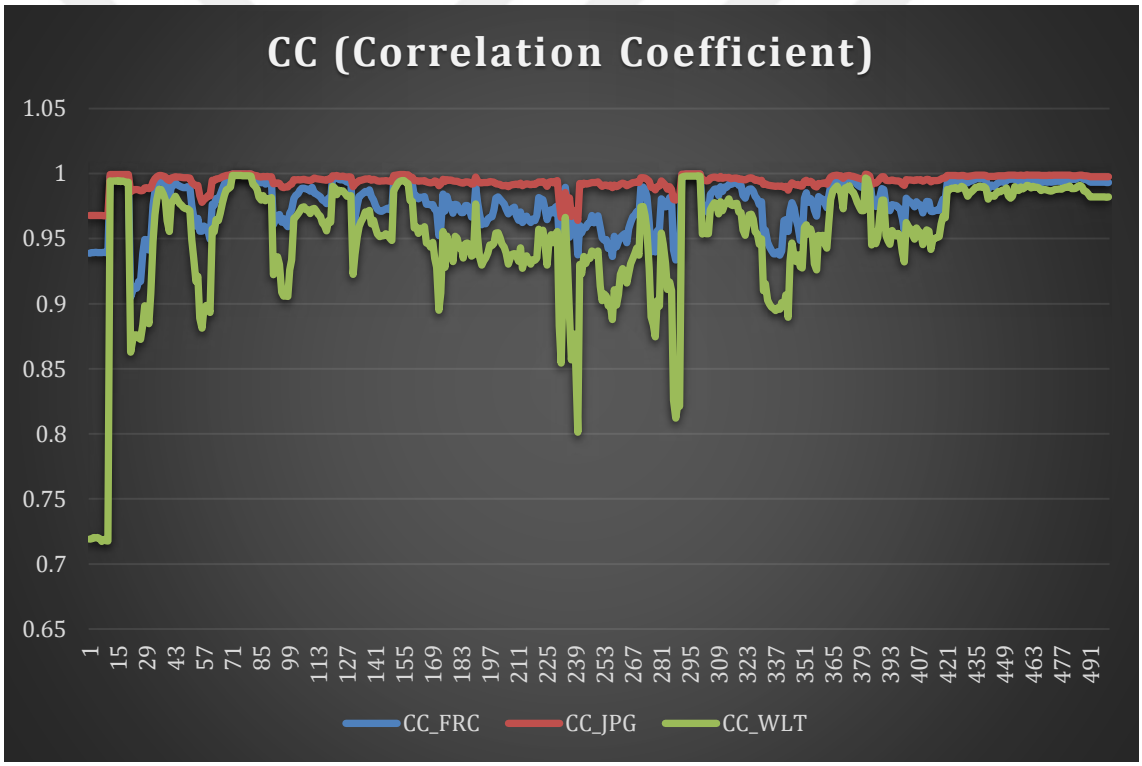

Figure 4.11. Value of CC Results in Images

4.4.8. Root mean square error (RMSE)

RMSE (Root Mean Square Error) or Root Mean Square Error is a frequently used error measurement metric [90]. RMSE is defined as the square root of the mean square root of the squares of the differences between predicted values and actual values. This metric is used to quantitatively assess the magnitude of error between actual values and predictions or

observations, especially in image processing, remote sensing, machine learning and other predictive modeling applications.

In image processing, RMSE is used to evaluate the quality of processes such as image restoration, image compression and image registration [82]. For example, RMSE can be used to measure the differences between the original and post-restoration version of an image. A low RMSE value indicates that the restoration process was of high quality, while a high RMSE value indicates that the restoration was less successful.

RMSE is calculated with the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.12}$$

Here (y_i) real values, $(\hat{y}_i)$ refers to the predicted values and (n) indicates the total number of samples.

The RMSE results of our study are given in Table 4.14. The results obtained for each image are given in Figure 4.12. The horizontal axis is the number of images and the vertical axis is the RMSE value.

Table 4.14. RMSE Results

RMSE			
	Average	**Standard Deviation**	**Variance**
FRC	5.906281559	2.072840354	4.296667133
JPG	3.308856584	0.614649952	3.78E-01
WLT	8.343580	2.325157	5.406356

