Journal of
Applied Remote Sensing

# Parallel positive Boolean function approach to classification of remote sensing images

Yang-Lang Chang
Tung-Ju Hsieh
Antonio Plaza
Yen-Lin Chen
Wen-Yew Liang
Jyh-Perng Fang
Bormin Huang

SPIE

# Parallel positive Boolean function approach to classification of remote sensing images

**Yang-Lang Chang,[a,*] Tung-Ju Hsieh,[b] Antonio Plaza,[c] Yen-Lin Chen,[b] Wen-Yew Liang,[b] Jyh-Perng Fang,[a] and Bormin Huang[d]**

[a] National Taipei University of Technology, Department of Electrical Engineering, No. 1, Sec. 3, Chung-Hsiao E. Road, Taipei, 10608 Taiwan
ylchang@ntut.edu.tw

[b] National Taipei University of Technology, Department of Computer Science and Information Engineering, No. 1, Sec. 3, Chung-Hsiao E. Road, Taipei, 10608 Taiwan

[c] University of Extremadura, Department of Technology of Computers and Communications, Escuela Politecnica, 10003 Caceres, Spain

[d] University of Wisconsin-Madison, Cooperative Institute for Meteorological Satellite, Studies Space Science and Engineering Center, Madison, Wisconsin 53706

**Abstract.** We present a parallel image classification approach, referred to as the parallel positive Boolean function (PPBF), to multisource remote sensing images. PPBF is originally from the positive Boolean function (PBF) classifier scheme. The PBF multiclassifier is developed from a stack filter to classify specific classes of land covers. In order to enhance the efficiency of PBF, we propose PPBF to reduce the execution time using parallel computing techniques. PPBF fully utilizes the significant parallelism embedded in PBF to create a set of PBF stack filters on each parallel node based on different classes of land uses. It is implemented by combining the message-passing interface library and the open multiprocessing (OpenMP) application programing interface in a hybrid mode. The experimental results demonstrate that PPBF significantly reduces the computational loads of PBF classification.  [DOI: 10.1117/1.3626866]

**Keywords:** parallel positive Boolean function; open multiprocessing; message-passing interface.

Paper 11078SSR received May 5, 2011; revised manuscript received Jul. 9, 2011; accepted for publication Aug. 2, 2011; published online Dec. 1, 2011.

## 1 Introduction

State-of-the-art sensors can make use of a growing number of spectral bands. This recent technology finds applications in many fields, such as satellite-based geospatial technology, monitoring systems, medical imaging, and industrial product inspection. Although high volumes of multisource remote sensing images are continuously being acquired and archived, existing methods have been shown inadequate for analyzing such large volumes of data. Besides, the demands for higher classification accuracy of remote sensing images have also necessitated an increasing use of different information collected from different sources. Therefore, huge volumes of multisource data should be combined so as to yield strengthened capabilities for land-cover classification.[1] On the other hand, there is still a need for satellite sensor networks to maintain a formation that moves through a swarm for future earth observation. Data fusion methods thus serve as a foundation for thematic mapping in the context of a distributed geo-information network, such as the multiple sources of sensor swarms, allowing large volumes of

*Corresponding author.

1931-3195/2011/$25.00 © 2011 SPIE

each data type to be analyzed by the most appropriate procedures.[2] Without the support of new scientific concepts and novel technological methods, these large-scale and widely distributed data would restrain any systematic exploitation. As a result, a vital demand for new concepts and techniques is of importance in treating the fusion of high-dimensional data sets.

With this diversity of high volumes of multisource data, high-performance computing (HPC) is needed to acquire and fuse data from different sources, and further to obtain more accurate interpretations of the acquired information. Currently, HPC can handle large volumes of widely distributed remote sensing data sets and has necessitated the making of local decisions that are eventually conveyed to a master node prior to the making of global decisions. Many efforts have been devoted to the high-performance parallel computing in the remote sensing community, as witnessed both in the forms of hardware performance development[3–7] and software algorithm improvements.[8–13]

In this regard, this paper presents an alternative promising HPC concept, known as the parallel positive Boolean function (PPBF), which adopts a novel parallel approach to the supervised classification of remote sensing images. The PPBF is proposed in accordance with the positive Boolean function (PBF) classifier, which has been successfully applied to hyperspectral image classifications[14] and multisource data fusions.[15,16] The PBF classifier is developed from a stack filter. It implements the minimum classification error (MCE)[17] as a criterion to improve classification performance. MCE minimizes the expected error rate and gives an arbitrary combination of the distribution function. The goal of MCE learning is to discriminate the observations correctly for best classification rather than to fit the sample distributions well. Compared to the distribution estimation method, MCE shows a significant improvement in terms of classification accuracy.[17] Each stack filter corresponding to a PBF possesses the well-known weak superposition property (known as the threshold decomposition property) and the ordering property (also known as the stacking property).[16,18] These special characteristics make the PBF classifier possible to be implemented in parallel in terms of different land-use classes. In this paper, the proposed PPBF algorithm aims at decomposing the computations of the PBF method into successively smaller PBF stack filters implemented with the message-passing interface (MPI) library[19] and the open multiprocessing (OpenMP) application programing interface (API)[20] in a hybrid mode. The experimental results show that PPBF can significantly reduce the computational burden of PBF classification. The rest of this paper is organized as follows. Section 2 describes the proposed PPBF method in details. Section 3 demonstrates a set of experiments that validate the feasibility and utility of the proposed approach. In Sec. 4, several conclusions are presented.

## 2 Methodology

In this paper, a realization of PPBF, namely a hybrid-PPBF (combining MPI and OpenMP) scheme,[21] is examined to validate the flexibility of proposed PPBF scheme. It is implemented to completely combine MPI library and OpenMP API and fully utilize the coarse-grain parallelism among multicomputers and the fine-grain parallelization of multiprocessors. In parallel computing, granularity is defined as the ratio of computation to the amount of communication. In fine-grain parallelism, each parallel task is relatively small in terms of code size and data are transferred among parallel units frequently. The finer the granularity is, the greater the potential for parallelism. However, it incurs more synchronization and communication overhead compared to coarse-grain parallelism. The proposed PPBF multiclassification scheme is presented in two phases. They are PPBF learning (training) and PPBF classification phases, as shown in Fig. 1. Note that the flows of the PPBF learning and classification phases shown in the related figures are indicated as black and white arrows, respectively. Because of the symmetrical properties of the stack filters implemented in PBF, the proposed PPBF can perfectly partition them into well load-balanced parallel tasks by each distinct class $\omega_k$ of labeled and unlabeled samples.[16] This partitioning scheme can be adapted to two domains of both spectral and spatial partitions, especially for hyperspectral images and the fusions of multisource remote sensing images.

