Transworld Research Network
37/661 (2), Fort P.O., Trivandrum-695 023, Kerala, India

Recent Advances in Hyperspectral Signal and Image Processing, 2006:
ISBN: 81-7895-218-1 Editor: Chein-I Chang

Pixel purity index-based algorithms for endmember extraction from hyperspectral imagery

Farzeen Chaudhry[1], Chao-Cheng Wu[1], Weimin Liu[1], Chein-I Chang[1] and Antonio Plaza[2]
[1]Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250; [2]Computer Science Department, University of Extremadura, Avda. de la Universidad s/n 10.071 Caceres, Spain

Abstract

Pixel purity index (PPI) has been widely used in hyperspectral image analysis for endmember extraction because of its publicity and availability in the ENvironment for Visualizing Images (ENVI) developed by Research Systems. Unfortunately, its detailed implementation has never been made

Correspondence/Reprint request: Dr. Chein-I Chang, Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA. E-mail: cchang@umbc.edu

available in the literature. This chapter investigates the PPI based on limited published sources and reinterprets the PPI from a statistical signal processing view point. So, the first step is to develop a version of the PPI which can be implemented in any general-purpose scientific package without appealing for the ENVI, so that the users with no access to the ENVI can still run their own PPI algorithm (using for instance the MATLAB software). Since the main idea of the PPI is to generate a sufficiently large number of random vectors, referred to as skewers to cover all possible directions, the final set of PPI-generated endmembers may vary due to the use of different sets of skewers. In order to resolve such an inconsistent final selection set of endmembers, two algorithms are also developed. One is called Automatic PPI (APPI) which takes advantages of random nature of the skewers to implement the PPI as many runs as possible with different sets of skewers, then finds their common endmembers for all runs. The APPI is terminated automatically when the common endmembers remain unchanged between two consecutive runs, in which case the final common endmembers are desired endmembers. In contrast to the APPI, a second algorithm, to be called Fast Iterative PPI (FIPPI) makes use of an initialization algorithm to produce an appropriate set of initial endmembers so that the final set of endmembers converges to desired endmembers. In order to ease computational complexity as well as to compact the information, PPI-based algorithms generally require either the Principal Components Analysis (PCA) or Maximum Noise Fraction (MNF) transform to perform dimensionality reduction. Unfortunately, the PCA and MNF are second order statistics-based transforms, they do not necessarily preserve information of high order statistics. Therefore, this chapter further investigates the ICA-based PPI-based algorithms where the MNF originally used in the PPI-based algorithms is replaced by the ICA to perform dimensionality reduction. Experimental results demonstrate that the ICA-based PPI generally performs more effectively than the MNF-based PPI in endmember extraction.

1. Introduction

Endmember extraction has received considerable interest in hyperspectral image analysis in recent years. According to the definition given in [1], an endmember is an idealized pure signature for a class. Finding pure signatures from hyperspectral imagery is considered to be an important and crucial step in hyperspectral data exploitation, particularly, classification. Many endmember extraction algorithms (EEAs) have been developed for this purpose. One of the most popular EEAs is the pixel purity index (PPI) developed by Boardman et al. [2] which was designed to search for a set of vertices of a convex geometry in a given data set that are supposed to represent pure signatures present in the data. The PPI has been widely used due to its publicity and availability

provided by the ENVI developed by the Research Systems. On the other hand, also due to its propriety and limited published results, its detailed implementation has never been made available in the public domain. Therefore, most of the people who use the PPI for endmember extraction either appeal for the ENVI software or implement their own versions of the PPI based on whatever available in the literature. This chapter presents our experience with the PPI and investigates several issues resulting from the implementation of the PPI. One of major issues is the sensitivity of two parameters used in the PPI which are k (number of so-called skewers required for implementation) and t (cut-off threshold value for the scores produced by the PPI to extract candidate pixel vectors for final selection of endmembers). Another is the issue of initial skewers that are randomly generated by the PPI. The ENVI users do not allow option to choose their own initial skewers. A third issue is the requirement of human intervention to manually select a final set of endmembers via a visualization tool provided by the ENVI. Most importantly, the PPI is not an iterative process. Therefore, it does not guarantee that the PPI-found endmembers are actually true endmembers.

In order to address these issues, this chapter develops three algorithms to implement the PPI. The first one, called MATLAB-based PPI (although any other language such as Research Systems IDL or C++ could be used for implementation) retains three specific features of the ENVI's PPI, dimensionality reduction, random initial endmembers, called "skewers" and non-iterative algorithmic structure. It is based on the concept of the ENVI's PPI in the ENVI software but can be implemented by standard programming languagues without resorting to ENVI. It keeps track of the PPI score for each of sample pixel vectors, which is defined as number of times a sample pixel vector projected on the extrema of the skewers used in the PPI. This is in contrast to the ENVI's PPI that does not automatically provide PPI scores for all the sample pixel vectors, instead it requires users to manually select endmembers. Our MATLAB-based PPI was verified by the PPI in the ENVI in the sense that both produced very close results.

Another is called Fast Iterative PPI (FIPPI) recently developed by Chang and Plaza [3] which is completely different from the the ENVI's PPI. It eliminates the three specific features required the ENVI's PPI. First of all, it makes use of a recently developed concept, called Virtual Dimensionality (VD) [4-5] to estimate the number of initial endmembers required to initialize the algorithm. Secondly, it is an iterative algorithm which implements an initialization algorithm to produce an appropriate set of initial endmembers rather than skewers. Since it is iterative, an iterative rule is included to improve subsequent iterations. A stopping rule is also provided to terminate the algorithm so that the final set of endmembers converges to desired true endmembers. Because of nature in its iterative process, the selection of initial

endmembers plays a key role in its convergence. Therefore, skewers generated by the ENVI's PPI may no longer be appropriate. Thirdly, the FIPPI takes advantage of an initialization algorithm to produce a better set of initial endmembers to achieve fast convergence. As a consequence of such an iterative process, the FIPPI extracts only one pixel for each of endmembers as opposed to the ENVI's PPI and MATLAB-based PPI that extract all sample vectors corresponding to each of endmembers based on their PPI scores.

As a complete opposite of the FIPPI, a third algorithm, called Automatic PPI (APPI) was developed recently in [6] to take advantage of random nature of skewers selected for the PPI. The idea is to implement the PPI independently over and over again until the common endmembers in final sets remain unchanged. Since each independent run of the PPI generates the same number of different skewers, the final set of selected endmembers produced by each run of the PPI is also different. Nevertheless, if there exist endmembers in the data, these endmembers should be selected by each run no matter what skewers are selected initially as long as the number of skewers is sufficiently large. Consequently, the APPI automatically produces a final set of desired endmembers without human intervention.

When the PPI is implemented, it generally requires Principal Components Analysis (PCA) [7] or Maximum Noise Fraction (MNF) [8-9] transform to perform dimensionality reduction to ease computational complexity as well as compact information in transformed components. However, due to the fact that both the PCA and MNF are second order statistics-based transforms, they cannot effectively capture information of high order statistics which is particularly important and crucial in endmember extraction. This is because endmembers are generally rare and can be considered as insignificant targets whose information may only be characterized by high order statistics. In order to resolve such dilemma, this chapter further investigates PPI-based algorithms which utilize a high order statistics-based transform, Independent Component Analysis (ICA) [10] rather than second order statistics-based transforms to perform dimensionality reduction. Experimental results demonstrate that the ICA-based PPI generally performs more effectively than the MNF-based PPI in endmember extraction. It should be noted that the PPI using the PCA for dimensionality reduction was not as effective as that using the MNF. So, in this chapter, only the MNF was adopted for dimensionality reduction in the PPI experiments for comparative analysis.

2. Pixel purity index (PPI)-based algorithms

The PPI has been available through the ENVI and has been widely used for endmember extraction. Unfortunately, the lack of detailed step-by-step implementations of the ENVI's PPI seems to prevent it from being used by those who do not have ENVI software. So, in this section, we intend to offer

our understanding of the ENVI's PPI and provide our version of its implementations so that those who are interested in the PPI can easily implement them to repeat our experimental results without appealing for particular software.

2.1. MATLAB-PPI

The MATLAB software package has been widely used in engineering. For this reason, we consider highly desirable to develop a MATLAB-based PPI algorithm, as described below, to implement the algorithm in the same way that ENVI's PPI does, but without appealing for the ENVI software.

MATLAB-PPI algorithm

1. Find the VD to estimate the number of bands required for dimensionality reduction.
2. Apply the MNF or PCA transform to reduce dimensionality.
3. Initialization:
   Let k be a sufficiently large positive integer and use a random generator available in the MATLAB to produce a set of k unit vectors called "skewers", $\{\mathbf{skewer}_j\}_{j=1}^{k}$.
4. For each $\mathbf{skewer}_j$, all the data sample vectors are projected onto $\mathbf{skewer}_j$ to find sample vectors at its extreme positions to form an extrema 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 extrema set $S_{extrema}(\mathbf{skewer}_j)$, it is very likely that some sample vectors may appear in more than one extrema set. Define an indicator function of a set S, $I_S(\mathbf{r})$ by

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

   where $N_{PPI}(\mathbf{r})$ is defined as the PPI score of the sample vector $\mathbf{r}$.
5. Find the PPI scores $N_{PPI}(\mathbf{r})$ for all the sample vectors defined by (1).
6. Let t be a threshold value set for the PPI score. Extract all the sample vectors with $N_{PPI}(\mathbf{r}) \geq t$ which will be the desired set of final endmembers. Usually, the threshold t can be set to 1.

The above PPI algorithm is obviously not an iterative process. It is performed for a set of k skewers. Once it is done, the process is terminated. In addition, in order to ensure that our MATLAB-PPI algorithm operates in the same way that the ENVI's PPI does, all the steps described above have been verified via experiments by the PPI in the ENVI 3.6 version with no use of a

visualization tool provided in ENVI's PPI where both versions produce the same results. Several drawbacks can be observed from the above PPI algorithm.

1. Since the PPI algorithm is not an iterative process, it does not guarantee that all the PPI-generated endmembers are actually true endmembers due to the fact that the *k* skewers are randomly generated. Different runs for implementing the PPI algorithm may produce different sets of endmembers.
2. Since the PPI algorithm is very sensitive to noise, the noise estimation used in the MNF transform is crucial.
3. When the MNF transform is implemented for dimensionality reduction, there is no provided guideline that helps users select how many dimensions needed to be retained.
4. No criteria are provided for how to select the parameter *k* and how to select an appropriate threshold *t* which determines the number of endmembers.
5. It requires human intervention to manually select endmembers via a visualization tool.

