Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 689602, 8 pages
http://dx.doi.org/10.1155/2013/689602

Research Article

Image Retrieval Based on Fractal Dictionary Parameters

Yuanyuan Sun,[1] Rudan Xu,[1] Lina Chen,[2] and Xiaopeng Hu[1]

[1] *College of Computer Science and Technology, Dalian University of Technology, Dalian 116042, China*
[2] *National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China*

Correspondence should be addressed to Yuanyuan Sun; syuan@dlut.edu.cn

Received 29 September 2013; Accepted 9 December 2013

Academic Editor: Hai Yu

Content-based image retrieval is a branch of computer vision. It is important for efficient management of a visual database. In most cases, image retrieval is based on image compression. In this paper, we use a fractal dictionary to encode images. Based on this technique, we propose a set of statistical indices for efficient image retrieval. Experimental results on a database of 416 texture images indicate that the proposed method provides a competitive retrieval rate, compared to the existing methods.

1. Introduction

With the popularity of the computer and the rapid development of multimedia technology, image information is growing rapidly. How to effectively manage these resources has become a focus of many scholars' study. The early image retrieval technology is based on text, which relies on manual work that it could hardly meet the users' needs. Subsequently, a concept of content-based image retrieval (CBIR) is proposed. A new time has come that image management system can analyze image and extract features automatically. Nowadays, CBIR is used in many techniques, including fractal-based image retrieval.

Fractal image coding is based on approximating an image by an attractor of a set of affine transformations [3]. To a certain extent, fractal codes reflect spatial relationships between regions, which can describe image content. Sloan [4] first proposed fractal codes based on CBIR. Zhang et al. [5] presented an approach to texture-based image retrieval that determines image similarity on the basis of matching fractal codes. Pi et al. [3] proposed four statistical indices utilizing histograms of range block mean and contrast scaling parameters. Huang et al. [2] used a new statistical method based on kernel density estimation.

All of the aforementioned retrieval indices are based on techniques which are similar to traditional fractal coding and are generated by image self-similarity. In this paper, we propose a retrieval method, regarding a shared dictionary as a medium and using the similarity between images and the dictionary as retrieval indices. All the data obtained in the experiments reflect the differences between query image and the dictionary. The remainder of the paper is organized as follows. Section 2 introduces the fractal image coding based on the fractal dictionary. The proposed indices and retrieval method are described in Section 3. Experiment results are reported in Section 4, which is followed by the conclusion.

2. Fractal Image Coding Based on M-J Set

Images can be viewed as vectors and can be encoded by a set of transforms. Usually, the transform can be generated by collage theorem [6]. In this theorem, a suitable transform is constructed as a "collage," and the "collage error" is represented as the distance between the collage and the image. In the traditional fractal encoding, devised by Jacquin, image is titled by "range blocks," each of which is mapped from one of the "domain blocks" as depicted in Figure 1; the combined mappings constitute transforms on the image as a whole [6].

Encoding image needs a suitable transform minimizing the collage error for each range block, which requires recoding blocks mapping parameters, such as contrast scaling s and luminance offset g. These parameters are applied on (1) for image reconstruction. In (1) I_j is defined as the image at jth iteration. I_0, the initial image, can be an arbitrary image with the same size of encoding image. Consider

$$I_j = s \times I_{j-1} + g. \quad (1)$$

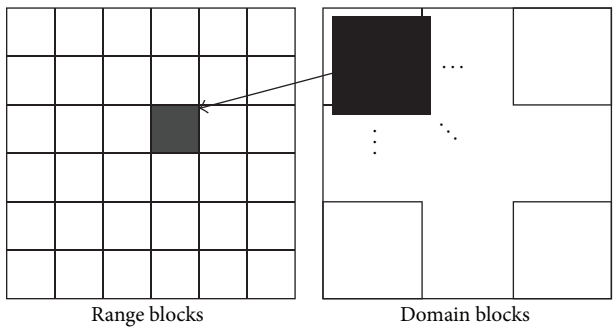

Figure 1: One mapping from a domain block to range block.

$$\begin{pmatrix} 245 & 239 & 249 & 239 \\ 245 & 245 & 239 & 235 \\ 245 & 245 & 245 & 245 \\ 245 & 235 & 235 & 239 \end{pmatrix}$$

Figure 2: Pixel matrix of a block.

2.1. Block Truncation Coding. Delp and Mitchell [7] presented a block truncation coding (BTC) scheme for image compression. It uses a two-level nonparametric quantizer that adapts to local properties of the image. In this scheme, an original image is first divided into nonoverlapping square blocks of size $N \times N$. For each pixel in a block, if its value is greater than the block mean (x_{ave}), the point is marked as 1, otherwise as 0, forming a two-value matrix consisting of 1 and 0. The equation is defined as follows:

$$\widehat{x} = \begin{cases} 1, & x \geq x_{\text{ave}}, \\ 0, & x < x_{\text{ave.}} \end{cases} \tag{2}$$

Then the matrix is reshaped to a $1 \times N^2$ row vector to generate a binary sequence. Finally, the corresponding decimal, defined as BTC value, is recorded.

