International Journal of High Speed Electronics and Systems
Vol. 18, No. 2 (2008) 469–482

World Scientific
www.worldscientific.com

A PYRAMID-BASED BLOCK OF SKEWERS FOR PIXEL PURITY INDEX FOR ENDMEMBER EXTRACTION IN HYPERSPECTRAL IMAGERY

CHEIN-I CHANG, MINGKAI HSUEH, WEIMIN LIU, CHAO-CHENG WU,
FARZEEN CHAUDHRY, GREGORY SOLYAR

Remote Sensing Signal and Image Processing Laboratory
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County, Baltimore, MD 21250
cchang@umbc.edu

ANTONIO PLAZA

Computer Science Department, University of Extremadura
Avda. de la Universidad s/n,10.071 Caceres, SPAIN
aplaza@unex.es

Pixel Purity Index (PPI) has been widely used for endmember extraction. Recently, an approach using blocks of skewers was proposed by Theiler et al., called blocks of skewers (BOS) method, to improve computation of the PPI. It utilizes a block of skewers to reduce number of calculations of dot products operated by the PPI on each skewers with all data sample vectors. Unfortunately, the BOS method also suffers from the same drawbacks that the PPI does in terms of several parameters which are needed to be determined *a priori*. Besides, it also has an additional parameter, block size, B needed to be determined where no guideline is provided of how to select this parameter. In this paper, the BOS method is also investigated. Most importantly, a new pyramid-based block design for the BOS method is also introduced as opposed to the cube-based block designed used by Theiler et al.'s BOS. One major advantage of our proposed pyramid-based BOS over Theiler et al.'s cube-design BOS is the hardware design for Field Programmable Gate Arrays (FPGAs) implementation.

Keywords: Blocks of skewers (BOS) method; Field Programmable Gate Arrays (FPGAs); Pixel purity index (PPI).

1. Introduction

Endmember extraction is one of fundamental tasks in hyperspectral data exploitation. According to the definition given in [1], an endmember is an idealized, pure signature for a class. The importance of endmember extraction can be found in many applications in image classification, particularly, spectral unmixing where the signature matrix used for unmixing is made up of endmembers that are presumably resident in the image data. An endmember extraction algorithm (EEA) finds and locates endmembers present in image data. Over the past years, many EEAs have been developed and reported in literature such as pixel purity index (PPI) [2], N-finder (N-FINDR) algorithm [3], iterative error analysis (IEA) [4], automated morphological endmember extraction (AMEE) algorithm [5], minimum volume transform [6], convex geometry [7], convex cone analysis [8], etc.

and a comparative study among different EEAs was also conducted recently in [9]. Among these EEAs are PPI and N-FINDR algorithms which have been widely used in remote sensing community. In particular, the PPI is one of most popular EEAs due to its availability in the Environment for Visualization Imaging (ENVI) software. Its idea can be briefly described as follows. First of all, the PPI generates a large number of so-called "skewers" which are random vectors and then computes the dot product of each skewer with each pixel, where each of dot products can be carried out independently. Finally, a small number of pixels are selected manually as "pure" signatures by using an ENVI-visualization tool. Since the PPI requires a huge number of random skewers to produce reasonable and acceptable results, the computational cost is extremely high. To address this issue, a PPI-based approach using block of skewers (BOS), referred to as BOS-PPI was proposed by Theiler et al. [10]. By virtue of the BOS, the number of independent skewers which are used in the PPI can be drastically reduced, while the linearly dependent skewers generated by linear combinations of block of skewers (BOS) can be also used as skewers to find endmembers. In this case, the skewers used as a base for BOS can be considered as independent skewers. Since the derived dot products only involve scalar operations, they can be computed directly from the results of the original independent skewers used in the BOS. However, such a BOS-PPI method also inherits same drawbacks as the PPI does. Most importantly, there is an additional parameter, the block size B that is used to form a block of skewers and must be determined a *priori* where no guideline is provided for how to do it. In order to address this issue a new pyramid-based block of skewer PPI algorithm was proposed in [11] and was compared to the cube-based block designed and originally used by Theiler et al. [10]. In this paper, the FPGA implementations for the PPI and two versions of BOS-PPI, cube-based and pyramid-based will be also investigated. In particular, a new reconfigurable system for FPGA to implement our proposed pyramid-based BOS-PPI is further developed.

The remainder of this paper is organized as follows. Section 2 reviews the PPI [2] and delineates the PPI algorithm to be used in this paper. Section 3 describes the BOS method proposed in [10] and further develops a new pyramid-based BOSPPI and cube-based BOSPPI algorithms. Section 4 presents experiments based on computer simulations and real HYDICE images. Section 5 demonstrates the FPGA design architecture of our BOSPPI algorithm. Finally, some conclusions are drawn in Section 6.

2. Pixel Purity Index (PPI)

The PPI has been widely used for endmember extraction in remote sensing image processing applications. Although the PPI can be implemented through the ENVI software, its detailed implementation has not been reported. In this section, we make an effort to describe steps that we believe the PPI is carried out by the software as follows.

