International Archive of Applied Sciences and Technology
Int. Arch. App. Sci. Technol; Vol 5 [4]December 2014: 11-25

[ISO9001: 2008 Certified Organization]
www.soeagra.com/iaast.html

IAAST
ONLINE ISSN 2277- 1565
PRINT ISSN 0976 - 4828

CODEN: IAASCA **RESEARCH ARTICLE**

A Study Of Iterated Transform Method With Special Reference To Image Compression

Anil Kumar Gupta and Meenakshi Choudhary***
*University Computer Centre, Institute of Basic Science, Khandari Campus, Dr. B R A University, Agra
**University Computer Centre, Institute of Basic Science, Khandari Campus, Dr. B R A University, Agra

ABSTRACT
An image compression scheme based on iterated transforms has been presented in this chapter. Results are examined as a function of several encoding parameters including maximum allowed scale factor, number of domains, and resolution of scale and offset values, minimum range size, and target fidelity. The performance of the algorithm, evaluated by means of fidelity versus the amount of compression, is compared with an adaptive discrete cosine transform image compression method over a wide range of compressions.
Keywords: *Image compression, fractals, iterated transformations.*

Received 02/11/2014 **Revised 02/12/2014** **Accepted 13/12/2014**

Citation of this article
Meenakshi Choudhary and Anil Kumar Gupta. A Study Of Iterated Transform Method With Special Reference To Image Compression. Int. Arch. App. Sci. Technol; Vol 5 [4] December 2014: 01-04. DOI.10.15515/iaast.0976-4828.5.4.11-25

INTRODUCTION

Because of the increasing use of digital imagery, there is currently considerable interest in the image compression problem. This interest has led to the establishment by the Joint Photographic Experts Group of an image compression standard based on discrete cosine transforms. Although the use of this standard is becoming common, there are alternative methods for compressing images. One such alternative, iterated transformations, has been presented by Jacquin [6-8]. This method has its foundation in the theory of iterated function systems (IFSs), developed by Hutchinson [5] and Barnsley [1], and recurrent iterated function systems (RIFS) [2] For a description of the basic iterated transform method, refer to [6-8] This method has been extended to include individual transforms which are not contractive [4] Because this is a relatively new method, little information is currently available on its performance. In writing even a simple iterated transform encoder, there are a host of parameters which can be varied. Currently, there is no information available on the dependence of system performance on such parameters. The purpose of this paper is to present results on the dependence of compression, fidelity and encoding time, on several pertinent system parameters. For the purpose of defining the parameters of interest in this paper, a brief summary of the method is presented in the following paragraphs. A more detailed description of the method in the notation used in this correspondence, and a description of the basic implementation used here can be found in [4]. Moustafa and Alsayyh [10] suggested a new hybrid image compression technique. Three transform-based techniques discrete Fourier transform (DFT), discrete wavelet transform (DWT), and discrete cosine transform (DCT) have been combined for image compression to confer the good char- acteristics of these methods. The proposed method works for forfeiture compression and the attained results show that it works well compared to existing approaches. To test the level of compression, the quantitative measures of the peak signal-to-noise ratio (PSNR) & mean squared error (MSE) are used to ensure the effectiveness of the suggested system. In domain of image processing; image compression is highly demanded due to the fast advancement of communications, computer, and internet technology [11-14].

THEORETICAL BACKGROUND

The image is encoded in the form of an iterative system (a space and a map from the space to itself) $W : F \to F$. The space F is a complete metric space of images, and the mapping W (or some iterate of W) is a contraction. The contractive mapping fixed point theorem ensures convergence to a fixed

point upon iteration of W. The goal is to construct the mapping W with fixed point 'close' (based on a properly chosen metric $\delta(f,g)$) to a given image that is to be encoded, and such that W can be stored compactly. The collage theorem provides motivation that a good mapping can be found [1]. Decoding then consists of iterating the mapping W from any initial image until the iterates converge to the fixed point.

Let $I=[0,1]$ and I^m be the m-fold Cartesian product of I with itself. Let F be the space consisting of all graphs of real Lebesgue measurable functions $z=f(x,y)$ with $(x,y,f(x,y))\in I^3$. Let $D_1,D_2,D_3,...D_n$ and $R_1,R_2,R_3,...,R_n$ be subsets of I^2 and $v_1,v_2,v_3,...,v_n : I^3 \to I^3$ be some collection of maps.

Define $w_i = v_i\big|_{D_i\times I}$

The maps $w_1,w_2,...,w_n$ are said to tile I^2 if for all $f\in F$, $\cup_{i=1}^n w_i(f)\in F$

This means the following: for any image $f\in F$, each D_i defines a part of image $f\cap(D_i\times I)$ to which w_i is restricted. When w_i is applied to this part, the result must be a graph of this function over R_i, and $I^2=\cup_{i=1}^n R_i$. This is illustrated in Fig. 1. This means that the union $\cup_{i=1}^n w_i(f)$ yields a graph of a function over I^2, and the set $R_i's$ are disjoint. The map W is defined as

$$W=\cup_{i=1}^n R_i \qquad \text{... (1)}$$

Since the goal is to limit the memory required to specify W, I^2 is partitioned by geometrically simple sets R_i with $I^2=\cup_{i=1}^n R_i$. For each R_i, $D_i\subset I^2$ and $w_i : D_i\times I\to I^3$ is sought such that $w_i(f)$ is as δ close to

$$f\cap(R_i\times I), w_i(f) \qquad \text{...(2)}$$

is minimized. The motivation for minimizing expression (2) is provided by the collage theorem [1]. Those initiated to IFS theory may find it surprising that when the transformations w_i are constructed, it is not necessary to impose any contractivity conditions on the individual transforms. The necessary contractivity requirement is that W be eventually contractive [4]. A map $W : F\to F$ is eventually contractive if there exists a positive integer m such that the m^{th} iterate of $W(W^{0m})$ W is contractive (as measured by an appropriate metric). All contractive maps are eventually contractive, but not vice versa. A brief explanation of how a transformation $W : F\to F$ can be eventually contractive but not contractive is in order. The map W is composed of a union of maps w_i operating on disjoint parts of an image. If any of the W, are not contractive, then W will also not be contractive. The iterated transform W^{0m} is composed of a union of compositions of the form $w_{i1}\circ w_{i2}\circ ...w_{im}$ Since the product of the contractivities bounds the contractivity of the compositions, the compositions may be contractive if each contains sufficiently contractive w_{ij}. Thus W will be eventually contractive if it contains sufficient 'mixing' so that the contractive w, eventually dominate the expansive ones.

Implementation-

For 256×256 pixel 8 bit per pixel (bpp) images the model was scaled to $[0,255]\times[0,0255]\times[0,255]$.