Figure 2 shows a block after an original image is divided. Its x_{ave} is 241.875. There are 9 nodes bigger than x_{ave}.

The two-value matrix is shown in Figure 3. In the matrix, nine nodes are noted as 1 and the others as 0. The binary sequence is 1010 1100 1111 1000, so the BTC value is 44272.

2.2. Fractal Dictionary Based on M-J Set. In fractal decoding, the initial image and the final image have no direct relationship. Therefore, we compress enough domain blocks into a file as a dictionary. An image is encoded by finding best-matching domain blocks in the dictionary and decoded like traditional decoding process, by affine transform on these best-matching domain blocks.

Mandelbrot set (abbreviated as M set) and Julia set (abbreviated as J set) are the classical sets in fractal study. They both contain abundant information. Each point in the M set corresponds to different parameters for the J set construction. The structures of J sets have self-similarity and infinity, which are rich enough to present an image. We use it for a dictionary [8]. The process is as follows.

$$\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3: Bit matrix of the block in Figure 2.

Step 1. Choose parameters for a J set. We use a standard M set's boundary points as generation parameters for J set.

Step 2. Generate the J sets. We use the above parameters to generate J sets. According to the time-escaped algorithm [9], points in J sets are represented as the escape time, which must be satisfied with the following equation:

$$V_{(k,l)} \leq \text{Max\_Iterative}, \quad 0 \leq k < M, \ 0 \leq l < M, \tag{3}$$

where Max_Iterative is the max escape time.

Step 3. Quantize the image of J set. The values of the pixels assigned as the escape time are relatively small. It is better to multiply them by an expansion number as follows, so that the pixel values are between 0 and 255 equally:

$$\widehat{V}_{k,l} = \left(V_{k,l} \times H\right) \mod 256. \tag{4}$$

Note that H is the expansion number.

Step 4. Classify the domain block. The J set from Step 3 is divided into $K \times K$ nonoverlapping blocks. We calculate the BTC value of each block and use it as a classifier. If a block is the same as one in the BTC queue or if the queue has v blocks, we ignore this block. Otherwise, we compute the collage error between the calculating block and each one in the queue. If the collage error is less than a threshold, this block would be ignored, otherwise, added to the queue.

Step 5. Output the dictionary. All the blocks are written into a file by ascending BTC. We call this file dictionary.

Like traditional fractal encoding, an original image is first divided into nonoverlapping range blocks of size $B \times B$. For each range, we search a best-matching block with the smallest norm in the BTC queue after its BTC value is calculated. Finally, we get the parameters of each range block: one BTC value (btc_i), matching block number (T_i), contrast scaling parameter (s_i), luminance offset (g_i), and affine transformation (Γ_i).

In the decoding process, we use the (btc_i, T_i) as an index to locate the best-matching block. The original image can be decoded as follows:

$$R_i = s_i \times D_i + g_i, \tag{5}$$

where D_i is the domain block in the dictionary after affine transformation.

3. The Proposed Indices

It has been demonstrated in the literature that a grayscale histogram (color histogram) provides good indexing and

