J Real-Time Image Proc (2012) 7:105–129
DOI 10.1007/s11554-010-0151-z

ORIGINAL RESEARCH PAPER

Real-time N-finder processing algorithms for hyperspectral imagery

Chao-Cheng Wu · Hsian-Min Chen · Chein-I Chang

Received: 17 February 2009 / Accepted: 5 January 2010 / Published online: 16 February 2010

Abstract N-finder algorithm (N-FINDR) is probably one of most popular and widely used algorithms for endmember extraction in hyperspectral imagery. When it comes to practical implementation, four major obstacles need to be overcome. One is the number of endmembers which must be known a priori. A second one is the use of random initial endmembers to initialize N-FINDR, which generally results in different sets of final extracted endmembers. Consequently, the results are inconsistent and not reproducible. A third one is requirement of dimensionality reduction (DR) where different used DR techniques produce different results. Finally yet importantly, it is the very expensive computational cost caused by an exhaustive search for endmembers all together simultaneously. This paper re-designs N-FINDR in a real time processing fashion to cope with these issues. Four versions of Real Time (RT) N-FINDR are developed, RT Iterative N-FINDR (RT IN-FINDR), RT SeQuential N-FINDR (RT SQ N-FINDR), RT Circular N-FINDR, RT SuCcessive N-FINDR (RT SC N-FINDR), each of which has its own merit for implementation. Experimental results demonstrate that real time processing algorithms perform as well as their counterparts with no real-time processing.

Keywords N-FINDR · Real-time circular N-FINDR (RT Circular N-FINDR) · RT iterative N-FINDR (RT IN-FINDR) · Real-time SeQuential N-FINDR (RT SQ N-FINDR) · Real-time SuCcessive N-FINDR (RT SC N-FINDR) · Virtual dimensionality (VD)

1 Introduction

N-finder algorithm (N-FINDR) [1] and pixel purity index (PPI) [2] are probably the two most popular and widely used endmember extraction algorithms in the literature. While both deserve their novelty in algorithm design, they also unfortunately suffer from drawbacks and disadvantages in practical implementation. Since the issues of implementing PPI in practical applications have been well documents in [3–6], this paper mainly focuses on N-FINDR and addresses several issues in practical implementation of N-FINDR. The first is determination of the number of endmembers, denoted by p, which must be known a priori. Unfortunately, the prior knowledge of p is never known in real applications. Over the past years, finding an appropriate value of the p is generally carried out on a trial-and-error basis empirically. The problem with this approach is that when the value of p varies, the entire process of N-FINDR must be re-initiated and previous results cannot be used for updates. Therefore, it is highly

C.-C. Wu · C.-I. Chang (✉)
Remote Sensing Signal and Image Processing Laboratory, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250, USA
e-mail: cchang@umbc.edu

C.-C. Wu
e-mail: chaowu1@umbc.edu

H.-M. Chen
Department of Radiology, China Medical University Hospital, Taichung, Taiwan, ROC
e-mail: hsmin6511@gmail.com

H.-M. Chen
Department of Biomedical Engineering, HungKuang University, Taichung, Taiwan, ROC

C.-I. Chang
Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan, ROC

desirable as well as very crucial to have the value of p estimated reliably to avoid repeatedly implementing N-FINDR over and over again. Recently, this issue has been addressed by a new concept, called virtual dimensionality (VD) developed in [7, 8] which has shown success in estimating the p reliably [5, 9]. A second issue encountered in implementation of the N-FINDR is the use of random initial endmembers which produce inconsistent results in finding endmembers. In other words, different sets of random initial endmembers may result in different sets of final extracted endmembers by the N-FINDR. Consequently, the N-FINDR results are not reproducible. Fortunately, this dilemma can be resolved by specifying an appropriate set of initial endmembers obtained by a custom-designed initialization algorithm [3–5]. A third issue in implementing the N-FINDR is its very high and expensive computation complexity. This is primarily caused by an exhaustive search for an optimal set of p endmembers simultaneously among all possible p-endmember combinations in the data. When the data sample pool is huge, which is indeed the case of hyperspectral imagery, the computational cost can become un-manageable and forbidden. This situation will become much worse if the N-FINDR must be performed empirically by trying various values of p. Since the first issue has been well studied in [3–9], only the last two issues will be addressed in this paper.

As originally designed, the N-FINDR was developed to find a set of p vertices that forms a p-vertex simplex to yield the maximum volume among all possible p-vertex simplexes. The p vertices of the found simplex with the maximum volume are assumed the desired endmembers. Since these p vertices must be found simultaneously, the N-FINDR is referred to as SiMultaneous N-FINDR (SM N-FINDR) thereafter in this paper. Two obstacles results from implementing the SM N-FINDR. In order to find an optimal set of p endmembers, an exhaustive search must be conducted for all possible p-vertex simplexes. The total number of p-combinations, i.e., p-vertex simplexes needed to be compared will be $\binom{N}{p} = \frac{N!}{(n-p)!p!}$ provided that the total number of data sample vectors is N. Second, since two different p-combinations are generally uncorrelated, one cannot take advantage of another. If the value of p changes, the entire exhaustive search must be re-deployed again. Two approaches are developed in this paper to address this issue. One is to reduce the search region for the SM N-FINDR to a feasible region in which potential endmember candidates are assumed to be present. In doing so, the exhaustive search performed by SM N-FINDR is broken up into two iterative processes specified by two procedures, outer and inner loops where the inner loop searches for local optimal endmember candidates, called suboptimal endmembers, while the outer loop updates the suboptimal endmembers to global optimal endmembers. The resulting N-FINDR is referred to as Iterative N-FINDR (IN-FINDR). The other approach is to perform the SM N-FINDR sequentially so that the desired endmembers can be generated one after another in a successive manner instead of generating p endmembers all together simultaneously as the SM N-FINDR does. Such a resulting SM N-FINDR is referred to as SeQuential N-FINDR (SQ N-FINDR). There are two advantages of implementing the SQ N-FINDR over the SM N-FINDR. One is that the SQ N-FINDR takes advantage of previously generated endmembers so that when the value of p grows, the endmembers generated for a small value of p are also part of endmembers for a larger value of p. Because of such a sequential process, a second advantage is a significant reduction of computational cost. However, a trade-off for these two advantages is that p endmembers produced by the SQ-N-FINDR may not be an optimal set of p endmembers. Nevertheless, this disadvantage can be remedied by two ways. One is to use SQ N-FINDR to run the inner and out loops implemented in the IN-FINDR. Specifically, the SQ N-FINDR is used to run the inner loop of the IN-FINDR to produce one final set of p endmembers which will be further used as a new set of initial p endmembers for the next run of the SQ N-FINDR in the inner loop again. The entire process will be run by the SQ N-FINDR repeatedly over and over again in the outer loop of the IN-FINDR until the final generated endmembers remain unchanged. The other is to appeal for a well-designed initialization algorithm that produces a good set of initial endmembers to be used for the SQ N-FINDR. In this case, the IN-FINDR only needs to implement the SQ N-FINDR in its inner loop without using the outer loop to deal with the issue caused by random endmembers to be used in its inner loop.

