DOI: 10.24996/ijs.2026.67.1.38

ISSN: 0067-2904

Enhanced Fractal Image Compression using Quadtree Partitioning and Double Moment Descriptors

Israa S. Rasheed*, Bushra A. Sultan
Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Received: 11/9/2024 Accepted: 15/1/2025 Published: xx

Abstract

Fractal Image Compression (FIC) is a robust method for reducing multimedia applications' data storage requirements and communication costs. This research introduces a partitioning strategy based on quadtree as variable length block partitioning with Prewitt Operator as a criterion for guiding the partitioning decision. The proposed system achieves significant time savings while increasing the Compression Ratio and identifies a near-optimal PSNR compared with the previous studies. The increase in PSNR reached 25% at its highest value while maintaining the compression ratio, which increased to 30% without a significant change in the time required for implementation. Additionally, a novel set of moment descriptors is introduced that developed in integration with the last set to highlight specific block regions, enhancing overall computational efficiency. The system operates on color images with different image sizes 512 × 512 pixels and 256 × 256 pixels, and color depth is 24 bits. This was also compared to verify the superiority of the results by the image size in pixels. Experimental findings validate the efficacy of the enhanced FIC algorithm, demonstrating enhanced encoding speeds and improved image quality in the reconstructed output.

Keywords: Fractal image Compression, Quadtree, Prewitt, Block Indexing, Moment Descriptor.

تقسيم رباعي معتمد على Quadtree قائم على مشغل Prewitt مع الوصف المزدوج لتحسين نظام ضغط الصور الكسورية

إسراء سالم رشيد*, بشرى عبدالله سلطان
قسم علوم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق

الخلاصة

إن ضغط الصور الكسرية (FIC) هي طريقة قوية لتقليل متطلبات تخزين البيانات وتكاليف الاتصال في تطبيقات الوسائط المتعددة. تقدم هذه الدراسة استراتيجية تقسيم تعتمد على Quadtree كتقسيم كتلة بطول متغير مع مشغل Prewitt كمعيار لتوجيه قرار التقسيم. يحقق النظام المقترح توفيرا كبيرا للوقت ويحدد نسبة ضغط شبه مثالية مقارنة بالدراسة السابقة وصلت الزيادة في نسبة إشارة إلى ضوضاء إلى 25٪ عند أعلى قيمة لها، مع الحفاظ على نسبة الضغط، والتي زادت إلى 30٪ دون تغيير كبير في الوقت المطلوب للتنفيذ. بالإضافة إلى ذلك، يتم تقديم مجموعة جديدة من أوصاف الخصائص التي تم تطويرها بالتكامل مع المجموعة السابقة

*Email: esraa.salem2101m@sc.uobaghdad.edu.iq

لتسليط الضوء على مناطق كتلة معينة، مما يعزز الكفاءة الحسابية الشاملة. تمت أيضًا مقارنة آلية تشغيل النظام على أحجام صور مختلفة حيث يعمل النظام على صور ملونة بأحجام صور مختلفة وهي 512 × 512 بكسل و256 × 256 بكسل وعمق لون 24 بت. للتحقق من تفوق النتائج وفقًا لحجم الصورة بالبيكسل. تثبت النتائج التجريبية فعالية خوارزمية FIC المحسنة، مما يوضح سرعات ترميز محسّنة وجودة صورة محسنة في الناتج المعاد بناؤه.

I.Introduction

The daily interest in multimedia usage, such as digital images and video, has led to vast development in research into compression techniques. The growth of better-quality and lower-priced picture acquisition technologies has caused tremendous attention to both the size and resolution of the image and, as a result, the construction of optimal compression methods. Although the capacity for storage and bandwidth of transferring has accordingly increased in recent years, many applications still need compression [1].

Image compression reduces the required bits to represent an image by efficiently exploiting redundancy in the image itself. Redundancy utilization can be purely statistical or combined with psycho-visual effects [2].

Image compression schemes are generally categorized as lossless and lossy compression schemes. A lossless compression scheme is a compression with no error where the original data can be retrieved after decompression. A low compression ratio is provided by this scheme, but it has several applications, like image compression systems used in the compression of medical images, in which it is unacceptable to lose any information. In lossy compression, during the compression process, some parts of the original data are wasted; hence, after the decompression process, an approximate amount of the original data is retrieved [3]. The goal of an image compression algorithm is to employ redundancy in an image so that the smallest number of bits still represents the image, nevertheless maintaining the acceptable visual quality of the decompressed image [4] [5]. Storing images pixel by pixel is the simplest way, but it is complicated. A larger image requires more storage space. Instead of storing pixel values directly, different encoding schemes are acquired. These encoding systems include Huffman encoding and GIF. Both are lossless schemes. Other algorithms cause the image to lose data, but they reduce storage space. These algorithms include Fourier transform, cosine transform, JPEG, and fractal image compression [6].

This work proposes an improved image compression strategy that employs Fractal Image Compression (FIC), utilizing symmetry prediction and block indexing with a new set of moment descriptors. The designed FIC employs a quadtree as a variable range block portioning mechanism, and the partitioning decision is based on the Prewitt Operator. The precise algorithm used is fractal with block indexing. This is applied on the well known color images with different image sizes in which size 512 × 512 pixels and size 256 × 256 pixels, and a color depth of 24 bit. The metrics used to evaluate performance are mean square error (MSE) and peak signal-to-noise ratio (PSNR) measured in dB. Compression Ratio (CR) and bit rate (BR) parameters were also used.

The following subsequent sections include: Fractal Image Compression, Related Work, Fractal image compression using block indexing technique, Test Results, Conclusion and Future work.

II.Fractal image compression

The partition iterated function system (PIFS), on which fractal image coding is based, divides an original input image into a collection of non-overlapping sub-blocks, known as range blocks (R), that collectively cover the entire image. Every range block has a size of N X N. In addition, the original picture is divided into a collection of additional overlapping sub-blocks known as domain blocks (D), each of which has a size double that of a range block. It is not required for the domain blocks to fill the entire image; they are free to overlap [7].

Second, to make each domain block the same size as the range block, they are all shrunk using pixel averaging or down sampling. An extended domain pool is created by applying eight symmetrical transformations (rotations and flips) to each contracted domain block. This is represented as follows: for every range block, we search the domain pool to find the best-matched domain block D with a contractive affine transformation. The extremely computationally complex encoding process is the issue with fractal coding [8]. The optimal fitting discovery of range blocks and multiple domain blocks takes up most of the encoding time, making fractal encoding a costly procedure that severely restricts the algorithm's practical applications.

Although it has many advantages, Fractal image encryption implementation has several common problems. These problems have been treated as challenges in the appropriateness of FC and today's popular compression technology. For further research, these challenges are the motivation in both areas and can be listed as follows [9] [10]:

a) Create range and domain blocks by employing various partitioning methods.

b) The process of matching range-domain field blocks to find the best likeness between them; then,, measurements are used to determine the minimum distortion.

c) Reduce encoding time by adopting acceleration techniques to reduce the time required.

d) Reduce the domain pool to reduce calculations in the matching process.

III. Related Work

Geroge [11] proposed a method that uses a moment indexing block and halting condition to speed up the matching process for FIC. The author suggests using an extra filtering and partitioning scheme to speed up and boost the compression ratio. This method doubles the encoding speed by ten times without sacrificing image quality.

Al-Hilo and George propose expediting the compression of fractal-colored images [12]. To boost the compression rate and utilize moment characteristics as a descriptor for range and domain blocks, they substitute the (Y, U, V) component for the (R, G, B) component with a 24-bit/pixel resolution. This expedites the fractal encoding process. The speed that is being offered is around 96% faster than regular FIC.

Kovacs [13] created another classification using the following two parameters: Normalized Root Mean Square Error (NRMS) and Approximate First Derivative (AFD). The similarity between picture blocks is measured using these metrics.

A technique that was previously proposed in [14] was improved by George and Al-Hilo [15]. Using DPCM and shift coding to encode the range mean and scale parameters results in a 3% increase in compression ratio. Furthermore, this technique improves the PSNR by around 5.3% and reduces encoding time by 66%.

Additionally, George and Al-Hilo [14] suggested a speed-up strategy that used first-order centralized moments with the predictor to minimize the number of symmetry transformations of the domain block from 8 to 1. This predictor provides the necessary block transform to achieve optimal range-domain matching and minimize the encoding time. Compared to the usual approach, this method speeds up the encoding process by almost seven times while maintaining the compression ratio and PSNR.

By using the correlation information feature to create range-domain block matching near neighbors in the space, Wang et al. [7] established a fast FC technique. With the preservation of the encoding time and compression ratio, this characteristic produces a superior reconstruction of picture quality compared to previous fast fractal coding research.

Hasan and Wu [16] introduced the Adaptive Fractal Image Compression method (AFIC) based on multiple strategies to reduce the complexity of the matching process. The strategies are Adaptive Quadtree Partitioning Technique (AQPT), Zero Mean Intensity Level (ZMIL), Reducing Domain Image Size (RDIS), Range Exclusion (RE), and Variance Domain Selection

(VDS***).*** These strategies work together to improve the compression ratio while also reducing encoding time. Compared to previous comparable research, this approach reduced time while decreasing reconstructed image quality.

Wang and Zheng [17] presented a novel FIC schema in which the Absolute Pearson's Correlation Coefficient (APCC) is utilized to categorize the range and domain blocks. Further, they grouped the domain blocks into groups using the APCC to accelerate the matching process between range and domain blocks.

Wang et al. suggested a rapid fractal encryption technique using Standard Deviation (STD) and Discrete Cosine Transform (DCT) [18] To restrict searches in the domain pool. They also employed the Auxiliary Encoding Algorithm (AEA) to increase the reconstructed image resolution if the range block size was large. They compared the suggested algorithm's performance to the entire FIC search.

Valarmathi et al. [19] introduced a technique to categorize domain blocks within three groups and approved the Pearson correlation coefficient with fic in 2015. Furthermore, this method by [8] employs an iteration-free approach. This approach has been used for grayscale pictures.

Jafarzadeh et al. had previously offered a strategy that used local binary features, clever STD thresholding, and Humming distance approaches [20]**.** Compared to the complete search approach, this method minimizes FIC encoding time and decomposes the rebuilt image's PSNR value.

Scientific researchers have examined quadtree coding, one of the most prominent hierarchical segmentation-based coding schemes [21]. It iteratively separates the picture into basic geometric areas. In general, coding systems based on hierarchical segmentation (e.g., quadtree, HV) yield superior compression performance than fixed-block partitioning-based systems (e.g., JPEG, subband coding, classical vector quantization) [22].

The grayscale image was divided by Veenadevi and Ananth [23] into non-overlapping blocks, according to the threshold value and Quadtree decomposition, to obtain fractal image compression. These methods are applied to compress satellite images using threshold values and Huffman coding to encode and decode the image. Pandey and Seth [24] proposed a compression method for the fractal images using quad-tree decomposition. Rationally, in the Quadtree method, the block size changed due to the image features and partitioning into blocks with different lengths [25]. Quadtree (QT) is used to enhance IFS performance. It is used as a variable range block partitioning scheme instead of a fixed one. The criteria guiding the decomposition process is the information richness of the region; it was used to decide the initial partitioning of the range blocks [26].

IV. Fractal image compression using block indexing technique

An improved fractal image compression (FIC) scheme was presented by Sultan et al. [26]. This scheme suggests using zero-mean terms for range blocks by applying moment descriptors, which speed up the block-matching process through symmetry prediction and block indexing. The system uses the quadtree method to partition range blocks as variable partitioning, with the decision based on Sobel-based edge magnitude and contrast of the block. The system also introduces a new series of moment descriptors to emphasize the weights of different parts of each block. Additionally, it investigates the performance of different combinations of double moment descriptors and presents a fast computational technique to calculate the attended moments to enhance the overall computation cost [25]. The following subsections will summarize the basic steps outlined by Sultan et al. and utilized in this research, coupled with modified steps presented in this paper.