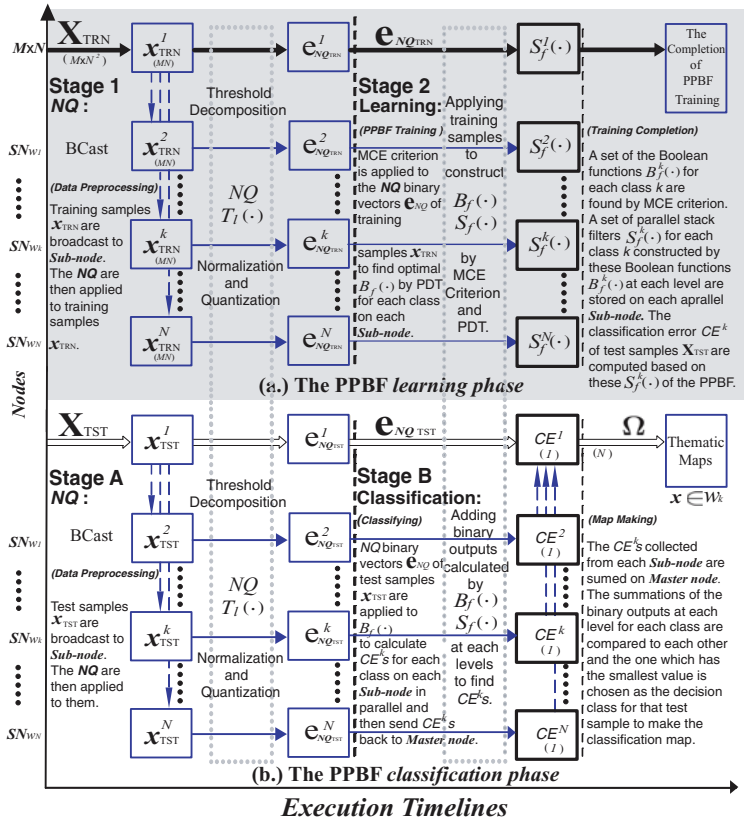


**Fig. 1** Parallel mechanism of the PPBF method: (a) The training phase and (b) classification phase of proposed PPBF method.

The principle of parallelization is generally focused on how to maximize processor utilizations, minimize communication overheads, and balance computational burdens. Foster proposed a theoretical framework for designing a parallel algorithm.[22] The methodology structures the design process as four distinct steps, namely, partitioning, communication, agglomeration, and mapping (PCAM). ($i$) In the partitioning step, the computational tasks and data are first recognized and divided into parallel partitions. ($ii$) Data sharing between computational tasks is analyzed in the communication step. ($iii$) In the agglomeration step, the previously defined task and communication structures are evaluated. If necessary, the computational tasks are then grouped into larger tasks to improve the performances of parallelism. ($iv$) Each task is finally assigned to a processor or a thread to balance the work loads in the mapping step.

On the basis of the PCAM methodology, a sophisticated parallelization analysis for the load balancing is evaluated prior to the mechanism design of the proposed PPBF. Hence, the designed PPBF learning phase in this paper can be divided into two stages as shown in Fig. 1(a). The first stage (stage 1) is a threshold decomposition, also known as normalization and quantization ($NQ$). On master node (MN), the training samples $\mathbf{X}_{\mathrm{TRN}} = (x^k_{\mathrm{TRN}_1}, \ldots, x^k_{\mathrm{TRN}_l}, \ldots, x^k_{\mathrm{TRN}_n})$ of different classes $\omega_k$, $k \in \{1, \ldots, N\}$, are first distributed to each different parallel subnode (SN) $\omega_k$. Then the threshold decomposition is executed to normalize and quantize the scales of these training samples $\mathbf{X}_{\mathrm{TRN}}$ of multiple sources in each SN for the data preprocessing prior to PPBF classification. In stage 2 (Learning), PPBF learning is performed to obtain the optimal Boolean functions $B^k_f(\cdot)$ for each class on different SNs in parallel. There are also two stages in the PPBF classification phase. The first stage (stage A) is functionally similar to stage 1 ($NQ$) of PPBF learning phase. It is a threshold decomposition stage that normalizes and quantizes the scales of the test samples $\mathbf{X}_{\mathrm{TST}}$ of different classes in different SNs. In stage B

(Classification), the classification processes of the proposed PPBF are performed in parallel for multiclassification on each different SNs and, finally, the classification map is determined on the MN. The methodology of the proposed PPBF is illustrated in detail in Secs. 2.1–2.3.

### 2.1 Implementation of Hybrid Parallel Positive Boolean Function

Parallel computers can be categorized into two classes according to the organization of the system. The first one is the distributed-memory multiprocessors or multicomputers (clusters). MPI is the most commonly used programing environment in such systems. Because of the high communication expenses in the distributed cluster environment, coarse granularity design is typically used by the MPI programs to reduce the communication-to-computation ratios. A quantitative characterization of communication overhead becomes crucial for making performance-optimization decisions. Hence, in this regard, the major design issues of the MPI-PPBF rest on how to minimize the communication delays on the MPI clusters. The second class of parallel computers is the shared-memory multiprocessors, such as symmetric multiprocessors (SMP) and multicore computers, also known as chip multiprocessors (CMP). Because shared-memory systems have lower communication overhead, parallel programs can be written with a finer granularity. OpenMP is the most frequently used mechanism for developing parallel program in this environment. For CMP, the effects of computation-to-synchronization ratios will be particularly important to the performance. Thus, the major design issue of the OpenMP-PPBF includes the proper use of synchronization in multicore systems.

The MPI library mainly supports a development environment for parallel programing in a cluster with distributed-memory organization. It can implement a message-passing mechanism among the nodes in a cluster. OpenMP API provides a set of compiler directives run in a multiprocessor (multicore) with shared-memory architecture. It can execute parallel programing on an SMP/CMP. Both of them can be implemented in a similar fashion that supports the single-program multiple-data (SPMD) programing model for MPI and the fork-join model for OpenMP with the same multiple-instruction, multiple-data (MIMD) architectures (Flynn's taxonomy).[23] The MPI library and OpenMP API are also compatible with the most widely used computer language. Because of the symmetrical and parallel properties of the superposition and stack properties of the PBF,[18] the proposed PPBF can be efficiently adapted to these different degrees of parallelism. Consequently, the proposed PPBF technique can be implemented by the hybrid-PPBF, which combines the cost-effective MPI clusters and the sophisticated OpenMP multicore architectures to reduce workloads.

With highly symmetrical and parallel properties, the proposed hybrid PPBF can be well divided into successively parallel levels of the MCE learning modules and the stack filters[16] in a variety of parallel partitions as shown in Fig. 2. These properties, the substances of parallelism contained in the PPBF can also be explored. As a consequence, an appropriate granularity of parallelism can therefore be conducted from a wide range scale of the PPBF parallelism especially accommodated to the cluster networks and the CMP/SMP architectures. This allows accessing and exploring the parallelism in different levels of parallel computing standards, such as the MPI executed on a cluster and the OpenMP run on an SMP/CMP. These advantages can be proven by the high parallel speedup ratios, as demonstrated in Sec. 3.

