Parallel Spatial–Spectral Hyperspectral Image Classification With Sparse Representation and Markov Random Fields on GPUs

Zebin Wu, *Member, IEEE*, Qicong Wang, Antonio Plaza, *Fellow, IEEE*, Jun Li, *Member, IEEE*, Le Sun, *Member, IEEE*, and Zhihui Wei

***Abstract*—Spatial–spectral classification is a very important topic in the field of remotely sensed hyperspectral imaging. In this work, we develop a parallel implementation of a novel supervised spectral–spatial classifier, which models the likelihood probability via $l_1 - l_2$ sparse representation and the spatial prior as a Gibbs distribution. This classifier takes advantage of the spatial piecewise smoothness and correlation of neighboring pixels in the spatial domain, but its computational complexity is very high which makes its application to time-critical scenarios quite limited. In order to improve the computational efficiency of the algorithm, we optimized its serial version and developed a parallel implementation for commodity graphics processing units (GPUs). Our parallel spatial–spectral classifier with sparse representation and Markov random fields (SSC-SRMRF-P) exploits the low-level architecture of GPUs. The parallel optimization of the proposed method has been carried out using the compute unified device architecture (CUDA). The performance of the parallel implementation is evaluated and compared with the serial and multicore implementations on central processing units (CPUs). In fact, the proposed method has been designed to adequately exploit the massive data parallel capacities of GPUs together with the control and logic capacities of CPUs, thus resorting to a heterogeneous CPU–GPU framework in the design of the parallel algorithm. Experimental results using real hyperpsectral images demonstrate very high performance for the proposed CPU–GPU parallel method, both in terms of classification accuracy and computational performance.**

***Index Terms*—Compute unified device architecture (CUDA), graphics processing units (GPUs), hyperspectral imaging, Markov random fields (MRFs), parallel implementation, sparse representation, spatial–spectral classification.**

Manuscript received October 22, 2014; revised February 08, 2015; accepted March 13, 2015. Date of publication March 30, 2015; date of current version July 30, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61471199, Grant 61101194 and Grant 11431015, in part by the Jiangsu Provincial Natural Science Foundation of China under Grant BK2011701, in part by the China Scholarship Fund under Grant 201406845012, in part by the Research Fund for the Doctoral Program of Higher Education of China under Grant 20113219120024, in part by the Jiangsu Province Six Top Talents project of China under Grant WLW-011, and in part by the China Academy of Space Technology Innovation Foundation under Grant CAST201227.

Z. Wu is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China, and also with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, Cáceres E-10003, Spain (e-mail: zebin.wu@gmail.com).

Q. Wang, L. Sun, and Z. Wei are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: wqclandy@163.com; sunlecncom@163.com; gswei@njust.edu.cn).

A. Plaza is with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, Cáceres E-10003, Spain (e-mail: aplaza@unex.es).

J. Li is with the Guangdong Provincial Key Laboratory of Urbanization, Geosimulation, and Center of Integrated Geographic Information Analysis, School of Geography and Planning, Sun Yat-sen University, Guangzhou 510275, China (e-mail: lijun48@mail.sysu.edu.cn).

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/JSTARS.2015.2413931

I. Introduction

CLASSIFICATION is an important technique for remotely sensed hyperspectral data exploitation [1]. Hyperspectral instruments acquire images in hundreds of narrow and continuous spectral bands, which provide detailed spectral signatures of the observed image objects [2]. Since different materials usually reflect electromagnetic energy differently at specific wavelengths, it is generally known that land-cover classification can be performed more effectively with hyperspectral scenes than with panchromatic and multispectral images [3]. Therefore, hyperspectral image (HSI) classification has been applied in many application domains, such as environmental monitoring, geologic surveys, and agricultural production. Moreover, high classification accuracy is important for proper decision-making in many critical scenarios [1], including for instance precision agriculture, urban planning, or military reconnaissance. However, HSI classification is subject to important challenges, such as high dimensionality and insufficient training samples in practice, which pose significant challenges to HSI classification.

Among many techniques for HSI classification, supervised approaches have received a lot of interest [2]. Classification with kernel methods such as support vector machines (SVMs) [4] has been complemented using spatial information extraction using different strategies, most notably Markov random fields (MRFs) [5], [6] and morphological profiles [7], including techniques embedded in multiple kernel learning frameworks [8]. Sparse representation methods [9], [10] have also been widely used for HSI classification. Other important trends in HSI classification comprise the use of graph-based methods [11], [12]. Probabilistic classifiers such as the multinomial logistic regression (MLR) [13] (possibly complemented with spatial-based techniques such as the MRF) have also found great success in the recent literature, including sparse versions [14], strategies embedded in active learning frameworks [15], [16], and generalized kernel-based approaches [17]. Most of these techniques rely on the assumption that, in natural scenes, there are large

homogeneous regions, where the neighboring pixels within the regions probably consist of the same type of material and have similar spectral characteristics. As a result, the spatial contextual correlation has been exploited in combination with spectral information via a preprocessing procedure [18], or during the classification process itself [19]–[21] (possibly including sparse representation strategies [22]), or as a postprocessing [23].

Taking advantage of the spatial piecewise smoothness and the expected correlation of neighboring pixels in natural scenes, in [21], we proposed a novel supervised spectral–spatial classifier with sparse representation and MRF-based spatial prior under the Bayesian framework. This classifier models the likelihood probability via an $l_1 - l_2$ sparse representation method, and the spatial prior is modeled as a Gibbs distribution, which specifies an MRF on the classification labels. As shown in [21], the classification accuracy of this method is comparable or superior to that exhibited by many other state-of-the-art techniques. However, its computational complexity was shown to be very high, thus limiting its application in time-critical scenarios. The reason is not only the extremely high dimensionality of hyperspectral data, but also that the sparse representation-based strategy adopted by this classifier searches for dedicated atoms in the training dictionary for each pixel under test [24], not involving a training stage like other supervised methods such as the SVM. This results in a computational complexity that is even higher than SVM-based classification.

With the rapid development of high performance computing technologies, it is now possible to significantly accelerate the HSI processing algorithms in parallel computer architectures [25], such as Beowulf clusters and distributed computers [26], multicore central processing units (CPUs) [27], field-programmable gate arrays (FPGAs) [28], or graphics processing units (GPUs) [29]. Among these, heterogeneous CPU–GPU techniques offer a tremendous potential to bridge the gap toward real-time analysis of HSIs [30], combining the massively parallel and data intensive computation abilities of GPUs [31] with the good logic control ability of CPUs [32], [33].

In this paper, we develop a new strategy for accelerating the computational performance of spatial–spectral classifiers with sparse representation and MRFs on CPU–GPU heterogeneous platforms. The proposed parallel implementation properly exploits the CPU–GPU architecture at low level and makes effective use of the computational power of both CPUs and GPUs in synergetic fashion. The operations relevant to large data and intensive computations are carried out on the GPUs, while the rest of the operations (mostly related with control) are executed on the CPU. A parallel optimization of the proposed methods using NVIDIA's compute unified device architecture (CUDA) for GPUs are described and evaluated in detail. The performance of the proposed CPU–GPU parallel implementation is assessed using real hyperspectral datasets and compared with both serial and multicore implementations on CPUs.

This paper is organized as follows. Section II describes the supervised HSI classification using sparse representation and MRFs. Section III presents its parallel implementation on CPU–GPU heterogeneous platforms. Section IV experimentally assesses the proposed method in terms of both classification accuracy and computational performance. Finally, Section V concludes with some remarks and hints at plausible future research lines.

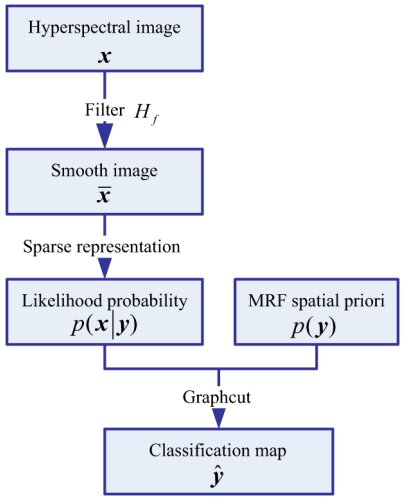


Fig. 1. Framework of the proposed spatial–spectral classifier based on sparse representation and MRF (SSC-SRMRF).

II. Spatial–Spectral Classifier Based on Sparse Representation and MRF

A. Classifier Framework

To describe the proposed classifier in mathematical terms, let us assume that there are K distinct classes, including l training samples in total. The kth class has l_k training samples, denoted as $\boldsymbol{A}^k = [\boldsymbol{a}_1^k, \boldsymbol{a}_2^k, \ldots, \boldsymbol{a}_{l_k}^k] \in \boldsymbol{R}^{L \times l_k}$, where L is the number of spectral bands. Let $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}] \in \boldsymbol{R}^{L \times N}$ be a HSI with N pixels, and $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]$ be an image of class labels, where $\boldsymbol{x}_i \in \boldsymbol{R}^L$ is an L-dimensional hyperspectral pixel observation, and $y_i \in \{1, 2, \ldots, K\}$ is the class label corresponding to the ith pixel.