By virtue of IN-FINDR and SQ N-FINDR, this paper develops the concept of real time processing N-FINDR that actually addresses all the above issues together in one-shot operation. In order to implement N-FINDR in real time, the N-FINDR must be carried out in a causal fashion in the sense that only the data sample vectors that were visited before the currently being processed data sample vector can be used for data processing. As a result, the random initial condition cannot be used to initialize the N-FINDR. In addition, the dimensionality reduction (DR) required in the N-FINDR cannot not be implemented since it needs the entire data set to perform DR. To realize causality imposed on implementation of the N-FINDR, four versions of real-time (RT) N-FINDR, RT Iterative N-FINDR (RT IN-FINDR), RT SeQuential N-FINDR (RT SQ N-FINDR), RT Circular N-FINDR, RT SuCcessive N-FINDR (RT SC N-FINDR) are developed, each of which has its own merit in

implementation. Several benefits can result from real time processing. First, the issue in using random initial conditions is resolved. Second, the requirement of DR is also removed. Third, the computational complexity is significantly reduced. Finally, it can be implemented by on-board or on-line processing to save data storage, communication, and transmission which are particularly important in satellite space-borne data processing. In order to evaluate various versions of RT versions of N-FINDR derived in this paper, experiments using two sets of data, synthetic and real images are conducted for performance analysis.

The remainder of this paper is organized as follows. Section 2 develops various real-time processing versions of the N-FINDR for practical implementation. Section 3 discusses design rationales and philosophies for real-time processing of N-FINDR presented in Sect. 2 and their correlations. Sections 4 and 5 conduct synthetic and real image experiments for performance evaluation, respectively. Section 6 summarizes the results and concludes with some remarks.

2 Iterative N-FINDR

In order to develop a real-time processing algorithm for the commonly used N-FINDR, we need to investigate issues that prevent it from real-time implementation. To do so, the N-FINDR is first reviewed and then followed by a re-designed N-FINDR, referred to as iterative N-FINDR (IN-FINDR) that can be viewed as a sequential version of the N-FINDR which will serve as the first step toward design of real-time processing N-FINDR algorithms.

2.1 N-FINDR

Despite that the original N-FINDR was not available in the open literature, its idea developed by Winter in [1] can be briefly described in the following algorithmic implementation based on our interpretation, referred to as SM N-FINDR where all endmembers must be found simultaneously.

2.1.1 SM N-FINDR

1. Preprocessing:
   (a) Let p be the number of endmembers required to generate. This prior knowledge should be provided in advance.
   (b) Apply a DR transform such as MNF to reduce the data dimensionality from L to $p-1$ where L is the total number of spectral bands.
2. Exhaustive search: For any set of p data sample vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p$, we use this set to form a vertex set for a p-vertex simplex specified by $s(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p)$ and define its volume, $V(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p)$ by

$$V(\mathbf{e}_1, \ldots, \mathbf{e}_p) = \frac{\left|\det\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \mathbf{e}_1 & \mathbf{e}_2 & \ldots & \mathbf{e}_p \end{bmatrix}\right|}{(p-1)!}. \quad (1)$$

   Find a set of p data sample vectors in the data set, denoted by $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \ldots, \mathbf{e}_p^*\}$, that are used as p vertices to construct a p-vertex simplex to yield the maximum value of (1), i.e.,

$$\left\{\mathbf{e}_1^*, \mathbf{e}_2^*, \ldots, \mathbf{e}_p^*\right\} = \arg\left\{\max_{\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p\}} V(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_p)\right\}. \quad (2)$$

   The set of $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \ldots, \mathbf{e}_p^*\}$ is the desired set of p endmembers needed to be found.
3. In order to complete the above exhaustive search in step 2 a comparison among $\binom{N}{p} = \frac{N!}{(n-p)!p!}$ p-vertex simplexes must be conducted in accordance with the criterion specified by (2).

Figure 1 depicts a block diagram of implementing N-FINDR where an exhaustive search is conducted via a counter k starting from 1 to the total $= N!/[p!(N-p)!]$ which is the total number of p-vertex simplex needed to be compared.

2.2 Iterative N-FINDR

Apparently, the SM N-FINDR described above cannot be implemented in real time since both the data dimensionality reduction in step 1 and the exhaustive search in steps 2–3 require availability of the complete set of data sample vectors. In addition, an exhaustive search for finding an optimal set of endmembers, $\{\mathbf{e}_1^*, \mathbf{e}_2^*, \ldots, \mathbf{e}_p^*\}$ via (2) requires tremendous computing time to complete this process by exhausting all possible p-vertex simplexes. Such excessive computational complexity is indeed the major hurdle that prevents the SM N-FINDR from practical applications. In order to cope with these drawbacks, we re-invent the wheel by re-designing the SM-NFINDR in a novel fashion that the SM N-FINDR is broken up into two iterative procedures, an inner loop and an outer loop, so that the SM-N-FINDR can be carried out sequentially rather than simultaneously. The resulting N-FINDR is called IN-FINDR (RT IN-FINDR) and can be described as follows.

2.2.1 Iterative N-FINDR (IN-FINDR)

1. Preprocessing:
   (a) Let p be the number of endmembers required to generate.
   (b) Apply a DR transform such as MNF to reduce the data dimensionality from L to p where L is the total number of spectral bands.