2.2. Fast Iterative PPI (FIPPI)

As indicated, the PPI-based algorithms discussed above are non-iterative. To the contrary, the FIPPI is an iterative algorithm which iteratively searches for a better set of candidates for endmembers after each iteration. When it converges, the final set of sample vectors will be the desired set of endmembers. Unlike the PPI which generates initial skewers randomly, the FIPPI takes advantage of the automatic target generation process (ATGP) implemented in the automatic target detection and classification algorithm (ATDCA) proposed by Ren and Chang [4,11] to generate an appropriate set of initial endmembers that speed up the algorithm to converge rapidly. The FIPPI also provides a new iterative rule and a stopping rule to ease tremendous computation involved in the PPI.

On the other hand, the FIPPI is also completely automatic and unsupervised. It is terminated as long as the implemented stopping rule is met. This is considered to be one of the most significant advantages of the FIPPI over the ENVI's PPI and MATLAB-PPI. For various users of the PPI they may select different sets of endmembers for the same set of data based on their visualization. Such a problem can be avoided by the FIPPI because the FIPPI are automatic and the final set generated by the FIPPI is always the same regardless of who is a user of the FIPPI. One major difference between the FIPPI and the PPI is that the FIPPI generates one pixel for one endmember compared to the PPI which extracts all pixels corresponding to endmembers.

The PPI has been widely used in remote sensing community because of its inclusion in ENVI software package. However, according to our experience, it suffers from several major drawbacks. First and foremost, its computational complexity is very high. For $k = 10^4$ runs, the algorithm took more than 50 minutes to project every data sample vector of the Cuprite AVIRIS image scene into 10^4 skewers in a PC with AMD Athlon 2.6 GHz processor and 512 MB of RAM. It should be noted that in order to reduce such high computational complexity, most of the ENVI users preprocess the data by the maximum noise fraction (MNF) transform so that the original data dimensionality can be reduced to ease computations. Second, the PPI does not have a specific stopping rule, nor does it use any convergence criterion. It is recommended by the authors to run the algorithm using as many runs as possible in order to obtain optimal results. As a result, the PPI can only guarantee to produce optimal results asymptotically. In addition, based on our experiments the PPI is also very sensitive to the initial values of parameters *k* and particularly, *t*. Furthermore, the ENVI's PPI makes use of a built-in algorithm to randomly generate a large set of so-called skewers and the users of the ENVI's PPI cannot select their own initial sets of endmembers. A final major drawback of the PPI is the need for a supervised procedure to manually select a final set of endmembers which largely depends upon human interpretation. In order to address these issues, a fast iterative algorithm to implement PPI is developed and presented in this section. It is referred to as fast iterative PPI (FIPPI) and is described in detail as follows.

FIPPI algorithm

1) Initialization: Find the VD using the Harsanyi-Farrand-Chang (HFC) method in [6] and use it as an estimate of *p*, the number of endmembers required to generate.
2) Dimensionality reduction: Apply the MNF transform for dimensionality reduction and retain the first *p* components. Let $\left\{\mathbf{skewer}_j^{(0)}\right\}_{j=1}^{p}$ be an initial set of *p* skewers generated by selecting those pixels that correspond to target pixels generated by ATGP in [7].
3) Iterative rule: At iteration $n \geq 0$, for each $\mathbf{skewer}_j^{(n)}$ all the sample vectors are projected onto this particular $\mathbf{skewer}_j^{(n)}$ to find those which are at its extreme positions to form an extrema set, denoted by $S_{extrema}(\mathbf{skewer}_j^{(n)})$. Find the sample vectors that produce the largest $N_{PPI}(\mathbf{r}_j^{(n)})$ $N_{PPI}(\mathbf{r}_j^{(k)})$ defined by (1) and let them be denoted by $\left\{\mathbf{r}_j^{(k)}\right\}$.

4) Stopping rule: Form the joint set, $\{\mathbf{skewer}_j^{(n+1)}\} = \{\mathbf{r}_j^{(n)}\}_{N_{PPI}(\mathbf{r}_j^{(n)})>0} \cup \{\mathbf{skewer}_j^{(n)}\}$. If $\{\mathbf{skewer}_j^{(n+1)}\} = \{\mathbf{skewer}_j^{(n)}\}$, then no new endmembers are added to the skewer set. In this case, the algorithm is terminated. Otherwise, let $n \leftarrow n+1$ and go to step 3.

It should be noted that in step 1 of initialization, the ATGP method to generate the initial set of p endmembers can be replaced by any method including the one used in the PPI that randomly generates so-called "skewers", $\{\mathbf{skewer}_j^{(0)}\}$ as long as the initialization method provides a good estimate of initial endmembers. But as will be shown in experiments, such randomly generated initial endmembers can only slow down the algorithm.

Several major advantages can be obtained from the proposed FIPPI. First, it is more computationally efficient than the original PPI since the FIPPI is an iterative algorithm and the PPI is not. Second, the parameters k and t are not required. So, there is no need for users to input these values to avoid human subjectivity. Instead, only a desired number of endmembers to generate, p, is assumed to be known in advance. The value of this parameter can be calculated using the HFC method for estimation of VD. Third, by means of p, we can define a stopping rule for the FIPPI. The FIPPI starts with an appropriate set of initial endmember set $\{\overline{\mathbf{e}}_j^{(0)}\}_{j=1}^{p}$ generated by the ATGP and iteratively refines this initial selection by looking at the N_{PPI} values for selected p pixels at each iteration until a set of final endmembers $\{\overline{\mathbf{e}}_j\}_{j=1}^{p}$ is identified via an implemented stopping rule. The FIPPI has been experimentally tested by using the AVIRIS Cuprite data set. It has been found that the FIPPI can converge very rapidly in a small number of iterations, and thus the computational complexity is largely reduced. A final major advantage of the FIPPI over the PPI is that it is fully automated, and does not require any human supervision.

2.3. Automatic PPI (APPI)

The Automatic PPI (APPI) presented in this section inherits the original structure of the PPI, but remedies all the drawbacks in the original algorithm. Specifically, it not only improves the PPI in computational complexity and stability, but also requires no human intervention. Interestingly, unlike the Fast Iterative PPI (FIPPI) proposed in [3] which generates an appropriate set of initial endmembers to replace randomly generated initial endmembers, the APPI actually takes advantage of the nature in random initial endmembers to implement the PPI as a random algorithm in a number of runs until a stopping

rule is satisfied, in which case a final set of desired endmembers is generated. Consequently, the issue of determining the parameter *t* is resolved. In other words, the final set of desired endmembers is automatically determined by the stopping rule without appealing for the parameter *t* and human intervention. To our surprise, the APPI produces nearly the same results by the FIPPI.

Like the PPI, the APPI also requires dimensionality reduction to ease computational burden. This issue can be resolved by using the VD to provide a good estimate for the number of dimensions required to be retained [12].

Automatic PPI (APPI) algorithm

1. Use the VD to determine the number of dimensions required to be retained, denoted by *p*.
2. Apply either PCA or MNF transform for dimensionality reduction and set a counter to $n = 1$ for runs.
3. For each *n* and *K* fixed at twice of n_{VD}, generate *K* skewers, $\{\mathbf{skewer}_j^{(n)}\}_{j=1}^{K}$ randomly. Let $\{\mathbf{r}_i^{(n)}\}_{i=1}^{N}$ denote the *i*-th sample vector considered in the *n*-th run where the total number of sample vectors is *N*.
4. For each $\mathbf{skewer}_j^{(n)}$, all the data sample vectors are projected onto $\mathbf{skewer}_j^{(n)}$ to find sample vectors at its extreme positions to form an extrema set for this particular skewer $\mathbf{skewer}_j^{(n)}$, denoted by $S_{extrema}(\mathbf{skewer}_j^{(n)})$. Despite the fact that a different $\mathbf{skewer}_j^{(n)}$ generates a different extrema set $S_{extrema}(\mathbf{skewer}_j^{(n)})$, it is very likely that some sample vectors may appear in more than one extrema set. Define an indicator function of a set *S*, $I_S(\mathbf{r}_i^{(n)})$ by (1) where $N_{PPI}(\mathbf{r}_i^{(n)})$ is defined as the PPI score of the sample vector $\mathbf{r}_i^{(n)}$. Let $\{score_{PPI}^{(n)}(j)\}_{j=1}^{p}$ denote the *p* highest PPI scores where $score_{PPI}^{(n)}(j)$ is the *j*-th highest PPI score.
5. Find a set of sample vectors that corresponds to $score_{PPI}^{(n)}(j)$, denoted by $E_j^{(n)}$. It should be noted that there generally have more one endmember pixel in the set $E_j^{(n)}$ that corresponds to $score_{PPI}^{(n)}(j)$. Let $\Omega^{(n)} = \bigcup_{j=1}^{p} E_j^{(n)}$ be the set of endmember pixels generated at the *n*-th run.
6. Find $E^{(n)} = \bigcap_{m=1}^{n} \Omega^{(m)} = \bigcap_{m=1}^{n} \left(\bigcup_{j=1}^{p} E_j^{(m)} \right)$, the set of endmember pixels common in all runs up to *n*, i.e., $1 \le m \le n$. Let $n \leftarrow n+1$. If $n < 3$, go to step 3. Otherwise, continue.

7. If $E^{(n)} \neq E^{(n-1)}$, go to step 3. Otherwise, the algorithm is terminated. In this case, the sample pixel vectors in $E^{(n)}$ are the desired endmembers.

It should be noted that step 7 specifies a stopping rule to terminate the APPI algorithm. A spectral measure such as spectral angle mapper (SAM) is required to measure the similarity between two sample vectors. If their similarity is less than ε, they will be considered as two pixels specified by the same signature, i.e., an endmember. When $E^{(n)} = E^{(n-1)}$, the algorithm is terminated. In this case, if the number of elements of $E^{(n)}$ is p, the sample pixel vectors $\mathbf{r}^{(m)}$ for all $1 \le m \le n$ that yields first p largest $N_{PPI}\left(\mathbf{r}_i^{(n)}\right)$ counts will be the same. That is, if we arrange all sample pixel vectors in accordance with magnitude of PPI scores for each run, $1 \le m \le n$. The first p sample pixel vectors in $\left\{\mathbf{r}_j^{(m)}\right\}_{j=1}^{p}$ for each run $1 \le m \le n$ will be identical. This implies that $\left\{\mathbf{r}_j^{(1)}\right\}_{j=1}^{p} = \left\{\mathbf{r}_j^{(2)}\right\}_{j=1}^{p} = \cdots = \left\{\mathbf{r}_j^{(n)}\right\}_{j=1}^{p}$ are the desired p endmembers $\left\{\mathbf{e}_j\right\}_{j=1}^{p}$. In addition, according to our experiments, the number of k skewers set to be $2n_{\mathrm{VD}}$ is sufficiently enough for the APPI to converge.