### 2.2 Stages 1 and A: Threshold Decomposition (NQ)

We assume there are $N$ different classes. The threshold decomposition $T_\ell(\cdot)$, also known as ($NQ$) function, are initially applied to the band vectors $b^k$ (feature sets) of training samples $\mathbf{X}_{\mathrm{TRN}} = (\mathbf{x}^k_{\mathrm{TRN}_1}, \ldots, x^k_{\mathrm{TRN}_l}, \ldots, x^k_{\mathrm{TRN}_n})$ for all classes $\omega_k$, $k \in \{1, \ldots, N\}$ and test samples $\mathbf{X}_{\mathrm{TST}} = (x_{\mathrm{TST}_1}, \ldots, x_{\mathrm{TST}_l}, \ldots, x_{\mathrm{TST}_n})$ to generate the new $NQ$ binary vectors $\mathbf{e}_{NQ}$. These samples $\mathbf{X}_{\mathrm{TRN/TST}_l}$ for all bands, where $l \in \{1, \ldots, n\}$, are first calculated simultaneously on each SN in parallel, as shown in stages 1 and A of Fig. 1. They are normalized to

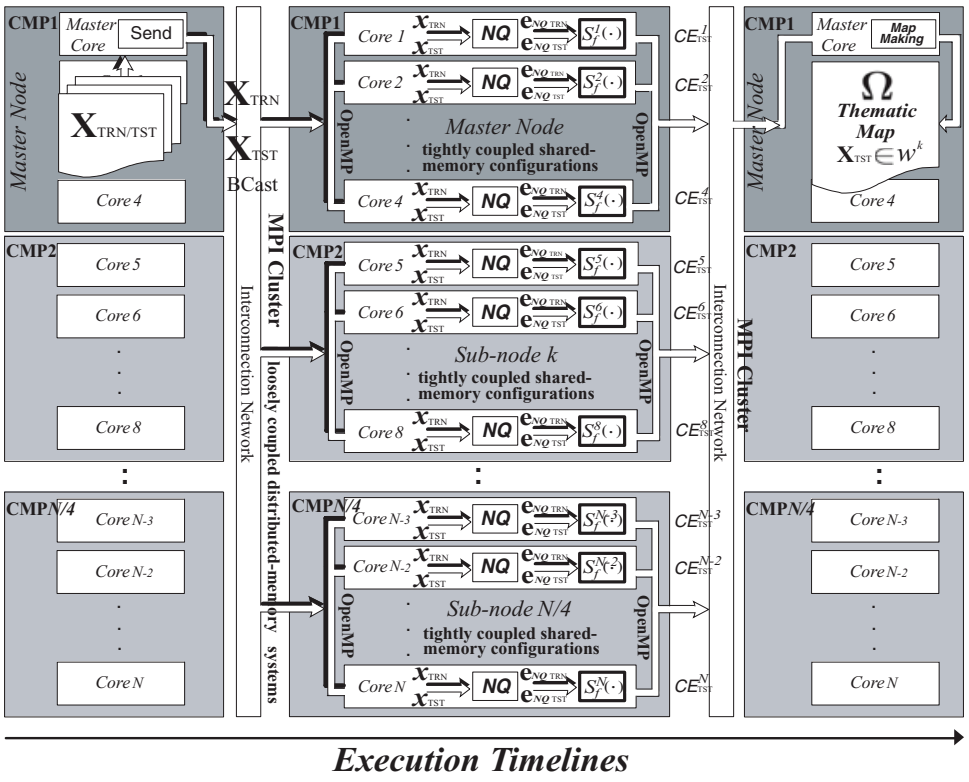


**Fig. 2** Hierarchical structure of hybrid PPBF.

the range $(0, 1)$ by the nonlinear sigmoid function.[24] These normalized vector $\mathbf{e}_{N_{\mathrm{TRN/TST}_l}}$ are next uniformly quantized into $L-1$ levels on each SN $\omega_k$ to produce binary vectors of the training samples $\mathbf{e}^k{}_{NQ_{\mathrm{TRN}_{\ell,l}}} = (\mathbf{e}^k_{NQ_{\ell,1}}, \ldots, \mathbf{e}^k_{NQ_{\ell,l}}, \ldots, \mathbf{e}^k_{NQ_{\ell,n}})$ in PPBF learning phase and the test samples $\mathbf{e}_{NQ_{\mathrm{TST}_{\ell,l}}} = (\mathbf{e}_{NQ_{\ell,1}}, \ldots, \mathbf{e}_{NQ_{\ell,l}}, \ldots, \mathbf{e}_{NQ_{\ell,n}})$ in PPBF classification phase, where $\ell \in \{L-1, \ldots, 1\}$. It uses the $NQ$ function $T_\ell(\cdot)$ to transform the $\mathbf{X}_{\mathrm{TRN/TST}}$ samples of real numbers into the binary vectors $\mathbf{e}^k{}_{NQ_{\mathrm{TRN}}}$ for all different classes $\omega_k$ and $\mathbf{e}_{NQ_{\mathrm{TST}}}$ on each parallel SN $\omega_k$.

As shown in Fig. 3, all band vectors $\mathbf{X}_{\mathrm{TRN/TST}}$ are normalized and quantized into $L$ levels of binary numbers stored at each threshold level $\ell$ of parallel stack filters $S^k_f(\cdot)$. A set of binary vectors $\mathbf{e}_{NQ_{\mathrm{TRN/TST}}}$ is then constructed in these levels of stack filters for the MCE learning and classification. These $NQ$ binary vectors $\mathbf{e}_{NQ_{\mathrm{TRN/TST}}}$ on each SN $\omega_k$ are crucial to subsequent PPBF classifications for both PPBF learning and classification phases, as shown in Fig. 1.

## 2.3 Stages 2 and B: Parallel Positive Boolean Function Learning and Classification

The proposed PPBF is designed to develop a multiclass classifier that can fully utilize MCE[17] learning ability as its optimization criterion.[16] With the discrete and nonlinear binary capabilities of the PPBF, general information contained in the $NQ$ binary vectors $\mathbf{e}_{NQ}$ produced by previous stages 1 and A, such as amplitudes, phase differences, degrees of polarization, etc., are well preserved and also completely explored.[25] One of the essential features in this setup lies in that the chosen PPBF must be able to handle the extremely different types of remote sensing data and yet accept diverse input feature vectors, while maintaining a sufficiently discrete and nonlinear learning ability in parallel. Fig. 1 shows the parallel realization of the PBF learning with MCE

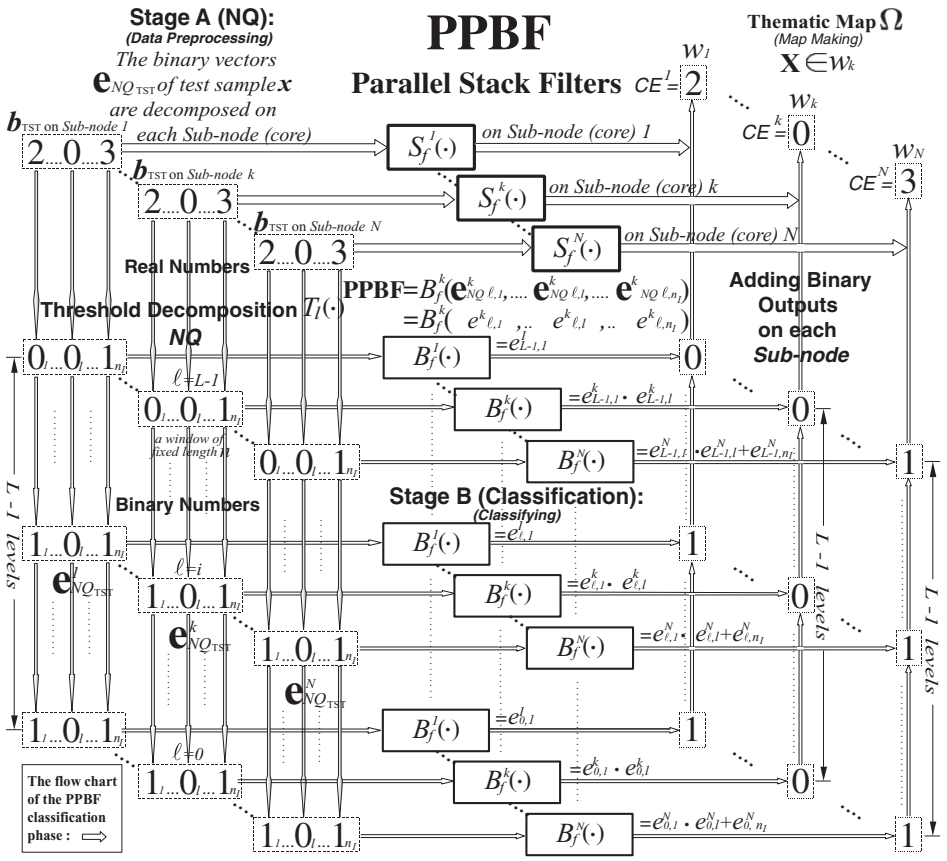


**Fig. 3** An example of a PPBF classification approach corresponding to a stack filter implemented in Stage B (Classification).