1. *Zero-mean constructed blocks by IFS coding*

IFS coding using zero-mean block matching replaces block offsets with average brightness values. Therefore, that adjustment informed the creation of equations for particular IFS

mapping steps. According to Sultan et al. (2018), the contractive affine approximation for a range block with pixel values (r0,…..,rn-1) and a domain block with pixels (d0,…..,dn-1) is [26]:

$$ø = \sum_{l=0}^{n-1} r_l d_l - n\bar{r}\bar{d} \quad (1)$$

$$s = \frac{ø}{n\sigma_d^2} \quad (2)$$

$$\chi 2 = n\sigma_r^2 + s^2 n\sigma_d^2 - 2sø \quad (3)$$

Where r_l is the most appropriate approximated value of the l^{th} byte value of the range block, while in the matched best domain block, d_l is the identical byte value. s represents the degree of scaling. The averages of domain and range blocks are $\bar{d}$ and $\bar{r}$ respectively.

In the calculation of $\chi 2$ (see equation 3), the degree of scaling (s) had to be within the range [-smax, smax] for each range-domain matching case. The degree of scaling (s) and r must then be processed by quantization via computing the next equations [27].

$$\tilde{s} = Q_s I_s \quad (4)$$

$$I_s = round\left(\frac{s}{Q_s}\right) \quad (5)$$

$$\tilde{r} = Q_{\bar{r}} I_r \quad (6)$$

$$I_r = round\left(\frac{\bar{r}}{Q_{\bar{r}}}\right) \quad (7)$$

Where

$$Q_s = \frac{s_{max}}{2^{bs-1} - 1} \quad (8)$$

$$Q_{\bar{r}} = \frac{255}{2^{br} - 1} \quad (9)$$

s_{max} is the highest permissible value of the scale coefficients.

Q_s and $Q_{\bar{r}}$ are the quantization steps of the scale and $\bar{r}$ coefficients respectively.

bs is the number of scale bits, and br is the number of range mean bits.

The quantized value of scale (s) and $\bar{r}$ are used to calculate the χ^2 sum of square error using (equations 4 and 6) [28].

2. *Isometric Process Predictor*

Table (1) explains the eight isometric mappings [14]. A complete search over a set of 8 isometric instances per block is illicit due to the large number of computations involved. The aim for the blocks that were not chosen as the best solution and had no possibility of being chosen should be to chop off their isometric states [28]. Block indexing and transform prediction computations ought to be less complicated than the whole computation. This would lessen the search load by narrowing the pool of viable candidates to a minimum error. As a result, in this procedure, the FIC is expedited by a first-order moments descriptor [14]. The following sections describe the theoretical basis for the predictor of isometric processes.

Table 1: Isometric Transformation [14]

ID	Operation	Equation	Results
Zero	Iden.	x' = x cos(0) - y sin(0) y' = – x sin(0) + y cos(0)	x' =x y' =y
One	Rot. (+90)	x' = x cos(90) - y sin(90) y' = –x sin(90) + y cos(90)	x' =y y' = –x
Two	Rot. (+180)	x' = x cos(180) - y sin(180) y' = –x sin(180) + y cos(180)	x' = –x y' = –y
Three	Rot. (+270)	x' = x cos(270) - y sin(270) y' = –x sin(270) + y cos(270)	x' = –y y' =x
Four	Ref. - X-axis	x' = –x cos(0) - y sin(0) y' = –x sin(0) + y cos(0)	x' = –x y' =y
Five	Ref. - X-axis+Rot. (+90)	x' = –x cos(90) - y sin(90) y' = –x sin(90) + y cos(90)	x' = –y y' = –x
Six	Ref. - X-axis+Rot.(+180)	x' = –x cos(180) - y sin(180) y' = –x sin(180) + y cos(180)	x' =x y' = –y
Seven	Ref. - X-axis+Rot.(+270)	x' = –x cos(270) - y sin(270) y' = –x sin(270) + y cos(270)	x' =y y' =x

The Abbreviations Iden., Rot., Ref. denote to Identity, Rotation, and Reflection operations respectively.

The image block I(x,y) such that {x,y| 0,1,…., K-1}, define its first-order centralized moments as [34]:

$$Mo_x = \sum_{x=0}^{K-1}\sum_{y=0}^{K-1} I(x,y)(x-c) \quad (10)$$

$$Mo_y = \sum_{y=0}^{K-1}\sum_{x=0}^{K-1} I(x,y)(y-c) \quad (11)$$

Where $c = \frac{K-1}{2}$

Merging equations in Table (1) with equations (10) and (11), the relationship between the values of the new moments (Mo'_x, Mo'_y) of the transformed block with the values of its old moments (Mo_x, Mo_y) appears before the transformation can be determined; that relationships is shown in Table (2)[14]

Table 2: Moments Relationship Prior and Posterior to Applying the Isometric Transformation [14]

ID	Operation	Relationship
Zero	Iden.	$Mo'_x=Mo_x$, $Mo'_y=Mo_x$
One	Rot. (+90)	$Mo'_x=Mo_y$,$Mo'_y= -Mo_x$
Two	Rot. (+180)	$Mo'_x=-Mo_x$,$Mo'_y= -Mo_y$
Three	Rot. (+270)	$Mo'_x=-Mo_y$,$Mo'_y= Mo_x$
Four	Ref. - X-axis	$Mo'_x=-Mo_x$,$Mo'_y= Mo_y$
Five	Ref. - X-axis+Rot. (+90°)	$Mo'_x=-Mo_y$,$Mo'_y=- Mo_x$
Six	Ref. - X-axis+Rot.(+180°)	$Mo'_x=Mo_x$,$Mo'_y=-Mo_y$
Seven	Ref. - X-axis+Rot.(+270°)	$Mo'_x=Mo_y$,$Mo'_y= Mo_x$

2.1 Blocks classification

The classifying blocks method is proposed and established based on moment criteria. The classification is based on the state of the first-order moment values (represented by Mo_x and Mo_y). The following three status criteria have been used where:

- Case-1: Is $|Mo_x| \geq |Mo_y|$ or not?
- Case-2: Is $Mo_x \geq 0$ or not?
- Case-3: Is $Mo_y \geq 0$ or not?

The three Boolean criteria usage produces eight block classes, as illustrated in Table (3) [27]. For all cases of range-domain matching that are in Table (4) [28], the predictor determines both the domain and the range blocks situation. Next, the isometric transformation index was extracted by the predictor which is necessary to ensure that the domain and range blocks match as closely as possible [28].

Table 3: The Truth Table for Eight Block Classes [27]

Block Class ID	Boolean Criteria		
	$\|Mo_x\| \geq \|Mo_y\|$	$Mo_x \geq 0$	$Mo_y \geq 0$
Zero	True	True	True
One	True	True	False
Two	True	False	True
Three	True	False	False
Four	False	True	True
Five	False	True	False
Six	False	False	True
Seven	False	False	False

Table 4: The Required Isometric Operation to Convert the Block State [28]

		Domain Blocks ID							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Range Blocks ID	**0**	0	6	4	2	7	3	1	5
	1	6	0	2	4	1	5	7	3
	2	4	2	0	6	3	7	5	1
	3	2	4	6	0	5	1	3	7
	4	7	3	1	5	0	6	4	2
	5	1	5	7	3	6	0	2	4
	6	3	7	5	1	4	2	0	6
	7	5	1	3	7	2	4	6	0

3. *Moment's Ratio and Moment Ratio Index*

Compute Moment's ratio by applying the following equation [27]:

$$Ratio_M = \begin{cases} \left|\frac{Mo_y}{Mo_x}\right| \times Nm & if\ |Mo_x \geq Mo_y| \\ \left|\frac{Mo_x}{Mo_y}\right| \times Nm & if|Mo_y \geq Mo_x| \end{cases} \quad (12)$$

Mo_x is the moment around x-axis coordinates, while Mo_y is the moment around y-axis coordinates, Nm denotes the maximum moment ratio number. Al-Hilo and George (2008) arrived at the following conclusion: It is not necessary for any two blocks to have similar ($Ratio_M$) factors to satisfy the affine transform necessarily; rather, this only applies if almost all of the two blocks (domain and range) satisfy the conductive affine transform [12]. The following equation determines the cumulative moment ratio index value, which is a linear combination of two descriptors (RatioM1 and RatioM2). [27]

$$I_M = \lceil Ratio_{M1} \times (Nm + 1) + Ratio_{M2} \rceil \quad (13)$$

The index (I_M) is a way to categorize domain and range blocks (each category contains blocks with identical indexes). This factor optimizes the range-domain block search process by focusing on domain blocks with equivalent I_M values to those examined with IFS.

4. *The Proposed Fractal Image Compression (FIC) System*

The aims of the proposed FIC are:

• Using a set of moment descriptors presented by Sultan et. al. previous study [26] and combining them with a recent set of moment descriptors. The former ones are (W1,W2 and W3) [26] while the current ones are (W4, W5 and W6) as presented minutely in section(4.1). The latter moment descriptors are characterized by their excelling performance and the moments' new weight of a particular section from the block. The inspection of feasible pairings of double moment descriptors was successfully and accurately done.

• As not fixed (i.e., variable) range blocks partitioning technique, Quadtree was used to promote IFS performance according to what was treated in Sultan et. al. previous study [26]. This research also relies on the Quadtree instead of fixed partitioning, with a difference in the criteria used to calculate each region's information richness. The Prewitt operator is used to determine the partitioning strategy of the range blocks.

• To reduce duplication in calculation, the FIC method is rebuilt to include the moment equations.

More information on the planned upgraded FIC will be discovered in the next paragraph.

4.1 The Moments and The Speeding-up Mechanism

The previously tested set of weights is adopted to develop sets of moments. These weights are the following:

$$W_1(l) = \begin{cases} \frac{2}{K}\left(l - \frac{K}{2}\right) & for\ l = \left[0, \frac{K}{2}\right) \\ -W_1(K-1-l) & for\ l = \left[\frac{K}{2} - 1, K\right) \end{cases} \quad (14.a)$$

$$W_1^{int}(l) = \lceil W_1(l) \times 100 \rceil \quad for\ l = [0, K) \quad (14.b)$$

$$W_2(l) = \begin{cases} \frac{2}{K}\left(l - \frac{1}{2}\right) & for\ l = \left[0, \frac{K}{2}\right) \\ -W_2(K-1-l) & for\ l = \left[\frac{K}{2} - 1, K\right) \end{cases} \quad (15.a)$$

$$W_2^{int}(l) = \lceil W_2(l) \times 100 \rceil \quad for\ l = [0, K) \quad (15.b)$$

$$W_3(l) = \begin{cases} sin\left(\frac{\Pi\, l}{\mathrm{K}-1}\right) & for\ l = \left[0, \frac{K}{2}\right) \\ -W_3(K-1-l) & for\ l = \left[\frac{K}{2}-1, K\right) \end{cases} \quad (16.a)$$

$$W_3^{int}(l) = \lceil W3(l) \times 100 \rceil \quad for\ l = [0, K) \quad (16.b)$$

$$W_4(l) = l - \frac{K}{2} \quad (17.a)$$

$$W_4^{int}(l) = \lceil W_4(l) \times 100 \rceil \quad for\ l = [0, K) \quad (17.b)$$

The following set of weights is introduced newly to produce the new sets of moments, they are as follows:

$$W_5(l) = \begin{cases} sin\left(\frac{2\Pi\, l}{\mathrm{K}-1}\right) & for\ l = \left[0, \frac{K}{2}\right) \\ -W_5(K-1-l) & for\ l = \left[\frac{K}{2}-1, K\right) \end{cases} \quad (18.a)$$

$$W_5^{int}(l) = \lceil W_5(l) \times 100 \rceil \quad for\ l = [0, K) \quad (18.b)$$

$$W_6(l) = \begin{cases} tan\left(\frac{\Pi\, l}{\mathrm{K}-1}\right) & for\ l = \left[0, \frac{K}{2}\right) \\ -W_6(K-1-l) & for\ l = \left[\frac{K}{2}-1, K\right) \end{cases} \quad (19.a)$$

$$W_6^{int}(l) = \lceil W_6(l) \times 100 \rceil \quad for\ l = [0, K) \quad (19.b)$$

Figure. 1 and Fig.2 present the previously suggested [26] and recently suggested weight functions allocated to each row or column throughout the block when its size equals 16.