Despite the fact that the VD was used to estimate highest values of the PPI scores needed to be considered, the actual number of endmembers is generally equal to or smaller than the n_{VD} as will be shown in the experiments.

2.4. ICA-based PPI (ICA-PPI)

One of important steps carried out by the PPI is the use of the MNF transform for dimensionality reduction. However, since the MNF transform is a 2nd order statistics-based transformation, it may not effectively capture information characterized by statistics higher than the 2nd order. This section presents another version of PPI that replaces the MNF transform with the well-known high-order statistics-based transform, called FastICA which was developed by Hyvarinen and Oja based on the Independent Component Analysis (ICA) [13]

2.4.1. ATGP

The automatic target generation process (ATGP) was previously developed to find potential target pixels that can be used to generate a signature matrix used in an orthogonal subspace projection (OSP) approach in [4,14]. It is one of two processes used in the automatic target detection and classification algorithm (ATDCA) developed in [4,11]. It makes use of an orthogonal subspace projector defined in [4,14] by

$$P_{\mathbf{U}}^{\perp} = \mathbf{I} - \mathbf{U}\left(\mathbf{U}^T\mathbf{U}\right)^{-1}\mathbf{U}^T. \quad (2)$$

The ATGP repeatedly makes use of Eq. (2) to find target pixel vectors of interest from the data without prior knowledge regardless of what types of pixels are these targets. It can be briefly described as follows. Assume that $\mathbf{t}_0$ is an initial target pixel vector. The ATGP begins with the initial target pixel vector $\mathbf{t}_0$ by applying an orthogonal subspace projector $P_{\mathbf{t}_0}^{\perp}$ specified by Eq. (2) with $\mathbf{U} = \mathbf{t}_0$ to all image pixel vectors. It then finds a target pixel vector, denoted by $\mathbf{t}_1$ with the maximum orthogonal projection in the orthogonal complement space, denoted by $<\mathbf{t}_0>^{\perp}$ that is orthogonal to the space, $<\mathbf{t}_0>$ linearly spanned by $\mathbf{t}_0$. The reason for this selection is that the selected $\mathbf{t}_1$ generally has the most distinct features from $\mathbf{t}_0$ in the sense of orthogonal projection because $\mathbf{t}_1$ has the largest magnitude of the projection in $<\mathbf{t}_0>^{\perp}$ produced by $P_{\mathbf{t}_0}^{\perp}$. A second target pixel vector $\mathbf{t}_2$ can be found by applying an orthogonal subspace projector $P_{[\mathbf{t}_0\mathbf{t}_1]}^{\perp}$ with $\mathbf{U} = [\mathbf{t}_0\,\mathbf{t}_1]$ to the original image and a target pixel vector that has the maximum orthogonal projection in $<\mathbf{t}_0, \mathbf{t}_1>^{\perp}$ is selected as $\mathbf{t}_2$. The above procedure is repeated over and over again to find a third target pixel vector $\mathbf{t}_3$, a fourth target pixel vector $\mathbf{t}_4$, etc. until a certain stopping rule is satisfied. In this paper, the stopping rule is determined by the number of target pixel vectors required to generate, p which is estimated by the VD. Using the p as a stopping criterion, the ATGP can be implemented in the following steps.

Automatic Target Generation Process (ATGP)

1) Initial condition: Select an initial target pixel vector of interest denoted by $\mathbf{t}_0$. In order to initialize the ATGP without knowing $\mathbf{t}_0$, we select a target pixel vector with the maximum length as the initial target $\mathbf{t}_0$, namely, $\mathbf{t}_0 = \arg\{\max_{\mathbf{r}} \mathbf{r}^T\mathbf{r}\}$, which has the highest intensity, i.e., the brightest pixel vector in the image scene. Set $k = 1$ and $\mathbf{U}_0 = [\mathbf{t}_0]$.

   (It is worth noting that this selection may not be necessarily the best selection. However, according to our experiments it was found that the brightest pixel vector was always extracted later on, if it was not used as an initial target pixel vector in the initialization.)

2) At n-th iteration, apply $P_{\mathbf{t}_0}^{\perp}$ via (2) to all image pixels $\mathbf{r}$ in the image and find the n-th target $\mathbf{t}_n$ generated at the n-th stage which has the maximum orthogonal projection as follows.

$$\mathbf{t}_n = \arg\left\{\max_{\mathbf{r}}\left[\left(P^{\perp}_{[\mathbf{t}_0\mathbf{U}_{n-1}]}\mathbf{r}\right)^T\left(P^{\perp}_{[\mathbf{t}_0\mathbf{U}_{n-1}]}\mathbf{r}\right)\right]\right\} \quad (3)$$

where $\mathbf{U}_{n-1} = [\mathbf{t}_1\mathbf{t}_2\cdots\mathbf{t}_{n-1}]$ is the target matrix generated at the $(n\text{-}1)^{st}$ stage.

3) Stopping rule: If $n < p-1$, let $\mathbf{U}_n = [\mathbf{U}_{n-1}\mathbf{t}_n] = [\mathbf{t}_1\mathbf{t}_2\cdots\mathbf{t}_n]$ be the n-th target matrix, go to Step 2. Otherwise, continue.

4) At this stage, the ATGP is terminated. At this point, the target matrix is $\mathbf{U}_{p\text{-}1}$, which contains p-1 target pixel vectors as its column vectors, which do not include the initial target pixel vector $\mathbf{t}_0$.

2.4.2. ICA-based dimensional reduction

The algorithm presented in this subsection to perform dimensionality reduction is one of two algorithms developed in [12]. It makes use of the ATGP to generate an appropriate set of initial projection vectors which will initialize the FastICA. The details of its implementation is summarized as follows.

ICA-DR algorithm

1. Use the VD to determine the number of dimensions, p, required to be retained.
2. Perform sphereing on the data matrix $\mathbf{X}$ and let the resulting sphered data matrix be denoted by $\hat{\mathbf{X}}$.
3. Apply the ATGP to $\hat{\mathbf{X}}$ to find p target pixel vector, $\{\mathbf{t}_i\}_i^p$.
4. Use the FastICA to find p independent components, $\{\mathrm{IC}_i\}_{i=1}^p$ where the i-th IC_i is generated by the FastICA with the i-th target pixel vector chosen to be the initial projection vector instead of being generated randomly.

2.4.3. ICA-based PPI

ICA-based PPI (ICA-PPI) algorithm

1. Use the VD to determine the number of dimensions to be retained in the ICA-transformed images.
2. Perform dimensionality reduction by the ICA-DR algorithm to retain VD-determined independent components.
3. For each given k, generate k skewers, $\{\mathbf{skewer}_j\}_{j=1}^k$ randomly and set a counter $n = 1$.
4. 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 extrema set for the skewer $\mathbf{skewer}_j$, denoted by $S_{extrema}(\mathbf{skewer}_j)$.
5. Let $n \leftarrow n+1$. If $n < 3$, go to step 3. Otherwise, continue.

6. Find $E^{(n)} = \bigcap_{m=1}^{n} \{\mathbf{r}_j^{(m)}\}_{j=1}^{k}$ and $E^{(n-1)} = \bigcap_{m=1}^{n-1} \{\mathbf{r}_j^{(m)}\}_{j=1}^{k}$.

7. If $E^{(n)} \neq E^{(n-1)}$, go to step 3. Otherwise, the algorithm is terminated. In this case, the sample pixel vectors in $E^{(n)}$ are the final endmembers.

3. Synthetic images to be used for computer simulations

The synthetic image to be studied for experiments is simulated based on an image scene shown in Fig. 1(a), which has a size of 64×64 pixel vectors with 15 panels in the scene and the ground truth map in Fig. 1(b) [4]. It was acquired by 210 spectral bands with a spectral coverage from 0.4μm to 2.5 μm. 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 in experiments. The spatial resolution is 1.56m and spectral resolution is 10nm.

Within the scene in Fig. 1(a) there is a large grass field background, and a forest on the left edge. Each element in this matrix is a square panel and denoted by p_{ij} with rows indexed by i and columns indexed by j = 1,2,3. For each row $i = 1,2,\cdots,5$, there are three panels p_{i1}, p_{i2}, p_{i3}, painted by the same paint but with three different sizes. The sizes of the panels in the first, second and third columns are 3m×3m and 2m×2m and 1m×1m respectively. Since the size of the panels in the third column is 1m×1m, they cannot be seen visually from Fig. 1(a) due to the fact that its size is less than the 1.56m pixel resolution. For each column j = 1,2,3, the 5 panels p_{1j}, p_{2j}, p_{3j}, p_{4j}, p_{5j} have the same size but with five different paints. However, it should be noted that the panels in rows 2 and 3 were made by the same material with two different paints. Similarly, it is also the case for panels in rows 4 and 5. Nevertheless, they were still considered as different panels but our experiments will demonstrate that detecting panels in row 5 (row 4) may also have effect on detection of panels in row 2 (row 3). The 1.56m-spatial resolution of the image scene suggests that most of the 15 panels are one pixel in size except that $p_{21}, p_{31}, p_{41}, p_{51}$ which are two-pixel panels, denoted by p_{211}, p_{221}, p_{311}, p_{312}, p_{411}, p_{412}, p_{511}, p_{521}. Since the size of the panels in the third column is 1m×1m, they cannot be seen visually from Fig. 1(a) due to the fact that its size is less than the 1.56m pixel resolution. Fig. 1(b) shows the precise spatial locations of these 15 panels where red pixels (R pixels) are the panel center pixels and the pixels in yellow (Y pixels) are panel pixels mixed with the background. Fig. 1(c) plots the 5 panel spectral signatures $\mathbf{p}_i$ for j = 1,2,…,5 obtained by averaging R pixels in the 3m×3m and 2m×2m panels in row i in Fig. 1(b).

