IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 24, 2020, accepted March 9, 2020, date of publication March 13, 2020, date of current version March 25, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980747

Impact of Spatial Dynamic Search With Matching Threshold Strategy on Fractal Image Compression Algorithm Performance: Study

ABDUL-MALIK H. Y. SAAD[1], (Member, IEEE), MOHD ZAID ABDULLAH[1], NAYEF ABDULWAHAB MOHAMMED ALDUAIS[2], (Member, IEEE), AND HISHAM HAIDER YUSEF SA'AD[3]

[1]School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Nibong Tebal 14300, Malaysia
[2]Faculty of Computer Science and Information Technology (FSKTM), Universiti Tun Hussein Onn Malaysia, Parit Raja 86400, Malaysia
[3]Faculty of Computer Science and IT, Al-Razi University, Sana'a, Yemen

Corresponding author: Abdul-Malik H. Y. Saad (eng.abdulmalik@gmail.com)

This work was supported by the Universiti Sains Malaysia (USM) through the USM Short-Term Grant 304/PELECT/6315292.

ABSTRACT Fractal image compression (FIC) is a very popular technique in image compression applications due to its simplicity and superior performance. However, it has a major drawback, which is the long encoding time. This is due to the requirement of performing huge similarity search for encoding each small portion in the image. Thus, reducing the search time of FIC while keeping the quality of reconstructed images at acceptable level is still an active research topic. Therefore, this paper has focused on studying the search problem of the conventional full-search FIC algorithm and the impact of employing a spatial dynamic search technique instead with the matching threshold strategy. Unlike the conventional full-search method that is a spatially static where the search starts from a fixed position (normally from the top-left corner of the image to the bottom-right corner) regardless of the position of the range block being encoding, the idea of the dynamic search method is simple, but effective, and it is based on starting the search from the closest domain block to the range block that needs to be encoded. These two search schemes are tested under different matching threshold values, in which the search is terminated whenever a domain block with an acceptable matching level is found. To make the study comprehensive, the test is performed for different image sizes and types, range block and partitioning step sizes, and quantization levels. The experimental results show the significant impact of using the dynamic search method instead of the conventional search method specifically when the threshold is large. For the best encoding parameters, the improvement amount that can be achieved is near to 90 % in terms of search reduction and 1.6 dB PSNR in terms of image quality.

INDEX TERMS Fractal image compression, fast search method, full search scheme.

I. INTRODUCTION

Nowadays, images are produced in huge numbers from countless devices used in our daily life, such as mobile phones, surveillance systems, medical devices, etc. As a result, the amount of data produced every single minute is extremely huge, thus making the process of storing these data on digital devices a very costly problem. Furthermore, transmitting such big data through the internet network requires a high-speed and unlimited data package, which adds an additional cost into the operating expenses. Therefore, an efficient solution is to use one of the compression methods to reduce the transmission cost and time, as well as storage volume.

Among the most-effective image compression methods, FIC method is simple and has unique features such as resolution independent and high compression rate [1]–[3]. Due to these features, FIC has been not only used for compression ordinary images, but also for compression images like magnetic-resonance-imaging (MRI) [4], [5] and hyperspectral types [6], [7]. In general, fractal algorithm is a lossy compression method which is reliant on portioned iteration function system (PIFS). It attempts to find out the similarities between the image blocks and use them for encoding the image [8]. In the encoding process, an image is partitioned

The associate editor coordinating the review of this manuscript and approving it for publication was Gerardo Di Martino.

into small non-overlapping blocks called range blocks, and large blocks called domain blocks (usually four times larger than the range block size). For each range block, the search process attempts to find the best-matched domain block. To do so, all domain blocks need to be examined and the one with the least matching error is selected as the best-matched domain block [2]. Since each range block requires thousands of matching processes, the overall procedure is very time consuming. This is considered as the main drawback for FIC [9]. Therefore, many researchers seek to develop new fractal-based compression methods which can reduce the number of searches as means for speeding-up the encoding time [4]. So far, various schemes and speed-up methods have been proposed [10]–[13]. Depending on the search approach used, the developed FIC methods can be categorized into three different types: (1) full search [13], (2) partial search [10], [11], and (3) searchless [12] based methods. Among these search approaches, the full-search is considered the most-time consuming. This is due to the fact that for each range block, the entire image must be searched in order to find the best-matched domain block. On the other hand, the partial search is the second most-time consuming approach that reduces the search space in order to improve the encoding time; generally, the smaller the search space size, the lower the encoding time. Based on this approach, it can be found in literature hundreds of published articles that introduce different methods for decreasing the search amount, such as using genetic algorithm [14], [15], DCT [16], [17], DWT [7], [18], standard deviation [17, 20], nearest neighbour search [20], [21], feature vector [22], etc in Although all these developed partial-search FIC methods have successfully reduced the encoding-time, they suffer from two main weaknesses. First, the quality of the decoded image is relatively low as the best-matched domain blocks can be discarded when the search space is limited. Second and the most important one is, they usually use complex approaches, along with the fractal algorithm, to help for reducing or limiting the search. This is resulting to increase the complexity of the implementation, especially if they are going to be implemented in hardware. Thus, these method rarely utilized in developing fractal-based coding chips. Since this work is the kernel for designing a real-time fractal encoding hardware architecture, the developed method must be efficient, simple as possible and hardware friendly.

As the encoding time of the partial-search based methods is still high and not always sufficient for high-speed applications, the researchers have also proposed various fractal algorithms based on the searchless approach such as [23], [24]. Although these methods are the fastest among the others, they are suffering from poor reconstruction fidelity [12], [25]. When the researchers try to overcome the fidelity problem through encoding the image blocks in small sizes, another drawback arose which is the low compression rate. In this case, the range block size can be smaller as 2×2 or even 1×1 pixel in order to maintain the image quality at an acceptable level [26]. In conclusion, this type of method is not useful for the applications targeting good reconstruction image quality or high compression rate.

As full-search FIC method is usually much simpler and has relatively higher compression rate and image quality compared to the partial- and non-search based methods, this paper aims to study its search problem and the advantage of the dynamic search approach in comparison to the conventional static search approach. Unlike the later approach, the former is based on giving a search priority to the closest domain blocks over the farthest one. In this case, for a given range block, the search process will start from the spatial nearest domain block and expand further until finding an acceptable matched domain block or covering the entire image. The acceptance level is predefined with a matching threshold value based on the image quality requirement. In this case, the bigger the threshold, the faster the matched domain block is found, but the lower the compression quality obtained.

To the best of the authors knowledge, all developed full-search FIC methods are based on the conventional search approach. Accordingly, the search process starts from a certain position, usually from the top left corner of the image to the bottom right corner, regardless of the position of the range block required to be encoded. In this case, there is no priority giving to such domain block over the others. The adoption of this approach is, in fact, due to the results of a previous study done by Fisher [27]. In this study, it has been computed the distributions of the differences in the positions of the range blocks and their matched domain blocks. The study results show that there is no real advantage of the close domain blocks over the far ones in general. Consequently, the developed full-search based methods such as [13], [28] did not give higher priority for the adjacent domain blocks. However, when looking deeply into Fisher [27] study, it can be seen that the study's results are only valid for a special case; that is, no matching threshold (i.e. $thr = 0$) was applied in the search process. For the other case that is typically used (i.e. when a threshold is set), the advantage of the close domain blocks has not been tested comprehensively. Therefore, this paper aims to study the impact of utilizing the dynamic search approach when the matching threshold is applied. Thus, the number of searches performed to encode an image and the quality of the decoded image are analysed for various threshold values, image types, partitioning steps and range block sizes. Besides, the typical quantization bit sizes used for the contrast scaling coefficients have also been considered in order to study their effect on the search number. The overall results show a significant reduction in the search amount when utilising the dynamic search strategy compared to the conventional search strategy. Moreover, the decoded image quality is generally improved. Thus, adopting this approach in such hardware works as in [13], [28] or in similar future works will result to improve their encoding-time performance significantly with only a slight modification required in their architectures, particularly the memory control unit.

This paper is organized as follows: Section II includes a review of related works. Then, the conventional full

search FIC algorithm is illustrated in Section III. Section IV describes the dynamic search scheme. Section V shows the experimental results and the study findings. Finally, a conclusion is made in Section VI.

II. RELATED WORK

In the literature, there are several published works that focus on reducing the encoding time of the fractal method, with the help of various techniques [10], [14], [16], [17], [19], [29]–[32]. For instance, Jaferzadeh, *et al.* [33] have introduced a fractal method based on a new local binary feature scheme that is similar to the local binary patterns method Wang, *et al.* [34] have developed a fractal compression method based on quadtree division, neighbour search and asymptotic strategy. In another work, Wang, *et al.* [35] have used the standard deviation feature to segment the range blocks into two classes, where the range blocks of only one class are encoded by searching of the suitable matched domain block with the help of particle swarm optimisation technique, while the range blocks in the other class are encoded directly without search by storing their average values. Similarly, but with more defined classes, the research articles [10], [22], [36], [37] have introduced different methods to divide the image blocks into groups, and restricting the search within the blocks of the same class only. Quite similar to this approach is the approach of neighbourhood search, or nearest neighbour search, where the search is restricted to a small set of the domain blocks that share similar feature vector with the corresponding range block [20], [21], [38], [39]. This approach has been employed in both spatial and frequency domain using discrete cosine [40] and discreet wavelet [41] transforms.

