A Parallel Differential Box-Counting Algorithm Applied to Hyperspectral Image Classification

Yu-Chang Tzeng, Kuo-Tai Fan, and Kun-Shan Chen

***Abstract*—In this letter, spatial information through fractal measures is adopted to combine with the spectral information to improve the land cover classification. The spectral features alone and later combined with texture features, using MODIS/ASTER airborne simulator imagery, were fed into a neural classifier. Classification performance was evaluated by a confusion matrix measured by overall accuracy and kappa coefficient. In particular, a parallel differential box-counting (DBC) (PDBC) algorithm for fractal estimation was implemented on a multicore PC. The computation efficiency was ensured through the use of PDBC algorithm which is much faster than that of the original DBC. Furthermore, multicore processors offer great potential for speeding up the computation by partitioning the load among the cores. Multithreading technique is adopted to fully explore its multicore capability. Experimental results demonstrate that the proposed approach provides substantial improvements in classification accuracy while requiring much less computation time without extra hardware resources.**

***Index Terms*—Differential box counting (DBC), fractal dimension (FD), multicore, multithreading.**

I. Introduction

With the revolutionizing advents in sensor technology, hyperspectral images are increasingly being produced. Considering the amount of data to be processed and the high computational cost required by image processing algorithms, conventional computing environments are simply impractical. High-performance computing techniques such as cluster computing, grid computing, and parallel algorithms on a multicore processor go a long way to increasing computational performance [1]–[3]. Of these, multicore processors present an opportunity for speeding up the computation by partitioning the load among the cores [4]–[6]. As the multicore processor systems are becoming more widespread, the demand for efficient parallel algorithms in the field of remote sensing of images is also increasing.

Traditional spectral-based classifiers suffer from ineffectiveness due to the lack of image's spatial properties. Sometimes, spectral analysis methods fail to produce satisfactory results because objects with similar spectral signatures cannot be spectrally separated and thus distinguished. It has been realized that the data captured in neighboring pixels may provide useful and yet supplementary information. Hence, integration of information from both spectral and spatial domains has potential to enhance the classification performance [1], [7]–[9]. Fractal dimension (FD) was introduced to characterize spatial complexity of geographic phenomena at multiple scales in terms of self-similarity [10]. It is useful in the measurement, analysis, segmentation, and classification of shape and texture. However, the use of FD as a texture measure has limitations because some textures may have the same FD but yet with quite different appearances. Mandelbrot [10] also introduced the lacunarity to characterize different texture appearances, which may share the same FD value. Lacunarity measures the deviation of a geometric structure from translational invariance (or gappiness) [11]. In other words, lacunarity represents the distribution of gap sizes: low-lacunarity geometric objects are homogeneous because all gap sizes are the same, whereas high-lacunarity objects are heterogeneous [12]. It is worth to note that lacunarity is a scale-dependent texture measure, in that objects that are homogeneous at a small scale can be heterogeneous at a larger scale [13].

In this study, texture information is extracted and combined with the spectral information to get better classification accuracy. FD, the texture information applied, is estimated by a differential box-counting (DBC) technique [14]. Li *et al.* [15] proposed a model to assign the smallest number of boxes to cover the entire image surface at each selected scale as required, thereby yielding more accurate estimates. However, the original DBC suffers computational inefficiency because the FD is computed sequentially. It is not problematic if the image size is relatively small. However, it becomes troublesome when it is applied to hyperspectral images. As such, a parallel DBC (PDBC) is proposed to improve the computation efficiency. The PDBC algorithm is described in the following section. In Section III, application to a hyperspectral image for land cover classification is given. Computational efficiency of both methods is evaluated. The effectiveness of integrating spatial information with spectral information is also assessed. Finally, conclusions are drawn.

II. PDBC Algorithm

The box-counting methods estimate the FD of a geometric object according to the following equation:

$$D = \lim_{r \to 0} \frac{\log(N_r)}{\log(1/r)} \tag{1}$$

Manuscript received April 20, 2010; revised December 28, 2010, March 31, 2011, and June 16, 2011; accepted July 20, 2011. Date of publication October 6, 2011; date of current version February 8, 2012. This work was supported by the National Science Council, Taiwan, under Grant NSC 98-2116-M-239-001.