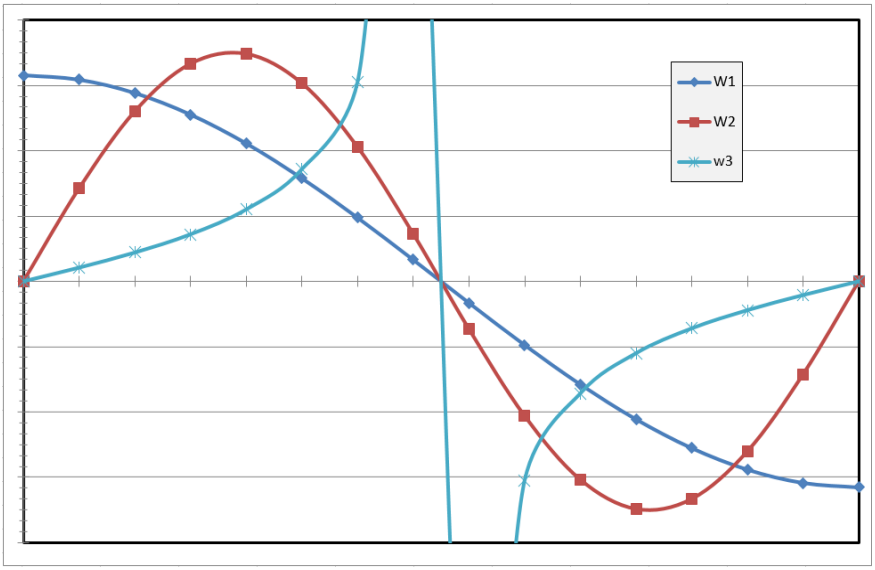


Figure 1: The Previous Weights (W1, W2, W3) [26]

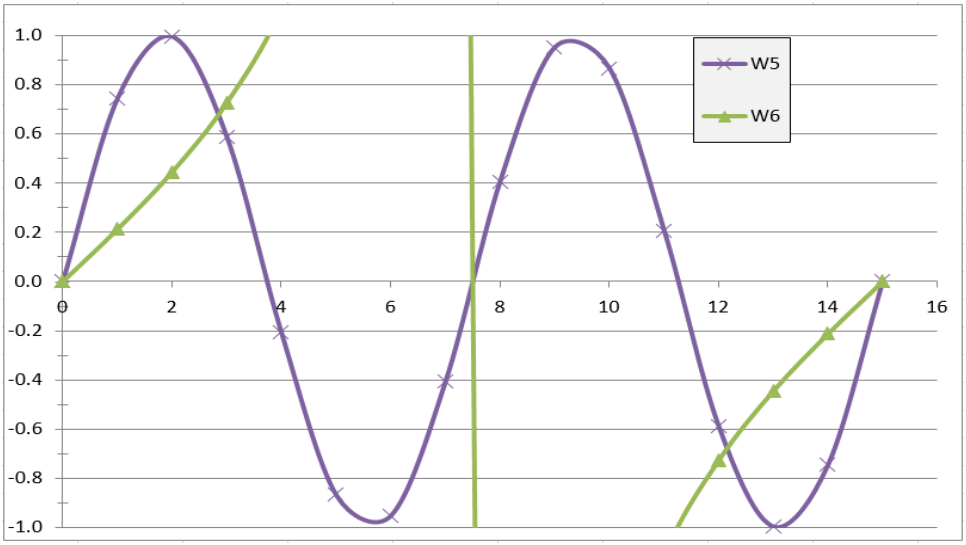


Figure 2 :. The Proposed Weights (W5, W6)

The moment sets around the x-axis and around the y-axis applying the weights functions supplied in the equation stated as:

$$Mox_l(x,y) = \sum_{r=x}^{r+K-1} \sum_{c=y}^{c+K-1} f(r,c) W_l^{int}(r-x) \quad (20)$$

$$Moy_l(x,y) = \sum_{c=y}^{c+K-1} \sum_{r=x}^{r+K-1} f(r,c) W_l^{int}(c-y) \quad (21)$$

Where i represent moments with the corresponding weight, l={1,2,3,4,5,6}. The block's length is denoted by K; its axis coordinates (x,y) relative to the left-top corner of it; the two-dimensional image array is represented by f(); the pre-calculated index of weights is referred to as W^{int}; and the low-order moments adjacent to the x and y axis are denoted by Mo_x and Mo_y, respectively.

Therefore, the moments were calculated for each single block from the overlapping blocks included in the domain pool. To prevent redundant summing within each moment descriptor equation, the following scenario is used for quick calculations: two 2-Dimentional arrays, named Sx and Sy are created so that:

$$Sx(x,0) = \sum_{y=0}^{K-1} f(x,y) \quad (22)$$

$$Sx(x,y) = Sx(x,y-1) - f(x,y-1) + f(x,y+K-1) \quad (23)$$

$$Sy(0,y) = \sum_{x=0}^{K-1} f(x,y) \quad (24)$$

$$Sy(x,y) = Sy(x-1,y) - YB(x-1,y) + YB(x+K-1,y) \quad (25)$$

A simple example of Sx (0,0) and Sx (0,1) and how the value of them had been computed for a row sample shown in Fig.3 [26], when block size (K) equal to 8.

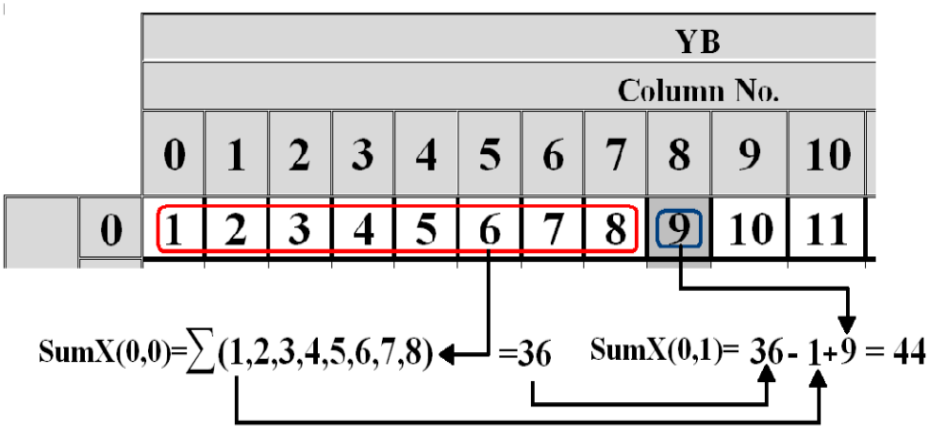


Figure 3: Simple Example About Computing the Value of Sx(0,0) & Sx (0,1) [26].

The constructed arrays Sx and Sy may now be utilized as follows:

$$Mox_l(x,y) = \sum_{r=x}^{r+K-1} Sx(r,y) \times W_l^{int}(r-x) \quad (26)$$

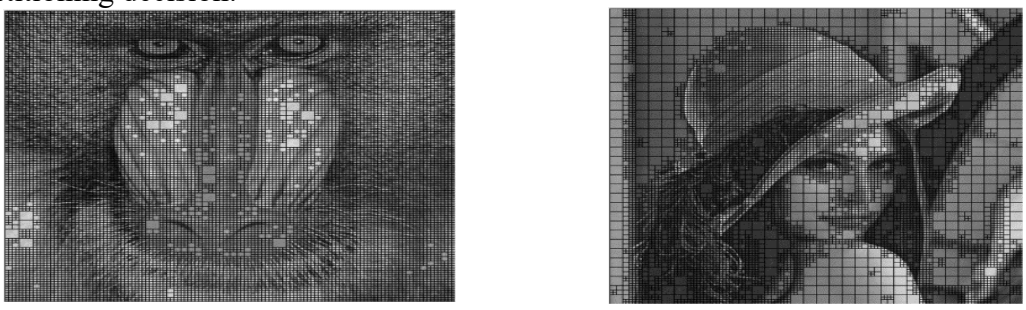
$$Moy_l(x,y) = \sum_{c=y}^{y+K-1} SumY(x,c) \times W_l^{int}(c-y) \quad (27)$$

As previously stated, the efficacy of utilizing the potential pairings of moment combinations extracted by computing W1 to W6 weights to calculate the descriptors that are utilized for indexing the blocks to speed up the range-domain search task, {that is, (Mo1 ,Mo2), (Mo1 ,Mo3), (Mo2 ,Mo3), (Mo1 ,Mo5), (Mo1 ,Mo5), (Mo2 ,Mo5), (Mo2 ,Mo6), (Mo3 ,Mo5)} have been inspected.

4.2 The proposed range pool partitioning scheme

The suggested partitioning scheme for range pool blocks is a quad-tree, which divides the range array into blocks of non-overlapping variable length. The criteria for instructing the decomposition process depend on edge identification by applying a Prewitt filter (see Equation 28a and b). The permissible block length (PBL) is 16, 8, and 4.

The well-known Baboon image is shown in Fig. 4a after executing the presented partitioning scheme, which used Prewitt as the partitioning decision. As well as the well-known Lenna image shown in Fig. 4b after executing the presented partitioning scheme, which used Prewitt as a partitioning decision.



(a) (b)

Figure 4: (a and b) The proposed variable partitioning scheme applied on the Baboon and Lenna test image, Threshold values are val[8]= 36 and val[16]= 48

4.3 Enhanced FIC encoding process

The improved encoding algorithm presented for range blocks is summarized by the steps in the following two figures; the first one (Fig. 5) is about the preprocessing and main processing steps for range and domain preparation, as shown below:

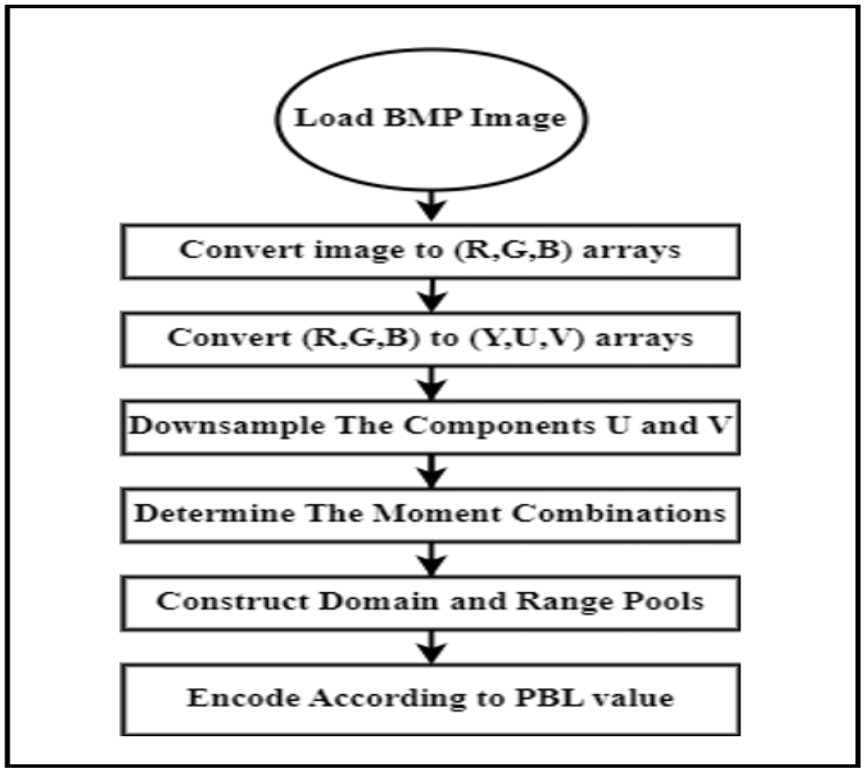


Figure 5: Preprocessing and Preparation Steps of Range and Domain Pools

The second figure (Fig. 6) deals with encoding process details according to PBL value, as shown below:

