J. Vis. Commun. Image R. 25 (2014) 1102–1111

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

An improved differential box-counting method to estimate fractal dimensions of gray-level images

Yu Liu, Lingyu Chen, Heming Wang, Lanlan Jiang, Yi Zhang, Jiafei Zhao, Dayong Wang, Yuechao Zhao, Yongchen Song *

Key Laboratory of Ocean Energy Utilization and Energy Conservation of Ministry of Education, Dalian University of Technology, Dalian 116024, PR China

ARTICLE INFO

Article history:
Received 22 May 2013
Accepted 22 March 2014
Available online 1 April 2014

Keywords:
Differential box-counting method (DBC)
Fractal dimension
Gray-level images
Grid box size
Synthetic images
Over-counting
Texture images
Under-counting

ABSTRACT

The differential box-counting (DBC) method is one of the frequently used techniques to estimate the fractal dimension (FD) of a 2D gray-level image. This paper presents an improved DBC method based on the original one for improvement of the accuracy. By adopting the modifying box-counting mechanism, shifting box blocks in (x, y) plane and selecting appropriate grid box sizes, it can solve the two kinds of problems which the DBC has: over-counting boxes along z direction and under-counting boxes just at the border of two neighboring box blocks where there is a sharp gray-level abruption exits. The experiments using two sets of synthetic images and one set of real natural texture images demonstrate that the improved DBC method can solve the two kinds of problems perfectly, simultaneously, and can outperform other DBC methods in the accuracy.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Most of the objects in the real world are so complex and irregular in nature that they cannot be described by classical Euclidean geometry [1–4]. Fractals, initially developed by Mandelbrot [5], are mathematical sets with high degrees of geometrical complexity that can model many natural phenomena. In fractal theory, a fractal is defined as a set for which the Hausdorff–Besicovich dimension (simply, Hausdorff dimension) is greater than the topological (Euclidean) dimension [1,5]. One of the fundamental properties of a fractal is its self-similarity. The fractal dimension (FD) is a useful numerical quantity for expressing the self-similarity of a fractal. Quite a few different methods have been proposed to estimate the FD. Pentland [6] proposed a method of estimating FD by using the Fourier power spectrum of image intensity surfaces. Peleg et al. [7] adopted Mandelbrot's idea of the ε-blanket method and extended it to surface area calculation. Clarke [8] used the concept of triangular prism surface area to compute the FD. Gangepain and Roques-Carmes [9] developed the popular reticular cell-counting method which improved upon by Voss [10] by incorporating probability theory. Later, Keller et al. [11,12] contributed a further refinement by a way of linear interpolation. Lopes and Betrouni [13] summarized and classified those methods into three major categories: the box-counting (BC) methods, the fractional Brownian motion (fBm) methods and the area measurement methods.

The box-counting dimension calculated by the box-counting methods is the most frequently used method for measurements in various application fields. The reason for its dominance lies in its simplicity and automatic computability [14]. To the gray-level images, the differential box-counting (DBC) method proposed by Sarkar and Chaudhuri [15–17] was considered as a better one, as was also supported by the investigation in many literatures. It is a generalization of classical BC method to estimate the FD of gray-level images [18–22].

However, there are some drawbacks in the original DBC method. Both Chen et al. [1] and Li et al. [19] pointed the shortcomings of over-counting or under-counting the number of boxes. Later many modified or improved DBC methods have been proposed for those problems. Jin et al. [22] described a relative differential box-counting (RDBC) method by using a practical algorithm with the upper and lower limits for estimating FD. They showed that their method is better with smaller distance error and covers the full dynamic range of FD values. Biswas et al. [21] proposed the modified implementation of DBC by adopting a parallel algorithm to make its computability more efficient. In the shifting differential box-counting (SDBC) method by Chen et al. [1], the concept of shift operation is similar with that in RDBC. However, the authors also pointed that the SDBC cannot work to

* Corresponding author. Fax: +86 41184708015.
E-mail address: songyc@dlut.edu.cn (Y. Song).

http://dx.doi.org/10.1016/j.jvcir.2014.03.008
1047-3203/© 2014 Elsevier Inc. All rights reserved.

the images in which a sharp gray-scaled variation exists at the border of the neighboring box blocks. Li et al. [19] proposed an improved DBC. Three modifications were carried out for better performance of estimating FD in their method. But the images with a sharp gray-scaled variation were not used in their experiments.

Base on the above summary, we can find that, so far, there is no one improved DBC algorithm which can deal with the two problems of over-counting and under-counting the number of boxes, simultaneously. According to our experience on the analysis of gray level images, the two drawbacks in the DBC method may provide unreasonable FD values which may be less than two, particularly to the images with a sharp gray-scaled variation at the border of the neighboring box blocks. Therefore, for computing FD more accurate, it will be necessary and significant meaningful to improve a method which can deal with the two problems, simultaneously. In this paper, an improved DBC method has been proposed to improve the accuracy of estimating FD. In our improved method, the modifying box-counting mechanism, shifting box blocks in (x, y) plane and selecting appropriate grid box sizes have been implemented. The experiments results using the two set of synthesized images and one set of texture images demonstrated the advantages of our method.

The remainder of this research is organized as follows. Section 2 is the basic definition of FD and DBC method. The drawbacks of DBC and two kinds of problems are discussed as well. Section 3 describes our improved method with three improvements, and the procedure. Section 4 gives results and discussions of the three experiments. Section 5 draws the conclusions.

2. Basic definition and different DBC approaches

2.1. Basic fractal dimension definition

Mandelbrot [5] defined a fractal as a bounded set A in R for which the Hausdorff–Besicovich dimension is strictly larger than the topological dimension. Consider a bounded set A in Euclidean n-space. The set is said to be self-similar if A is the union of $N_r(A)$ distinct (non-overlapping) subsets, each of which is similar to A scaled down by a ratio r. Thanks to the celebrated box-counting theorem [23], fractal dimension, D, of A can be rewritten in the form

$$1 = N_r(A) \cdot r^D \quad \text{or} \quad D = \frac{\ln N_r(A)}{\ln(1/r)}. \tag{1}$$

The D can only be calculated for deterministic fractals. For an object with deterministic self-similarity, its FD is equal to its box-counting dimension D_B. However, natural scenes practically do not exhibit deterministic self-similarity. Therefore, in general, it is hard to directly compute the exact value of D. Fortunately; those natural scenes exhibit some statistical self-similarity. The box-counting dimension D_B is an estimate of D, which is calculated by using a box-counting method. To calculate the FD of a gray scale image, the DBC method is the most frequently used algorithm which is introduced as follow.

2.2. DBC and drawbacks

The method is based on the reticular cell counting approach proposed by Gangepain and Roques-Carmes [9]. The reticular cell counting approach treats a box which contains at least one sample of gray-level intensity surface as a countable box. And then perform the least square linear fit of $\log N_L$ versus $\log L$ to obtain the D, where the L is the box size, N_L is the number of boxes needed to cover the whole gray-level intensity surface. Sarkar and Chaudhuri [16] adopted a differential method to count the N_L, so they call their method as the *differential box-counting* approach (DBC). Detail description is listed below.