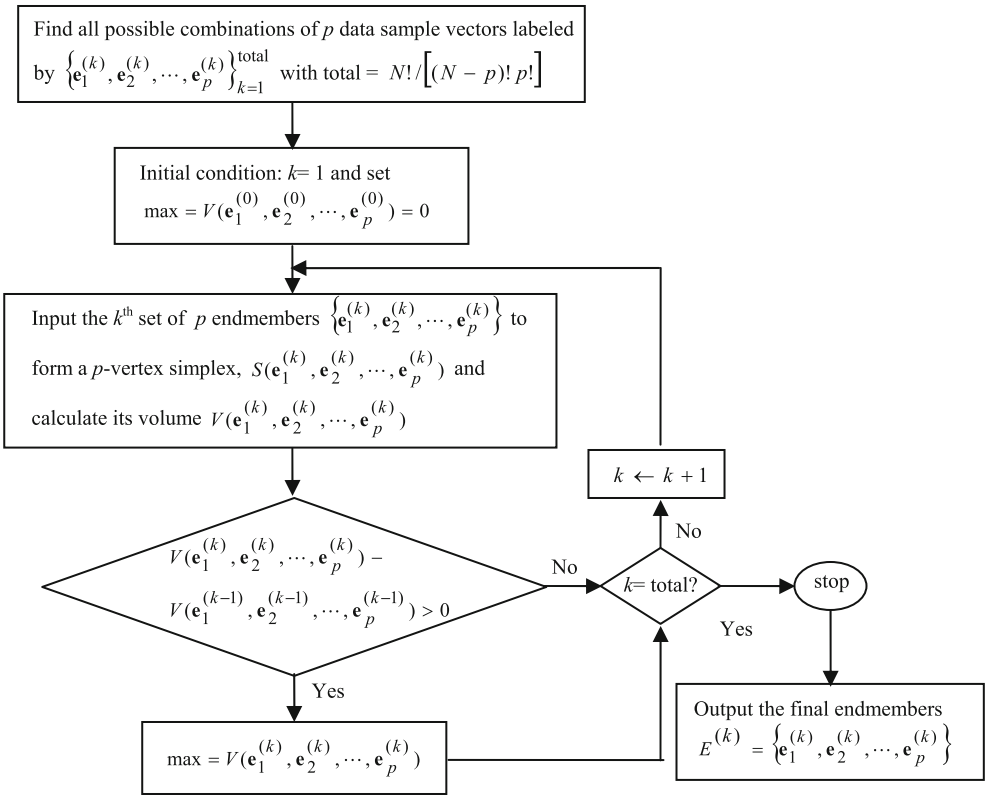


Fig. 1 Block diagram of N-FINDR implementation

2. Initialization: Let $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_p^{(0)}\}$ be a set of initial vectors randomly selected from the data. $k = 0$.
3. Outer loop: At iteration $k \geq 1$, compare the endmembers extracted at kth and $k-1$ iteration. If $\{\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}\} = \{\mathbf{e}_1^{(k-1)}, \mathbf{e}_2^{(k-1)}, \ldots, \mathbf{e}_p^{(k-1)}\}$,, the algorithm is terminated and $\{\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}\}$ is the final selected endmembers. Otherwise, find the volume of the simplex specified by the p vertices $\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}, V(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)})$ defined by (1).
4. Inner loop: For $1 \leq j \leq p$, we recalculate $V(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{j-1}^{(k)}, \mathbf{r}, \mathbf{e}_{j+1}^{(k)}, \ldots, \mathbf{e}_p^{(k)})$ for all data sample vectors $\mathbf{r}$. If any of these p recalculated volumes, $V(\mathbf{r}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}), V(\mathbf{e}_1^{(k)}, \mathbf{r}, \mathbf{e}_3^{(k)}, \ldots, \mathbf{e}_p^{(k)}), \ldots, V(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{p-1}^{(k)}, \mathbf{r})$, is greater than $V(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)})$, go to step 5. Otherwise, let $k \leftarrow k+1$ and go to step 3.
5. Replacement rule: The endmember which is absent in the largest volume among the p simplexes, $S(\mathbf{r}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}), S(\mathbf{e}_1^{(k)}, \mathbf{r}, \mathbf{e}_3^{(k)}, \ldots, \mathbf{e}_p^{(k)}), \ldots S(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_{p-1}^{(k)}, \mathbf{r})$, will be replaced by the sample vector $\mathbf{r}$. Assume that such an endmember is denoted by $\mathbf{e}_j^{(k+1)}$. A new set of endmembers is then produced by letting $\mathbf{e}_j^{(k+1)} = \mathbf{r}$ and $\mathbf{e}_i^{(k+1)} = \mathbf{e}_i^{(k)}$ for $i \neq j$ and go to step 4 with the next data sample.

Figure 2 describes a block diagram of how the IN-FINDR can be broken up into two iterative procedures, an inner loop, and an outer loop via two counters, j and k.

It should be noted that the above IN-FINDR is generally not optimal as is SM N-FINDR. However, it can be considered as nearly optimal. The similarity between SM N-FINDR and IN-FINDR can be illustrated by the similarity between finding multiple integral and iterated integrals where the region to be integrated is generally a multi-dimensional space compared to the regions to be calculated by iterated integrals which are usually one-dimensional space. Using this context as interpretation, the exhaustive search implemented in SM N-FINDR is similar to finding a multiple integral which integrates a multi-dimensional region in the original data space, whereas the inner and outer loops carried out in IN-FINDR iteratively is similar to finding iterated integrals over one-dimensional space. In most cases, the results obtained by iterated integrals are the same as that obtained by directly finding multiple integrals according to Fubini's Theorem [10].

3 Various versions of real-time N-FINDR

The IN-FINDR presented in the previous section provides a key to real time processing of the SM N-FINDR by converting a simultaneous search for p endmembers to a sequential search for p endmembers. Nevertheless, there