In a Bayesian framework [19], the proposed classifier can be defined by maximizing the posterior of $\mathbf{y}$ for a given HSI $\mathbf{x}$ in the following expression:

$$p(\boldsymbol{y}\,|\boldsymbol{x}\,) \propto p(\boldsymbol{x}\,|\boldsymbol{y})p(\boldsymbol{y}) \tag{1}$$

where $p(\boldsymbol{x}\,|\boldsymbol{y}\,)$ is the likelihood distribution and $p(\boldsymbol{y})$ is the prior distribution of the class labels $\boldsymbol{y}$. Based on model (1), a sparse representation can be introduced to model the probability $p(\boldsymbol{x}\,|\boldsymbol{y}\,)$, which directly forms a structured dictionary [9] from the concatenation of several class-wise subdictionaries consisting of training samples, and expresses an unknown pixel as a sparse vector whose nonzero entries correspond to the weights of the selected training samples. Then, the MRF is used to model the spatial prior $p(\boldsymbol{y})$. The framework adopted to implement the proposed spatial–spectral classifier based on sparse representation and MRF (SSC-SRMRF) is summarized in Fig. 1. Details of the classifier are given in the following subsections.

B. Sparse Representation Classification Based on $l_1 - l_2$ Regularization

Sparse representation methods rely on the assumption that, if a single pixel in a HSI belongs to the kth class, then its spectrum approximately lies in a low-dimensional subspace spanned by the training samples of the kth class, which can be compactly represented by a linear combination of these training samples [9]. The sparse representation of an unknown pixel is expressed as a sparse vector whose nonzero entries correspond to the weights of the selected training samples [22]. The class label of the test sample can be easily determined via the sparse vector that is recovered by solving a sparse optimization problem. For an unknown test sample $\boldsymbol{x}_i \in \boldsymbol{R}^L$, it can be modeled as follows:

$$\begin{aligned}\boldsymbol{x}_i &= \boldsymbol{A}^1\boldsymbol{s}^1 + \boldsymbol{A}^2\boldsymbol{s}^2 + \cdots + \boldsymbol{A}^K\boldsymbol{s}^K + \boldsymbol{n} \\ &= [\boldsymbol{A}^1 \cdots \boldsymbol{A}^K][\boldsymbol{s}^1 \cdots \boldsymbol{s}^K]^{\mathrm{T}} + \boldsymbol{n} = \boldsymbol{A}\boldsymbol{s}_i + \boldsymbol{n}\end{aligned} \tag{2}$$

where $\boldsymbol{A} = [\boldsymbol{A}^1, \boldsymbol{A}^2, \ldots, \boldsymbol{A}^K] \in \boldsymbol{R}^{L\times l}$ is a structured dictionary formed from the concatenation of several class-wise subdictionaries consisting of training samples $l = \sum_{k=1}^{K} l^K$, $\boldsymbol{n}$ is Gaussian noise with zero mean, $\boldsymbol{s}_i = [{\boldsymbol{s}_i}^1, \ldots, {\boldsymbol{s}_i}^K]^{\mathrm{T}} \in \boldsymbol{R}^l$ is a sparse vector formed by concatenating the sparse vectors $\{{\boldsymbol{s}_i}^k\}_{k=1,2,\ldots,K}$, and ${\boldsymbol{s}_i}^k = [s_{i1}^k, s_{i2}^k, \ldots, s_{il_k}^k]^{\mathrm{T}} \in \boldsymbol{R}^{l_k}$ is an unknown sparse vector whose entries are the weights of the corresponding atoms in the subdictionary $\boldsymbol{A}^k$. According to the physical characteristics of the HSI, $\boldsymbol{s}_i$ satisfies the nonnegativity and sum-to-one constraints.

For a certain HSI, its noise $\boldsymbol{n}$ is fixed, and $\sum_{k=1}^{K} {\boldsymbol{s}_i}^K = 1$ (sum-to-one constraint), thus the likelihood probability $p(\boldsymbol{x}_i\,|y_i)$ (the condition likelihood of any $\boldsymbol{x}_i$ given the training samples in the kth class) is proportional to $\sum_{j=1}^{l_k} {s_{ij}}^k$, i.e.,

$$p(\boldsymbol{x}_i\,|y_i = k\,) \propto \sum_{j=1}^{l_k} {s_{ij}}^k. \tag{3}$$

Let us now suppose that the probabilities of every sample assigned to the kth class are equal, i.e., $p(y_i) = \frac{1}{K}$, and the samples are independent of each other. Then, the posterior probability of a spectral-only classification process can be simply formulated as follows:

$$\begin{aligned}p(y_i = k\,|\boldsymbol{x}_i\,) &\propto p(\boldsymbol{x}_i\,|y_i = k)p(y_i) \\ &\propto \sum_{j=1}^{l_k} {s_{ij}}^k.\end{aligned} \tag{4}$$

Therefore, the spectral-only classification process can be simplified as the solution of sparse coefficients $\boldsymbol{s}$ of sample $\mathbf{x}$ under the sum-to-one constraint. Under a sparse representation framework, the likelihood probability of sample $\mathbf{x}_i$ can be solved by the following $l_1 - l_2$ optimization problem

$$\begin{aligned}\hat{\boldsymbol{s}}_i = \arg\min_{\boldsymbol{s}} &\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{s}_i\|_2^2 + \lambda(\|\boldsymbol{s}_i||_1 - \|\boldsymbol{s}_i||_2) \\ &\text{s.t. } \boldsymbol{s}_i \geq 0, \quad \mathbf{1}^{\mathrm{T}}\boldsymbol{s}_i = 1\end{aligned} \tag{5}$$

where $\|\boldsymbol{s}_i||_1 = \sum_{j=1}^{J} |s_{ij}|$ is l_1-norm, $\|\boldsymbol{s}_i||_2 = \left(\sum_{j=1}^{J} {s_{ij}}^2\right)^{\frac{1}{2}}$ is l_2-norm, and λ is a scalar regularization parameter. In order to reduce the influence of spectral noise and enhance the spatial piecewise smoothness of the classified image, we use a filtering function to preprocess the HSI. Moreover, $\|\boldsymbol{s}_i||_1 \equiv 1$ under the sum-to-one constraint. Then, model (5) can be rewritten as

$$\begin{aligned}\hat{\boldsymbol{s}}_i = \arg\min_{\boldsymbol{s}} &\left\{\frac{1}{2}\|H_f * \boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{s}_i\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{s}_i\|_2^2\right\} \\ &\text{s.t. } \boldsymbol{s}_i \geq 0, \quad \mathbf{1}^{\mathrm{T}}\boldsymbol{s}_i = 1\end{aligned} \tag{6}$$

where H_f is a filter function, such as the mean filter and Gaussian filter. The aforementioned expression can be rewritten with Lagrange multipliers. It can be efficiently solved based on the alternating direction method of multipliers (ADMM) as described in [34] and [35]

$$\hat{\boldsymbol{s}}_i = \arg\min_{\boldsymbol{s}} \left\{\begin{array}{c}\frac{1}{2}\|H_f * \boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{s}_i\|_2^2 - \frac{\lambda}{2}\|\boldsymbol{s}_i\|_2^2 \\ + l_{\{1\}}(1^T\boldsymbol{s}_i) + l_{\boldsymbol{R}_+^n}(\boldsymbol{s}_i)\end{array}\right\} \tag{7}$$

where $\boldsymbol{R}_+^n$ denotes a positive vector space, and $l_C(x)$ is an indicative function (if $x \in C$, $l_C(x) = 1$; $x \neq 0$, $l_C(x) = +\infty$).

C. Spatial Prior Based on MRF

The above model takes advantage of the spectral information only. In order to include spatial information, the MRF is an efficient tool for modeling the spatial prior. According to Hammersly–Clifford theory [36], Gibbs distribution is equivalent to MRF. Thus, a Gibbs distribution is used in this work to model the spatial priori in order to specify an MRF on the classification labels. It encourages neighboring pixels to be labeled into the same class in any direction and can lead to piecewise smoothness in the final classification map. The spatial priori is defined as follows:

$$p(\boldsymbol{y}) = \frac{1}{z}\exp\left\{-\sum_{|i-j|<\delta} |y_i - y_j|\right\} \tag{8}$$

where z is a normalization coefficient, i and j refer to pixel indexes. $\forall i \in \{1, 2, \ldots, N\}$, class label $y_i \in \{1, 2, \ldots, K\}$. δ is used to control the size of neighborhood. In this work, we take account only a small neighborhood of 2×2 pixels for simplicity. Thus, δ is set to be 1, unless otherwise specified.

D. Spatial–Spectral Classifier Based on Sparse Representation and MRF

Given the definitions in Sections II-B and II-C, our SSC-SRMRF can be summarized by the following expression:

$$\hat{\boldsymbol{y}} = \arg\min_{\boldsymbol{y}} \left\{-\sum_{i\in\{1,2,\ldots,N\}} \log p(\boldsymbol{x}_i\,|y_i) + \mu_s \sum_{|i-j|<\delta} |y_i - y_j|\right\} \tag{9}$$

where the first term is spectral fidelity term $p(x_i\,|y_i\,) = \sum_{k=1}^{l_k} \hat{s}_i^k$ and $\hat{\boldsymbol{s}}$ is the solution of model (7). The second term is *a priori* constraint of spatial correlation, which encourages the pixels in a neighborhood to belong to the same class in the spatial domain. Finally, μ_s is a smoothness parameter introduced to balance the two aforementioned terms.