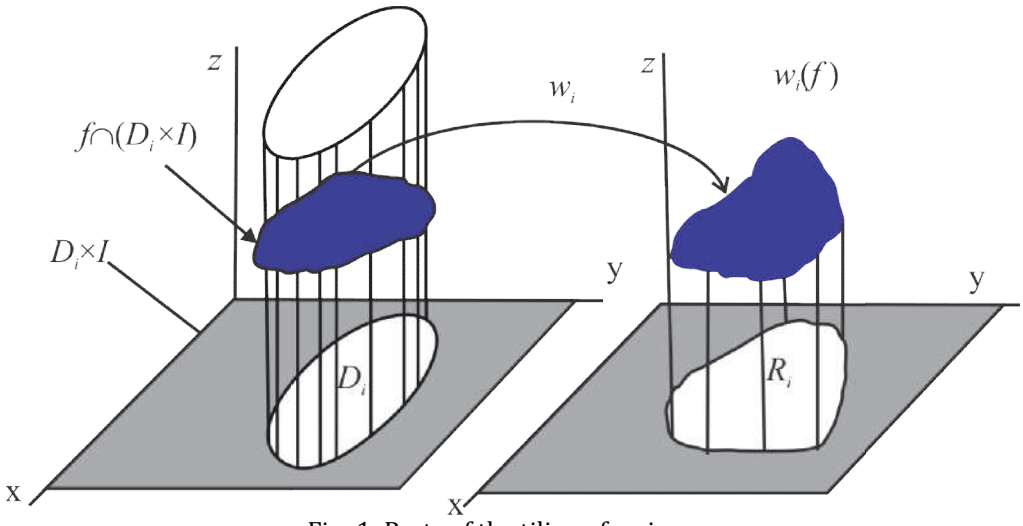

Fig.-1: Parts of the tiling of an image

For other image sizes, appropriate scaling can be employed). To limit the memory required to specify w_i, only maps of the

$$w_i\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}a_i & b_i & 0\\c_i & d_i & 0\\0 & 0 & s_i\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}+\begin{bmatrix}e_i\\f_i\\o_i\end{bmatrix} \quad \text{... (3)}$$

form are considered, where W, is restricted to $D_i\times I$ In terms of an image, x and y represent the pixel coordinates, and z represents the pixel intensity. The pixel intensities are clipped when W, maps outside the allowed intensity range. For the transformation w_i to be compactly specified, R, and D_i must be compactly specified. This can be done by choosing R_i and D_i from a small set of potential candidates. Also, many of the computations are simplified if R_i and D_i are geometrically simple. Let R be the collection of subsets of I^2 from which the R, are chosen, and let D be the collection of subsets of I^2 from which the D_i are chosen. The set R was chosen to consist of $4\times4,8\times8,16\times16$ and 32×32 no overlapping sub squares of $[0,255]\times[0,0255]$ (i.e, squares of size $s\times s$ have their upper left corners at integer multiples of s). The collection D consisted of $8\times8,16\times16,32\times32$ and 64×64 subsquares with sides parallel to or slanted at 45^0 angles from the natural edges of the image. Although the $D_i's$ and $R_i's$ are not strictly the domains and ranges of the $w_i's$. The terminology will be used because it is descriptive. Domain squares of size $s\times s$ were restricted to have corners on a lattice with vertical and horizontal spacing of $s/2$, This choice of D will be called D'. It is clear then that the size and position of R, the size, position and orientation (i.e. 0^0 or 45^0) of D_i, and which one of the eight possible symmetries (rotation and flip operator v_i) for mapping one square onto another, uniquely define the coefficients a_i,b_i,c_i,d_i,e_i and f_i in (3). In this implementation, only domains with side length twice that of the range are allowed, resulting in contraction in the xy plane. Therefore, each range pixel corresponds to a two by two pixel area in the corresponding domain. The average of the four domain pixel intensities are mapped to the area of the range pixel when computing $w_i(f)$. When D_i is oriented at 45°, the averaging of pixels is more complicated, the details of which are not very significant. What is significant is that the method for averaging pixels in the encoding and decoding procedures is consistent. Insisting that w_i, map (the graph above) D, to (a graph above) R,

while minimizing expression (ii) determines s_i and o_i. In this way w_i is determined uniquely for a chosen metric. In this chapter, the root mean square (rms) error δ_{rms} was chosen as the metric. For further comment on this choice of the metric, see [4]. Because s_i and o_i must be stored with a fixed number of bits (n_s and n_r respectively), they must be discretized. The values for s_i were restricted to the range $s_{\max} \geq |s_i| \geq s_{\max/10}$ and $s_i = 0$. Where $s_{\max}$ was the maximum allowable s_i. The distribution of the discretized s_i's was chosen such that any desired scale factor could be represented within some fixed percentage (i.e. a logarithmic scale). The minimum and maximum possible values for o; are restricted by the value of the corresponding s_i. Given n_0 the discretized values for o_i were distributed linearly over this interval. Only the discretized values of s_i and o_i are used when calculating the values of s_i and o_i which minimizes expression (ii). Once the choice of R and D has been made, the encoding problem is reduced to choosing a set $\{R_i\} \subset R$ and the corresponding set $\{D_i\} \subset D$ such that good compression and an accurate encoding of the image results. To take advantage of local `flatness' in the image and to reduce the error in regions of high variability, a recursive quadtree partitioning method was used to allow the range squares to vary in size. The method used to find the D_i's determines how much computation time the encoding takes. A search through all of D would clearly result in the choice that would best minimize expression (ii), but for applications for which encoding time is a consideration; such a search may require too much computation time. Therefore a classification scheme was used to reduce the amount of computation needed to find a good covering. The strategy of the classification scheme is important, although the fine details of the implementation method are not critical. The following paragraph reviews the important aspects of the classification scheme used. The classification scheme used was based on the simple ideas that good covers would have matching regions of bright and dark and that any strong edges (variations in intensity) should also match. The classification scheme in [6] was generally based on these ideas. The classification scheme began by computing the average intensity for each quadrant of the (range or domain) square. Then a symmetry operation was applied to force the square into an orientation with its brightest quadrant in the upper left quadrant, and to put the second brightest quadrant into the upper right quadrant (or the third brightest if the second brightest could not be so oriented). This process divided the squares into three main classes (and defined a symmetry operation for each square). Each of these three main classes was then subdivided by determining the amount of the variation of the average brightness of the sub-quadrants for each quadrant of the square. The quadrants of the square were thereby ordered from first to fourth by the amount of variation within each quadrant. There are 24 possible permutations for the order of the relative brightness variations. This results in a total number of 72 classes. The symmetry operations determined in this classification for a given R_i and D_i defined the rotation and flip operation v_i for mapping D_i to R_i further increasing the time saved during encoding. The encoding process proceeded as follows. Initially, the range squares R; were chosen to be 64 subsquares of size 32 x 32. The first 32 x 32 range square was classified using the same classification procedure as the domains. A search was then performed for the domain square (with side length twice that of the range square) in the same class (or similar classes) which best minimized expression (ii). If this best domain square and its corresponding w resulted in an error (given by expression (ii)) less than a predetermined tolerance e_c, w (and D) was stored and the process was repeated for the next range square. If the predetermined tolerance was not satisfied, the range square was subdivided into four equal squares. This quadtreeing process was repeated until the tolerance condition was satisfied, or a range square of a predetermined minimum size $r_{\min}$ was reached. For range squares of size $r_{\min}$, the best w was stored whether or not e_c was satisfied. The process was continued until the entire image was encoded. The average storage requirement for a single w was about 30 bits, and was dependent on several factors. The rotation and flip operator v, required 3 bits, and for most encodings presented here n, and n o were stored with a total of 12 bits. The position