PPI Algorithm

1. Apply the MNF transform to reduce sensitivity of noise and dimensionality.

2. Let $\{\mathbf{skewer}_j\}_{j=1}^{k}$ be a large set of k randomly generated L-dimensional vectors, called "skewers" where k is an arbitrary, but predetermined positive integer. Also assume that t is a preset threshold positive integral value.
3. For each $\mathbf{skewer}_j$, all the data sample pixel vectors are projected onto $\mathbf{skewer}_j$ to find sample pixel vectors at its extreme positions to form an extreme set for the skewer $\mathbf{skewer}_j$, denoted by $S_{extrema}(\mathbf{skewer}_j)$. Despite the fact that a different $\mathbf{skewer}_j$ generates a different extreme set $S_{extrema}(\mathbf{skewer}_j)$, it is very likely that some sample pixel vectors may appear in more than one extreme set. Define an indicator function of a set S, $I_S(\mathrm{r})$ by

$$I_S(\mathbf{r}) = \begin{cases} 1; \text{ if } \mathbf{r} \in S \\ 0; \text{ if } \mathbf{r} \notin S \end{cases} \text{ and } N_{PPI}(\mathbf{r}) = \sum_j I_{S_{extrema}(\mathbf{skewer}_j)}(\mathbf{r}). \quad (1)$$

   where $N_{PPI}(\mathbf{r})$ is defined as the PPI counts of the sample vector $\mathbf{r}$.
4. Find all the sample pixel vectors with $N_{PPI}(\mathbf{r}) \geq t$ defined by Eq. (1), denoted by denoted by $\{\mathbf{r}_i\}_{i=1}^{p}$ where t is a threshold of determining how many candidate pixels will be selected for visualization. Load this set of $\{\mathbf{r}_i\}_{i=1}^{p}$ pixels in an L-dimensional visualization tool to select pixel vectors that correspond to pure pixel vectors.

3. Blocks-of-Skewers (BOS) Based PPI Method

In the block-of-skewers (BOS) method, the dot product of a block of B skewers is calculated with a single data sample pixel vector. If a new skewer was the linear combination of original block B of skewers, the dot product of this new skewer with the same data point would also be linear combination of the dot product of the block of skewers.

3.1. *Cube-based block-of-skewers (BOS)*

In the original PPI algorithm, the number of skewers, k must be set to be very large in order to ensure that all the desired endmembers will not missed by the PPI algorithm. However, it also requires a huge number of dot products which must be performed between each of k skewers and each data sample vector in the image. As a result, computational complexity is usually very intensive. Recently, [10] looked into this issue and developed the so-called Block of Skewers (BOS) method to mitigate this dilemma. In their BOS method a block size, B for skewers is proposed and the skewers to be used for the PPI are linear combinations of the skewers that form the block B. By virtue of the BOS, the number of dot products is significantly reduced, which is the key to make hardware implementation possible. The block diagram of the BOS method is depicted below in Fig. 1.

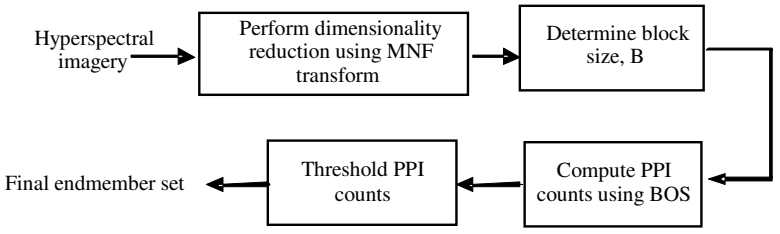


Fig. 1. A block diagram of BOS Method

The idea of the BOS method can be explained by using an example of block size 3 for illustration shown in Fig. 2 with 9 pixels in an *L*-dimensional image cube.

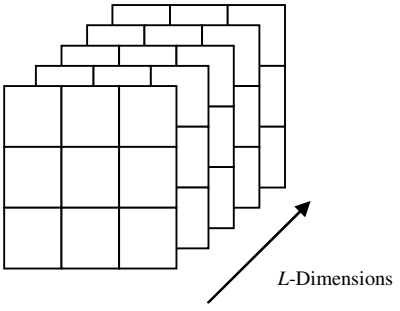


Fig. 2. *L*-Dimensional Image Cube

Assume that there are three skewers specified by $\{\text{skewer}_i\}_{i=1}^{3}$. The BOS method uses $\{\text{skewer}_i\}_{i=1}^{3}$ to form a basic block of size 3 and generates all remaining skewers that are linear combinations of these three skewers via the following equation

$$a_1\mathbf{skewer}_1 + a_2\mathbf{skewer}_1 + a_3\mathbf{skewer}_3 \tag{2}$$

with three real-valued combinatorial coefficients a_1, a_2, and a_3. If the coefficients a_1, a_2, and a_3 are constrained to either +1 or -1, that is, $a_i \in \{1,-1\}$, there are 8 linear combinations that can be generated as additional skewers for the PPI implementation. In this case, a total of 11 skewers can be used as skewers for the PPI. Among these 11 skewers the three $\{\text{skewer}_i\}_{i=1}^{3}$ are independent skewers and the remaining 8 skewers generated by (2) are dependent skewers. Unfortunately, similar drawbacks to those found in the PPI are also occurred in the BOS. In particular, the BOS-PPI does not provide a criterion as to how to choose an appropriate block size B. In addition, the selection of $a_i \in \{1,-1\}$, which is essentially the corners of the cube, is not necessarily optimal. Such an issue can be addressed by a new proposed pyramid-based BOS PPI described in Section 3.2. In other words, the BOS method also suffers from additional drawbacks which do not occur in the PPI algorithm.