E. Serial Algorithm

In order to obtain the final classification, we first use ADMM to solve the model (7) and estimate the abundance coefficients of the test samples. Then, we use the spatial–spectral model (9). Taking into consideration that class label y_i is a discrete integer, the MRF model (9) is difficult to solve. Recently, graph cuts [37]–[39], which are energy minimization algorithms, have been adopted to efficiently tackle this kind of optimization problems. The graph cut algorithm presented in [38] has computational complexity of polynomial order, and therefore we use it to solve model (9) in this work. Furthermore, taking into consideration that it is time-consuming to serially perform the classification per pixel, and that many optimized linear algebra libraries (MKL [40], CULA [41], and CUBLAS [42]) achieve high execution performance for matrix computing, especially for GPU parallel implementations [43], [44], we carefully orchestrate the pixel-wise algorithm as matrix batch computing by fitting all the test samples in a matrix for improving its execution performance. To sum up, the serial algorithm for the spatial–spectral classifier based on sparse representation and MRFs (referred to hereinafter as SSC-SRMRF-S) is given in Algorithm 1.

Algorithm 1. Serial algorithm of spatial–spectral classifier based on SR and MRF (SSC-SRMRF-S)

Input: Training samples set $\boldsymbol{A} \in \boldsymbol{R}^{L\times l}$, number of classes K, test samples set $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \boldsymbol{R}^{L\times N}$,
Initialization: Set $m = 0$, choose threshold t_1, t_2, maximum iteration M, filter function H_f, regularization parameters $\mu > 0$, $\lambda > 0$, $\mu_s > 0$, initialize $\boldsymbol{u}_0 = (0, 0, \ldots, 0)$, $\boldsymbol{d}_0 = (0, 0, \ldots, 0)$
Step 1. $\tilde{\boldsymbol{x}} = H_f * \boldsymbol{x}$
Step 2. Calculate $\hat{\boldsymbol{s}}$
Do:
Step 2.1. $\boldsymbol{H} = \boldsymbol{A}^T\boldsymbol{A} + (\mu - \lambda)\boldsymbol{I}$
Step 2.2. $\boldsymbol{w} = \boldsymbol{A}^T\tilde{\boldsymbol{x}} + \mu(\boldsymbol{u}^{(m)} + \boldsymbol{d}^{(m)})$
Step 2.3. $\boldsymbol{c} = \boldsymbol{H}^{-1}\boldsymbol{1}(\boldsymbol{1}^T\boldsymbol{H}^{-1}\boldsymbol{1})^{-1}$
Step 2.4. $\boldsymbol{s}^{(m+1)} = \boldsymbol{H}^{-1}\boldsymbol{w} - \boldsymbol{c}\boldsymbol{1}^T\boldsymbol{H}^{-1}\boldsymbol{w} + \boldsymbol{c}$
Step 2.5. $\boldsymbol{u}^{(m+1)} = \max(\boldsymbol{s}^{(m+1)} + \boldsymbol{d}^{(m)}, 0)$
Step 2.6. $\boldsymbol{d}^{(m+1)} = \boldsymbol{d}^{(m)} - (\boldsymbol{s}^{(m+1)} - \boldsymbol{u}^{(m+1)})$
Step 2.7. $r_1 = \left\|\boldsymbol{s}^{(m+1)} - \boldsymbol{u}^{(m+1)}\right\|_F$,
$r_2 = \left\|\boldsymbol{u}^{(m+1)} - \boldsymbol{u}^{(m)}\right\|_F$
Step 2.8. $m = m + 1$
While $r_1 \geq t_1$, $r_2 \geq t_2$, or $m < M$
Step 3. $\hat{p}(\boldsymbol{x}_i\,|y_i = k) = \sum_{j=1}^{l_k} \hat{s}_{ij}^k$, for $i = 1, 2, \ldots, N$

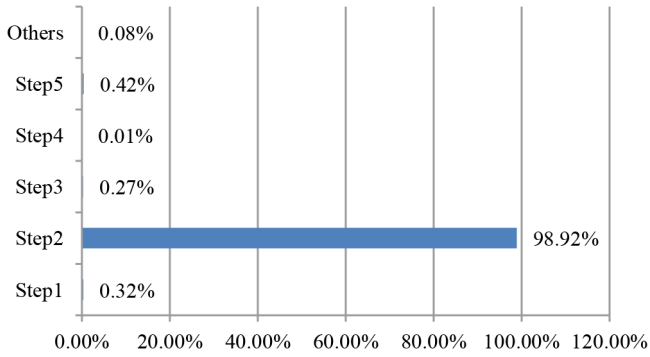


Fig. 2. Percentage of total CPU time consumed by different steps when processing the AVIRIS Indian Pines dataset.

Step 4. $\boldsymbol{P} = [\hat{p}(\boldsymbol{x}_1\,|y_1\,), \ldots, \hat{p}(\boldsymbol{x}_N\,|y_N\,)]$
Step 5. $\boldsymbol{y} = \mathrm{Graphcut}(\boldsymbol{P}, \mu_s)$
Output: $\boldsymbol{y}$ the Class labels of $\boldsymbol{x}$.
End

III. Parallel Implementation on CPU–GPU Heterogeneous Platform

In this section, we describe the parallel implementation of the spatial–spectral classifier with sparse representation and MRFs (referred to hereinafter as SSC-SRMRF-P) on a heterogeneous CPU–GPU platform, including how the overall computation is decomposed and scheduled onto the CPU and GPU, and how data transfers between the CPU and GPU memories are carried out.

A. Framework for the Parallel Implementation

Although the solution of SSC-SRMRF can be effectively computed based on the ADMM and graph cuts, it is still quite expensive in computational terms. Our theoretical analysis shows that the iterative solution for $\mathbf{s}$ in Step 2 of Algorithm 1 is the most time consuming part of the algorithm, as it includes heavy computations with big matrices. Furthermore, the experimental analysis of the serial algorithm execution using the AVIRIS Indian Pines hyperspectral dataset with 1043 training samples (details of the dataset will be given in Section IV-A) indicates that the computations involved in Step 2 take almost 99% of the total execution, as Fig. 2 shows. However, the calculation of Graphcut in Step 5 is quite efficient, and has been proven to exhibit polynomial computational complexity in [38]. Moreover, Fig. 2 reveals that the computation time of Graphcut represents less than 0.5% of the total execution time of Algorithm 1.

Taking into consideration that the CPU is efficient in performing logic control operation and small data computations, while the GPU architecture is more effective for massive data parallel computations with more structured memory access patterns, we assign the calculations related with high dimensional pixel vectors and big matrices to the GPU. Meanwhile, we allocate part of the computations operating on small data

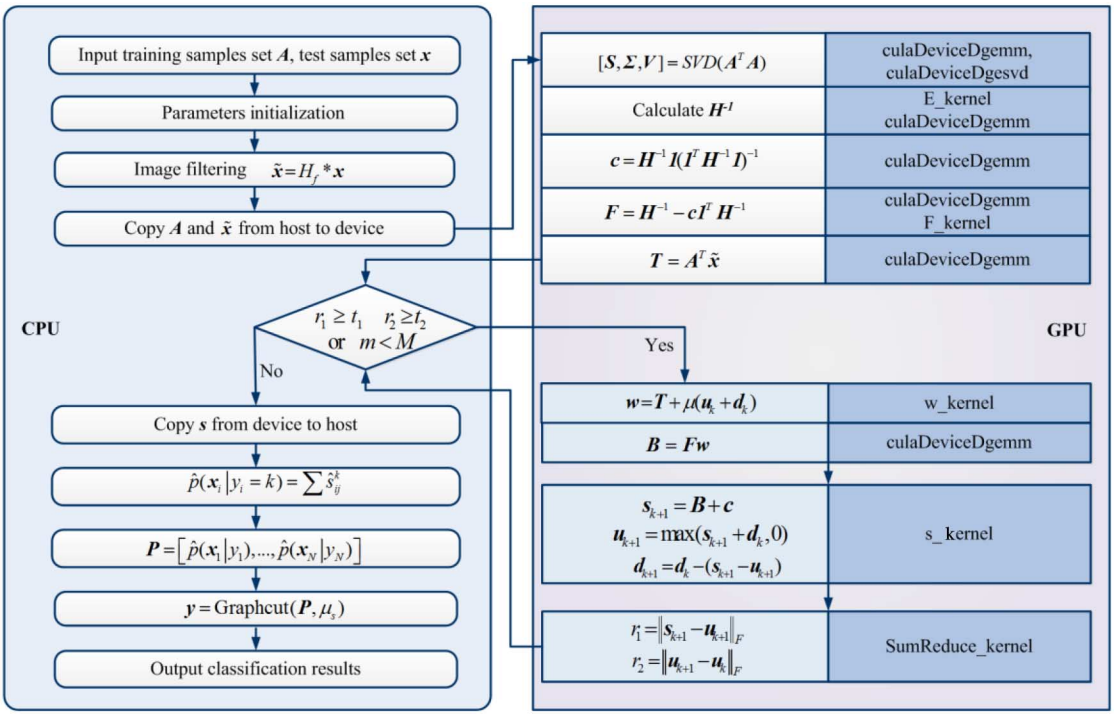


Fig. 3. Framework of the parallel spatial–spectral classifier based on sparse representation and MRF (SSC-SRMRF-P).

structures (such as Graphcut) to the CPU, so as to dramatically reduce the data transfer between the device (GPU) and the host (CPU). According to CUDA programming paradigms, when executing a function (or kernel) on the device (GPU), one has to allocate memory on it, transfer data from the host to the GPU, and finally transfer data back to the CPU, freeing the device memory. The data transfer is very time-consuming, so we try to minimize the number of data movements by optimizing the work allocation between the CPU and the GPU. The SSC-SRMRF-P is designed as illustrated graphically in Fig. 3.

In the following section, we describe the different steps and architecture-related optimizations carried out in the development of the proposed parallel implementation.

