Chaos, Solitons and Fractals 106 (2018) 16–22

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

journal homepage: www.elsevier.com/locate/chaos

Fractal image compression using upper bound on scaling parameter

Swalpa Kumar Roy[a,*], Siddharth Kumar[b], Bhabatosh Chanda[c], Bidyut B. Chaudhuri[a], Soumitro Banerjee[d]

[a] Computer Vision & Pattern Recognition Unit, Indian Statistical Institute, Kolkata 700108, India
[b] Department of Computer Science, San Jose Sate University, San Jose, CA 95192, USA
[c] Electronics & Communication Sciences Unit, Indian Statistical Institute, Kolkata 700108, India
[d] Department of Physical Sciences, Indian Institute of Science Education and Research, Mohanpur Campus, Kolkata 741246, India

ARTICLE INFO

Article history:
Received 14 June 2017
Revised 29 October 2017
Accepted 9 November 2017

Keywords:
Fractal coding speedup
Scaling parameter upper-bound
Image data compression

ABSTRACT

This paper presents a novel approach to calculate the affine parameters of fractal encoding, in order to reduce its computational complexity. A simple but efficient approximation of the scaling parameter is derived which satisfies all properties necessary to achieve convergence. It allows us to substitute to the costly process of matrix multiplication with a simple division of two numbers. We have also proposed a modified horizontal-vertical (HV) block partitioning scheme, and some new ways to improve the encoding time and decoded quality, over their conventional counterparts. Experiments on standard images show that our approach yields performance similar to the state-of-the-art fractal based image compression methods, in much less time.

1. Introduction

Fractal image compression (FIC) is one of the important methods of gray scale image coding, fully automated by Jacquin [1]. The basic FIC is based on the observation that images usually exhibit affine redundancy. Here an image is segmented into a number different sized blocks and the encoding process consists of approximating the small image blocks, called *Range blocks* (RBs), from the larger blocks, called *Domain blocks* (DBs), of the image, by searching the best matching affine transformation from a DB pool, much akin to the image compression by *vector quantization* (VQ) method [2]. In the encoding process, separate transformations for each RB are obtained. For decoding, the set of affine transformations, when iterated upon arbitrary initial image, produces a fixed point (attractor) that approximates back the target image. This scheme, named *Partitioned Iterative Function System* (PIFS), was proposed by Fisher [3]. Although block matching is time-consuming, FIC offers high compression ratio and very fast decoding. Since domain search involves pixel-by-pixel matching as well as contraction and rotation, in order to find the suitable affine transform, FIC encoding presents computational challenge. Hence, reducing the matching complexity has been a subject of comprehensive research efforts. Such methods may be grouped into the *classification based approach, feature-vector based approach*, and *meta-heuristic approach*.

Classification based approaches use one or more common characteristics like mean and standard deviation of pixel intensity of the image blocks as features. These features are then used to classify a DB into one of a fixed number of classes. Then, the domain inspection is limited only to its class of the RBs, thereby reducing the search time. Fisher's [4] classification scheme is one such approach which divides a block into four quadrants of the same size. Then, for each quadrant, average pixel intensity and the corresponding variances are calculated and used to classify the DBs into one of the ${}^4P_4 = 24$ classes, based on their relative ordering. Hurtgen & Stiller [5] and Bhattacharya [6] modified the Fisher's method and obtained some improved performance.

Saupe [7] proposed a technique based on feature vector for improving the DB search. It achieves good output fidelity and better compression efficiency with fewer number of comparisons. But drawbacks of this method are that the encoding time is significantly increased and the feature vectors have larger dimension. Saupe's [8,9] idea of discarding low variance DBs gives a considerable gain in the speed but it is still slower than some previous methods [10]. To further improve the selection of relevant DBs, Tong [11] proposed an adaptive method to eliminate the DBs with variance below a certain threshold, hence the name is *Adaptive Search*. This method allows for a good speed-up by eliminating the irrelevant blocks using the value of the scaling parameter only. Further improvements in speed were obtained by discarding some of

* Corresponding author.
E-mail addresses: swalpa@students.iiests.ac.in (S.K. Roy), siddharth.kumar@sjsu.edu (S. Kumar), chanda@isical.ac.in (B. Chanda), bbc@isical.ac.in (B.B. Chaudhuri), soumitro@iiserkol.ac.in (S. Banerjee).

https://doi.org/10.1016/j.chaos.2017.11.013
0960-0779/© 2017 Elsevier Ltd. All rights reserved.

the DBs at the time of DB pool creation [12–14]. It can be observed that many DBs are never used in a typical fractal compression and hence can be discarded. The *No Search* method proposed by Furao [15] was improved by Wang [16], which only searches neighbors of a given RB instead of creating the virtual domain pool. The gain in encoding speed obtained by these methods is often achieved at the expense of image quality. However, the main issue lies in maintaining a balance between quality and time.

There exist several works on finding fast match based on meta-heuristic algorithms like Genetic Algorithm (GA) [17], Particle Swarm Optimization(PSO) [18] etc. These methods have also been applied in FIC to reduce the encoding time. A major drawback of these evolutionary algorithms is that they do not guarantee the best match for the given RB. Moreover, their encoding time is much higher compared to the classification and feature-vector based approaches.

Zhao et al. [19] presented a new affine transform for fractal image encoding. This scheme is fundamentally more generalized form of conventional affine transform and contains two scaling parameters. Lu et al. [20] presented a fractal coding scheme based on Huber norm and a new matching error function, along with the no-search method to reduce computational costs.