1. It does not provide any clue to choose the block size *B*. It must be done by trial and error.

2. Obviously, the selection of linear combination coefficients in step 4 to be $a_{ji} \in \{-1,1\}$ is not optimal.

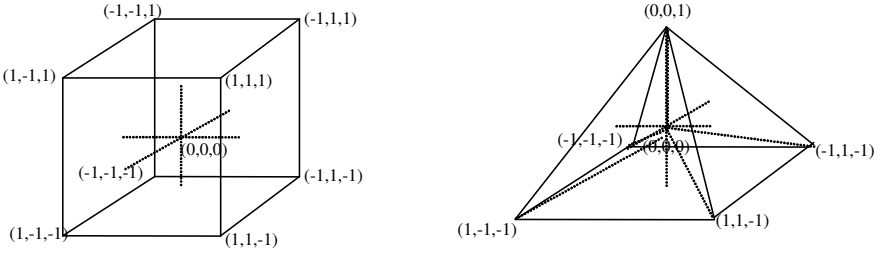
3.2. *Pyramid-based block-of-skewers (BOS)*

The pyramid-based BOS-PPI was originally proposed in [11] which can be summarized as follows.

Using the VD to determine the number of dimensions required to be retained in the Minimum Noise Fraction (MNF) transformed image, the pyramid-based BOS-PPI can be described by the following steps.

1. Apply the MNF transform to perform dimensionality reduction.
2. Determine the block size to be used, B.
3. Generate a set of B skewers randomly, denoted by {$\mathbf{skewer}_1$, $\mathbf{skewer}_2$, … , $\mathbf{skewer}_B$} where for $1 \le j \le B$ $\mathbf{skewer}_j$ is an L-dimensional column vector given by $\mathbf{skewer}_j = (skewer_{j1}, \cdots, skewer_{jL})^T$ and set $n = 1$.
4. Find a set of block skewers which are given by $\mathbf{Bskewer}_j = \sum_{i=1}^{B} a_{ji}\mathbf{skewer}_i$ with $a_{ji} \in \{-1,1\}$. As a result, there are 2^B block skewers, denoted by $\mathbf{Bskewer}_1, \mathbf{Bskewer}_2, \cdots, \mathbf{Bskewer}_{2^B}$ where for each $1 \le j \le 2^B$, $\mathbf{Bskewer}_j$ is also an L-dimensional column vector given by $\mathbf{Bskewer}_j = (Bskewer_{j1}, \cdots, Bskewer_{jL})^T$.
5. For each data sample pixel vector $\mathbf{r}$, calculate the dot product of $\mathbf{r}$ with each of 2^B block skewers $\{\mathbf{Bskewer}_1, \mathbf{Bskewer}_2, \cdots, \mathbf{Bskewer}_{2^B}\}$ by $<\mathbf{Bskewer}_j, \mathbf{r}> = \sum_{i=1}^{B} a_i <\mathbf{skewer}_i, \mathbf{r}>$ where $<\mathbf{skewer}_j, \mathbf{r}>$ was already calculated in step 4.
6. For all the sample pixel vectors, find $N_{PPI}(\mathbf{r}) = \sum_j I_{S_{extrema}(\mathbf{Bskewer}_j)}(\mathbf{r})$ defined by Eq. (1), denoted by $\{\mathbf{r}_j^{(n)}\}_{j=1}^{k}$.
7. Let $n \leftarrow n+1$. If $n < 3$, go to step 4. Otherwise, continue.
8. Find $E^{(n)} = \bigcap_{m=1}^{n} \{\mathbf{r}_j^{(m)}\}_{j=1}^{k}$ and $E^{(n-1)} = \bigcap_{m=1}^{n-1} \{\mathbf{r}_j^{(m)}\}_{j=1}^{k}$.
9. If $E^{(n)} \neq E^{(n-1)}$, go to step 4. Otherwise, the algorithm is terminated. In this case, the sample pixel vectors in $E^{(n)}$ are the desired endmembers.

Furthermore, based on the experimental study conducted in [11], a small block size generally performed better than a large block size by reducing the number of linear combinations as well as redundancy. Empirically, it was concluded that block size of 3 is a good choice. Block size of 2 also performs well but less effective than B = 3. But with this block size it is almost the same as the original PPI. Consequently, in our design, we use block size of B = 3 and the coefficient a_i's are assumed to be either -1 or 1. In this case, there are 8 linear combinations which form the eight vertices of a cube with its center at the origin (0,0,0) as shown in Fig. 3(a). This is the case which "Cube-based Block of Skewers" is implemented. On the contrary, in the pyramid based design, the number of linear coefficient combinations for the block of skewers will be reduced to



Fig, 3. Cube-based and Pyramid-based Block of Skewer

(0, 0, 1), (1, 1, -1), (1, -1, -1), (-1,1, -1) and (-1, -1, -1), as pictorially shown in Fig. 3(b) where only 5 derived skewers will be used compared to 8 used for cube based design.