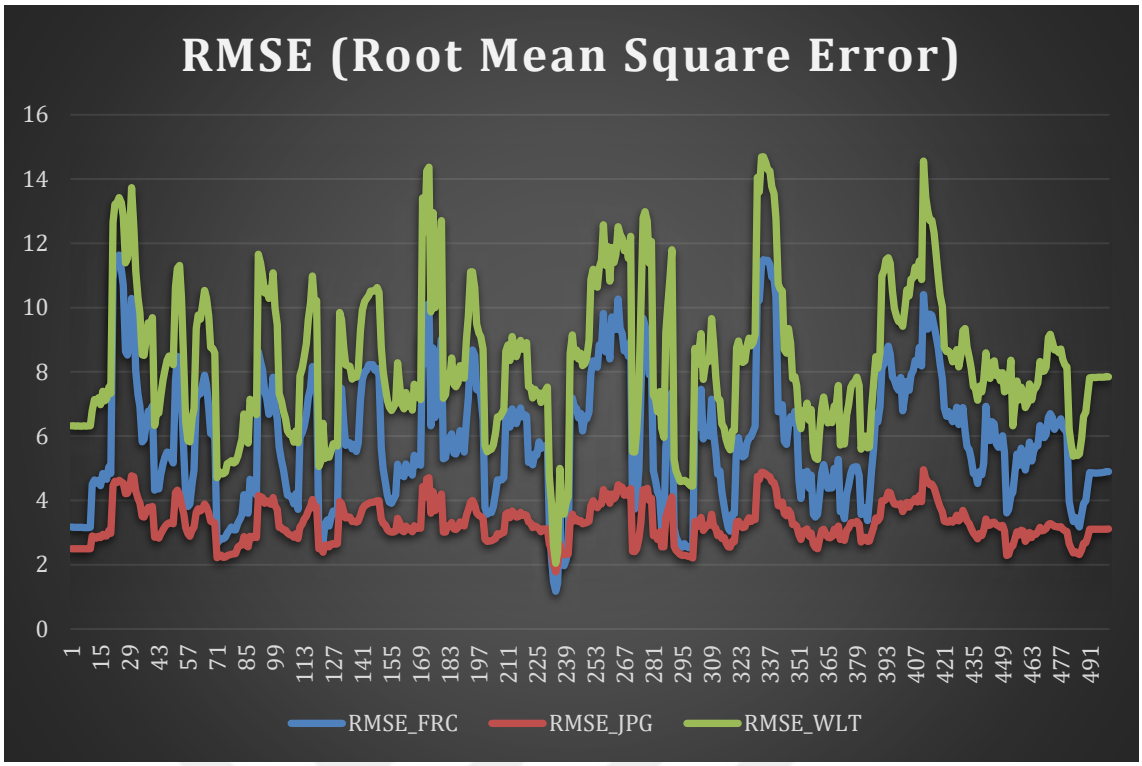

Figure 4.12. Value of RMSE Results in Images

4.4.9. Mean absolute error (MAE)

MAE (Mean Absolute Error) or Mean Absolute Error is defined as the average of the absolute values of the differences between predicted values and true values, especially in the fields of image processing, machine learning and statistical modeling. MAE is an error metric that measures how close the predicted data is to the true data. A low MAE indicates high performance of the model or processing.

In the field of image processing, MAE is used to evaluate the performance of image restoration, noise reduction and other image enhancement techniques. MAE measures the effectiveness of these processes by averaging the absolute values of the pixel-wise differences between the original and processed (after restoration or noise reduction) images [90].

MAE is calculated with the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.13}$$

Here (y_i) real values, ($\hat{y}_i$) refers to the predicted values and (n) indicates the total number of samples.

The MAE results of our study are given in Table 4.15. The results obtained for each image are given in Figure 4.13. The horizontal axis is the number of images and the vertical axis is the MAE value.

Table 4.15. MAE Results

MAE			
	Average	**Standard Deviation**	**Variance**
FRC	0.835801342	0.058168678	0.003383595
JPG	0.851891269	0.026084916	6.80E-04
WLT	0.907202	0.029985	0.000899