Consider an image of size $M \times N$ as a three-dimensional (3D) spatial surface with (x, y) denoting pixel position on the image plane, and the third coordinate (z) denoting pixel gray-level. In the DBC method, the (x, y) plane is partitioned into non-overlapping blocks of size $s \times s$, where $M/2 \geqslant s \geqslant 2$ and s is an integer. Then let an estimate of $r = s/M$. On each block, there is a column of boxes of size $s \times s \times s'$, where s' is the height of each box, $G/s' = M/s$ and G is the total number of gray-levels. Assign numbers $1, 2, \ldots$ to the boxes as shown in Fig. 1. Let the minimum and maximum gray-level (write as $I_{\min}$ and $I_{\max}$) in the (i, j)th block fall in box number k and l, respectively. The boxes covering this block are counted in the number as

$$n_r(i,j) = l - k + 1, \tag{2}$$

where the subscript r denotes the result using the scale r.

For example, $s = s = 3$ and $n_r(i, j) = 3–1 + 1$ as illustrated in Fig. 1. Considering contributions from all blocks, N_r is counted for different values of r as

$$N_r = \sum n_r(i,j). \tag{3}$$

Then the FD can be estimated from the least squares linear fit of $\log(N_r)$ versus $\log(1/r)$.

Sarkar and Chaudhuri [16] compared their method with other four algorithms in their paper. They showed DBC have a lower computational complexity, best in terms of efficiency and dynamic range of FDs, but lower accuracy. However, some disadvantages of DBC have been discussed in other literatures [1,19]. They pointed that the DBC method has two major problems.

(1) Over-counting the number of boxes covering the image intensity surface. It may occur in the z direction and in the x or y directions as well, if the boxes do not shift appropriately along those directions.
(2) In contrast, under-counting the number of boxes may happen at the border of the neighboring box blocks where there is a sharp gray-scaled variation exists on the image intensity surface.

Because of the above two drawbacks, unreasonable results will be obtained when using the method to measure the following images.

(1) Images with the same roughness of the intensity surface but different gray-levels. For simplify, if a new image is generated for which the gray-level of each pixel in the original image is either increasing or decreasing by a constant value, then the FD of the new image should stay the same as that of the original image theoretically since they have the same degree of roughness. But these two estimates appear to be distinct when the DBC method is adopted.
(2) Images which have sharp gray-level abruption just at the border of two neighboring box blocks (see detailed description in Section 4.2). The estimated FDs by the DBC method may lie outside the range between 2.0 and 3.0. These results, of course, are not reasonable.

To solve those problems and compute FD efficiently and accurately, some modified or improved DBC methods based on the original one have been proposed. Jin et al. [22] described a relative differential box-counting (RDBC) method by using a practical algorithm and setting the upper and lower limits of the scale for estimating FD. N_r is counted in their method as follows.

$$N_r = \sum ceil[d_r(i,j)/s'], \tag{4}$$

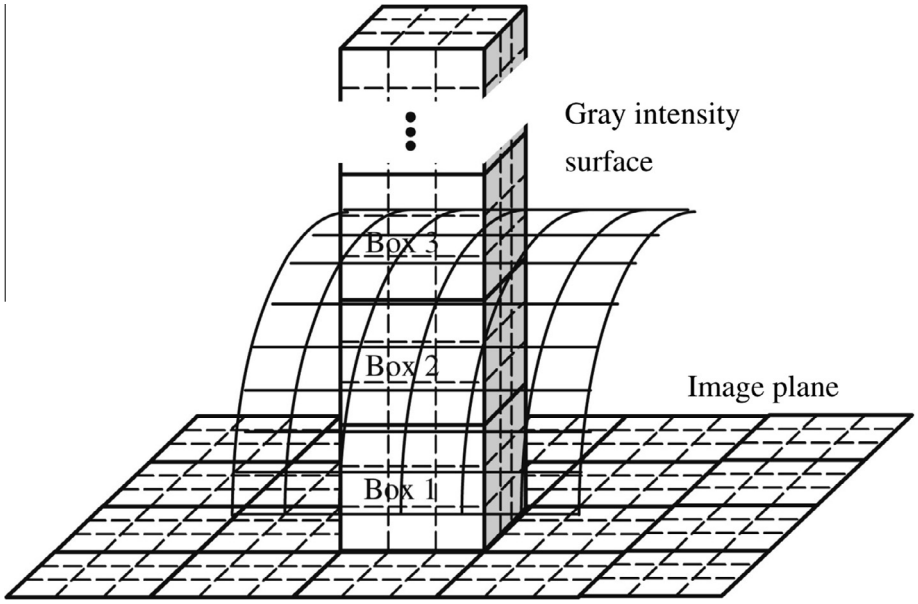

Fig. 1. Sketch of determination of the number of boxes (n_r) by DBC method (quoted from Sarkar and Chaudhuri [19]).

where $d_r(i,j)$ is defined as $d_r(i,j) = I_{\max} - I_{\min}$, and *ceil*[ ] denote the ceiling function.

Chen et al. [1] proposed the shifting differential box-counting (SDBC) method in 2003. In the SDBC method, the authors took measures to let the boxes shift along the z direction as well, i.e. the intervals in the z direction shift appropriately. Different with the RDBC, Chen's SDBC used below formula to count the $n_r(i,j)$. Then the N_r was calculated by using the formula (3) as well.

$$n_r(i,j) = ceil[(I_{\max} - I_{\min} + 1)/s']. \tag{5}$$

The authors pointed out that the SDBC algorithm achieves the estimated FD values closer to the exact values than the DBC does. But it cannot work to the images with a sharp gray-scaled variation exists just at the border of the neighboring box blocks. They proposed another method, the scanning box-counting (SBC) method, for solving this kind of problem instead.

In 2009, Li et al. [19] proposed their DBC method for calculating the FDs of the gray-level images. In their method, three modifications, box height selection, box number calculation and image intensity surface partition, were taken to overcome the drawbacks of DBC. Formula (6) was used to count n_r.

$$n_r(i,j) = \begin{cases} ceil\left(\frac{I_{\max} - I_{\min}}{s''}\right), & I_{\max} \neq I_{\min}, \\ 1, & I_{\max} = I_{\min}, \end{cases} \tag{6}$$

where the s'' is the modified box height.

In their paper, discuss the authors did not images with a sharp gray-scaled variation just at the border of two neighboring box blocks in their experiments. Although the method is accuracy to the images which have low FDs, it deviates from the theoretical FDs very much and generates great fit errors to those images have high FDs.

In our paper, the DBC, RDBC, SDBC and Li's improved DBC are all used to test the experiments and compare with our improved method.

