Parallel computation for blood cell classification in medical hyperspectral imagery

Wei Li1,2,3, Lucheng Wu1, Xianbo Qiu1,2,3, Qiong Ran1,2 and Xiaoming Xie1

1 College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, People’s Republic of China
2 Institute of Microfluidic Chip Development in Biomedical Engineering, Beijing University of Chemical Technology, Beijing 100029, People’s Republic China

E-mail: liwei089@ieee.org and xbqiu@mail.buct.edu.cn

Received 25 April 2016, revised 7 July 2016
Accepted for publication 12 July 2016
Published 27 July 2016

Abstract
With the advantage of fine spectral resolution, hyperspectral imagery provides great potential for cell classification. This paper provides a promising classification system including the following three stages: (1) band selection for a subset of spectral bands with distinctive and informative features, (2) spectral-spatial feature extraction, such as local binary patterns (LBP), and (3) followed by an effective classifier. Moreover, these three steps are further implemented on graphics processing units (GPU) respectively, which makes the system real-time and more practical. The GPU parallel implementation is compared with the serial implementation on central processing units (CPU). Experimental results based on real medical hyperspectral data demonstrate that the proposed system is able to offer high accuracy and fast speed, which are appealing for cell classification in medical hyperspectral imagery.

Keywords: medical hyperspectral imagery, cell classification, local binary pattern, sparse representation, graphic processing unit

(Some figures may appear in colour only in the online journal)

1. Introduction

Hyperspectral imagery consists of 2-dimensional spatial bands along a third spectral dimensionality, where each pixel represents a sequence of signatures in narrow contiguous wavelengths. Since the rich spectral signatures can be viewed as quantitative diagnostic information, hyperspectral technology is currently considered as a compact tool in imaging and spectroscopy utilized in medical sciences, which is known as medical hyperspectral imagery [1, 2]. Medical hyperspectral imagery has been gradually applied for tongue tumor detection [3, 4], intestinal ischemia detection [5], cancer detection [6], healing in foot ulcers of diabetic patients [7], and diagnosis of hemorrhagic shock [8].

Blood cell classification helps doctors diagnose patients' diseases, such as leukemia, the presence of infections, some particular cancers, and acquired immunodeficiency syndrome, depending on the counts of different cell classes [9–11]. Traditionally, cells are observed by a specialist and the percentage of the occurrence of each type of cell is counted under a fluorescence microscope, which is very tedious, expensive, and time-consuming [12, 13]. Therefore, simple and reliable techniques for rapid cell classification are necessary.

In this work, a practical cell classification in medical hyperspectral imagery is proposed. The imaging systems acquisition includes microscope, liquid crystal tunable filter (LCTF), and silicon charge-coupled devices (CCDs), as illustrated in figure 1. LCTF is adopted because it provides high image quality and is suitable for visible (VIS) and infrared (NIR) regions. Silicon CCDs are popularly used in the VIS and NIR regions in medical hyperspectral imaging systems [14, 15]. By collecting spectral information at each pixel, the resulted image generates a 3-dimensional hypercube including spatial and spectral information as shown in figure 2. The spectral signature of blood cell makes a
Figure 1. Medical hyperspectral imaging system acquisition setup.

continuous spectrum along the direction of the spectral axis in VIS range (i.e. 400–720 nm) [16].

It is worth noting that this type of imaging data has the properties of high-dimensionality and spatial correlation structure. To fully exploit the advantages of the data, feature extraction algorithms should be developed based on the spectral and spatial information. First of all, the high-dimensional spectrum space often results in over-dimensionality and statistical ill-conditioning for most statistical pattern classification tasks. Moreover, the spectral bands are highly-correlated and substantial redundancy exists, which can potentially deteriorate classification performance, especially when the number of available labeled samples is limited. Dimensionality reduction seeks to decrease computational complexity and ameliorate statistical ill-conditioning by discarding redundant features. Band selection and projection-based techniques are two fundamental strategies for this purpose. In band selection, it is to find a small subset of original bands including desirable information; while the projection-based strategy is to project original bands into a lower dimensional subspace based on some criterion function.