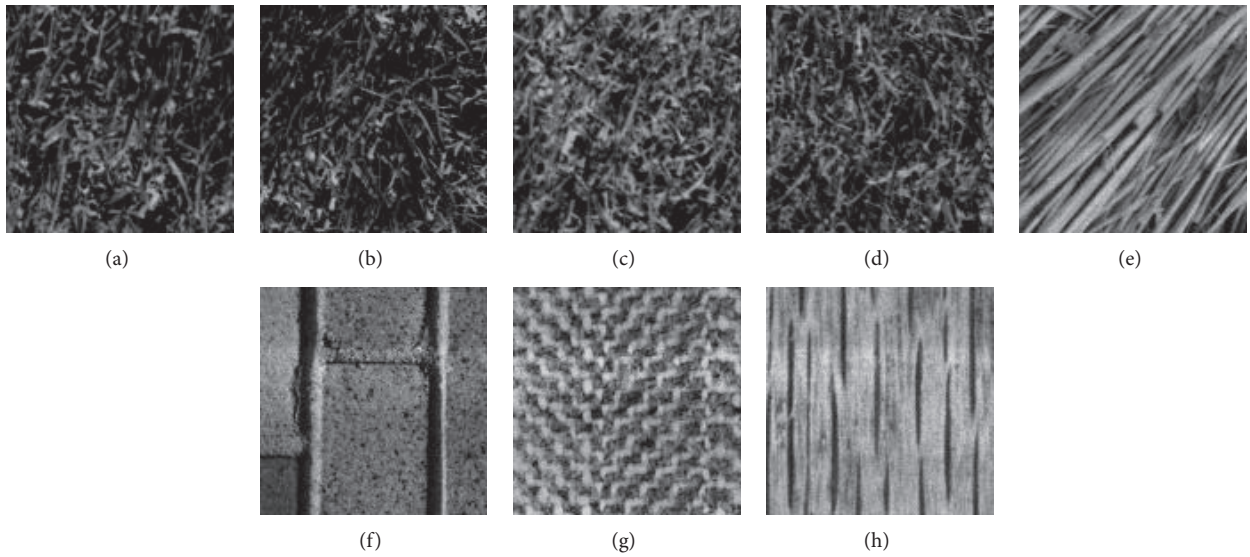

Figure 4: Example of 128 × 128 texture images. (a)–(d) Four similar images. (e)–(h) Four different texture images.

retrieval performance while being computationally inexpensive [10]. However, it is still a coarse feature when applied in image retrieving systems. In the existing method, we know that the histogram of fractal coding for image retrieval is effective [2]. In this paper, we use the following indices in the retrieving system.

3.1. Dictionary of Collage Error (DE). Collage error is the real-value of the distance between the range block and best-matching domain block. The smaller it is, the closer the decoded image is to the original image. Consider

$$E(R) = \frac{\sqrt{\|R - sD - gU\|^2}}{B \times B}. \tag{6}$$

In (6), U is a matrix whose elements are all ones. As (6) shows, it can also demonstrate the distance between the original image and dictionary. So, the distribution of collage error can be used as a parameter to classify texture images.

We quantize collage error to an integer interval (K). It is rounded to a nearest integer when it is smaller than K, or it is cut into K if it is bigger than K. In this paper, K is 13.

Figure 5 shows that similar texture images share almost the same distributions, while differing from different texture images. Hence, the distance between the same texture images is smaller than the different ones.

3.2. Dictionary of BTC (DB). The domain blocks in the dictionary are classified by BTC value as a category. So BTC value can also be treated as an index when we search a best-matching block. An image has a feature on BTC distribution (DB) and can be a scale in the image retrieving system.

In this paper, DB is a quantized value ranging from 0 to 15. We calculate the DB of images in Figure 4.

Figure 6 shows that the DB of four similar images, Figures 6(a)–6(d), are distributed similarly while Figures 6(e)–6(h) are not. Based on the above observation, we choose the DB as an image index. However, experimental results prove that it is only a coarse parameter, which will be discussed in Section 4.

3.3. Joint of Dictionary of BTC and S (JDBS). Schouten and De Zeeuw [11] have proved that contrast scaling parameters (s) in fractal coding can be used in retrieving images. But it is still a rough feature, just as DE and DB. Combing BTC with s, we present a 2D joint histogram with its character expressed in Figure 7.

Note that num_i is a sum number when $S = s_i$, BTC = btc_i. Then, we get the result of JDBS shown in Figure 8.

Figure 8 shows peaks of the same texture images coordinated roughly near while far in different texture images visually. We believe that JDBS are more precise than the above indices in retrieving images.

3.4. Similarity Measurement. To measure the similarity between two images, we choose L_2 as distance metric, which is expressed as follows:

$$d_{L2}(Q, V) = \sqrt{\sum_{i=1}^{n} (q_i - v_i)^2}, \tag{7}$$

where $\{q_1, \ldots, q_n\}$ and $\{v_1, \ldots, v_n\}$ are our proposed indices of query image and candidate images, respectively. The distances corresponding to DE, DB, and JDBS for images are listed in Table 1. The query image is image (a) in Figure 4.

It can be observed that the corresponding indices are roughly close for similar texture images and different for different texture images. However, (d)((*a*), (c)) is bigger than (d)((a), (f)) and (d)((a), (h)) on DE. In fact, (a) and (c) share similar texture, while (a), (f) and (h) are different texture images. This causes an unexpected result that image (f) and image (h) are retrieved, while image (c) is lost when the query image is (a). Hence, DE is only a coarse index.