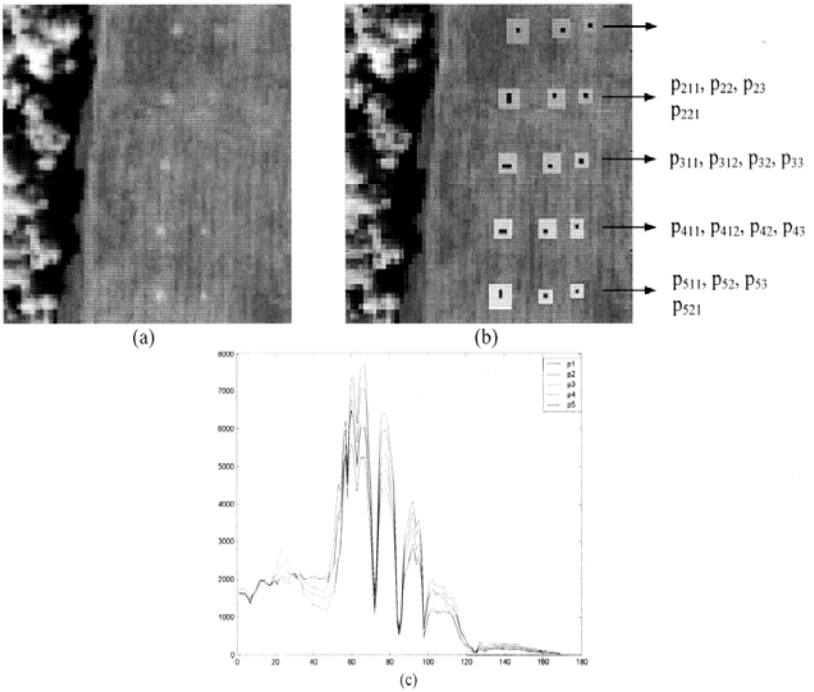


Figure 1. (a) A HYDICE panel scene which contains 15 panels; (b) Ground truth map of spatial locations of the 15 panels; (c) Spectral signatures of $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ and $\mathbf{p}_5$.

It should be noted the R pixels in the 1m×1m panels are not included because they are not pure pixels, mainly due to that fact that the spatial resolution of the R pixels in the 1m×1m panels is 1m smaller than the pixel resolution is 1.56 m. These panel signatures along with the R pixels in the 3m×3m and 2m×2m panels were used as required prior target knowledge for the following comparative studies.

Twenty-five panels were simulated by the five different panel signatures $\{\mathbf{p}_i\}_{i=1}^{5}$ in Fig. 1(c) and are arranged in a matrix with five rows and five columns shown in Fig. 2(a) with their spatial coordinates listed in Fig. 3.

For each row, there are 5 panels with different sizes of 2×2 pixels, 1×2 pixels and 1×1 pixel, but simulated by a major panel signature. For example,

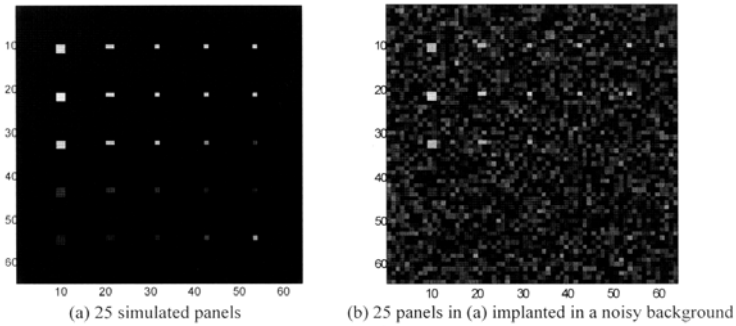


Figure 2. 25 simulated panels implanted in a noisy background.

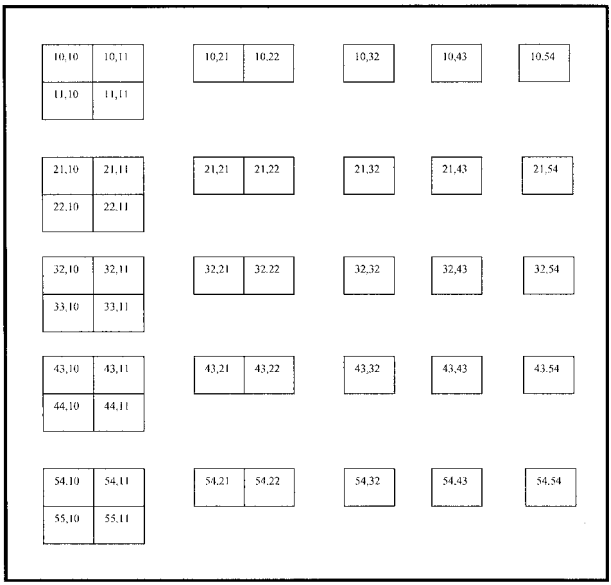


Figure 3. Spatial coordinates of all the panel pixels in the 25 panels

for row *i*, the five panels were simulated by the major signature, $\mathbf{p}_i$ and are arranged by Fig. 4.

For each column, there are five panels with the size, but simulated by the five distinct major panel signatures $\{\mathbf{p}_i\}_{i=1}^{5}$.

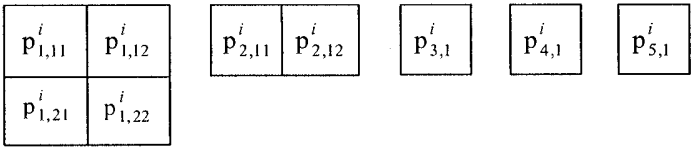


Figure 4. Simulated panels.

Table 1. Compositions of 25 panels simulated by the five panel signatures $\{\mathbf{p}_i\}_{i=1}^{5}$.

Panel pixel	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$	$\mathbf{p}_5$
$\{p^1_{ij}\}^{2,2}_{i=1,j=1}$, $p^1_{2,11}, p^1_{2,12}, p^1_{3,1}$	100%	0%	0%	0%	0%
$p^1_{4,1}$	50%	50%	0%	0%	0%
$p^1_{5,1}$	25%	75%	0%	0%	0%
$\{p^2_{ij}\}^{2,2}_{i=1,j=1}$, $p^2_{2,11}, p^2_{2,12}, p^2_{3,1}$	0%	100%	0%	0%	0%
$p^2_{4,1}$	0%	50%	50%	0%	0%
$p^2_{5,1}$	0%	25%	75%	0%	0%
$\{p^3_{ij}\}^{2,2}_{i=1,j=1}$, $p^3_{2,11}, p^3_{2,12}, p^3_{3,1}$	0%	0%	100%	0%	0%
$p^3_{4,1}$	0%	0%	50%	50%	0%
$p^3_{5,1}$	0%	0%	25%	75%	0%
$\{p^4_{ij}\}^{2,2}_{i=1,j=1}$, $p^4_{2,11}, p^4_{2,12}, p^4_{3,1}$	00%	0%	0%	100%	0%
$p^4_{4,1}$	0%	0%	0%	50%	50%
$p^4_{5,1}$	0%	0%	0%	25%	75%
$\{p^5_{ij}\}^{2,2}_{i=1,j=1}$, $p^5_{2,11}, p^5_{2,12}, p^5_{3,1}$	0%	0%	0%	0%	100%
$p^5_{4,1}$	50%	0%	0%	0%	50%
$p^5_{5,1}$	75%	0%	0%	0%	25%

More specifically, the panels in the 1st column have the same size of 2×2 of pixels denoted by $\{p^i_{1,11}, p^i_{1,12}, p^i_{1,21}, p^i_{1,22}\}_{i=1}^{5}$. The panels in the 2nd column panels have the same size of 1×2 pixels, denoted by $\{p^i_{2,11}, p^i_{2,12}\}_{i=1}^{5}$. The panels in 3rd, 4th and 5th columns are all made up of single pixel, denoted by $\{p^i_{3,1}\}_{i=1}^{5}$,

$\{p_{4,1}^{i}\}_{i=1}^{5}, \{p_{5,1}^{i}\}_{i=1}^{5}$ respectively. The compositions of these 25 panels were simulated according to Table 1. For example, the panels in the 1st, 2nd and 3rd columns are all pure pixels where the panels in row I were simulated by panel signature $\mathbf{p}_i$. The pixels in the 4th and 5th columns are mixed pixels with mixtures of $\{50\%\mathbf{p}_i + 50\%\mathbf{p}_{i+1}\}_{i=1}^{4}$ and $\{25\%\mathbf{p}_i + 75\%\mathbf{p}_{i+1}\}_{i=1}^{5}$ respectively where the *i*-th panel signature $\mathbf{p}_i$ was used to simulate panels in row *i*. As an example, the panel pixels $p_{4,1}^{5}$ and $p_{5,1}^{5}$ in the 5th row are mixed pixels simulated by mixtures of $\{50\%\mathbf{p}_5 + 50\%\mathbf{p}_1\}$ and $\{25\%\mathbf{p}_5 + 75\%\mathbf{p}_1\}$ respectively. These twenty-five panels in Fig. 2(a) were implanted in the noisy background in a way that the corresponding background pixels were replaced by the implanted panel pixels where the image background was simulated by a grass signature extracted in Fig. 1 and corrupted with Gaussian noise of SNR 30:1. The resulting synthetic image is shown in the Fig. 2(b).

4. Synthetic image-based computer simulations

In this section, the synthetic image simulated in Fig. 2(b) was used for computer simulations to study characteristics of each of the four algorithms presented in this chapter, MATLAB-PPI, FIPPI, APPI, ICA-based PPI and further conduct a comparative analysis. Since all these four algorithms require dimensionality reduction, the VD was used to estimate the number of dimensions required to be retained for the synthetic image in Fig. 2(b), which was 6 with the false alarm probability $P_F \leq 10^{-2}$. Based on the VD estimate, 6 dimensions were retained after the MNF transform and shown in Fig. 5.

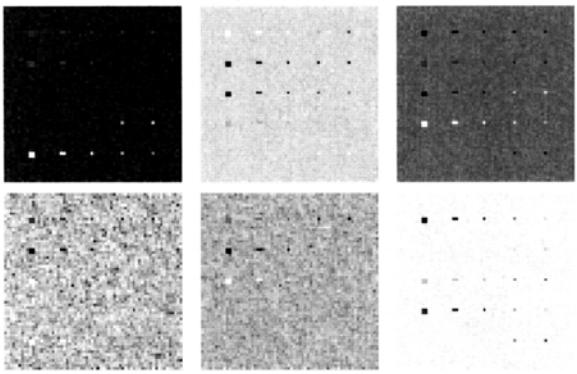


Figure 5. 6 MNF-transformed components.