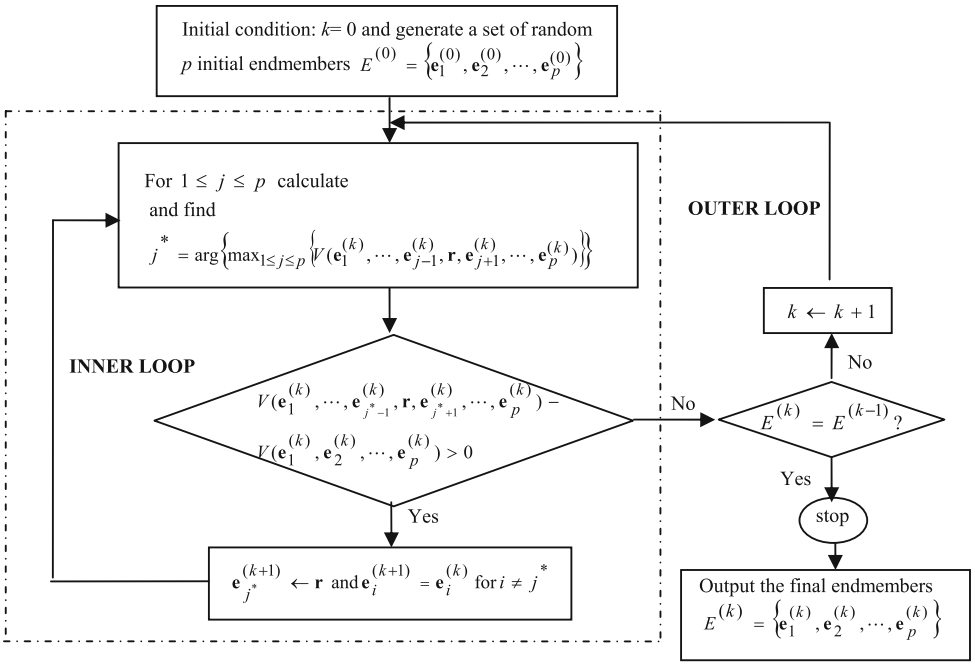


Fig. 2 Block diagram of IN-FINDR implementation

are two more issues needed to be addressed, step 1(b) which requires dimensionality reduction, and step 2 which uses randomly generated vectors as initial endmembers. While the number of endmembers, p must be known a priori in step 1(a), we can actually re-design the N-FINDR from a real-time processing view point by eliminating step 1(b) and step 2 all together in such a way that the dimensionality reduction required in step 1(b) is no longer needed and the p random initial vectors in step 2 are replaced with the first p data sample vectors inputted from the data set. The whole issue in real-time processing comes down to the inner and outer loop implemented in the IN-FINDR which converts the IN-FINDR to a sequential version of N-FINDR in steps 3–5. More specifically, steps 4–5 implemented in the IN-FINDR are replaced by a sequential search to update new endmembers in real time as new data samples come in. To accomplish this goal, three real-time processing versions of implementing the inner loop of the IN-FINDR are presented in the following subsections.

3.1 Real-time SeQuential N-FINDR (RT SQ N-FINDR)

According to the inner loop of the IN-FINDR, the replacement rule described in step 5 was executed by calculating the volume of a p-vertex simplex p times. This section presents an interesting alternative process, called Real Time SeQuential N-FINDR (RT SQ N-FINDR) to implement the inner loop of the IN-FINDR in real time which can be described as follows.

3.1.1 RT SQ N-FINDR

1. Initial condition: Assume that $\{\mathbf{r}_i\}_{i=1}^N$ are data sample vectors inputted according to 1, 2,…,N. Input the first p data sample vectors as the initial set of initial vectors, $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_p^{(0)}\}$, i.e., $\mathbf{e}_i^{(0)} = \mathbf{r}_i$ for $1 \le i \le p$. Set $k = 0$.
2. Let $k \leftarrow k+1$, i.e., kth data sample vector and calculate the volumes of the simplexes $S(\mathbf{r}_k, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)})$.$S(\mathbf{e}_1^{(k)}, \mathbf{r}_k, \mathbf{e}_3^{(k)}, \ldots, \mathbf{e}_p^{(k)})$, $\ldots, S(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{p-1}^{(k)}, \mathbf{r}_k)$ and find the index j_k that yields the maximum volume among all p endmembers, $\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \ldots, \mathbf{e}_p^{(k)}$ the r_k replaces, i.e.,
$$j_k = \arg\left\{\max_{1\le j\le p} V\left(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{j-1}^{(k)}, \underbrace{\mathbf{r}_k}_{j}, \mathbf{e}_{j+1}^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right)\right\}. \quad (3)$$
3. Check if
$$V\left(\mathbf{e}_1^{(k)}, \mathbf{e}_2^{(k)}, \mathbf{e}_3^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right) > V\left(\mathbf{e}_1^{(k)}, \ldots, \underbrace{\mathbf{r}_k}_{j_k}, \ldots, \mathbf{e}_p^{(k)}\right). \quad (4)$$
4. If (4) is true, go to step 5. Otherwise, let
$$\mathbf{e}_{j_k}^{(k)} \leftarrow \mathbf{r}_k. \quad (5)$$
and continue.
5. Check if $k = N$. If yes, the algorithm is terminated. Otherwise, let $\mathbf{e}_i^{(k+1)} = \mathbf{e}_i^{(k)}$ for $1 \le i \le p$ and go to step 2.

Figure 3 shows a block diagram to implement the RT SQ N-FINDR where the counter k is used to track how many endmembers, i.e., how many passes have been processed.

Two following comments are noteworthy.

1. Interestingly, if we replace the inner loop of the IN-FINDR by the RT SQ N-FINDR and each run of the RT SQ N-FINDR is considered one pass, the IN-FINDR can be implemented in real time as an RT multiple-pass SQ N-FINDR where the number of passes to complete the IN-FINDR is determined by the index k used in its outer loop. Accordingly, the RT multiple-pass SQ N-FINDR can be considered as real-time processing of IN-FINDR, RT IN-FINDR.
2. It is important to realize that (3) can be implemented without dimensionality reduction. First, the volume is calculated by a determinant not a matrix inverse. Second, the determinant is actually a product of all nonzero eigenvlaues of a matrix in (1) which has only $p-1$ nonzero eigenvalues. In this case, we only need to calculate the characteristic polynomial equation to find all nonzero eigenvalues without performing dimensionality reduction.

3.2 Real-time circular N-FINDR

In the above RT SQ N-FINDR, it requires computation of the volume of a p-vertex simplex p times for each data sample $\mathbf{r}_k$ after the first p samples, i.e., $k > p$. To further reduce computational complexity, it is highly desirable to calculate one simplex volume for each data sample vector in a circular manner of finding one optimal endmember at a time while the other endmembers remaining fixed. As a result of such a circular process, a significant reduction in computational complexity can be achieved. This section proposes an alternative algorithm to the RT SQ N-FINDR, to be called real-time circular N-FINDR (RT Circular N-FINDR) whose idea can be illustrated as follows.