To the best of our knowledge, there exists no work dealing with alternative ways of speed-up other than small variations of previously discussed approaches [21,22]. However, a simple observation shows that any approach to FIC needs computation of affine parameters that involves expensive matrix multiplication step for obtaining the scaling parameter (s). So, the question is, can we replace matrix multiplication by some cheaper calculation, so that time complexity is reduced without sacrificing the image quality. This paper is an attempt to answer this point.

The method proposed here uses an approximation of the scaling parameter that avoids the computation of the inner product, thereby considerably speeding up the search process. Also, in this method we use a modified HV partitioning approach. In addition, some optimization used to reduce the computation of affine transforms yields a superior compression compared to the conventional HV [23] partitioning based approach.

Some of the salient aspects of the proposed approximation are as follows:

- It attains considerable speed-up in the process of domain-block search.
- It can be combined with most existing fractal coding methods to reduce the search time.

The organization of the rest of this paper is as follows. Section 2 contains a brief presentation of the basic fractal coding scheme [24]. Section 3 contains the detailed mathematical and derivation for the upper bound of the scaling parameter. Section 4 presents the different partitioning schemes, their limitations and our modified HV partition scheme that yields superior results. In Section 5 we present the performance of our method and benchmark it against some popular and widely used FIC methods. Finally, conclusions are drawn in Section 6.

2. Full search fractal image coding

In the baseline FIC, an image f of $M \times N$ pixel is partitioned into a set of non-overlapping RBs of $r \times r$ pixels (called range pool R). The encoding of each RB consists of finding the best affine transformation by searching a set of global or partial overlapping domain blocks of size $2r \times 2r$ pixels (called domain pool or virtual codebook D). The DBs are extracted from f by sliding a window of size $2r \times 2r$ pixels, starting from the top-left corner, at integral steps δ (domain step size) in both horizontal and vertical directions. When $\delta = 1$ there are N_D $((M - 2r + 1) \times (N - 2r + 1))$ DBs in D. For convenience of range-domain comparison, each domain block D_j must be spatially contracted (or decimated) by pixel averaging to match the RB size and used as the codebook Ω $\{D_j^c | D_j^c = S(D_j)\}$ to approximate each RB with a gray level transformation τ, such that $\tau(D_j^c)$ provides a suitable matching for RB in the least squares sense. Moreover, the codebook is enlarged by including all rotations and reflections of each domain. The key highlight of FIC scheme, a range block $R_i \in R$ is compared with every domain blocks $D_j^c \in \Omega$ to obtain minimum of the following transform:

$$\min_{D_j^c \in \Omega} \mathrm{MSE}(R_i, D_j^c) = \min_{D_j^c \in \Omega} \min_{s, o \in \Re} \left\| R_i - (s \cdot \varphi(D_j^c) + oI) \right\|_2^2 \tag{1}$$

Where $\|.\|_2$ is the ℓ_2-norm, s and o are the contrast scaling and the luminance shift or offset parameters of the transformation (τ) , I is the constant block with unity value at every pixel and φ is the isometry transformation. In Eq. (1), the part $\tau(D_j^c) = s \cdot \varphi(D_j^c) + oI$, is the gray level transformation. The optimal affine parameters are obtained when $\frac{\partial \mathrm{MSE}}{\partial s_j} = 0$ and $\frac{\partial \mathrm{MSE}}{\partial o_j} = 0$. Applying these partial derivatives in Eq. (1) we obtain:

$$s_i = \frac{\left\langle R_i - \bar{r}_i I, D_j^c - \bar{d}_j . I \right\rangle}{\|D_j^c - \bar{d}_j I\|^2} = \frac{\sigma_{R_i D_j^c}}{\sigma^2_{D_j^c}}$$
$$o_i = \bar{r}_i - s_i \bar{d}_j \tag{2}$$

Here $\bar{r}_i$ and $\bar{d}_j$ are the average pixel intensities of R_i and D_j^c, respectively. Also, $\sigma_{D_j^c}$ is the standard deviation of D_j^c, and $\sigma_{R_i D_j^c}$ is the covariance between R_i and D_j^c, respectively. The discrepancy or distortion can be simplified as follows:

$$\min_{D_j^c \in \Omega} \mathrm{MSE}(R, D_j^c) = \min_{D \in \Omega} \min_{s, o \in \Re} \sum_{j=0}^{r-1} \sum_{i=0}^{r-1} (r_{ij} - (s \cdot \varphi(d_{ij}) + o))^2 \tag{3}$$

The scaling parameter s is clamped in $[-1, 1]$ to ensure convergence in the decoding phase. During the actual coding process, a coding scheme just based on the quantization of the optimal parameters cannot be used, rather to ensure contractivity we also need a constraint on the range of the contrast scaling parameter and both s and o are uniformly quantized, yielding s_q and o_q. Hence, it is better to minimize the MSE using the quantized parameters. This error $\widehat{\mathrm{MSE}}(\mathrm{R_i}, \mathrm{D_j^c})$, also known the *collage error*, is given by

$$\begin{aligned} &\min_{D_j^c \in \Omega} \widehat{\mathrm{MSE}}(R_i, D_j^c) \\ &= \min_{D_j^c \in \Omega} \min_{s_q, o_q \in \mathbb{R}} \widehat{\mathrm{MSE}} \|R_i - (s_q \varphi(D_j^c) + o_q I)\|^2 \end{aligned} \tag{4}$$

where s_q and o_q are the quantized scaling and luminance parameters. The minimization of Eq. (4) is the problem of full search fractal coding, which is computationally intensive. Domain block classification schemes can substantially reduce the number of domains to be searched for the suitable affine transform.