Y.-C. Tzeng and K.-T. Fan are with the Department of Electronic Engineering, National United University, Miao-Li City 36003, Taiwan (e-mail: john@nuu.edu.tw; ktfan@nuu.edu.tw).

K.-S. Chen is with the Center for Space and Remote Sensing Research, National Central University, Chung-Li 320, Taiwan (e-mail: dkschen@csrsr.ncu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LGRS.2011.2166243

where r is the side length of the boxes that covers the space occupied by the geometric object under consideration and N_r is the total number of boxes needed to contain all the points of the geometric object. Consider an image of size M by M, the DBC technique [14] partitions the image into several grids. Each grid is of size s, where $M/2 \geq s > 1$ and $r = s/M$ are estimated. If the total number of gray levels is G, then $s' = G/(M/s)$. On each grid, there is a column of boxes of size $s \times s \times s'$. Let the minimum and maximum gray levels of the image grid (k, l) be g_l and g_u, respectively; then, the number of boxes between the minimum and maximum gray levels at grid (k, l) is counted by

$$n_r(k,l) = g_u/s' - g_l/s' + 1. \tag{2}$$

Taking contributions from all the grids, the total number of boxes becomes

$$N_r = \sum_{k,l} n_r(k,l). \tag{3}$$

N_r is counted for different values of r (or s). Then, by using (1), FD D is estimated by least squares of $\log(N_r)$ against $\log(1/r)$ through a linear equation

$$y = Dx + c \tag{4}$$

where $y = \log(N_r)$ and $x = \log(1/r)$. For n different values of r (or s), the FD D and offset c can be computed by

$$D = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{5}$$

$$c = \frac{1}{n}\sum_{i=1}^{n} y_i - D\frac{1}{n}\sum_{i=1}^{n} x_i. \tag{6}$$

The fitting error of the estimation is given by

$$E = \frac{1}{n}\sqrt{\sum_{i=1}^{n} \frac{(Dx_i + c - y_i)^2}{1 + D^2}} \tag{7}$$

which measures the quality of regression. The lower the value of E, the better is the regression. To verify the quality of the estimated FDs, some Brodatz texture images [16] were used. The image size M is equal to 256 for images D84 and D92 and is equal to 128 for the rest of images. The total number of gray levels G is equal to 256 for all images. The FD D was calculated by selecting a grid size $s = 2, 4, 8, \ldots$ and $M/2$. The calculated FDs were compared with those results by Sarkar and Chaudhuri [14] and Li *et al.* [15]. As shown in Table I, closed results were obtained. The FDs estimated by DBC were in the range from 2.61 to 2.88, which is somewhat narrower than that of Sarkar and Chaudhuri and Li *et al.* The fitting errors were smaller than those of Sarkar and Chaudhuri but slightly greater than those of Li *et al.*

The DBC method is applied to estimate the local FD of each pixel in an image. By counting the total number of boxes in a subimage of window size M centered at a pixel, the local FD is computed. The pseudocode of a DBC algorithm is shown in Fig. 1. For a given grid size s, the window sequentially moving pixelwise results in an image N containing the total number of boxes of each pixel. Finally, for different values of s, a fractal image F can be generated. Since the local FD is sequentially evaluated, numerous redundant calculations cause it to be inefficient. Consequently, as shown in Fig. 2, by rearranging the computation sequence, a PDBC algorithm can be devised with a memory buffer B.

TABLE I
CALCULATED FDS OF BRODATZ TEXTURE IMAGES

Texture Images	DBC		Sarkar [14]		Li [15]	
	D	E	D	E	D	E
D03	2.73	0.025	2.60	0.032	2.86	0.007
D04	2.81	0.019	2.66	0.026	2.88	0.004
D05	2.65	0.021	2.45	0.032	2.65	0.010
D09	2.73	0.022	2.59	0.028	2.77	0.006
D24	2.86	0.011	2.45	0.022	2.77	0.005
D28	2.88	0.008	2.55	0.033	2.74	0.009
D33	2.61	0.017	2.23	0.007	2.49	0.007
D54	2.75	0.017	2.39	0.023	2.74	0.007
D55	2.66	0.027	2.48	0.031	2.73	0.008
D68	2.74	0.019	2.52	0.024	2.75	0.005
D84	2.80	0.009	2.60	0.029	2.82	0.005
D92	2.83	0.006	2.50	0.023	2.60	0.005

```
Input: Image H with x samples, y lines, and m bands.
Given window size M, grid size s, and total gray level G.
r = M/s,  s' = G/r
for n = 0 to m-1
  for j = 0 to y-1
    for i = 0 to x-1
      N[n,j,i] = 0
      for k = 0 to r-1
        for l = 0 to r-1
          g_u = max(H[n,j+k*s:j+k*s+s-1,i+l*s:i+l*s+s-1])
          g_l = min(H[n,j+k*s:j+k*s+s-1,i+l*s:i+l*s+s-1])
          N[n,j,i] += g_u / s' - g_l / s' + 1
        endfor
      endfor
    endfor
  endfor
endfor
Output: Image N.
```