3. Our method

According to the analysis of the two shortcomings, it is easy to find that the accuracy of the original DBC method is limited. Therefore, in the study, three modifications in our improved DBC method are proposed for improvement, which are *modifying box-counting mechanism* (i.e. *shifting boxes along z direction*), *shifting box blocks in* (x,y) *plane* and *selecting appropriate grid box sizes*. The purpose of our method is to solve all the two problems simultaneously and achieve high accuracy.

3.1. Modifying Box-counting mechanism

As we study, in the DBC method, the boxes are fixed on certain place in the space of intensity surface of image under a certain box height. This leads to over-count the number of boxes along z direction sometimes. For calculating the minimum number of boxes in a certain block, the boxes must shift along the z direction appropriately, RDBC uses the (4), SDBC uses the (5), and Li's DBC uses the (6). For more reasonable, in our method, n_r is defined by the (7). As we study, it is more reasonable and considerate. For example, if $I_{\min} = 0$, $I_{\max} = 5$, then there are 6 gray levels not 5 gray levels. Therefore, for covering the gray levels completely, the numerator in the (7) must be $I_{\max} - I_{\min} + 1$. Moreover, if $I_{\min} = I_{\max}$, only one box is need obviously. However, if $s' = sG/M$ is smaller than 1, the n_r will be larger than 1. So the n_r must be defined as one box for the condition of $I_{\min} = I_{\max}$.

$$n_r(i,j) = \begin{cases} ceil\left(\frac{I_{\max} - I_{\min} + 1}{s'}\right), & I_{\max} \neq I_{\min}, \\ 1, & I_{\max} = I_{\min}. \end{cases} \tag{7}$$

The physical meaning of (7) is that the boxes are assigned from the minimum gray-level in a certain (i,j)th block rather than gray-level 0. It is expected that $n_r(i,j)$ is the least number of boxes covering the surface of the (i,j)th block.

3.2. Shifting box blocks in (x, y) plane

This improvement is to solve the second problem. Because the under-counting will happen at the border of the neighboring box blocks where there is a sharp gray-scaled variation exists, the solution, used in our method, is shifting the box blocks in (x,y) plane with some pixels.

Consider an image of size $M \times N$ as a three-dimensional (3D) spatial surface with (x,y) denoting pixel position on the image plane, and the third coordinate (z) denoting pixel gray-level. In

our method, the (x, y) plane is partitioned into non-overlapping blocks of size $s \times s$ as well. On each block there is a column of boxes of size $s \times s \times s'$, where s' is the height of each box, $G/s' = M/s$ and G is the total number of gray-levels. Let the number of the needed boxes to cover the intensity surface in the (i, j)th block is counted as n_r_old. Then shift the block in the (x, y) plane with in δ pixels under the set rule. The number of boxes, marked as n_r_new, is afresh counted in the new block. Comparing n_r_old with n_r_new, the final adopted n_r is obtained by

$$n_r = \max(n_r\_old, n_r\_new). \tag{8}$$

Both n_r_old and n_r_new are counted by the (7).

After doing this, the under-counting the number of the boxes will be overcome if there is a sharp gray-scaled variation exists at the border of the neighboring box blocks, and the more accurate number of the needed boxes will be obtained. The total number of boxes N_r covering the full image surface at the scale s is computed with (3). At last, the FD can be estimated from the least squares linear fit of $\log(N_r)$ versus $\log(1/r)$. r is also defined as $r = s/M$.

The rule of shifting block in (x, y) plane is described here as illustrated in Fig. 2. Assume that the procedure starts scanning image from the top left corner. If the (i, j)th block is not at the borders of the (x, y) plane, the new block will be shifted to the $(i + \delta, j + \delta)$th position. The process can be represent as

$$(i,j) \overset{shift\ (\delta)}{\longrightarrow} (i+\delta, j+\delta). \tag{9}$$

When the block arrives at the boundary of the image, the shifting will be treated particularly. This is because when the blocks at boundary, shifting blocks under the regular rule will make the blocks out of the image, then the program will report errors. So we treat those blocks particularly and shift them in the other directions. The details are as following.

At the right boundary, $i < M$, $j = N$, the shifting is defined as

$$(i,j) \overset{shift\ (\delta)}{\longrightarrow} (i+\delta, j-\delta). \tag{10}$$

At the bottom, $i = M$, $j < N$,

$$(i,j \overset{shift\ (\delta)}{\longrightarrow} (i-\delta, j+\delta). \tag{11}$$

At the bottom right corner, $i = M$, $j = N$,

$$(i,j) \overset{shift\ (\delta)}{\longrightarrow} (i-\delta, j-\delta). \tag{12}$$

For example, in the Fig. 2, $s \times s = 3 \times 3$, $M \times N = 21 \times 21$, the gray scales are the original positions of the box block; the red panes are the new positions of the box block after shifting with 1 pixel ($\delta = 1$).

δ is the number of pixels for shifting a block along one direction. Because the pixels of a given image are integers, so δ must be integer as well. Moreover, δ is just for the program to catch the border of the neighboring box blocks, so the smallest value is perfect. Because large value will make the computational FD value unfaithful. That is why we choose δ as 1.

3.3. Selecting appropriate grid box sizes

It is worth noting that the choice of grid box sizes is critical. For digital images, M, G and s are finite integers. If the M cannot be partitioned by s completely, some partitions of size smaller than $s \times s$ will produce at the image margins. In such case, gray-levels in those partitions would be treated to be zero. With no doubt, the