3. Proposed upper bound on scaling parameter

Using the results of Eq. (2) we can compute the values of the contrast scaling and luminance offset for obtaining the mean square error between R_i and D_j^c. It is also evident that the inner product of the RB and the DB intensities $\left\langle R_i - \bar{r}_i I, D_j^c - \bar{d}_j . I \right\rangle$ has to be pre-computed before we calculate the distortion and determine fitness of the DB, based on the mean square error, which is computationally expensive. To circumvent this time consuming matrix product calculation, we propose the use of an approximate value of the scaling parameter based on its upper limit derived as follows, from Eq. (4), we get

$$\begin{aligned}\sqrt{\widehat{MSE}(R_i, D_j^c)} &= \|R_i - s_q D_j^c - oI\| \\ &= \|R_i - s_q D_j^c - (\bar{r}_i - s_q \bar{d}_j)I\| \\ &= \|(R_i - \bar{r}_i I) - s_q (D_j^c - \bar{d}_j I)\| \\ &\geq \left| \|R_i - \bar{r}_i I\| - s_q \|D_j^c - \bar{d}_j I\| \right|\end{aligned} \tag{5}$$

by omitting the small amount of quantization error for the RB. The condition of Eq. (5) is equivalent to $\left|\sigma_{R_i} - s_q \sigma_{D_j^c}\right| \leq \mathrm{RMS}(R_i, D_j^c)$, where RMS is the root mean square matching error for the RB. This result allows not to consider those DBs for which the following condition is not satisfied.

$$\left|\sigma_{R_i} - s_q \sigma_{D_j^c}\right| \geq T \tag{6}$$

Here T is the threshold selected for domain-range matching in the least square sense. Since, this condition is dependent on the value of the scaling parameter only, it effectively reduces the search time.

This process of rejecting the DBs based on the above condition still needs the computation of the matrix inner product of R_i and D_j^c, which can be avoided as follows. Using Eqs. (2) and (3), we obtain:

$$\begin{aligned}&MSE(R_i, D_j^c) \\ &= \|R_i - s_q D_j^c - \bar{r}_i + s_q \bar{d}_j\|^2 \\ &= \|R_i - \bar{r}_i - s_q (D_j^c - \bar{d})\|^2 \\ &= \|R_i - \bar{r}_i\|^2 - 2 s_q \langle R_i - \bar{r}_i I, D_j^c - \bar{d}_j.I\rangle + s_q^2 \|D_j^c - \bar{d}_j\|^2\end{aligned} \tag{7}$$

Using Eq. (2), we can write:

$$\langle R_i - \bar{r}_i I, D_j^c - \bar{d}_j.I\rangle = s_q \|D_j^c - \bar{d}_j\|^2 \tag{8}$$

Substituting Eq. (8) in Eq. (7), we obtain:

$$\begin{aligned}MSE(R_i, D_j^c) &= \|R_i - \bar{r}_i\|^2 - 2{s_q}^2 \|D_j - \bar{d}_j\|^2 + s_q^2 \|D_j^c - \bar{d}_j\|^2 \\ &= \|R_i - \bar{r}_i\|^2 - s_q^2 \|D_j^c - \bar{d}_j\|^2\end{aligned} \tag{9}$$

where the s_q is the scaling parameter, $\bar{r}_i$ and $\bar{d}_j$ are the mean of R_i and D_j, respectively. We know that, $MSE(R_i, D_j^c) \geq 0$, hence using the results in Eq. (9) we can write:

$$\begin{aligned}&\|R_i - \bar{r}_i\|^2 - s_q^2 \|D_j^c - \bar{d}_j\|^2 \geq 0 \\ \Rightarrow\quad &\sigma_{R_i}^2 - {s_q}^2 \sigma_{D_j^c}^2 \geq 0 \\ \Rightarrow\quad &s_q \leq \frac{\sigma_{R_i}}{\sigma_{D_j^c}}\end{aligned} \tag{10}$$

From the inequality in Eq. (10), we get the upper bound of the scaling parameter, which is equal to the ratio of the standard deviation of the RB (R_i) and the DB (D_j^c).

Using the scaling parameter s_q by Eq. (10) saves big amount of computational time. To further reduce the computation time, we adopt Fisher's classification [3], where for a given RB we first determine it's class and then, calculate the scaling parameter s_q from a domain pool, which is of similar class with RB. We choose the optimal and uniformly quantized scaling parameter s_{opt} from a set of scaling parameters s_q for which the MSE is minimum. This would allow to eliminate the unlikely DBs in a much faster way than the "state-of-the-art" methods. In the results section we have shown that despite taking an approximate value for the scaling parameter, there is hardly any loss in output quality as compared to the scaling value using Eq. (2).

4. Modified HV partitioning

The FIC's exhibit trade-offs between block size and the amount of compression achieved. To obtain high compression, variable block sizes are useful. A simple and efficient method to incorporate this is the quadtree approach [4], where a range block is split continually into four sub-blocks of equal size, until the matching error is less than a threshold. However, partitioning into two sub-blocks can be sufficient in some cases. The horizontal or vertical (HV) [23] partitioning may be more efficient in such case, since it partitions by a factor of two.

In the HV partitioning, the partitions may not be of equal size. One bit is employed for each block to denote weather it has been partitioned. Another bit is employed to denote the type of partition (horizontal or vertical) and $\log_2(E_{\max})$ bits for the index of the partition, where $E_{\max}$ is the maximum edge size of the block considered in the scheme. In our case this value is 4 as the maximum edge size of a block is 16 pixels. Fig. 1(b) shows the HV partitioning on the Lenna image. It avoids redundant partitioning, and provides a better match.