Fig. 1. Pseudocode of a DBC algorithm.

In Fig. 1, the region of interest for max and min functions is computed by $x_l = i + l * s$, $x_u = x_l + s - 1$, $y_l = j + k * s$, and $y_u = y_l + s - 1$, which ends up with two multiplications, four additions, and two subtractions. Extra two divisions, two additions, and one subtraction are required to calculate $N[n, j, i]$ and $2s^2$ comparisons for max and min functions. Since indices k and l loop $(M/s)^2$ times, the original DBC requires $2M^2$ comparisons, $6(M/s)^2$ additions, $3(M/s)^2$ subtractions, $2(M/s)^2$ multiplications, and $2(M/s)^2$ divisions for each pixel. In Fig. 2, except one addition less, the pseudocodes marked in Fig. 2 have the same computation complexities as those marked in Fig. 1. Furthermore, $(M/s)^2$ additions are required for sum function, and the region of interest for sum function is computed by $x_l = i$, $x_u = i + r - 1$, $y_l = j$, and $y_u = j + r - 1$ which gives rise to two additions and two subtractions. Similarly, the indices j and i that loop xy/s^2 times result in $2xy$ comparisons, $7xy/s^2 + (M/s)^2 xy/s^2$ additions,

Input: Image H with x samples, y lines, and m bands.
Given window size M, grid size s, and total gray level G.
$r = M/s$, $s' = G/r$
for $n = 0$ to m-1
  for $k = 0$ to s-1
    for $l = 0$ to s-1
      for $j = 0$ to $(y+M-1-k)/s$-1
        for $i = 0$ to $(x+M-1-l)/s$-1
          $g_u = max(H[n,k+j*s:k+j*s+s\text{-}1,l+i*s:l+i*s+s\text{-}1])$
          $g_l = min(H[n,k+j*s:k+j*s+s\text{-}1, l+i*s:l+i*s+s\text{-}1])$
          $B[j,i] = g_u / s' - g_l / s' + 1$
        endfor
      endfor
      for $j = 0$ to $(y-1-k)/s$-1
        for $i = 0$ to $(x-1-l)/s$-1
          $N[n,j,i] = sum(B[j:j+r\text{-}1,i:i+r\text{-}1])$
        endfor
      endfor
    endfor
  endfor
endfor
Output: Image N.

Fig. 2. Pseudocode of a PDBC algorithm.

TABLE II
COMPARISON OF COMPUTATION COMPLEXITY OF THREE METHODS

Operations	PDBC	DBC	Li [16]
Comparison	$2s^2$	$2M^2$	$(2s^2+1)[(M-1)/(s-1)]^2$
Addition	$7+(M/s)^2$	$6(M/s)^2$	$2[(M-1)/(s-1)]^2$
Subtraction	5	$3(M/s)^2$	$7[(M-1)/(s-1)]^2$
Multiplication	2	$2(M/s)^2$	$4[(M-1)/(s-1)]^2$
Division	2	$2(M/s)^2$	$[(M-1)/(s-1)]^2$

$5xy/s^2$ subtractions, $2xy/s^2$ multiplications, and $2xy/s^2$ divisions for each pixel. The indices k and l run s^2 times, for an image with x samples and y lines, PDBC requires $2s^2$ comparisons, $7 + (M/s)^2$ additions, 5 subtractions, 2 multiplications, and 2 divisions for each pixel in average.