In the same track, Kovács [10] has proposed a fractal encoding method based on an improved classification scheme. With this scheme, the image blocks are sorted into disjoint classes with the help of two parameters, i.e. the direction of approximate first derivative and the normalized root-mean-square error of the fitting plane in the corresponding block. In [17], the image blocks are classified into smooth, diagonal/sub-diagonal edge and horizontal/vertical edge classes. The classification task is performed using only the lowest horizontal and vertical DCT coefficients of the given block. In [18], fast wavelet transform-based classification is introduced. On the other hand, Tong and Pi [31] have proposed fractal compression method based on reducing the domain pool size through removing unqualified domain blocks. This requires compute the standard deviations STDs for each range and domain blocks, and for a pair of range and domain blocks, if the difference in their computed standard deviation value is larger than the defined threshold, the domain block will be considered as an unqualified block and hence be rejected. Four years later, Wu, *et al.* [19] has enhanced the Tong's STD method by introducing a domain intelligent classification algorithm that classify the domain blocks based on their STD values and those with a similar STD values are grouped together in one class. For a given range block, STD value is computed first to determine the proper class; the class that contains domain blocks of similar STD values; for the search process. However, Wang, *et al.* [16] found that, for some range blocks which have more texture characteristics and few self-similarity, the quality of the reconstructed image is poor. To solve this problem, they have proposed a hybrid encoding method using STD and DCT.

Using hybrid methods to further investigate new techniques, Xing-Yuan, *et al.* [14] have employed genetic algorithm and simulated annealing algorithm with fractal method. Similarly, Wu, *et al.* [15] have used genetic algorithm but with schema theorem to help restrict the search on a particular region Jaferzadeh, *et al.* [30] utilized fuzzy clustering and DCT algorithms. Although all these discussed approaches were able to improve the encoding time noticeably, however, their attained speeds which usually above 1 s are still not sufficient for the most of high-speed applications, such as those producing numerous of images every second like video applications. In order to solve this problem, in fact, it is required to have hardware solutions. To do that, it must first develop simple, efficient and hardware-friendly fractal compression algorithm that incudes simple and less number of operations. These features are not available in the previous discussed methods which contain pre-processing tasks and/or hybrid complicated algorithms. For this reason, it has been taking into account developing a fractal image compression algorithm that is simple, but efficient, and does not require massive additional operations.

III. FULL-SEARCH FRACTAL IMAGE COMPRESSION ALGORITHM

Fractal image compression algorithm searches for a set of transformations that map an image into itself. In essence FIC divides an image into non-overlapping $r \times r$ square blocks called range blocks (referred to as R) and overlapping $d \times d$ blocks called domain blocks (referred to as D). The overlapping partitioning for constructing the domain blocks is performed with step-size stp that can be in the range of 1 to d, the larger the stp value, the lower the number of the domain blocks constructed. The domain blocks number Nd for an $W \times H$ image size can be computed by the following formula:

$$Nd = \left(\frac{(W-d)}{stp} + 1\right) \times \left(\frac{(H-d)}{stp} + 1\right) \tag{1}$$

Since the size of the domain blocks is generally 4 times larger than the range blocks' sizes (i.e. $d = 2 \times r$), every domain block must be spatially contracted to the size of the range block during matching process. Contraction is performed by taking the average of every 2×2 pixel.

For increasing the match level between a couple of range and domain blocks, the affine transform is applied. The affine transformer adjusts the intensity values of the contracted domain block by multiplying them with a particular contrast scaling factor σ and then adding brightness offset value β to

the results. The scaling factor must lie between −1 and 1 to ensure the contractivity of PIFS [24], [42]. The transformation is given as

$$T(D) = \sigma.D + \beta.I \tag{2}$$

where D is the contracted domain block, and I is all-ones matrix of $r \times r$ dimension (i.e. all the elements are equal to one). The similarity between two blocks is measured by mean-squared error (MSE) as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(R_i - (\sigma \cdot D_i + \beta))^2 \tag{3}$$

where N is the number of pixels in the range block (i.e. $N = r \times r$), R_i and D_i are the i_{th} intensity values of range and contracted domain blocks, respectively. To minimize the matching error computed by (3), the coefficients σ and β require to be computed as follows:

$$\sigma = \frac{N\sum_i^N R_i D_i - \sum_i^N R_i \sum_i^N D_i}{N\sum_i^N D_i^2 - \sum_i^N D_i \sum_i^N D_i} \tag{4}$$

$$\beta = \frac{\sum R - \sigma \sum D}{N} \tag{5}$$

Following the computation, the values of σ and β are quantized into specific bit-size, usually between 5 to 2 bits for σ and 7 bits for β. For a given R, D blocks are searched and matched one by one in order to find D with MSE less than a pre-defined threshold value, thr. The fractal codes associated with the located matched D need to be stored for decoding purpose. The codes are σ, β and the coordinates for located matched D (denoted as (xd, yd) for x- and y-axes). Altogether for all R will form the compressed image file. Therefore, the size of the compression file CF_{Size} can be computed by the following expression:

$$CF_{Size} = N_R\left(\sigma_{BitSize} + \beta_{BitSize} + x_{d_{BitSize}} + y_{d_{BitSize}}\right) \tag{6}$$

where N_R is the total number of the range blocks in the encoded image and equal to $N_R = (W \times H)/(r \times r)$). The remaining coefficients are the bit-sizes of the fractal codes. Based on (6), the compression ratio CR can be calculated as $CR = OF_{Size}/CF_{Size}$, where OF_{Size} is the original image file size.

The conventional full-search mechanism is demonstrated in Fig. 1. As it is clear from the figure, the search is performed based on Raster Scan method, where the search starts from the domain block located at the top-left corner of the image and ends with the domain block at the bottom-right corner. For every range block, the starting point is always the same regardless of its location. The search can be stopped whenever an acceptable matched domain block is found. This process is repeated for each R.

IV. DYNAMIC FULL-SEARCH METHOD

Unlike the conventional full-search method, the spatially dynamic search method is based on giving a priority for the nearest domain blocks over the farthest ones. In this case, the search will start from the closest domain block for the corresponding range block and moving further away from this point until an acceptable matched domain block is found or the whole image is searched. To demonstrate the search mechanism, two scenarios of range block locations are shown in Fig. 2, where Fig. 2(a) shows the search scenario for the range block R located at the centre of the image, and Fig. 2(b) shows the search scenario for the range block R located at the top-left corner. For both scenarios, it is obvious that the closest domain block for R is the domain block that completely contains R inside it, and it is denoted as D_0 in Fig. 2. In the case where D_0 does not match with R, the search space is expanded one-step in all directions to involve the second-closest domain blocks in the search. This will form a search window size of 3×3 domain blocks. The expanding process is continued further by increasing the search window size into 5×5, 7×7 and so on until an acceptable matched domain block is found or the entire image is searched. At every time the search is expanded, the search will start from the domain block that lies on the same horizontal line as D_0, but from the right side (e.g. D_1 in Fig 2. (a) and D_1 and D_4 in Fig. 2(b)), and then moves clockwise as illustrated by the arrows drawn in the figure.

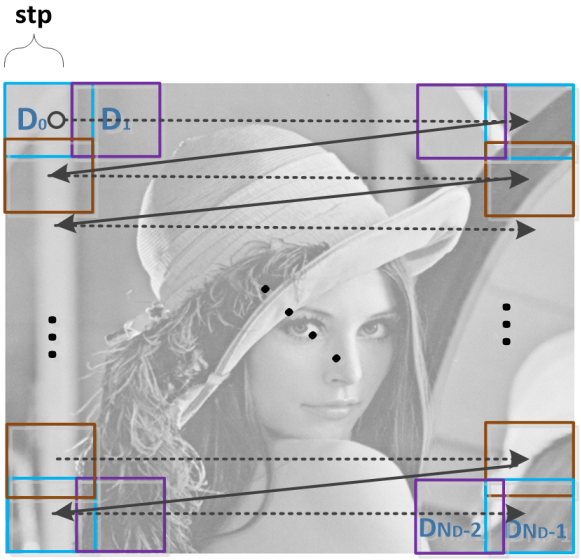

FIGURE 1. Conventional full-search mechanism demonstration.

As seen in Fig. 2(b), the domain blocks do not surround the range block from all directions, only from the right and bottom sides, therefore the search can be expanded only in these two directions. In other words, the domain blocks inside the search window as well as the image will be only considered and the others will be ignored (i.e. having a negative coordinate). For example, the 3×3 search window shown in Fig2. (b) includes only four domain blocks inside (i.e. from D_0 to D_3) compared to nine domain blocks (i.e. from D_0 to D_8) as in Fig. 2(a).