Of R_i was inferred from the ordering. Approximately 15 bits were required to identify the size of R_i and the location (and orientation) of D_i, this number being dependent on both the choice of D and the level in the quadtree. When $s_i = 0$ only o_i and the size of R_i are required, so the transformation can be stored more compactly. The compressions quoted in this paper were computed from the actual size of the compressed data files.

Clearly this encoding method has many parameters which influence the compression, accuracy and speed of the encoding. How these parameters effect the encoding is not a priori obvious. The following discussion will consider how varying the number of possible domains effects the encoding accuracy, compression and speed. The following assumes that one encoding, call it W', uses a given set of possible domains D', while the other W'' uses a domain set D'' such that $D' \subset D''$. First, it is clear that for any given range square, the best covering (in the sense of minimizing expression (ii)) from D'' must be as good as or better than the best covering taken from D'. However, this does not guarantee an improved encoding within the quad tree method. It is possible that some large range which was subdivided in W' (and then covered very well) will not be subdivided in W'', resulting in a less accurate encoding. Second, since D'' contains D', the individual w_i transforms of W'' require more bits to store than the $w_i's$ of W'. However, this does not mean that W'' must have a poorer compression than W'. If a large block is covered, rather than subdivided, then only one transform (instead of at least four) needs to be stored, resulting in a net savings in the total number of required bits. Finally, although the construction of W'' requires searching through a bigger domain pool than the construction of W', it does not follow that the encoding process must take more time. When a large block is covered the four smaller ranges do not need to be covered, thus saving time. This paper describes the dependence of the performance of the encoding scheme on the following parameters : (1) the number of bits used to store the scale factor (n_s) and offset (n_o), (2) the maximum allowed s_i ($s_{\max}$)s, (3) the number of possible domains, (4) the criterion used to decide if a covering is acceptable (e_c) (5) the minimum range size in the quadtree subdivision ($r_{\min}$), and (6) the number of domain classes searched (n_c). It is not obvious how adjusting these parameters effects the performance of the encoding method, since for each of these parameters it is possible to construct arguments similar to that given previously for the total number of possible domains.

Before presenting results, a tutorial example is given to better illustrate how the method works. Let values of $z = 0$ be represented as black, $z = 1$ as white, with intermediate values as shades of gray. Consider the image in Fig. 2, and the sixteen transformations given in Table 1, where the first eight transformations are restricted to act on the region $\left\{(x,y) : 0 \le x \le \frac{1}{2}, 0 \le y \le \frac{1}{2}\right\}$ and the second eight transformations are restricted to act on the region $\left\{(x,y) : \frac{1}{2} \le x \le 1, 0 \le y \le \frac{1}{2}\right\}$. The map W is defined as the union of these 16 $w_i's$ and encodes the image shown. This can be easily demonstrated. The staring point of the iteration is arbitrarily chosen as $z = 0.5$ for $\{(x,y) : 0 \le x \le 1, 0 \le y \le 1\}$.

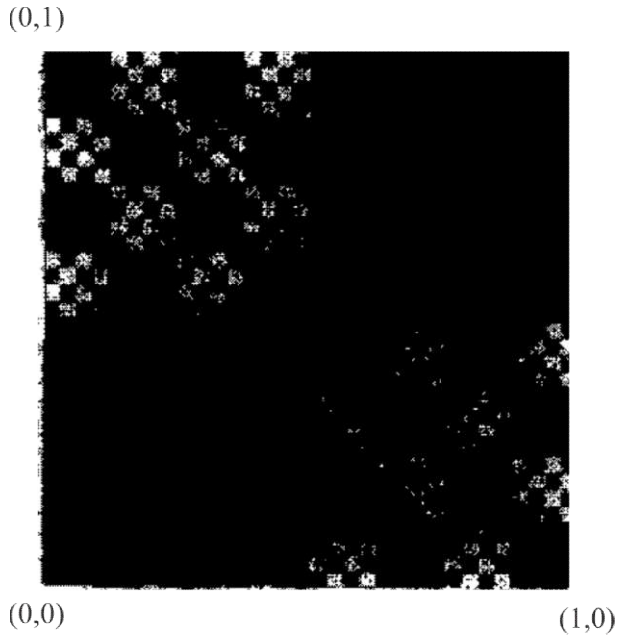

Fig.-2: A gray scale image

Table-1: A set of 16 transformations that encode the image in Fig.2

a	b	c	d	s	e	f	o
0.5	0.0	0.0	0.5	2.0	0.75	0.0	0.0
0.5	0.0	0.0	0.5	2.0	0.75	0.25	0.0
0.5	0.0	0.0	0.5	2.0	0.50	0.0	0.0
0.5	0.0	0.0	0.5	2.0	0.50	0.25	0.0
0.5	0.0	0.0	0.5	2.0	0.0	0.5	0.0
0.5	0.0	0.0	0.5	2.0	0.0	0.75	0.0
0.5	0.0	0.0	0.5	2.0	0.25	0.5	0.0
0.5	0.0	0.0	0.5	2.0	0.25	0.75	0.0
0.0	-0.5	0.5	0.0	0.25	0.25	-0.25	0.0
0.0	-0.5	0.5	0.0	0.25	0.25	0.0	0.25
0.0	-0.5	0.5	0.0	0.25	0.5	-0.25	0.25
0.0	-0.5	0.5	0.0	0.25	0.5	0.0	0.0
0.0	-0.5	0.5	0.0	0.25	0.75	0.25	0.0
0.0	-0.5	0.5	0.0	0.25	0.75	0.5	0.25
0.0	-0.5	0.5	0.0	0.25	1.0	0.25	0.25
0.0	-0.5	0.5	0.0	0.25	1.0	0.5	0.0

The first six iterates, and the fixed point is shown in Fig. 3. In practice, the values x, y and z are discretized. When the image in this example is discretized as 128×128 pixels, and 8 bits per pixel, the encoder described in this chapter automatically encodes this image (using an equivalent set of 16 transformations) with the resulting compression equal to 356:1. Note that $s_{\max} = 2.0$ sma for this encoding. The encoder, when constrained to have $s_i < 1$, requires 520 transformations to encode this image, with a resulting compression of less than 10: 1.