criterion for finding the optimal PBF $B_f(\cdot)$ in learning phase and the proposed PBF-based multiclassifier with stack filters for making the thematic map in classification phase.[16]

The PPBF broadcasts the $NQ$ binary vectors $\mathbf{e}^k{}_{NQ}$ to each parallel SN $\omega_k$, segments the process of learning stage (stage 2) into parallel MCE training modules on each parallel SN $\omega_k$ simultaneously, divides the procedures of classification stage (stage B) into parallel stack filters implemented concurrently on each parallel SN $\omega_k$, and then reassembles the summations of the binary outputs at different levels for each distinct class on the MN. As shown in Fig. 3, these collected summations of binary outputs are compared to each other and the smallest value is chosen as the final decision class of the thematic maps. Two stages regarding the MCE learning and classification of PPBF classifier are illustrated in details as follows.

### 2.3.1 Stage 2 (learning)

The proposed PPBF decomposes the computation of the MCE learning[16] inside a stack filter into successive levels of the MCE modules and explores the stackable features of MCE training at each threshold level $\ell$ in parallel. Each binary level of $NQ$ binary training vectors $\mathbf{e}^k{}_{NQ_{\mathrm{TRN}}}$ can be well adapted to the parallel characteristics of the MCE criterion, which has the ability to learn from both positive and negative samples to improve the classification accuracy. The proposed PPBF algorithm is therefore designed to distribute all training and test samples $\mathbf{X}_{\mathrm{TRN/TST}}$ of different classes to each parallel SN $\omega_k$ before the MCE learning parameters, and the classification maps are well determined individually. In our generalized positive Boolean function (GPBF) approach,[16] an optimal stack filter $S_f^k(\cdot)$ is defined as a filter whose MCE value between the filter's output and the desired signals is minimum. The multilevel MCE values can

be decomposed into the summation of classification errors, also called occurrences, occurring at each threshold level $\ell$.

As shown in Fig. 1, we assume there are $N$ classes of training samples. $M$ stands for a fixed number of training samples for each class $\omega_k$. $M \times N$ training samples are decomposed into binary vectors $\mathbf{e}^k{}_{NQ_{\mathrm{TRN}}}$, being the whole set of the binary training vectors $\mathbf{e}^k{}_{NQ_{i,j,\ell,l}}$ for all classes $\omega_k$ at each threshold level $\ell$ with a slice $l$ inside a window of fixed length $n$, where $k \in \{1, \ldots, N\}$, $i \in \{1, \ldots, N\}$, $j \in \{1, \ldots, M\}$, $\ell \in \{L-1, \ \ldots, 1\}$, and $l \in \{1, \ldots, n\}$, as shown in Fig. 3. Let $x_{i,j}$ represent $M \times N$ training samples and $\mathbf{e}^k{}_{NQ_{i,j}}$ stand for $M \times N$ binary training vectors, both with a window of fixed length $n$ at each level $\ell$. The desired value of an occurrence $\mathbf{e}^k_{NQ_{i,j,\ell,l}}$, also known as a binary training vector at each threshold level $\ell$ with a slice inside a window of fixed length $n$, can be treated as the error value of sample $x_{i,j}$ at class $\omega_k$. If sample $x_{i,j}$ belongs to class $\omega_k$, then it means that no error occurs (i.e., the desired value of the occurrence $\mathbf{e}^k{}_{NQ_{i,j,\ell,l}}$ is 0). Otherwise, it is equal to 1,

$$d(\mathbf{e}^k{}_{NQ_{i,j,\ell,l}}) = \begin{cases} 0 & \text{if } i = k, x_{i,j} \in \omega_k \\ 1 & \text{if } i \neq k, x_{i,j} \notin \omega_k. \end{cases} \tag{1}$$

Figure 3 shows the PBF criterion,[25] a Boolean function $B^k_f(\cdot)$ is defined as an occurrence $\mathbf{e}^k_{NQ_{i,j,\ell,l}}$ at level $\ell$ in a window of fixed length $n$ (the number of Boolean binary variables) for the class $\omega_k$. Coyle and Lin[26] formulated the estimation error as the sum of $2^n$ possible binary window vectors, which are observed from a window of length $n$. That is,

$$\min \sum_{j=0}^{2^n-1} [\alpha_j p(0|x_{i,j}) + \beta_j p(1|x_{i,j})], \tag{2}$$

where $\alpha_{i,j}$, $\beta_{i,j}$ are predetermined constants that represent the cost coefficients in an observation window of length $n$. The $p(0|x_{i,j})$ and the $p(1|x_{i,j})$ are the probabilities that the filter outputs a 0 from a negative occurrence and a 1 from a positive occurrence, respectively, when the MCE observes a binary vector $x_{i,j}$ in its window.

Each parallel MCE learning module is trained by using its own training samples of the class $\omega_k$ as positive occurrences and the other classes as negative occurrences. The output of decision class can be collected by summing the binary filtering occurrences at each threshold level $\ell$ of the parallel MCE learning modules on each parallel SN. Here, we apply this window of length $n$ to the number of feature bands for the fusion of hyperspectral and SAR images. As an example, Fig. 3 shows that the sliding windows of fixed length $n$ consist of binary vectors $\mathbf{e}^k{}_{NQ_{\mathrm{TRN}}}$ of all classes. They are used as the MCE training parameters for each MCE learning modules of the proposed PPBF.