Based on the obtained subspace, spatial features are subsequently exploited. For instance, morphological profile (MP) generated by certain morphological operators (e.g. opening and closing) is widely used for modeling structural information. There are some other feature extractions, such as grayscale cooccurrence matrix (GLCM), local binary pattern (LBP) operator [17], Gabor texture features, and gradient orientation features, etc. Classifier is then followed by the extracted spectral and spatial information. Popular classifiers include Bayes classifier (e.g. maximum likelihood estimation (MLE) and Gaussian mixture model (GMM)) [18], support vector machine (SVM), and extreme learning machine (ELM) [19]. Recently, sparse representation-based classification (SRC) [4] and collaborative representation-based classification (CRC) were developed, which are essentially based on the concept that a sample can be representation as a linear combination of labeled samples via the $l_1$ or $l_2$-norm minimization.

Recent advances in high-performance computing technologies have opened new avenues to accelerate hyperspectral image processing algorithms and make many classification algorithms more efficient and practical. Thus, parallel computation is discussed; in doing so, classifier in this work is suitable for time-critical scenarios. Regarding computing platforms for a computationally efficient hyperspectral processing, graphic processing units (GPU) has been demonstrated to be appropriate and offers a tremendous potential to bridge the gap toward real-time analysis of hyperspectral images. Actually, GPU has been drawn great attention to high performance computing community because it can provide very high levels of computing performance at very low cost; in particular, it is suitable to real-time onboard processing due to its portability. Among recent developments, GPUs have been applied to hyperspectral image analysis. Yang et al [20] proposed GPU implementation for unsupervised band selection and compared it with cluster implementation. A parallel endmember extraction algorithm was also accelerated by GPUs [21]. An real-time LBP computing implemented in server platforms such as GPUs, mobile processor and a hybrid programming model image coprocessor was designed in [22]. Tan et al [23] provided a novel two-level parallel computing framework to accelerate the SVM algorithm by utilizing coarse-grained parallelism on GPUs and task parallelism on central processing units (CPUs). An improved implementation of sparse representation classification optimized on GPUs was designed in [24]. Sergio et al [25] proposed several optimizations for accelerating computational performance of automatic target detection and classification algorithm on GPUs. In [26], a parallel target and anomaly detection algorithm by utilizing GPUs to achieve a higher performance compared with a cluster-based approach was proposed.

In this paper, band selection-based dimensionality reduction is adopted due to its relative simplicity and overall robust performance. The criterion of linear prediction error (LPE) [27], based on band dissimilarity, is used for unsupervised band selection. LBP operator in a set of selected bands to produce a comprehensive description of spatial texture information. CRC with distance-weighted Tikhonov regularization (CRT) [28] is then employed for classification based on the induced feature space. In the proposed classification framework, parallel computation is implemented individually for aforementioned three steps, i.e. band selection, feature extraction, and selected classifier.

There are two primary contributions in this research. (1) A novel strategy is proposed for cell classification in medical hyperspectral imagery, which fully utilizes the spectral and spatial information and results in excellent classification performance. (2) The whole framework is further optimized using GPU parallel computing to dramatically accelerate the efficiency. The implementation is carried out using NVIDIA’s compute unified device architecture (CUDA) and demonstrates significant superiority when compared with traditional serial CPU.
The remainder of this paper is organized as follows. Section 2 details the proposed classification framework. Section 3 provides the parallel implementation on GPU using CUDA. Section 4 presents the medical hyperspectral data and experimental results, including the classification accuracy and processing performance of GPU-based parallel computation. Finally, section 5 makes several concluding remarks.

2. Cell classification with spectral-spatial features

The proposed classification framework is illustrated as figure 3, including band selection, spatial feature extraction, and the selected classifier. Let a dataset with training samples \( X = \{x_i\}_{i=1}^n \in \mathbb{R}^d \) (\( d \) is the number of spectral bands) and class labels \( \omega_i \in \{1, 2, \ldots, C\} \), where \( C \) represents the number of classes, and \( n \) is the total number. Let \( n_l \) be the number in \( l \)th class, and \( \sum_{l=1}^C n_l = n \).