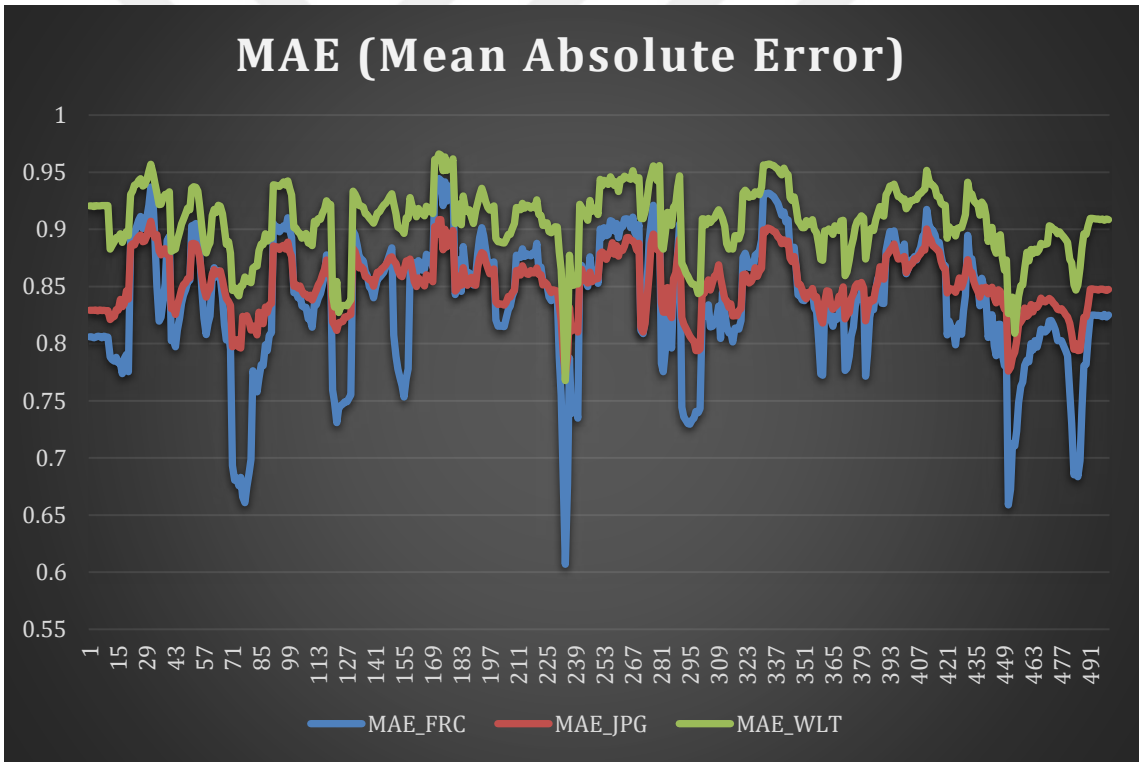

Figure 4.13. Value of MAE Results in Images

5. APPLICATION RESULTS AND DISCUSSION

5.1. Results

Table 5.1. Algorithm Results

	Comparison method	Rotation	Rblock Size	Dblock Multiplier	PSNR
1	Frobenius norm	Yes	2	2	34.69
2	Frobenius norm	Yes	2	4	31.89
3	Frobenius norm	Yes	4	2	27.58
4	Frobenius norm	Yes	4	4	26.21
5	Frobenius norm	No.	2	2	33.19
6	Frobenius norm	No.	2	4	30.63
7	Frobenius norm	No.	4	2	26.94
8	Frobenius norm	No.	4	4	25.34
9	PSNR	No.	4	4	25.81
10	Total Absolute Difference	Yes	2	2	11.73
11	Total Absolute Difference	Yes	4	4	11.34
12	Total Absolute Difference	No.	4	4	11.69

After comparing different methods, these values were selected as the optimal (time, quality and compression ratio) solution space.

As the first parameter, block angle rotation, one of the Affine transformations, was performed. When rotation was performed, it was observed that there was a loss of time but an improvement in the results obtained.

As the second parameter, we tried to determine the size of Range and Domain blocks. In the application, the smaller the size of the blocks, the better the results obtained.

Finally, 3 different comparison methods were applied to find the similarity between Range and Domain blocks. Although the best results are obtained when comparing by PSNR, it is more efficient to use the Frobenius norm comparison method since it increases the algorithm time by more than 4 times.

The Frobenius norm is a type of norm for matrices and is often used to measure the magnitude of the difference between two matrices. The Frobenius norm of a matrix is defined as the square root of the sum of the squares of all the elements of the matrix. Mathematically, the Frobenius norm of matrix A is expressed as:

$$|A|_F = \sqrt{\sum_{i,j} |a_{ij}|^2} \quad (5.1)$$

Here $(a_{ij})$of the matrix (i) line and (j) element in the column. The Frobenius norm is the Euclidean norm of a matrix or v is also known as the norm because it is directly related to the idea of transforming the elements of a matrix into a single vector and calculating the Euclidean length of that vector.

5.1.1. Properties and use of the frobenius norm

The Frobenius norm is often used to measure similarities or differences between matrices, especially in fields such as linear algebra and computer science. For example, the Frobenius norm is used to measure the magnitude of the difference between one matrix and another matrix or to determine how close a matrix is to zero. It is also used in various matrix operations in machine learning and optimization problems.

5.2. Discussion

As a result of the study, in the first stage, the methods that can be used to compare Range blocks and Domain blocks with each other in the algorithm were evaluated. As seen in the values given in Table 5.1, the best results were obtained in the Frobenius norm comparison method when other variables were kept constant.

Comparison with other methods based on PSNR value did not result in a significant change in the total image PSNR but increased the total image processing time by about two times. Therefore, it was concluded that it was not suitable for our optimal solution.

A method of subtracting pixels from each other and comparing based on the sum of the differences was tried, which reduced the compression time to about a quarter. However, a comparison of the results shows that the image quality is considerably degraded and the similarity is much reduced, even according to the human eye criterion.

In terms of other variables, an improvement of approximately 1 dB was observed in the results obtained when each Domain block was compared 4 times by rotating 0, 90, 180 and 270 degrees. Although there is no noticeable change in image quality, it is considered as a significant gain in terms of the performance of the results obtained.

Another important criterion, the Range block size and the multiplier of Domain blocks were tested. As expected, the best results were obtained for the smallest size and multiplier.

However, these block size and multiplier parameters exponentially increase the number of parts and thus exponentially increase the processing time. For this reason, our algorithm was run with block size 4 and multiplier 4 as the optimum solution.

As a result of the study, it is seen that the fractal image compression method achieves a very serious compression ratio, especially in images obtained from UAV applications. As can be seen in the results given in Table 4.7, compared to JPEG compression, which is the most well-known of the traditional methods, the fractal compression rate is around 55, while the JPEG format gives results around 48. In Wavelet compression, on the other hand, very large standard deviation values were obtained and reliable results were not possible. In the study by Riyaz and Kumari, DCT, DWT and Fractal compression results were obtained on 3 images [91]. Like our study, they obtained the results given in Table 5.2.

Table 5.2. Study Results of Riyaz and Kumari

	Metric	**DCT**	**DWT**	**FRC**
Image 1	**CR**	1.03	1.82	3.52
	PSNR	28.03	58.31	26.07
	MSE	102.9	0.09	161.84
Image 2	**CR**	1.13	1.95	3.83
	PSNR	26.86	43.11	25.78
	MSE	134.7	3.2	172.92
Image 3	**CR**	1.13	1.8	3.34
	PSNR	32.17	37.2	22.51
	MSE	36.69	12.48	367.1

Figure 5.1. Original Image

Figure 5.2. Fractal Compressed Example

Figure 5.3. JPEG Compressed Sample

When we look at the results of the measurements made on the basis of the quality of the image, the difference in quality seems to be high in pixel-based evaluation methods, such as MSE, while the difference is much less in metrics that provide results similar to human eye evaluation, such as SSIM.