The flowchart of the full-search fractal image encoding algorithm used for evaluation the dynamic and conventional search approaches is shown in Fig. 3. As shown in the

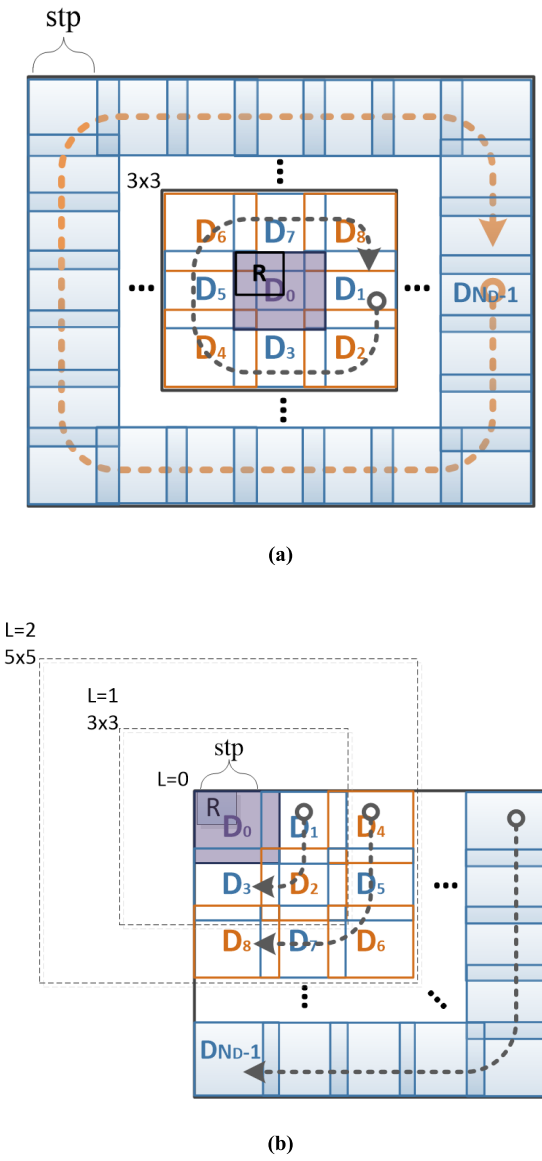

FIGURE 2. Dynamic search mechanism examples for range blocks existing in: (a) the centre of the image, and (b) the top left corner of the image.

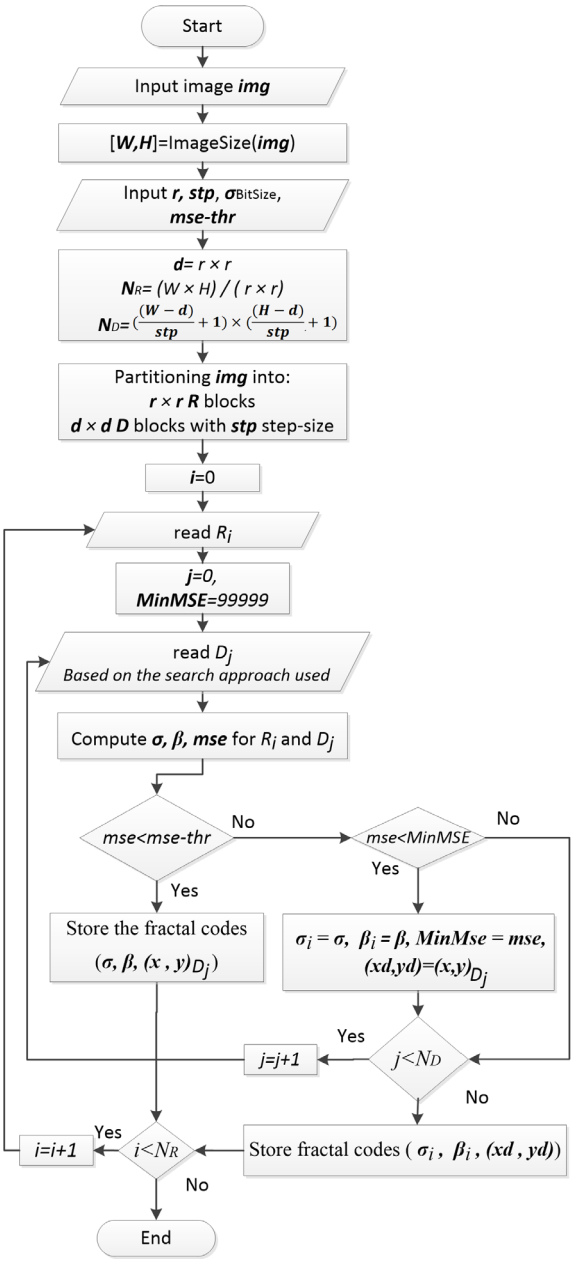

FIGURE 3. Flowchart of the tested full-search fractal image compression algorithm used for evaluation dynamic and conventional search approaches.

flowchart, the image is input first and then its size, the width W and height H, is measured. In order to study the impact of using dynamic search approach over the conventional static search approach under various typical coding parameters, the range block size r, the partitioning step-size of domain blocks *stp*, the quantization bit-size of the contrast scaling coefficient $\sigma_{BitSize}$, and the *MSE* threshold *mse-thr* need to be defined in the beginning. With these defined parameters, the number of range blocks N_R and domain blocks N_D are computed to determine the end of search and encoding. For each $R_i, \forall i = \{0, 1, \ldots, N_R - 1\}$, the domain blocks $D_{j=0,1,\ldots N_D-1}$ will be matched one by one, starting from D_0 onwards. As mentioned earlier, the location of D_0 and other domain blocks are fixed for the conventional search method, but not for the dynamic search method. Therefore, for each R_i, the locations of all D_j (i.e. $(x, y)_{D_j}$) must be computed again to coincide with what is shown in Fig. 2. Algorithm 1 shows in details the computation performed and the whole search process of the dynamic approach.

In the matching process of a pair of R_i and D_j, the following coefficients σ, β and *mse* are computed with the equations (4), (5) and (3), respectively. If *mse* is less than or equal to the defined *mse-thr* value, then D_j will be considered as the matched domain block for R_i. In this case, the corresponding values of σ, β together with the coordinates of $D_j(x, y)_{D_j}$ will be stored as the fractal codes for R_i. If not, the search will continue until finding a D_j with $mse \leq mse\text{-}thr$ or reaching the end of search (i.e. when $j = N_D - 1$). For the latter

case, that the search has reached the end without finding the desirable matched domain block, the fractal codes of the most-suitable D_j will be stored instead. To do so, every time found D_j with $mse \leq MinMSE$ (the lowest mse obtained so far), its fractal codes are stored in temporary variables (i.e. σ_i, β_i and (xd, yd)). Hence, if the search is reached the end, these variables will retain the fractal codes of the most-suitable matched D_j, and they will be stored easily in the compression file. These processes are repeated for each R_i in the encoded image.

V. RESULTS AND DISCUSSIONS

Full-search fractal image compression using dynamic and conventional static search methods is implemented on MATLAB R2018a to study if there is a preference of starting the search from the nearest region for the range block over the starting from a fixed-region regardless of the locations of the range blocks. To do so, a dataset of eight grayscale images that have been shown in Fig. 4 are used in the study. In this dataset, there are four images for each 256×256 and 512×512 image size. The 256×256 images are Lena, LiftingBody, CameraMan and LivingRoom, and the 512×512 images are Goldhill, Lake, Aerial and Elaine. These images with different sizes and textures are used to make the study more comprehensive.

The performance of both search methods is analyzed using (1) the number of searches required for encoding an image, and (2) the peak signal-to-noise ratio (PSNR) for the decoded image. For particular image and encoding parameters, the number of searches required by each search method is computed, where the lower the search number computed, the higher the encoding speed that will be achieved. The amount of speed improvement will approximately equal the amount of improvement in the searches number as it is the dominant task. For the second performance parameter, which is the quality of the decoded image, it has used the PSNR formula as:

$$PSNR = 10\log_{10}\frac{255 \times 255}{\frac{1}{W \times H}\sum\left(f - \tilde{f}\right)^2}$$

where f and $\tilde{f}$ are the original and decoded images, respectively.

In addition to testing the dynamic search approach with different image types and sizes, it has also been tested under different range block sizes r, partitioning steps stp, contrast scaling coefficient bit-sizes $\sigma_{BitSize}$, and matching thresholds thr. Since these parameters are set differently according to performance requirement, but with regular values such as $r = 4, 8$, $stp = 4, 8$, and $\sigma_{BitSize} = 2, 5$, the combination of all of these common values have been considered in the evaluation. All together, they form eight combinations. For each combination, various thr values between 0 to 700 are tested to study the efficiency of the dynamic search method when the value of thr increases. The results from all these tested parameters lead to have a broad view.