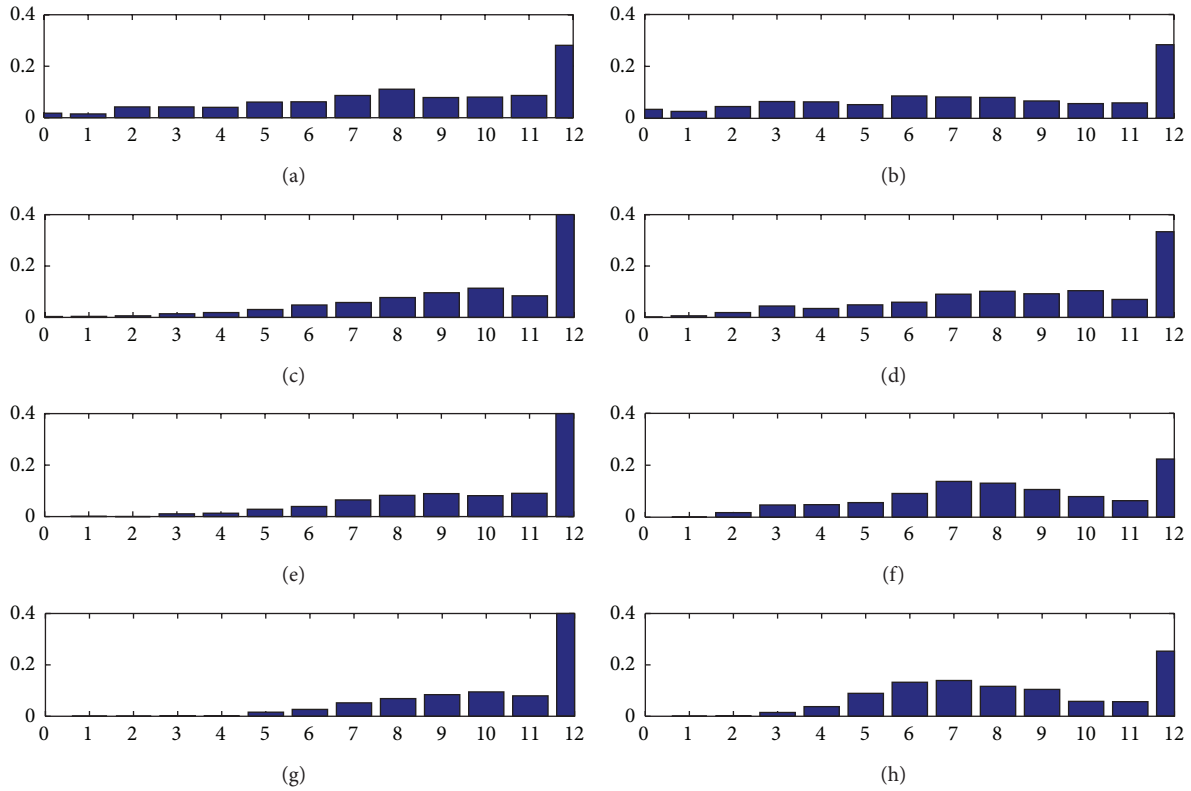

Figure 5: Dictionary of collage error corresponding to the eight texture images in Figure 4.