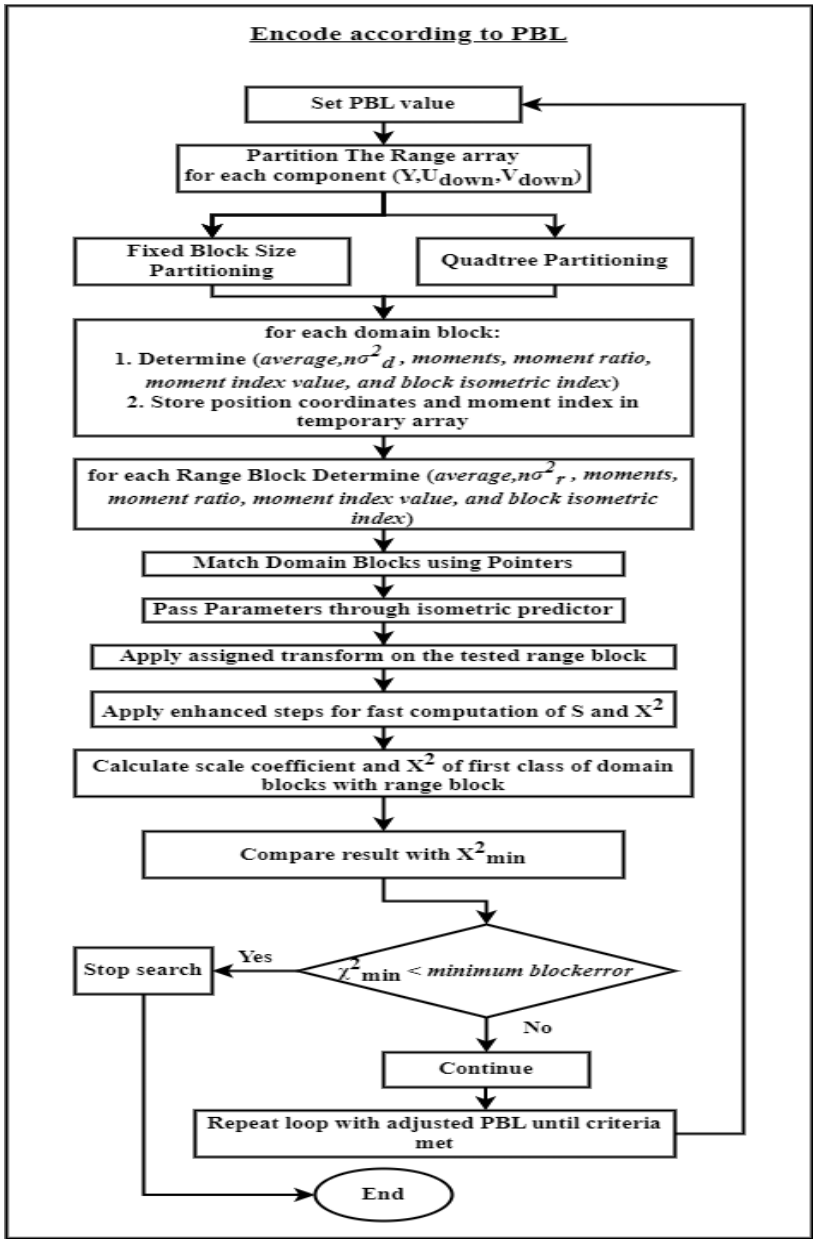


Figure 6: Encoding Process Steps According to PBL Value

V. Test Results

The presented study system was implemented by using Delphi 2010 Programming Language. The experiments were done under the environmental conditions: (Windows-11 pro) operating system, laptop computer – Lenovo (processor 11th Gen Intel(R) Core(TM) i5-1135G7, CPU 2.42 GHz, and 4GB RAM). The experiments were performed on the widely recognized Lena and Baboon image files (in which characteristics: Size = 512 × 512 pixels and Size = 256 × 256 pixels, color depth = 24 bit).

The difference between the reconstructed resulting image and the original one has been determined, and the used error metrics were (mean square error (MSE) and peak signal-to-noise ratio (PSNR)) measured in dB. Besides these fidelity metrics, some complementary metrics were used to describe the system's performance. Both CR and bit rate (BR) parameters have been used to describe the compression gain.

Table 5 lists the examined control parameters (the parameters names and their default values). Choosing these values after extensive testing.

Table 5: The Values of The Control Parameters

Parameter	Range or Value
S_{max}	3
bs	6
br	8
Minimum Block Error	1.5
$Val_{Prewitt}$	30 + Inc_Val
Inc_Val	{3,6,9 ... 27}
Nm	{35,40,45 ... 60}

Table 6 shows the symbols with their descriptions used in the presented tables and presented figures in this section.

Table 6: The Symbols Used in The Test Results

Notation	Description
T0	By using a mixture of Mo1 and Mo2, T0 Represents a moment index ratio derived from it.
T1	By using a mixture of Mo1 and Mo3, T1 Represents a moment index ratio derived from it.
T2	By using a mixture of Mo2 and Mo3, T2 Represents a moment index ratio derived from it.
T3	By using a mixture of Mo1 and Mo5, T3 Represents a moment index ratio derived from it.
T4	By using a mixture of Mo1 and Mo6, T4 Represents a moment index ratio derived from it.
T5	By using a mixture of Mo2 and Mo5, T5 Represents a moment index ratio derived from it.
T6	By using a mixture of Mo2 and Mo6, T6 Represents a moment index ratio derived from it.
T7	By using a mixture of Mo3 and Mo5, T7 Represents a moment index ratio derived from it.
F14	FIC presented for blocks as fixed partitioning with a block length of 4 with image size 256 × 256
F18	FIC presented for blocks as fixed partitioning with a block length of 8 with image size 256 × 256
F24	FIC presented for blocks as fixed partitioning with a block length of 4 with image size 512 × 512
F28	FIC presented for blocks as fixed partitioning with a block length of 8 with image size 512 × 512
Q1P	FIC presented for blocks as quadtree partitioning scheme utilizing Prewitt as partitioning decision with block length 4 and 8 with image size 256 × 256
Q2P1	FIC presented for blocks as quadtree partitioning scheme utilizing Prewitt as partitioning decision with block length 4 and 8 with image size 512 × 512
Q2P2	FIC presented for blocks as quadtree partitioning scheme utilizing Prewitt as partitioning decision with block length 4, 8 and 16 with image size 512 × 512

A. Moment Combination Test

In this set of tests, the presented moments effects by using the different combinations of them are clarified for the following images:

a. Image size 256 × 256 pixels, compute:
i. Block length (4×4). (That is, F14) as listed in Table 3.
ii. Block length (8×8). (That is, F18) as listed in Table 4.
b. Image size 512 × 512 pixels, compute:
i. Block length (4×4). (That is, F24) as listed in Table 5.
ii. Block length (8×8). (That is, F28) as listed in Table 6.

2. Fixed Partitioning: When the blocks have been partitioned fixedly to:

3. Quadtree Partitioning Using Prewitt Filter

This section involves the presented applied FIC, having examined the quadtree partitioning scheme using the Prewitt filter as a partitioning decision.

The value of block length, according to image size, is set equal to:

i. For image 256 × 256, the variable block length is set to (4 and 8). That is Q1P for using the Prewitt filter, as listed in Tables 7 and 8.
ii. For image 512 × 512, the variable block length is set once to (4 and 8) and another to (4,8 and 16). That is:
1. For block length equal to (4 and 8), Q2P1 for using Prewitt filter, as listed in Tables 7 and 8.
2. For block length equal to (4, 8, and 16), Q2P2 for using the Prewitt filter, as listed in Tables 7 and 8.

B. Fixed Partitioning Versus Quadtree Partitioning

The performed tests determine the effect of the pre-proposed quadtree partitioning scheme with moment combination versus fixed block partitioning. The moments' combination that was used in calculating the results of this test, which achieved satisfactory results in Fixed Partitioning, in agreement with the best NM value in this research, which is equal to 50, as indicated by the yellow color in Tables 7 to 10.

Table 7: Test Results of Lenna and Baboon images of size (256x256) and block size4 (i.e. F14)

		Lenna					Baboon				
	Nm	CR	MSE	PSNR	BR	ET(Sec.)	CR	MSE	PSNR	BR	ET(Sec.)
T0	35	8.240	59.737	30.368	2.913	0.155	8.211	496.297	21.173	2.923	0.105
	40	8.238	64.173	30.057	2.913	0.130	8.209	508.188	21.071	2.924	0.087
	45	8.234	65.688	29.956	2.915	0.113	8.213	520.676	20.965	2.922	0.073
	50	8.236	70.505	29.649	2.914	0.102	8.212	538.535	20.819	2.922	0.066
	55	8.234	72.011	29.557	2.915	0.094	8.210	547.965	20.743	2.923	0.061
	60	8.234	76.746	29.280	2.915	0.085	8.213	559.998	20.649	2.922	0.055
T1	35	8.253	65.230	29.986	2.908	0.112	8.204	503.109	21.114	2.925	0.103
	40	8.248	68.763	29.757	2.910	0.097	8.209	518.185	20.986	2.924	0.082
	45	8.247	72.711	29.515	2.910	0.085	8.208	532.819	20.865	2.924	0.069
	50	8.247	76.497	29.294	2.910	0.077	8.209	545.311	20.764	2.923	0.062
	55	8.241	79.343	29.136	2.912	0.070	8.209	558.583	20.660	2.923	0.056
	60	8.242	84.561	28.859	2.912	0.066	8.208	569.208	20.578	2.924	0.053
T2	35	8.258	64.939	30.006	2.906	0.145	8.213	495.330	21.182	2.922	0.118
	40	8.252	67.041	29.867	2.908	0.126	8.214	507.888	21.073	2.922	0.098
	45	8.245	71.396	29.594	2.911	0.107	8.214	520.128	20.970	2.922	0.084
	50	8.244	75.146	29.372	2.911	0.100	8.214	534.851	20.849	2.922	0.073
	55	8.241	78.838	29.164	2.912	0.088	8.215	545.977	20.759	2.922	0.065
	60	8.241	81.560	29.016	2.912	0.081	8.215	557.478	20.669	2.922	0.061
T3	35	8.223	142.473	26.594	2.919	0.106	8.207	520.293	20.968	2.924	0.084
	40	8.258	123.554	27.212	2.906	0.096	8.203	567.835	20.589	2.926	0.059
	45	8.217	138.234	26.725	2.921	0.075	8.209	589.872	20.423	2.924	0.051
	50	8.207	144.087	26.545	2.924	0.065	8.209	602.197	20.333	2.924	0.048
	55	8.209	166.875	25.907	2.924	0.057	8.212	610.802	20.272	2.922	0.042
	60	8.214	173.286	25.743	2.922	0.052	8.223	621.057	20.200	2.919	0.040
T4	35	8.327	44.669	31.631	2.882	0.597	8.207	392.798	22.189	2.924	0.753
	40	8.430	45.417	31.559	2.847	0.432	8.208	399.155	22.119	2.924	0.674
	45	8.429	45.796	31.523	2.847	0.393	8.212	378.881	22.346	2.923	1.147
	50	8.414	47.234	31.388	2.852	0.361	8.204	409.789	22.005	2.925	0.554
	55	8.411	48.882	31.239	2.854	0.333	8.209	384.181	22.285	2.924	0.937
	60	8.401	49.822	31.157	2.857	0.307	8.203	413.669	21.964	2.926	0.467
T5	35	8.238	211.235	24.883	2.913	0.082	8.212	551.547	20.715	2.923	0.081
	40	8.225	124.707	27.172	2.918	0.114	8.217	597.226	20.369	2.921	0.060
	45	8.257	121.463	27.286	2.907	0.103	8.198	632.858	20.118	2.927	0.049
	50	8.243	149.336	26.389	2.911	0.081	8.207	645.178	20.034	2.924	0.045
	55	8.209	163.148	26.005	2.924	0.065	8.203	660.615	19.931	2.926	0.041
	60	8.208	196.298	25.202	2.924	0.058	8.206	677.170	19.824	2.925	0.039
T6	35	8.330	45.715	31.530	2.881	0.606	8.212	402.849	22.079	2.922	0.756
	40	8.466	47.861	31.331	2.835	0.448	8.212	405.470	22.051	2.923	1.003
	45	8.449	48.414	31.281	2.841	0.402	8.214	383.612	22.292	2.922	1.142
	50	8.441	46.737	31.434	2.843	0.368	8.213	414.864	21.952	2.922	0.544
	55	8.413	48.836	31.243	2.853	0.338	8.217	392.863	22.188	2.921	0.932
	60	8.423	48.740	31.252	2.849	0.312	8.215	422.630	21.871	2.921	0.459
T7	35	8.498	48.055	31.313	2.824	0.479	8.212	406.172	22.044	2.922	0.918
	40	8.497	49.099	31.220	2.825	0.427	8.210	415.387	21.946	2.923	0.662
	45	8.536	50.231	31.121	2.812	0.422	8.224	424.373	21.853	2.918	0.568
	50	8.587	50.396	31.107	2.795	0.407	8.217	427.864	21.818	2.921	0.516
	55	8.569	50.520	31.096	2.801	0.344	8.220	433.457	21.761	2.920	0.473
	60	8.494	52.135	30.960	2.825	0.307	8.218	433.923	21.757	2.921	0.433