Algorithm 1 Dynamic Search Algorithm

BEGIN

1 **INPUT**: $-r, stp, N$ //N *is the image width or height, considering square image size*
2 **OUTPUT**: −FRACTAL CODES
3 $N_r = \frac{N}{r}; N_d = \left(\frac{N-2r}{stp}\right)+1;$ //N_r *and* N_d *are the number of range and domain blocks in one row or column, respectively.*
4 FOR EACH $R_i i = 0 \cdots (N_r)^2 - 1$
5 // *Compute x and y coordinates of* R_i *and* $D_{j=0}$
6 $x_r = mod\left(\frac{i}{N_r}\right) * r; y_r = int\left(\frac{i}{N_r}\right) * N_r;$
7 $x_{D_0} = int\left(\frac{x_r}{stp}\right) * stp;\ y_{D_0} = int\left(\frac{y_r}{stp}\right) * stp;$
8 $j = 1;$
9 FOR $l = 1$ TO $\frac{(N-2r)}{stp}$ // l determines the search window size
/∗*Compute the coordinates of every* D_j *in* $(2l+1) \times (2l+1)$ *search window size as follows*∗/
//With clock-wise search direction and starting from the D_j *that lies on the same horizontal line as* D_0 *but from the right-side.*
//First, get $D_j$located at the low-half of the right-side of the search window
10 $x_{D_j} = x_{D_0} + l * stp;$
11 FOR $y_{D_j} = y_{D_0}$TO $y_{D_0} + l * stp$ STEP stp
12 IF VALID $(x_{D_j} y_{D_j})$ THEN *//to ignore the coordinates not in the image*
13 *Read* D_j *at* $(x_{D_j} y_{D_j})$
// ….***Do Matching process for*** $\mathbf{R_i}$ ***and*** $\mathbf{D_j}$ ***blocks*** ….
14 $j = j + 1$
15 END IF
16 END FOR
// Get D_j located at the bottom-side
17 $y_{D_j} = y_{D_0} + l * stp;$
18 FOR $x_{D_j} = x_{D_0} + (l-1) * stp$ TO $x_{D_0} - l * stp$ STEP stp
19 *Check validity and do matching as in lines* $12-15$
20 END FOR
// Get D_j located at the left-side
21 $x_{D_j} = x_{D_0} - l * stp;$
22 FOR $y_{D_j} = y_{D_0} + (l-1) * stp$ TO $y_{D_0} - l * stp$ STEP stp
23 *Check validity and do matching as in lines* $12-15$
24 END FOR
// Get D_j from the top-side
25 $y_{D_j} = y_{D_0} - l * stp;$
26 FOR $x_{D_j} = x_{D_0} - (l-1) * stp$ TO $x_{D_0} + l * stp$ STEP stp
27 ***Check validity and do matching as in lines 12-15***
28 END FOR
// Last, get $D_j$located at the top-half of the right-side
29 $x_{D_j} = x_{D_0} + l * stp;$
30 FOR $y_{D_j} = y_{D_0} - (l-1) * stp$ TO $y_{D_0} - stp$ STEP stp
31 ***Check validity and do matching as in lines 12-15***
32 END FOR
33 END FOR
34 END FOR

END

For clarity, it has presented the results in two sub-sections; the first sub-section presents the results from having a small range block size (i.e. $r = 4$), while the second sub-section presents the results from having a large range block size (i.e. $r = 8$).

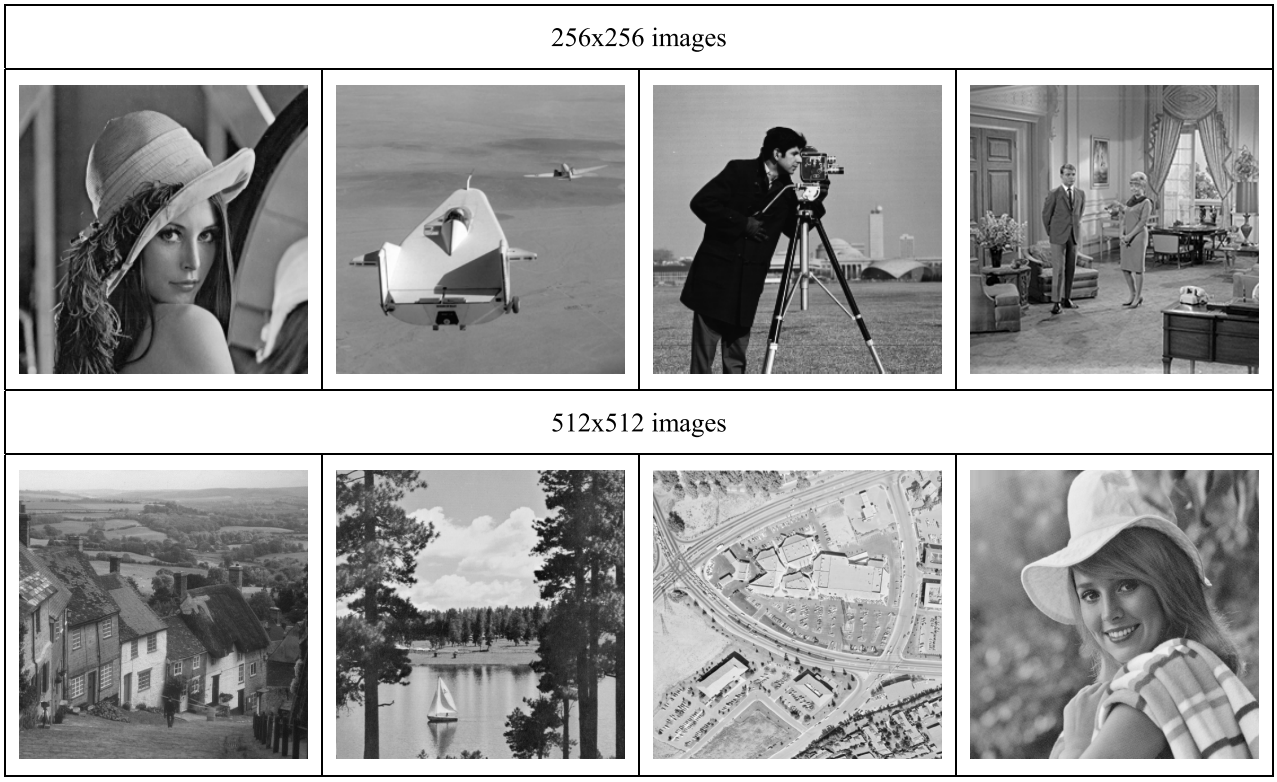

FIGURE 4. Dataset of eight standard images of different textures and sizes used for evaluating the dynamic search FIC method.

A. RESULTS FOR 4 × 4 RANGE BLOCK SIZE

For this size of range block, the dynamic and conventional search methods have been tested for all 256 × 256 images in the dataset and for various *thr* values. The amounts of search reduction achieved by utilising the dynamic search method are shown in Fig. 5. The embedded four graphs in this figure show the results from the four defined combinations of *stp* and $\sigma_{BitSize}$ values. As it is clear from this figure, the percentage of the search reduction accomplished increases steadily as *thr* increases. For the values of *thr* near to zero, the dynamic and the conventional search methods have a similar search amount and there is no clear advantage for the close domain blocks over the farther ones. In fact, this result is similar to that found by Fisher [27]. However, for the other case, i.e. $thr \gg 0$, the improvement is clear and steady, which indicates an increase in the probability of finding an appreciate matched domain block close to the corresponding range block. On other words, when $thr \gg 0$, it can be said that there is a preference for the nearest domain blocks over the farthest ones. However, the improvement amounts as shown in Fig. 4 are different for each combination of *stp* and $\sigma_{BitSize}$ values, and images.

Among the four combinations of *stp* and $\sigma_{BitSize}$ values, the combination of $stp = 4$ and $\sigma_{BitSize} = 5$ has the best search reduction for all four images as shown in Fig. 5(a). This was expected as this combination holds the smaller *stp* and the higher $\sigma_{BitSize}$ values. Accordingly, the encoded image has been partitioned into greater number of domain blocks N_D (exactly four times of that for $stp = 8$, use (1) to calculate the exact number of N_D for the both situations), and therefore there are greater chances to find an acceptable matched domain block near to the encoded range block. Additionally, the higher defined precision of σ makes the convergence faster as the candidate domain block becomes more fit to the mapped range block. As a result, the dynamic search was able to reduce the searches number by more than 40% for all tested images and about 90% for LiftingBody image when $thr = 700$.

For the lower precision of σ (i.e. $\sigma_{BitSize} = 2$) as in Fig. 5(c) or the larger *stp* (i.e. $stp = 8$) as in Fig. 5(b), the amounts of search reduction achieved by the dynamic search method are approximately similar, but lower than that shown in Fig. 5(a) for $stp = 4$ and $\sigma_{BitSize} = 5$. Among the tested images, the amounts of decreases for Lena and LiftingBody images were higher than that for CameraMan and LivingRoom images. For example, at $thr = 700$, the amount of decrease was over 20 % for Lena and Lifting body images, and only around 5% for the remaining images. The reason for that is more likely because the image of Camera-man and Livingroom has generally larger small details compared to the image of Lena and Lifting Body as shown in Fig. 5, and this makes the preference of the close domain blocks less.