Here we define a new parameter named Partition Step Size (*PSS*), which when added to the row-index (k) or the column-index (l) gives the next index of the next row or column. It is defined as follows:

$$\begin{aligned}&r_{j+1} = r_j + PSS, \qquad c_{j+1} = c_j, \\ &\qquad\qquad\qquad or \\ &c_{i+1} = c_i + PSS, \qquad r_{j+1} = r_j\end{aligned} \tag{11}$$

where r_j, c_i denote the current row and column, respectively. r_{j+1} and c_{i+1} denote the next row and column considered in the HV scheme. In short, *PSS* can be defined as the number of rows (or columns) after which we compute the partition index in the HV partitioning scheme. For example, $PSS = 1$ will imply that after considering the current row (or column) with index k (or l), we go to row $k+1$ (or column $l+1$), same as the conventional HV partition. Therefore, using a value of $PSS \geq 1$, we can both speed-up the partitioning process and reduce the size of the coded file. There is a trade-off between the value of *PSS* and the quality of the resulting image. A large value of *PSS* means faster partitioning, but results in a poor quality compression. For our implementation we have taken $PSS = 2$ which allows us to circumvent the above disadvantage without adversely affecting the decompressed image quality. Moreover, the following modifications have been made to speed-up the search-match process.

During domain search we compute some characteristics of the range block R_i and then the pool is searched based on these characteristics. For practical considerations we have used two types of characteristics as follows:

1. To extract the first characteristic, we suppose that R_i is divided into four partitions, then compute the sum of the pixel gray level values for each of the partition (S_i). Now, using isometric transformations, the four quadrants can be arranged in an decreasing order of the sum of gray level values i.e. ($S_1 > S_2 > S_3 > S_4$) starting with the top-left quadrant. The sequence of the blocks is considered as the first characteristic.
2. Let $V_1 = S_1 - S_2$, $V_2 = S_2 - S_3$, $V_3 = S_3 - S_4$, then we take the order of the sequence of V_1, V_2, V_3 as the second characteristic for classification. The six possible orders of V_1, V_2, V_3 form the basis of this classification.

Now, for each R_i we compute these characteristics and find the domain blocks with the same characters, level by level. This makes the process of matching faster and more accurate.

5. Results and discussion

The proposed technique has been tested on benchmark images chosen from the USC-SIPI Image Database [25] as shown in the Fig. 2. We used OpenCV library for reading the images and implemented the algorithm in C++. All the programs were tested on an

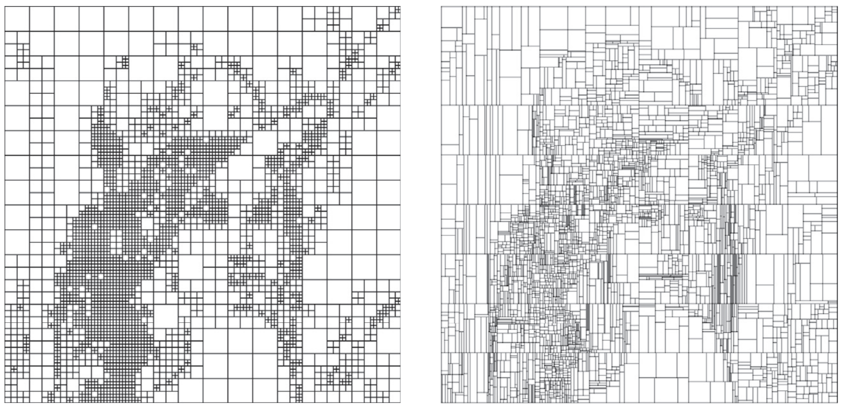
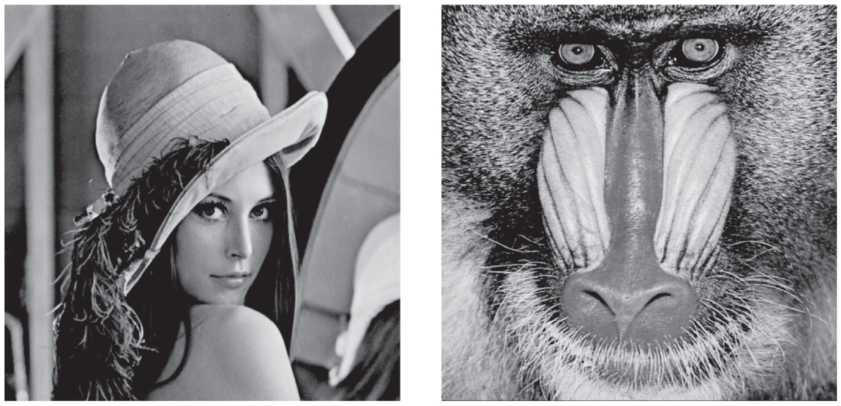
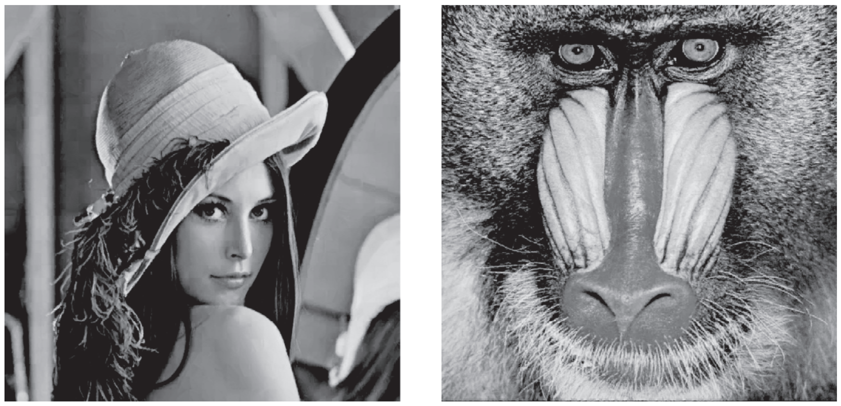


Fig. 1. (a) Quadtree partitioning on Lenna image (5848 RBs). (b) HV partitioning on Lenna image (4606 RBs).