### 2.3.2 Stage B (classification)

On the basis of the threshold decomposition property, a binary test (unlabeled) vector $\mathbf{e}_{NQ_{\mathrm{TST}}}$, also known as an input of PPBF in the classification phase, can be decomposed into the binary vectors occurrences $\mathbf{e}_{NQ_{\mathrm{TST}_\ell}}$ with a window of fixed length $n$ ($n$ feature bands) at each threshold level $\ell$ in a stack filter $S^k_f(\cdot)$. Considering two samples $u$, $v$ and a class $\omega_k$, we assume that $\mathbf{e}_{NQ_{\mathrm{TST}_u}} \leq \mathbf{e}_{NQ_{\mathrm{TST}_v}}$ for all dimensional elements as indicated above. If sample $u$ does not belong to class $\omega_k$ (an error occurs), then sample $v$ does not belong to class $\omega_k$. On the other hand, if sample $v$ is an element of class $\omega_k$ (no error), the sample $u$ should be an element of class $\omega_k$. Here, we define $E_f(\cdot)$ as an error function. Thus, $E_f(u) \leq E_f(v)$, if $\mathbf{e}_{NQ_{\mathrm{TST}_u}} \leq \mathbf{e}_{NQ_{\mathrm{TST}_v}}$ for all dimensional elements. We further define $E_f(x)$ to be PPBF $B^k_f(\cdot)$ of an occurrence $\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}}$ of the class $\omega_k$ at each threshold level $\ell$ with a slice $l$ inside a window of fixed length $n$. If $\mathbf{e}_{NQ_{\mathrm{TST}_{u,\ell}}} \leq \mathbf{e}_{NQ_{\mathrm{TST}_{v,\ell}}}$, then $B^k_f(\mathbf{e}_{NQ_{\mathrm{TST}_{u,\ell}}}) \leq B^k_f(\mathbf{e}_{NQ_{\mathrm{TST}_{v,\ell}}})$, which satisfies the stacking property and indicates that the stacking property can effectively solve the classification problems in each parallel stack filter $S^k_f(\cdot)$, as shown in Fig. 3.

According to PBF criteria,[25] a Boolean function $B_f^k(x)$ can be defined as an occurrence $\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}}$ at each threshold level $\ell$ on each parallel SN $\omega_k$ for the class $\omega_k$, where $k \in \{1, \ldots, N\}$ and $\ell \in \{L-1 \ldots, 1\}$. The classification error ($\mathrm{CE}^k$) on the parallel SN $\omega_k$ of the class $\omega_k$, which is defined as the expected value ($E$) of the differences between the desired values $d(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}})$ and the stack filter's binary outputs $S_f(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}})$ as implemented on each parallel SN $\omega_k$ for class $\omega_k$, is determined by

$$
\begin{aligned}
\mathrm{CE}_x^k &= E\left[\left|d(b_{\mathrm{TST}_x}) - S_f^k(b_{\mathrm{TST}_x})\right|\right] \\
&= E\left\{\left|\sum_{\ell=1}^{L-1} d[T_l(b_{\mathrm{TST}_x})] - B_f^k[T_l(b_{\mathrm{TST}_x})]\right|\right\} \\
&\quad \text{(threshold decomposition NQ on each parallel SN } \omega_k\text{)} \\
&= E\left[\sum_{\ell=1}^{L-1}\left|d(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}}) - B_f^k(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}})\right|\right] \\
&\quad \text{(stack property on each parallel SN } \omega_k\text{)} \\
&= \sum_{\ell=1}^{L-1} E\left[\left|d(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}}) - B_f^k(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}})\right|\right] \\
&= \sum_{\ell=1}^{L-1}\left|d(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell}}}) - B_f^k(\mathbf{e}_{NQ_{\mathrm{TST}_{x,\ell,l}}})\right| \qquad (3) \\
&\quad (l \in \{1, \ldots n\}),
\end{aligned}
$$

where a stack filter $S_f^k(\cdot)$, a threshold function $T_\ell(\cdot)$, and a Boolean function $B_f^k(\cdot)$ are used at each threshold level $\ell$ of the SN $\omega_k$. The $b_{\mathrm{TST}_x}$ are band vectors of arbitrary test samples $x$.

The classification error $\mathrm{CE}_x^k$ outputs [i.e., the summations of binary outputs produced from each Boolean function $B_f^k(\cdot)$ of a parallel stack filter $S_f^k(\cdot)$ at each level $\ell$] are compared to each other among these parallel SNs $\omega_k$, where $k \in \{1, \ldots, N\}$. The smallest number of $\mathrm{CE}_x^k$ is finally chosen as an exact class $\omega_k$ for the test sample $x$ to make a thematic map. An example of the classification procedure for an actual test sample $x$ is shown in Fig. 3. In this phase, the $NQ$ binary vectors $\mathbf{e}_{NQ_{\mathrm{TST}}}$ are applied to a set of $B_f^k(\cdot)$ corresponding to the successive levels of MCE learning modules to obtain the $CE^k s$ of class $\omega_k$ on each parallel SN $\omega_k$ in the hybrid-PPBF hierarchical structure. This shows that the proposed PPBF method can be used as a basis for a parallel multiclassifier.

On the whole, the entire PPBF learning and classification phases can be divided into three components: ($i$) A table possessing $2^{n+1}$ entries (obtained by multiplying the possible binary window vectors $2^n$ by two probabilities of the filter's outputs, 0 and 1, as shown in Eq. (2), called a probability density table (PDT), is constructed to represent the possible threshold occurrences of the desired values and the filter's outputs at each threshold level $\ell$. ($ii$) An optimal stack filter $S_f^k(\cdot)$ with the minimal MCE value is calculated by the graphic search-based algorithm[27] based on this PDT. Both components 1 and 2 are implemented on the MN and SNs. ($iii$) Finally, the classification can be obtained by summing the binary filtering results on the MN, which will be reported in detail in Sec. 3. With this criterion, this operation can be well parallelized according to different class $\omega_k$ in each parallel node.

Because the MPI is beneficial when problem granularity is greater whereas the OpenMP takes advantages when task granularity is fine enough, hence, the major design issues over the MPI-PPBF and OpenMP-PPBF are, respectively, how to minimize the communication delays through the MPI cluster with loosely coupled distributed-memory architectures and how to optimize the cache performance via the OpenMP API with tightly coupled shared-memory

configurations. Fortunately, due to the highly symmetrical and parallel properties of PPBF, these major considerations can be taken into account. The PBF not only can be well divided into successively parallel stack filters but can also be further explored by the substantial parallelism of PPBF in a variety of parallel partitions. In order to demonstrate that the proposed PPBF can be efficiently adapted to different degrees of parallelism, a promising hybrid PPBF, which combines both MPI-PPBF and OpenMP-PPBF, is therefore proposed. In contrast to the stand-alone MPI-PPBF and OpenMP-PPBF, the proposed hybrid PPBF can be employed on a hierarchical architecture of CMP/SMP clusters as shown in Fig. 2. It can combine both MPI and OpenMP mixed features operated in a hybrid mode to efficiently exploit different levels of parallelization.

According to the PCAM methodology,[22] two main mapping strategies for optimal load balancing are proposed for hybrid-PPBF scheme. (*i*) Task mappings divided by distinct class $\omega_k$ for each processor provide the PPBF the best way to share data between parallel tasks for PPBF. Using proposed task mappings, the hierarchical structure of CMP/SMP clusters can well adopt a mixing-mode–specific parallelization strategy and can exploit distributed-memory parallelism across the nodes of a MPI cluster, and shared-memory parallelism within each core of CMP/SMP nodes. (*ii*) Data mappings distributed by all training and test samples $\mathbf{X}_{\mathrm{TRN/TST}}$ to each SN give the PPBF a great opportunity to optimize the load balancing for the hybrid PPBF.