B. Details of the Parallel Implementation

The calculation of Step 2 in Algorithm 1 is a complicated iterative procedure, in which the matrix inversion of $\boldsymbol{H}$ inside the loop is the most complex operation. In the case of HSI classification, the labeled samples in matrix $\boldsymbol{A}$ are fixed. Thus, it is possible to precompute the singular value decomposition (SVD) of $\boldsymbol{A}^T\boldsymbol{A}$ (i.e. $\boldsymbol{A}^T\boldsymbol{A}=\boldsymbol{S}\cdot\boldsymbol{\Sigma}\cdot\boldsymbol{V}$), and then calculate $\boldsymbol{H}^{-1}=\boldsymbol{S}\cdot diag(1./(diag(\boldsymbol{\Sigma})+(\mu-\lambda)))\cdot\boldsymbol{V}$ with the same time complexity as a matrix multiplication operation, where *diag(*$\mathbf{x}$*)* denotes the main diagonal of $\mathbf{x}$, if $\mathbf{x}$ is a matrix, or a matrix with $\mathbf{x}$ as its main diagonal, if $\mathbf{x}$ is a vector. Therefore, this operation is ideally suited for GPU implementation as it avoids the inverse operation, which is very difficult to implement on GPUs (the same functionality can be obtained with simpler operations). For the purpose of efficient parallelization, the calculation of Step 2 can be modified as indicated in Algorithm 2.

Algorithm 2. Modification of Step 2 in Algorithm 1

Step 2.1. $[\boldsymbol{S},\boldsymbol{\Sigma},\boldsymbol{V}]=SVD(\boldsymbol{A}^T\boldsymbol{A})$
Step 2.2. $\boldsymbol{H}^{-1}=\boldsymbol{S}\cdot diag(1./(diag(\boldsymbol{\Sigma})+(\mu-\lambda)))\cdot\boldsymbol{V}$
Step 2.3. $\boldsymbol{c}=\boldsymbol{H}^{-1}\mathbf{1}(\mathbf{1}^T\boldsymbol{H}^{-1}\mathbf{1})^{-1}$
Step 2.4. $\boldsymbol{F}=\boldsymbol{H}^{-1}-\boldsymbol{c}\mathbf{1}^T\boldsymbol{H}^{-1}$
Step 2.5. $\boldsymbol{T}=\boldsymbol{A}^T\tilde{\boldsymbol{x}}$
Do:
Step 2.6. $\boldsymbol{w}=\boldsymbol{T}+\mu(\boldsymbol{u}_k+\boldsymbol{d}_k)$
Step 2.7. $\boldsymbol{B}=\boldsymbol{F}\boldsymbol{w}$
Step 2.8. $\boldsymbol{s}_{k+1}=\boldsymbol{B}+\boldsymbol{c}$
Step 2.9. $\boldsymbol{u}_{k+1}=\max(\boldsymbol{s}_{k+1}+\boldsymbol{d}_k,0)$
Step 2.10. $\boldsymbol{d}_{k+1}=\boldsymbol{d}_k-(\boldsymbol{s}_{k+1}-\boldsymbol{u}_{k+1})$
Step 2.11. $r_1=\|\boldsymbol{s}_{k+1}-\boldsymbol{u}_{k+1}\|_F$, $r_2=\|\boldsymbol{u}_{k+1}-\boldsymbol{u}_k\|_F$
Step 2.12. $k=k+1$
While $r_1\geq t_1, r_2\geq t_2$, or $k<M$

First, the matrix multiplication $\boldsymbol{A}^T\boldsymbol{A}$ in Step 2.1 of Algorithm 1 is realized using functions `culaDeviceDgemm`, which is included in the highly efficient GPU-accelerated linear algebra libraries of CULA [40], developed by EM photonics in partnership with NVIDIA. Then, the SVD of $\boldsymbol{A}^T\boldsymbol{A}$ is calculated by `culaDeviceDgesvd` of CULA in the GPU.

For the computation of Step 2.2, a kernel function called `E_kernel` is defined to carry out the operation $diag(1./(diag(\boldsymbol{\Sigma})+(\mu-\lambda)))$. Since the size of $\boldsymbol{\Sigma}$ is $l\times l$, we start a $l\times l$ thread grid on the GPU, and each thread takes charge of the calculation for one matrix element. The `THREAD_SIZE` and `BLOCK_SIZE` variables, respectively

denoting the number of processing *threads* and processing *blocks*, are set to 16×16 and $((l+16-1)/16) \times ((l+16-1)/16)$ in order to optimize the allocation of resources in the GPU. Immediately after finishing the execution of `E_kernel`, `culaDeviceDgemm` is invoked to complete the computation of Step 2.2.

The main operations from Step 2.3 to 2.7 are related with matrix multiplications, which can be efficiently performed by function `culaDeviceDgemm`. In addition, the subtraction operation in Step 2.4 and addition operation in Step 2.6 are implemented by kernel functions `F_kernel` and `w_kernel`, respectively, which conduct the mapping between thread and pixel on GPU, and each thread is responsible for the calculation related to one pixel.

Since the cost of I/O communication between the host (CPU) and the device (GPU) is quite expensive, the data transfers between the host and the device should be minimized. With this issue in mind, it is better to compute them on GPU, in spite of the simplicity of Steps 2.9 and 2.10. Furthermore, bearing in mind that $\boldsymbol{s}_{k+1}$, $\boldsymbol{u}_{k+1}$, and $\boldsymbol{d}_{k+1}$ are of the same size, we can perform a similar mapping between a thread and a pixel. Thus, it is reasonable to realize the Step 2.8, Step 2.9, and Step 2.10 together in a kernel function `s_kernel`.

In the procedure of calculating r_1 and r_2 for iteration termination condition, the main operation is the squared sum of matrix elements. This operation is efficiently realized as a kernel function `SumReduce_kernel` by a summation reduction model on the GPU, as illustrated Fig. 4. At the beginning, r_1 and r_2 are transferred from GPU global memory to shared memory; then, we calculate the first operation of subtraction and square, and synchronize the threads by means of `syncthreads()`. Then, the vectors are split into several segments with the same block size, and the summation of these segments is performed on the shared memory. During the circulation, adjacent threads operate with neighboring elements, thus avoiding bank conflicts. Therefore, we make effective use of shared memories, which act as small and very fast cache memories available for the processing elements within the same block, in order to speed up the execution. For the kernel `SumReduce_kernel`, every thread needs 12 registers, every block needs 8 kB of shared memory, and the size of the block is set to 1024. Taking into consideration that every streaming multiprocessor in NVIDIA Tesla C2075 (details of the platform will be given in Section IV-A) supports 1536 threads at most, one block can be launched, and the theoretical occupancy of `SumReduce_kernel` is $1024/1536 \approx 66.7\%$ in this case. Moreover, taking the AVIRIS Indian Pines hyperspectral dataset as an example, the practical occupancy of `SumReduce_kernel` calculated by the NVIDIA visual profiler is 66.3%.

Once the stopping condition is satisfied, we perform the memory copy of `s` from device to host, and the Graphcut procedure is executed on the CPU.

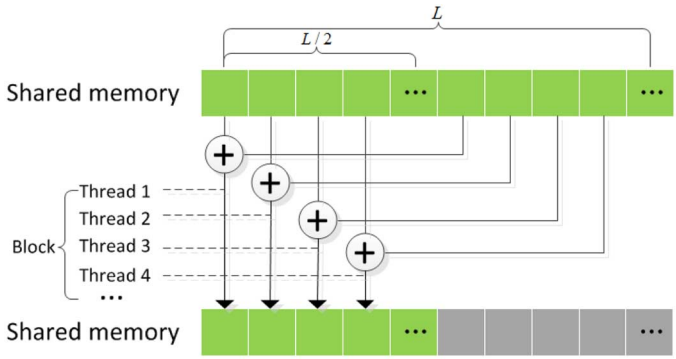


Fig. 4. Summation reduction model on the GPU.

TABLE I
HARDWARE SPECIFICATIONS AND COMPUTING CAPABILITIES OF THE CONSIDERED PLATFORMS

Specification		NVIDIA Tesla C2075 platform
CPU	Processor number	Intel Xeon E5-2609
	Processor base frequency	2.4 GHz
	Number of cores	8 in total (2 CPUs)
	Main memory	32 GB
GPU	Architecture	Fermi
	Frequency of CUDA cores	1150 MHz
	Number of CUDA cores	448
	Double precision floating point performance (peak)	515 Gflops
	Single precision floating point performance (peak)	1.03 Tflops
	Dedicated memory	6 GB
	Memory speed	1500 MHz
	Memory interface	384 bit
	Memory bandwidth	144 GB/sec
	CUDA compute capability (version)	2.0
	Threads/warp	32
	Maximum warps/SM	48
	Maximum threads/SM	1536
	Maximum blocks/SM	8
	32 bit registers/SM	32768

TABLE II
SOFTWARE SPECIFICATIONS OF THE CONSIDERED PLATFORMS

Specification	Version
OS	Windows 7 64 bit
CUDA version	5.5
CULA library	R17
MKL version	11.2
OpenMP	2.0
Compiler	Visual C++ 2010

C. Overall Approach of SSC-SRMRF-P

With the aforementioned observations in mind, a detailed step-by-step algorithm description of the parallel spatial–spectral classifier with sparse representation and MRFs on CPU–GPU heterogeneous platforms (SSC-SRMRF-P) is summarized in Algorithm 3.