As a result of examining all metrics, fractal compression method does not seem to be suitable for applications such as medical imaging where critical details are important, as it may cause data loss.

However, it can also be used in UAV applications where the end user is a human and is usually monitored on a small screen. In these applications, this method shows significant potential in terms of preserving limited resources such as bandwidth and recording space, rather than preserving image quality. As can be seen in Figure 5.1, Figure 5.2 and Figure 5.3, the quality lost by the fractal compression method can be negligible in Men in The Loop applications.

It is obvious that much more successful results can be achieved with the future studies detailed in Chapter 6.

6. FUTURE WORK

The improvements and research directions proposed in this chapter have the potential to make fractal compression technology more efficient, useful, applicable and widespread. These studies will contribute to expanding the body of knowledge in the field of fractal compression and to the wider use of this technology.

6.1. The Future of Fractal Compression Technology

Fractal compression technology is considered to be one of the leading innovative approaches in the field of digital image processing. The future of this technology has great potential in line with continuous technological developments and increasing data needs [1]. In the future, it is expected that fractal compression techniques will be developed, their application areas will be expanded and current limitations will be overcome.

6.2. Technological Innovations and Optimizations

The computational efficiency of fractal compression algorithms will be the focus of future work. By using advanced hardware and software technologies, it is possible to make these algorithms faster and more energy efficient [77], [92], [93]. Furthermore, the integration of artificial intelligence and machine learning techniques can enable automatic optimization of algorithms and performance improvements.

6.3. Expanding Application Areas

Fractal compression techniques will have wider applications, especially in areas such as video compression, real-time communications and big data analytics. The integration of this technology with mobile devices, smart city infrastructures and IoT (Internet of Things) devices can increase the efficiency of these platforms by optimizing data management.

This technology could also be useful for IR imaging systems where the amount of similarity is likely to be very high.

6.4. Sustainability and Energy Efficiency

Energy efficiency will play an important role in the development of fractal compression technology. Especially in applications where energy consumption is critical, fractal compression techniques will be optimized for low power consumption.

The future of fractal compression technology will be shaped by continuous research and development activities and is expected to have a wider range of applications and high technological capabilities. This thesis aims to contribute to the scientific literature in this field by exploring in depth the strategies and innovative applications to expand the potential of fractal compression technology.

6.5. Suggested Improvements and New Research Directions

In order to overcome the current limitations of fractal compression technology and to further expand the potential of this technology, the proposed improvements and research directions aim to make this technology more effective and efficient. This chapter presents some critical paths for future research in the field of fractal compression and suggestions for improvement.

6.6. Algorithm Optimization

Improving the computation time of fractal compression algorithms is an important area of research, especially for real-time applications. Making algorithms faster and more energy efficient could expand the application areas of this technology.

6.7. Artificial Intelligence Integration

The integration of artificial intelligence and machine learning techniques into fractal compression processes can offer great opportunities, especially in pattern recognition and automatic parameter tuning. This integration can improve the quality of compression and improve the adaptability of algorithms.

6.8. Advanced Resolution Independence Studies

Further development of the resolution independence property of fractal compression could provide significant improvements for high-resolution images. In particular, this would improve image compatibility across different platforms and devices.

6.9. Cross Disciplinary Practices

Research into applications of fractal compression techniques in different disciplines, such as video processing, medical imaging and remote sensing, can expand the reach of this technology. These applications offer the opportunity to test the effectiveness of fractal

compression techniques on various types of data and to determine how these techniques can be adapted to specific needs.

REFERENCES

[1] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial intelligence for decision making in the era of Big Data – evolution, challenges and research agenda," *Int. J. Inf. Manag.*, vol. 48, pp. 63–71, Oct. 2019, doi: 10.1016/j.ijinfomgt.2019.01.021.

[2] W. Huang *et al.*, "Joint Power, Altitude, Location and Bandwidth Optimization for UAV With Underlaid D2D Communications," *IEEE Wirel. Commun. Lett.*, Apr. 2019, doi: 10.1109/lwc.2018.2878706.

[3] J. Ji, K. Zhu, C. Yi, and D. Niyato, "Energy Consumption Minimization in UAV-Assisted Mobile-Edge Computing Systems: Joint Resource Allocation and Trajectory Design," *IEEE Internet Things J.*, May 2021, doi: 10.1109/jiot.2020.3046788.

[4] N. M. P. S. Chavan, N. M. P. S.Bhore, N. M. M. K. Kute, and N. M. S. J. Patil, "Survey on Various Image Compression Techniques Used in Image Processing to Improve the Quality of Image," *Int. J. Adv. Res. Sci. Commun. Technol.*, Jun. 2022, doi: 10.48175/ijarsct-5124.

[5] S. Keyworth and S. Wolfe, "UAVS for land use applications: UAVs in the civilian airspace institution of engineering and technology," in *IET Seminar on UAVs in the Civilian Airspace*, Mar. 2013, pp. 1–13. doi: 10.1049/ic.2013.0071.