One comment on the selection of (0, 0, 1), (1, 1, -1), (1, -1, -1), (-1,1, -1) and (-1, -1, -1) to form the pyramid in Fig. 3(b) is noteworthy. The idea beyond the pyramid design is derived from the fact that a cube can be decomposed into 4 pyramids as shown in Fig. 4, referred to as east, west, north and south pyramids where pyramids form basic blocks for a cube.

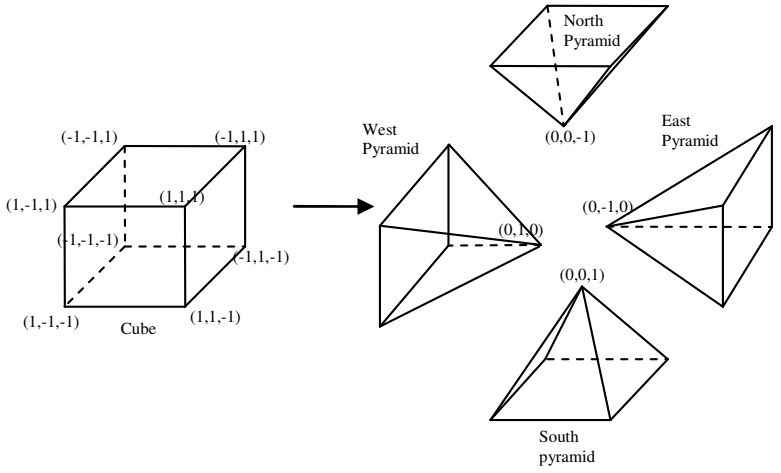


Fig. 4. Decomposition of a cube into four pyramids

As a result, any one of these four pyramids can be used for the pyramid design and produces similar results. In this paper, the south pyramid is selected for our pyramid design with no particular preference except its better visual interpretation. Since the center point (0,0,0) in Fig. 3(a) cannot be used as the top vertex of a pyramid, it is linearly translated to the point (0,0,1). However, it should be noted that such a linear translation does not have any impact on the pyramid design since the BOS is performed by linear combinations.

4. Experimental Results

The HYDICE image is shown in Fig. 5(a), which has a size of 64×64 pixel vectors with 15 panels in the scene.

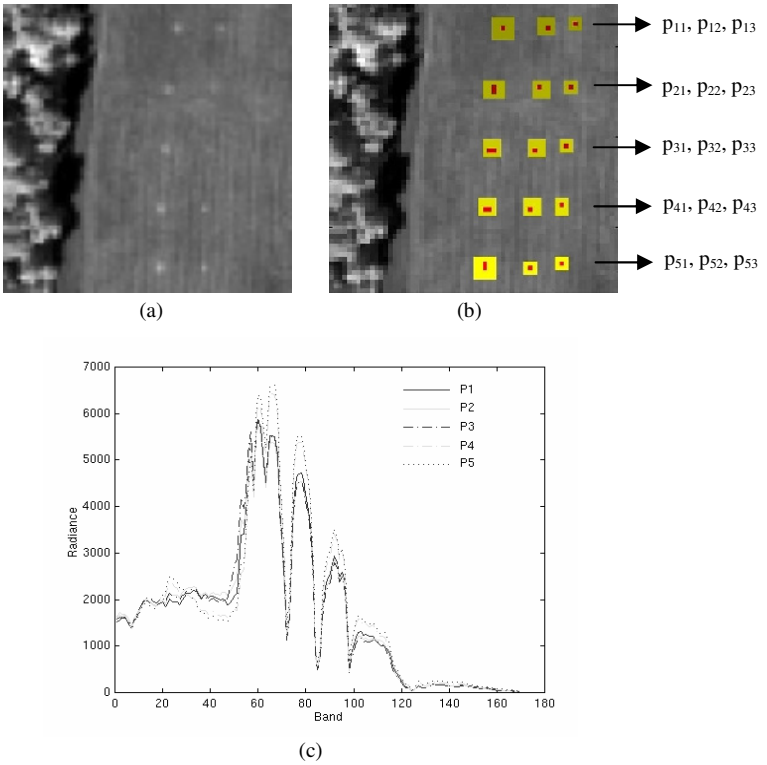


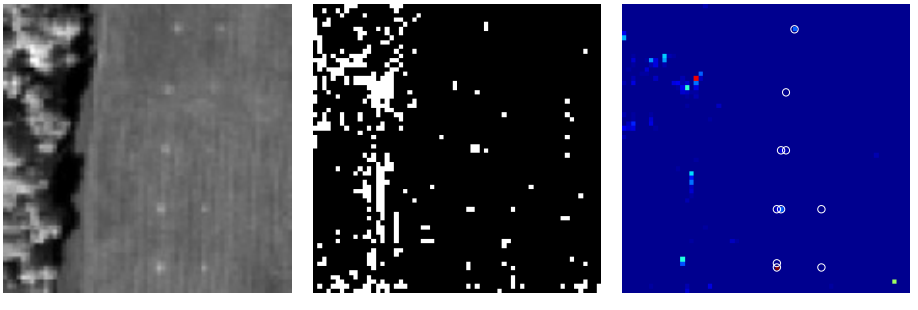
Fig. 5. (a) A HYDICE panel scene which contains 15 panels; (b) Ground truth map of spatial locations of the 15 panels; (c) Spectral signatures of p_1, p_2, p_3, p_4 and p_5