Assume that $\{\mathbf{r}_i\}_{i=1}^{N}$ are data sample vectors. We first use the 1st p pixel vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_p$ as p initial endmembers, $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_p^{(0)}\}$, then we begin to process whether or not the first initial endmember $\mathbf{e}_1^{(0)}$ needs to be replaced according to the criterion as a new data sample vector $\mathbf{r}_{p+1}$ is processed. The new updated first endmember is then denoted by $\mathbf{e}_1^{(1)}$ with superscript indicating the first iteration. After the $(p+1)$st is processed, we input the next new data sample vector $\mathbf{r}_{p+2}$ to process if the second initial endmember $\mathbf{e}_2^{(0)}$ needs to be replaced where the new updated second endmember is denoted by $\mathbf{e}_2^{(1)}$. The process is continued on until it reaches the 2pth data sample vector used to update the pth initial endmember $\mathbf{e}_p^{(0)}$ with the new updated pth endmember $\mathbf{e}_p^{(1)}$. After this stage, all the p initial endmembers have been updated a new set of p endmembers $\{\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \ldots, \mathbf{e}_p^{(1)}\}$ have been updated in which case the entire process is completed as a cycle. By the time the $(2p+1)$ data sample vector comes in, a second cycle is begun to update $\{\mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \ldots, \mathbf{e}_p^{(1)}\}$. The same p-cycle process is repeated over and over again until it reaches the last data sample vector $\mathbf{r}_N$. For an illustrative purpose, Fig. 1 details the circular iterative procedure carried out by the RT Circular N-FINDR where only

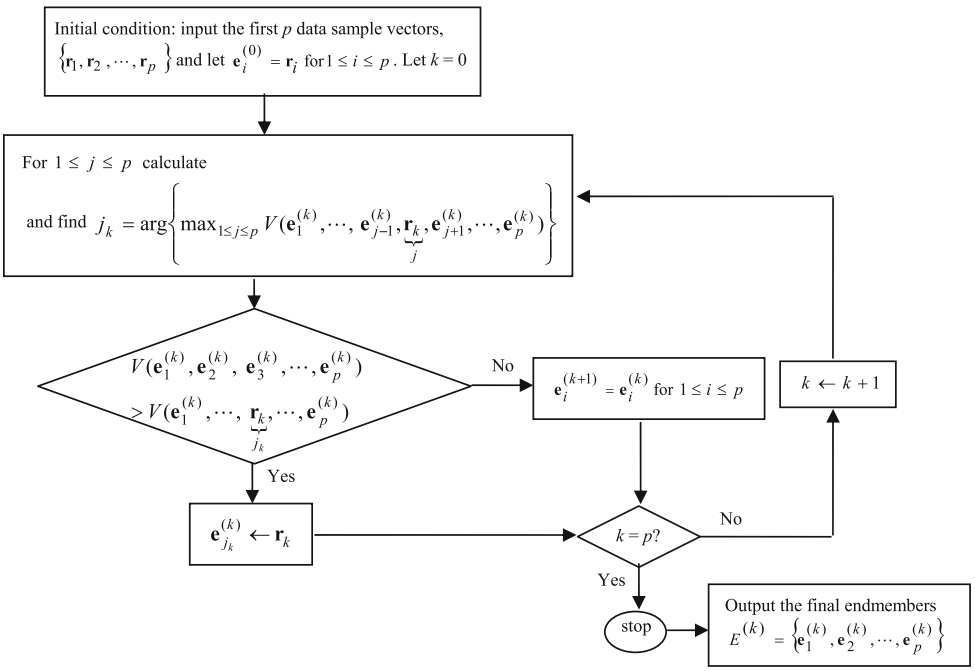


Fig. 3 Block diagram of RT SQ N-FINDR implementation

indexes of data sample vectors are used for simplicity and j is defined as $j \equiv \hat{j}\,(\mathrm{mod}p)$ if $j = kp + \hat{j}$ with $1 \le \hat{j} \le p$.

Using the concept of the iterative circular procedure described in Fig. 4, a detailed implementation of the RT Circular N-FINDR is given below.

3.2.1 Real-time circular N-FINDR

1. Initial condition: Assume that $\{\mathbf{r}_i\}_{i=1}^{N}$ are data sample vectors inputted according to 1, 2,…,N. Input the first p pixel vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_p$ as p initial endmembers, $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_p^{(0)}\}$ by setting $\mathbf{r}_i = \mathbf{e}_i^{(0)}$. Set $k = 0$ and $j = 1 \equiv \hat{j}\,(\mathrm{mod}p)$.
2. Let $(kp + \hat{j})^{\mathrm{th}}$ pixel vector in the image cube data be denoted by $\mathbf{r}_{kp+\hat{j}}$ calculate $V(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{\hat{j}-1}^{(k)}, \mathbf{r}_{kp+\hat{j}}, \mathbf{e}_{\hat{j}+1}^{(k)}, \ldots, \mathbf{e}_p^{(k)})$,, the volume of $S(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{j-1}^{(k)}, \mathbf{r}_{kp+\hat{j}}, \mathbf{e}_{j+1}^{(k)}, \ldots, \mathbf{e}_p^{(k)})$, and check if
$$V\left(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{\hat{j}-1}^{(k)}, \mathbf{r}_{kp+\hat{j}}, \mathbf{e}_{\hat{j}+1}^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right) \le V\left(\mathbf{e}_1^{(k)}, \ldots, \mathbf{e}_{\hat{j}-1}^{(k)}, \mathbf{e}_{\hat{j}}^{(k)}, \mathbf{e}_{\hat{j}+1}^{(k)}, \ldots, \mathbf{e}_p^{(k)}\right)$$
If it is, go to step 4. Otherwise, continue.
3. Replacement rule: The endmember pixel $\mathbf{e}_{\hat{j}}^{(k)}$ will be replaced by the sample vector $\mathbf{r}_{kp+\hat{j}}$ and will be re-labeled by $\mathbf{e}_{\hat{j}}^{(k+1)}$. A new set of endmembers is then produced by letting $\mathbf{e}_{\hat{j}}^{(k+1)} = \mathbf{r}_{kp+\hat{j}}$ and $\mathbf{e}_i^{(k+1)} = \mathbf{e}_i^{(k)}$ for $i \neq \hat{j}$. Go to step 4.
4. If the next pixel vector $\mathbf{r}_{i+1}$, $\mathbf{r}_{i+1} \neq \mathbf{r}_N$, let $k \leftarrow k + 1$ and $j \leftarrow j + 1$. Find $j \equiv \hat{j}\,(\mathrm{mod}\,p)$ and go to step 2. Otherwise, the algorithm is terminated.

Two reasons for the RT Circular N-FINDR to be implemented in such a circular order are (1) to make sure that all endmembers have equal opportunity to be replaced if necessary and (2) to reduce computational complexity to calculate a p-vertex simplex volume only once at a time instead of p-vertex simplexes p times at a time.