(a) Lenna (512 × 512) (b) Baboon (512 × 512)



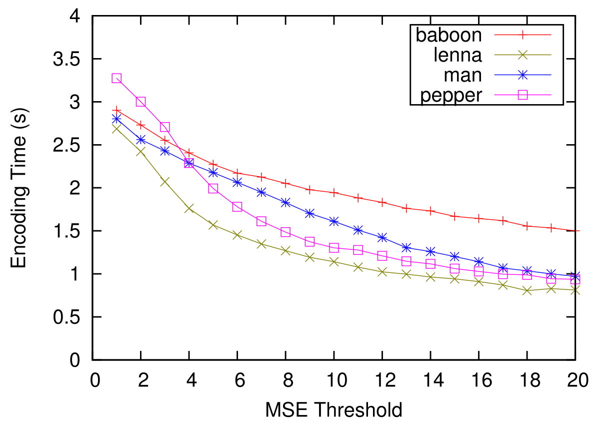
(c) Decoded (PSNR: 37.67 dB, SSIM: 94.20%, bpp: 0.60) (d) Decoded (PSNR: 25.80 dB, SSIM: 72.30%, bpp: 0.60)

Fig. 2. (a) - (b) Benchmark test images. (c) - (d) Decoded images at best possible quality.

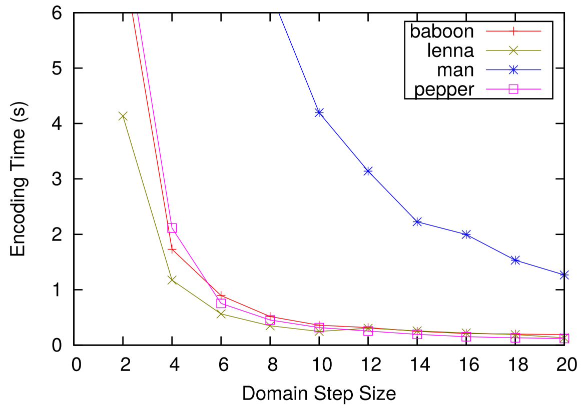
Ubuntu 14.04 LTS machine running on Intel i5-3210M CPU along with 8GB of DDR3 memory.

Fig. 3(a) shows the variation of encoding time with the target MSE threshold. It can be seen that there is not much variation in the encoding time with change in the MSE threshold, suggesting that the proposed method performs equally well for any given target fidelity in terms of speed.

Fig. 3(b) shows the variation of encoding time with domain step-size δ. For $\delta = 2$ the overlap between the domains is maximum and for $\delta = 20$ it is the least. Hence, smaller value of δ means more number of domains to search hence, the encoding is slower, while large values of δ will faster encoding.

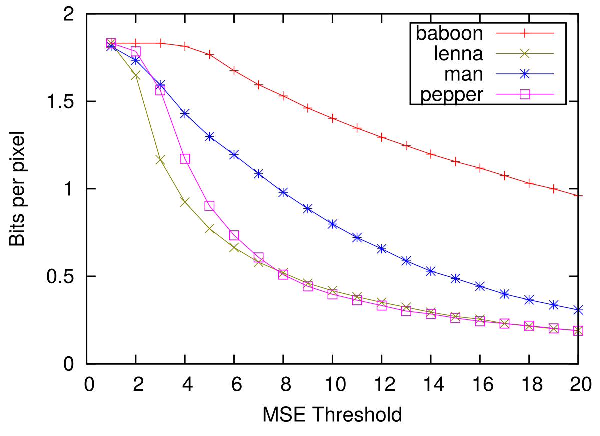


(a)

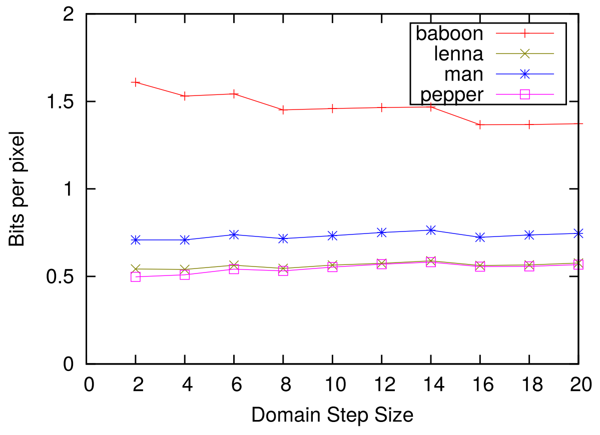


(b)

Fig. 3. Encoding time for the proposed method with respect to MSE threshold and domain step size.



(a)



(b)

Fig. 4. Variation of bits per pixel (bpp) for different values of MSE threshold and domain step size.