For the last combination shown in Fig. 5(d) for lower $\sigma_{BitSize} = 2$ and the larger $stp = 8$, the percentage of

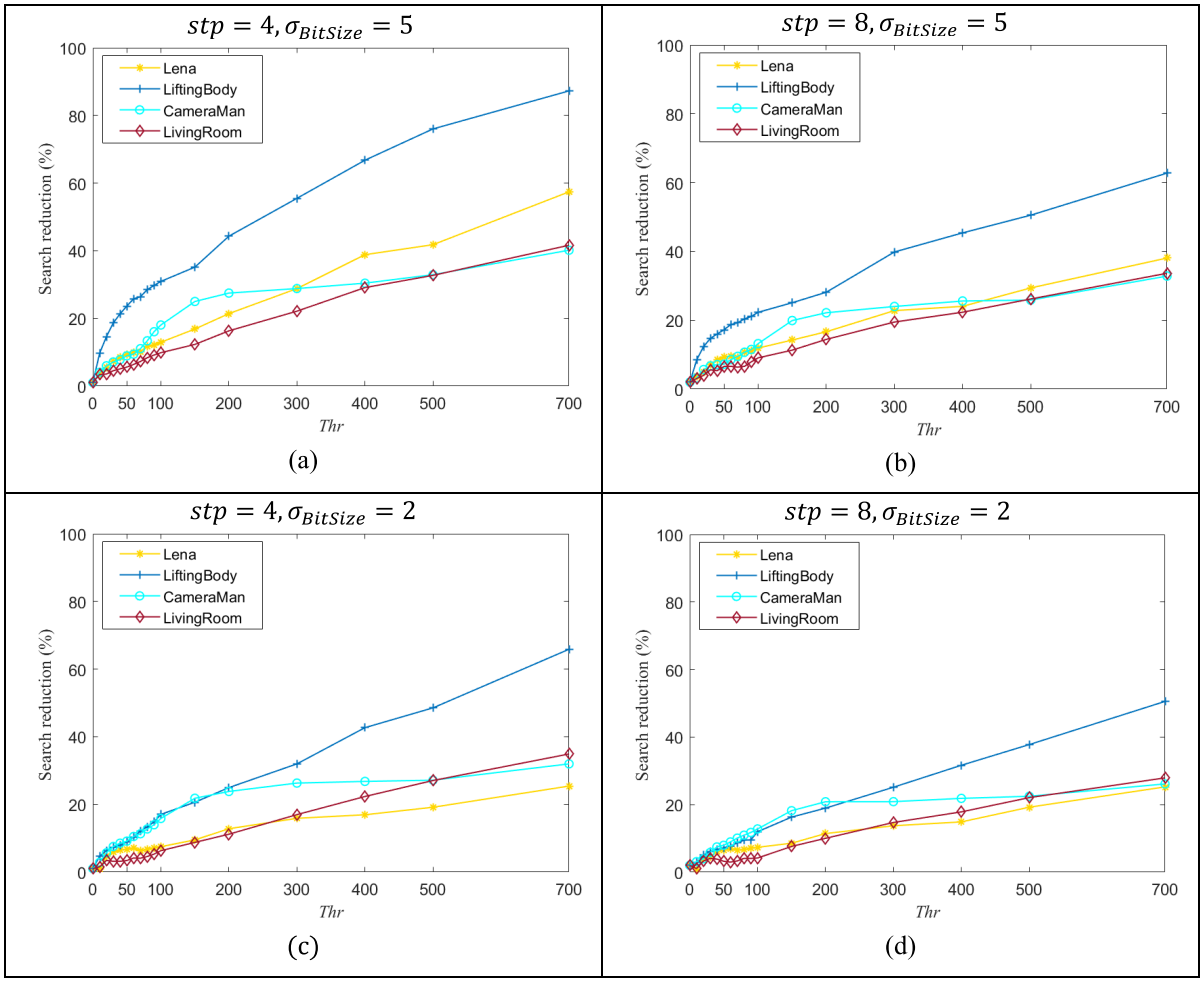

FIGURE 5. Dynamic search scheme performance in term of search improvement for various *thr* values, four 256 × 256 sample images, $r = 4$, and the combinations of (a) $stp = 4, \sigma_{BitSize} = 5$, (b) $stp = 8, \sigma_{BitSize} = 5$, (c) $stp = 4, \sigma_{BitSize} = 2$, and (d) $stp = 8, \sigma_{BitSize} = 2$.

search improvement achieved by dynamic search method was the lowest compared to the results of other combinations. This is expected as both defined values for *stp* and $\sigma_{BitSize}$ were the worst. However, the dynamic search method was able to reduce the search significantly when *thr* is large. For example, at $thr = 700$, the search reduction is more than 25 %.

B. RESULTS FOR 8 × 8 RANGE BLOCK SIZE

Similar to the previous section for 4 × 4 range block size, but for 8 × 8 range block size in this section, the dynamic search method has been tested with all combinations of $\sigma_{BitSize}$ and *stp* and the results are shown in Fig. 6. Comparing the results shown in this figure with that in Fig. 5, it can be seen that the amount of search reductions obtained from 8 × 8 range block size is lower than that for 4 × 4 range block size for all combinations. This is not strange since for larger block size, the matching degrees between image's blocks become lower generally. For this reason, the similarity level between the adjacent blocks is also reduced, resulting in less preference of the close domain blocks over the farther ones.

An in-depth look into the results of each combination in Fig. 6 has clearly shown that the amount of search reduction shown in Fig. 6 (a) for the combination $\sigma_{BitSize} = 5$ and $stp = 4$ was the highest among others, then the results from the combinations involving either $\sigma_{BitSize} = 2$ or $stp = 8$ as in Figs. 6 (b) and (c), and the last is the results from the combination $\sigma_{BitSize} = 2$ and $stp = 8$ shown in Fig. 6 (d). For the best combination, the amount of search reduction started to improve noticeably from $thr > 100$ for all tested images, except for the LiftingBody image, where it started immediately from $thr > 0$. At $thr = 700$, improvement amounts were between 20 % to 50 %.

For the results of the second best combinations shown in Figs. 6 (b) and (c), it can be seen that there are around 10 to 15 % drop maximum in the search reduction amount compared to the results of the best combination in Fig. 6(a). It can be seen also that the results from the combination with high precision of σ (i.e. $\sigma_{BitSize} = 5$) are slightly better than that with low *stp* (i.e. $stp = 4$). This is similar also to that for r = 4.

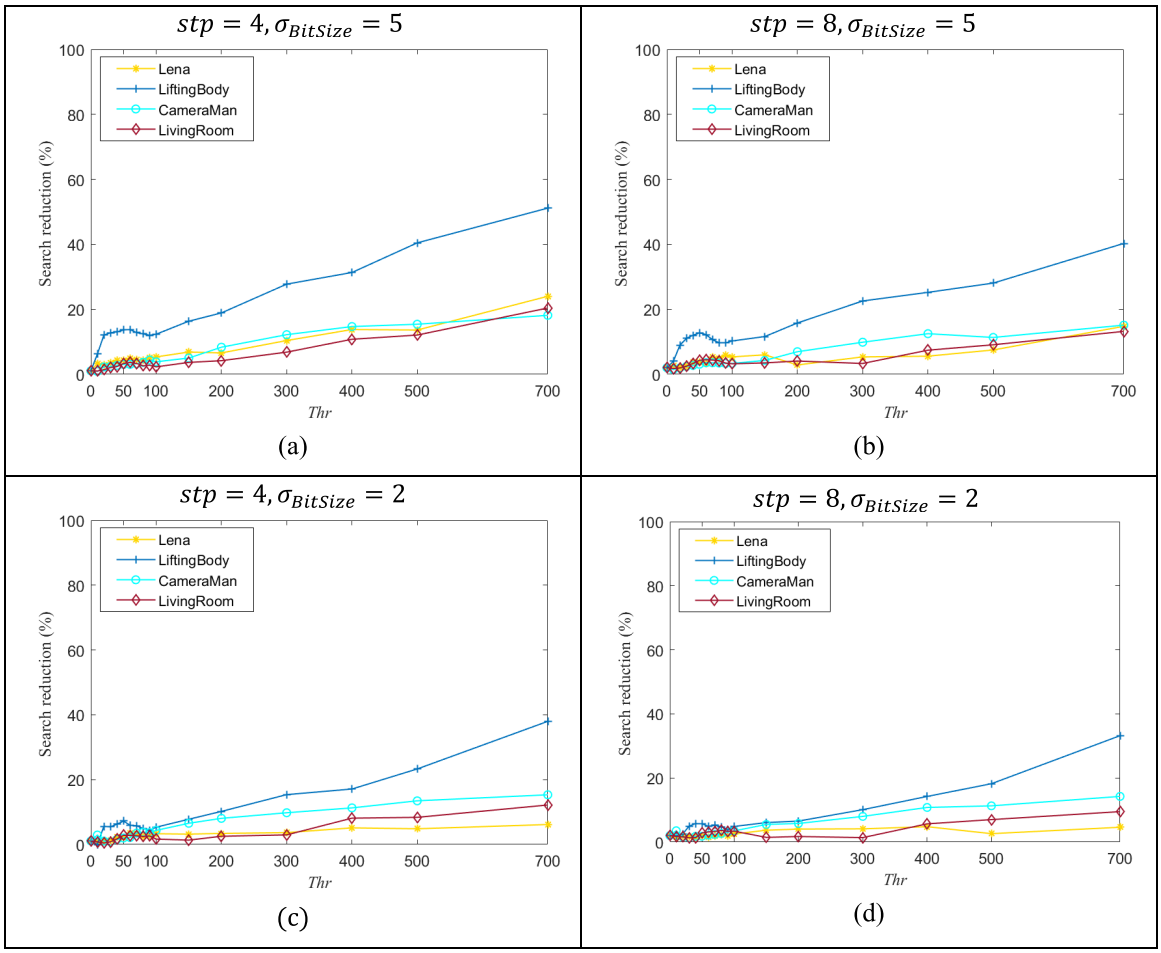

FIGURE 6. Dynamic search scheme performance in term of search improvement for various *thr* values, four 256 × 256 sample images, $r = 8$, and the combinations of (a) $stp = 4$, $\sigma_{BitSize} = 5$, (b) $stp = 8$, $\sigma_{BitSize} = 5$, (c) $stp = 4$, $\sigma_{BitSize} = 2$, and (d) $stp = 8$, $\sigma_{BitSize} = 2$.