4.1. MATLAB-PPI

The MATLAB-PPI was performed on the synthetic image in Fig. 2(b) with k = 200 skewers. The resulting PPI count image is shown in Fig. 6 where 98 pixels with PPI count greater than 0 were extracted, of which the 35 were panel pixels and the rest of 63 pixels were the background pixels. It should be noted that the gray values in Fig. 6(b) are used to reflect the values of PPI count where the brighter the intensity, the higher the PPI count.

Table 2 tabulates the PPI counts for the 35 panel pixels where the panel pixels in the same row produced the same PPI count.

It should be noted that panels in each row were simulated by one pure signature with the first 7 panel pixels (4 panel pixels in the 1st column, 2 panel pixels in the 2nd column and 1 panel pixel in the 3rd column) simulated by the same pure signature and the remaining two panel pixels (1 panel pixel in the 4th column and 1 panel pixel in 5th column) being mixed signatures. Therefore,

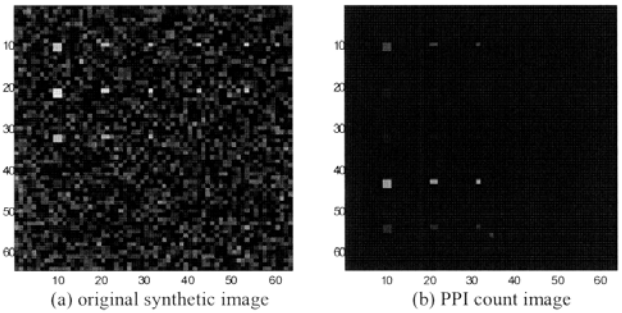


Figure 6. Result of MATLAB-PPI (k = 200 Skewers).

Table 2. Spatial coordinates of the 35 panel pixels along with their corresponding PPI counts produced by the MATLAB-PPI.

panels in row 1	$p^1_{1,11}$	$p^1_{1,12}$	$p^1_{1,21}$	$p^1_{1,22}$	$p^1_{2,11}$	$p^1_{2,12}$	$p^1_{3,1}$
PPI count	74	74	74	74	74	74	74
panels in row 2	$p^2_{1,11}$	$p^2_{1,12}$	$p^2_{1,21}$	$p^2_{1,22}$	$p^2_{2,11}$	$p^2_{2,12}$	$p^2_{3,1}$
PPI count	54	54	54	54	54	54	54
panels in row 3	$p^3_{1,11}$	$p^3_{1,12}$	$p^3_{1,21}$	$p^3_{1,22}$	$p^3_{2,11}$	$p^3_{2,12}$	$p^3_{3,1}$
PPI Count	60	60	60	60	60	60	60
panels in row 4	$p^4_{1,11}$	$p^4_{1,12}$	$p^4_{1,21}$	$p^4_{1,22}$	$p^4_{2,11}$	$p^4_{2,12}$	$p^4_{3,1}$
PPI count	90	90	90	90	90	90	90
panels in row 5	$p^5_{1,11}$	$p^5_{1,12}$	$p^5_{1,21}$	$p^5_{1,22}$	$p^5_{2,11}$	$p^5_{2,12}$	$p^5_{3,1}$
PPI count	90	90	90	90	90	90	90

there are 35 panel pixels representing 5 distinct pure panel signatures $\{\mathbf{p}_i\}_{i=1}^{5}$ among a total of the 45 panel pixels simulated in Fig. 2(a). Since the VD estimated for this image was 6, this implied that the remaining 63 pixels were selected to represent an extra 6th pure signature which is grass signature with their spatial coordinates and associated PPI counts tabulated in Table 3.

Table 3. Spatial coordinates and the corresponding PPI counts of the 63 background pixels produced by the MATLAB-PPI.

pixel location	PPI count	pixel location	PPI count	pixel location	PPI count	pixel location	PPI count	pixel location	PPI count
56,35	21	47,22	3	64,32	2	13,13	1	40,41	1
9,14	11	63,23	3	59,33	2	28,13	1	5,43	1
56,25	9	62,24	3	26,35	2	25,17	1	12,46	1
48,15	5	42,32	3	56,36	2	13,19	1	14,47	1
1,20	5	2,40	3	16,39	2	15,20	1	48,48	1
38,23	5	41,61	3	39,40	2	21,23	1	55,50	1
8,27	5	19,2	2	14,54	2	37,25	1	58,52	1
24,17	4	36,10	2	32,57	2	17,26	1	41,55	1
22,44	4	22,16	2	28,3	1	37,26	1	45,56	1
24,53	4	4,17	2	10,4	1	17,28	1	52,57	1
10,6	3	14,25	2	50,4	1	23,31	1		
4,9	3	52,31	2	31,7	1	56,34	1		
53,15	3	20,32	2	26,8	1	6,41	1		

Comparing Table 3 to Table 2, the PPI counts produced by pure panel pixels were much greater than those produced by background pixels. Additionally, no panel pixels with mixed signatures in the 4th and the 5th columns were extracted. Interestingly, some background pixels were extracted more than once due to the presence of noise and those background pixels which were extracted more than others which appeared to have purest signatures.

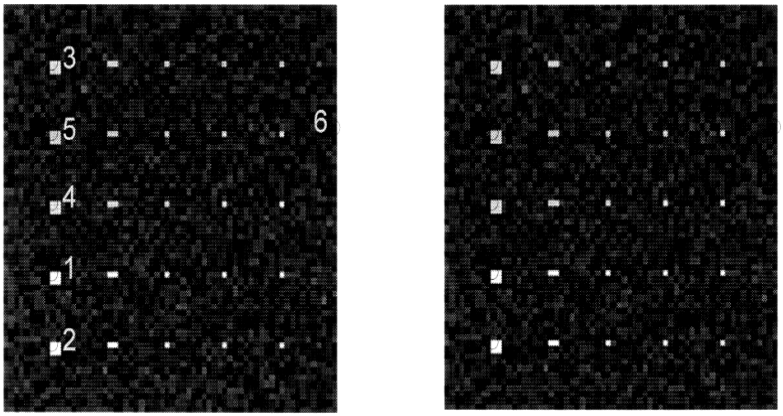
4.2. Fast Iterative PPI (FIPPI)

In order to implement the FIPPI, we first used the VD estimate to determine the number of target pixels required to be generated by the ATGP as initial endmembers. Fig. 7(b) shows the 6 ATGP-generated pixels marked by red circles, of which five pixels numbered by 3, 5, 4, 1, and 2 happened to be also panel pixels, $\mathrm{p}_{1,11}^{1}$, $\mathrm{p}_{1,11}^{2}$, $\mathrm{p}_{1,11}^{3}$, $\mathrm{p}_{1,11}^{4}$, and $\mathrm{p}_{1,11}^{5}$.

The FIPPI was terminated after 3 runs where a total of 9 pixels were extracted and selected as final desired endmembers marked by red circles in

Fig 7(b). As shown in Fig. 7(b), 5 out of 9 selected endmembers were pure panel pixels which are $p^1_{1,11}$, $p^2_{1,11}$ $p^3_{1,11}$, $p^4_{1,11}$ and $p^5_{1,11}$ representing five distinct pure panel signatures, $\{\mathbf{p}_i\}_{i=1}^5$ and the remaining 4 endmembers were background pixels.

A comment is noteworthy. Unlike the other three algorithms, MATLAB-PPI, APPI and ICA-based PPI, the FIPPI does not need random skewers where the number of skewers was set to k = 200. Instead, the FIPPI only needs the same number of the dimensions, p required to be retained after dimensionality reduction and this p was determined by the VD.



(a) ATGP-generated initial endmembers (b) Final selected endmembers

Figure 7. Result of FIPPI.

4.3. Automatic PPI (APPI)

Since the APPI is a random algorithm, it was implemented repeatedly with different set of randomly generated skewers with k = 200. The APPI was terminated after 10 runs. Like Fig. 6, the PPI count results produced by the APPI are shown in Fig. 8(b) where the brightness of the intensity in the PPI count image represents the values of the PPI counts. There were a total of 56 pixels were extracted, of which the 35 were pure panel pixels to represent five distinct pure panel signatures $\{\mathbf{p}_i\}_{i=1}^5$ and the rest of 21 were background pixels.

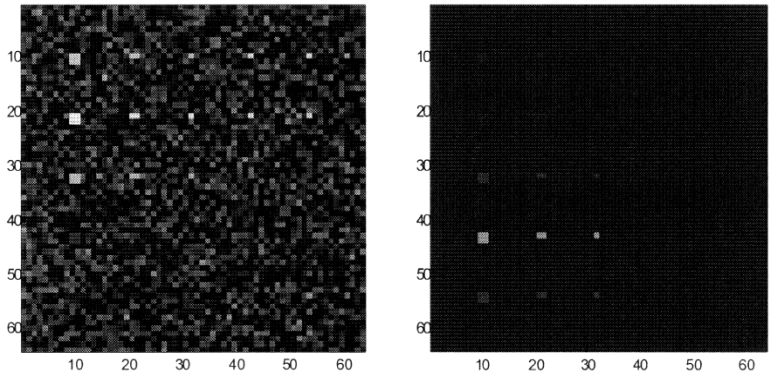


Figure 8. Result of APPI (k = 200 skewers).

Table 4. Spatial coordinates of the 35 pure panel pixels with their corresponding PPI counts produced by the APPI.

panels in row 1	$p^1_{1,11}$	$p^1_{1,12}$	$p^1_{1,21}$	$p^1_{1,22}$	$p^1_{2,11}$	$p^1_{2,12}$	$p^1_{3,1}$
PPI count	373	373	373	373	373	373	373
panels in row 2	$p^2_{1,11}$	$p^2_{1,12}$	$p^2_{1,21}$	$p^2_{1,22}$	$p^2_{2,11}$	$p^2_{2,12}$	$p^2_{3,1}$
PPI count	237	237	237	237	237	237	237
panels in row 3	$p^3_{1,11}$	$p^3_{1,12}$	$p^3_{1,21}$	$p^3_{1,22}$	$p^3_{2,11}$	$p^3_{2,12}$	$p^3_{3,1}$
PPI Count	319	319	319	319	319	319	319
panels in row 4	$p^4_{1,11}$	$p^4_{1,12}$	$p^4_{1,21}$	$p^4_{1,22}$	$p^4_{2,11}$	$p^4_{2,12}$	$p^4_{3,1}$
PPI count	425	425	425	425	425	425	425
panels in row 5	$p^5_{1,11}$	$p^5_{1,12}$	$p^5_{1,21}$	$p^5_{1,22}$	$p^5_{2,11}$	$p^5_{2,12}$	$p^5_{3,1}$
PPI count	435	435	435	435	435	435	435

Table 4 tabulates the accumulated PPI counts of the 35 pure panel pixels resulting from the 10 runs and Table 5 also tabulates the spatial coordinates and the PPI counts of the remaining 21 APPI-extracted background pixels.