It has 210 spectral bands with a spectral coverage from 0.4μm to 2.5μm. Low signal/high noise bands: bands 1-3 and bands 202-210; and water vapor absorption bands: bands 101-112 and bands 137-153 were removed. So, a total of 169 bands were used. The spatial resolution is 1.56m and spectral resolution is 10nm. Within the scene in Fig. 5(a) there is a large grass field background, a forest on the left edge and a barely visible road running on the right edge of the scene. There are 15 panels located in the center of the grass field and are arranged in a 5×3 matrix as shown in Fig. 5(b) which provides the ground truth map of Fig. 5(a). There are 15 panels located on the field and are arranged in a 5×3 matrix as shown in Fig. 5(b) of a ground truth map. Each element in this matrix is a square panel and denoted by p_{ij} with row indexed by $i=1,2,\cdots,5$ and column indexed by $j=1,2,3$. For each row $i=1,2,\cdots,5$, the three panels p_{i1}, p_{i2}, p_{i3} were made by the same material but have three different sizes. For each column $j=1,2,3$, the five panels

$p_{1j}, p_{2j}, p_{3j}, p_{4j}, p_{5j}$ have the same size but were made by five different materials. It should be noted that the panels in rows 2 and 3 are made by the same material with different paints, so did the panels in rows 4 and 5. Nevertheless, they were still considered as different materials. The sizes of the panels in the first, second and third columns are $3m \times 3m$, $2m \times 2m$ and $1m \times 1m$ respectively. So, the 15 panels have five different materials and three different sizes. The ground truth map provides the precise spatial locations of these 15 panels. As shown in Fig. 5(b), black pixels are the panel center pixels and the pixels in the white masks are either panel boundary pixels mixed with background pixels or background pixels close to panels. The 1.5 m-spatial resolution of the image scene suggests that all of these panels are only one-pixel wide except that $p_{21}, p_{31}, p_{41}, p_{51}$ which are two-pixel panels, all the remaining panels are one-pixel wide. Fig. 5(c) plots the five panel spectral signatures in Fig. 5(b) where the *i*-th panel signature, denoted by p_i was obtained by averaging the black panel center pixels in row *i*.

4.1. *Cube-based BOS Design*

The cube-based BOS was performed repeatedly on the real HYDICE image for 1250 times to generate 10,000 linear combinations of 3 independent skewers. A total of 363 pixels were extracted in Fig. 6 with the PPI count image and the color PPI count image shown in Fig. 6(b-c). Among the extracted pixels, there were 10 panel pixels extracted in Fig. 6(c) which represent five distinct panel signatures, $\{\mathbf{p}_i\}_{i=1}^{5}$.

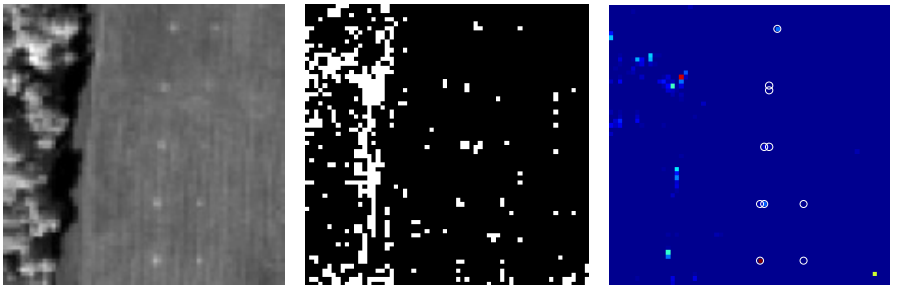


(a) Original image (b) PPI count image (c) Color PPI count image

Fig. 6. Results of cube-based BOS for real HYDICE Image (Repetitions = 1250)

4.2. *Pyramid-based BOS Design*

Similar experiments were also performed for the pyramid-based BOS-PPI. In this case, the pyramid based block was also repeated 1250 times to generate 5000 linearly dependent skewers. A total of 432 pixels were extracted by the pyramid-based BOS in Fig. 7 with the PPI count image and the color PPI count image shown in Fig. 7(b-c). Among the extracted pixels, there were 10 panel pixels extracted in Fig. 7(c) which represent five distinct panel signatures $\{\mathbf{p}_i\}_{i=1}^{5}$.