The compression rates has been presented in terms of bits per pixel (bpp). Since the test images considered are 8 bpp images, the image must be encoded in less than 8 bpp to achieve compression. Fig. 4(a) shows the variation of bpp with the selected MSE threshold value. Lower MSE threshold implies better decoded quality, but leads to a lower compression ratio since more transformations have to be encoded. Consequently, as we increase the threshold, the compression ratio also increases. Fig. 4(b) shows the variation of bpp for different values of domain step size. It can be seen that there is less variation in bpp with respect to change in domain step-size.

The decoding process begins with an image with all the pixels with the gray value of 128. Then the encoded affine transforms are applied recursively on the initial image. The final decoded image is obtained in 8 iterations, shown in Fig. 2(c)-(d). For decoded image quality, we have used Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [26]. The SSIM values ranges from 0 to 1, where a value close to 1 indicates that the decoded image is almost error-free.

Table 1 shows the distance between the histograms of the original test images and decoded images using various histogram comparison metrics available [27–29]. For the Correlation and Intersection metrics, the higher the metric, the more accurate the match. For the other two metrics, the less the result, the better the match.

The scatter plot in Fig. 5(a) shows the variation of error obtained for different values of the affine parameters *s* and *o*. The error shown corresponds to the values of the parameters calculated using Eq. (2). On the other hand, the scatter plot in Fig. 5(b) shows the variation of error obtained for different values of the affine parameters *s* and *o* computed in our proposed technique, using Eq. (10). Table 2 shows the comparison of proposed scheme with other well-known FIC schemes, with respect to PSNR, BPP, and Time (in sec), for four standard images (Lenna, Man, Boat, Baboon) of size (512×512). The best results are reported for the proposed method with quadtree (proposed-Quadtree) and HV (proposed-HV) partitioning schemes. The data shows that the proposed-HV scheme provides better image quality compared to other methods, but with an increased encoding time. It also depicts that the proposed-Quadtree schemes considerably outperforms other well-known FIC methods like Polvere's Mass Center Features (Mass-Center) [31], Fisher's 24 scheme [4], Saupe's multi-dimensional codebook search (Saupe) [32] and Hurtgen's hierarchical codebook search scheme (Hurtgen) [5], in term of average PSNR, BPP and encoding time. Combining Saupe's search with Polvere's Mass Center Features (Saupe-MC) [31], and combining Saupe's search with Fisher's classification (Saupe-Fisher) [31] give faster encoding times, but proposed method is substantially better in decom-

Table 1
Histogram comparison between original and decoded images for proposed method.

Histogram Comparison	Baboon	Lenna	Man	Pepper
Correlation	0.876	0.993	0.969	0.976
Intersection	87.311	103.15	34.980	99.427
χ^2-Test [27]	2.532	0.391	4.961	0.612
Bhattacharyya distance [28]	0.231	0.021	0.048	0.019
Kullback–Leibler divergence [29]	12.214	3.071	9.642	6.102

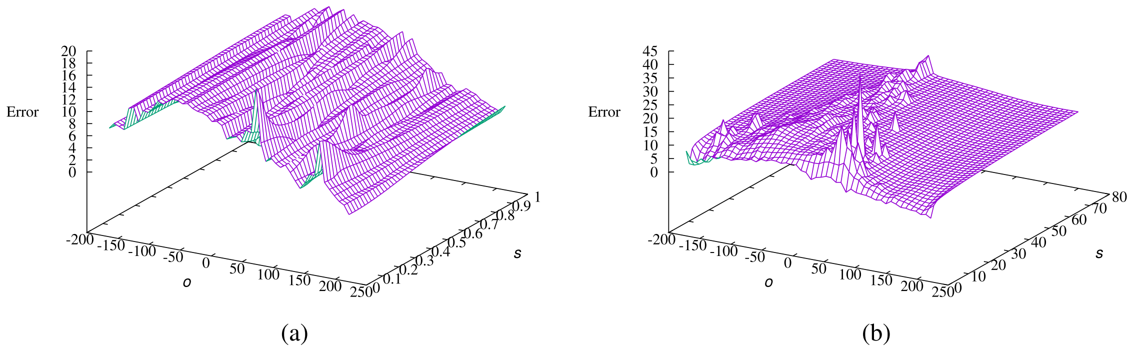


Fig. 5. (a) Plot of error variation with scaling (s) and offset (o) parameters for the Lenna image, using the convectional technique of the calculation of 's and (b) Using our proposed technique based on the upper bound approximation.

Table 2
Comparison of proposed scheme with other conventional fractal coders, w.r.t. PSNR, BPP, and Time (in sec), for four standard images.

Images	Quality	Fisher [30]	Polvere [31]	Saupe [32]	Hurtgen [5]	Saupe-MC [31]	Proposed-Quadtree	Proposed-HV
Baboon	PSNR	24.28	24.43	25.54	24.41	24.45	25.21	25.80
	BPP	0.60	0.61	0.60	0.60	0.60	0.60	0.60
	Time	1.60	2.80	15.40	2.8	3.37	1.57	8.90
Boat	PSNR	33.95	34.82	35.61	34.62	34.63	34.21	35.83
	BPP	0.60	0.60	0.60	0.60	0.61	0.60	0.60
	Time	2.40	4.70	12.40	4.07	4.60	2.19	11.20
Lenna	PSNR	35.58	36.10	36.20	35.90	35.90	36.12	37.67
	BPP	0.60	0.60	0.60	0.59	0.60	0.60	0.60
	Time	2.20	3.60	13.20	3.50	4.30	1.90	12.19
Man	PSNR	30.1992	30.50	31.80	30.45	30.44	31.25	32.60
	BPP	0.60	0.60	0.60	0.61	0.60	0.60	0.71
	Time	1.8	2.9	15.4	3.30	3.90	1.15	11.60
Average	PSNR	31.00	31.46	31.25	31.35	31.35	**31.69**	**32.72**
	BPP	0.60	0.60	0.60	0.60	0.60	**0.60**	0.60
	Time	2.00	3.50	14.10	3.41	4.04	**1.70**	10.97