Comparing Table 5 to Table 3, the number of background pixels extracted by the APPI was fewer than those found by the MATLAB-PPI because those pixels which were not extracted by the MATLAB-PPI consistently in every

Table 5. Spatial coordinates and the PPI counts of the 21 background pixels extracted by the APPI.

pixel location	PPI count	pixel location	PPI count	pixel location	PPI count	pixel location	PPI count
56,35	127	8,27	44	42,32	29	22,16	20
9,14	79	48,15	43	41,55	27	24,53	20
56,25	71	17,28	39	63,23	25	13,19	15
24,17	52	62,24	38	39,40	24		
1,20	49	2,40	35	31,7	22		
38,23	47	41,61	32	52,31	22		

run were neither extracted by the APPI. It should be noted that the APPI-extracted background pixels were actually a subset of the background pixels extracted by the MATLAB-PPI.

4.4. PPI-based algorithms using ica to perform dimensionality reduction

In this section, we repeated the same computer simulations conducted in Sections 4.1-4.3 except that the MNF used in Section 4.1-4.3 to perform dimensionality reduction was replaced by the ICA-DR algorithm described in Section 2.4.2. In analogy with the MNF, 6 ICA-transformed components were retained and shown in Fig. 9.

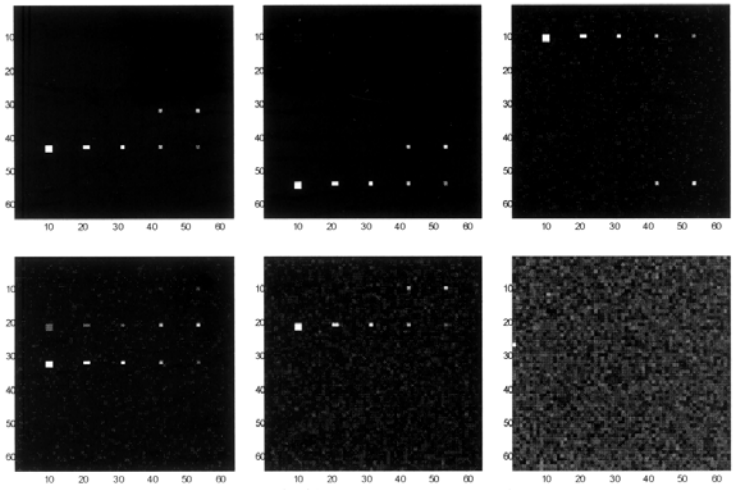


Figure 9. 6 ICs produced by ICA-DR.

Using these 6 ICs in Fig. 9 as dimensionality-reduced image data cube, the MATLAB-PPI extracted a total of 45 pixels in Fig. 10(b), of which 35 are pure panel pixels to represent five pure panel signature $\{\mathbf{p}_i\}_{i=1}^{5}$. Table 6 tabulates the PPI counts of the 35 panel pixels where the pure panel pixels in the same row produced the same PPI count. Additionally, similar to the result in Fig. 6, all the 10 mixed panel pixels in the 4th and 5th columns were not extracted. Instead, the remaining 10 extracted pixels were background pixels.

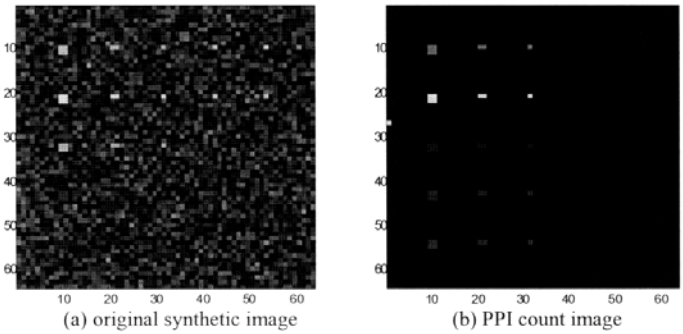


Figure 10. Result of MATLAB-PPI (k = 200 Skewers).

Table 6. Spatial coordinates of the 35 panel pixels along with their corresponding PPI counts produced by the MATLAB-PPI.

panels in row 1	$p^1_{1,11}$	$p^1_{1,12}$	$p^1_{1,21}$	$p^1_{1,22}$	$p^1_{2,11}$	$p^1_{2,12}$	$p^1_{3,1}$
PPI count	64	64	64	64	64	64	64
panels in row 2	$p^2_{1,11}$	$p^2_{1,12}$	$p^2_{1,21}$	$p^2_{1,22}$	$p^2_{2,11}$	$p^2_{2,12}$	$p^2_{3,1}$
PPI count	50	50	50	50	50	50	50
panels in row 3	$p^3_{1,11}$	$p^3_{1,12}$	$p^3_{1,21}$	$p^3_{1,22}$	$p^3_{2,11}$	$p^3_{2,12}$	$p^3_{3,1}$
PPI Count	79	79	79	79	79	79	79
panels in row 4	$p^4_{1,11}$	$p^4_{1,12}$	$p^4_{1,21}$	$p^4_{1,22}$	$p^4_{2,11}$	$p^4_{2,12}$	$p^4_{3,1}$
PPI count	73	73	73	73	73	73	73
panels in row 5	$p^5_{1,11}$	$p^5_{1,12}$	$p^5_{1,21}$	$p^5_{1,22}$	$p^5_{2,11}$	$p^5_{2,12}$	$p^5_{3,1}$
PPI count	71	71	71	71	71	71	71

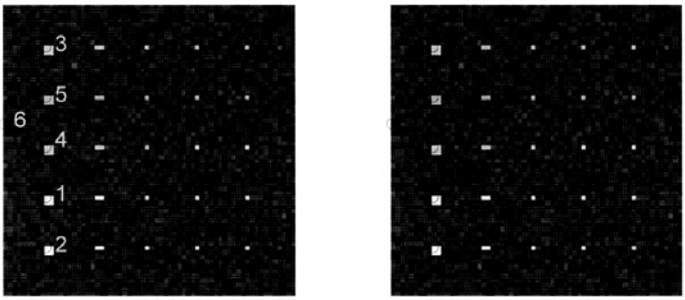
Table 7 also tabulates their spatial coordinates and corresponding PPI counts where all the PPI counts for these 10 background pixels happened to be all 1's.

Next, we implemented the FIPPI by first applying the ATGP to the ICA-DR image components in Fig. 9 to generate 6 target pixels marked by red circles and shown in Fig. 11(b). These 6 ATGP-generated pixels were then used as initial endmembers where four out of the 6 ATGP-generated pixels numbered by were also panel pixels, $p^1_{1,11}$, $p^2_{1,11}$, $p^3_{1,11}$, $p^4_{1,11}$, and $p^5_{1,11}$. The FIPPI was terminated after 3 runs where a total of 13 pixels were extracted and selected as final desired endmembers marked by red circles in Fig 11(b). As shown in Fig. 11(b), 5 out of 13 selected endmembers, $p^1_{1,11}$, $p^2_{1,11}$, $p^3_{1,11}$, $p^4_{1,11}$, and $p^5_{1,11}$ were pure panel pixels representing five distinct pure panel signatures, $\{\mathbf{p}_i\}_{i=1}^5$ and the remaining 8 endmembers were background pixels.

Finally, the APPI was also implemented using the 6 ICs in Fig. 9 as a single image cube. It was terminated after 5 runs with 37 pixels extracted, of

Table 7. Background PPI counts produced by the PPI algorithm.

pixel location	PPI count	pixel location	PPI count
27,1	44	15,20	1
27,50	5	63,23	1
52,33	4	16,48	1
7,1	3	14,54	1
54,19	2	46,63	1



(a) ATGP-generated initial endmembers (b) Final selected endmembers

Figure 11. Result of FIPPI.

which 35 were the pure panels pixels to represent five distinct pure panel signatures, $\{\mathbf{p}_i\}_{i=1}^{5}$. The results are shown in Fig. 12 and Table 8 where the PPI counts of the 35 pure panel pixels plus spatial coordinates of two background pixels along with their corresponding PPI counts are tabulated in Table 8.

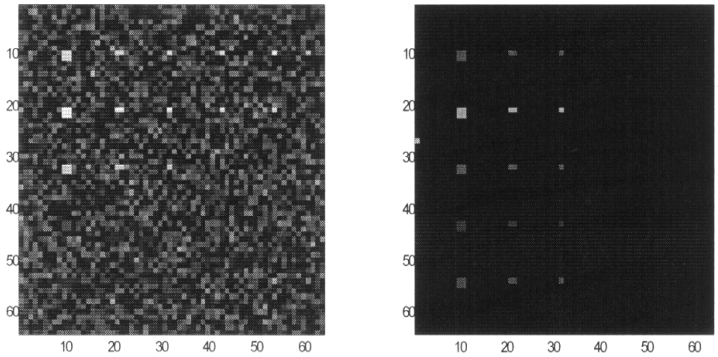


Figure 12. Result of APPI (200 Skewers)

Table 8. Spatial coordinates of the 37 extracted pixels along with their corresponding PPI counts produced by the APPI.

panels in row 1	$p^1_{1,11}$	$p^1_{1,12}$	$p^1_{1,21}$	$p^1_{1,22}$	$p^1_{2,11}$	$p^1_{2,12}$	$p^1_{3,1}$
PPI count	332	332	332	332	332	332	332
panels in row 2	$p^2_{1,11}$	$p^2_{1,12}$	$p^2_{1,21}$	$p^2_{1,22}$	$p^2_{2,11}$	$p^2_{2,12}$	$p^2_{3,1}$
PPI count	288	288	288	288	288	288	288
panels in row 3	$p^3_{1,11}$	$p^3_{1,12}$	$p^3_{1,21}$	$p^3_{1,22}$	$p^3_{2,11}$	$p^3_{2,12}$	$p^3_{3,1}$
PPI Count	315	315	315	315	315	315	315
panels in row 4	$p^4_{1,11}$	$p^4_{1,12}$	$p^4_{1,21}$	$p^4_{1,22}$	$p^4_{2,11}$	$p^4_{2,12}$	$p^4_{3,1}$
PPI count	368	368	368	368	368	368	368
panels in row 5	$p^5_{1,11}$	$p^5_{1,12}$	$p^5_{1,21}$	$p^5_{1,22}$	$p^5_{2,11}$	$p^5_{2,12}$	$p^5_{3,1}$
PPI count	335	335	335	335	335	335	335

background pixels	pixel location	PPI count
	27,1	282
	27,50	28

Comparing Table 8 to Table 7, the number of background pixels extracted by the APPI was fewer than MATLAB-PPI because those pixels which were not extracted consistently in every run by MATLAB-PPI were not extracted by the APPI. It should be noted that the APPI-extracted background pixels were actually a subset of the background pixels extracted by the MATLAB-PPI.