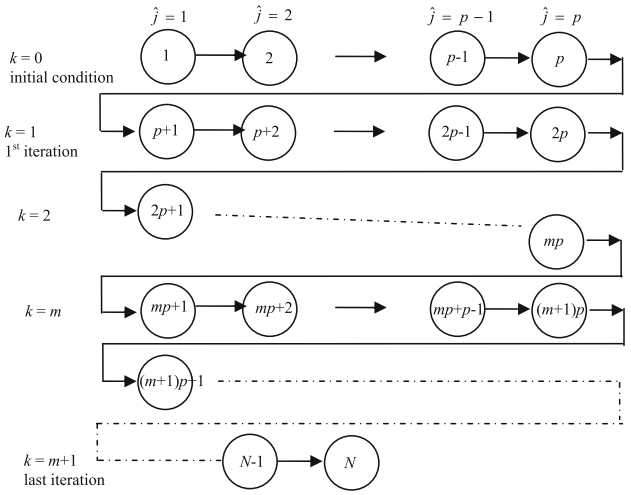


Fig. 4 Iterative circular procedure carried out by RT Circular N-FINDR

According to the above implementation, the initial endmembers used by the RT Circular N-FINDR have tremendous impact on final selected endmembers. This issue can be addressed by an algorithm similar to multiple-pass SQ N-FINDR, to be called multiple-pass RT circular N-FINDR which re-runs the RT Circular N-FINDR as a second round process using the final p endmembers found in its first round process as a new set of initial endmembers. The final set of p endmembers found by a second run of the RT Circular N-FINDR should result in a better set of p endmembers than those p endmembers found by its first round. Similarly, the same process can be repeated over and over again for the third round RT Circular N-FINDR using the second round found p endmembers as a new initial set of p endmembers, the fourth round RT Circular N-FINDR using the third round found p endmembers as a new initial set of p endmembers, and so on. Another advantage of using multiple passes is to add a global shift at mth run which makes $\mathbf{r}_{kp+\hat{j}}$ to replace $\mathbf{e}_{(\hat{j}+m)\mathrm{mod}p}^{(k)}$ instead of $\mathbf{e}_{\hat{j}}^{(k)}$. It prevents each data sample from being replaced by a pixel at the same location in different passes. The idea of using passes to describe how many times required to re-run the same algorithm is exactly the same as multiple-pass RT SQ N-FINDR which uses the counter index in the outer loop by the RT IN-FINDR. The number of passes the RT Circular N-FINDR should be run is determined by the final sets of endmembers found by two consecutive runs are identical. Accordingly, the RT multiple-pass circular N-FINDR resolves the issue of dependency on initial conditions by including an outer loop to re-run the RT Circular N-FINDR as the RT multiple-pass SQ N-FINDR does for the RT IN-FINDR in the sense that the RT SQ N-FINDR only executes the inner loop of the RT IN-FINDR. The difference between the RT multiple-pass circular N-FINDR and the RT multiple-pass SQ N-FINDR is their replacement rules described in step 3. The latter replaces the endmember which yields the smallest simplex volume among all the existing p endmembers as a new data sample vector comes in as opposed to the former which only replaces one endmember at a time in a circular order which starts from $j = 1$ to $j = p$ and then circulates back to $j = 1$ to $j = p$ over and over again when new data sample vectors are fed in. Compared to the RT Circular N-FINDR, the RT SQ N-FINDR requires an additional register to record the endmember to be replaced during the data processing, while the RT Circular N-FINDR does not. Interestingly, the RT multiple-pass circular N-FINDR can be also implemented as the RT IN-FINDR with the RT SQ N-FINDR replaced by the RT Circular N-FINDR in which case the RT multiple-pass circular N-FINDR can be also viewed as another version of the RT IN-FINDR.

Finally, we would like to conclude with the following two remarks.

(a) Since the RT Circular N-FINDR calculate one simplex volume for an endmember at a time compared to the RT SQ N-FINDR which calculates p simplex volume for an endmember, the number of passes required for RT multiple-pass circular N-FINDR is generally greater than that required for RT multiple-pass SQ N-FINDR. According to our extensive experiments on various data sets, the RT multiple-pass Circular N-FINDR always completes its process before p passes and never goes beyond p passes. However, on some occasions, there are multiple pixels whose spectral signatures are very close and can be used to specify the same endmembers. Under such circumstance, these different pixels may be extracted by the RT Circular N-FINDR at different runs and may cause the RT multiple-pass circular N-FINDR not to converge after p passes. Additionally, due to the global shift implemented in each pass, we can always terminate the RT multiple-pass circular N-FINDR after it completes p passes and the result obtained at the pth pass will be used as the final results. This same criterion is also applied to the RT multiple-pass SQ N-FINDR even though such a rare case may never occur to the RT SQ N-FINDR because the RT SQ N-FINDR always searches for p optimal endmembers instead of one optimal endmember at a time by the RT Circular N-FINDR.

(b) The RT multiple-pass circular N-FINDR and the RT multiple-pass SQ N-FINDR may not be considered as real-time processing algorithms since each new pass creates an extra time lag. However, if such a time lag is negligible, both can be considered as near real-time processing algorithms. Additionally, both algorithms may not be able to produce outputs in real time if too many passes are required. Nevertheless, they are still considered as a causal processing because their executing processes require no future data sample vectors to update endmembers.

3.3 Real time SuCcessive N-FINDR (RT SC N-FINDR)

When the RT IN-FINDR (i.e., RT multiple-pass SQ N-FINDR) and RT multiple-pass circular N-FINDR described in the previous section are performed, there is no knowledge about how many passes will be required before the algorithms are terminated. This section presents a new algorithm, to be called teal time p-Pass SuCcessive N-FINDR (RT p-Pass SC N-FINDR) by combining the concepts of the RT SQ N-FINDR and multiple-pass RT circular N-FINDR to perform what the RT multiple-pass SQ N-FINDR does for the RT IN-FINDR except that we know exactly how many passes the algorithm should run, which is the number of endmembers, p. The details of RT p-Pass SC N-FINDR is provided in the following algorithm.