[6] H. Qu, W. Zhang, J. Zhao, Z. Luan, and C. Chang, "Rapid Deployment of UAVs Based on Bandwidth Resources in Emergency Scenarios," in *2020 Information Communication Technologies Conference (ICTC)*, May 2020, pp. 86–90. doi: 10.1109/ICTC49638.2020.9123274.

[7] B. Senapatil and S. Kisan, "Color image compression using fractal geometry," Feb. 2018, doi: 10.1109/icsns.2018.8573638.

[8] G. Lu, "Fractal image compression," *Signal Process. Image Commun.*, Oct. 1993, doi: 10.1016/0923-5965(93)90055-x.

[9] S. Liu, W. Bai, N. Zeng, and S. Wang, "A Fast Fractal Based Compression for MRI Images," *IEEE Access*, Jan. 2019, doi: 10.1109/access.2019.2916934.

[10] A. H. Ali, A. N. Abbas, L. E. George, and M. R. Mokhtar, “Image and audio fractal compression: comprehensive review, enhancements and research directions,” *Indones. J. Electr. Eng. Comput. Sci.*, Sep. 2019, doi: 10.11591/ijeecs.v15.i3.pp1564-1570.

[11] Y. Zeng, J. Xu, and R. Zhang, “Energy Minimization for Wireless Communication With Rotary-Wing UAV,” *IEEE Trans. Wirel. Commun.*, Apr. 2019, doi: 10.1109/twc.2019.2902559.

[12] gd-admin, “Miniature OEM Tri-band Digital IP MESH Data Link,” iwavecomms.com/. Accessed: Jul. 06, 2024. [Online]. Available: iwavecomms.com:443/miniature-oem-tri-band-digital-ip-mesh-data-link-product/

[13] J. Hemanth *et al.*, “Computer vision and data storage in UAVs,” 2020, pp. 23–45. doi: 10.1049/PBCE120F_ch2.

[14] Q. Song, S. Jin, and F.-C. Zheng, “Completion Time and Energy Consumption Minimization for UAV-Enabled Multicasting,” *IEEE Wirel. Commun. Lett.*, Jun. 2019, doi: 10.1109/lwc.2019.2894684.

[15] “Google Books Link.” Accessed: Jul. 06, 2024. [Online]. Available: books.google.com.tr/books?id=gz6swuOPylMC

[16] B. Wohlberg and G. De Jager, “A review of the fractal image coding literature,” *IEEE Trans. Image Process.*, vol. 8, no. 12, pp. 1716–1729, Dec. 1999, doi: 10.1109/83.806618.

[17] M. F. Barnsley, *Fractals Everywhere*. Courier Corporation, 2012.

[18] Q. Wang, “Fractal Image Encoding Based on Minimum Iterated Function System,” Jul. 2023, doi: 10.1109/icivc58118.2023.10270655.

[19] R. Lopes and N. Betrouni, “Fractal and multifractal analysis: A review,” *Med. Image Anal.*, vol. 13, no. 4, pp. 634–649, Aug. 2009, doi: 10.1016/j.media.2009.05.003.

[20] B. B. Mandelbrot, A. Blumen, M. Fleischmann, D. J. Tildesley, and R. C. Ball, “Fractal geometry: what is it, and what does it do?,” *Proc. R. Soc. Lond. Math. Phys. Sci.*, vol. 423, no. 1864, pp. 3–16, Jan. 1997, doi: 10.1098/rspa.1989.0038.

[21] K. Jaferzadeh, I. Moon, and S. Gholami, “Enhancing fractal image compression speed using local features for reducing search space,” *Pattern Anal. Appl.*, vol. 20, no. 4, pp. 1119–1128, Nov. 2017, doi: 10.1007/s10044-016-0551-1.

[22] M. Barni, *Document and Image Compression*. CRC Press, 2018.

[23] J. M. Bogoya, A. Vargas, and O. Schütze, “The Averaged Hausdorff Distances in Multi-Objective Optimization: A Review,” *Mathematics*, vol. 7, no. 10, Art. no. 10, Oct. 2019, doi: 10.3390/math7100894.

[24] A. E. Jacquin, “Image coding based on a fractal theory of iterated contractive image transformations,” *IEEE Trans. Image Process.*, vol. 1, no. 1, pp. 18–30, Jan. 1992, doi: 10.1109/83.128028.

[25] S. Liu, Z. Pan, W. Fu, and X. Cheng, “Fractal generation method based on asymptote family of generalized Mandelbrot set and its application,” *J. Nonlinear Sci. Appl.*, vol. 10, pp. 1148–1161, Mar. 2017, doi: 10.22436/jnsa.010.03.24.

[26] R. DeLorto, “Fractal dimension and Julia sets”.

[27] X. Tang, D. Yu, and J. Yan, “A Study of Sierpinski Fractals Based on Regular Polygons and Circles,” *J. Appl. Math. Comput.*, pp. 16–27, Jan. 2024, doi: 10.26855/jamc.2024.03.0003.

[28] A. Somalwar, “Analysis of Fractals from a Mathematical and Real-World Perspective,” 2016. Accessed: Jul. 08, 2024. [Online]. Available: semanticscholar.org/paper/Analysis-of-Fractals-from-a-Mathematical-and-Somalwar/ef82cda13139fd56ec89959c9095c25035c35775

[29] M. Fitzsimmons and H. Kunze, “Small and minimal attractors of an IFS,” *Commun. Nonlinear Sci. Numer. Simul.*, Jun. 2020, doi: 10.1016/j.cnsns.2020.105227.