It is interesting to note that, according to our experiments, if the number of skewers, k used in the above experiments was reduced from 200 to 64, it also produced the same results. This suggested that when the ICA is used for dimensionality reduction, the number of skewers required is generally much fewer than that required for second order statistics-based transform, such as PCA or MNF transform.

5. Real HYDICE data

The real HYDICE image shown in Fig. 1(a) was used for experiments to evaluate the performance of the MATLAB-PPI, FIPPI and APPI with the MNF and ICA-DR for dimensionality reduction. The VD estimated for this image scene and empirically selected to be 13 with $P_F = 10^{-4}$. In this case, only 13 dimensions were retained after the ICA-DR algorithm and MNF transform.

5.1. MNF-dimensionality reduction

First of all, the MNF was performed to reduce dimensionality from 169 bands to 13 dimensions. Fig. 13 shows the first 13 components produced by the MNF transform.

Since the real image scene in Fig. 1(a) is generally more complicated than the synthetic image in Fig. 2(b), it usually requires much greater number of random skewers than that needed for the synthetic image to ensure that there are sufficient skewers to cover all random directions and secure the endmembers of interest in the scene. In this case, k was empirically selected to $k = 1000$ five times $k = 200$ skewers used for computer simulations in Section 4. According to the experiments conduced for k =1000, 2000 and 10000 in [6], k =1000 was a reasonable selection because results produced by these three cases were very close and did not make significant difference. For details, we refer to [6].

For $k = 1000$ skewers, 103 pixels were extracted by the MATLAB-PPI and are shown in Fig. 14 where its PPI count image is also provided in Fig 14(a) for visual assessment. Among the extracted 103 pixels, 6 were panel pixels, p_{11}, p_{311}, p_{312}, p_{411}, p_{412}, p_{521} marked by white circles. Similarly, the APPI was applied to the 13 MNF components in Fig. 13. It was terminated

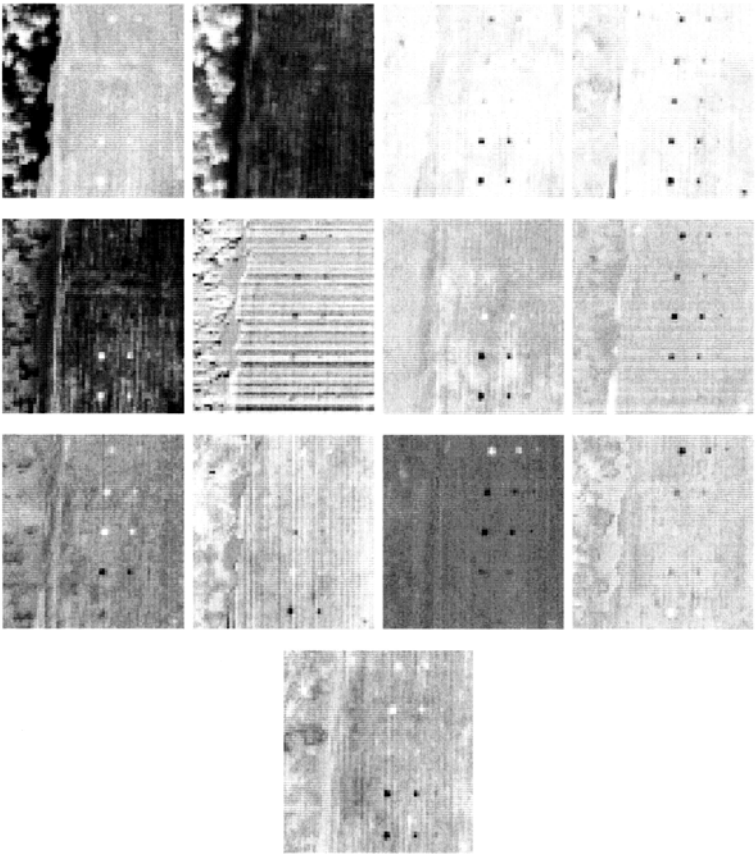
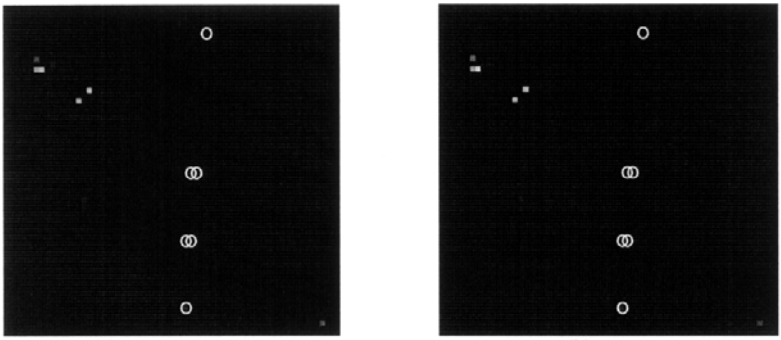


Figure 13. First 13 components produced by the MNF transform.

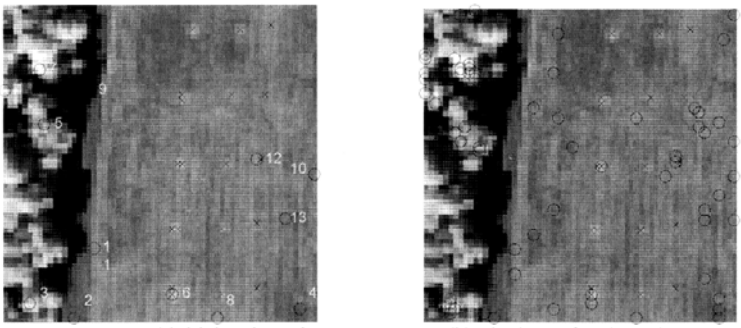
after 7 runs and a total of 60 were extracted. Interestingly, both the MATLAB-PPI and APPI extracted identical panel pixels shown in Fig. 14(b) which represented five distinct panel signatures, $\{\mathbf{p}_i\}_{i=1}^{5}$.

In order to implement the FIPPI, the ATGP was first applied to 13 target pixels to be used as initial endmembers as shown in Fig. 15(a) where the 13 ATGP-generated pixels labeled by the order they were generated and marked by red circles, of which one pixels numbered by 6 happened to be also a panel pixel. The FIPPI was terminated after 9 runs where a total of 68 pixels were extracted



(a) MATLAB-PPI (b) APPI

Figure 14. PPI count images produced by MATLAB-PPI and APPI with k = 1000 skewers.



(a) ATGP-generated initial endmembers (b) Final set of endmembers

Figure 15. Results produced by the FIPPI.

and selected as final desired endmembers marked by red circles in Fig 15(b). As shown in Fig. 15(b), 2 out of 68 selected endmembers were pure panel pixels which are p_{312} and p_{521} representing two distinct panel signatures $\mathbf{p}_3$ and $\mathbf{p}_5$, and the remaining 66 endmembers were background pixels.

From the experiments conducted in this subsection, it is evident that the parameter *k*, the number of skewers played a crucial role in endmember extraction and has tremendous impact on PPI counts. The larger the *k*, the number of skewers, the better the performance. However, it should be noted that the dimensionality reduction is also another parameter which is crucial in

determination of the *k*. In our experiments, the number of dimensions to be retained by the dimensionality reduction is determined by the VD. So, this issue can be resolved by the VD.

5.2. ICA- dimensionality reduction

Following the same argument given in Section 4.4, the same experiments were also conducted for the MATLAB-PPI and APPI with the dimensionality reduction performed by the ICA instead of MNF transform as done in Section 4.4 shows the first 13 ICs shown in Fig. 16 generated by the ICA-based DR.

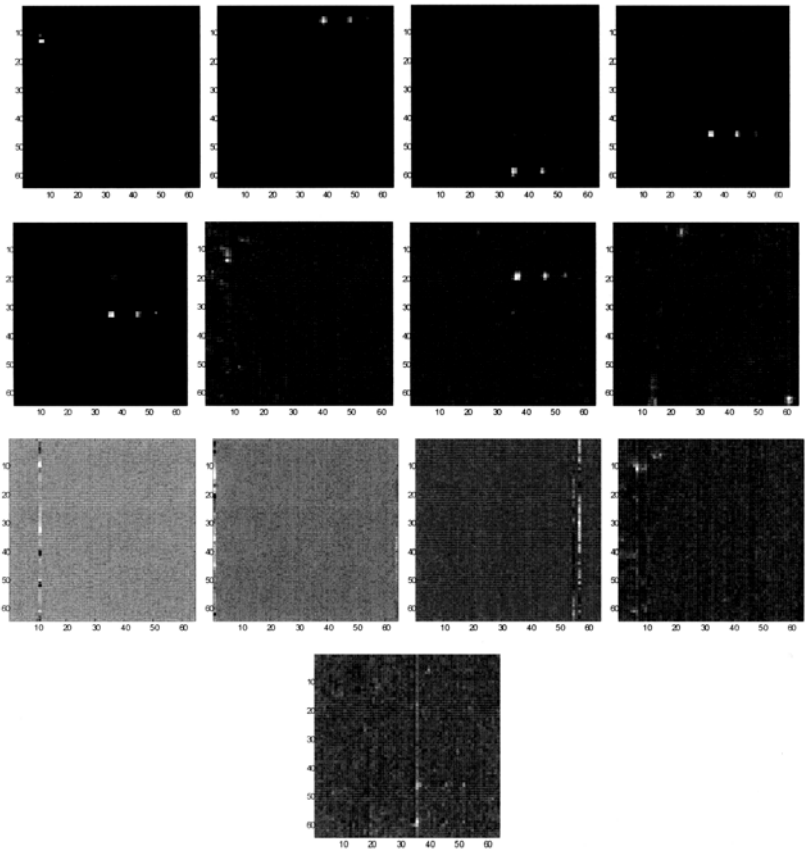
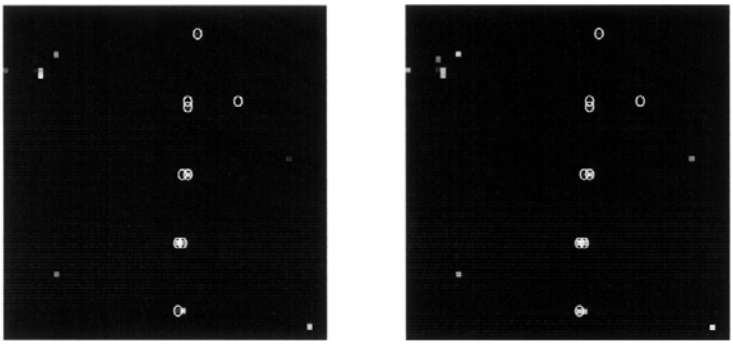


Figure 16. First 13 ICs produced by the ICA-DR algorithm.