2.1. Band selection

Before the spatial feature extraction, band-selection-based dimensionality reduction is conducted with the LPE criterion, which is an unsupervised band selection method to find a small set of distinctive and informative bands. The objective of LPE is to find the most dissimilar bands. Assume there are two bands \( B_1 \) and \( B_2 \) in a selected band subset \( \Phi \). To find a third band \( B_3 \) that is the most dissimilar to \( B_1 \) and \( B_2 \), one can calculate the formula as,

\[
a_0 + a_1B_1 + a_2B_2 = B',
\]

where \( B' \) is a linear prediction of the third band \( B_3 \) using \( B_1 \) and \( B_2 \), and \( a_0, a_1, a_2 \) are the parameters that can minimize the linear prediction error, i.e., \( e = \|B_3 - B'\|_2 \). For any band \( B \), the parameter vector \( a = (a_0, a_1, a_2)^T \) can expressed using \( \Phi a = B \), where \( a \) is estimated using a least-square solution,

\[
a = (\Phi^T\Phi)^{-1}\Phi^TB.
\]

where \( \Phi \) is a matrix whose first column is \( I \), second and third columns include all the pixels in \( B_1 \) and \( B_2 \), respectively. In doing so, after an exhaustive search, the band yielding the maximum error is considered as the most dissimilar band to \( B_1 \) and \( B_2 \) and will be determined as \( B_3 \) for \( \Phi \).

It is apparent that LPE begins with the best two-band combination, and then augments this two-band combination to three, four, and so on, until a desired number of bands is selected. For each selected band, the LBP operator is then used to extract spatial features as described previously.

2.2. LBP operator

LBP [29, 30] belongs to a gray scale and rotation invariant texture operator. Given a center pixel (scalar value) \( t_c \), each neighbor of a local region is assigned with a binary label,
Figure 4. Example of LBP binary thresholding: (a) center pixel $t_c$ and its 8 circular neighbors $\{t_i\}_{i=0}^7$ with radius $r = 1$, (b) a $3 \times 3$ sample block, and (c) binary labels of 8 neighbors.

Figure 5. CPU-GPU hybrid parallel implementation of the LPE band selection algorithm.

Figure 6. A reduction model on the GPU.

which can be either ‘0’ or ‘1’, depending on whether the center pixel has a larger intensity value or not. The neighboring pixels are from a set of equally-spaced samples over a circle of radius $r$ centered at the center pixel. Radius $r$ determines how far the neighboring pixels are located away from the center pixel. Along with selected $m$ neighbors $\{t_i\}_{i=0}^{m-1}$, LBP code for the center pixel $t_c$ is given by

$$LBP_{m,r}(t_c) = \sum_{i=0}^{m-1} U(t_i - t_c)2^i.$$  \hspace{1cm} (3)

where $U(t_i - t_c) = 1$ if $t_i > t_c$ and $U(t_i - t_c) = 0$ if $t_i \leq t_c$.

Figure 4 illustrates an example of binary thresholding process of 8 ($\{m, r\} = (8, 1)$) circular neighbors given the center pixel $t_c$. LBP code is then calculated in a clock-wise direction,
that is, the binary label sequence ‘11 001 010’ = 83. Suppose that the coordinate of \( t \) is \((0, 0)\), each neighbor \( i \) has coordinate of \((r \sin(2\pi i/m), r \cos(2\pi i/m))\). In practice, parameter set \((m, r)\) may change, such as \((4, 1), (8, 2), \) etc. The output of LBP operator in equation (3) indicates that the binary labels in a neighborhood, represented as a \( m \)-bit binary number (including \( 2^m \) distinct values), reflect texture orientation and smoothness in a local region. After obtaining LBP code, an occurrence histogram, as a nonparametric statistical estimation, is computed over a local patch. A binning procedure is required to guarantee the histogram features have the same dimension.

2.3. CRT classifier

Before introducing CRT, CRC is presented first. In CRC, an approximation of a testing sample \( y \) is represented via a linear combination of all available labeled training data (as a dictionary), \( X \). The weight vector \( \alpha^{(CRC)} \) for the linear combination such that \( \|y - X\alpha^{(CRC)}\|_2^2 \) is minimized under the constraint \( \|\alpha^{(CRC)}\|_2^2 \) is also minimized

\[
\text{arg} \min_{\alpha^{(CRC)}} \|y - X\alpha^{(CRC)}\|_2^2 + \lambda \|\alpha^{(CRC)}\|_2^2,
\]

where the \( \lambda \) is a scalar regularization parameter. Taking derivative with regard to \( \alpha^{(CRC)} \) and setting the resultant equation to zero yields

\[
\alpha^{(CRC)} = (X^TX + \lambda I)^{-1}X^Ty.
\]