[30] U. P. Dolhare and V. V. Nalawade, “FIXED POINT THEOREMS IN DIGITAL IMAGES AND APPLICATIONS TO FRACTAL IMAGE COMPRESSION,” 2018.

[31] U. Nandi and J. K. Mandal, “Fractal image compression by using loss-less encoding on the parameters of affine transforms,” in *2014 First International Conference on Automation, Control, Energy and Systems (ACES)*, Feb. 2014, pp. 1–6. doi: 10.1109/ACES.2014.6807984.

[32] H. T. Chang, “Arbitrary affine transformation and their composition effects for two-dimensional fractal sets,” *Image Vis. Comput.*, vol. 22, no. 13, pp. 1117–1127, Nov. 2004, doi: 10.1016/j.imavis.2004.05.003.

[33] M. F. Barnsley, “Transformations Between Fractals,” in *Fractal Geometry and Stochastics IV*, C. Bandt, M. Zähle, and P. Mörters, Eds., Basel: Birkhäuser, 2009, pp. 227–250. doi: 10.1007/978-3-0346-0030-9_8.

[34] D. Viswanath, “The fractal property of the Lorenz attractor,” *Phys. Nonlinear Phenom.*, vol. 190, no. 1–2, pp. 115–128, Mar. 2004, doi: 10.1016/j.physd.2003.10.006.

[35] B.-C. Anghelina and R. Miculescu, “On the localization of Hutchinson–Barnsley fractals,” *Chaos Solitons Fractals*, vol. 173, p. 113674, Aug. 2023, doi: 10.1016/j.chaos.2023.113674.

[36] C. Fu and Y. Chen, “Study on Fractal Images Construction with Topology Invariance IFS Attractor,” in *2009 International Forum on Computer Science-Technology and Applications*, Dec. 2009, pp. 294–297. doi: 10.1109/IFCSTA.2009.77.

[37] M. F. Barnsley and J. Hutchinson, “New Methods in Fractal Imaging,” in *International Conference on Computer Graphics, Imaging and Visualisation (CGIV’06)*, Jul. 2006, pp. 296–301. doi: 10.1109/CGIV.2006.65.

[38] S. Jampala, “Fractals: classification, generation and applications,” in *[1992] Proceedings of the 35th Midwest Symposium on Circuits and Systems*, Aug. 1992, pp. 1024–1027 vol.2. doi: 10.1109/MWSCAS.1992.271120.

[39] Y.-G. Wu, “FAST FRACTAL IMAGE ENCODER DESIGN,” *Synchroinfo J.*, Jan. 2021, doi: 10.36724/2664-066x-2021-7-4-40-44.

[40] I. Association, R. Dobrescu, and D. Popescu, “Image Processing Applications Based on Texture and Fractal Analysis,” 2013, pp. 235–259. doi: 10.4018/978-1-4666-3994-2.ch014.

[41] J. Kominek, “Algorithm for Fast Fractal Image Compression,” *Proc. SPIE - Int. Soc. Opt. Eng.*, Mar. 2000, doi: 10.1117/12.206368.

[42] J. Martínez Rojas, J. Alpuente, I. Rojas, and S. Peña, “Fractal-based image enhancement techniques inspired by differential interference contrast microscopy,” *J. Opt. Pure Appl. Opt.*, vol. 11, p. 065503, Mar. 2009, doi: 10.1088/1464-4258/11/6/065503.

[43] V. V. Bulkunde, N. P. Bodne, and S. Kumar, “Implementation of Fractal Image Compression on Medical Images by Different Approach,” *Int. J. Trend Sci. Res. Dev.*, Jun. 2019, doi: 10.31142/ijtsrd23768.

[44] T. Singh and T. Babu, “Fractal Image Processing and Analysis for Compression of Hyperspectral Images,” Jul. 2019, doi: 10.1109/icccnt45670.2019.8944503.

[45] A. K. Biswas, S. Karmakar, and S. Sharma, “Performance analysis of a new fractal compression method for medical images based on fixed partition,” *Int. J. Inf. Technol.*, Feb. 2021, doi: 10.1007/s41870-020-00598-3.

[46] A. Fathi and N. S. Abduljabbar, “Survey on Fractal image compression,” *J. Res. Sci. Eng. Technol.*, Sep. 2020, doi: 10.24200/jrset.vol8iss3pp5-8.

[47] S. Poobal and G.RAVINDRAN, “The performance of fractal image compression on different imaging modalities using objective quality measures,” *Int. J. Eng. Sci. Technol.*, vol. 3, Jan. 2011.

[48] G. Kaur, F. Hitashi, and G. Singh, “PERFORMANCE EVALUATION OF IMAGE QUALITY BASED ON FRACTAL IMAGE COMPRESSION,” *Int. J. Comput. Technol.*, vol. 2, Jan. 2003, doi: 10.24297/ijct.v2i1.2608.

[49] T. M. Hasan and X. Wu, “An Adaptive Algorithm for Improving the Fractal Image Compression (FIC),” *J. Multimed.*, vol. 6, no. 6, pp. 477–485, Dec. 2011, doi: 10.4304/jmm.6.6.477-485.

[50] P. Sharma and D. R. Mahajan, “A REVIEW ON COMPRESSION TECHNIQUES WITH RUN LENGTH ENCODING,” vol. 2, no. 8, 2013.