Since it was shown in the synthetic image-based computer simulations (see Section 4) that the ICA generally required much fewer skewers than that needed for the MNF transform, in the following ICA-based experiments, the number of skewers, k was empirically chosen to be 200 and the VD was set to 13 which the same number for the MNF transform.

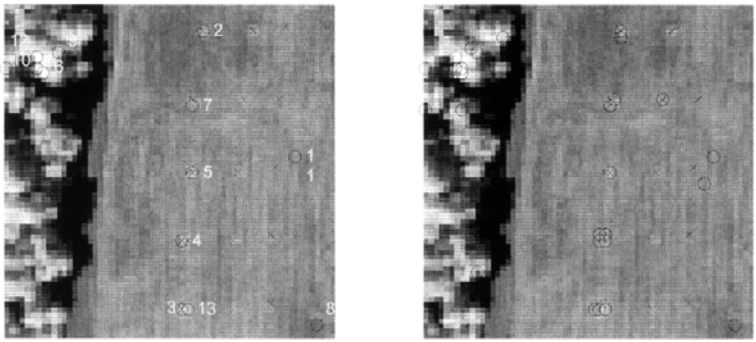
Using the 13 ICs obtained in Fig. 16 and k = 200 skewers, the MATLAB-PPI extracted 23 pixels among which 9 were panel pixels as shown by white open circles in Fig. 17(a) for visual assessment. The APPI was also performed with k = 200 skewers on the 13 ICs in Fig. 16 and terminated after 3 runs. A total of 20 pixels were extracted where 9 out of the extracted 20 pixels were also panel pixels as shown by white open circles in Fig. 17(b) for visual assessment. In analogy with Fig. 14, both PPI and APPI also extracted the same set of 9 panel pixels, p_{11}, p_{211}, p_{221}, p_{22}, p_{311}, p_{312}, p_{411}, p_{412}, and p_{511} which represented five distinct panel signatures, $\{\mathbf{p}_i\}_{i=1}^{5}$.

Finally, the FIPPI was applied to the 13 ICs in Fig. 16. Fig. 18(a) shows the 13 pixels found by the ATGP along with their appearance order labeled by numbers where the five pixels labeled by 2, 7, 5, 4, 3 and marked by red open circle were the panel pixels, p_{11}, p_{221}, p_{312}, p_{411} and p_{521}. The FIPPI was terminated after 4 runs where a total of 28 pixels were extracted and selected as final desired endmembers marked by red circles in Fig 18(b). As shown in Fig. 18(b), 7 out of 28 selected endmembers, p_{11}, p_{221}, p_{22}, p_{312}, p_{411}, p_{412}, p_{521} were pure panel pixels to represent five distinct pure panel signatures, $\{\mathbf{p}_i\}_{i=1}^{5}$ and the remaining 21 endmembers were background pixels.



(a) MATLAB-PPI (b) APPI

Figure 17. PPI count images produced by MATLAB-PPI and APPI with k = 200 skewers.



(a) ATGP-generated initial endmembers (b) Final set of endmembers

Figure 18. Results produced by the FIPPI.

Comparing the results presented in Section 5.1 and Section 5,2, it is clearly demonstrated that the efficiency and effectiveness of MATLAB-PPI and APPI was greatly increased by replacing the MNF transform with the ICA in dimensionality reduction.

6. Conclusions

Pixel Purity Index (PPI) is a well-known and popular approach to endmember extraction. It has been widely used in ENVI software. With very limited information published in the open literature, the details of PPI implementation are not available. This chapter investigates the PPI and re-derives a general-purpose language (e.g., MATLAB) version of the PPI, called MATLAB-PPI that can be very easily implemented by any user without appealing for the ENVI software. Since the PPI uses randomly generated skewers as initial candidate endmembers, it suffers from inconsistent results. In order to cope with this problem, two methods are developed. One is called Fast Iterative PPI (FIPPI) which implements an initialization algorithm to produce an appropriate set of initial endmembers used for the MATLAB PPI so that the final selected set endmembers is consistent. Another is called Automatic PPI (APPI) which implement the MATLAB PPI in different runs so that the final selected set of desired endmembers is their common set. Experiments through synthetic image-based computer simulations and real image data demonstrate that both FIPPI and APPI improve the MATLAB PPI in computational complexity and produce consistent final set of endmembers. Since the PPI-based algorithms require dimensionality reduction as its pre-processing, the issues of how many dimensions need to be retained and the MNF transform used in PPI for dimensionality reduction are also thoroughly investigated. As for the number of dimensions required to be retained by dimensionality

reduction, a recently developed concept, called Virtual Dimensionality (VD) is used to estimate this number. It is found by experiments that the VD provides a reasonable estimate. With regard to the second issue, experiments in this chapter also demonstrate that if the PPI-based algorithms are implemented by replacing the MNF transform with the ICA for dimensionality reduction, the performance of ICA-based PPI algorithms can be significantly improved over the MNF-based PPI algorithms. Table 9 and Table 10 summarize the comparative performance of the PPI, FIPPI and the APPI using synthetic image-based simulations and the real HYDICE image in terms of the number of extracted pure panels and background pixels.

Table 9. Results based on MNF-transformed image components.

MNF-DR	pure panel pixels		background pixels	
images	synthetic	HYDICE	synthetic	HYDICE
PPI	35 (k=200)	27 (k=1000)	63 (k=1000)	8 (k=1000)
FIPPI	5	2	4	66
APPI	35	23	21	6

From the above tables it is clear that, the APPI performed better than the PPI on MNF-transformed images since the APPI reduces the number of extracted background pixels which are actually falsely alarmed endmembers. Also the APPI operating on ICA-transformed images yielded the best performance in terms of number of extracted panel pixels as endmembers with only few background pixels extracted as falsely alarmed endmembers. Finally, it should be noted that the ATGP-generated target pixels used for the FIPPI were obtained from dimensionality-reduced image components. However, as a matter of fact, the ATGP can be applied to operate full bands without DR. In this case, we may expect that using full bands to produce ATGP-generated target pixels as initial endmembers for the FIPPI may improve its performance. Table 11 tabulates the results of the MNF-based FIPPI and ICA-based FIPPI with initial endmembers produced by ATGP operating full bands.

Table 10. Results based on image components obtained by ICA-DR algorithm.

ICA-DR	pure panel pixels		background pixels	
images	synthetic	HYDICE	synthetic	HYDICE
PPI	35 (k=200)	9 (k=200)	10 (k=200)	13 (k=200)
FIPPI	5	7	8	21
APPI	35	9	2	11

Table 11. Results based on image components obtained by FIPPI with ATGP using full bands.

MNF images	pure panel pixels		background pixels	
	synthetic image	HYDICE	synthetic image	HYDICE
	k=200	k=200	k=200	k=200
FIPPI	5	4	7	63

ICA images	pure panel pixels		background pixels	
	synthetic image	HYDICE	synthetic image	HYDICE
	k=200	k=200	k=200	k=200
FIPPI	4	7	9	23

Comparing Table 11 to Table 9 and Table 10, it is interesting to note that using the ATGP-generated initial endmembers can only improve MNF-based FIPPI, but not necessarily true for the ICA-based FIPPI.

References

1. R.A. Schwengerdt, "*Remote Sensing: Models and Methods for Image Processing*," 2nd. Ed., Academic Press, 1997, p. 447.
2. J.W. Boardman, "Geometric mixture analysis of imaging spectrometery data," *Proc. Int. Geoscience and Remote Sensing Symp.*, vol. 4, pp. 2369-2371, 1994.
3. C.-I Chang and A. Plaza, "Fast iterative algorithm for implementation of pixel purity index," *IEEE Geoscience and Remote Sensing Letters*. (to appear)
4. C.-I Chang, *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*, Kluwer Academic/Plenum Publishers, 2003.
5. C.-I Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 42, no. 3, pp. 608-619, March 2004.
6. F. Chaudhry, *Pixel Purity Index-Based Endmember Extraction for Hyperspectral Data Exploitation*, MS Thesis, Department of Computer Science and Electrical Engineering, August 2005.
7. J.A. Richard and X. Jia, *Remote Sensing Digital Image Analysis*, 3rd ed., New York, Spring Verlag, 1999.
8. A. A Green, Mark Berman, Paul Switzer and Maurice D. Craig, "A transformation for ordering multispectral data in term of image quality with implication to noise removel", *IEEE Trans. on Geoscience and Remote Sensing*, vol. 26, no. 1, pp. 65-74, January 1988.
9. J.B. Lee, A.S. Woodyatt and M. Berman, "Enhancement of high spectral resolution remote sensing data by a noise-adjusted principal components transform," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 28, no. 3, pp. 295-304, May 1990.
10. A. Hyvarinen, J. Karhunen and E. Oja, *Independent Component Analysis*, John Wiley & Sons, 2001.

11. H. Ren and C.-I Chang, “Automatic spectral target recognition in hyperspectral imagery,” IEEE Trans on Aerospace and Electronic System, vol.39, no.4, pp. 1232-1249, October 2003.
12. J. Wang and C.-I Chang, “Dimensionality reduction by independent component analysis for hyperspectral image analysis,”*IEEE International Geoscience and Remote Sensing Symposium*, Seoul, Korea, July 25-29, 2005.
13. A. Hyvarinen and E. Oja, “A fast fixed-point for independent component analysis,” *Neural Computation*, vol. 9, no. 7, pp. 1483-1492, 1997.
14. J.C. Harsanyi and C.-I Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 32, no. 4, pp. 779-785, July, 1994.