After obtaining \( \alpha^{(CRC)} \), \( X \) and \( \alpha^{(CRC)} \) are separated into \( l \) class-specific sub-dictionaries according to the given class.
labels of the training samples \( \{X_i\}_{i=1}^{C} \in \mathbb{R}^{d \times n_l} \) and \( \{y_i\}_{i=1}^{C} \in \mathbb{R}^{n \times 1} \). Class label of the testing sample is then determined according to the class that minimizes the residual between the class-specific approximation and the testing pixel. That is

\[
\hat{r}_i^{CRC}(y) = \left\| X_i \alpha_{i}^{CRC} - y \right\|_2,
\]

and class label \(\text{CRC}(y) = \arg \min_{i=1, \ldots, C} \hat{r}_i^{CRC}(y)\).

In [28], a distance-weighted Tikhonov regularization has been considered to adjust the weight vector of CRC (denoted as CRT) which is still solved by an \(\ell^2\)-norm regularization

\[
\arg \min_{\alpha^{CRT}} \left\| y - X \alpha^{CRT} \right\|_2^2 + \lambda \left\| \Gamma y \alpha^{CRT} \right\|_2^2,
\]

where \(\Gamma\) is a biasing Tikhonov matrix which is designed in the form of

\[
\Gamma = \begin{bmatrix}
\left\| y - x_i \right\|_2 & 0 \\
0 & \cdots & \left\| y - x_i \right\|_2
\end{bmatrix}.
\]

And then, the weight vector \(\alpha_i\) can be recovered in a closed-form solution

\[
\alpha^{\text{CRT}} = (X^TX + \lambda \Gamma \Gamma^T)^{-1}X^Ty.
\]

3. GPU-based cell classification

The GPU is usually treated as a parallel computer with shared memory architecture. As all processors of the GPU can share variables within a global address space, it fits the algorithms parallelism very well. To achieve satisfied parallel performance, the throughput of algorithms is very critical in GPU parallel computation design. Follows are the GPU implementations on the band selection, spatial feature extraction, and the selected classifier, respectively.

3.1. GPU implementation on LPE

In order to improve the computing performance of LPE, a GPU-based parallel algorithm is developed as described in the flowchart in figure 5. According to equation (2), \(\Phi\) represents the selected bands set and \(B\) is a linear prediction of \(\Phi\). Equation (2) is a typical matrix operation problem, which is with high computational cost, especially matrix inversion and multiplication. Thus, the calculation of \(a\) is the most time-consuming part in the LPE algorithm.

Fortunately, there are many high performance parallel libraries such as CUDA developed by EM photonics in partnership with NVIDIA, and CUBLAS provided by NVIDIA CUDA that can be used. The \(a\) and \(B\) can be calculated with the high-efficient GPU-accelerated linear algebra libraries of CUDA, such as the operations of matrix multiplication and inversion which use function \texttt{cudaDeviceSgemm}, \texttt{cudaDeviceSgerf}, and \texttt{cudaDeviceSgetri}, respectively. The operation of \(e = \left\| B_1 - B \right\|_2\) could be efficiently implemented by summation reduction as illustrated in figure 6. Finally, the column where maximum \(e\) exists is used as the selected band, and then, the band is inserted into \(\Phi\) for the next iterator until it reaches desired band numbers.
Algorithm 1. Parallel classification based on CRT on GPU.