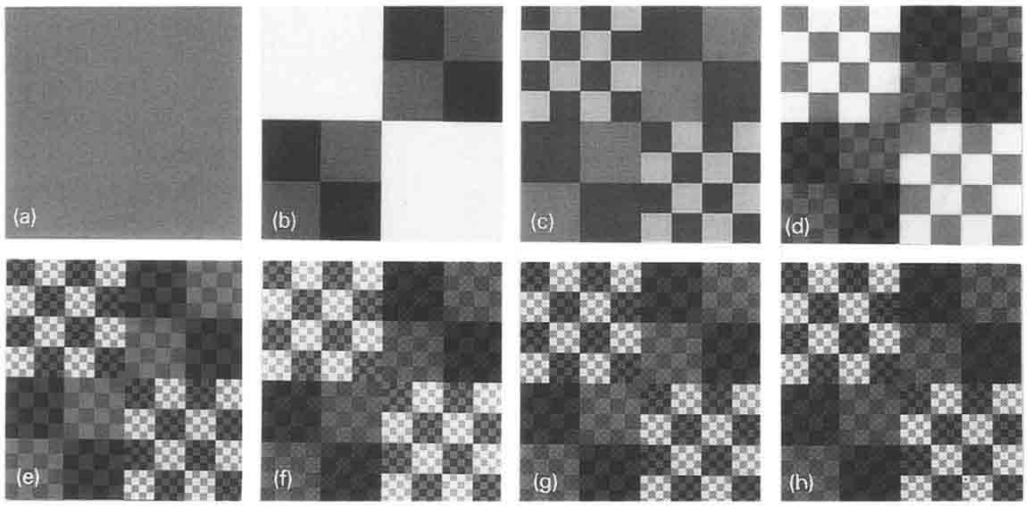

Fig.-3: The initial image, the first six iterations and the fixed point for the map composed of the 16 transformations in Table-1

RESULTS

The peak-signal-to-noise ratio (PSNR) was used to determine image fidelity. PSNR is defined as

$$PSNR = -20\log_{10}\left(\frac{rms}{2^n - 1}\right)$$

where rms is the root mean square error of the reconstructed image and n is the number of bits per pixel of the image.

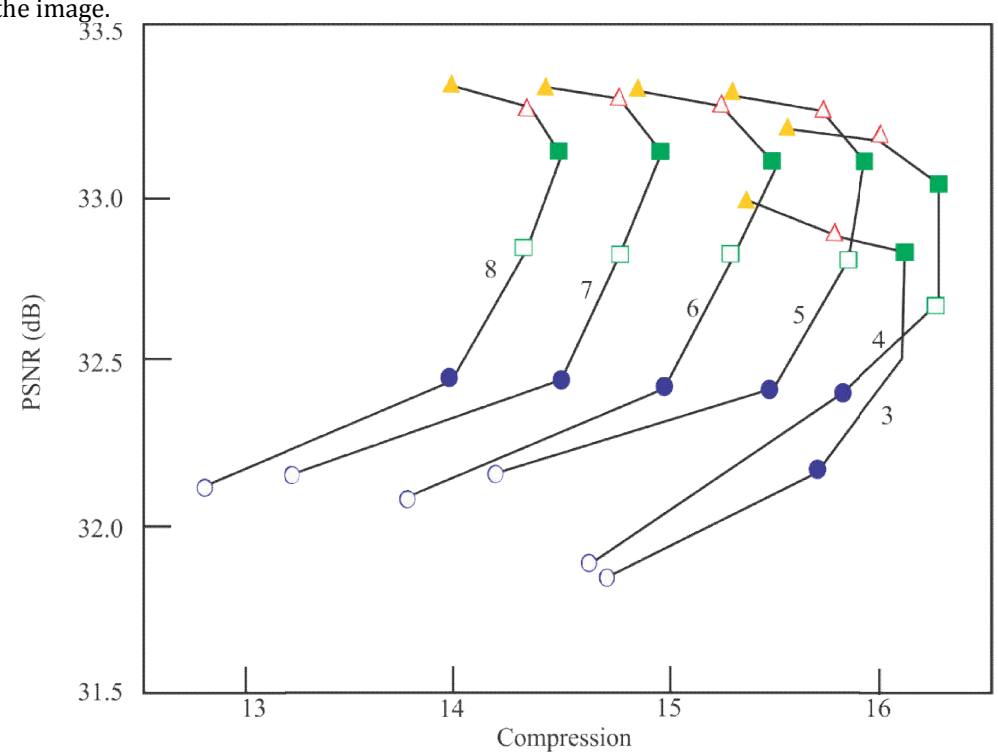

Fig. 4. PSNR versus compression as a function of n_s, and n_0 for 512×512 Lena. The values of n_0=3, 4, 5,6, 7 and 8 are denoted by ○, ●, □, ■, △ and ▲, respectively. The values of n_s are noted on the plot, the lines connecting points with common n_s. Other parameters were D' and (r_{min}, s_{max}, e_c, n_c) equal to (4, 1.0, 8.0, 4), respectively.

Figure 4 shows PSNR versus compression as a function of n_s and n_o the number of bits used to store values for S and O respectively. This data results from encodings of the 512×512 pixel resolution, 8-bpp image of Lena. For these encodings, the full set D' as described in Section 1 was used. The other

parameters used for these encodings were: minimum range size of 4×4 pixels ($r_{\min} = 4$), maximum allowed scale factor set to 1.0 ($s_{\max} = 1.0$), root mean square error tolerance (e_c) set to 8.0, and number of classes searched equal to 4 ($n_c = 4$). Results for n_s and n_o equal to 3, 4, 5, 6, 7 and 8 are shown. The curves connect the data points with the same value of n_s. The different symbols indicate the value of n_o. By connecting the similar symbols, one can visualize the curves of constant n_o.