The computation complexity of DBC and PDBC are listed in Table II and compared with that of Li *et al.* [15]. Because $(M-1)/(s-1)$ is greater than M/s, DBC always spends less computation time than that of Li *et al.* On the other hand, because M/s should be greater or equal to two, PDBC improves at least four times for the comparison, multiplication, and division operations and at least two times for the addition and subtraction operations compared to DBC. Since PDBC is only a parallel version of DBC, they have exactly the same accuracy. From Tables I and II, we can conclude that Li *et al.* method gives the best accuracy, but it also suffers from least efficiency. In contrast, PDBC has the benefits of the best efficiency with a very acceptable accuracy.

For easy understanding, an example was given here to illustrate how the DBC and PDBC work. By selecting $M = 4$ and $s = 2$, for a one band 7×7 image, some snapshots of the evaluation sequence of DBC and PDBC were shown in Fig. 3. The region of the image was enclosed by a black box, the pixel evaluated was marked as a red symbol, and the corresponding window and grid were represented by red and blue boxes, respectively. Because DBC computed one window at a time, only the pixel marked in Fig. 3 (a)–(d), respectively, was evaluated for $(j, i) = (0, 0)$, (0, 1), (0,2), and (0,3). On the other hand, because PDBC evaluated all nonoverlapping

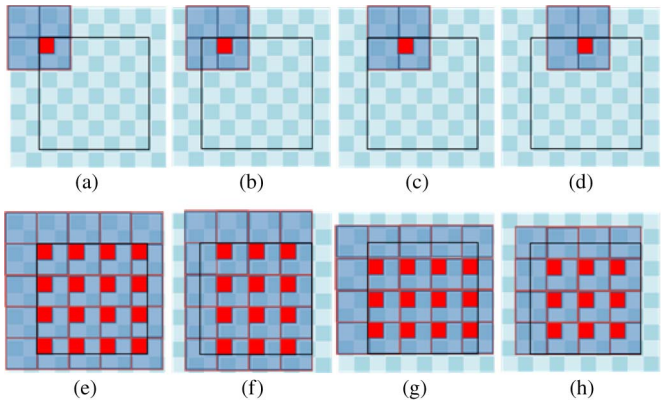


Fig. 3. Snapshots of the computation sequence: Upper row (a)–(d) for DBC and lower row (e)–(h) for PDBC.

Fig. 4. Test site. (a) Color composed image (RGB: MASTER bands 5, 8, and 3). (b) Ground truth map.

grids at the same time, all the pixels marked in Fig. 3 (e)–(h), respectively, were evaluated simultaneously for $(k, l) = (0, 0)$, (0,1), (1,0), and (1,1). It is obvious that PDBC required much less computation time than DBC.

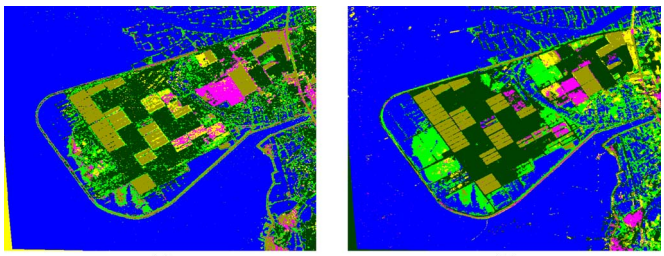
In this letter, both DBC and PDBC were implemented on a multicore PC. In addition, multithreading was used to gain the benefit of their computing power. Multithreading allows a program to be divided into tasks that can operate independently from each other [17]. By multithreading the first loop of DBC (as shown in Fig. 1), a multithreading DBC (MDBC) algorithm is devised. In the same way, a multithreading PDBC (MPDBC) algorithm is achieved by multithreading the first three loops of PDBC (shown in Fig. 2). Although feature selection or reduction is a common practice, this letter intends to retain all the texture measures for all bands to demonstrate the computational effectiveness and efficiency of PDBC.

III. EXPERIMENTAL RESULTS

The Au-Ku plantation field over the west coast of Taiwan was chosen as a test site. The experimental data were gathered by the MODIS/ASTER (MASTER) airborne simulator. The MASTER data were available in 50 bands. Composed from MASTER data (bands 5, 8, and 3 corresponding to the red, green, and blue (RGB) bands), an image of the test site was illustrated in Fig. 4(a). The image size is 834 samples and 652 lines. A ground survey of the test site overpassing the flight is shown in Fig. 4(b). The test site mainly contained six ground cover types: sugar cane A, sugar cane B, bare soil, rice, grass, and seawater. For each class, the numbers of the training and testing samples were given in Table III.