TABLE III
CLASSIFICATION ACCURACIES (%) AND EXECUTION TIME OBTAINED FOR THE AVIRIS INDIAN PINES DATASET

Class	Class name	Training samples	Test samples	SSC-SRMRF-S	SSC-SRMRF-L	SSC-SRMRF-M	SSC-SRMRF-P
1	Alfalfa	6	48	94.79	94.17	94.17	94.17
2	Corn-notill	144	1290	96.71	96.70	96.79	96.78
3	Corn-min	84	750	94.51	94.51	94.57	94.53
4	Corn	24	210	94.76	94.67	94.71	94.81
5	Grass/pasture	50	447	97.67	97.65	97.52	97.47
6	Grass/trees	75	672	99.90	99.90	99.90	99.90
7	Grass/pasture-mowed	3	23	96.52	95.65	96.52	96.52
8	Hay-windrowed	49	440	99.73	99.70	99.70	99.73
9	Oats	2	18	85.56	85.56	85.56	85.56
10	Soybeans-notill	97	871	95.22	95.25	95.33	95.28
11	Soybeans-min	247	2221	98.51	98.51	98.53	98.53
12	Soybean-clean	62	552	96.70	96.74	96.74	96.76
13	Wheat	22	190	99.95	99.95	99.95	99.95
14	Woods	130	1164	99.48	99.47	99.46	99.46
15	Building-grass-trees-drives	38	342	96.17	96.20	96.17	96.17
16	Stone-steel towers	10	85	98.71	98.71	98.71	98.71
OA (%)		1043	9323	97.57	97.57	97.59	97.59
AA (%)				96.55	96.46	96.52	96.52
κ				0.9723	0.9723	0.9726	0.9725
Time (s)				4015.3	348.61	82.6	18.6
Acceleration factor (X)				–	11.52	48.6	215.9

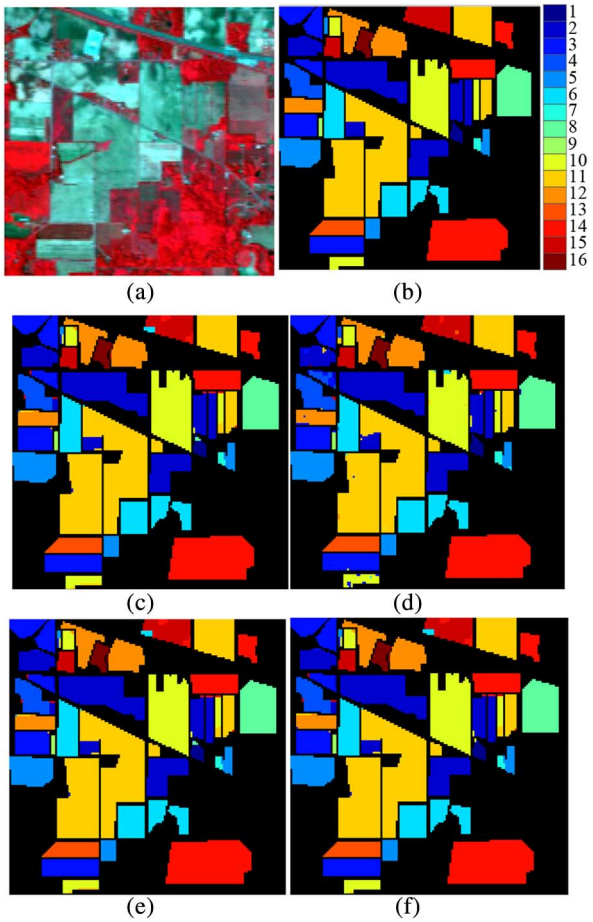


Fig. 5. Classification experiments with the AVIRIS Indian Pines dataset. (a) False color composition. (b) Ground truth as a collection of mutually exclusive classes. Some of the classification maps obtained after 10 Monte Carlo runs: (c) SSC-SRMRF-S. (d) SSC-SRMRF-L. (e) SSC-SRMRF-M. (f) SSC-SRMRF-P.

Algorithm 3. Parallel spatial–spectral classifier with SR and MRF on CPU–GPU heterogeneous platforms (SSC-SRMRF-P)

Input: Training samples set $\boldsymbol{A} \in \boldsymbol{R}^{L\times l}$, number of classes K, test samples set $\boldsymbol{x}= [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \boldsymbol{R}^{L\times N}$,
Initialization: Set $m = 0$, choose threshold t_1, t_2, maximum iteration M, filter function H_f, regularization parameters $\mu > 0$, $\lambda > 0$, $\mu_s > 0$, initialize $\boldsymbol{u}_0 = (0, 0, \ldots, 0)$, $\boldsymbol{d}_0 = (0, 0, \ldots, 0)$
Step 1. $\tilde{\boldsymbol{x}} = H_f * \boldsymbol{x}$
Step 2. Copy data from host to device
Step 3. Invoke `culaDeviceDgemm` and `culaDeviceDgesvd` to calculate $[\boldsymbol{S}, \boldsymbol{\Sigma}, \boldsymbol{V}] = SVD(\boldsymbol{A}^T\boldsymbol{A})$ on GPU
Step 4. Invoke kernel function `E_kernel` and `culaDeviceDgemm` to compute $\boldsymbol{H}^{-1}$ on GPU
Step 5. Invoke `culaDeviceDgemm` to compute $\boldsymbol{c} = \boldsymbol{H}^{-1}\boldsymbol{1}(\boldsymbol{1}^T\boldsymbol{H}^{-1}\boldsymbol{1})^{-1}$ on GPU
Step 6. Invoke `F_kernel` and `culaDeviceDgemm` to compute $\boldsymbol{F} = \boldsymbol{H}^{-1} - \boldsymbol{c}\boldsymbol{1}^T\boldsymbol{H}^{-1}$ on GPU
Step 7. Invoke `culaDeviceDgemm` to calculate $\boldsymbol{T} = \boldsymbol{A}^T\tilde{\boldsymbol{x}}$ on GPU
Do:
Step 8. Invoke kernel function `w_kernel` to calculate $\boldsymbol{w} = \boldsymbol{T} + \boldsymbol{\mu}(\boldsymbol{u}_k + \boldsymbol{d}_k)$ on GPU
Step 9. Invoke `culaDeviceDgemm` to calculate $\boldsymbol{B} = \boldsymbol{F}\boldsymbol{w}$ on GPU
Step 10. Invoke kernel function `s_kernel` to compute $\boldsymbol{s}_{k+1} = \boldsymbol{B} + \boldsymbol{c}$, $\boldsymbol{u}_{k+1} = \max(\boldsymbol{s}_{k+1} + \boldsymbol{d}_k, 0)$ and $\boldsymbol{d}_{k+1} = \boldsymbol{d}_k - (\boldsymbol{s}_{k+1} - \boldsymbol{u}_{k+1})$ on GPU
Step 11. Invoke kernel function `SumReduce_kernel` to calculate $r_1 = \|\boldsymbol{s}_{k+1} - \boldsymbol{u}_{k+1}\|_F$, and $r_2 = \|\boldsymbol{u}_{k+1} - \boldsymbol{u}_{k+1}\|_F$ on GPU
Step 12. $k = k + 1$

TABLE IV
CLASSIFICATION ACCURACIES (%) AND EXECUTION TIME OBTAINED FOR THE ROSIS PAVIA UNIVERSITY DATASET

Class	Class name	Training samples	Test samples	SSC-SRMRF-S	SSC-SRMRF-L	SSC-SRMRF-M	SSC-SRMRF-P
1	Asphalt	548	6304	96.73	96.70	96.75	96.61
2	Meadows	540	18 146	98.38	98.38	98.38	98.38
3	Gravel	392	1815	95.78	95.78	95.72	95.78
4	Trees	524	2912	99.29	99.29	99.17	99.29
5	Metal sheets	265	1113	100	100.00	100	100
6	Bare soil	532	4572	99.80	99.80	99.80	99.80
7	Bitumen	375	981	100	100.00	100	100
8	Bricks	514	3364	87.91	87.94	87.88	87.94
9	Shadows	231	795	98.60	98.60	99.02	99.02
OA (%)		3921	40 002	97.46	97.46	97.46	97.46
AA (%)				97.39	97.39	97.41	97.43
$\boldsymbol{\kappa}$				0.9654	0.9654	0.9654	0.9653
Time (s)				612 398.0	43 661.5	8896.4	2455.1
Acceleration factor (X)				–	14.03	68.8	249.4

While $r_1 \geq t_1, r_2 \geq t_2$ or $k < M$
Step 13. Copy $\boldsymbol{S}$ from device to host
Step 14. Compute $\hat{p}(\boldsymbol{x}_i|y_i = k) = \sum_{j=1}^{l_k} \hat{s}_{ij}^k$, for $i = 1, 2, \ldots, N$ on CPU
Step 15. Compute $\boldsymbol{P} = [\hat{p}(\boldsymbol{x}_1|y_1), \ldots, \hat{p}(\boldsymbol{x}_N|y_n)]$ on CPU
Step 16. Calculate $\boldsymbol{y} = \mathrm{Graphcut}(\boldsymbol{P}, \mu_s)$ on CPU
Output: $\boldsymbol{y}$, the Class labels of $\boldsymbol{x}$.
End

IV. PERFORMANCE EVALUATION

In this section, we present a summary of the performance tests that have been conducted in order to assess the computational performance and classification accuracy of the proposed method.

A. Experimental Configuration

The experimental platform used in our tests is a heterogeneous processor consisting of two CPUs and a GPU. The hardware specifications and computing capabilities, as well as the software specifications of the considered platforms, are respectively listed in Tables I and II.