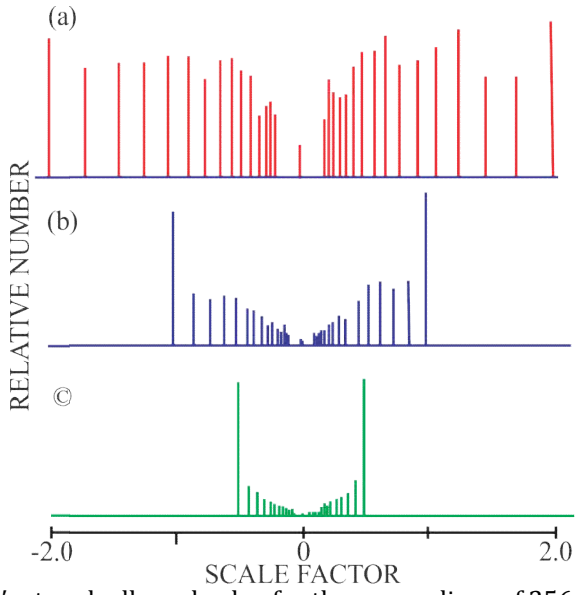

Fig. 5. Relative number of s_i's at each allowed value for three encodings of 256×256 Lena. The value of s_{max} for a, b and c were 2.0, 1.0 and 0.5, respectively. Other parameters were n_s= 5, n_0 = 7, with other parameters the same as Fig. 4.

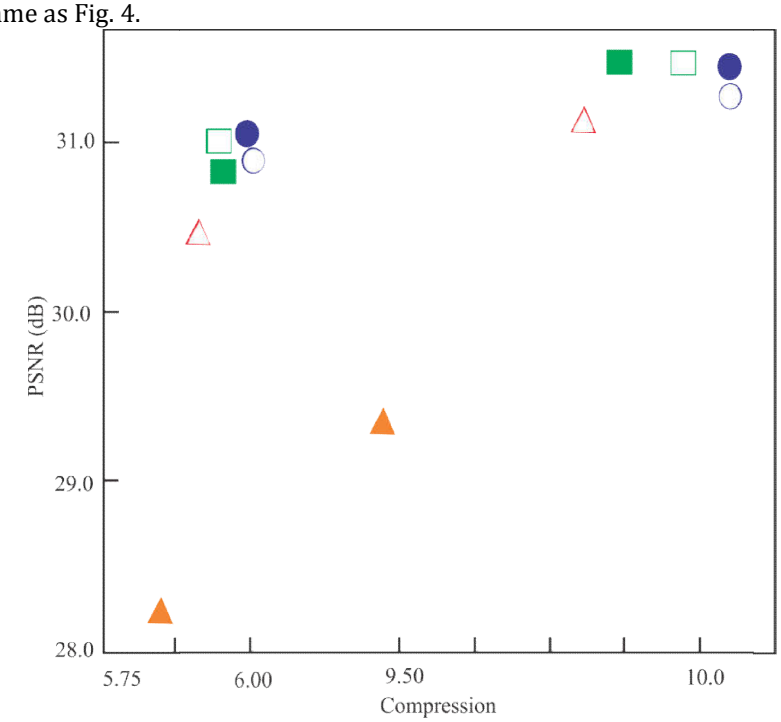

Fig. 6. PSNR versus compression as a function of s_{max}. The curve at lower compression is for 256×256 'tank farm', the curve at higher compression for 256 ×256 Lena. The values of s_{max}=0.5, 0.8, 1.0, 1.2, 1.5 and 2.0 are denoted by ○, ●, □, ■, △ and ▲ respectively. Other parameters were n_c=72, with others the same as Fig. 5.

The lack of contractivity condition on s_i is reflected in Fig.5 and 6 where data form encodings with $s_{\max}$ as high as 2.0 are presented. Fig. 5 shows the distribution of the scale factors used in three different encodings of 256 x 256 pixel resolution, 8-bpp Lena. The values of $s_{\max}$ for these encodings were 2.0, 1.0 and 0.5 in Figs. 5(a), 5(b) and 5(c), respectively. Other parameters were $N_s = 5, n_o = 7$ with all other parameters the same as in Fig. 4. In Fig. 6 PSNR versus compression is plotted for different values of $s_{\max}$. The curve at lower compression in Fig. 6 is for the 256×256 pixel resolution, 8 bit per pixel image of `tank farm' shown in Fig. 7(a). The decoded image for $s_{\max} = 1.2$ is shown in Fig. 7(b). The curve at higher compression is for 256×256 Lena.

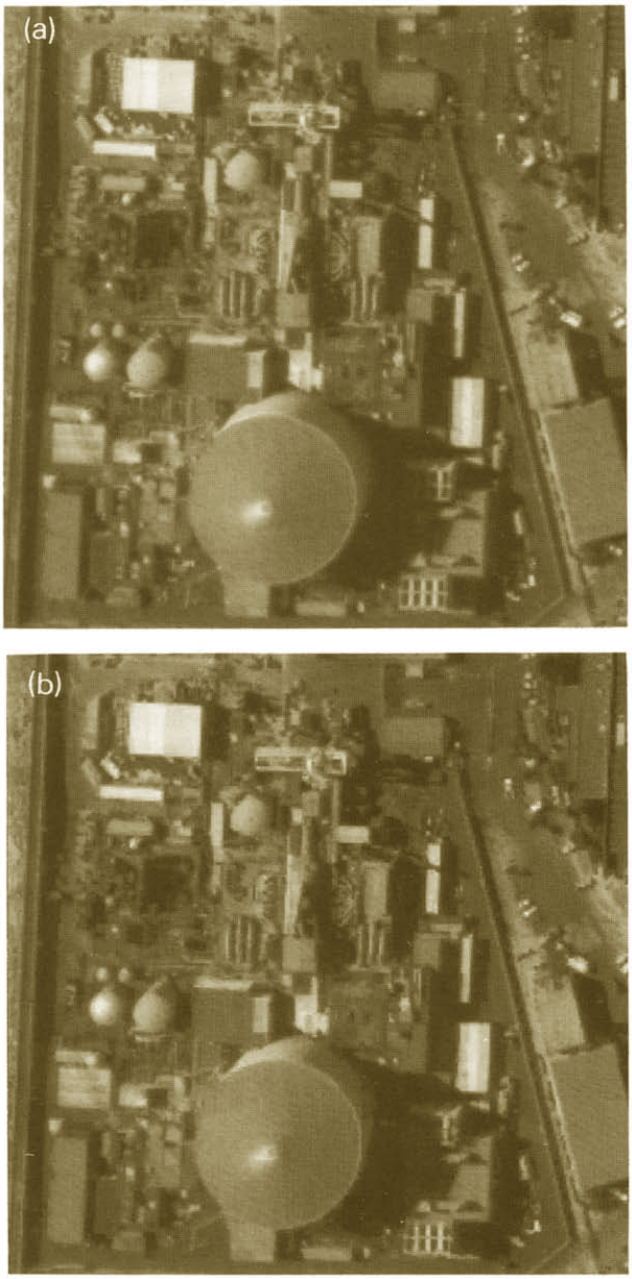

Fig. 7. (a) 256 x 256 8 bit/pixel original image of 'tank farm'. 7(b) Decoded image with s_{max}= 1.2. For this image the compression is 5.96:1 and the PSNR is 30.98 dB.