3.3.1 p-Pass RT SC N-FINDR

1. Initial condition: Set $j = 1$ and input the first p pixel vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_p$ as p initial endmembers, $\{\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \ldots, \mathbf{e}_p^{(0)}\}$.
2. At the jth pass, we are supposed to find the jth endmember e_j and the endmembers $\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}$ prior to the jth endmember e_j are assumed to be known. $\{\mathbf{r}_i\}_{i=1}^{N}$ are data sample vectors inputted according to 1, 2,…,N. Let $k = 1$.
3. For $\mathbf{r}_k \notin \{\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}\}$ calculate the volume of the simplex, $S(\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}, \mathbf{r}_k, \mathbf{e}_{j+1}, \ldots, \mathbf{e}_p)$ and set $\max\_\mathrm{volume}(j) = V(\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}, \mathbf{r}_k, \mathbf{e}_{j+1}, \ldots, \mathbf{e}_p)$. Then calculate $V(\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}, \mathbf{r}_{k+1}, \mathbf{e}_{j+1}, \ldots, \mathbf{e}_p)$ and compare it to $\max\_\mathrm{volume}(j)$. If $V(\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}, \mathbf{r}_{k+1}, \mathbf{e}_{j+1}, \ldots, \mathbf{e}_p) > \max\_\mathrm{volume}(j)$, then $\max\_\mathrm{volume}(j) \leftarrow V(\mathbf{e}_1^{(*)}, \mathbf{e}_2^{(*)}, \ldots, \mathbf{e}_{j-1}^{(*)}, \mathbf{r}_{k+1}, \mathbf{e}_{j+1}, \ldots, \mathbf{e}_p)$ and check if
$$\mathbf{r}_{k+1} = \mathbf{r}_N. \tag{6}$$
4. If (6) is not true, let $k \leftarrow k + 1$ and go to step 3. Otherwise, continue.
5. In this case, the jth endmember $\mathbf{e}_j^{(*)}$ is found and continue to find the next $(j + 1)$ endmembrer, $\mathbf{e}_{j+1}^{(*)}$. If $j = p$, the algorithm is terminated. Otherwise, let $j \leftarrow j + 1$ and go to step 2.

Figure 5 provides a block diagram of the RT SC N-FINDR implementation which describes the above procedure in a simple manner where the counter j keeps track of how many endmembers, i.e., how many passes, have been processed.

It should be noted that the above RT p-Pass SC N-FINDR is also implemented by two loops, outer loop specified by j to keep tracks of how many passes the algorithm has been executed and inner loop specified by k which updates endmembers to find the jth endmember, $\mathbf{e}_j^*$ for a specific j. But these two inner and outer loops are different from the inner and outer loops implemented by the RT IN-FINDR. More specifically, the RT IN-FINDR uses the outer loop to eliminate the dependency of the algorithm on the specific initial condition, while the inner loops is to find all the p endmembers for a given initial condition. By contrast, the RT SC N-FINDR uses the outer loop to specify

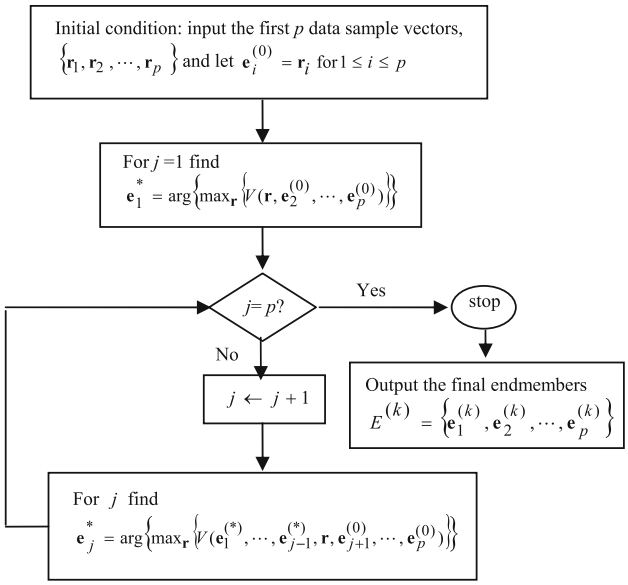


Fig. 5 Block diagram of RT SC N-FINDR implementation

a particular endmember needed to be generated by starting an initial endmember and then uses the inner loop to keep updating the initial endmember by using incoming data sample vectors to find this particular endmember. Therefore, the RT SC N-FINDR performs more like the RT 1-pass SQ N-FINDR which is a special case of the IN-FINDR that runs the RT SQ N-FINDR only one time.

It is also interesting to note that when the RT multiple-pass SQ N-FINDR and RT multiple-pass circular N-FINDR are implemented as p passes, they become RT p-Pass SQ N-FINDR and RT p-Pass Circular N-FINDR. However, we should note that if the RT multiple-pass SQ N-FINDR and RT multiple-pass circular N-FINDR implemented as the RT IN-FINDR, both require variable numbers of passes to complete its search for p endmembers which are generally fewer than p passes.

Finally, assume that c_j is the cost of calculating the volume of a j-vertex simplex. Table 1 tabulates the computational complexity based on number of times to calculate various simplexes required by the N-FINDR, RT IN-FINDR, RT SQ N-FINDR, RT p-pass Circular N-FINDR, RT p-pass SC N-FINDR where the K is the total number of times for the outer loop to be executed in the IN-FINDR.

By concluding this section, it should be noted that all the RT versions of N-FINDR presented in this section, RT IN-FINDR (RT multiple-pass SQ N-FINDR or RT multiple-pass circular N-FINDR), RT SQ N-FINDR, RT Circular N-FINDR, RT Circular N-FINDR and RT p-Pass SC N-FINDR, do not require dimensionality reduction as does the SM N-FINDR. This advantage allows algorithms to be applicable to feasibility of real time processing. In addition, the use of the first p endmembers as initial endmembers can be considered as a special case of so-called endmember initialization algorithm (EIA) [5]. In a causal processing, we do not have such a luxury to use a custom-designed EIA. The only option that we have is to use the first p endmembers as the initial endmembers. However, on the other hand, since an EIA-based N-FINDR can be considered as a special case of SQ N-FINDR due to the fact that the EIA-based N-FINDR uses a particular set of initial endmembers generated by an EIA, it is a suboptimal version of the IN-FINDR which iteratively runs the SQ N-FINDR as a multiple-pass SQ N-FINDR by improving its initial endmembers in each run of SQ N-FINDR. In this case, it is not included in our comparative analysis.

4 Synthetic image experiments

In this section, synthetic images were simulated by the Cuprite image data shown in Fig. 6a which is available at the USGS website [11]. The scene is a 224-band image with size of 350 × 350 pixels and was collected over the Cuprite mining site, Nevada, in 1997. It is well understood mineralogically. As a result, a total of 189 bands were used for experiments where bands 1–3, 105–115 and 150–170 have been removed prior to the analysis due to water absorption and low SNR in those bands. Although there are more than five minerals on the data set, the ground truth available for this region only provides the locations of the pure pixels: Alunite (A), Buddingtonite (B), Calcite (C), Kaolinite (K) and Muscovite (M). The locations of these five pure minerals are labeled by A, B, C, K, and M, respectively, and shown in Fig. 6b. Available from the image scene is a set of spectra, reflectance spectra shown in Fig. 6c which will be used to simulate synthetic images. The sample mean of the entire image shown in Fig. 6a was used to simulate the background for image scene in Fig. 7 also plotted in Fig. 6c.