TABLE III
DATA SET SUMMARY

Class Label	Land Cover Types	Number of Training Pixels	Number of Verification Pixels
1	Sugar Cane A	1110	660
2	Sugar Cane B	288	126
3	Bare Soil	540	568
4	Rice	300	310
5	Grass	343	269
6	Sea Water	595	225



(a) (b)

Fig. 5. Classification maps. (a) Using spectral features. (b) Using both the spectral and spatial features.

As outlined above, various DBC algorithms (DBC, PDBC, MDBC, and MPDBC) were implemented on a 64-bit Intel Celeron Core 2 Quad platform. To compare the algorithm performance, the texture measures from Gray-Level Co-Occurrence Matrix (GLCM) [18] were also extracted. Co-occurrence measures use a gray-tone spatial-dependence matrix to calculate texture values. Six commonly used texture measures, namely, mean, variance, contrast, homogeneity, dissimilarity, and entropy, were adopted. A 3×3 sliding window was used to compute the measures. To this end, a dynamic learning neural network (DLNN) classifier [19] was adopted. Because of redundant bands in the data, proper bands were selected by principal component analysis (PCA) [20]. The first six principal components were integrated later with texture measures from fractal and GLCM. For FD-based method, the dimension of features is 6(spectral) + 6(textures), while, for GLCM, the dimension is $6(\text{spectral}) + 6 \times 6(\text{texture measures})$.

The classification map of using spectral features only is shown in Fig. 5(a), while the confusion matrix is shown in Table IV. The classification accuracy reached only 80.12% and the kappa coefficient of 0.747. Numerous pixels of class 1 (sugar cane A) were mutually misclassified with those pixels of class 2 (sugar cane B), class 4 (rice), and class 5 (grass), individually. Poor classification was due to mainly land cover types bearing similar spectral signatures. By applying the DBC techniques to each band, images containing the total number of boxes of each pixel were calculated by selecting window size $M = 16$ and grid sizes $s = 2$, 4, and 8. Fractal images were then produced. Also, the extracted texture was selected by PCA. Together with the spectral information, the first six principal components were applied to DLNN. The classification map of using both the spectral information and spatial information is shown in Fig. 5(b), and its confusion matrix is given in Table V. Now, the overall accuracy was high as 95.13%, and the kappa coefficient was 0.938. All classes were properly classified to an acceptable level.

TABLE IV
CONFUSION MATRIX OF USING SPECTRAL INFORMATION ONLY

class	1	2	3	4	5	6	Producer's
1	487	73	2	14	84	0	73.79
2	12	111	0	1	2	0	88.10
3	0	12	556	0	0	0	97.89
4	31	1	48	226	4	0	72.90
5	141	0	0	1	127	0	47.21
6	3	0	0	0	0	222	98.67
User's	72.26	56.35	91.75	93.39	58.53	100.0	
overall accuracy = 80.12%; kappa coefficient = 0.7470							

TABLE V
CONFUSION MATRIX OF COMBINED SPECTRAL AND SPATIAL FEATURES USING FD

class	1	2	3	4	5	6	Producer's
1	655	3	2	0	0	0	99.24
2	20	106	0	0	0	0	84.13
3	0	1	550	17	0	0	96.83
4	0	16	17	268	9	0	86.45
5	8	0	0	0	261	0	97.03
6	2	0	0	0	10	213	94.47
User's	95.62	84.13	96.67	94.04	93.21	100.0	
overall accuracy = 95.13%; kappa coefficient = 0.938							

TABLE VI
CONFUSION MATRIX OF COMBINED SPECTRAL AND SPATIAL FEATURES USING GLCM TEXTURES

class	1	2	3	4	5	6	Producer's
1	600	40	14	4	2	0	90.91
2	4	119	0	0	3	0	94.44
3	0	2	556	10	0	0	97.89
4	6	0	15	288	1	0	92.90
5	4	0	4	3	258	0	95.91
6	0	0	0	0	0	225	100.0
User's	97.72	73.91	94.40	94.43	97.73	100.0	
overall accuracy = 94.81%; kappa coefficient = 0.934							

Now, the texture measures were tested by GLCM. The confusion matrix obtained by combined spectral and spatial features is shown in Table VI. The overall accuracy was 94.81%, and the kappa coefficient was 0.934. Compared to the results produced by fractal measures, the classification accuracy and kappa coefficient values were very close to each other, and both were high too. The computational complexity of GLCM in terms of window size M is shown in Fig. 6. Compared to FD-based method, GLCM involved in much more texture information leading to lesser computation efficiency. From the band reduction point of view, the FD-based method offers a promising alternative for remote sensing image classification.