Figure 8 shows PSNR versus compression as a function of the number of domains for 512×512 Lena. All the results in Fig. 8 used the parameters $s_{\max} = 1.0, r_{\min} = 4, n_s = 5, n_o = 7, e_c = 8.0$. The set

$D^{1/4}$, with number of domains reduced by (approximately) a factor of four from the set D^1, was obtained by restricting the domains to have corners on a lattice with vertical and horizontal spacing of s (as opposed to $s/2$).

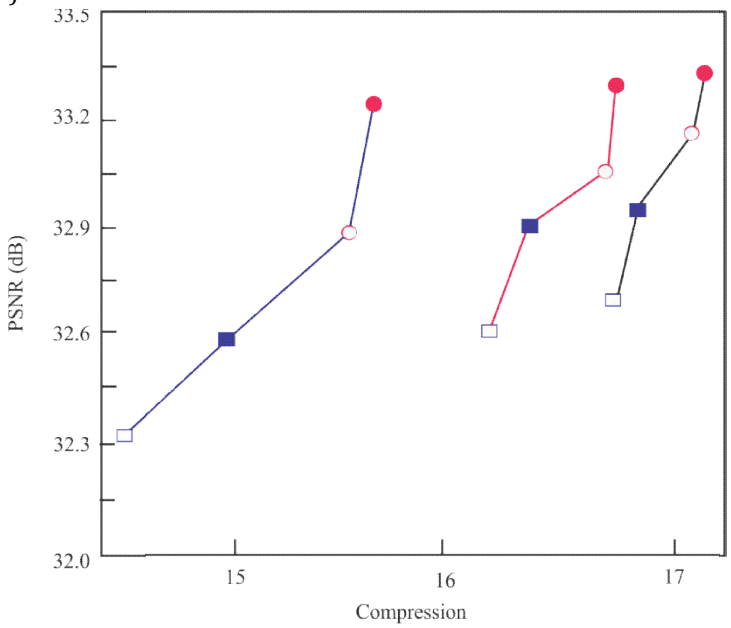

Fig. 8. PSNR versus compression as a function of the number of domains for encodings of 512 x 512 Lena. The sets D^1, $D^{1/2}$ and $D^1/4$ and $D^{1/8}$ are denoted by ●, ○, ■ and □, respectively. Data for n_c = 1, 4 and 12 appear as the solid, dashed and dotted curves, respectively. Other parameters were s_{max} = 1.0 with other parameters the same as in Fig. 5.

The set $D^{1/2}$, with number of domains reduced by a factor of two from the set D^1, was obtained by eliminating the domains with sides slanted at 45^0 angles from the natural edges of the image. The set $D^{1/8}$ was obtained by both restricting the lattice spacing and eliminating the diagonal domains. As mentioned in section 1, the domains were classified into 72 classes, and only a predetermined number of classes (n_c) were searched during encoding. Therefore, increasing n_c effectively increased the number of possible domains. Figure 8 shows data $n_c = 1$ (connected by solid line), 4 (dashed) and 12 (dotted) for each of the domain sets. The open symbols are for cases in which no diagonal domains are allowed ($D^{1/2}$ and $D^{1/8}$), while the solid symbols include diagonal domains (D^1 and $D^{1/4}$). The circles indicate domain lattice spacing of $s/2$ (D^1 and $D^{1/2}$), while the squares indicate lattice spacing of s ($D^{1/4}$ and $D^{1/8}$). Finally, Fig. 9 shows PSNR versus compression data of 512 x 512 Lena for $e_c = 8.0, 5.0, 11.0$. The open symbols represent data for $r_{\min} = 4$. Data is shown for n_c =1 (○), 4 (◇) and 72 (△).

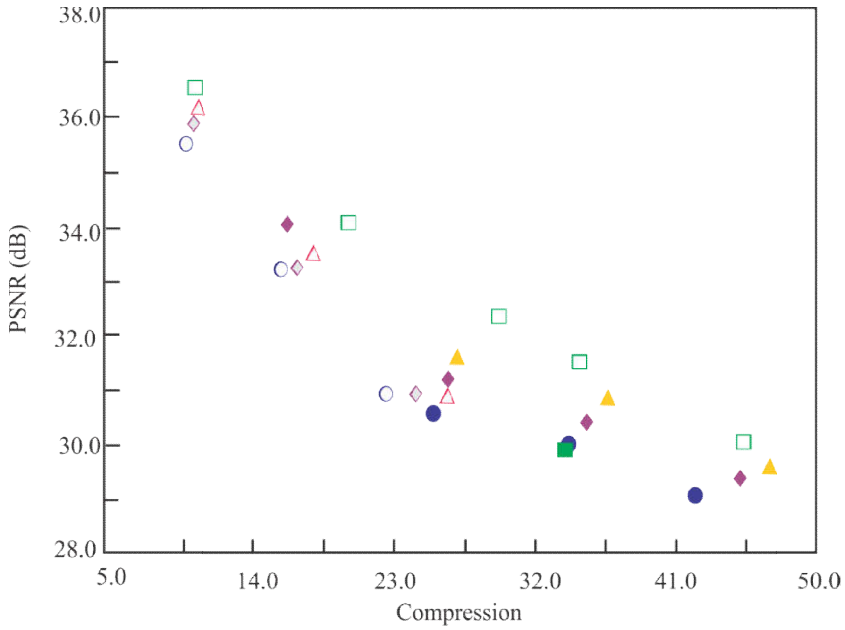

Fig. 9. PSNR versus compression as a function of e_c, r_{min} and n_c for 512×512 Lena. Open symbols represent r_{min}=4 and solid symbols r_{min} = 8. Data for e_c = 5.0, 8.0 and 11.0 appear in clusters with increasing compression. Within each cluster, n_c = 1 (○), 4 (◇) and 72 (△). ADCT data (□) is shown for comparison.

The value of e_c, separates the data into three widely spaced clusters, $e_c = 5.0$ for the cluster with the highest PSNR, and $e_c = 11.0$ for the cluster with the lowest PSNR. The solid symbols represent the same set of data, but with $r_{\min} = 8$. Other parameters were set as follows: $s_{\max} = 1.2, n_s = 5, n_o = 7$. The original image of Lena and the decoded image with $r_{\min}$, e_c and n_c equal to (8, 8, 72), respectively, are shown in Fig. 10. The set of domains D^1 was used for these encodings. For the purpose of comparison, data obtained from an adaptive discrete cosine transform (ADCT) method is also shown. The ADCT method used was similar to that described in [3].

Fig. 10. (a) 512×512 8 bit/pixel original image of Lena, (b) Decoded image with e_c = 8, r_{min}=8 and n_c=72. For this image the compression is 36.78:1 and the PSNR is 30.71 dB.