For the results of the last combination shown in Fig. 6 (d), the improvement amounts are the lowest, but are still comparable to the results in Fig. 6 (c). The drop in the performance of the dynamic search scheme is, as mentioned earlier, because the similarity degree between adjacent blocks is reduced generally when the block and step sizes are larger, and the precision of contrast scaling coefficient is lower. As a result, the efficiency of the dynamic search method compared to the conventional static search method is dropped to the lowest level.

To make the comparison between the dynamic and conventional static search methods complete, it is necessary to take into account the quality of the decoded images resulting from each search method. Accordingly, PSNR has been computed for the resultant images in respect of the defined combinations of r, $\sigma_{BitSize}$ and stp. Based on the results obtained, it can be said that the dynamic search method is superior to the conventional static search method generally. However, for rare cases, the conventional search method shows slightly better PSNR values with a maximum increase of 0.2 dB. In fact, this amount of improvement is insignificant when it is compared to what can be achieved by the dynamic search method, i.e. nearly 10x superior. Fig. 7 is evidence for the superiority of the dynamic search method in respect of all combinations. In this figure, the average amount of improvement for all tested images at various *thr* values are shown, where Fig. 7 (a) shows the average improvements resulting from the four combinations of $\sigma_{BitSize}$ and stp when $r = 4$ and Fig. 7 (b) shows the results for the similar combinations but for $r = 8$. From these two sub-figures, it can be clearly seen that the results for $r = 4$ are better than that for $r = 8$, and this is also similar to the search reduction results shown in Figs. 5 and 6. This is due to the same reason mentioned earlier; the smaller the range block size r, the higher the degree of matching between the adjacent blocks, hence the search is reduced and the decoded image quality is improved when utilizing the dynamic search method. The amount of improvement becomes clearer when *thr* is large.

Among the results of the four combinations of $\sigma_{BitSize}$ and stp shown in Fig. 7 for both r = 4 and 8, the results from the best combination ($\sigma_{BitSize} = 5$ and $stp = 4$) still shows the best performance among the others, and then the results from

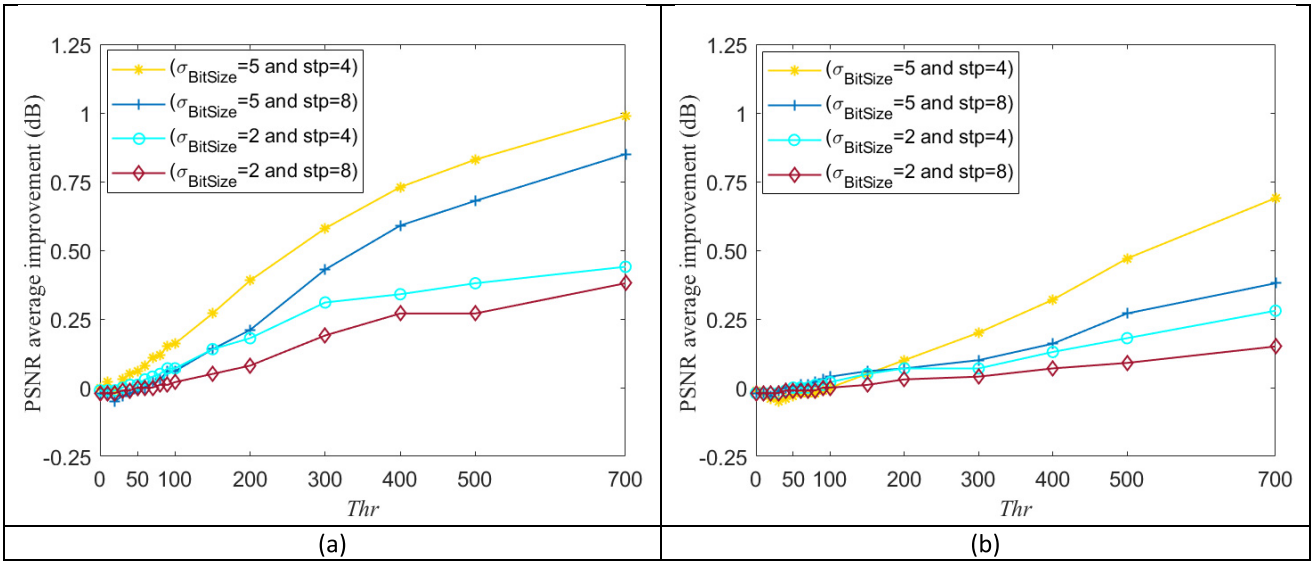

FIGURE 7. PSNR average improvement vs varies *thr* using dynamic search method for the four combinations of $\sigma_{BitSize}$ and *stp*, where (a) for $r = 4$ and (b) for $r = 8$.

the combinations ($\sigma_{BitSize} = 5$ and $stp = 8$), ($\sigma_{BitSize} = 2$ and $stp = 4$) and ($\sigma_{BitSize} = 2$ and $stp = 8$), respectively. This is identical to the order of each combination in the search reduction performance. From Fig. 7, it can also be seen that the improvement amount in the quality of the decoded images when adopting the dynamic search method is enlarged as *thr* is increased. From the best combination, the average improvement in PSNR at $thr = 700$ is about 1 dB for r = 4 and 0.7 dB for r = 8.

For subjective assessment, it has displayed a sample of decompressed images encoded by both the dynamic and conventional static search methods in Fig. 8. The displayed images are the results from the best combination of *r*, $\sigma_{BitSize}$ and *stp*, and $thr = 700$. From this figure, it is clear that the images encoded by the dynamic search method are visually superior to that for the conventional search method. It can also be seen that for this large defined *thr* value, the conventional search method yields more blocking artifacts compared to the proposed dynamic search method. This is because, when the *thr* value is large, the first located domain block with $mse \leq thr$ can be either a good match as *mse* value is close to zero or a low match as *mse* is close to the predefined *thr*. Hence, in the case of dynamic search method, which is starting the search from the adjacent domain blocks for the corresponding range block, the first located matched domain blocks are mostly having low *mse* as the similarity between the adjacent blocks is generally higher than that between the far ones. However, in the case of the conventional search method, the *mse* for the first located matched domain blocks are usually high.

For 512 × 512 image size, the dynamic search still shows superior performance in terms of number of searches performed and PSNR, in comparison to the conventional search. The amount of improvements achieved from each combination of $\sigma_{BitSize}$, *r* and *stp* are quite similar to that obtained for 256 × 256 image size.

TABLE 1. Performance comparison between the dynamic and conventional search methods for $r = 4$, $\sigma_{BitSize} = 5$, $stp = 4$ and $thr = 700$.

Image dataset	Conventional search method		Dynamic search method		Improvement amounts	
	Searches no	PSNR (dB)	Searches no	PSNR (dB)	Searches no	PSNR (dB)
Lena	158356	25.60	67426	26.92	57 %	1.3
LiftingBody	285677	29.56	36409	30.90	87 %	1.3
CameraMan	768263	25.71	459747	26.31	40 %	0.6
LivingRoom	537562	23.78	313761	24.46	42 %	0.7
Goldhill	1026531	26.36	82582	27.87	92 %	1.5
Lake	519596	26.41	283893	26.52	45 %	0.1
Aerial	2661426	24.40	1479547	24.61	44 %	0.2
Elaine	99205	28.32	20312	29.94	80 %	1.6

The numerical results and improvement amounts attained by dynamic search approach for each image in Fig. 7 are shown in Table 1. From this table, it can see the effectiveness of the dynamic search approach in comparison to the conventional search approach.