(a) Original image (b) PPI count image (c) Color PPI count image

Fig. 7. Results of pyramid-based BOS for real HYDICE Image (Repetitions = 1250)

Comparing the results in Fig. 7 to that in Fig. 6 the pyramid-based design and the cube-based design performed very similarly with the pyramid-based design using fewer skewers than that used by the cube-based design.

5. FPGA Implementation of Pyramid Based BOS PPI Algorithm

In this section, we further discuss the hardware implementation of our proposed pyramid based BOS-PPI. We first focus on the part of the algorithm which represents the most significant fraction of the execution time. Obviously, the projections of the image data cube onto the independent skewers that form the BOS dominate the computational cost. Here, we focus on this particular attribute for hardware acceleration to account for its highly parallelizable operations. Table 2 provides the pseudo-code implementation for the part of the BOS-PPI. Due to the limitation of the hardware resources, a single FPGA chip might not be able to accommodate the entire Hyperspectral image scene in one shot. As a result, the image will be processed partially in parallel block-wise for the hyperspectral image cube fed into the hardware at a time. By way of this processing, the entire image will be processed after several iterations. By the end, the algorithm will be repeated until a stopping rule is met. An MINMAX module is used to store the pixel location if it is greater than a selected local extreme.

Table 2. Pseudo-code for parallel implementation of BOS-PPI algorithm

```
do
  for-parallel (allowed image partition for hardware capacity)
    for-parallel (each pixel "r" in image partition)
      for-parallel (each skewer "s")
          Product_{j,band} = P.E.(r[i^th pixel][band],s[j^th skewer][band]);
          temp-summation_{j,band} = temp-summation_{j,band-1} + Product_{j,band};
          pass "temp-summation_{j,band}" to the next P.E.
      endfor
    endfor
  endfor
    MINMAX; // log the local extrema and memorize the spatial coordinates
    // change set of skewers according to the results
while (meet stopping rule?)
```

Based on the pseudo-code provided in Table 2, the architecture of the proposed pyramid-based BOS-PPI is shown in Fig. 8. Three independent skewers are used to generate four linearly dependent skewers. The modules to implement the pyramid-based BOS-PPI are described as follows. The block diagrams for the individual design are shown in Figs. 9, 10 and 11.

Module 1: Dot-product module – a group of Processing Element (P.E.) for performing dot product in parallel. P.E. is the basic computation component.
Module 2: Pyramid-based Block of Skewer Generator – compute the dot-product results for the block of skewers from the independent skewers
Module 3: MINMAX module – compare the incoming dot-product result with the current Maximum and Minimum value in the memory. Bank of MINMAX modules will be duplicated and used to speed up the output.

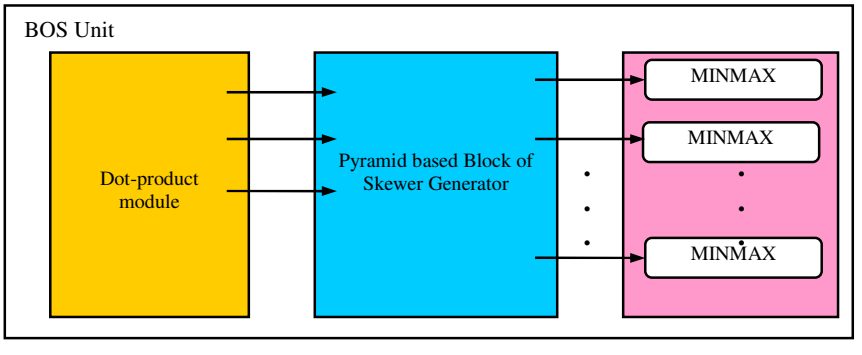


Fig. 8. Pyramid-based BOS-PPI Architecture

Table 1 tabulates the computational complexity (CC) of the original PPI, cube-based BOS-PPI and pyramid-based BOS-PPI where K is the number of independent skewers, B is the size of Block of skewers, L is the number of spectral bands, and N is the total number of pixel vectors [12]. As we can see from the table, the CC of cube-based BOS-PPI is 3/8 of the original PPI, because it only uses 3/8 of independent skewers required for the original PPI to achieve the same amount of dot products. Despite the fact that it may require extra time to perform the linear combinations of dot-product resulting from independent skewers, the addition-based computational time can be neglected compared with a large number of multiplications. In comparison between pyramid and cube based design, the former only needs half of additions that are required by the latter.

Table 1. Computational complexity of different PPI implementations

	Original PPI	Cube-based BOSPPI	Pyramid-Based BOSPPI
Multiplication	$K \times L \times N$	$(3/8) \times K \times L \times N$	$(3/8)K \times L \times N$
Addition	0	$K \times N$	$(1/2) \times K \times N$

Fig. 9 shows the design of the dot-product module. The dot products of every incoming pixel vector with the three original independent skewers are computed in parallel and fed into the Pyramid based Block of Skewer PPI Generator. The P.E. (Processing Element) is a building block of dot-product module. Based on the functionality, we name it as P.E.1 or P.E.2. The difference between these two blocks is that P.E.1 only does multiplication while P.E.2 performs accumulation after the multiplication. As a result, P.E.2 is fundamentally equivalent to a MAC (Multiplier and ACcumulate) computation unit. On a whole, each P.E. is responsible for computing the multiplication of the original skewers with incoming pixel vector in a particular band. Then it accumulates the result from the previous P.E. and passes it on to the next P.E. until it reaches the MINMAX module. The data is transmitted in serial to the data bus so that the data can be de-serialized before performing any computation. The same rule is also applied to the MINMAX module.