DISCUSSION

In order to compare the relative merits of various sets of parameters it is necessary to be able to decide what it means to say that one encoding is `better' than another. For images encoded using the method described in this correspondence, it has been observed that the PSNR is a reasonable measure of the visual quality of the images (i.e. given two encodings, the one with the larger PSNR looks better). It is clear that the best possible encoding would have both maximum compression and maximum fidelity. In practice this is usually not possible, since adjusting a parameter to improve the compression almost always results in degradation in the fidelity. Consequently it is difficult to compare two encodings if one is more accurate with poorer compression. The following observation suggests how such a comparison can be made. Varying the target fidelity e_c for a particular set of encoding parameters moves an encoding along an empirical barrier by trading compression for accuracy in a roughly linear way (see Fig. 9). An encoding resulting from a different choice of parameters is better if it has a higher compression than the barrier for the same fidelity or better fidelity for the same compression. It is not practical to compute a barrier for each encoding, so that the better encoding is chosen using an estimate of the barrier. In the results presented below, this is what is meant when one combination of parameters is described as superior to another. An inclusion of the encoding time further complicates the issue and will be neglected, except where specifically indicated. Figure 9 illustrates these points. Note that in the three data points with $r_{\min} = 8$ and n_c increasing e_c resulted in a marked improvement in compression with a corresponding decrease in PSNR. The curve obtained by connecting these data points represents the empirical barrier. By increasing n_c to 4 and 72, both the compression and the PSNR were improved, resulting in an improved performance barrier. This improvement was achieved at the cost of increased encoding time. The set of data with $r_{\min} = 4$ shows similar behavior with some notable differences. The slopes of the empirical barriers for the $r_{\min} = 4$ data are steeper than those of the data with $r_{\min} = 8$. The data with $e_c = 11.0$ indicates that increasing ne resulted in an increase in compression and a decrease in PSNR. Therefore it is not obvious that increasing ne improved the encoding. In light of the discussion in the previous paragraph, the slope of the empirical performance barriers for the $r_{\min} = 4$ data is steep enough that one can conclude that increasing ne did result in better encodings (i.e. the improvement in compression more that compensated for the decrease in fidelity). In Fig. 4 a grid of data

is presented with different combinations of n_s and n_o .The data is plotted in this way so that one can choose the `best' combination for n_s and n_o By comparing these data with slope of the barrier in Fig 9 (for $r_{\min} = 4$) one can see that the combination n,.=5, n o =7 is the best. Extensive comparisons (a few hundred encodings of several images at various values of e_c, and $r_{\min}$) has shown that plots equivalent to Fig. 4 do not all yield the same best combination. Nonetheless the combination $n_s = 5$, $n_o = 7$ is the best (compromise) choice for the entire set. Consequently, all other results are given for encodings using $n_s = 5$, $n_o = 7$. The data in Fig. 5 show the relative number of each scale factor used to encode Lena (encodings of other images resulted in similar data). Concentrating on Fig. 5(b) ($s_{\max} = 1.0$), because the larger scale factors seem to be preferred; one might hypothesize that a distribution of $s's$ with more large values and less small values might yield improved encodings. Experiments with both linear and inverse logarithmic distributions for allowed values of s resulted in no improvement to the results presented here. In addition, the minimum allowed nonzero s can be changed. Initial data indicates that the values used were good. In Fig 5, note that for smaller $s_{\max}$ a disproportionate number of $s_i's$ at the extremes of the allowed range of s are used. Therefore one might hypothesize that increasing $s_{\max}$ will result in improved encodings. In Fig. 6, this is shown to be the case. The data in this figure for both Lena and `tank farm' show similar results. The encodings with $s_{\max} = 0.5$ and 0.8 yielded the poorest results. The encodings improved with increasing $s_{\max}$ up to $s_{\max} = 1.5$. The $s_{\max} = 2.0$ case yielded a result marginally worse than the $s_{\max} = 1.5$ encoding. Results for a variety of other images and encoding parameters indicate that $s_{\max} = 1.2$ or 1.5 usually yields the best PSNR versus compression results. These results are particularly interesting because they show that it is possible to find an iterative system with a fixed point which is closer to a target by allowing some of the individual transforms to be non-contractive. It is of interest that every one of the encodings with $s_{\max} \geq 1.0$ (numerous images and several hundred separate encodings) converged to a fixed point. In a few cases (with $s_{\max} = 2.0$) a mapping took more than ten iterations to converge, but in all cases with $s_{\max} \leq 1.2$, ten iterations were sufficient. Since in practice it is desirable to perform only a small number of decoding iterations, $s_{\max} \leq 1.0$ or 1.2 were used for all other encodings in this correspondence. The question of contractivity is important, and is very much dependent on the metric which is used. For instance, the following procedure can be used to check if a mapping $W = \cup_i w_i$ is eventually contractive for the metric $\delta_{\sup}(f,g) = \sup_{(x,y)\in I^2} |f(x,y) - g(x,y)|$. Begin with an image f such that $f(x,y) = 1.0$. Define w_i' as w_i with $o_i = 0$,and $W' = \cup_i w_i'$. Then $\text{up}_{(x,y)\in I^2} \{W'^{0n}(f(x,y))\}$ will be the contractivity σ of W'^{0n}. To check if W is eventually contractive, iterate W' until $\sigma < 1$. This test only determines eventual contractivity after an encoding has been made. A similar procedure for the rms metric is not known to the authors. It is relevant to note that a mapping W which is contractive for $\delta_{\sup}$ may only be eventually contractive for δ_{rms}. Unlike the sup metric, the condition $s_i < 1.0$ is not sufficient to ensure contractivity for the rms metric. However, this condition is sufficient to ensure eventual contractivity for the rms metric. Unlike the parameters investigated in Fig. 4-6 and 9, the number of available domains more directly effects the encoding time. This is important to keep in mind when examining the data in Fig. 8.The data indicate that increasing n, typically resulted in a moderate increase in the PSNR with a marked increase in the compression. Conversely, including diagonals typically resulted in little change in the compression, but in a marked increase in the PSNR. Decreasing the lattice spacing (increasing the number of domains) resulted in