Table 8: Test Results of Lenna and Baboon images of size (256x256) ang block size 8 (i.e. F18)

		Lenna					Baboon				
	Nm	**CR**	**MSE**	**PSNR**	**BR**	**ET (Sec.)**	**CR**	**MSE**	**PSNR**	**BR**	**ET (Sec.)**
T0	35	32.955	234.049	24.438	0.728	0.137	32.856	786.088	19.176	0.730	0.081
	40	32.911	241.230	24.307	0.729	0.115	32.839	793.239	19.137	0.731	0.069
	45	32.894	243.923	24.258	0.730	0.098	32.856	808.137	19.056	0.730	0.059
	50	32.850	251.467	24.126	0.731	0.086	32.856	815.591	19.016	0.730	0.053
	55	32.850	282.919	23.614	0.731	0.078	32.856	830.825	18.936	0.730	0.049
	60	32.867	271.553	23.792	0.730	0.071	32.856	832.990	18.924	0.730	0.046
T1	35	32.916	260.181	23.978	0.729	0.143	32.861	787.018	19.171	0.730	0.083
	40	32.933	247.333	24.198	0.729	0.119	32.845	795.543	19.124	0.731	0.070
	45	32.867	256.047	24.048	0.730	0.103	32.856	800.119	19.099	0.730	0.061
	50	32.856	257.542	24.022	0.730	0.088	32.856	810.691	19.042	0.730	0.055
	55	32.861	328.915	22.960	0.730	0.080	32.867	823.860	18.972	0.730	0.050
	60	32.834	273.772	23.757	0.731	0.073	32.845	829.317	18.944	0.731	0.046
T2	35	32.883	216.335	24.780	0.730	0.237	32.850	780.453	19.207	0.731	0.113
	40	32.927	230.168	24.510	0.729	0.194	32.795	789.448	19.158	0.732	0.094
	45	32.922	253.037	24.099	0.729	0.170	32.850	796.914	19.117	0.731	0.078
	50	32.856	250.132	24.149	0.730	0.147	32.795	806.467	19.065	0.732	0.069
	55	32.889	246.541	24.212	0.730	0.131	32.812	811.710	19.037	0.731	0.062
	60	32.883	269.957	23.818	0.730	0.117	32.839	812.755	19.031	0.731	0.057
T3	35	32.861	266.430	23.875	0.730	0.064	32.867	815.395	19.017	0.730	0.066
	40	32.955	326.004	22.999	0.728	0.076	32.839	843.229	18.871	0.731	0.053
	45	32.861	297.929	23.390	0.730	0.077	32.916	864.702	18.762	0.729	0.042
	50	32.966	276.656	23.711	0.728	0.086	32.867	884.583	18.663	0.730	0.038
	55	32.960	309.039	23.231	0.728	0.073	32.872	894.960	18.613	0.730	0.036
	60	32.894	305.778	23.277	0.730	0.064	32.872	910.607	18.538	0.730	0.034
T4	35	32.889	259.774	23.985	0.730	0.098	32.823	807.113	19.062	0.731	0.068
	40	32.938	262.542	23.939	0.729	0.062	32.872	819.051	18.998	0.730	0.059
	45	32.905	278.747	23.679	0.729	0.054	32.834	830.471	18.938	0.731	0.051
	50	32.894	295.268	23.429	0.730	0.047	32.812	844.090	18.867	0.731	0.049
	55	32.845	292.717	23.466	0.731	0.044	32.834	849.455	18.839	0.731	0.044
	60	32.867	303.615	23.308	0.730	0.041	32.817	866.103	18.755	0.731	0.039
T5	35	32.872	287.519	23.544	0.730	0.088	32.878	858.344	18.794	0.730	0.062
	40	32.949	289.496	23.514	0.728	0.098	32.806	876.161	18.705	0.732	0.045
	45	32.817	318.549	23.099	0.731	0.093	32.839	893.735	18.619	0.731	0.040
	50	32.839	352.078	22.664	0.731	0.079	32.889	902.876	18.575	0.730	0.037
	55	32.828	342.142	22.789	0.731	0.062	32.856	909.690	18.542	0.730	0.035
	60	32.845	358.064	22.591	0.731	0.059	32.916	926.250	18.464	0.729	0.034
T6	35	32.867	256.301	24.043	0.730	0.081	32.834	815.457	19.017	0.731	0.071
	40	32.872	274.507	23.745	0.730	0.061	32.861	827.660	18.952	0.730	0.059
	45	32.922	299.965	23.360	0.729	0.054	32.828	843.677	18.869	0.731	0.051
	50	32.872	313.213	23.172	0.730	0.048	32.817	855.880	18.807	0.731	0.046
	55	32.845	331.019	22.932	0.731	0.044	32.856	861.263	18.779	0.730	0.043
	60	32.834	344.712	22.756	0.731	0.042	32.812	874.808	18.712	0.731	0.040
T7	35	35.643	353.590	22.646	0.673	0.059	35.630	887.001	18.652	0.674	0.044
	40	34.541	326.902	22.987	0.695	0.119	35.502	911.054	18.535	0.676	0.048
	45	34.372	336.137	22.866	0.698	0.041	35.298	931.147	18.441	0.680	0.053
	50	34.342	361.015	22.556	0.699	0.038	35.457	942.712	18.387	0.677	0.042
	55	34.336	415.849	21.941	0.699	0.036	35.604	948.676	18.360	0.674	0.037
	60	34.312	431.773	21.778	0.699	0.035	35.553	949.087	18.358	0.675	0.042

Table 9: Test Results of Lenna and Baboon images of size (512x512) ang block size 4 (i.e. F24)

		Lenna					Baboon				
	Nm	**CR**	**MSE**	**PSNR**	**BR**	**ET (Sec.)**	**CR**	**MSE**	**PSNR**	**BR**	**ET (Sec.)**
T0	35	8.092	24.003	34.328	2.966	1.129	7.730	154.197	26.250	3.105	1.369
	40	8.086	25.181	34.120	2.968	0.905	7.728	160.906	26.065	3.106	1.068
	45	8.080	26.397	33.915	2.970	0.753	7.727	169.271	25.845	3.106	0.874
	50	8.076	27.227	33.781	2.972	0.653	7.727	175.620	25.685	3.106	0.730
	55	8.072	28.062	33.650	2.973	0.579	7.727	182.670	25.514	3.106	0.627
	60	8.076	28.889	33.524	2.972	0.500	7.724	187.649	25.397	3.107	0.545
T1	35	8.125	25.119	34.131	2.954	0.968	7.733	158.478	26.131	3.104	1.209
	40	8.121	26.218	33.945	2.955	0.781	7.733	165.924	25.932	3.104	0.963
	45	8.120	27.679	33.709	2.956	0.660	7.732	174.025	25.725	3.104	0.775
	50	8.110	28.512	33.581	2.959	0.571	7.731	182.239	25.524	3.104	0.657
	55	8.104	29.507	33.432	2.961	0.562	7.729	188.867	25.369	3.105	0.576
	60	8.106	30.699	33.260	2.961	0.466	7.729	194.946	25.232	3.105	0.495
T2	35	8.133	24.251	34.284	2.951	1.363	7.741	151.289	26.333	3.101	1.888
	40	8.131	25.108	34.133	2.952	1.033	7.736	159.918	26.092	3.102	1.322
	45	8.126	26.296	33.932	2.954	0.873	7.738	167.829	25.882	3.101	1.077
	50	8.118	27.227	33.781	2.956	0.735	7.736	173.402	25.740	3.103	0.897
	55	8.113	28.175	33.632	2.958	0.717	7.736	180.740	25.560	3.103	0.779
	60	8.112	29.202	33.477	2.959	0.579	7.734	186.979	25.413	3.103	0.665
T3	35	7.994	32.941	32.954	3.002	0.747	7.728	170.584	25.811	3.105	1.232
	40	8.073	32.643	32.993	2.973	0.541	7.722	204.644	25.021	3.108	0.627
	45	7.989	33.472	32.884	3.004	0.439	7.719	224.828	24.612	3.109	0.420
	50	7.993	34.215	32.789	3.003	0.410	7.733	238.701	24.352	3.104	0.353
	55	7.994	36.301	32.532	3.002	0.405	7.728	241.113	24.309	3.106	0.342
	60	7.994	37.555	32.384	3.002	0.306	7.739	257.905	24.016	3.101	0.264
T4	35	8.343	18.318	35.502	2.877	8.518	7.828	108.704	27.768	3.066	11.097
	40	8.535	19.526	35.225	2.812	5.900	7.940	115.061	27.522	3.023	9.108
	45	8.494	19.988	35.123	2.826	5.307	7.936	117.430	27.433	3.024	8.086
	50	8.475	20.120	35.094	2.832	4.837	7.926	118.647	27.388	3.028	7.532
	55	8.460	20.319	35.052	2.837	4.464	7.915	120.780	27.311	3.032	6.776
	60	8.449	20.388	35.037	2.841	4.150	7.909	121.746	27.276	3.034	6.174
T5	35	7.994	40.779	32.026	3.002	0.699	7.722	205.808	24.996	3.108	1.067
	40	7.986	45.057	31.593	3.005	0.484	7.726	217.325	24.760	3.106	0.878
	45	7.990	43.262	31.770	3.004	0.430	7.723	260.638	23.970	3.108	0.444
	50	7.981	43.541	31.742	3.007	0.384	7.722	278.896	23.676	3.108	0.352
	55	7.984	45.397	31.561	3.006	0.367	7.714	298.800	23.377	3.111	0.278
	60	7.987	48.034	31.315	3.005	0.283	7.721	310.803	23.206	3.108	0.227
T6	35	8.337	18.866	35.374	2.879	8.626	7.831	110.268	27.706	3.065	11.306
	40	8.524	20.094	35.100	2.815	6.074	7.953	118.718	27.386	3.018	9.329
	45	8.483	20.380	35.039	2.829	5.359	7.946	120.833	27.309	3.020	8.243
	50	8.474	20.624	34.987	2.832	4.898	7.939	121.713	27.278	3.023	7.533
	55	8.469	20.850	34.940	2.834	4.525	7.926	122.885	27.236	3.028	6.891
	60	8.453	21.134	34.881	2.839	4.181	7.920	124.623	27.175	3.030	6.370
T7	35	8.525	20.374	35.040	2.815	7.165	7.929	116.183	27.479	3.027	11.056
	40	8.461	21.228	34.862	2.837	5.385	7.922	117.605	27.427	3.030	9.580
	45	8.326	21.473	34.812	2.882	4.776	7.921	120.644	27.316	3.030	9.439
	50	8.329	21.582	34.790	2.881	4.272	7.892	125.987	27.128	3.041	7.471
	55	8.256	21.917	34.723	2.907	3.849	7.903	128.075	27.056	3.037	6.783
	60	8.230	22.061	34.695	2.916	3.597	7.850	130.108	26.988	3.057	6.187