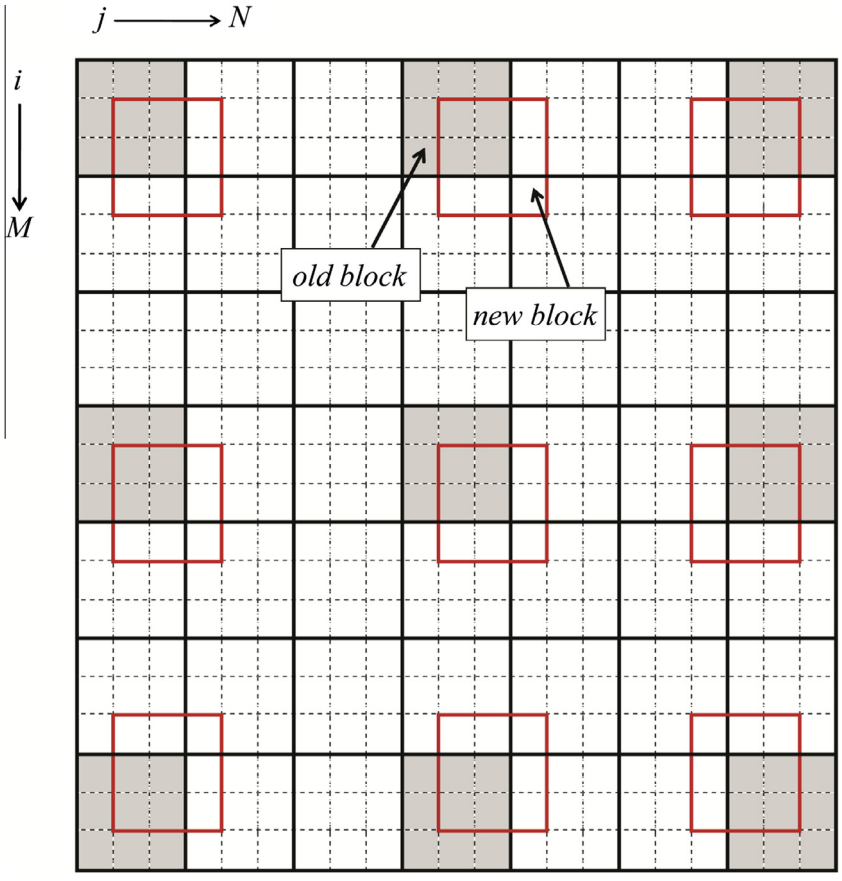

Fig. 2. Sketch of shifting the blocks in the (x, y) plane. $M \times N = 21 \times 21$, $s \times s = 3 \times 3$, $\delta = 1$.

results are not accurate. Sarkar and Chaudhuri [16] proposed a range of grid box sizes as $2 \leqslant s \leqslant M/2$ in their paper. Those lower border and higher borders are also used to test our method in the study. More detailed, the grid box sizes are selected as $s = 2^i$ (where i is an integer) in all methods for comparing their differences. The reason is that selection of grid box sizes as a power of 2 enables us to implement the programs in a more computationally efficient manner, which avoids scanning the whole image again and again. Moreover, those grid box sizes can partition the digital images completely, i.e. there are no partitions of size smaller than grid $s \times s$ at the image margins. Because the sizes of images used in this study are all 2^n (256, 512). If the size of an image is not a power of 2, the grid box sizes must be trained particularly in the program correspondingly. For example, in the Section 4.3, the image size is 640×640, so the grid box sizes should be selected as the divisors of 640, i.e. $s = [2, 4, 5, 8, 10, 16, 20, 32, 40, 64, 80, 128, 160, 320]$, or a power of 2 as $s = [2, 4, 8, 16, 32, 64, 128]$.

3.4. Procedure

Our procedure is listed in Table 1. It is implemented on the Matlab7.0 R14 (2004). It contains three major steps. (1) Selecting different grid box sizes to cover the image; (2) Counting the number of the boxes need to cover the object completely; (3) Performing the least-squares linear fit of $\log(N_r)$ against $\log(1/r)$ to obtain the FD by the value of the slope.

4. Experimental results

In the study, our method and other four differential box-counting algorithms have been compared through three experiments which have been performed on two sets of synthetic texture-like images, shown in Figs. 3 and 5, and one set of 16 natural textured images taken from Brodatz's image database [24], shown in Fig. 7. For each of these natural and synthetic textured images, all the gray-levels range from 0 to 255, i.e. the total gray-levels are 256.

4.1. Test on synthetic images I

In this test, a set of 16 natural textured images with the same resolution (256×256) has been synthesized, as shown in Fig. 3, in which each one is simply a copy of a seed image via 32 distinct gray-level increment. The upper-left image (i.e. image 1) is chosen as the seed image. The gray-level of each pixel of the seed image is shifted by increasing $(2k - 2)$, $1 \leqslant k \leqslant 16$, to generate the kth image in the 16 images set. It is worth noting that the maximum gray-level in the selected seed image is not bigger than 225. Therefore the resulting gray-levels in the generated images are still within the range of 0 to 255. Obviously, the minimum gray-levels in those images will be larger than or equal to 30.

After doing this, all intensity surfaces of the images are entirely identical except that they have different average gray-levels. In other words, all the intensity surfaces are only shifted up some gray-levels from the seed image along z direction in the space, so they have the same surface roughness. Intuitively, the corresponding FDs should be the same as well. In the experiment, the original DBC, RDBC, SDBC, Li's DBC (a is selected as 3 as they suggest) and our improved method ($\delta = 1$) are used. The estimated FDs are showed in the Fig. 4.

The results show all the methods compute the same estimated value expect the original DBC approach by which the estimated values of the FD drift along with the gray-level shifting. The reason is that, in the DBC algorithm, the boxes used for covering the image intensity surface cannot drift up or down along the z direction as the gray-level shifting up or down. They are fixed on the certain place in the space of intensity surface. The first box is always the one where the 0 gray-level in, and the last one is the one where the 255 in. This may result in over-counting the boxes along the z direction as described in Section 2.2 previously. Conversely, our improved method and other three methods have modified the mechanism to count the number of boxes by shifting the column of boxes along the z direction as the gray-level shifting up or down. The first box is always the one where the minimum gray-level in, and the last box to the maximum gray-level. Thus, it ensures that the n_r is the least number of the boxes need to cover the intensity surface in the (i, j)th block column. So they perform very well in this experiment. It proves that our improved DBC method, as well as other three methods, can reflect the roughness of the image more faithfully than the original DBC method and solve the first kind of problem perfectly.

Moreover, it can be found from the results that the FD values computed by our method are slightly larger than those by other methods. The reason is our improved method count the least number of boxes at the current block as other methods do, meanwhile, it avoids under-counting boxes at the border of the neighbor box blocks because of adopting shifting the block in (x, y) plane. So the count number is larger than which is counted in other methods if there is a gray-level sharp at the border of blocks. This will lead to the fitted slope is slightly bigger than other DBC methods. In the Fig. 4, the FDs estimated by RDBC and SDBC are very close to each other and approximately 0.01 bigger than DBC. Li's DBC is 0.045 and our method is 0.06.

Table 1
The algorithm process of our improved method.

	Start	
	Read image	% input the image
	[M, N] = size (image);	% read the sizes of the image
	Select appropriate grid box sizes $s(k)$;	
	$k = 1$;	% variable for using different grid box sizes
Step (1)	**While** $(s(k) < M/2)$;	
	$r = s(k)/M$;	% define r
Step (2)	**For** $i < M/s(k), j < N/s\ (k)$;	
	Count n_r_old;	% use (7)
	Shift block in (x, y) plane within δ pixels;	% under the set rule
	Count n_r_new;	% use (7)
	$n_r = \max(n_r\_old, n_r\_new)$;	
	End For	
	$N_r = \text{sum}\ (n_r)$;	% count the total number
	$k = k + 1$;	% for using different grid box sizes
	End While	
Step (3)	**Call Fit** $(\log N_r, \log(1/r))$;	% perform least-squares linear fit
	Obtain FD;	% FD equal to the slope of the fitted line
	End	