[51] R. Arshad, A. Saleem, and D. Khan, “Performance comparison of Huffman Coding and Double Huffman Coding,” in *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, Aug. 2016, pp. 361–364. doi: 10.1109/INTECH.2016.7845058.

[52] M. K. Mishra, T. K. Mishra, and A. K. Pani, “Parallel Lempel-Ziv-Welch (PLZW) Technique for Data Compression,” vol. 3, 2012.

[53] M. Smadi and Q. Abu Al-Haija, “Modified Lempel–Ziv Welch Source Coding Algorithm for Efficient Data Compression,” *J. Theor. Appl. Inf. Technol.*, vol. 61, pp. 200–205, Mar. 2014.

[54] M. W. Spratling, “A review of predictive coding algorithms,” *Brain Cogn.*, vol. 112, pp. 92–97, Mar. 2017, doi: 10.1016/j.bandc.2015.11.003.

[55] M. Al-Ani and F. Awad, “THE JPEG IMAGE COMPRESSION ALGORITHM,” *Int. J. Adv. Eng. Technol.*, vol. 6, pp. 1055–1062, May 2013.

[56] R. A.M, K. W.M, E. M. A, and W. Ahmed, “Jpeg Image Compression Using Discrete Cosine Transform - A Survey,” *Int. J. Comput. Sci. Eng. Surv.*, vol. 5, no. 2, pp. 39–47, Apr. 2014, doi: 10.5121/ijcses.2014.5204.

[57] X. Li, S. Zhang, and H. Zhao, “A Fast Image Compression Algorithm Based on Wavelet Transform,” *Int. J. Circuits Syst. Signal Process.*, Jul. 2021, doi: 10.46300/9106.2021.15.89.

[58] M. M. H. Chowdhury and A. Khatun, “Image Compression Using Discrete Wavelet Transform,” vol. 9, no. 4, 2012.

[59] J. S. Walker and T. Q. Nguyen, “Chapter 6 – Wavelet-Based Image Compression,” 2001.

[60] F. J. Hernandez-Lopez and O. Muñiz-Pérez, “Parallel fractal image compression using quadtree partition with task and dynamic parallelism,” *J. Real-Time Image Process.*, Jan. 2022, doi: 10.1007/s11554-021-01193-w.

[61] Hasanujjaman, A. Banerjee, U. Biswas, and M. K. Naskar, “Fractal Image Compression of an Atomic Image using Quadtree Decomposition,” *2019 Devices Integr. Circuit DevIC*, Mar. 2019, doi: 10.1109/devic.2019.8783961.

[62] A. A. Sideiri, N. Alzeidi, M. Hammoshi, M. S. Chauhan, and G. Alfarsi, “CUDA implementation of fractal image compression,” *J. Real-Time Image Process.*, Jul. 2019, doi: 10.1007/s11554-019-00894-7.

[63] B. S. Z. Abood, H. A. R. Akkar, and A. S. Al-Safi, “Fast Full-Search Algorithm of Fractal Image Compression for Acceleration Image Processing,” *Sci. J. Univ. Zakho*, Feb. 2023, doi: 10.25271/sjuoz.2023.11.1.1122.

[64] Hasanujjaman, U. Biswas, and M. K. Naskar, “Hardware Architecture of a Decoder for Fractal Image Compression,” Mar. 2019, doi: 10.1109/devic.2019.8783667.

[65] K. Gdawiec and D. Domańska, “Partitioned iterated function systems with division and a fractal dependence graph in recognition of 2D shapes,” in *International Journal of Applied Mathematics and Computer Science*, Dec. 2011, pp. 757–767. doi: 10.2478/v10006-011-0060-8.

[66] S. Mitra, C. Murthy, and M. Kundu, “A Study on Partitioned Iterative Function Systems for Image Compression,” *Fundam. Math.*, vol. 34, pp. 413–428, Jul. 1998, doi: 10.3233/FI-1998-34405.

[67] R. Sultana, N. Ahmed, and S. Basha, “Advanced Fractal Image Coding Based on the Quadtree,” vol. 2, Aug. 2011.

[68] K. M. S. Soyjaudah and I. Jahmeerbacus, “Fractal Image Compression Using Quadtree Partitioning,” *Int. J. Electr. Eng. Educ.*, vol. 39, no. 1, pp. 71–86, Jan. 2002, doi: 10.7227/IJEEE.39.1.7.

[69] V. Muneeswaran, P. Nagaraj, K. P. Sai, E. A. Kumar, and S. R. Chanakya, “Enhanced Image Compression Using Fractal and Tree Seed-Bio Inspired Algorithm,” Aug. 2021, doi: 10.1109/icesc51422.2021.9532850.

[70] Z. J. Ahmed, L. E. George, and Z. S. Abduljabbar, “Fractal Image Compression Using Block Indexing Technique: A Review,” *Iraqi J. Sci.*, Jul. 2020, doi: 10.24996/ijs.2020.61.7.29.

[71] S. R. Khaitu and S. P. Panday, “Fractal Image Compression Using Canonical Huffman Coding,” *J. Inst. Eng.*, Feb. 2020, doi: 10.3126/jie.v15i1.27718.

[72] R. Gupta, D. Mehrotra, and R. K. Tyagi, “Hybrid edge-based fractal image encoding using K-NN search,” *Multimed. Tools Appl.*, Mar. 2022, doi: 10.1007/s11042-022-12631-7.