Table 10: Test Results of Lenna and Baboon images of size (512x512) ang block size 8 (i.e. F28)

		Lenna					Baboon				
	Nm	**CR**	**MSE**	**PSNR**	**BR**	**ET (Sec.)**	**CR**	**MSE**	**PSNR**	**BR**	**ET (Sec.)**
T0	35	30.988	72.530	29.526	0.775	1.170	30.799	444.122	21.656	0.779	0.954
	40	30.952	75.299	29.363	0.775	0.937	30.795	451.290	21.586	0.779	0.769
	45	31.044	78.239	29.197	0.773	0.773	30.793	458.049	21.522	0.779	0.621
	50	32.321	80.567	29.069	0.743	0.655	30.799	467.918	21.429	0.779	0.520
	55	31.051	81.986	28.993	0.773	0.568	30.797	473.871	21.374	0.779	0.452
	60	31.036	84.731	28.850	0.773	0.507	30.808	480.709	21.312	0.779	0.398
T1	35	31.032	71.469	29.590	0.773	1.192	30.797	443.097	21.666	0.779	0.998
	40	31.116	74.280	29.422	0.771	0.955	30.792	454.098	21.559	0.779	0.784
	45	31.093	77.019	29.265	0.772	0.801	30.797	463.822	21.467	0.779	0.648
	50	31.017	78.899	29.160	0.774	0.689	30.790	470.649	21.404	0.779	0.547
	55	31.024	81.086	29.041	0.774	0.584	30.798	476.209	21.353	0.779	0.467
	60	31.029	83.296	28.925	0.773	0.511	30.801	485.698	21.267	0.779	0.411
T2	35	31.323	70.243	29.665	0.766	2.233	30.807	443.024	21.667	0.779	1.626
	40	31.105	72.428	29.532	0.772	1.834	30.799	450.972	21.589	0.779	1.282
	45	31.090	74.434	29.413	0.772	1.501	30.797	460.618	21.497	0.779	1.039
	50	31.071	76.115	29.316	0.772	1.270	30.801	467.840	21.430	0.779	0.884
	55	31.058	78.397	29.188	0.773	1.111	30.793	476.012	21.355	0.779	0.758
	60	31.036	80.648	29.065	0.773	0.964	30.797	480.972	21.310	0.779	0.643
T3	35	30.951	93.605	28.418	0.775	0.498	30.803	456.429	21.537	0.779	0.869
	40	31.002	96.077	28.305	0.774	0.382	30.789	478.334	21.334	0.780	0.562
	45	31.089	99.589	28.149	0.772	0.345	30.805	498.004	21.159	0.779	0.370
	50	31.054	99.142	28.168	0.773	0.308	30.792	514.770	21.015	0.779	0.290
	55	31.109	101.358	28.072	0.771	0.274	30.778	525.652	20.924	0.780	0.241
	60	31.100	103.073	27.999	0.772	0.246	30.796	540.839	20.800	0.779	0.213
T4	35	31.021	79.442	29.130	0.774	0.673	30.795	451.597	21.583	0.779	0.889
	40	32.227	79.401	29.133	0.745	0.546	30.799	478.105	21.336	0.779	0.451
	45	30.977	83.084	28.936	0.775	0.450	30.795	487.902	21.248	0.779	0.368
	50	30.933	85.798	28.796	0.776	0.383	30.792	496.280	21.174	0.779	0.319
	55	30.964	88.431	28.665	0.775	0.334	30.797	506.391	21.086	0.779	0.286
	60	30.958	90.780	28.551	0.775	0.297	30.793	518.459	20.984	0.779	0.248
T5	35	31.027	103.660	27.975	0.774	0.429	30.803	485.674	21.267	0.779	0.640
	40	30.980	102.090	28.041	0.775	0.414	30.783	511.051	21.046	0.780	0.403
	45	30.991	102.361	28.029	0.774	0.595	30.797	525.758	20.923	0.779	0.314
	50	30.978	105.577	27.895	0.775	0.313	30.795	530.614	20.883	0.779	0.268
	55	30.959	107.265	27.826	0.775	0.281	30.809	551.324	20.717	0.779	0.231
	60	30.957	112.079	27.636	0.775	0.248	30.803	556.234	20.678	0.779	0.212
T6	35	31.028	77.780	29.222	0.773	0.722	30.783	471.872	21.393	0.780	0.618
	40	30.962	81.428	29.023	0.775	0.556	30.786	484.965	21.274	0.780	0.441
	45	30.941	85.075	28.833	0.776	0.464	30.796	496.409	21.172	0.779	0.367
	50	30.922	87.387	28.716	0.776	0.390	30.796	504.331	21.104	0.779	0.320
	55	30.928	89.690	28.603	0.776	0.338	30.786	512.392	21.035	0.780	0.279
	60	30.939	92.237	28.482	0.776	0.301	30.790	521.722	20.956	0.779	0.244
T7	35	32.377	117.410	27.434	0.741	0.235	33.420	520.366	20.968	0.718	0.307
	40	32.351	122.135	27.262	0.742	0.202	33.328	530.916	20.881	0.720	0.318
	45	32.347	125.120	27.158	0.742	0.180	33.418	551.162	20.718	0.718	0.208
	50	32.305	129.864	26.996	0.743	0.167	33.352	559.353	20.654	0.720	0.249
	55	32.336	132.716	26.902	0.742	0.152	33.153	565.593	20.606	0.724	0.327
	60	32.320	137.887	26.736	0.743	0.141	33.420	581.750	20.483	0.718	0.169

It is worth noting that the three descriptors T0, T1, and T2 were calculated in a previous study by Sultan, George and Hassan [26]. By using these descriptors in the current study, the comparison with the study above, which employed the Sobel operator, becomes clear.
A comparison was made between these descriptors to choose from the charts shown in Fig. 7 (a to f), in which it is clear that the four ones (T2, T4, T6, and T7) are the highest in Cr and PSNR. The examination is done based on Inc_Val and the extent of its effect with different partitioning schemes.

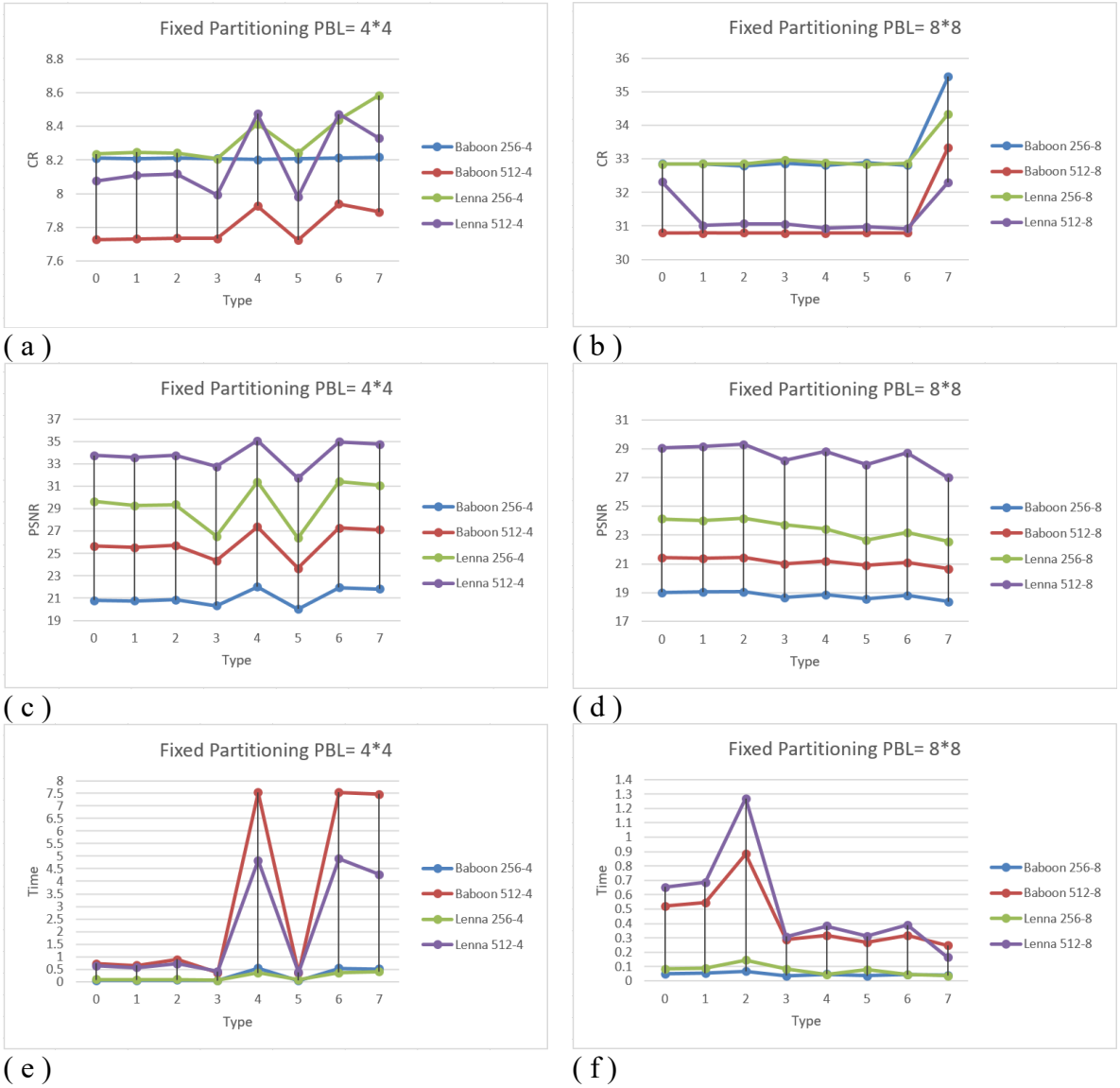


Figure 7: (a-f) The effect of proposed fixed partitioning scheme with all descriptor types

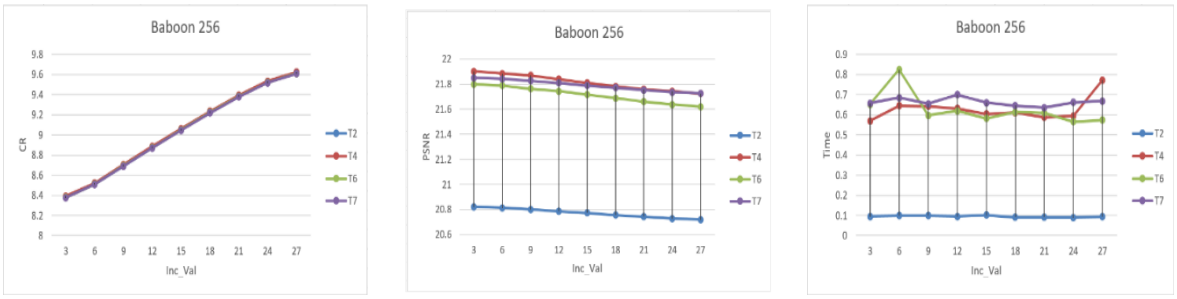
The moments combinations T2, T4, T6, and T7 were selected to implement the performance tests of quadtree partitioning.
The effect of $Val_{prewitt}$ on CR, PSNR, and ET was determined when Q1P, Q2P1, and Q2P2 had been applied to Lenna and Baboon images, as shown in Tables 7 and 8. The tables have been shortened by removing the columns MSE and BR because the PSNR is indicated for MSE, and BR can be easily calculated by knowing Cr.
Fig. 8 to Fig. 13 show the performance of the four types, along with the effect of changing the Inc_Val on the compression ratio, the PSNR, and the time spent executing the system.