Input: Training samples set \( \mathbf{X} \in \mathbb{R}^d \), testing samples set \( \mathbf{Y} \in \mathbb{R}^d \)
Initialize: Initial regularized parameter \( \lambda \)
Step 1. Copy data from host to device
step 2. Invoke \texttt{cudaDeviceSgemm} to calculate \( \mathbf{X}^\top \mathbf{X} \) and \( \mathbf{X}^\top \mathbf{Y} \); and then, invoke kernel function \texttt{normsKernel} to compute \( \mathbf{\Gamma}_x^\top \mathbf{\Gamma}_x \) on GPU.
while do
  Step 3. Invoke kernel function \texttt{geagKernel} to calculate \( \lambda \mathbf{\Gamma}_x^\top \mathbf{\Gamma}_x \)
  Invoke kernel function \texttt{addSubKernel} to calculate \( \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{\Gamma}_x^\top \mathbf{\Gamma}_x \)
  Invoke \texttt{cudaDeviceSgetrf} and \texttt{cudaDeviceSgetri} to calculate \( (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{\Gamma}_x^\top \mathbf{\Gamma}_x)^{-1} \)
  Invoke \texttt{cudaDeviceSgemm} to calculate \( \alpha^{(CRT)} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{\Gamma}_x^\top \mathbf{\Gamma}_x)^{-1} \mathbf{X}^\top \mathbf{y} \)
  step 4. \( i \leftarrow i + 1 \)
end while
step 5. Invoke kernel function \texttt{segment_kernel} to assign weighted vector according to the numbers of classes
while do
  step 6. Invoke \texttt{cudaDeviceSgemm} to calculate \( \mathbf{y}_t = \mathbf{X}_t \alpha^{(CRT)}_t \)
  Invoke kernel function \texttt{distNorms_kernel} to calculate \( r^{CRT}_t(\mathbf{y}) = \| \mathbf{X}_t \alpha^{(CRT)}_t - \mathbf{y} \|_2 \)
  step 7. \( l \leftarrow l + 1 \)
end while
step 8. Copy \( \mathbf{e} \) from device to host
step 9. Calculate \( CRT(\mathbf{y}) = \arg \min_{i=1,\ldots,C} r^{CRT}_i(\mathbf{y}) \) on CPU
Output: Class labels of \( \mathbf{Y} \).

3.2. GPU implementation on LBP

Figure 7 illustrates the GPU parallel implementation of the LBP feature selection algorithm. LBP feature extraction process mainly depends on a local patch with neighboring pixels as shown in figure 4, which could be parallel implemented on GPUs. Specifically, for each selected band, the LBP code image is divided into a stream to achieve task parallelism. The operator of flipping edge pixels is employed to handle edge pixels to make sure edge pixels with neighbors. Finding maximum value in normalizing operator is parallelly achieved by using reduction model described in figure 6.

Texture memory is applied to store the LBP code image that it could provide higher performance and bandwidth when the image has large special locality. Neighboring pixels are added cache as illustrated in figure 8. Then, the calculation of LBP code and histogram of each pixel is executed in an allocated thread on GPUs.

3.3. GPU implementation on CRT

Figure 9 illustrates GPU parallel implementation of the CRT classification algorithm. As indicated in equation (9), the process of obtaining \( \alpha^{(CRT)} \) is generally computationally intensive due to the computing of biasing Tikhonov matrix for each testing sample. \( \mathbf{Y} \) represents all the testing samples. In figure 9, terms \( \mathbf{X}^\top \mathbf{X} \) and \( \mathbf{X}^\top \mathbf{Y} \) are calculated in advance, since these elements remain unchanged in each loop. A detailed step-by-step algorithm description of the parallel CRT algorithm is provided in algorithm 1.

The operations of matrix multiplication and inversion are realized using CULA library. And then, the operation of calculating \( \mathbf{\Gamma}_x^\top \mathbf{\Gamma}_x \) has been implemented by a kernel function called \texttt{normsKernel}. The calculation of \( \alpha \) is optimized using CULA library. After that, the vector \( \alpha \) is divided into multiple matrices according to the number of labeled classes that it is convenient to calculate \( \mathbf{y}_t = \mathbf{X}_t \alpha \). This operation is implemented by a kernel function called \texttt{segment_kernel}. Finally, the reduction model described in figure 6 can be employed to calculate \( r^{CRT}_i(\mathbf{y}) = || \mathbf{X}_t \alpha^{(CRT)}_t - \mathbf{y} ||_2 \). The kernel function \texttt{distNorms_kernel} is implemented to realize this operation.

4. Experimental analysis

The experimental data collected by the imaging systems acquisition as illustrated in figure 1. Figure 10 presents the two experimental datasets including 3 classes which indicate background, white cell, and red cell, respectively. The first dataset consists of size 200 \( \times \) 200 pixels with 33 spectral bands of spectral coverage 400–720 nm and the second one is composed of size 150 \( \times \) 150 pixels with the same number of spectral bands.