In order to make the evaluation of the presented dynamic search scheme complete, it was necessary to compare its performance with other similar works. Hence, the presented fractal method has been compared in Table 2 with five other existing fractal methods in terms of speed, PSNR and compression ratio. In this table, it has tabulated the results from the presented method for two image sizes of Lena (i.e. 256 × 256 and 512 × 512). For these both image sizes,

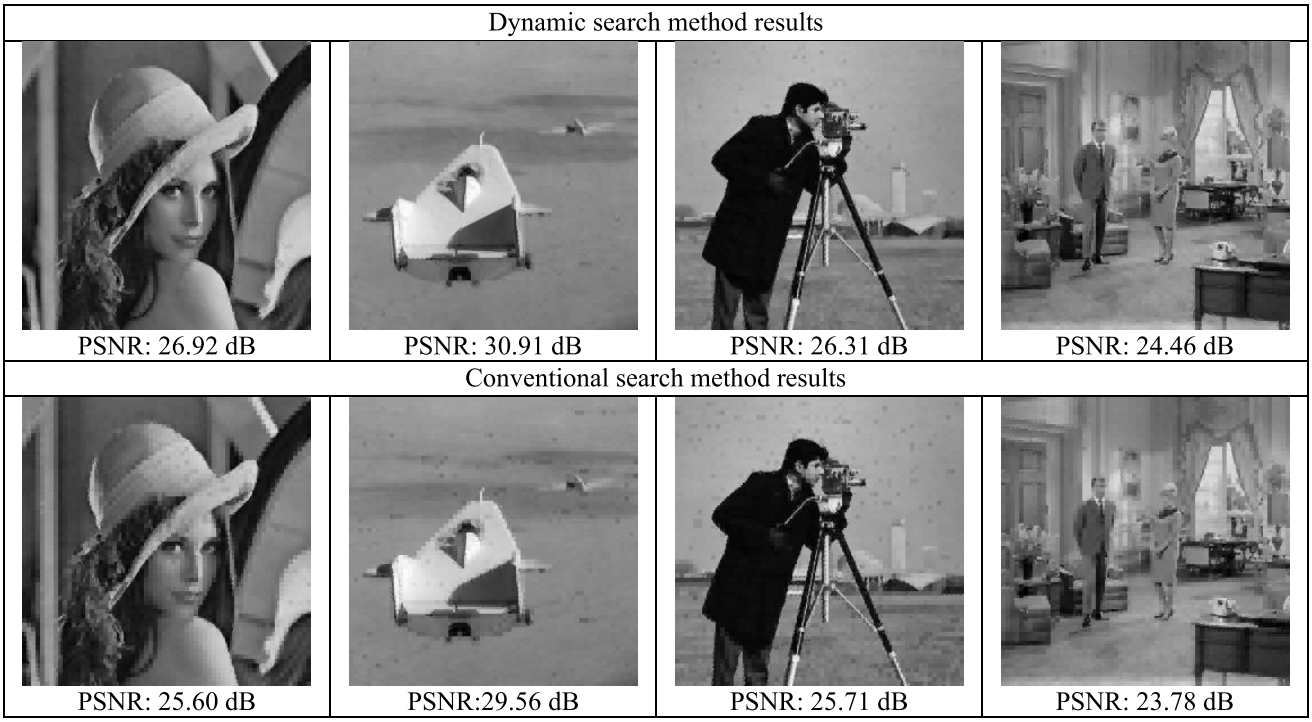

FIGURE 8. PFIC decompression results of four 256 × 256 tested images encoded by dynamic and conventional search methods.

TABLE 2. Performance comparison between the presented method and various existing fractal methods.

	Presented method		[35]	[18]	[10]	[34]	[33]
Implementation platform	Intel CPU i5 3.4 GHz		Intel (R) Core2 Duo CPU 2.0 GHz	Intel (R) Core2 Duo CPU 2.0 GHz	Pentium IV CPU 3GHz	Celeron 2.66 GHz	Intel CPU i5 2.5 GHz
Simulation tool	MATLAB		C++	MATLAB	C#	C++	MATLAB
Image name	Lena		Lena	Lena	Lena	Lena	Lena
Image size	256×256	512×512	256×256	256×256	512×512	256×256	256×256
Partitioning method	Fixed	Fixed	Fixed	Quadtree	Quadtree	Quadtree	Fixed
Range block size	4×4	8×8	8×8	-	-	-	4×4
Encoding time (Sec.)	0.3	0.9	6.4	56.4	2.5	1.9	4.2
PSNR	29.4	26.5	25.1	25.8	34.9	31.2	31.5
Compression ratio	4.9	21.3	27.1	15.6	5.6	11.5	5.3

the encoding parameters were defined as $thr = 700$, $stp = r$ and $\sigma_{BitSize} = 5$. In addition, for the image size 256 × 256, the range block size was 4×4 while for 512×512 was 8×8. As a result, it can evaluate the proposed method for low and high compression rates performance.

From Table 2, it is clearly that the presented method has the lowest encoding time compared to the others. The encoding time of the presented method is less than 1 s for both image sizes, then [34] 1.9 s, [10] 2.5 s, [33] 4.2 s, [35] 6.4 s and last [18] with 56.4 s. While in term of PSNR, the presented method comes at the intermediate level after [10] with 26.5 dB for 512×512 Lena image and [33], [34] with 29.4 dB for 256×256 image size, then [35] and [18], respectively. The increase in the PSNR value for [10] and [34] is attributed to the use of quadtree partitioning scheme with low matching error tolerance. However, the presented method is 14x faster than [33], 6x than [34] and 2.5x than [10] for a similar image size. Furthermore, the compression rate of the presented method is similar to [33], but 4x higher than [10]. Overall, it is very clear that the presented dynamic search scheme is significantly fast although it is simple.

Due to the simplicity and efficiency of the presented dynamic search method compared to the other existing complicated approaches, this method can be adopted in

developing hardware implementations for further decease in the processing time, without much overhead in implementation. Additionally, substitute the conventional full-search method used in [13], [28] by the presented method can increase the encoding speed by around 2 minimum (see Table 1). Thus, the planned future work is to utilise this dynamic search method in designing real-time image compression system on FPGA. Moreover, the presented method can be utilized in identical applications that require intensive self-similarity search of the image.

VI. CONCLUSION

This paper has studied and analyzed the impacts of dynamic and conventional static search mechanisms on the mapping speed and resultant image quality for the fractal image compression technique. The study has been conducted for various coding conditions and matching thresholding values. Unlike the conventional static search method, the dynamic search method is concerned with the high similarities between adjacent image blocks. Hence, for a given range block, the dynamic search method starts the search from the closest domain block and goes further until finding an accepted matched block or covering the entire image. On the other hand, the conventional static search method starts the search from a fixed location every time, regardless of the location of the encoded range block. By doing so, the experimental results showed that the dynamic search method outperforms the conventional search method in all aspects. However, the gap is increased significantly when the threshold value is defined larger, and differently based on the defined encoding parameters. For the best combination of the encoding parameters (i.e. $r = 4$, $\sigma_{BitSize} = 5$ and $stp = 4$), the improvement amount was the greatest and can reach over 90 % in terms of search reduction and 1.6 dB in terms of the quality of the decoded image. In conclusion, it can be said that there is a clear preference for the close image blocks over the far blocks when matching threshold strategy is implemented.