We evaluate the classification accuracy and execution performance of SSC-SRMRF-P using two real HSIs for which ground truth information is available. The first HSI used in our experiments is the Indian Pines image, collected by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor over NW Indiana in June 1992. The AVIRIS sensor collects 220 spectral bands in the range from 0.4 to 2.5 μm. In our experiments, the number of bands is reduced to 200 by removing 20 water absorption bands. This scene has a spectral resolution of 10 nm and a spatial resolution of 20 m by pixel, and the size in pixels is 145×145. The ground truth contains 16 land cover classes and a total of 10 366 labeled pixels. The number of pixels in the smallest class is 20, while the number of pixels in the largest class is 2468.

The second hyperspectral scene used in our experiments is the University of Pavia image [45], which was acquired in 2001 by the Reflective Optics System Imaging Spectrometer (ROSIS), flown over the city of Pavia, Italy. The sensor collects 115 spectral bands ranging from 0.43 to 0.86 μm, and has a spatial resolution of 1.3 m per pixel. The image scene, with a size of 610×340 pixels, is centered at the University of Pavia and has 103 bands after removing 12 noisy bands. There are nine ground-truth classes of interest.

The classification accuracies are measured by the overall accuracy (OA), average accuracy (AA), and kappa statistic (k) [46]. The OA is computed as the ratio between the correctly classified test samples and the total number of test samples, and the AA is the mean of the accuracies across the different classes. The kappa statistic is computed by weighting the measured accuracies. It incorporates both of the diagonal and off-diagonal entries of the confusion matrix and is a robust measure of the degree of agreement.

In order to demonstrate the performance improvements between the parallel implementations on multicore CPU platform and our considered GPU platform, a multicore implementation of SSC-SRMRF (SSC-SRMRF-M) has been carried out following the design principles in [29] and using OpenMP API (Application Program Interface) and the MKL (Intel Math Kernel Library) [40] for matrix multiplications. Besides, in order to distinguish the effects of MKL and OpenMP, another version (SSC-SRMRF-L) has been implemented by using the functions in MKL library (which were not used in the serial version). The versions of SSC-SRMRF-S and SSC-SRMRF-L are executed on one core of the Intel Xeon E5-2609 CPU, and the multicore version (SSC-SRMRF-M) is run on eight cores of the two Intel Xeon E5-2609 CPUs. These versions of the SSC-SRMRF algorithm were implemented using the C++ programming language. All of the measurements reported in the following experiments are achieved by the adopted classifiers after ten Monte Carlo runs. According to [21] and our repeated experiments, the parameters were empirically set to be $\lambda = 0.01$, $\mu = 0.01$, and $\mu_s = 2$. A Gaussian filter with variance of 1 and window size of 3×3 was used as the filter function H_f.

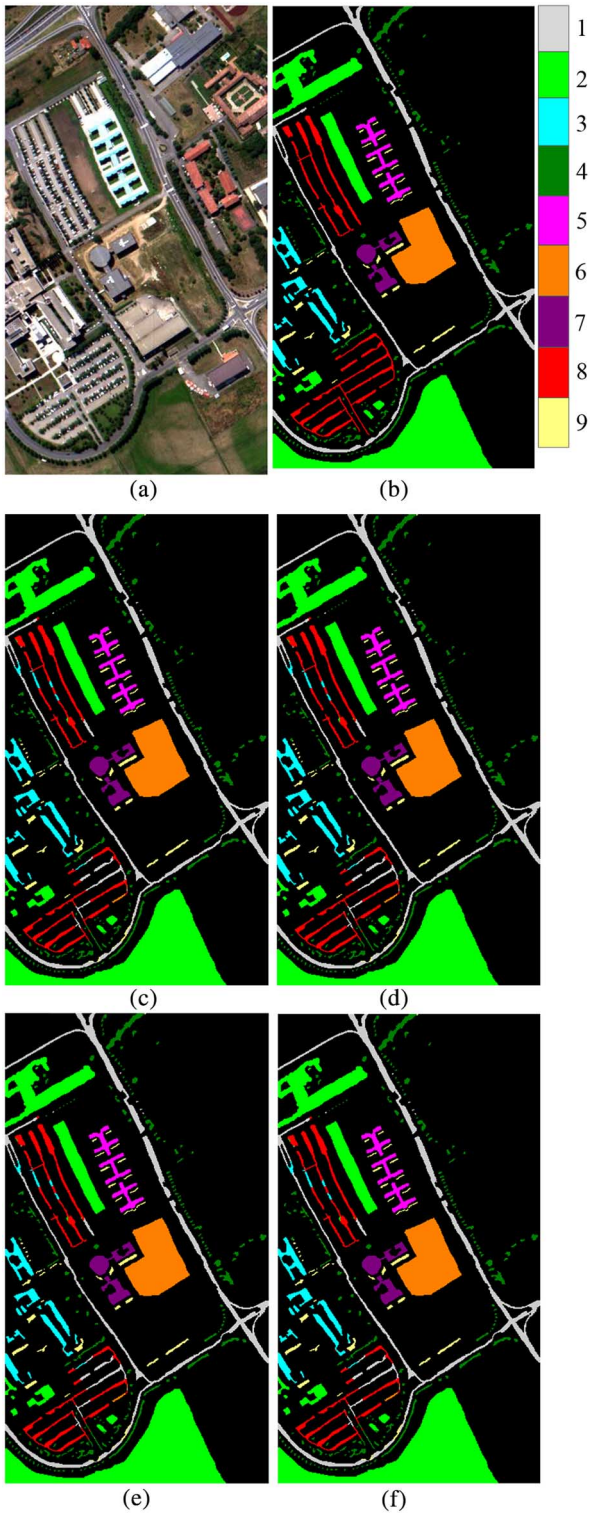


Fig. 6. Classification experiments with the ROSIS University of Pavia dataset. (a) False color composition. (b) Ground truth as a collection of mutually exclusive classes. Some of the classification maps obtained after 10 Monte Carlo runs: (c) SSC-SRMRF-S. (d) SSC-SRMRF-L. (e) SSC-SRMRF-M. (f) SSC-SRMRF-P.

B. Performance Assessment on AVIRIS Indian Pines Dataset

The first experiment was carried out using the AVIRIS Indian Pines dataset. Nearly 10% of the labeled pixels of each class (1043 pixels in total) were randomly chosen as training samples, and the remaining labeled pixels were used as test samples, as shown in Table I. The classification accuracies (OA, AA, and κ) are quantitatively shown in Table III, and graphically illustrated in Fig. 5. Table III indicates that the proposed SSC-SRMRF-S, SSC-SRMRF-L, SSC-SRMRF-M, and SSC-SRMRF-P obtain very similar classification accuracies, leading to smooth classification maps as depicted in Fig. 5.

The parallel implementation SSC-SRMRF-P achieves a remarkable acceleration factor of more than 210 relative to the serial version SSC-SRMRF-S. This is because the parallel version benefits from an efficient exploration of GPUs parallel capacities, as well as from the utilization of the highly efficient GPU-accelerated linear algebra libraries of CULA. As noted in Table III, the proposed parallel implementation of SSC-SRMRF-P is well below 20 s in processing time, including the loading times and the data transfer times from CPU to GPU and vice versa. This represents a significant improvement with regard to both serial and multicore versions.

C. Performance Assessment on ROSIS Pavia University Dataset

A second experiment was conducted on the ROSIS Pavia University dataset. In this experiment, nearly 9% of all labeled pixels were randomly chosen from each class as training samples, and the remaining labeled pixels are used as test samples. Table IV shows the obtained classification accuracies, whereas Fig. 6 shows some of the obtained classification maps after 10 Monte Carlo runs. The results indicate that the four implementations of SSC-SRMRF all obtain very competitive results. Here, it is also worth mentioning that they obtain very similar results in terms of classification accuracy. As a result, they can be considered approximately equivalent in terms of classification accuracy. The SSC-SRMRF-P achieved significant acceleration factors of almost 250x in the considered CPU–GPU heterogeneous architecture.

We now turn our attention to the execution efficiency. Although SSC-SRMRF-P gets significant acceleration factors on the Pavia University dataset, the execution time is still very high (almost 40 min). This makes it hard to adapt the algorithm to time-critical scenarios. Thus, it is important to find a balance between classification accuracy and execution performance.

In order to explore this issue, we carried out a third experiment to illustrate the impact of the size of the training set on both classification accuracy and execution performance. For this purpose, we consider different numbers of training samples, ranging from 40 to 120 of each class. Table V reports the processing times obtained for the three implementations of SSC-SRMRF on the considered CPU–GPU platform and for the ROSIS Pavia University scene. The measured times of each execution were always very similar. The reported times are mean of 10 executions in the platform.