## 3 Experimental Results

This paper focuses on the speedup aspect of PPBF as compared to the original PBF. The analysis of classification performance of PBF classifier can be found in our previous works.[15,16]

### *3.1 Experimental Setup*

A plantation area in Au-Ku on the west coast of Taiwan is chosen in this study. The image data were obtained by the MODIS/ASTER airborne simulator (MASTER) instrument, a hyperspectral sensor, and Airborne Synthetic Aperture Radar (AIRSAR) instrument as part of the Pacrim II project executed in 2000. The proposed PPBF method is applied to 35 bands selected from the 50 contiguous bands (excluding the low signal-to-noise ratio midinfrared channels) of MASTER and nine components of AIRSAR.[28] Nine components in the polarimetric SAR covariance matrix are preprocessed.[29] The MASTER and AIRSAR images had different spatial resolutions and different numbers of bytes of pixel values. The SAR data are registered to the same resolution of MASTER data using an averaging algorithm. Both of them are also linearly rescaled to the same number of bytes of pixel values.

The proposed hybrid-PPBF was developed to run on a homogeneous cluster (HC), which consists of eight machines with the Intel Core 2 Quad 2.4-GHz multiprocessor, 4-MB L2 cache and 4 GB of memory for each node, and was compared to its counterpart of the original PBF, which is a serial implementation written in C to run on the same stand-alone machines. The C code used for the serial PBF approach is essentially identical to the code used in C for the hybrid-PPBF approach. The HC was connected with a high-speed gigabit switch. It provides MPI standard libraries and OpenMP directives. A short description of the experimental setup is depicted in Table 1.

**Table 1** Experimental setup.

| | CMP/SMP multcore | Cluster network |
|---|---|---|
| Platform | Intel core 2 quad-cores | High-speed gigabit switch |
| Clock rate/speed | 2.4-GHz | Gigabit/second |
| Memory type | Tightly coupled shared-memory | Loosely coupled distributed-memory |
| Maximum capacity | 1 node (4 cores) | 8 nodes (32 cores) |
| API/library | OpenMP | MPI |
| Programming type | Multi-threaded | Message passing |

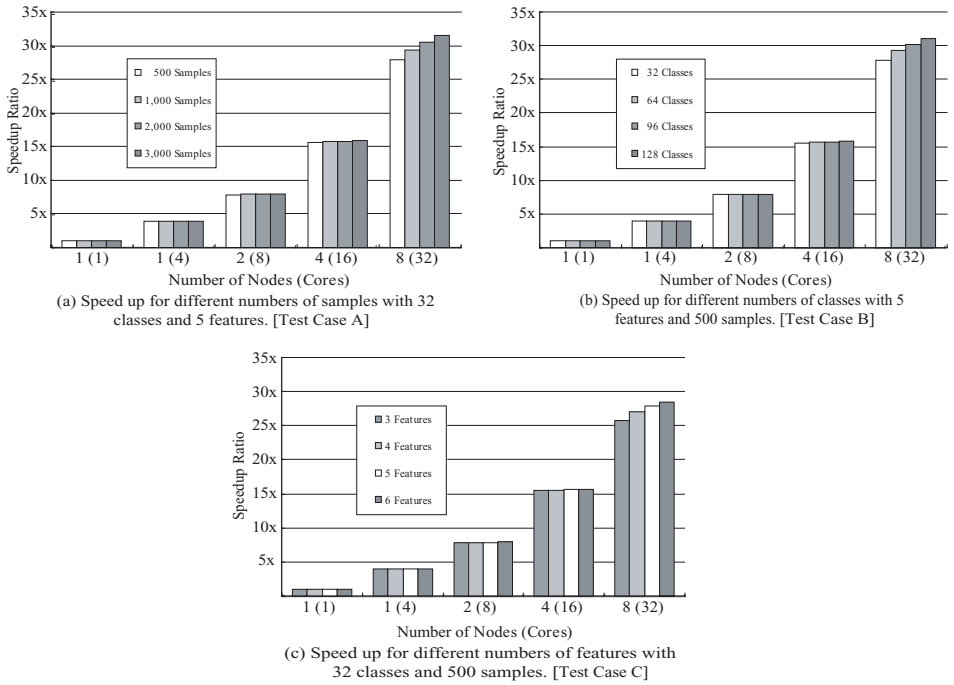


**Fig. 4** Hybrid-PPBF performance comparisons among five test configurations, namely 1 (1), 1 (4), 2 (8), 4 (16), and 8 (32) nodes (cores), in terms of three multicriteria analyses, which includes the three test cases of different numbers of (a) samples, (b) classes, and (c) features.

In order to illustrate the performance of proposed hybrid-PPBF method in a variety of parallel partitions, a variety of combinations of the simulated data sets had been uniformly chosen from the original MASTER/AIRSAR data sets. The simulated data sets were composed of different amounts of samples $\mathbf{X}_{\mathrm{TRN/TST}}$, classes $\omega_k$ and features (bands) $n$ (the number of bands in a window of fixed length $n$ in the experiments). Five different test configurations, known as 1 node (1 core) only (a single processor of a CMP node), one node with four cores (four multicores of a CMP node), two nodes with eight cores (eight multicores in two CMP nodes), four nodes with 16 cores (sixteen multicores in four CMP nodes), and 8 nodes with 32 cores (32 multicores in eight CMP nodes), were tested by varied combinations of different numbers of samples $\mathbf{X}_{\mathrm{TRN/TST}}$ (500, 1000, 2000, and 3000), classes $\omega_k$ (32, 64, 96, and 128) and features (bands) $n$ (3, 4, 5, and 6), which were from three test cases of the simulated data sets as presented in Fig. 4(a)–4(c), respectively.

The speedup of execution time measured by the first configuration (one node/one core operated in hybrid-PPBF scheme) was assumed to have a speedup ratio of 1 in the experiment. Note that this serial version of code is optimized by using optimization flags in the compilation. The computational speedups associated with different numbers of samples $\mathbf{X}_{\mathrm{TRN/TST}}$, features $n$ and classes $\omega_k$ were therefore compared to their first configuration as shown in Fig. 4. In order to evaluate the effectiveness of the proposed method, a speedup $S_p$ in Eq. (4) is used to evaluate the hybrid-PPBF performance running in the HC. $S_p$ is the ratio between sequential execution time and parallel execution time,

$$S_p = \frac{t_s}{t_p}, \tag{4}$$

where $t_s$ is the execution time of the sequential algorithm and $t_p$ is the execution time of the parallel algorithm with $p$ computing nodes (cores). The criteria for the performance measures are based on the same test configuration (i.e., 32 classes, 5 features, and 500 samples), as shown in Fig. 4, with different benchmarks (i.e., test cases A, B, and C) as described above.

### 3.2 Results Comparison

With identical experimental settings, the experimental results have shown that the PPBF achieves the same classification accuracies as described in Refs. 15 and 16. Comparison between these results confirms that the hybrid PPBF is thus more suitable for solving the multiclassification problems in terms of execution performance when it is compared to the original PBF classifier. The experimental results reported in Refs. 15 and 16 also validated the evaluation of classification accuracies and the unique properties of the proposed PPBF method. These encouraging results have shown that satisfactory classification accuracy could be achieved with only a few training samples. Especially for remote sensing images, it is generally difficult to provide sufficient labeled samples obtained from expensive ground surveys.[16]

To demonstrate the advantages of PPBF performances, the benchmark design is based on a variety of combinations, which include different numbers of classes $\omega_k$, features $n$, and samples $\mathbf{X}_{\mathrm{TRN/TST}}$ as illustrated in the aforementioned hybrid-PPBF experimental setup. These test cases are introduced to evaluate the flexibility and parallelism of PPBF scheme. Three multicriteria analyses of performance comparison, (which includes test case A, with different numbers of samples as the number of classes is set as 32 and the number of features is set as 5; test case B, with different numbers of classes as 4 features and 500 samples are applied to test; and test case C, with different numbers of features as 32 classes and 500 samples are fixed) are made for hybrid-PPBF measured in HC.