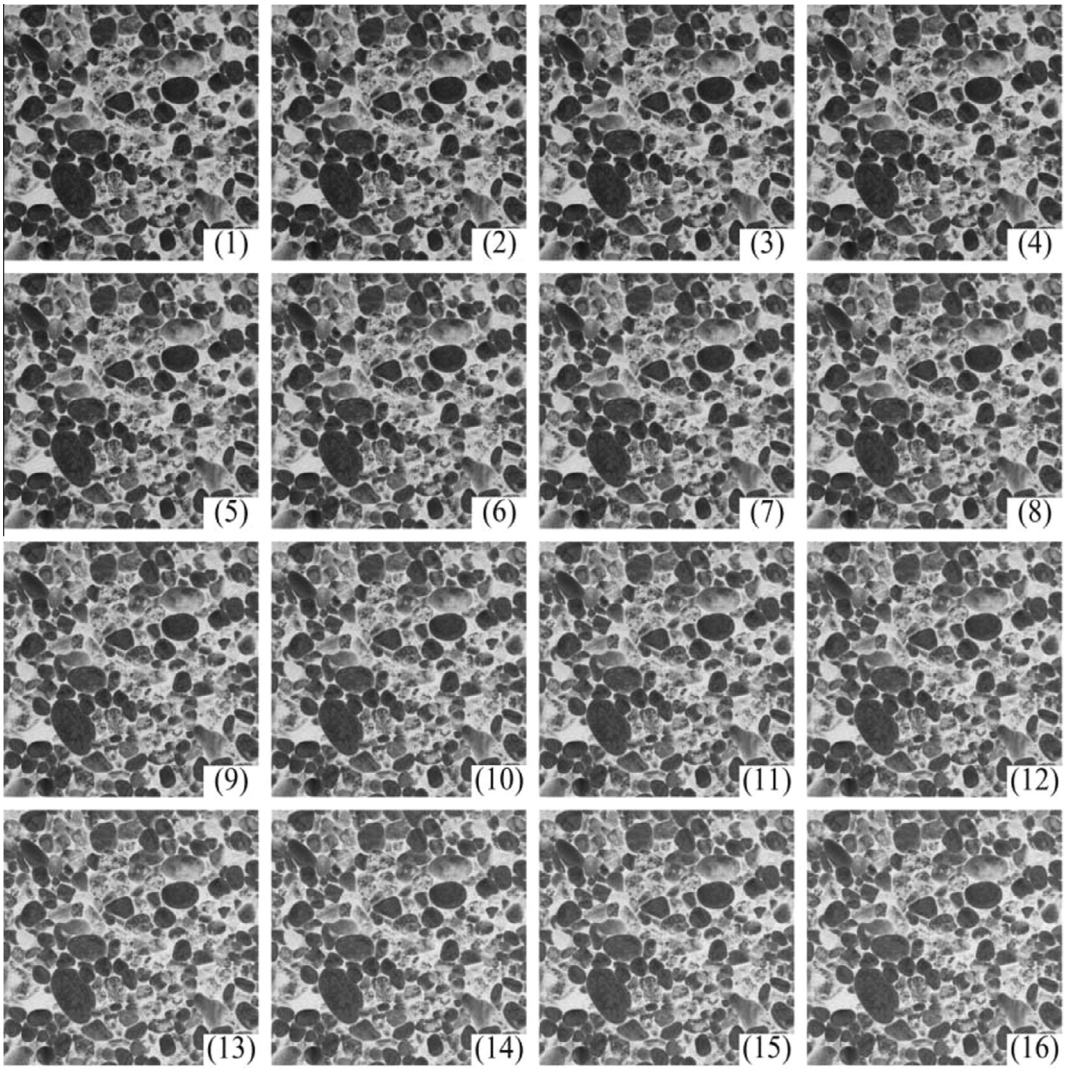

Fig. 3. Sixteen natural textured images with distinct gray-level shifts. The first image (numbered by 1) is select as the seed one. The others are generated by shifting up the gray-level of each pixel with $2 \times (k-1)$ from the seed one, k is the image number.

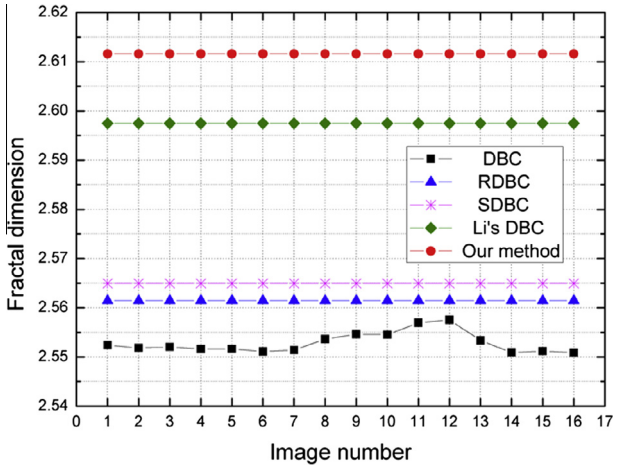

Fig. 4. Computational FDs by different methods for images in Fig. 3.

4.2. Test on synthetic images II

Section 4.1 showed our method can capture the least boxes to prevent over-counting boxes along z direction, and solve the first problem which DBC faced. To the other problem, our method still needs to experiment and discuss.

In this test, a set of 18 256×256 synthetic texture-like images are used, as shown in Fig. 5. The pixel gray-levels of the first image are all set to be 0. The pixel gray-levels of each of the four sub images (the resolution is 128×128) in the second image are 0, 90, 160, and 255, respectively. The remaining images (3 through 9) are generated from the previous level image by down sampling the resolution by 2 and then copying it to four quadrants. Similarly, the pixel gray-levels of the tenth image are all set to be 255. The pixel gray-levels of each of the four sub images in the tenth image are 0, 255, 255, and 0, respectively. Following the same process as before, the remaining images are also generated.

In this experiment, our method ($\delta = 1$) and other four methods (DBC, RDBC, SDBC and Li's DBC) are selected. The results are showed in Fig. 6. It can be found that some of the estimated FDs by the DBC, RDBC, SDBC, and Li's DBC methods lie outside the range between 2.0 and 3.0. Even to the Li's method, there are four images was computed to negative values. This result, of course, is unreasonable. It clearly demonstrates that those methods cannot exactly estimate the FD of this kind of images who having sharp gray-level abruption just at the border of two neighboring box blocks as we described previously in Section 2.2, thus it cannot exactly manifest the roughness of an image. The reason for the