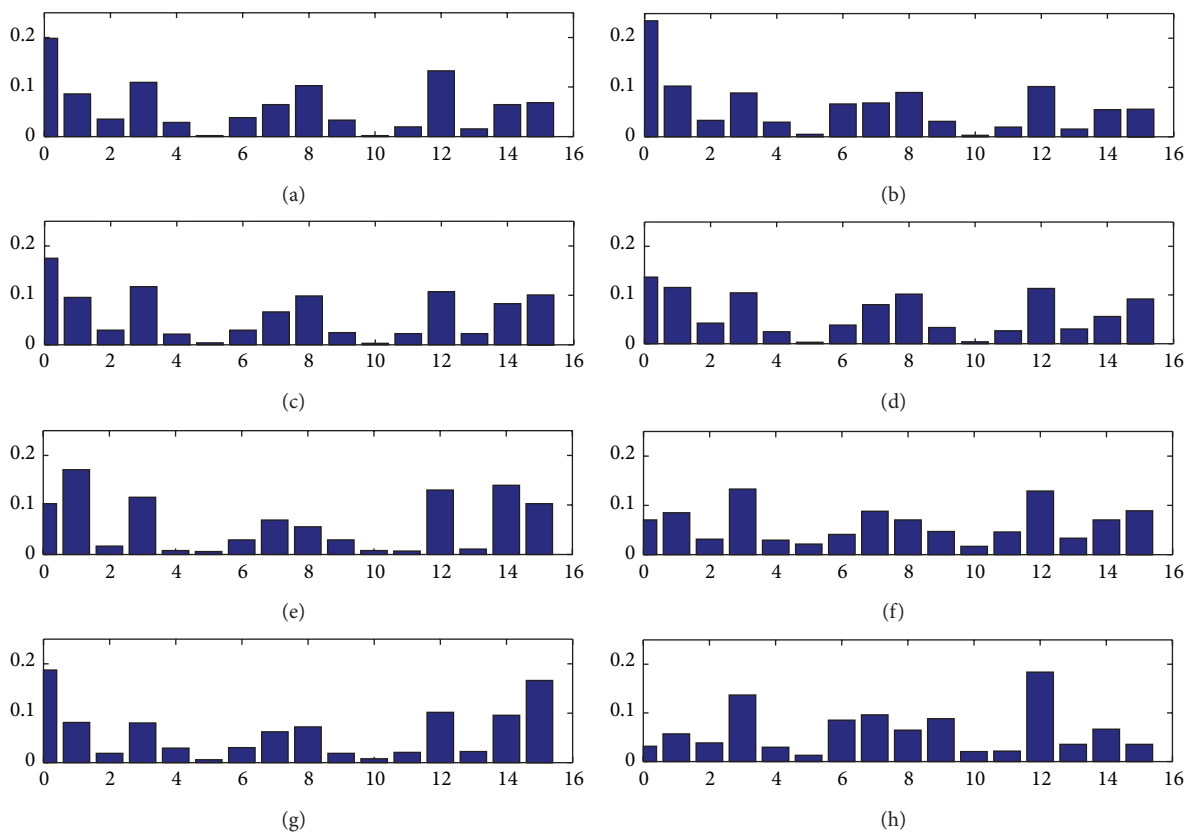

Figure 6: Dictionary of BTC corresponding to the eight texture images in Figure 4. BTC is quantized to 16.

Table 1: Distance between query image and candidate images.

Index	(d)((a), (b))	(d)((a), (c))	(d)((a), (d))	(d)((a), (e))	(d)((a), (f))	(d)((a), (g))	(d)((a), (h))
DE	0.0667	0.1885	0.0698	0.2359	0.0991	0.3154	0.1199
DB	0.6025	0.4963	0.5492	1.3222	1.3858	1.1931	2.5593
JDBS	0.0569	0.0577	0.0550	0.1185	0.1141	0.0941	0.1491

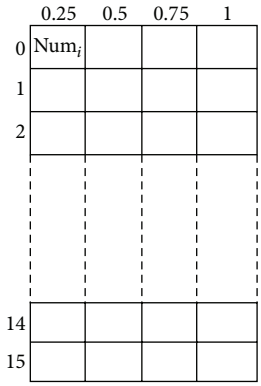

Figure 7: 2D joint structure of BTC and s, where BTC and s are quantized to 16 and 4, respectively.

Compared with DB and JDBS, the distance of DB between similar textured images has changed at 0.1; the JDBS changes at 0.001, and the distance between dissimilar texture images changes irregularly. Note that JDBS is more accurate.

3.5. The Operation Process. The whole operation process includes three parts: encoding image, extracting statistical features, and comparing their feature distances as shown in (7). The pseudocode is listed as in Figure 9.

4. Performance Evaluation

In this section, we present the performance of the proposed indices and compare our method with the literature's methods. The image retrieval system is shown in Figure 10. The test database is composed of 26 512 × 512 grayscale Brodatz texture images [12] and each image is separated into 16 subimages of size 128 × 128. Each sub-image is encoded based on *M-J* set fractal dictionary. A query image is randomly selected from the test database. All the sixteen retrieved subimages are selected based on the smallest distance criterion. In this paper, we use indices' length to evaluate the computational complexity.

Unfortunately, all the proposed indices have some inherent flaws. Although JDBS does better than DB, its vector length is longer than DB's, so it takes more computational complexity than that of DB. In order to reduce the complexity, we divide a retrieving index into several parts. For instance, we can replace the JDBS with the method of DS + DB. A list of candidate images is selected by matching DS + DB. Also, we can combine two indices. Let DE and DS be a 2D joint statistic of JDSE, then the performance will be enhanced. On the other hand, the computational complexity must be taken into consideration.

Table 2: Average retrieval rate of different retrieval methods.

Retrieval method	Length of feature vector	ARR (%)
HM [1]	16	42.01
JHMS [1]	64 × 4 = 256	50.11
HS + HM [1]	16 × 4 = 20	60.94
HM + HS + HE [1]	16 + 4 + 13 = 33	69.74
JHSE + HM [1]	16 + 4 × 13 = 68	64.95
KS [2]	11	39.47
KM [2]	11	32.84
KE [2]	11	40.10
KS + KM [2]	22	38.15
KS + KM + KE [2]	33	55.36
DB	16	60.86
JDBS	16 × 4 = 64	71.02
DS + DB	4 + 16 = 20	70.12
DS + DE + DB	13 + 4 + 16 = 33	75.20
JDSE + DB	13 × 4 + 16 = 68	78.67
DB	32	67.05
JDBS	32 × 4 = 128	73.68
DB + DS	32 + 4 = 36	70.36
DB + DS + DE	32 + 4 + 13 = 49	75.02
JDSE + DB	13 × 4 + 32 = 84	79.18

4.1. Average Retrieval Rate. Usually, average retrieval rate (ARR), as follows, can evaluate a technique's performance:

$$\mathrm{ARR} = \frac{\sum_{z=1}^{Z} m_z}{F \times Z}. \tag{8}$$

Note that F is denoted as the number of retrieved images, m_z is denoted as the number of correctly retrieved images at zth test and Z is the number of subimages in the test database. In this case, $F = 16$ and $Z = 416$. All experiments shown in Table 2 were conducted on a Core(TM) i5(2.40 GHz) PC. All data shown in Table 2, was acquired by the experiments.