On the basis of the experimental results as shown in Fig. 4, the speedup performance of three test cases are almost proportional to their corresponding number of nodes (cores), in general. Note that the white bars depicted in the performance charts use the same criterion in which the numbers of classes $\omega_k$, features $n$ and samples $\mathbf{X}_{\mathrm{TRN/TST}}$ were set to 32, 5, and 500, respectively, to provide an interactive comparison between three different test cases in the analysis. The performance corresponding to test cases A and B showed a better speedup than that of test case C, on average. This indicates that more features applied to the hybrid PPBF will degrade the speedup performance. Theoretically, it is because that the computational load of PBF classifiers increases exponentially while the feature amount of each class is growing. As a result, the proposed hybrid PPBF is very suitable for the available size increments of both sample and class amount applied to the classification processes. As also shown in Fig. 4, case B performed better than case C but slightly worse than case A. It can conclude that the greater the number of samples that are applied to the hybrid PPBF, the better the performance that can be obtained from the benefits of parallelism. As we can see in Table 2, all of the speedups measured in test case A are almost close to the number of cores for the applications of large-size samples. As a result, these obtained results have demonstrated the advantages of the proposed hybrid-PPBF technique.

**Table 2** Summary of execution time and speedup for test case A with 32 classes and five features.

| | No. nodes (Cores) | | | | |
|---|---|---|---|---|---|
| **No. samples** | **1 (1)** | **1 (4)** | **2 (8)** | **4 (16)** | **8 (32)** |
| | | | Execution time (s) | | |
| 500 | 52.47 | 13.33 | 6.67 | 3.37 | 1.88 |
| 1000 | 210.20 | 53.25 | 26.52 | 13.34 | 7.15 |
| 2000 | 841.04 | 211.67 | 105.81 | 53.25 | 27.50 |
| 3000 | 1899.99 | 477.98 | 238.67 | 119.89 | 60.34 |
| | | | Speedup ratio | | |
| 500 | 1.00× | 3.94× | 7.86× | 15.58× | 27.86× |
| 1000 | 1.00× | 3.95× | 7.93× | 15.76× | 29.42× |
| 2000 | 1.00× | 3.97× | 7.95× | 15.80× | 30.58× |
| 3000 | 1.00× | 3.98× | 7.96× | 15.85× | 31.49× |

## 4 Conclusion

This paper presents a novel PPBF supervised classification approach. Compared to a stand-alone PBF classifier, PPBF can reduce the computational burden resulting from large volumes of remote sensing images. Because of the highly symmetrical superposition and parallel stack properties of the PBF, the proposed PPBF can fully explore the significant parallelism embedded in PBF and decompose PBF into successively smaller PBF stack filters. Consequently, the proposed PPBF technique can be implemented in a hybrid mode to reduce workloads. They can further be efficiently adapted to different degrees of parallelism. The new proposed hybrid PPBF is a promising hierarchical structure that can well combine MPI-PPBF and OpenMP-PPBF, and properly utilize task and data mapping strategies to optimize load balancing. In the experiment, a homogeneous cluster was employed to measure the performance of the hybrid PPBF. Experimental results prove that the hybrid PPBF can well adapt to a variety of parallel partitions and significantly reduces the computational loads compared to a stand-alone PBF classifier. The proposed PPBF parallel structure allows us to explore the possibility of real-time operation, which will be the subject of our further studies.

## 5 Acknowledgments

This work was supported by National Science Council, Taiwan, under Grant No. NSC 99-2116-M-027-003. The authors thank the Center for Space and Remote Sensing Research of National Central University for providing MASTER/AIRSAR data sets and the EPS$^2$ Laboratory of National Taipei University of Technology for providing the clusters used in the experiments.

## References

1. A. H. S. Solberg, A. K. Jain, and T. Taxt, "Multisource classification of remotely sensed data: Fusion of landsat TM and SAR images," *IEEE Trans. Geosci. Remote Sens.* **32**(4), 768–778 (1994).
2. J. A. Richards, "Thematic mapping from sensor formations," in *Proc. IEEE Int. Geosci. Remote Sensing Symp.*, pp. 1505–1508 (2007).
3. A. Plaza, Q. Du, Y.-L. Chang, and R. L. King, "High performance computing for hyperspectral remote sensing," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **4**(3), 528–544 (2011).
4. J. Setoain, M. Prieto, C. Tenllado, A. Plaza, and F. Tirado, "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geosci. Remote Sens. Lett.* **4**(3), 441–445 (2007).
5. O. Nogues-Correig, E. C. Gali, J. S. Campderros, and A. Rius, "A gps-reflections receiver that computes doppler/delay maps in real time," *IEEE Trans. Geosci. Remote Sens.* **45**(1), 156–174 (2007).
6. E. Chu, D. Tremblay, K. Croft, and A. Griffin, "TES science investigator-led processing system," *IEEE Trans. Geosci. Remote Sens.* **44**(5), 1352–1358 (2006).
7. M. A. Fischman, A. Berkun, W. Chun, E. Im, and R. Andraka, "An onboard processor and adaptive scanning controller for the second-generation precipitation radar," *IEEE Trans. Geosci. Remote Sens.* **43**(4), 802–812 (2005).
8. C.-C. Chang, Y.-L. Chang, M.-Y. Huang, and B. Huang, "Accelerating regular LDPC code decoders on GPUs," *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **4**(3), 653–659 (2011).
9. A. Plaza, D. Valencia, J. Plaza, and C.-I. Chang, "Parallel implementation of endmember extraction algorithms from hyperspectral data," *IEEE Geosci. Remote Sens. Lett.* **3**(3), 334–338 (2006).
10. Y.-L. Chang, J.-P. Fang, J.-P. Huang, C.-C. Lin, H. Ren, and W.-Y. Liang, "A parallel computing technique for complete modular eigenspace feature extraction of hyperspectral images," in *Proc. IEEE Int. Geosci. Remote Sensing Symp.*, pp. 960–963 (2006).