Figure 11 describes the classification accuracy of CRT-LBP, which is compared with traditional CRT algorithm on the real medical hyperspectral data. It illustrates that with spatial feature, the CRT-LBP can exhibit better performance. Meanwhile, it can achieve the best classification performance when the parameter \( \lambda \) is set 1. In our experiments, we empirically set the number of bands to be selected equal to 10 and set \( m = 8 \) and \( r = 1 \) according to equation (3). 100 labeled pixels of each class are randomly chosen as training samples and the remaining labeled pixels are used as testing samples. The classification accuracy of each class is listed in tables 1 and 2 for two experimental data, respectively. It is easily observed that the classification accuracy of serial version (denoted as CRT-LBP-S) and the one of GPU parallel version (denoted as CRT-LBP-P) achieves the similar accuracy.
The proposed full classification framework has been carried out on a platform which contains an Intel(R) Core(TM) i7-4790 CPU at 3.60 GHZ and 8 GB of RAM, running Windows 7 (64-bit), and the GPU used is an NVIDIA GeForce GTX 960 which features 8 multiprocessors and 1024 CUDA cores operating at clock frequency of 1253 MHz, with global memory of 2 GB. All the serial and parallel versions of the LPE, LBP, and CRT are implemented using the C programming language on Microsoft Visual Studio 2013 integrated development environment. CUDA 7.0 and CULA R18 are used in the corresponding implementation.

Table 3 lists the obtained results in terms of computation times and speedups measured after comparing the parallel implementations of the proposed classification framework with the equivalent serial version using the two experimental data. It is obvious that for each key part (e.g. band selection, LPE, or feature extraction, LBP) of the classifier, parallel version has significant improvement on speedups. Table 4 further reports the analysis on the serial and GPU version with varying number of training samples per class using the first experimental data. At the same time, figure 12 illustrates the classification accuracy versus varying number of training samples per class. It is easily observed that the classification accuracy is improved significantly as the number size of
<table>
<thead>
<tr>
<th>Training samples</th>
<th>Serial version</th>
<th>Parallel version</th>
<th>Speed up (X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>175.36</td>
<td>53.34</td>
<td>3.29</td>
</tr>
<tr>
<td>100</td>
<td>1442.86</td>
<td>141.05</td>
<td>10.23</td>
</tr>
<tr>
<td>150</td>
<td>5103.62</td>
<td>245.31</td>
<td>20.80</td>
</tr>
<tr>
<td>200</td>
<td>11790.93</td>
<td>375.48</td>
<td>31.40</td>
</tr>
<tr>
<td>250</td>
<td>25018.50</td>
<td>477.37</td>
<td>52.41</td>
</tr>
</tbody>
</table>

Figure 12. Classification accuracy versus varying training samples per class for the CRT-LBP algorithm using the first experimental dataset.

training samples becomes larger, so does the computational efficiency of serial version of CRT algorithm. However, the parallel implementation of the algorithm keeps computational efficiency, which can be verified by the speedups. It has been demonstrated that the GPU platform is able to adapt computationally expensive classification problems.

5. Conclusions

In this paper, a novel strategy was proposed for cell classification in medical hyperspectral imagery. Specifically, the LPE algorithm was applied to select informative bands, and the LBP algorithm regarded as preprocessing approach was employed to improve classification accuracy by combining spectral features with spacial context, followed by the state-of-the-art CRT classifier. All of these approaches were parallel implemented on GPUs by using NVIDIA CUDA. Experimental results on two real medical hyperspectral datasets, with comparisons to equivalent serial versions, demonstrated the similar classification accuracy in high speed. In future work, the presented algorithm will be adapted to multi-GPU platforms.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants No. NSFC-61571033, 81371711, and partly by the Research Fund for Basic Researches in Central Universities under Grants No. XK1521, BUCTCR201401, BUCTCR201615.

References

[17] Masood K and Rajpoot N 2009 Texture based classification of hyperspectral colon biopsy samples using cibrp In IEEE Int. Symp. on Biomedical Imaging (Boston, MA) pp 1011–4
[19] Li W, Chen C, Su H and Du Q 2015 Local binary patterns and extreme learning machine for hyperspectral imagery