Compared to HM and KM, DB technique has a better performance. The average retrieval rate of DB (for 16 vector length) is 60.86%, while the HM and the KM provide 42.01% and 32.84% average retrieval rate at similar vector length. If the DB is quantized to 32, its performance goes up by 6.19%. At the same time, the performance of DS + DB goes up to 70.12% average retrieval rate, which is 9.18% higher than that of HS + HM and 31.97% higher than that of KS + KM, while their vector lengths are all around 21.

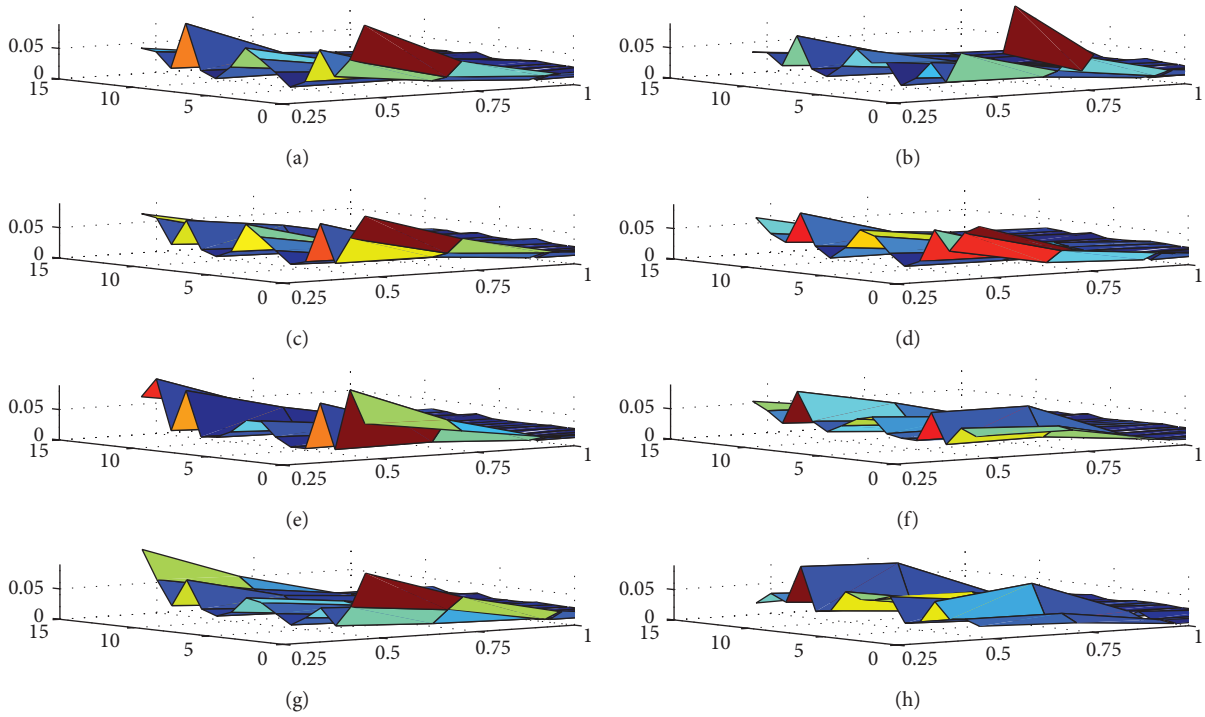

Figure 8: 2D joint histogram of BTC and s corresponding to the eight texture images in Figure 4.

```
/**** Part I: encode the image *****/
imA=ReadImage();
//read the fractal dictionary
dictionary=ReadFractalDictioanry();
//recode btc,index,s,o,t,e
fractalCode=GetFractalCode(imA,dictionary);

/*** Part II: extract statistical features ***/
//extract the statistical features of btc, btc-s, s, e
GetImAFeatures();

/**** Part III: compare distances *****/
//Compare these features with the features of images
//in the database, and get their distances. Images with
//top 16 smallest distances are the retrieval result.
[d1, d2,..., d416] = Compare(imA feautre, imB feautre);
```

Figure 9: The pseudocode of the whole operation process.