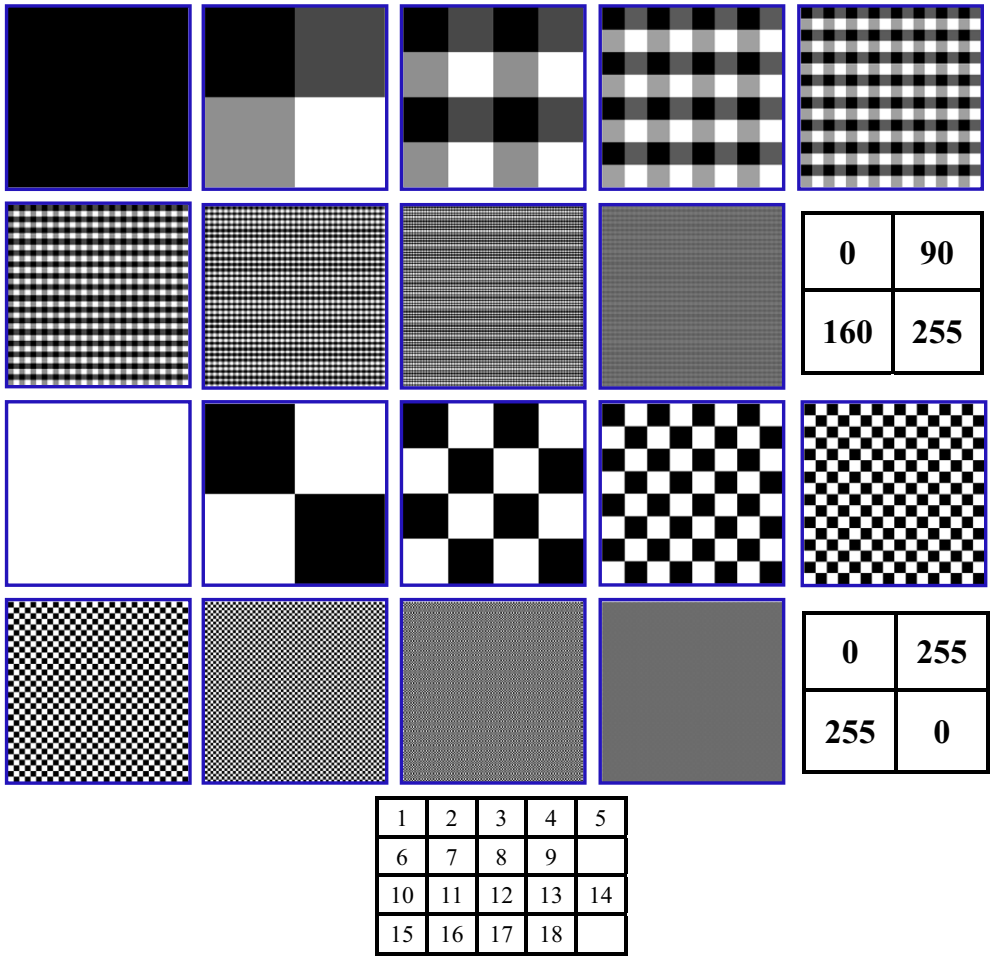

Fig. 5. Eighteen 256 × 256 synthetic texture-like images. The numbers in the figure present the pixel gray levels of each quadrant. Numbers in the bottom right corner present the image number.

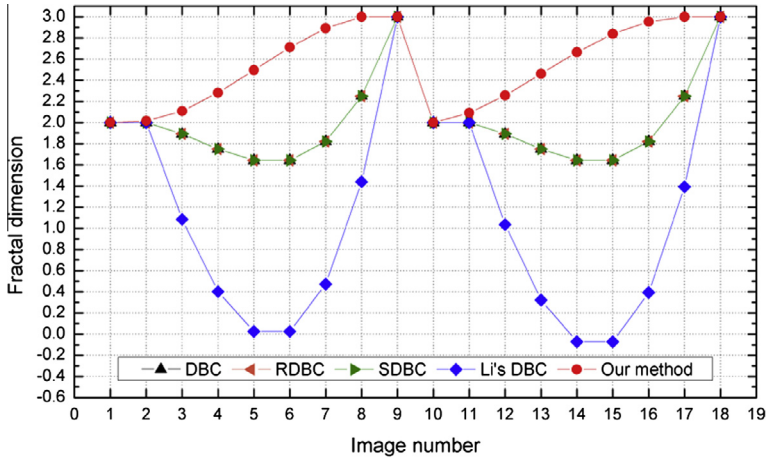

Fig. 6. The computational FDs of the images in Fig. 5 by using different methods. The triangles marked with black, orange, and green color are overlapped because of the same values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

problem is that they all slice a 3D image intensity surface into boxes with fixed sizes at the fixed locations in the (x, y) plane. It happens that some boxes will be missed when the sharp gray-level abruption occurs at the border of two neighboring box blocks.

It is worth noting that the DBC, RDBC, SDBC obtain the same results. This is also able to be explanatory. The gray-level of each pixel in a definitional block is equal to the same value, the sharp gray-level abruption just at the border of two neighboring boxes, so only one box is need in those three methods no matter if the box shifting along the z direction or not since there are no shifting in the (x, y) plane in their methods. Therefore, the estimated values by the three methods are overlapped in the Fig. 6.

Conversely, the estimated FDs by our method are reasonable and increase gradually as the intensity surfaces become more and more complex. Our improved DBC algorithm shifts the blocks within δ pixels in the (x, y) plane and then count the maximum number of n_r to avoid missing the boxes needed to cover the intensity surface. Therefore, it can reflect the roughness of intensity surface more faithfully for this kind of textured images and estimate the FDs more exactly. Compared to other methods in the Fig. 6, it can be found that only our method does deal with the second kind of problem discussed in the Section 2.2.

4.3. Test on natural texture images

Finally, a set of 16 640 × 640 Brodatz distinct textured images [24], as shown in Fig. 7 from Brodatz's database, is used to test by all methods to compare their accuracy. Since the FD is the slope of the least squares linear fitted straight line drawn from the set of the data pairs [log (1/r), log (Nr)], the fitted error providing the measure of fit is very significant to the estimate accuracy of the FD. The lower fitted error result from the better fit.

Let $y = dx + c$ be the fitted straight line, where y denotes log (N_r) and x denotes log $(1/r)$. Then the fitted error E expressed as the root mean-squared distance of the data points from the line is defined as

$$E = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\frac{(dx_i + c - y_i)^2}{1 + d^2}}. \quad (13)$$

The FDs and the errors E computed for the images (shown in Fig. 7) using the different algorithms are presented in Table 2. Those data are given in Figs. 8 and 9 as well.

The FDs by the original DBC method are in the range from 2.21 to 2.68, the average fitted error is 0.0141. For RDBC, they are 2.21–2.68 and 0.0140, 2.22–2.68 and 0.0136 for SDBC, 2.22–2.70 and 0.01467 for Li' DBC, and 2.27–2.73, 0.0107 by our method. It can be clearly found that the fitted error generated by our method is the smallest not only in the average values but also to every image. The average improved accuracy by 24.1% from the DBC. The more accurate numbers of the needed boxes are counted, the less the fitted error will produce. Therefore, comparing with other improved DBC methods, the counted number of the needed boxes in our improved method is more accurate. In other words, our improved DBC algorithm is superior to the other algorithms and can achieve higher accuracy and robustness since it has the least fitted error. Moreover, we can find there are small differences in FDs between those methods from the results. All of the approaches estimate the FDs are very close. The tendencies are similar very much. It also proves directly our method is valid and faithful.