As illustrated in Table V and Fig. 7, both the classification accuracy and the execution time of the implementations increase as the size of the training set increases. The acceleration factors tend to a plateau when the size of the training set increases, which is mainly related to the computing capabilities

TABLE V
EXECUTION TIMES AND ACCELERATION FACTORS OBTAINED FOR THE ROSIS PAVIA UNIVERSITY DATASET

DataSet	SSC-SRMRF-S	SSC-SRMRF-L		SSC-SRMRF-M		SSC-SRMRF-P					
						Total		Data transfer from host to device		Data transfer from device to host	
	Time (s)	Time (s)	Acceleration factor (X)	Time (s)	Acceleration factor (X)	Time (s)	Acceleration factor (X)	Time (s)	Percentage	Time (s)	Percentage
PaviaU (each class 20 training)	1113.69	171.931	6.48	94.36	11.80	15.68	71.04	0.10	0.62%	0.04	0.24%
PaviaU (each class 40 training)	4404.70	495.425	8.89	212.17	20.76	34.37	128.14	0.13	0.38%	0.07	0.22%
PaviaU (each class 60 training)	9584.03	991.034	9.67	346.85	27.63	62.15	154.22	0.20	0.31%	0.10	0.16%
PaviaU (each class 80 training)	19244.03	1814.66	10.60	490.92	39.20	99.28	193.83	0.28	0.28%	0.14	0.14%
PaviaU (each class 100 training)	29094.58	3070.46	9.48	663.93	43.82	149.29	194.88	0.40	0.27%	0.19	0.13%
PaviaU (each class 120 training)	41185.80	4038.4	10.20	905.93	45.46	202.06	203.83	0.53	0.26%	0.23	0.12%

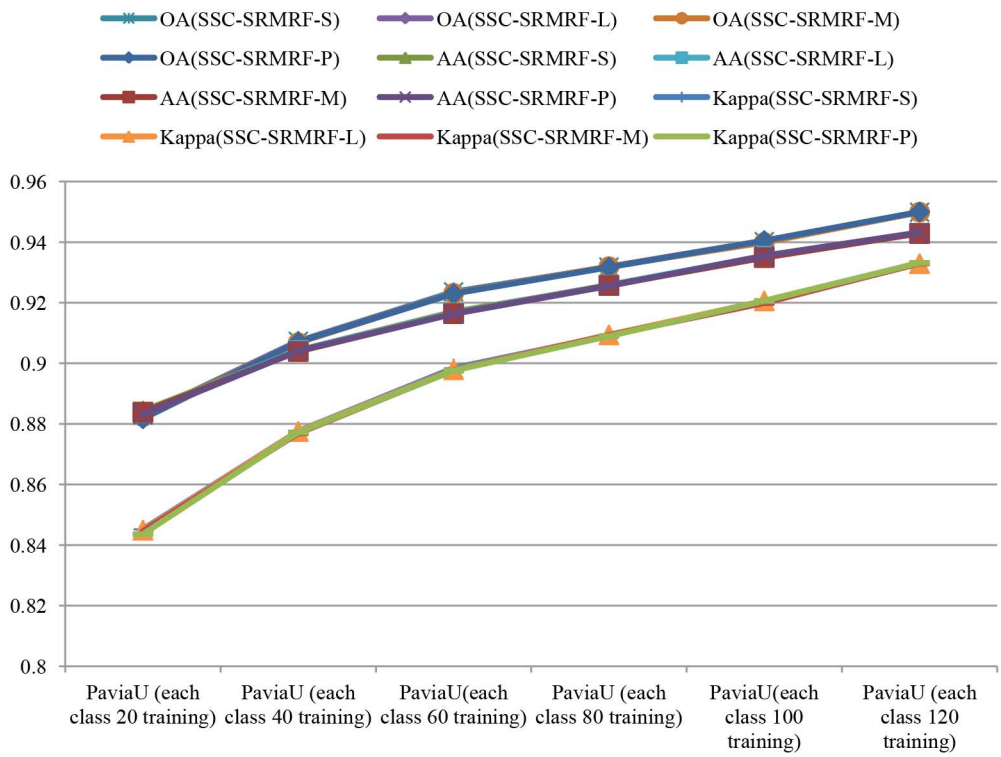


Fig. 7. OA, AA, and kappa statistic obtained after using different training sizes on the ROSIS Pavia University dataset.

of the considered platforms. However, the SSC-SRMRF-P provides more significant acceleration factors as the size of training set becomes larger. This is an important consideration, as it indicates that the proposed implementation scales better as the problem size becomes larger. Let us now take a closer look at the I/O communications between the host and the device, which often becomes the main bottleneck of parallel systems. As shown in Table V, the time required for data transfers between the host and the device contributes very little to the overall execution time of our SSC-SRMRF-P. In all the cases, less than 0.9% of the total solver time is consumed by data transfers. The most significant portion of the time taken by our parallel implementation is the pure computation steps, which is ideal in terms of parallel efficiency. As a result, the performance bottleneck becomes the processing units themselves and the algorithm is expected to scale very effectively for larger size problems.

Last but not least, the average acceleration factor of the parallel SSC-SRMRF-P reaches more than 150× with regards to the serial version SSC-SRMRF-S, while the multicore version SSC-SRMRF-M only gets an average acceleration factor of 31×. This is a quite remarkable parallelization result that proves the significant computing performance improvement that can be gained by our proposed implementation while retaining the similar classification accuracies obtained by the original algorithm.

V. Conclusion and Future Work

In this work, we have developed an efficient parallel implementation of a spatial–spectral HSI classifier with sparse representation and MRFs. The proposed implementation achieved significant acceleration factors on a CPU–GPU heterogeneous platform, a kind of low-weight hardware platform that offers a tremendous potential to bridge the gap toward real-time analysis of remotely sensed hyperspectral data. The parallel implementation of the proposed method has been carried out using the CUDA. The proposed implementations have been evaluated not only in terms of classification accuracy, but also in terms of computational performance, using an NVIDIA Tesla C2075 GPU connected to two Intel Xeon E5-2609 CPUs. The experimental results reported in this work reveal considerable acceleration factors while retaining very similar classification accuracies obtained by the proposed algorithm. It is felt that this is an important contribution, as there have been not many hyperspectral imaging classification techniques optimized in GPU architectures so far. The results reported in this work are very encouraging, but the proposed implementation is still far from real-time performance, although significantly faster than most other available techniques for HSI classification. In the future, we will explore implementations on more performing GPU core units that already are or will be available on the market in order to further pursue the long desired goal of achieving HSI classification in real time, which will be important for many remote sensing techniques and applications.

References

[1] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proc. IEEE*, vol. 101, no. 3, pp. 652–675, Mar. 2013.

[2] A. Plaza *et al.*, "Recent advances in techniques for hyperspectral image processing," *Remote Sens. Environ.*, vol. 113, Suppl. 1, pp. 110–122, Sep. 2009.

[3] A. Plaza, J. M. Bioucas-Dias, A. Simic, and W. J. Blackwell, "Foreword to the special issue on hyperspectral image and signal processing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 2, pp. 347–353, Apr. 2012.

[4] F. Melgani and L. Bruzzone, "Classification of hyperspectral remote sensing images with support vector machines," *IEEE Trans. Geosci. Remote Sens.*, vol. 42, no. 8, pp. 1778–1790, Aug. 2004.

[5] Y. Tarabalka, M. Fauvel, J. Chanussot, and J. A. Benediktsson, "SVM-and MRF-based method for accurate classification of hyperspectral images," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 4, pp. 736–740, Apr. 2010.

[6] B. Zhang, S. Li, X. Jia, L. Gao, and M. Peng, "Adaptive Markov random field approach for classification of hyperspectral imagery," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 5, pp. 973–977, May 2011.

[7] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles," *IEEE Trans. Geosci. Remote Sens.*, vol. 46, no. 11, pp. 3804–3814 , Nov. 2008.

[8] Y. Gu *et al.*, "Representative multiple kernel learning for classification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 7, pp. 2852–2865, Jul. 2012.

[9] X. Sun, Q. Qu, N. M. Nasrabadi, and T. D. Tran, "Structured priors for sparse-representation-based hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 7, pp. 1235–1239, Jul. 2014.

[10] J. Liu, Z. Wu, Z. Wei, L. Xiao, and L. Sun, "Spatial-spectral kernel sparse representation for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 6, pp. 2462–2471, Dec. 2013.

[11] G. Camps-Valls, T. Bandos Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 10, pp. 3044–3054, Oct. 2007.

[12] G. Camps-Valls, N. Shervashidze, and K. M. Borgwardt, "Spatio-spectral remote sensing image classification with graph kernels," *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 4, pp. 741–745, Apr. 2010.

[13] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral–spatial hyperspectral image segmentation using subspace multinomial logistic regression and Markov random fields," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 3, pp. 809–823, Mar. 2012.

[14] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 957–968, Jun. 2005.

[15] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Spectral–spatial classification of hyperspectral data using loopy belief propagation and active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 2, pp. 844–856, Feb. 2013.

[16] J. Li, J. M. Bioucas-Dias, and A. Plaza, "Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 11, pp. 4085–4098, Nov. 2010.

[17] J. Li, P. Reddy Marpu, A. Plaza, J. Bioucas-Dias, and J. Atli Benediktsson, "Generalized composite Kernel framework for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 9, pp. 4816–4829, Sep. 2013.

[18] Y. Tarabalka, J. Chanussot, and J. A. Benediktsson, "Segmentation and classification of hyperspectral images using watershed transformation," *Pattern Recog.*, vol. 43, no. 7, pp. 2367–2379, Jul. 2010.

[19] L. Xu and J. Li, "Bayesian classification of hyperspectral imagery based on probabilistic sparse representation and Markov random field," *IEEE Geosi. Remote Sens. Lett.*, vol. 11, no. 4, pp. 823–827, Apr. 2014.

[20] R. Ji *et al.*, "Spectral-spatial constraint hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 3, pp. 1811–1824, Mar. 2014.

[21] L. Sun, Z. Wu, J. Liu, and Z. Wei, "A novel supervised method for hyperspectral image classification with spectral-spatial constraints," *Chin. J. Electron.*, vol. 22, no. 3, pp. 1–7, Jul. 2013.

[22] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification via kernel sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 217–231, Jan. 2013.

[23] X. Kang, S. Li, and J. Benediktsson, "Spectral-spatial hyperspectral image classification with edge-preserving filtering," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 5, pp. 2666–2677, May 2014.

[24] Y. Chen, N. M. Nasrabadi, and T. D. Tran, "Hyperspectral image classification using dictionary-based sparse representation," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3973–3985, Oct. 2011.