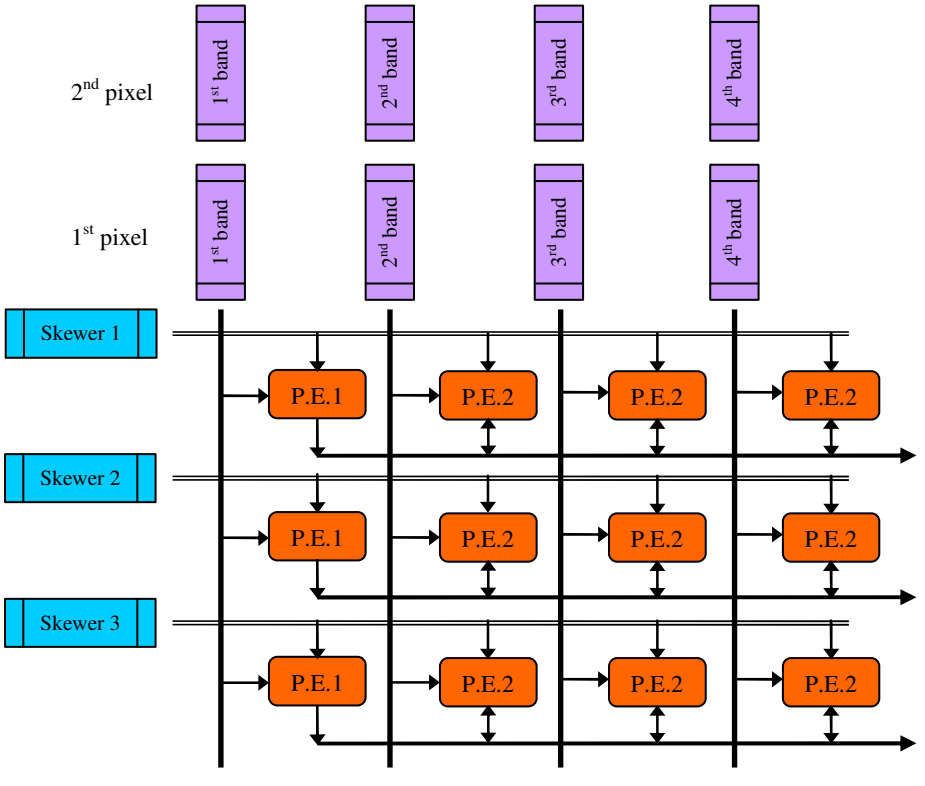


Fig. 9. Dot-Product modules

As discussed in the previous sections, the BOS method generates new skewers using dot products calculated from the independent random skewers. The pyramid-based BOS-PPI generator generates the bottom four corners of a pyramid by linear combinations of

three independent skewers. The design can be accomplished as shown in Fig. 10 where a square box "F/AS" performs as three input full-adder/subtractor to sum up the projection results of the linearly dependent skewers.