Table 11: Test Results of Quadtree Partitioning Scheme When Applied on Lenna Image with Image size (256x256) and (512x512), Nm=50 and Different PBL as Determined

		Image size(256x256)			Image size(512x512)					
	Inc_Val (30 +)	Q1P			Q2P1			Q2P2		
		CR	PSNR	ET (Sec.)	CR	PSNR	ET (Sec.)	CR	PSNR	ET (Sec.)
T2	3	9.961	29.199	0.143	11.794	33.245	1.137	14.420	32.249	1.109
	6	10.173	29.165	0.130	12.413	33.163	1.227	15.441	32.015	1.102
	9	10.424	29.109	0.156	13.056	33.079	1.209	16.315	31.831	1.130
	12	10.657	29.055	0.139	13.651	32.983	1.235	17.360	31.665	1.152
	15	10.853	29.034	0.143	14.221	32.897	1.273	18.220	31.483	1.152
	18	11.023	28.988	0.144	14.648	32.822	1.260	18.968	31.348	1.171
	21	11.186	28.934	0.139	15.124	32.757	1.324	19.849	31.238	1.180
	24	11.506	28.835	0.146	15.587	32.668	1.291	20.956	31.083	1.178
	27	11.708	28.768	0.143	15.995	32.605	1.290	21.854	30.958	1.211
T4	3	10.075	30.901	0.396	11.917	34.226	4.928	14.550	32.877	4.464
	6	10.272	30.837	0.429	12.512	34.198	4.825	15.560	32.557	4.339
	9	10.519	30.742	0.406	13.148	34.045	4.613	16.415	32.352	4.217
	12	10.758	30.644	0.427	13.716	33.876	4.476	17.451	32.067	4.042
	15	10.954	30.568	0.416	14.292	33.737	4.290	18.295	31.831	3.909
	18	11.125	30.477	0.411	14.695	33.622	4.099	19.032	31.620	3.814
	21	11.288	30.446	0.381	15.149	33.569	3.914	19.909	31.493	3.701
	24	11.603	30.198	0.393	15.621	33.470	3.750	21.011	31.254	3.595
	27	11.805	29.917	0.360	16.004	33.339	3.582	21.901	30.892	3.492
T6	3	10.078	31.068	0.448	11.897	34.260	4.910	14.546	32.793	4.636
	6	10.287	31.005	0.432	12.509	34.105	4.706	15.554	32.469	4.493
	9	10.532	30.896	0.422	13.132	33.975	4.512	16.410	32.216	4.385
	12	10.767	30.797	0.413	13.711	33.832	4.281	17.444	32.008	4.185
	15	10.965	30.728	0.411	14.271	33.682	4.169	18.294	31.799	4.038
	18	11.137	30.595	0.416	14.691	33.587	4.042	19.030	31.647	3.953
	21	11.299	30.539	0.404	15.145	33.457	3.899	19.906	31.463	3.764
	24	11.618	30.349	0.416	15.603	33.324	3.781	21.006	31.187	3.632
	27	11.817	30.095	0.393	15.995	33.227	3.621	21.890	31.008	3.601
T7	3	10.116	30.923	0.490	12.069	34.152	5.931	14.614	33.391	6.710
	6	10.325	30.879	0.453	12.703	34.076	5.953	15.621	33.151	6.719
	9	10.574	30.812	0.481	13.353	33.983	6.042	16.477	32.966	6.733
	12	10.807	30.737	0.483	13.959	33.880	6.027	17.511	32.812	6.695
	15	11.005	30.695	0.439	14.539	33.790	6.080	18.358	32.630	6.759
	18	11.176	30.661	0.462	14.975	33.717	6.083	19.091	32.500	6.735
	21	11.339	30.600	0.458	15.463	33.648	6.081	19.971	32.389	6.603
	24	11.657	30.491	0.451	15.936	33.577	6.114	21.072	32.207	6.608
	27	11.861	30.410	0.471	16.352	33.513	6.154	21.963	32.053	6.610

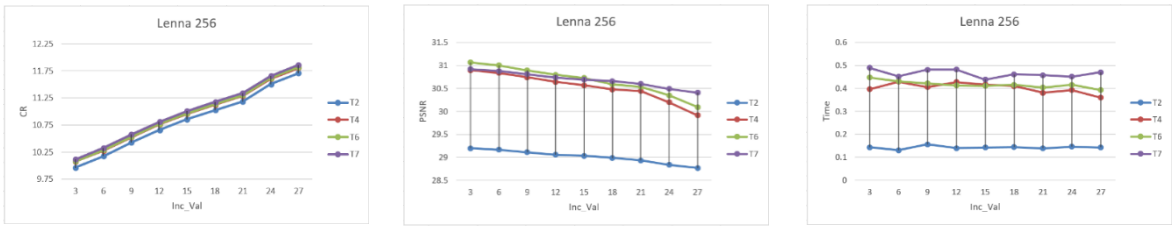
Table 12: Test Results of Quadtree Partitioning Scheme When Applied on Baboon Image with Image size (256x256) and (512x512), Nm=50 and Different PBL as Determined

		Image size(256x256)			**Image size(512x512)**					
		Q1P			**Q2P1**			**Q2P2**		
	Inc_Val (30 +)	**CR**	**PSNR**	**ET (Sec.)**	**CR**	**PSNR**	**ET (Sec.)**	**CR**	**PSNR**	**ET (Sec.)**
T2	3	8.381	20.823	0.095	8.834	25.645	1.177	9.348	25.421	1.250
	6	8.512	20.813	0.100	9.035	25.621	1.212	9.583	25.361	1.220
	9	8.693	20.801	0.100	9.214	25.591	1.216	9.823	25.283	1.198
	12	8.873	20.786	0.096	9.421	25.552	1.209	10.080	25.208	1.262
	15	9.052	20.774	0.102	9.610	25.516	1.193	10.316	25.138	1.241
	18	9.224	20.757	0.091	9.786	25.484	1.211	10.542	25.070	1.215
	21	9.383	20.744	0.092	9.943	25.456	1.207	10.753	24.978	1.232
	24	9.524	20.729	0.089	10.081	25.432	1.218	10.944	24.927	1.300
	27	9.613	20.719	0.094	10.245	25.396	1.206	11.335	24.842	1.241
T4	3	8.393	21.902	0.570	8.850	27.309	8.637	9.370	26.949	8.857
	6	8.522	21.884	0.646	9.051	27.259	8.604	9.614	26.739	8.020
	9	8.706	21.870	0.643	9.227	27.220	8.542	9.851	26.608	7.992
	12	8.887	21.839	0.632	9.434	27.159	8.431	10.107	26.492	7.794
	15	9.063	21.810	0.605	9.621	27.104	8.374	10.342	26.357	7.723
	18	9.236	21.783	0.611	9.796	27.056	8.247	10.567	26.244	7.584
	21	9.394	21.758	0.588	9.951	27.001	8.088	10.777	26.167	7.513
	24	9.535	21.744	0.595	10.090	26.952	8.001	10.968	26.055	7.383
	27	9.626	21.724	0.773	10.253	26.883	7.961	11.196	25.998	7.365
T6	3	8.380	21.799	0.652	8.851	27.216	8.845	9.382	26.737	8.281
	6	8.511	21.788	0.825	9.049	27.170	8.759	9.614	26.642	8.156
	9	8.694	21.764	0.597	9.224	27.125	8.654	9.851	26.531	8.122
	12	8.875	21.743	0.619	9.430	27.066	8.476	10.107	26.409	7.961
	15	9.052	21.715	0.582	9.618	27.000	8.454	10.342	26.295	7.918
	18	9.226	21.688	0.614	9.792	26.944	8.235	10.566	26.193	7.802
	21	9.383	21.661	0.609	9.947	26.894	8.283	10.777	26.061	7.586
	24	9.522	21.638	0.566	10.085	26.847	8.093	10.966	25.995	7.517
	27	9.612	21.620	0.573	10.249	26.781	7.762	11.194	25.902	7.405
T7	3	8.376	21.849	0.657	8.896	27.029	8.369	9.398	26.795	8.625
	6	8.508	21.842	0.685	9.095	27.001	8.297	9.631	26.734	8.651
	9	8.690	21.828	0.655	9.271	26.974	8.362	9.868	26.661	8.595
	12	8.870	21.810	0.701	9.476	26.938	8.319	10.123	26.581	8.599
	15	9.047	21.790	0.661	9.664	26.902	8.419	10.357	26.508	8.963
	18	9.220	21.771	0.645	9.838	26.863	11.678	10.582	26.442	8.821
	21	9.380	21.752	0.635	9.993	26.830	12.174	10.791	26.340	8.916
	24	9.519	21.735	0.663	10.131	26.798	12.247	10.980	26.279	8.764
	27	9.609	21.727	0.669	10.295	26.757	12.211	11.211	26.208	8.775



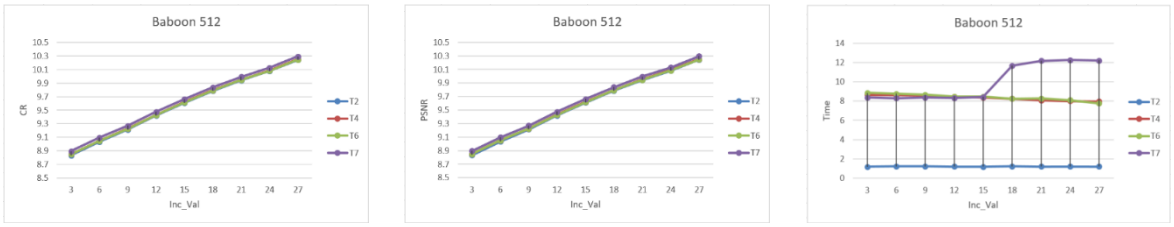
(a) (b) (c)

Figure 8:(a-c) The effect of quadtree partitioning scheme Q1P applied on Baboon image



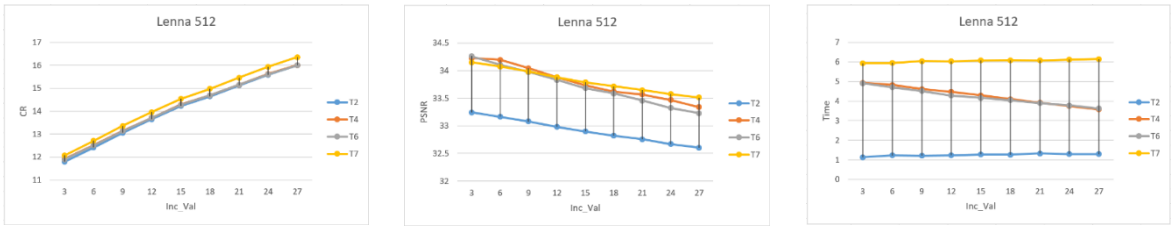
(a) (b) (c)

Figure 9: (a-c) The effect of quadtree partitioning scheme Q1P applied on Lenna image



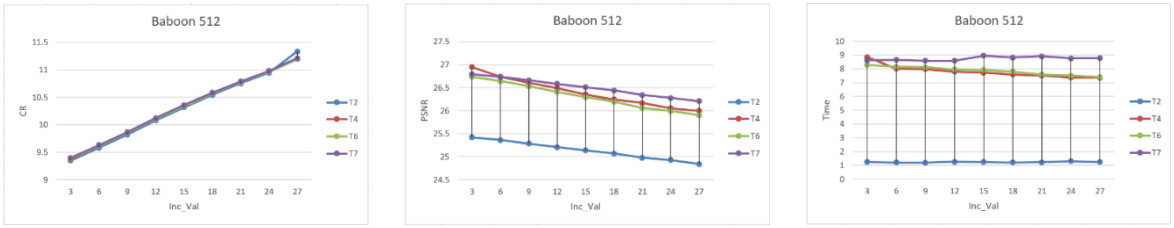
(a) (b) (c)