[25] C. A. Lee *et al.*, "Recent developments in high performance computing for remote sensing: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 508–527, Sep. 2011.

[26] A. Plaza, Q. Du, Y. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 3, pp. 528–544, Sep. 2011.

[27] E. Christophe, J. Michel, and J. Inglada, "Remote sensing processing: From multicore to GPU," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 643–652, Sep. 2011.

[28] C. Gonzalez, D. Mozos, J. Resano, and A. Plaza, "FPGA implementation of the N-FINDR algorithm for remotely sensed hyperspectral image analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 2, pp. 374–388, Feb. 2012.

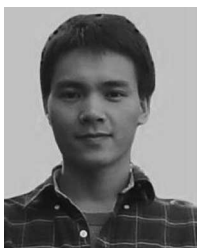
[29] S. Bernabe *et al.*, "Hyperspectral unmixing on GPUs and multi-core processors: A comparison," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 3, pp. 1386–1398, Mar. 2013.

[30] S. Bernabe, S. Lopez, A. Plaza, and R. Sarmiento, "GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 221–225, Mar. 2013.

[31] X. Wu, B. Huang, A. Plaza, Y. Li, and C. Wu, "Real-time implementation of the pixel purity index algorithm for endmember identification on GPUs," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 5, pp. 955–959, May 2013.
[32] J. M. P. Nascimento, J. M. Bioucas-Dias, J. M. Rodriguez Alves, V. Silva, and A. Plaza, "Parallel hyperspectral unmixing on GPUs," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 3, pp. 666–670, Mar. 2014.
[33] A. Barberis, G. Danese, F. Leporati, A. Plaza, and E. Torti, "Real-time implementation of the vertex component analysis algorithm on GPUs," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 251–255, Feb. 2013.
[34] J. M. Bioucas-Dias and M. A. T. Figueiredo, "Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing," in *Proc. 2nd Workshop WHISPERS*, 2010, pp. 1–4.
[35] W. Deng, W. Yin, and Y. Zhang, "Group sparse optimization by alternating direction method," Dept. of Comput. Appl. Math., Rice Univ., Tech Rep. TR11-06, 2011.
[36] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 6, pp. 721–741, Nov. 1984.
[37] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.
[38] M. Schmidt and K. Alahari, "Generalized fast approximate energy minimization via graph cuts: Alpha-Expansion Beta-Shrink moves," in *Proc. 27th Conf. Uncertainty Artif. Intell.*, Barcelona, Spain, Jul. 2011.
[39] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.
[40] Intel Developer Zone. (2014, Jun.) *Intel Math Kernel Library Reference Manual* [Online]. Available: https://software.intel.com/sites/products/documentation/hpc/mkl/mklman/
[41] EM Photonics. (2014, Jun.). *CULA Programmer's Guide* [Online]. Available: http://www.culatools.com/cula_dense_programmers_guide/
[42] NVIDIA Developer Zone. (2014, Jun.). *cuBLAS User Guide* [Online]. Available: http://docs.nvidia.com/cuda/cublas/index.html
[43] M. Anderson, D. Sheffield, and K. Keutzer, "A predictive model for solving small linear algebra problems in GPU registers," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2012, pp. 2–13.
[44] J. M. Molero, E. M. Garzón, I. García, E. S. Quintana-Ortí, and A. Plaza, "Efficient implementation of hyperspectral anomaly detection techniques on GPUs and multicore processors," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2256–2266, Jun. 2014.
[45] L. Sun, Z. Wu, J. Liu, L. Xiao, and Z. Wei, "Supervised spectral–spatial hyperspectral image classification with weighted Markov random fields," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1490–1503, Mar. 2015.
[46] G. M. Foody, "Classification accuracy comparison: Hypothesis tests and the use of confidence intervals in evaluations of difference, equivalence and non-inferiority," *Remote Sens. Environ.*, vol. 113, no. 8, pp. 1658–1663, Aug. 2009.

Zebin Wu (M'13) was born in Zhejiang, China, in 1981. He received the B.Sc. and Ph.D. degrees in computer science from Nanjing University of Science and Technology (NUST), Nanjing, China, in 2003 and 2007, respectively.

He is currently an Associate Professor with the School of Computer Science and Engineering, NUST, and also a Visiting Scholar at the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, Cáceres, Spain. His research interests include hyperspectral image processing, high performance computing, and computer simulation.



Qicong Wang was born in Henan, China, in 1988. He received the B.Sc. degree in computer science and technology from the School of Computer, Henan University of Science and Technology, Luoyang, China, in 2012. He is currently pursuing the M.S. degree at the School of Computer Science and Engineering, Nanjing University of Science and Technology.

His research interests are hyperspectral image classification and high performance computing.

Antonio Plaza (M'05–SM'07–F'15) was born in Caceres, Spain, in 1975. He received the computer engineer degree in 1997, the M.Sc. degree in 1999, and the Ph.D. degree in 2002, all in computer engineering.

He is an Associate Professor (with accreditation for Full Professor) with the Department of Technology of Computers and Communications, University of Extremadura, Cáceres, Spain, where he is the Head of the Hyperspectral Computing Laboratory (HyperComp). He has been the advisor of 12 Ph.D. dissertations and more than 30 M.Sc. dissertations. He was the Coordinator of the Hyperspectral Imaging Network, a European project with total funding of 2.8 million Euro. He authored more than 400 publications, including 130 JCR journal papers (82 in IEEE journals), 20 book chapters, and over 240 peer-reviewed conference proceeding papers (94 in IEEE conferences). He has edited a book: *High-Performance Computing in Remote Sensing* (CRC Press/Taylor & Francis) (the first book on this topic in the published literature) and guest edited 8 special issues on hyperspectral remote sensing for different journals. He has reviewed more than 500 articles for over 50 different journals. His research interests include remotely sensed hyperspectral image analysis and efficient implementations of large-scale scientific problems on high performance computing architectures.

Dr. Plaza is an Associate Editor for IEEE Access, and was a member of the Editorial Board of the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER (2011–2012) and the IEEE GEOSCIENCE AND REMOTE SENSING MAGAZINE (2013). He was also a member of the steering committee of THE IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (JSTARS). He is currently an Associate Editor for the *Journal of Real-Time Image Processing*. He served as the Director of Education Activities for the IEEE Geoscience and Remote Sensing Society (GRSS) from 2011 to 2012, and has been served as a President of the Spanish Chapter of IEEE GRSS (since November 2012). He has served as a Proposal Evaluator for the European Commission (Marie Curie Actions, Engineering Panel), the European Space Agency, the Belgium Science Policy, the Israel Science Foundation, and the Spanish Ministry of Science and Innovation. He has participated in the Tenure Track Selection Committee of different Universities in Italy, Spain and Australia. He is also currently serving as the Editor-in-Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. He was the recipient of the recognition of Best Reviewers of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS (2009), the recognition of Best Reviewers of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (2010), a journal for which he served as Associate Editor from 2007 to 2012, the 2013 Best Paper Award of the JSTARS journal, and the most highly cited paper (2005–2010) in the *Journal of Parallel and Distributed Computing*. He is a co-author of the 2011 Best Student Paper at the IEEE International Conference on Space Technology, and the recipient of the 2008 Best Paper Award at the IEEE Symposium on Signal Processing and Information Technology. He was the recipient of the Best Ph.D. Dissertation Award at University of Extremadura in 2002. .

Jun Li (M'13) received the B.S. degree in geographic information systems from Hunan Normal University, Changsha, China, in 2004, the M.E. degree in remote sensing from Peking University, Beijing, China, in 2007, and the Ph.D. degree in electrical engineering from the Instituto de Telecomunicações, Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Lisbon, Portugal, in 2011.

From 2007 to 2011, she was a Marie Curie Research Fellow with the Departamento de Engenharia Electrotécnica e de Computadores and the Instituto de Telecomunicações, IST, Universidade Técnica de Lisboa, in the framework of the European Doctorate for Signal Processing (SIGNAL). She has also been actively involved in the Hyperspectral Imaging Network, a Marie Curie Research Training Network involving 15 partners in 12 countries and intended to foster research, training, and cooperation on hyperspectral imaging at the European level. Since 2011, she has been a Postdoctoral Researcher with the Hyperspectral Computing Laboratory, Department of Technology of Computers and Communications, Escuela Politécnica, University of Extremadura, Caceres, Spain. Her research interests include hyperspectral image classification and segmentation, spectral unmixing, signal processing, and remote sensing.

Dr. Li has been a Reviewer of several journals, including the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, *Pattern Recognition*, *Optical Engineering*, *Journal of Applied Remote Sensing*, and *Inverse Problems and Imaging*. She was therecipent of the 2012 Best Reviewer Award of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING.

Le Sun (M'13) was born in Jiangsu, China, in 1987. He received the B.S. degree in applied mathematics from the School of Science, Nanjing University of Science and Technology, Nanjing, China, in 2009, where he is currently pursuing the Ph.D. degree at the School of Computer Science and Engineering.

His research interests include spectral unmixing, hyperspectral classification, image processing, sparse representation, and compressive sensing.

Zhihui Wei was born in Jiangsu, China, in 1963. He received the B.Sc. degree in applied mathematics, the M.Sc. degree in applied mathematics, and the Ph.D. degree in communication and information system from South East University, Nanjing, China, in 1983, 1986, and 2003, respectively.

He is currently a Professor and Doctoral Supervisor with Nanjing University of Science and Technology (NUST), Nanjing, China. His research interests include partial differential equations, image processing, multiscale analysis, sparse representation, and compressed sensing.