[73] A.-M. H. Y. Saad *et al.*, “Deep Pipeline Architecture for Fast Fractal Color Image Compression Utilizing Inter-Color Correlation,” *IEEE Access*, Jan. 2022, doi: 10.1109/access.2022.3213723.

[74] S. Padmavati and V. Meshram, “FPGA Implementation for Fractal Quadtree Image Compression,” Oct. 2018, doi: 10.26438/ijcse/v6i10.405409.

[75] S. Khatun and A. Iqbal, “A Review of Image Compression Using Fractal Image Compression with Neural Network,” *Soc. Sci. Res. Netw.*, Jan. 2018, doi: 10.2139/ssrn.3531802.

[76] R. Gupta, D. Mehrotra, and R. K. Tyagi, “Comparative analysis of edge-based fractal image compression using nearest neighbor technique in various frequency domains,” *Alex. Eng. J.*, Sep. 2018, doi: 10.1016/j.aej.2017.03.038.

[77] B. Varghese and S. Krishnakumar, “Fast Fractal Coding of MRI Images using Deep Reinforcement Learning,” *Int. J. Adv. Comput. Sci. Appl.*, Jan. 2021, doi: 10.14569/ijacsa.2021.0120492.

[78] D. W. Naaman, “Image Compression Technique Based on Fractal Image Compression Using Neural Network – A Review,” *Asian J. Res. Comput. Sci.*, Jul. 2021, doi: 10.9734/ajrcos/2021/v10i430249.

[79] “Foxtech BABY SHARK VTOL 260 for Inspection Mapping and Survey.” Accessed: Jun. 22, 2024. [Online]. Available: https://www.foxtechfpv.com/foxtech-baby-shark-vtol.html

[80] “FCB-EV7520.” Accessed: Jun. 22, 2024. [Online]. Available: image-sensing-solutions.eu/FCB-EV7520.html

[81] “BMP file format,” *Wikipedia*. Feb. 02, 2024. Accessed: Jul. 08, 2024. [Online]. Available:en.wikipedia.org/w/index.php?title=BMP_file_format&oldid=1202504744

[82] V.-I. Ungureanu, P. Negirla, and A. Korodi, “Image-Compression Techniques: Classical and ‘Region-of-Interest-Based’ Approaches Presented in Recent Papers,” *Sensors*, vol. 24, no. 3, Art. no. 3, Jan. 2024, doi: 10.3390/s24030791.

[83] D. Liu, D. Wang, and H. Li, “Recognizable or Not: Towards Image Semantic Quality Assessment for Compression,” *Sens. Imaging*, vol. 18, no. 1, p. 1, Dec. 2016, doi: 10.1007/s11220-016-0152-5.

[84] “A Formal Evaluation of PSNR as Quality Measurement Parameter for Image Segmentation Algorithms,” ar5iv. Accessed: Jul. 08, 2024. [Online]. Available: ar5iv.labs.arxiv.org/html/1605.07116

[85] N. Liu and G. Zhai, “Free Energy Adjusted Peak Signal to Noise Ratio (FEA-PSNR) for Image Quality Assessment,” *Sens. Imaging*, vol. 18, no. 1, p. 11, Feb. 2017, doi: 10.1007/s11220-017-0160-0.

[86] K. Gu, S. Wang, G. Zhai, S. Ma, X. Yang, and W. Zhang, “Content-weighted mean-squared error for quality assessment of compressed images,” *Signal Image Video Process.*, vol. 10, no. 5, pp. 803–810, Jul. 2016, doi: 10.1007/s11760-015-0818-9.

[87] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.

[88] K. Ohashi *et al.*, “Applicability Evaluation of Full-Reference Image Quality Assessment Methods for Computed Tomography Images,” *J. Digit. Imaging*, vol. 36, no. 6, pp. 2623–2634, Dec. 2023, doi: 10.1007/s10278-023-00875-0.

[89] J. li and W. Dai, “Image Quality Assessment Based on the Correlation coefficient and the 2-D Discrete Wavelet Transform,” *Proc. 2009 IEEE Int. Conf. Autom. Logist. ICAL 2009*, Aug. 2009, doi: 10.1109/ICAL.2009.5262815.

[90] T. Samajdar and Md. I. Quraishi, “Analysis and Evaluation of Image Quality Metrics,” in *Information Systems Design and Intelligent Applications*, J. K. Mandal, S. C. Satapathy, M. Kumar Sanyal, P. P. Sarkar, and A. Mukhopadhyay, Eds., New Delhi: Springer India, 2015, pp. 369–378. doi: 10.1007/978-81-322-2247-7_38.

[91] “IJSRD - International Journal for Scientific Research & Development| Vol. 5, Issue 07, 2017 | ISSN (online): 2321-0613,” vol. 5, no. 08.

[92] Hasanujjaman, U. Biswas, and M. K. Naskar, “Controlled hardware architecture for fractal image compression,” *Int. J. Nano Biomater.*, Jan. 2020, doi: 10.1504/ijnbm.2020.10029634.

[93] A.-M. H. Y. Saad and M. Z. Abdullah, “High-Speed Fractal Image Compression Featuring Deep Data Pipelining Strategy,” *IEEE Access*, Jan. 2018, doi: 10.1109/access.2018.2880480.