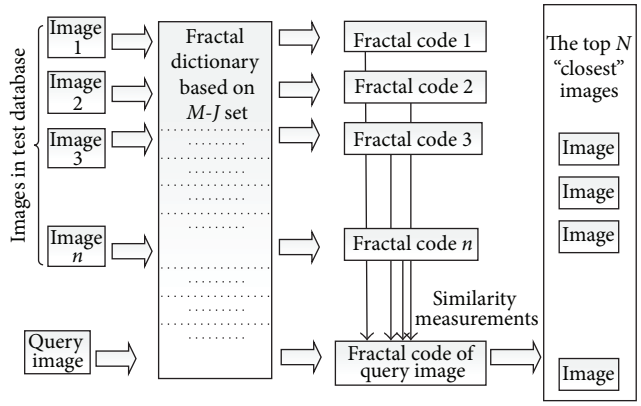

Figure 10: The image retriveing system based on M-J set fractal dictionary.

Compared with DS + DE + DB, JDSE + DB technique works well in average retrieval rate. When the DB is at 32 length, the total length of JDSE + DB is at 84, which is not much longer than 68 vector length of JDSE + DB and JHSE + HM. What more, due to the simple operations, the vector length does not impact on the computing complexity so much when the length is not too long. The time consumption in retrieving is less than 0.3 s in the experiments, so its computing complexity is tolerated. When JDSE + DB is at 68 and 84, the average retrieval rates reach 78.67% and 79.18%, respectively.

4.2. Precision against Recall Curve. Precision against recall curve is another method of evaluating retrieving performance. The higher both precision and recall are, the better

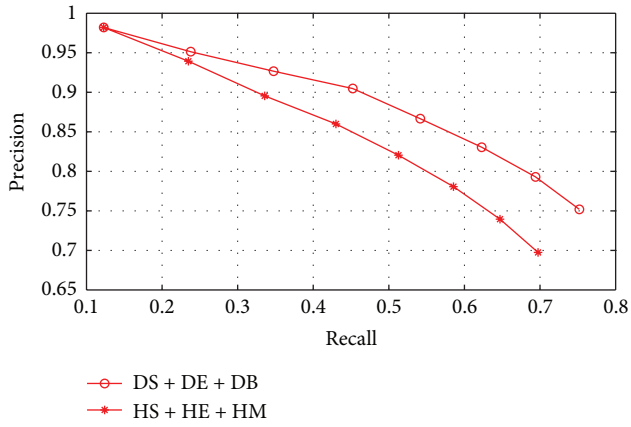

Figure 11: Precision against recall curve of HS + HE + HM and DS + DE + DB.

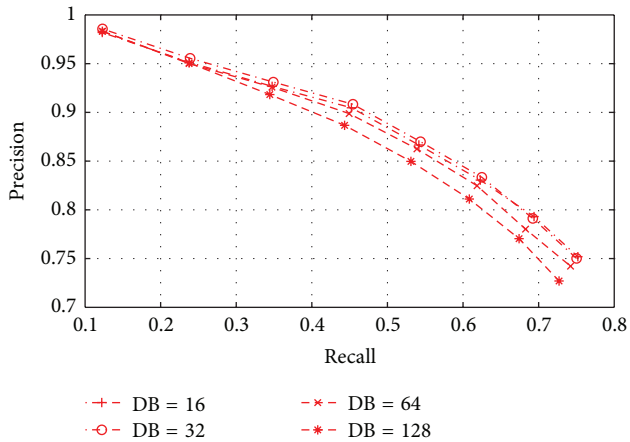

Figure 12: Precision against recall curve of DS + DE + DB at different DB quantizations.

the technique performs. The precision and recall are denoted as follows:

$$\text{Precision} = \frac{|\text{Retrieved} \cap \text{Relevant}|}{\text{Retrieved}},$$
$$\text{Recall} = \frac{|\text{Retrieved} \cap \text{Relevant}|}{\text{Relevant}}. \quad (9)$$

Note that retrieved is a set of retrieved images for a query and relevant is a set of relevant images for the query images [13] (in this case, relevant = 16). Based on top 2, 4, 6, 8, 10, 12, 14, and 16, we calculate the average precision and average recall of both HS + HE + HM and DS + DE + DB at 33 vector length.

The average precision (or recall) of DS + DE + DB is obviously higher than the average precision (or recall) of HS + HE + HM, when vector lengths are the same (see Figure 11). Besides, DS + DE + DB's slope varies slightly, and this implies that DS + DE + DB has a better performance.

Figure 12 shows precision against recall curve of DS+DE+DB at 16, 32, 64, and 128 DB vector lengths.