increases in both PSNR and compression. This means that. For all of these different ways of increasing the number of domains, the performance improved. Continuing studies are underway to determine at what point an increase in the number of domains results in decreased performance. In Table 2, the relative encoding times for several encodings are presented. The data indicate the relative encoding time as a function of n_c, e_c, $r_{\min}$ and the image size. On an HP-Apollo 400t workstation, the relative time units used in Table 2 are equal to approximately 1170 cpu seconds. This code has not been optimized for speed. Finally, returning to Fig. 9, data using the domain set $D^1, n_s = 5.0, n_o = 7.0$ and $s_{\max} = 1.2$, are presented over a wide range of compressions. The resulting compression was varied by the choice of e_c and $r_{\min}$. As in Fig. 8, it is shown that (at the cost of speed) the encodings can be improved by increasing n_c. Figure 9 and Table 2 indicate the trade off between encoding time and PSNR when varying n_c, and give an indication of the efficiency of the classification method. For comparison, results are also shown for an ADCT algorithm similar to that described in [3].The ADCT data shown is as good as the JPEG standard at lower compression, and better than the JPEG standard at higher compression [9]. The PSNR versus compression performance of the current iterated transform method is comparable to, but not as good as, the ADCT algorithm. It is interesting to compare the implementation presented here with that of [6].Among several other differences, some major differences are that in [6] the restriction $s_i \leq 1.0$ was imposed, a Hausdorff matrix as well as rms criteria was used during encoding, only 10 values for s were allowed, and the set of allowed domains was localized and small (about the same in number as $D^{1/8}$). Therefore, relative to any of the coding presented here, more transformations requiring fewer bits each were used to encode the image. It is interesting that this different approach yields comparable results, although the implementation described in this correspondence (last encoding in Table 2) results in an improvement of 1.4 dB at the same bit-rate. In a later reference by the same author [8], 512×512 images were encoded with an algorithm similar to that of [6]. In Table 2, the 512×512 encoding with $n_c = 1.0$ indicates a 1.8 dB improvement fidelity in a slightly better compression (Note that in [6-8] fidelity is computed as SNR not PSNR. In the discussion of Fig. 4, it is specifically noted that in choosing `best' values for n_s and n_o data from several images were considered. Data in Fig. 6 is presented for two images, and in Fig. 7 for one image. Results on several other images that have been encoded show qualitatively similar behavior. In reference to Fig. 9, the compression and PSNR obtained for a given set of encoding parameters depends on the image. Note that the resulting PSNR for the image is not necessarily close to e_c. Therefore it would be difficult to a priori choose parameters that will result in a target PSNR. An algorithm which accurately targets compression can be made with some simple modifications to the encoding procedure described here. (Instead of quadtreeing based on e_c the encoding process can be structured such that quadtreeing continues until a target number of transformations is reached.)

Table-2: Relative Time to encode Lena for various parameters

$s_{\max}$	n_c	$r_{\min}$	e_c	Resolution	rel	Compression	PNSR (dB)
1.2	1	4	8	512	1	15.95:1	33.13
1.2	4	4	8	512	3.1	17.04:1	33.19
1.2	72	4	8	512	35.5	17.87:1	33.40
1.0	4	4	8	512	3.1	16.74:1	33.30
1.2	4	4	5	512	5.3	10:49:1	35.92
1.2	4	4	11	512	2.0	24.62:1	30.85
1.2	72	8	8	512	7.5	36.78:1	30.71
1.2	1	4	8	256	0.14	9.09:1	30.63
1.2	72	4	8	256	4.5	9.97:1	31.53
1.2	72	4	10	256	3.7	11:85:1	30.58

In conclusion, it should be noted that because encoding an image is the interesting problem, decoding of images has only been briefly mentioned in this correspondence. The decoding of images using the iterated transform method is inherently fast (requiring an iteration which is computationally simpler

than the inverse transform required for ADCT), an important advantage depending on the application. Even though vector quantization (VQ) methods have more in common with iterated transforms, ADCT has been used for comparison of data because the method is more standardized. In general, VQ methods might be expected to encode and decode faster than iterated transforms. Because VQ uses a fixed code book and iterated transforms is self referential, iterated transforms might be expected to work better for applications that require encoding of a wide variety of images. Current work involves investigation of new classification schemes that maintain PSNR compression performance while reducing encoding time; using linear combinations of domains; different image partitioning methods; and application to color images.

REFERENCES

1. M.F. Barslcy, Fractals Everywhere, Academic Press, San Diego, CA, 1988.
2. M.F. Barnsley and A.E. Jacquin, "Application of recurrent iterated function systems to images", SPIE Visual Comm. Image Process, Vol. 1001, 1988, pp. 122 131.
3. W. Chen and W.K. Pratt, "Scene adaptive coder", IEEE Trans. Comm., Vol. 32, 1984, pp. 225 232.
4. Y. Fisher, E.W. Jacobs and R.D. Boss, "Iterated transform image compression", NOSC TR 1408. Naval Ocean Systems Center, San Diego, CA, 1991 (available from authors upon request).
5. J.E. Hutchinson, "Fractals and self-similarity". Indiana Univ. Mach. J., Vol. 35, 1981, p. 5.
6. A.E. Jacquin, A fractal theory of iterated Markov operators, with applications to digital image coding, Ph.D. Thesis, Department of Mathematics, Georgia Institute of Technology, 1989.
7. A.E. Jacquin, "A novel fractal block-coding technique for digital images", IEEE Internat. Conn. Acoust. Speech Signal
8. A.F. Jacquin, Fractal image coding based on a theory of iterated contractive image transformations", SPIE Visual, Comm_ Image Process. '90, Vol. 1360, 1990, pp. 227-239.
9. M. Rabbani and P.W. Jones, Digital Image Compression Techniques, SPIE Optical Engineering Press, Bellingham, Process. 1990. Vol. 4, 1990, pp. 2225 2228. WA, 1991, Chapter 10, pp. 121.
10. M.A. Moustafa., A. Alsayyh, novel fused Image compression technique using DFT, DWT, and DCT, Journal of information hiding and multimedia signal processing,2017, Vol. 8, No.2, pp. 261-271
11. Norouzi, A. Rahim, MSM, Altameem, A. Saba, T. Rada, A.E. Rehman, A. and Uddin, M. Medical image segmentation methods, algorithms, and applications IETE Technical Review, vol.31, no.3, pp. 199-213, 2014, doi. 10.1080/02564602.2014.906861.
12. Harouni, M., Rahim,MSM., Al-Rodhaan,M., Saba, T., Rehman, A., Al-Dhelaan, A. Online Per- sian/Arabic script classi_cation without contextual information, The Imaging Science Journal, vol. 62, no.8, pp. 437-448, 2014, doi. 10.1179/1743131X14Y.0000000083
13. Joudaki, S. Mohamad, D. Saba, T. Rehman, A. Al-Rodhaan, M. Al-Dhelaan, A. Vision based sign language classi_cation: a directional Review, IETE Technical Review, Vol.31 , no.5, pp. 383-391, 2014, doi. 10.1080/02564602.2014.961576
14. Saba T, Al-Zahrani S, Rehman A. Expert system for o_ine clinical guidelines and treatment Life Sci Journal 2012; vol. 9, no. 4, pp. 2639-2658, 2012.