REFERENCES

[1] 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.
[2] M. Polvere and M. Nappi, "Speed-up in fractal image coding: Comparison of methods," *IEEE Trans. Image Process.*, vol. 9, no. 6, pp. 1002–1009, Jun. 2000.
[3] C. E. Martin and S. A. Curtis, "Fractal image compression," *J. Funct. Program.*, vol. 23, no. 6, pp. 629–657, 2013.
[4] S. Liu, W. Bai, N. Zeng, and S. Wang, "A fast fractal based compression for MRI images," *IEEE Access*, vol. 7, pp. 62412–62420, 2019.
[5] S. Kadam and V. R. Rathod, "Medical image compression using wavelet-based fractal quad tree combined with Huffman Coding," in *Proc. 3rd Int. Congr. Inf. Commun. Technol.*, 2019, pp. 929–936.
[6] S. Zhu and X. Zong, "Fractal lossy hyperspectral image coding algorithm based on prediction," *IEEE Access*, vol. 5, pp. 21250–21257, 2017.
[7] W. Pan, Y. Zou, and L. Ao, "A compression algorithm of hyperspectral remote sensing image based on 3-D wavelet transform and fractal," in *Proc. 3rd Int. Conf. Intell. Syst. Knowl. Eng.*, Nov. 2008, pp. 1237–1241.
[8] 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.
[9] T. N. Son, O. M. Hung, D. T. Xuan, T. Van Long, N. T. Dzung, and T. M. Hoang, "Implementation of fractal image compression on FPGA," in *Proc. 4th Int. Conf. Commun. Electron. (ICCE)*, Aug. 2012, pp. 339–344.
[10] T. Kovács, "A fast classification based method for fractal image encoding," *Image Vis. Comput.*, vol. 26, no. 8, pp. 1129–1136, Aug. 2008.
[11] H. Wang, "Fast image fractal compression with graph-based image segmentation algorithm," *Int. J. Graph.*, vol. 1, no. 1, pp. 19–28, 2010.
[12] X.-Y. Wang, Y.-X. Wang, and J.-J. Yun, "An improved no-search fractal image coding method based on a fitting plane," *Image Vis. Comput.*, vol. 28, no. 8, pp. 1303–1308, Aug. 2010.
[13] M. Panigrahy, I. Chakrabarti, and A. S. Dhar, "Low-delay parallel architecture for fractal image compression," *Circuits, Syst., Signal Process.*, vol. 35, no. 3, pp. 897–917, Mar. 2016.
[14] W. Xing-Yuan, L. Fan-Ping, and W. Shu-Guo, "Fractal image compression based on spatial correlation and hybrid genetic algorithm," *J. Vis. Commun. Image Represent.*, vol. 20, no. 8, pp. 505–510, Nov. 2009.
[15] M.-S. Wu, J.-H. Jeng, and J.-G. Hsieh, "Schema genetic algorithm for fractal image compression," *Eng. Appl. Artif. Intell.*, vol. 20, no. 4, pp. 531–538, Jun. 2007.
[16] X. Wang, D. Zhang, and X. Guo, "Novel hybrid fractal image encoding algorithm using standard deviation and DCT coefficients," *Nonlinear Dyn.*, vol. 73, nos. 1–2, pp. 347–355, Jul. 2013.
[17] D. J. Duh, J. H. Jeng, and S. Y. Chen, "DCT based simple classification scheme for fractal image compression," *Image Vis. Comput.*, vol. 23, no. 13, pp. 1115–1121, Nov. 2005.
[18] X.-Y. Wang and D.-D. Zhang, "Discrete wavelet transform-based simple range classification strategies for fractal image coding," *Nonlinear Dyn.*, vol. 75, no. 3, pp. 439–448, Feb. 2014.
[19] X. Wu, D. J. Jackson, and H.-C. Chen, "A fast fractal image encoding method based on intelligent search of standard deviation," *Comput. Electr. Eng.*, vol. 31, no. 6, pp. 402–421, Sep. 2005.
[20] R. Gupta, D. Mehrotra, and R. K. Tyagi, "Comparative analysis of edge-based fractal image compression using nearest neighbor technique in various frequency domains," *Alexandria Eng. J.*, vol. 57, no. 3, pp. 1525–1533, Sep. 2018.
[21] C. S. Tong and M. Wong, "Adaptive approximate nearest neighbor search for fractal image compression," *IEEE Trans. Image Process.*, vol. 11, no. 6, pp. 605–615, Jun. 2002.
[22] N. Rowshanbin, S. Samavi, and S. Shirani, "Acceleration of fractal image compression using characteristic vector classification," in *Proc. Can. Conf. Electr. Comput. Eng.*, 2006, pp. 2057–2060.
[23] R. Gupta, D. Mehrotra, and R. K. Tyagi, "Adaptive searchless fractal image compression in DCT domain," *Imag. Sci. J.*, vol. 64, no. 7, pp. 374–380, Oct. 2016.
[24] K. Chen and X. Wu, "A full quadtree searchless IFS fractal image encoding algorithm applicable in both high and low compression rates," in *Proc. ACMSE Conf.*, 2018, p. 18.
[25] X.-Y. Wang and S.-G. Wang, "An improved no-search fractal image coding method based on a modified gray-level transform," *Comput. Graph.*, vol. 32, no. 4, pp. 445–450, Aug. 2008.
[26] D. J. Jackson, H. Ren, X. Wu, and K. G. Ricks, "A hardware architecture for real-time image compression using a searchless fractal image coding method," *J. Real-Time Image Process.*, vol. 1, no. 3, pp. 225–237, Mar. 2007.
[27] Y. Fisher, *Fractal Image Compression: Theory and Application*. New York, NY, USA: Springer-Verlag, 1995.
[28] A.-M.-H. Y. Saad and M. Z. Abdullah, "High-speed implementation of fractal image compression in low cost FPGA," *Microprocessors Microsyst.*, vol. 47, pp. 429–440, Nov. 2016.
[29] X.-Y. Wang, Y.-X. Wang, and J.-J. Yun, "An improved fast fractal image compression using spatial texture correlation," *Chin. Phys. B*, vol. 20, no. 10, Oct. 2011, Art. no. 104202.
[30] K. Jaferzadeh, K. Kiani, and S. Mozaffari, "Acceleration of fractal image compression using fuzzy clustering and discrete-cosine-transform-based metric," *IET Image Process.*, vol. 6, no. 7, pp. 1024–1030, Oct. 2012.
[31] C. S. Tong and M. Pi, "Fast fractal image encoding based on adaptive search," *IEEE Trans. Image Process.*, vol. 10, no. 9, pp. 1269–1277, Sep. 2001.
[32] X.-Y. Wang, X. Guo, and D.-D. Zhang, "An effective fractal image compression algorithm based on plane fitting," *Chin. Phys. B*, vol. 21, no. 9, Sep. 2012, Art. no. 090507.

[33] 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.
[34] X.-Y. Wang, F.-P. Li, and Z.-F. Chen, "An improved fractal image coding method," *Fractals*, vol. 17, no. 4, pp. 451–457, Dec. 2009.
[35] W. Xing-Yuan, W. Na, and Z. Dou-Dou, "Fractal image coding algorithm using particle swarm optimisation and hybrid quadtree partition scheme," *IET Image Process.*, vol. 9, no. 2, pp. 153–161, Feb. 2015.
[36] Y.-G. Wu, M.-Z. Huang, and Y.-L. Wen, "Fractal image compression with variance and mean," in *Proc. Int. Conf. Multimedia Expo. (ICME)*, vol. 1, Jul. 2003, pp. I-353–I-353-6.
[37] X.-Y. Wang and Y. Lang, "A fast fractal encoding method based on fractal dimension," *Fractals*, vol. 17, no. 04, pp. 459–465, Dec. 2009.
[38] I. Aggarwal and R. Gupta, "Study of neighborhood search-based fractal image encoding," in *Proc. 5th Int. Conf. Soft Comput. Problem Solving*, 2016, pp. 93–104.
[39] D. Saupe, "Accelerating fractal image compression by multi-dimensional nearest neighbor search," in *Proc. Data Compress. Conf. (DCC)*, 1995, pp. 222–231.
[40] Y.-L. Lin and M.-S. Wu, "An edge property-based neighborhood region search strategy for fractal image compression," *Comput. Math. Appl.*, vol. 62, no. 1, pp. 310–318, Jul. 2011.
[41] Y.-L. Lin and W.-L. Chen, "Fast search strategies for fractal image compression," *J. Inf. Sci. Eng.*, vol. 28, pp. 17–30, Jan. 2012.
[42] R. Chaudhari and S. Dhok, "Review of fractal transform based image and video compression," *Int. J. Comput. Appl.*, vol. 57, pp. 23–32, Jan. 2012.

ABDUL-MALIK H. Y. SAAD (Member, IEEE) was born in Jeddah, Saudi Arabia, in 1983. He received the B.Sc. degree (Hons.) in computer engineering from Hodeidah University, Hodeidah, Yemen, in 2006, and the M.Sc. degree in electronic systems design engineering and the Ph.D. degree in digital systems from Universiti Sains Malaysia (USM), in 2014 and 2018, respectively. He is currently working as an Assistant Professor with the School of Electrical and Electronic Engineering, USM. His research interests include digital system design and image processing.

MOHD ZAID ABDULLAH received the B.App.Sc. degree in electronic from Universiti Sains Malaysia (USM), in 1986, and the M.Sc. degree in instrument design and application, and the Ph.D. degree in electrical impedance tomography from the University of Manchester Institute of Science and Technology, U.K., in 1989 and 1993, respectively. He was a Test Engineer with Hitachi Semi-Conductor, Malaysia. He is currently a Professor with the School of Electrical and Electronic Engineering, USM. He has published numerous research articles in international journals and conference proceedings. His research interests include microwave tomography, digital image processing, computer vision, and ultrawide band sensing. One of his articles was awarded the Senior Moulton Medal for the Best Article published by the Institute of Chemical Engineering, in 2002.

NAYEF ABDULWAHAB MOHAMMED ALDUAIS (Member, IEEE) received the B.Eng. degree in computer engineering from Hodeidah University, Yemen, in 2007, and the master's and Ph.D. degrees in communication and computer engineering from the Faculty of Electrical and Electronic Engineering (FKEE), Universiti Tun Hussein Onn Malaysia (UTHM), Malaysia, in 2015 and 2019, respectively. He worked as an Assistant Lecturer with the Faculty of Computer Science and Engineering, Hodeidah University, Yemen, from 2007 to 2013. He is currently a Lecturer and a Researcher of the Internet of Things (IoT) with the Faculty of Computer Science and Information Technology (FSKTM), UTHM. He has authored numerous articles in journals and conference proceedings. His research interests include WSN and the IoT. He received numerous medals and scientific excellence certificates.

HISHAM HAIDER YUSEF SA'AD received the bachelor's degree computer engineering from Hodeidah University, Yemen, in 2010, the M.Sc. degree in electronic system design engineering from Universiti Sains Malaysia (USM), Malaysia, in 2015, and the Ph.D. degree in computational intelligence from the School of Electrical and Electronic Engineering USM, in 2018. He is currently an Assistant Professor and the Head of the Computer Science Department, Al Razi University, Yemen. His current research interests include data mining and artificial intelligence (fuzzy systems). He received the top five awards for his bachelor's degree.