For different combinations of M and s, the execution times for using the DBC, PDBC, MDBC, and MPDBC, respectively, are given in Fig. 7(a). In particular, for large M, the execution time of DBC grew rapidly as M/s increased. For PDBC and MPDBC, the execution time did not grow linearly as M/s increased. The group marked as FD presents the total execution time required for FD extraction, and the number in parentheses indicates the value of M. The speedup factors caused by parallelization and multithreading are given in Fig. 7(b). The speedup factor caused by parallelization was obtained by dividing the execution time of DBC by that of PDBC. The

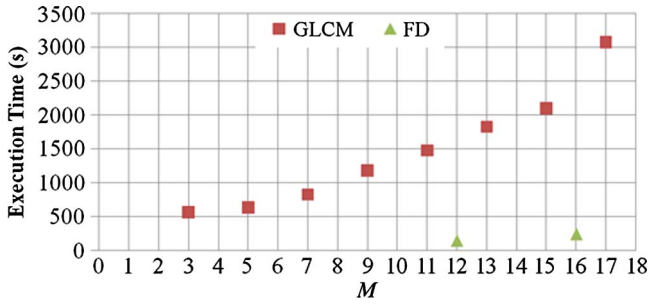


Fig. 6. Computational complexity of GLCM.

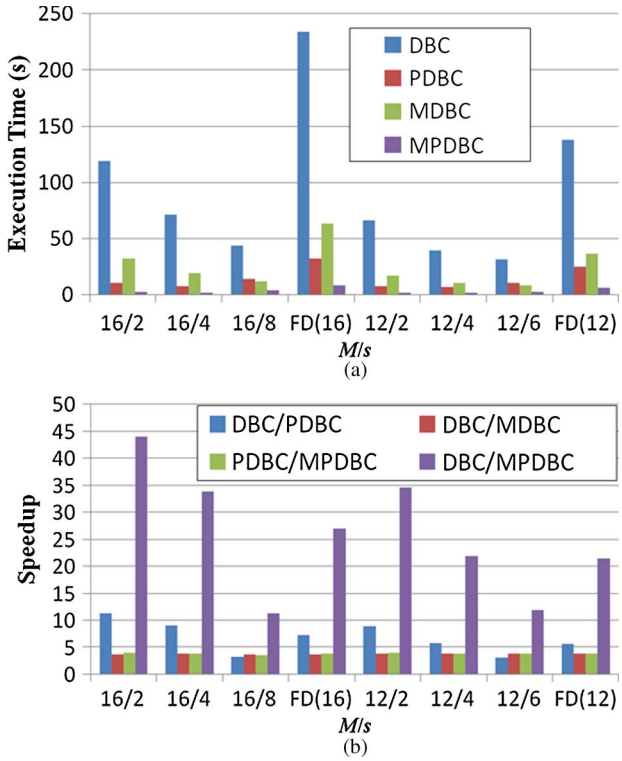


Fig. 7. Experimental result. (a) Execution time. (b) Speedup factor.

speedup factor for multithreading was obtained by dividing the execution time of DBC by that of MDBC and dividing the execution time of PDBC by that of MPDBC. The speedup factor for both parallelization and multithreading was obtained by dividing the execution time of DBC by that of MPDBC. Compared to the DBC, the PDBC improved 7.22 and 5.55 times, the MDBC improved 3.71 and 3.78 times, and the MPDBC improved 26.91 and 21.41 times for $M = 16$ and 12, respectively. On the other hand, compared to the PDBC, the MPDBC constantly improved 3.73 and 3.86 times for $M = 16$ and 12, respectively.

IV. Conclusion

The effectiveness of integrating the spectral information and texture information based on local FD for improving classification performance, using MASTER airborne simulator imagery, has been demonstrated. The computation efficiency was ensured through the use of PDBC algorithm which is much faster than that of the original DBC. The acceleration of computation time depends on the ratio of window size M and grid size s. It is found that the larger the M/s the higher the speedup factor. In addition, multithreading techniques offer further improvement on the multicore PCs. The speedup factor owing to multithreading is proportional to the number of cores that the multicore PC equipped. By using a platform with more cores, further improvement is expected. Experimental results demonstrate that the proposed approach provides substantial improvements in computation time without extra hardware cost.