To some extent, the curve varies greatly with the quantization levels of DB (see Figure 12). However, when the quantization level is in excess of 16, the curve changes slightly; on the other hand, the computation becomes more complex. Figure 13 shows that the average retrieval rates change significantly as the quantization levels of DB increase. When the level reaches 16, the curve gets to its peak. After 16, the distribution of DB becomes too detailed, which makes the DE + DS + DB lose statistical characteristics when DE is at thirteen vector length and DS is at four vector length, so the curve falls down and the average retrieval rates decrease. That is to say, when the vector length is 16, the performance and complexity can achieve a balance.

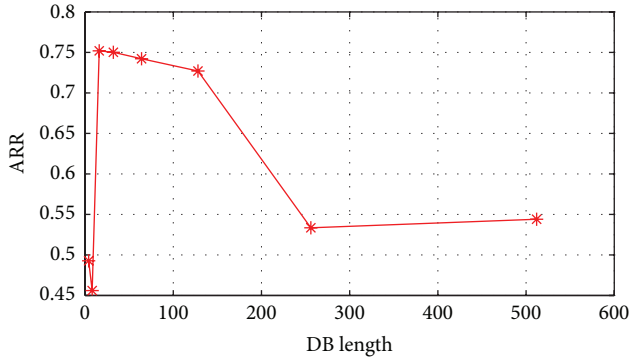

Figure 13: Average retrieval rate of DS + DE + DB where DB quantizations are 4, 8, 16, 32, 64, 128, 256, and 512.

5. Conclusions

In this paper, we have proposed a set of indices based on *M-J* fractal dictionary encoding. *M-J* fractal dictionary is a shared file composed of blocks of Julia set with BTC ascending. We proposed that DE, DB, and JDBS indices are close for similar texture images and different for different texture images. Subsequently, we calculated average retrieval rate, average precision, and average recall and compare with previous methods. We discussed further minimizing of the vector length of DS + DE + DB without a big loss of retrieval rate and gave the optimal length of the feature vector.

Experimental results on a database of 416 texture images showed that the proposed indices provided better performance than the previous methods. Also, DE + JDBS provided a 79.18% average retrieval rate at the maximum, and its computational complexity was tolerated. In addition, JDSE + DB and DS + DE + DB not only had low computational complexity, but also provided competitive retrieval rate, compared to existing methods.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (nos. 61103147, 61075018, 61070098, and 61272523), the National Key Project of Science and Technology of China (no. 2011ZX05039-003-4), and the Fundamental Research Funds for the Central Universities (no. DUT12JB06).

References

[1] M. Pi and H. Li, "Fractal indexing with the joint statistical properties and its application in texture image retrieval," *IET Image Processing*, vol. 2, no. 4, pp. 218–230, 2008.

[2] X. Huang, Q. Zhang, and W. Liu, "A new method for image retrieval based on analyzing fractal coding characters," *Journal of Visual Communication and Image Representation*, vol. 24, pp. 42–47, 2012.

[3] M. Pi, M. K. Mandal, and A. Basu, "Image retrieval based on histogram of fractal parameters," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 597–605, 2005.

[4] A. D. Sloan, "Retrieving database contents by image recognition: newfractal power," *Advanced Imaging*, vol. 9, no. 5, pp. 26–30, 1994.

[5] A. Zhang, B. Cheng, and R. S. Acharya, "Approach to query-by-texture in image database systems," in *Proceedings of the Digital Image Storage and Archiving Systems*, vol. 2606, pp. 338–349, October 1995.

[6] B. Wohlberg and G. De Jager, "A review of the fractal image coding literature," *IEEE Transactions on Image Processing*, vol. 8, no. 12, pp. 1716–1729, 1999.

[7] E. J. Delp and O. R. Mitchell, "Image compression using block trancation coding," *IEEE transactions on communications systems*, vol. 27, no. 9, pp. 1335–1342, 1979.

[8] Y. Y. Sun and R. Q. Kong, "The research on the image encryption and coding algorithms based on M-J fractal sets," *Computer Engineering*, vol. 39, pp. 230–233, 2013.

[9] B. B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman and Company, New York, NY, USA, 1983.

[10] B. M. Mehtre, M. S. Kankanhalli, A. D. Narasimhalu, and G. C. Man, "Color matching for image retrieval," *Pattern Recognition Letters*, vol. 16, no. 3, pp. 325–331, 1995.

[11] B. A. M. Schouten and P. M. De Zeeuw, "Image databases, scale and fractal transforms," in *Proceedings of the International Conference on Image Processing (ICIP '00)*, pp. 534–537, September 2000.

[12] www.cipr.rpi.edu/resource/stills/brodatz.html.

[13] T. Yokoyama, T. Watanabe, and H. Koga, "Similarity-based retrieval method for fractal coded images in the compressed data domain," *Image and Video Retrieval*, vol. 3568, pp. 385–394, 2005.