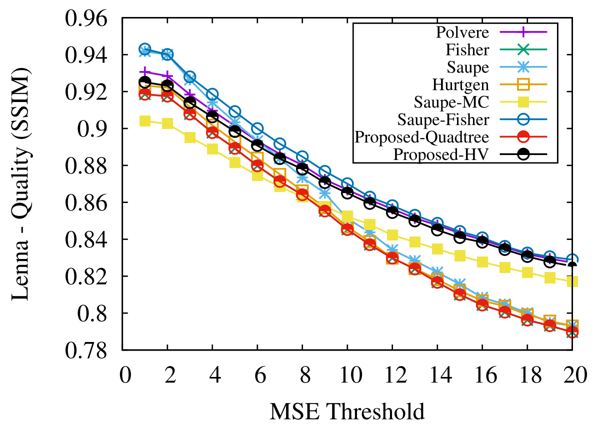


Fig. 6. Variation in image quality (SSIM) with change in the target MSE threshold for Lenna image.

pressed image quality. Zhao et al. [19] compressed a 256 × 256, 8 bpp Lenna image and yields the PSNR = 31.7 dB at a compression ratio (CR) = 8.4, whereas the proposed method yields PSNR = 32.72 at a CR = 13.33. Lu et al. [20] compressed a 512 × 512, 8 bpp Lenna image, achieves PSNR = 31.04 dB and takes 1.679 s time, the proposed method yields a slightly improved of PSNR = 31.69 dB taking 1.03 s time. Among all the FIC approaches based on metaheuristic, the most superior results were reported by Wu et. al. [33]. For the Lenna image (512 × 512), the best result obtained in terms of PSNR was 27.78 dB while in our method it is 36.12dB. Hence we can say that, the output quality of the proposed method clearly outperforms this approach.

Fig. 6 presents the plot for the variation of Sturctural Similarity Index (SSIM) with respect to the target MSE threhold for some of the widely used methods for FIC.

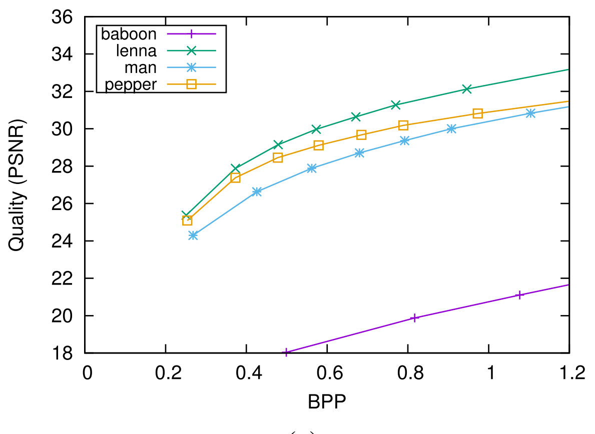
Fig. 7(a) shows the variation of the output image quality with varying BPPs in the compressed image for JPEG[1] [34], and Fig. 7(b) shows the same for the proposed method using quadtree partitioning. From the plot we can observed that for lower values of BPP (typically, < 1) the proposed method outperforms JPEG.

As the proposed method speeds-up the domain search and match procedure, we have obtained faster encoding time compared to some of the other fractal encoding methods. Effectively rendering a faster encoding time without any supplementary loss in the output quality.

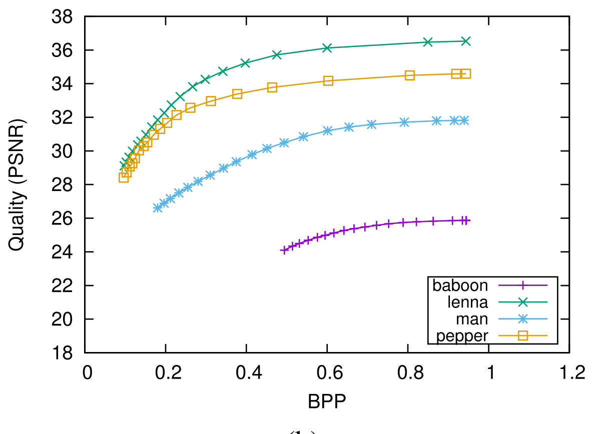
6. Conclusions

The huge encoding time for FIC is one of its major drawbacks. Many researches have investigated methods to reduce the encoding complexity of FIC. Clearly, the highest computation cost lies in pixel by pixel operation during domain-range matching. Therefore, we explored the possibility of substituting matrix multiplication with something that is faster and exhibits are the necessary characteristics. In this paper, we advocated the use of an approximation of an affine parameter, which involves much less computation compared to matrix inner product for the scaling parameter which also allows to filter out unsuitable domains during the search-match step of FIC. We have shown that using this parameter, along with adaptive search has many advantages over the traditional search-match process. We have also introduced a modified HV partitioning scheme, which greatly speeds up encoding while

[1] Available: https://jpeg.org/jpeg/index.html.