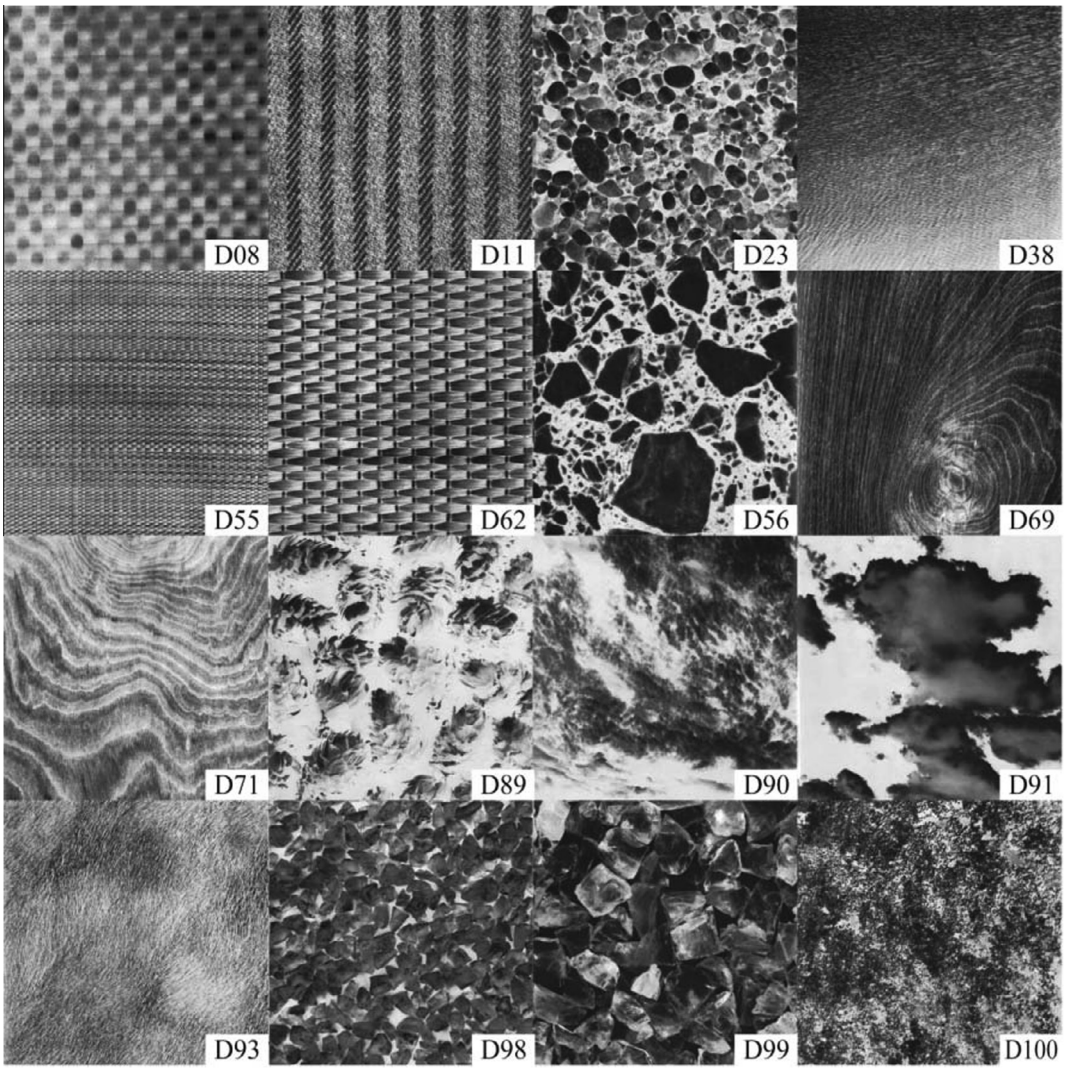

Fig. 7. Sixteen Brodatz texture images for the Brodatz database [24].

Table 2
The computational FDs and fitted errors of the texture images in Fig. 7 by using different methods.

Texture images	DBC		RDBC		SDBC		Li's DBC		Our method	
	FD	E	FD	E	FD	E	FD	E	FD	E
D08	2.41	0.010	2.42	0.010	2.43	0.009	2.44	0.011	2.47	0.007
D11	2.64	0.013	2.65	0.013	2.65	0.013	2.66	0.013	2.70	0.010
D23	2.46	0.018	2.46	0.018	2.47	0.017	2.48	0.019	2.52	0.014
D38	2.59	0.011	2.61	0.012	2.61	0.012	2.63	0.013	2.67	0.008
D55	2.64	0.015	2.64	0.015	2.64	0.015	2.65	0.016	2.71	0.011
D56	2.49	0.018	2.49	0.018	2.50	0.017	2.50	0.018	2.56	0.014
D62	2.39	0.019	2.39	0.019	2.40	0.019	2.41	0.020	2.46	0.016
D69	2.51	0.012	2.53	0.012	2.53	0.012	2.54	0.013	2.58	0.009
D71	2.54	0.010	2.55	0.010	2.55	0.010	2.57	0.011	2.60	0.007
D89	2.39	0.016	2.39	0.016	2.40	0.015	2.42	0.017	2.46	0.012
D90	2.34	0.014	2.35	0.015	2.36	0.014	2.37	0.016	2.41	0.012
D91	2.21	0.011	2.21	0.011	2.22	0.010	2.22	0.012	2.27	0.008
D93	2.68	0.010	2.68	0.010	2.69	0.010	2.70	0.010	2.73	0.007
D98	2.39	0.017	2.39	0.017	2.40	0.016	2.41	0.017	2.46	0.013
D99	2.38	0.016	2.38	0.015	2.39	0.015	2.40	0.016	2.44	0.012
D100	2.66	0.014	2.66	0.014	2.66	0.013	2.68	0.014	2.71	0.010

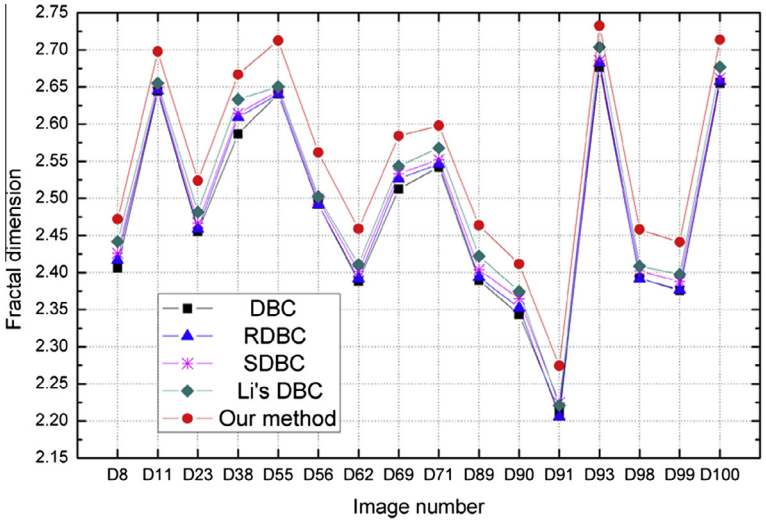

Fig. 8. The computational FDs of the images in Fig. 7 by using different methods.