11. H. M. Chi and O. K. Ersoy, "A statistical self-organizing learning system for remote sensing classification," *IEEE Trans. Geosci. Remote Sens.* **43**(8), 1890–1900 (2005).
12. X. Song, G. Fan, and M. Rao, "Automatic crp mapping using nonparametric machine learning approaches," *IEEE Trans. Geosci. Remote Sens.* **43**(4), 888–897 (2005).
13. J. T. Johnson, "A study of ocean-like surface thermal emission and reflection using voronovich's small slope approximation," *IEEE Trans. Geosci. Remote Sens.* **43**(2), 306–314 (2005).
14. Y. L. Chang, C. C. Han, K. C. Fan, K. S. Chen, C. T. Chen, and J. H. Chang, "Greedy modular eigenspaces and positive boolean function for supervised hyperspectral image classification," *Opt. Eng.* **42**, 2576–2587 (2003).
15. Y. L. Chang, C. C. Han, H. Ren, C.-T. Chen, K. S. Chen, and K. C. Fan, "Data fusion of hyperspectral and sar images," *Opt. Eng.* **43**, 1787–1797 (2004).
16. Y. L. Chang, L. S. Liang, C. C. Han, J. P. Fang, W. Y. Liang, and K. S. Chen, "Multisource data fusion for landslide classification using generalized positive Boolean functions," *IEEE Trans. Geosci. Remote Sens.* **45**(6, Pt 1), 1697–1708 (2007).
17. B. H. Juang, W. Chou, and C. H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. Speech Audio Process.* **5**(3), 257–265 (1997).
18. P. D. Wendt, E. J. Coyle, and N. J. Gallagher, "Stack filter," *IEEE Trans. Acoust. Speech Signal Process.* **34**(4), 898–911 (1986).
19. Message Passing Interface Forum, "MPI: a message-passing interface standard," *Int. J. Supercomput. Applications High Perform. Comput.* **8**(3/4), 159–416 (1994).
20. L. Dagum and R. Menon, "OpenMP: an industry-standard API for sharedmemory programming," *IEEE Comput. Sci. Eng.* **5**(1), 46–55 (1998).
21. Y.-L. Chang, J.-P. Fang, L.-S. Liang, L.-D. Chen, and K.-S. Chen, "A parallel positive Boolean function approach to supervised multispectral image classification," in *Proc. of IEEE Int. Geosci. Remote Sensing Symp.*, pp. 1525–1528 (2007).
22. I. Foster, *Designing and Building Parallel Programs*, Addison-Wesley, Reading, MA (1995).
23. M. J. Flynn, "Some computer organizations and their effectiveness," *IEEE Trans. Comput.* **21**(9), 948–960 (1972).
24. C.-I. Chang, *Recent Advances in Hyperspectral Signal and Image Processing*, Transworld Research Network, Trivandrum, Kerala (2006).
25. C. C. Han, "A supervised classification scheme using positive Boolean function," in *Proc. IEEE 16th Int. Conf. on Pattern Recognition* **2**, 100–103 (2002).
26. E. J. Coyle and J. H. Lin, "Stack filters and the mean absolute error criterion," *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-36**, 1244–1254 (1988).
27. C. C. Han, K. C. Fan, and Z. M. Chen, "Finding of optimal stack filter by graphic searching methods," *IEEE Trans. Signal Process.* **45**(7), 1857–1862 (1997).
28. S. J. Hook, J. J. Myers, K. J. Thome, M. Fitzgerald, and A. B. Kahle, "The MODIS/ASTER airborne simulator (MASTER)–a new instrument for earth science studies," *Remote Sens. Environ.* **76**, 93–102 (2000).
29. J. S. Lee, M. R. Grunes, T. L. Ainsworth, L. J. Du, D. L. Schuler, and S. R. Cloude, "Unsupervised classification using polarimetric decomposition and the complex wishart classifier," *IEEE Trans. Geosci. Remote Sens.* **37**, 2249–2258 (1999).


**Yang-Lang Chang** received his MS degree in computer engineering from Syracuse University in 1993 and his PhD degree in computer science and information engineering from the National Central University, Taiwan, in 2003. He is a senior member of IEEE and a member of SPIE and Phi Tau Phi Scholastic Honor Society. Dr. Chang has been a session chair, workshop chair and conference program committee member for several international conferences. He is a member of editorial advisory board of Open Remote Sensing Journal. He serves as a guest editor for the special

issue on High Performance Computing in Earth Observation and Remote Sensing in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS). Currently, he also serves as a general secretary of the Taipei Chapter of the IEEE Geoscience and Remote Sensing Society. He is the recipient of the best reviewer award of the IEEE JSTARS in 2010. His research interests are in remote sensing, high performance computing and hyperspectral image analysis.

**Tung-Ju Hsieh** is an assistant professor in the Department of Computer Science and Information Engineering at the National Taipei University of Technology. Prior to his current role he was a postdoc at the California Institute for Telecommunications and Information Technology (Calit2). He received his PhD in electrical and computer engineering from the University of California-Irvine in 2006. His research areas include high-performance computing, parallel visualization, scientific visualization, and computer graphics. His research is aimed at using real-time visualization technology to explore massive three-dimensional scientific simulation and sensing data.

**Antonio Plaza** received his MS and PhD degrees in computer engineering from the University of Extremadura, Caceres, Spain. Since 2000, he has been an associate professor with the Department of Technology of Computers and Communications, University of Extremadura, Caceres, Spain, where he is the head of the Hyperspectral Computing Laboratory (HYPERCOMP). He is the coordinator of the Hyperspectral Imaging Network (Hyper-I-Net), a European project designed to build an interdisciplinary research community focused on hyperspectral imaging activities. He has been a Proposal Reviewer with the European Commission, the European Space Agency, and the Spanish Government. He is the author or coauthor of more than 250 publications on remotely sensed hyperspectral imaging, including more than 40 Journal Citation Report papers, book chapters, and conference proceeding papers. His research interests include remotely sensed hyperspectral imaging, pattern recognition, signal and image processing, and efficient implementation of large-scale scientific problems on parallel and distributed computer architectures.

**Yen-Lin Chen** received his BS and PhD degrees in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000 and 2006, respectively. From 2007 to 2009, he was an assistant professor at the Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan. Since Aug. 2009, he has been an assistant professor at the Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan. He is a member of the IEEE, IAPR, and IEICE. In 2003, he received Dragon Golden Paper Award sponsored by the Acer Foundation and the Silver Award of Technology Innovation Competition sponsored by the AdvanTech. His research interests include image and video processing, embedded systems, pattern recognition, intelligent vehicles, and intelligent transportation system.

**Wen-Yew Liang** received his BS degree in computer science and engineering from Tatung Institute of Technology, Taiwan, in 1992, his MS degree in computer science from National Tsing Hua University, Taiwan, in 1994, and his PhD degree in computer science and information engineering from National Taiwan University in 1998. Currently, he is an assistant professor in the Department of Computer Science and Information Engineering, National Taipei University of Technology. He is a member of ACM, IEEE Computer Society, and IICM. His research interests include embedded systems, low power system design, parallel and distributed systems, mobile computing, and parallelization for geosciences and other scientific applications. In addition, he devotes himself to the development, promotion, and professional education of Android and Linux systems.

**Jyh-Perng Fang** received his B.S. degree in Electrical Engineering from Chung Yuan Christian College in 1977 and his MS and PhD degree in Electrical Engineering from National Taiwan University in 1986 and 2004 respectively. Currently, he is an associate professor in the Department of Electrical Engineering, National Taipei University of Technology. His research interests include VLSI physical design automation and assistive technology for blind people.

**Bormin Huang** received his MSE in aerospace engineering from the University of Michigan, Ann Arbor, and his PhD in the area of satellite remote sensing from the University of Wisconsin-Madison. He is currently a research scientist and principal investigator at the Space Science and Engineering Center, the University of Wisconsin-Madison, where he advises and supports both national and international MS and PhD students and visiting scholars. He has been chairing the SPIE Conference on Satellite Data Compression, Communications and Processing, the SPIE Europe Conference on High-Performance Computing in Remote Sensing, and the 2011 IEEE International Workshop on Parallel and Distributed Computing in Remote Sensing. He is an associate editor of the Journal of Applied Remote Sensing (JARS), the guest editor of JARS special section on Satellite Data Compression and the guest editor of JARS special section on High-Performance Computing in Applied Remote Sensing, and a guest editor of special issue on Advances in Compression of Optical Image Data from Space for the Journal of Electrical and Computer Engineering.