References

[1] A. J. Plaza, "Parallel processing of remotely sensed hyperspectral imagery: Full-pixel versus mixed-pixel classification," *Concurrency Comput.: Pract. Exp.*, vol. 20, no. 13, pp. 1539–1572, Sep. 2008.
[2] A. Cheriyadat, E. Bright, D. Potere, and B. Bhaduri, "Mapping of settlements in high-resolution satellite imagery using high performance computing," *GeoJournal*, vol. 69, no. 1/2, pp. 119–129, Jun. 2007.
[3] Z. Shen, J. Luo, G. Huang, D. Ming, W. Ma, and H. Sheng, "Distributed computing model for processing remotely sensed images based on grid computing," *Inf. Sci.*, vol. 117, no. 2, pp. 504–518, Jan. 2007.
[4] B. Thomaszewski, S. Pabst, and W. Blochinger, "Parallel techniques for physically based simulation on multi-core processor architecture," *Comput. Graphics*, vol. 32, no. 1, pp. 25–40, Feb. 2008.
[5] Q. Wang and J. Jaja, "Interactive high-resolution isosurface ray casting on multicore processors," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 3, pp. 603–614, May/Jun. 2008.
[6] K. Zeng, E. Bai, and G. Wang, "A fast CT reconstruction scheme for a general multi-core PC," *Int. J. Biomed. Imaging*, vol. 2007, pp. 1–9, 2007, Article ID 29160.
[7] S. Velasco-Forero and V. Manian, "Improving hyperspectral image classification using spatial preprocessing," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 2, pp. 297–301, Apr. 2009.
[8] P. Dong, "Fractal signatures for multiscale processing of hyperspectral image data," *Adv. Space Res.*, vol. 41, no. 11, pp. 1733–1743, 2008.
[9] G. Rellier, X. Descombes, F. Falzon, and J. Zerubia, "Texture feature analysis using a Gauss–Markov model in hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 7, pp. 1543–1551, Jul. 2004.
[10] B. B. Mandelbrot, *The Fractal Geometry of Nature*. New York: Freeman, 1983.
[11] Y. Gefen, Y. Meir, and A. Aharony, "Geometric implementation of hypercubic lattices with noninteger dimensionality by use of low lacunarity fractal lattices," *Phys. Rev. Lett.*, vol. 50, no. 3, pp. 145–148, Jan. 1983.
[12] P. Dong, "Lacunarity for spatial heterogeneity in GIS," *Geographic Inf. Sci.*, vol. 6, no. 1, pp. 20–26, 2000.
[13] R. E. Plotnick, R. H. Gardner, and R. V. O'Neill, "Lacunarity indices as measures of landscape texture," *Landsc. Ecol.*, vol. 8, pp. 20–26, 1993.
[14] N. Sarkar and B. B. Chaudhuri, "An efficient differential box-counting approach to compute fractal dimension of images," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 1, pp. 115–120, Jan. 1994.
[15] J. Li, Q. Du, and C. Sun, "An improved box-counting method for image fractal dimension estimation," *Pattern Recognit.*, vol. 42, no. 11, pp. 2460–2469, Nov. 2009.
[16] P. Brodatz, *Texture: A Photographic Album for Artists and Designer*, New York: Dover, 1966. [Online]. Available: http://sampl.ece.ohio-state.edu/database.htm
[17] J. Beveridge and R. Wiener, *Multithreading Applications in Win32: The Complete Guide to Threads*. Boston, MA: Addison-Wesley, 1997.
[18] R. M. Haralick, K. Shanmugan, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.
[19] Y. C. Tzeng, K. S. Chen, W. L. Kao, and A. K. Fung, "A dynamic learning neural network for remote sensing applications," *IEEE Trans. Geosci. Remote Sens.*, vol. 32, no. 5, pp. 1096–1102, Sep. 1994.
[20] C. R. Rao, "The use and interpretation of principal component analysis in applied research," *Sankhya A*, vol. 26, no. 4, pp. 329–358, Dec. 1964.