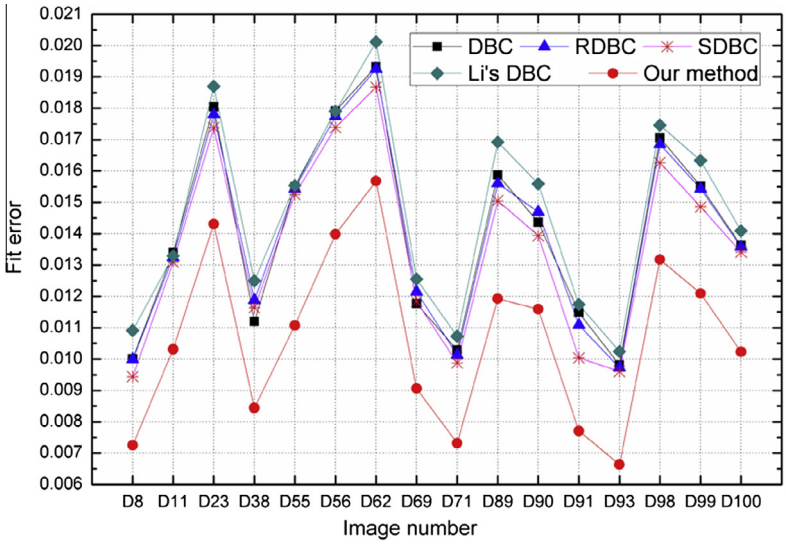

Fig. 9. The computational fitted errors of FDs of the images in Fig. 7 by using different methods.

5. Conclusions

There are two kinds of problems (i.e. boxes are over-counted in z direction and under-counted at the border of the two neighbor box blocks) in the classical DBC to estimate the FD of a gray image. Based on the problems, an improved differential box-counting method is proposed in the article to deal with them simultaneously. The modifications are modifying the box count mechanism, shifting the block in (x, y) plane and selecting appropriate grid box sizes. Three experiments, experiments on two set of the synthetic images and 16 texture images, have been implemented to test our improved method and compare it with other DBC methods.

The first two experiments showed that our method can compute reasonable and accurate FD estimates, and can solve the two kinds of problems, simultaneously. The last experiment shows that our improved method can estimate FD accurately and robustness since it has the least fitted error (improving the accuracy by 24.1% from the original DBC). It is a robust and more accurate method.

Acknowledgments

This study has been supported by the National Natural Science Foundation of China (Grant Nos. 51106019 and 51006016), the National High Technology Research and Development of China (863) Program (Grant No. 2009AA63400), and the National Basic Research Program of China (973) Program (Grant No. 2011CB707300). It has been also supported by the Fundamental Research Funds for the Central Universities.

References

[1] W.-S. Chen, S.-Y. Yuan, C.-M. Hsieh, Two algorithms to estimate fractal dimension of gray-level images, Opt. Eng. 42 (2003) 2452–2464.
[2] P. Asvestas, G.K. Matsopoulos, K.S. Nikita, A power differentiation method of fractal dimension estimation for 2-D signals, J. Vis. Commun. Image Represent. 9 (1998) 392–400.
[3] L.M. Kaplan, C.-C.J. Kuo, Texture segmentation via Haar fractal feature estimation, J. Vis. Commun. Image Represent. 6 (1995) 387–400.
[4] P. Soille, J.-F. Rivest, On the validity of fractal dimension measurements in image analysis, J. Vis. Commun. Image Represent. 7 (1996) 217–229.
[5] B.B. Mandelbrot, The Fractal Geometry of Nature, Freeman, San Francisco, CA, 1982.
[6] A.P. Pentland, Fractal based description of natural scenes, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 661–674.
[7] S. Peleg, J. Naor, R. Hartley, D. Avnir, Multiresolution texture analysis and classification, IEEE Trans. Pattern Anal. Mach. Intell. 4 (1984) 518–523.
[8] K.C. Clarke, Computation of the fractal dimension of topographic surfaces using the triangular prism surface area method, Comput. Geosci. 12 (1986) 713–722.
[9] J.J. Gangepain, C. Roques-Carmes, Fractal approach to two dimensional and three dimensional surface roughness, Wear 109 (1986) 119–126.
[10] R. Voss (Ed.), Random Fractals: Characterization and Measurement, Plenum, New York, 1986.
[11] J.M. Keller, S. Chen, R.M. Crownover, Texture description through fractal geometry, Vision Graphics Image Process. 45 (1989) 150–166.
[12] J.M. Keller, R.M. Crownover, R.Y. Chen, Characteristics of natural scenes related to the fractal dimension, IEEE Trans. Pattern Anal. Mach. Intell. 9 (1987) 621–627.
[13] R. Lopes, N. Betrouni, Fractal and multifractal analysis: a review, Med. Image Anal. 13 (2009) 634–649.
[14] H.O. Peitgen, H. Jurgens, D. Saupe, Chaos and Fractals: New Frontiers of Science, first ed., Springer, Berlin, 1992.
[15] N. Sarkar, B.B. Chaudhuri, An efficient approach to estimate fractal dimension of textural images, Pattern Recogn. 25 (1992) 1035–1041.
[16] N. Sarkar, B.B. Chaudhuri, An efficient differential box-counting approach to compute fractal dimension of image, IEEE Trans. Syst. Man Cybern. 24 (1994) 115–120.
[17] N. Sarkar, B.B. Chaudhuri, Multifractal and generalized dimensions of gray-tone digital images, Signal Process. 42 (1995) 181–190.
[18] S. Buczkowski, S. Kyriacos, F. Nekka, L. Cartilier, The modified box-counting method-analysis of some characteristic parameters, Pattern Recogn. 31 (1998) 411–418.
[19] J. Li, Q. Du, C. Sun, An improved box-counting method for image fractal dimension estimation, Pattern Recogn. 42 (2009) 2460–2469.
[20] A.K. Bisoi, J. Mishra, On calculation of fractal dimension of images, Pattern Recogn. Lett. 22 (2001) 631–637.
[21] M.K. Biswas, T. Ghose, S. Guha, P.K. Biswas, Fractal dimension estimation for texture images: a parallel approach, Pattern Recogn. Lett. 19 (1998) 309–313.
[22] X.C. Jin, S.H. Ong, Jayasooriah, A practical method for estimating fractal dimension, Pattern Recogn. Lett. 16 (1995) 457–464.
[23] M.F. Barnsley, Fractal Everywhere, 2en ed., Academic Press Professional, New York, 1993.
[24] P. Brodatz, Texture: A Photographic Album for Artists and Designers, New York, 1966.