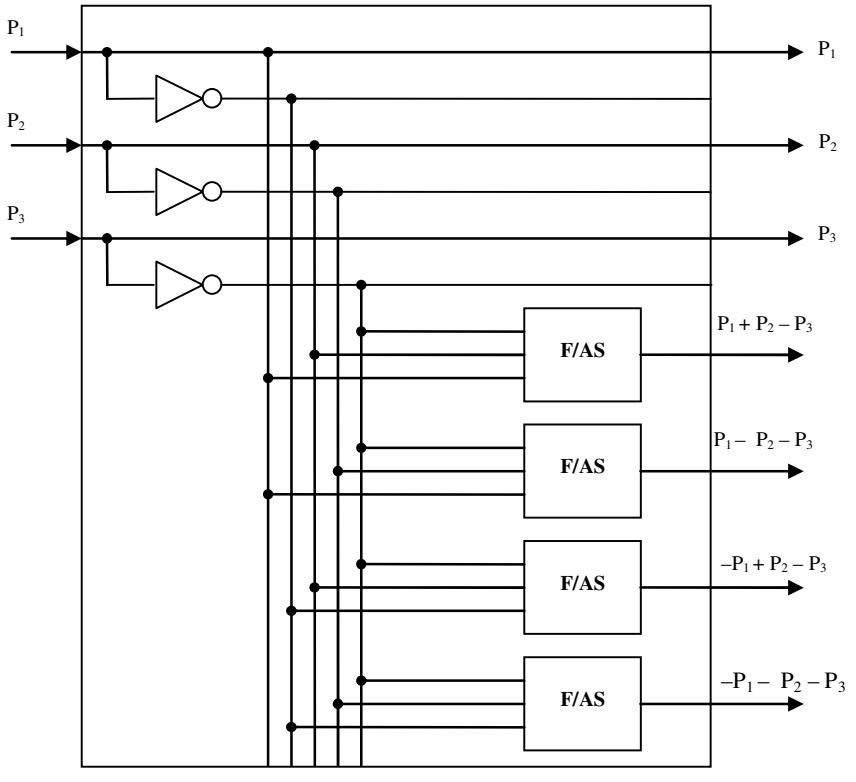


Fig. 10. Pyramid-based block of skewer generator

Fig. 11 shows architecture of a MINMAX module which finds the minimum and the maximum of the projected scores of the pixels onto the original skewers and pyramid generated skewers. It updates 2 registers, MIN and MAX by comparing the current result "C" to the value stored in the MIN and MAX registers. The Min comparator compares the "C" against the value stored in "MIN". If C<MIN, the "Min Comparator" updates the MIN register by "C". Same rule is also applied to the "Max Comparator". If the new dot product is greater than the previous value stored in the MAX register, it will be updated with the new max value. A separate MINMAX module will be used for each individual skewer.

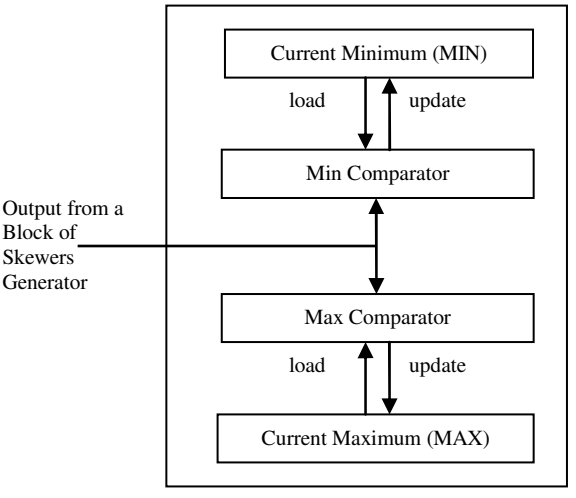


Fig. 11. Architecture of a MINMAX module

6. Conclusions

This paper investigates two endmember extraction methods, the PPI and the cube-based blocks of skewers (BOS) method. A new PPI-based BOS method, called a pyramid-based block of skewers (BOS) method is derived to address some issues encountered in the BOS and its structure also makes it better suited for hardware acceleration. The experimental results demonstrate that block design using pyramids produces nearly the same results as cube-based BOS, the one proposed in [10]. This suggests that the pyramid-based block design can perform as well as the cube-based block design where the former requires fewer skewers than the latter. This is of particular importance for hardware design and implementation due to the reduction of the computational complexity.

7. References

1. R.A. Schwengerdt, *Remote Sensing: Models and Methods for Image Processing*, 2nd. Ed., Academic Press, 1997, p. 447.
2. J.W. Boardman, F.A. Kruse and R.O. Green, "Mapping target signatures via partial unmixing of AVIRIS data," *Summaries of JPL Airborne Earth Science Workshop*, Pasadena, CA, 1995.
3. M.E. Winter, "N-finder: an algorithm for fast autonomous spectral endmember determination in hyperspectral data," *Image Spectrometry V*, Proc. SPIE 3753, pp. 266-277, 1999.
4. R.A. Neville, K. Staenz, T. Szeredi, J. Lefebvre and P. Hauff, "Automatic endmember extraction from hyperspectral data for mineral exploration," *4th International Airborne Remote Sensing Conf. and Exhibition/21st Canadian Symposium on Remote Sensing*, Ottawa, Ontario, Canada, pp. 21-24, June 1999.
5. A. Plaza, P. Martinez, R. Perez and J. Plaza, "Spatial/spectral endmember extraction by multidimensional morphological operations," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 40, no 9, pp. 2025-2041, September 2002.

6. M.D. Craig, “Minimum-volume transforms for remotely sensed data," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 32, no 3, pp. 542-552, May 1994.
7. J.W. Boardman, “Geometric mixture analysis of imaging spectrometery data,” *Proc. Int. Geoscience and Remote Sensing Symp*., vol. 4, pp. 2369-2371, 1994.
8. A. Ifarraguerri and C.-I Chang, "Hyperspectral image segmentation with convex cones," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 37, no 2, pp. 756-770, March 1999.
9. A. Plaza, P. Martínez, R. Pérez and J. Plaza, “A Quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data,” *IEEE Trans. on Geoscience and Remote Sensing*, vol. 42, no 2, pp.650-663, March 2004.
10. J. Theiler, D.D. Lavenier, N.R. Harvey, S.j. Perkins and j.J. Szymanski, “Using blocks of skewers for fast computation of pixel purity index,” Proc. SPIE, vol. 4132, pp. 61-71, 2000.
11. F. Chaudhry, “Pixel Purity Index-Based Endmember Extraction for Hyperspectral Data Exploitation”, M.S. thesis, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD, 2005.
12. D.Lavenier and J. Theiler, "FPGA implementation of the Pixel Purity Index algorithm for hyperspectral images," *Tech Rep. LA-UR-00-2426*, Los Alamos National Laboratory, 2000.