(a)



(b)

Fig. 7. Variation in image quality (PSNR) with change in the BPP for the standard images using the; (a) JPEG and the (b) Proposed-Quadtree (Proposed_QT) technique.

maintaining high fidelity of the decoded image. Significantly good results were obtained when the scaling parameter approximation is used in conjunction with adaptive search and the proposed partition scheme. Comparison with our benchmark full search fractal encoding confirms that our algorithm not only accelerated the block matching procedure, but also provided a high compression ratio with remarkable visual quality.

Acknowledgment

We are thankful to Mario Polvere, University of Salerno, Italy for sharing the program codes of various classification and feature vector based FIC schemes.

References

[1] Jacquin AE. Image coding based on a fractal theory of iterated contractive image transformations. IEEE Trans Image Process 1992;1(1):18–30.
[2] Ramamurthi B, Gersho A. Classified vector quantization of images. IEEE Trans Commun 1986;34(11):1105–15.
[3] Fisher Y, Jacobs E, Boss R. Fractal image compression using iterated transforms. In: Image and text compression. Springer; 1992. p. 35–61.
[4] Fisher Y. Fractal image compression with quadtrees. In: Fractal image compression. Springer; 1995. p. 55–77.
[5] Hürtgen B, Stiller C. Fast hierarchical codebook search for fractal coding of still images. In: Berlin-DL tentative. International Society for Optics and Photonics; 1993. p. 397–408.
[6] Bhattacharya N, Roy SK, Nandi U, Banerjee S. Fractal image compression using hierarchical classification of sub-images. In: Proceedings of the 10th international conference on computer vision theory and applications; ISBN 978-989-758-089-5; 2015. p. 46–53.
[7] Saupe D. Accelerating fractal image compression by multi-dimensional nearest neighbor search. In: Data compression conference, 1995. DCC'95. proceedings. IEEE; 1995. p. 222–31.
[8] Saupe D. Lean domain pools for fractal image compression. In: Electronic imaging: science & technology. International Society for Optics and Photonics; 1996. p. 150–7.
[9] Saupe D, et al. Breaking the time complexity of fractal image compression. Citeseer; 1994.
[10] Saupe D, Ruhl M. Evolutionary fractal image compression. In: Image processing, 1996. proceedings., international conference on, 1. IEEE; 1996. p. 129–32.
[11] Tong CS, Pi M. Fast fractal image encoding based on adaptive search. IEEE Trans Image Process 2001;10(9):1269–77.
[12] Monro DM, Dudbridge F. Fractal approximation of image blocks. In: Acoustics, speech, and signal processing, 1992. ICASSP-92., 1992 IEEE international conference on, 3. IEEE; 1992. p. 485–8.
[13] Suad J. Fractal image compression based on entropy technique. IOSR J VLSI Signal Process 2013;2(1):27–30.
[14] Lai C-M, Lam K-M, Siu W-C. A fast fractal image coding based on kick-out and zero contrast conditions. IEEE Trans Image Process 2003;12(11):1398–403.
[15] Furao S, Hasegawa O. A fast no search fractal image coding method. Signal Process Image Commun 2004;19(5):393–404.
[16] Wang X-Y, Wang Y-X, Yun J-J. An improved no-search fractal image coding method based on a fitting plane. Image Vis Comput 2010;28(8):1303–8.
[17] Golberg DE. Genetic algorithms in search, optimization, and machine learning, 1989. Addion Wesley; 1989. p. 102.
[18] Kennedy J. Particle swarm optimization. In: Encyclopedia of machine learning. Springer; 2011. p. 760–6.
[19] Zhao Y, Yuan B. A new affine transformation: its theory and application to image coding. IEEE Trans Circuits Syst Video Technol 1998;8(3):269–74.
[20] Lu J, Ye Z, Zou Y. Huber fractal image coding based on a fitting plane. IEEE Trans Image Process 2013;22(1):134–45.
[21] Wohlberg B, De Jager G. A review of the fractal image coding literature. IEEE Trans Image Process 1999;8(12):1716–29.
[22] Lazar MS, Bruton LT. Fractal block coding of digital video. IEEE Trans Circuits Syst Video Technol 1994;4(3):297–308.
[23] Fisher Y, Menlove S. Fractal encoding with HV partitions. In: Fractal image compression. Springer; 1995. p. 119–36.
[24] Jacquin AE. Fractal image coding: a review. Proc IEEE 1993;81(10):1451–65.
[25] Weber G. Usc-sipi image database: version 4, Dept Elect Eng-Syst. Tech Rep. Univ Southern California, Los Angeles, CA, USA; 1993.
[26] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 2004;13(4):600–12.
[27] Yates F. Contingency tables involving small numbers and the χ 2 test. Suppl J R Stat Soc 1934;1(2):217–35.
[28] Bhattachayya A. On a measure of divergence between two statistical population defined by their population distributions. Bull Calcutta Math Soc 1943;35:99–109.
[29] Kullback S, Leibler RA. On information and sufficiency. Ann Math Stat 1951;22(1):79–86.
[30] Fractal image compression: theory and application. Fisher Y, editor. New York: Springer-Verlag; 1994. 0-387-94211-4.
[31] Polvere M, Nappi M. Speed-up in fractal image coding: comparison of methods. IEEE Trans Image Process 2000;9(6):1002–9.
[32] Saupe D. Fractal image compression via nearest neighbor search. Univ., Inst. für Informatik; 1996.
[33] Wu M-S. Genetic algorithm based on discrete wavelet transformation for fractal image compression. J Vis Commun Image Represent 2014;25(8):1835–41.
[34] Skodras A, Christopoulos C, Ebrahimi T. The JPEG 2000 still image compression standard. IEEE Signal Process Mag 2001;18(5):36–58. doi:10.1109/79.952804.