Figure 10: (a-c) The effect of quadtree partitioning scheme Q2P1 applied on Baboon image



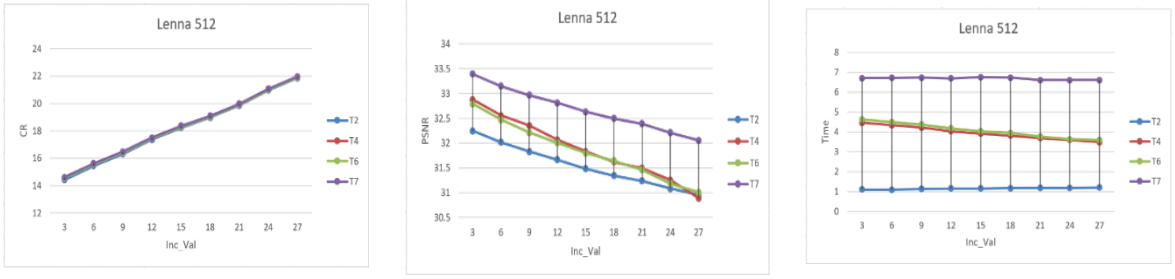
(a) (b) (c)

Figure 11: (a-c) The effect of quadtree partitioning scheme Q2P1 applied on Lenna image



(a) (b) (c)

Figure 12: (a-c) The effect of quadtree partitioning scheme Q2P2 applied on Baboon image



(a) (b) (c)

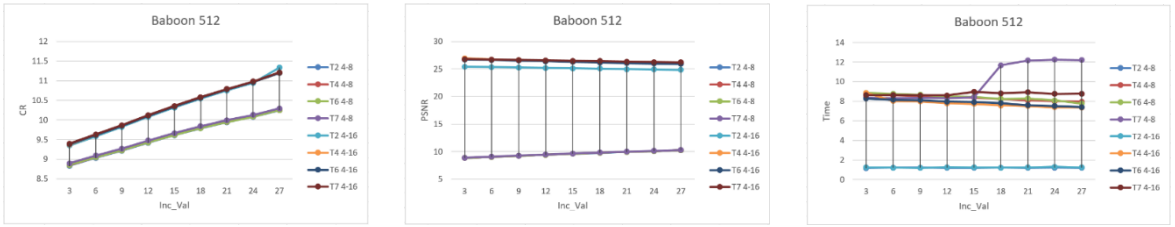
Figure 13: (a-c) The effect of quadtree partitioning scheme Q2P2 applied on Lenna image

C. Quadtree Partitioning with Different Image Size

Implementing the proposing system within the quadtree partitioning mechanism on an image sample with a size of 512 × 512 pixel has many features, including the greater facility due to the wider capacity it has in distributing pixels, and thus the algorithm performance appears more accurately.

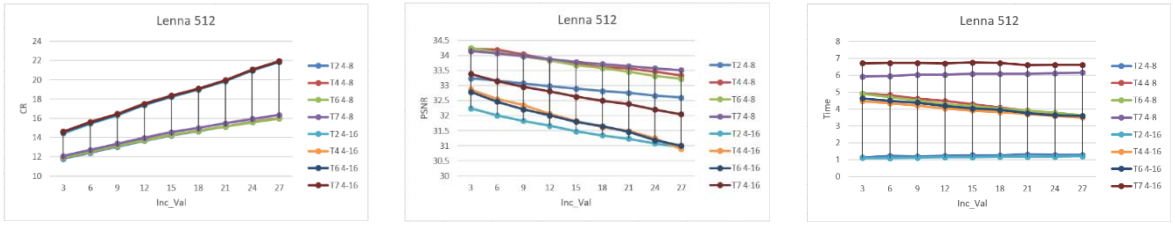
In order to further verify the effectiveness of the quadtree partitioning mechanism in this system, a comparison was made as to whether the image was divided into blocks of variable lengths (the smallest being 4 and the largest being 8) or (the smallest being 4 and the largest being 16). Here the measurement becomes clear about how effective are types that are more stable in accordance with the change in Inc_Val

According to what was listed in Tables (11 and 12), the comparison in the compression process standards between the various partitioning of the same image appears clearly in the figures Fig.14 and Fig.15.



(a) (b) (c)

Figure 14: (a-c) The effect of different PBL in quadtree partitioning scheme applied on Baboon image with size 512 × 512 pixel



(a) (b) (c)

Figure 15: (a-c) The effect of different PBL in quadtree partitioning scheme applied on Lenna image with size 512 × 512 pixel

VI.Conclusions and Future Work

As a result of the practical tests carried out in the above submitted procedure, the subsequent observations emerged:

1. Prewitt operator, which guides quadtree partitioning, can be considered as a beneficial partitioning mechanism. This is evident in the fact that the compression ratio has improved by 15% to 20% over the fixed partitioning scheme. Also, the improvement in PSNR reaches almost 4 dB.
2. The quadtree partitioning applied on images with size 512 × 512 pixels, when dividing the blocks starting from the largest length of 16 PBL, then 8, down to the smallest length of 4 PBL, achieved better results than partitioning starting with a length of 8, then 4 PBL. This resulted from taking advantage of the high compression ratio available when compressing 16 PBL blocks length, as well as taking advantage of the high PSNR in preserving image data when compressing 4 PBL blocks length.
3. The double moment strategy used in the proposed FIC variable partitioning schemes led to better results than fixed partitioning results by measuring the values of CR and PSNR. When using Double moment, the study reached a compression ratio of (21.963) for the Lenna image and (11.335) for the Baboon image, and the highest PSNR is (34.260) for the Lenna image and (27.216) for the Baboon.
4. The introduced moments are appropriate for performing dual descriptions of the blocks that functionally have the potential to accelerate significantly certain IFS encoding process's steps. This is because when searching for a block, two of its attributes are known, which makes the process of finding that specific block faster. Also, the block of 16 PBL has two blocks of 8 PBL and four blocks of 4 PBL. Therefore, the processing time will be reduced by half in some areas and by a quarter in other areas of the image.
5. Ideas about upcoming projects:
a. Another new criterion of quadtree partitioning could be developed and presented. Some suggestions include second order derivatives filters such as Robert Operator, Robinson compass mask, and Frei-Chen edge detector.
b. Examining a combination of different moment configurations such as triple moment sets. Where a triad consisting of M2, M3, and M5 can be chosen, given that these three are the common elements among the four descriptor types chosen in this research, which increases the likelihood of better results appearing when they are combined together.

References

[1] V. Y. Varma, T. N. Prasad, and N. V. P. S. Kumar, "Image compression methods based on transform coding and fractal coding," *International Journal of Engineering Sciences & Research Technology*, vol. 6, no. 10, pp. 481-487, 2017.

[2] G. K. AL-Khafaji, M. H. Rasheed, M. Siddeq, and M. Rodrigues, "Adaptive polynomial coding of multi-base hybrid compression," *International Journal of Engineering, Transactions B : Applications*, vol. 36, no. 2, pp. 236-252, 2023.

[3] V. J. Rehna, "Hybrid approaches to image coding: A review," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 7, 2011.

[4] M. Santhi and R. S. D. Wahida Banu, "Modified SPIHT algorithm for coding color image using inter-color correlation," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 10, no. 3, pp. 256, Mar. 2010.

[5] L. E. George and B. Sultan, "Image compression based on wavelet, polynomial and quadtree," *Journal of Applied Computer Science & Mathematics*, vol. 11, no. 5, pp. 15-20, 2011.

[6] A. Purushothaman and K. Sheeba, "A survey on fractal image compression techniques," *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, vol. 3, no. 1, 2016.

[7] Q. Wang, D. Liang, and S. Bi, "Fast fractal image encoding based on correlation information feature," *2010 3rd International Congress on Image and Signal Processing (CISP)*, pp. 540-543, 2010.

[8] H. T. Chang and C. J. Kuo, "Iteration-free fractal image coding based on efficient domain pool design," *IEEE Transactions on Image Processing*, vol. 9, pp. 329-339, 2000.

[9] S. D. Kamble, N. V. Thakur, L. G. Malik, and P. R. Bajaj, "Color video compression based on fractal coding using quadtree weighted finite automata," *Information Systems Design and Intelligent Applications: Proceedings of Second International Conference INDIA 2015*, pp. 649-658, Springer, 2015.

[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," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 3, pp. 1564-1570, 2019.

[11] L. E. George, "IFS coding for zero-mean image blocks," *Iraqi Journal of Science*, vol. 47, pp. 190-194, 2005.

[12] E. Al-Hilo and L. E. George, "Speeding-up fractal colored image compression using moments features," *2008 Digital Image Computing: Techniques and Applications (DICTA 2008)*, pp. 486-490, 2008.

[13] T. Kovács, "A fast classification-based method for fractal image encoding," *Image and Vision Computing*, vol. 26, pp. 1129-1136, 2008.

[14] L. E. George and E. Al-Hilo, "Speeding-up color FIC using isometric process based on moment predictor," *International Conference on Future Computer and Communication, ICFCC 2009*, pp. 607-611, *2009*.

[15] L. E. George and E. Al-Hilo, "Fractal color image compression by adaptive zero-mean method," *2009 International Conference on Computer Technology and Development*, pp. 525-529, 2009.

[16] T. M. Hasan and X. Wu, "An adaptive fractal image compression," *IJCSI International Journal of Computer Science Issues*, vol. 10, no. 2, pp. 98-110, 2013.

[17] J. Wang and N. Zheng, "A novel fractal image compression scheme with block classification and sorting based on Pearson's correlation coefficient," *IEEE Transactions on Image Processing*, vol. 22, pp. 3690-3702, 2013.

[18] X. Wang, D. Zhang, and X. Guo, "Novel hybrid fractal image encoding algorithm using standard deviation and DCT coefficients," *Nonlinear Dynamics*, vol. 73, pp. 347-355, 2013.

[19] M. Valarmathi, M. Sobia, and R. B. Devi, "Iteration-free fractal image compression using Pearson's correlation coefficient-based classification," *Informatics and Communication Technologies for Societal Development: Proceedings of ICICTS 2014*, pp. 157-166, Springer, 2015.

[20] K. Jaferzadeh, I. Moon, and S. Gholami, "Enhancing fractal image compression speed using local features for reducing search space," *Pattern Analysis and Applications*, pp. 1-10, 2016.

[21] Y. Chen and P. Hao, "Integer reversible transformation to make JPEG lossless," *Proceedings 7th International Conference on Signal Processing (ICSP'04)*, vol. 1, pp. 835-838, 2004.

[22] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *IEEE Transactions on Image Processing*, vol. 3, no. 3, pp. 327-331, May 1994.

[23] S. V. Veenadevi and A. G. Ananth, "Fractal image compression using quadtree decomposition and Huffman coding," *Signal Image Processing (SIPIJ)*, vol. 3, no. 2, pp. 207-212, 2012.

[24] S. Pandey and M. Seth, "Hybrid fractal image compression using quadtree decomposition with Huffman coding," *International Journal of Science and Research (IJSR)*, vol. 3, no. 6, pp. 943-948, 2014.

[25] Z. J. Ahmed, L. E. George, and Z. S. Abduljabbar, "Fractal image compression using block indexing technique: A review," *Iraqi Journal of Science*, vol. 61, no. 11, pp. 1798-1810, 2020.

[26] B. A. Sultan, L. E. George, and N. F. Hassan, "The use of quadtree range domain partitioning with fast double moment descriptors to enhance FIC of colored image," *ARO-The Scientific Journal of Koya University*, vol. 6, no. 1, pp. 13-22, 2018.

[27] S. L. Mahmoud, "The use of double moment-based descriptors to speed up FIC," *Al-Nahrain Journal of Science*, vol. 15, pp. 200-202, 2012.

[28] L. E. George and S. L. Mahmoud, "Image steganography using an accelerated affine block matching scheme," *International Journal of Computer Information Systems*, vol. 2, pp. 1-7, 2011.