The synthetic image to be simulated for experiments has size of 200 × 200 pixel vectors with 25 panels of various sizes which are arranged in a 5 × 5 matrix and located at the center of the scene shown in Fig. 7. The 25 panels in

Table 1 Computational complexity of various versions of N-FINDR

N-FINDR	RT IN-FINDR	RT SQ N-FINDR	RT p-pass Circular N-FINDR	RT p-pass SC N-FINDR
$\frac{N!}{(N-p)!p!} \cdot c_p$	$c_p \cdot p \cdot (N-p) \cdot K$	$c_p \cdot p \cdot (N-p)$	$c_p \cdot p \cdot (N-p)$	$c_p \cdot p \cdot (N-p)$

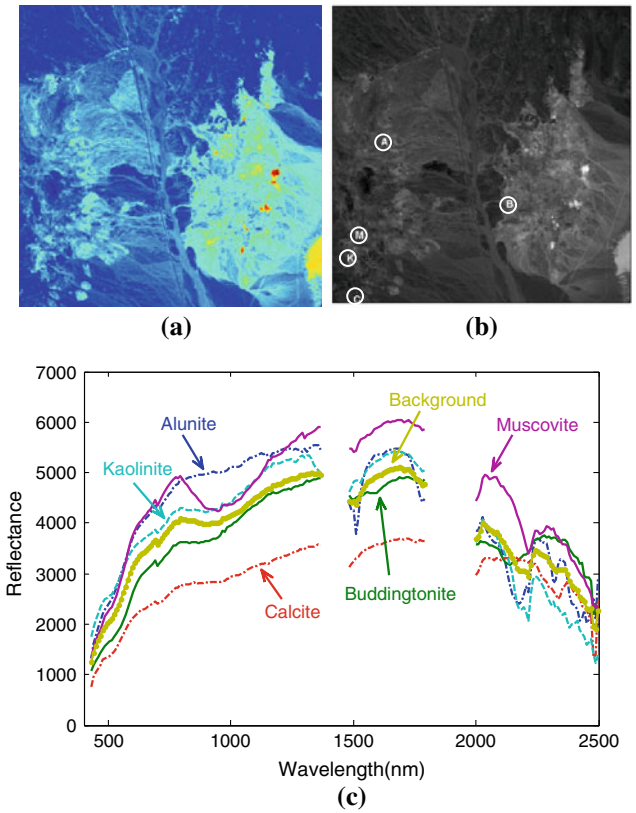


Fig. 6 **a** Cuprite AVIRIS image scene, **b** spatial positions of five pure pixels corresponding to minerals: alunite (A), buddingtonite (B), calcite (C), kaolinite (K) and muscovite (M); **c** Five mineral reflectance spectra and background signature (**b**)—which is the sample mean of the image in (**a**)

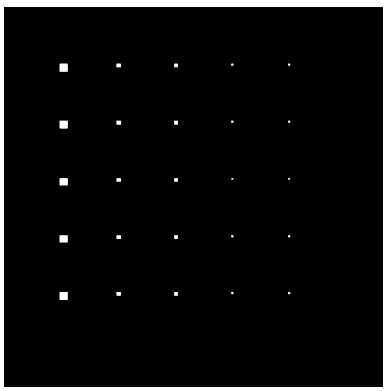


Fig. 7 25 simulated panels according to Tables 2 and 3

Fig. 7 were simulated to be used for the six scenarios described below. The five mineral spectral signatures, $\{\mathbf{m}_i\}_{i=1}^{5}$ in Fig. 6c are used to simulate these 25 panels where each row of five panels was simulated by the same mineral signature and each column of 5 panels has the same size. Among 25 panels are five 4×4 pure-pixel panels, $p_{4\times4}^{i}$ for $i = 1,\ldots,5$ lined up in five rows in the first column and five 2×2 pure-pixel panels, $p_{2\times2}^{i}$ for $i = 1,\ldots,5$ lined up in five rows in the second column for pure pixel classification; the five 2×2-mixed pixel panels, $\{p_{3,jk}^{i}\}_{j=1,k=1}^{2,2}$ for $i = 1,\ldots,5$ lined up in five rows in the third column for mixed pixel classification and both the five subpixel panels, $p_{4,1}^{i}$ for $i = 1,\ldots,5$ lined up in five rows in the fourth column and the five sub-pixel panels, $p_{5,1}^{i}$ for $i = 1,\ldots,5$ lined up in five rows in the fifth column for subpixel classification. The purpose of introducing the five panels in the third column and subpixel panels in the fourth and fifth columns was designed to conduct a study and analysis on five mineral signatures with different mixing in a pixel and five mineral signatures embedded in single pixels at subpixel scale.

Tables 2 and 3 tabulate the mixing details of mineral composition in the 20 mixed pixels in the third column in Fig. 7 and the 5 subpixel panels with 50% abundance of mineral signatures in the fourth column and the 5 subpixel panels with 25% abundance of mineral signatures in the fifth columns in Fig. 7, respectively.

Therefore, in Fig. 7 there are a total of 130 panel pixels present in the scene, 80 pure panel pixels in the first column, 20 pure panel pixels in the second column, 20 mixed panel pixels in the third column, five 50%-abundance subtarget panel pixels in the fourth column and five 25%-abundance subpixel target panel pixels in the fifth column.

Once targets are simulated as above, we further use the sample mean of the image scene in Fig. 6a as a background signature to simulate a zero mean Gaussian noise-corrupted image background. It should be noted that since all

Table 2 Mixed panel pixels in the third column for simulations

Row 1	$p_{3,11}^{1} = 0.5A + 0.5B$	$p_{3,12}^{1} = 0.5A + 0.5C$
	$p_{3,21}^{1} = 0.5A + 0.5K$	$p_{3,22}^{1} = 0.5A + 0.5M$
Row 2	$p_{3,11}^{2} = 0.5A + 0.5B$	$p_{3,12}^{2} = 0.5B + 0.5C$
	$p_{3,21}^{2} = 0.5B + 0.5K$	$p_{3,22}^{2} = 0.5B + 0.5M$
Row 3	$p_{3,11}^{3} = 0.5A + 0.5C$	$p_{3,12}^{3} = 0.5B + 0.5C$
	$p_{3,21}^{3} = 0.5C + 0.5K$	$p_{3,22}^{3} = 0.5C + 0.5M$
Row 4	$p_{3,11}^{4} = 0.5A + 0.5K$	$p_{3,12}^{4} = 0.5B + 0.5K$
	$p_{3,21}^{4} = 0.5C + 0.5K$	$p_{3,22}^{4} = 0.5K + 0.5M$
Row 5	$p_{3,11}^{5} = 0.5A + 0.5M$	$p_{3,12}^{5} = 0.5B + 0.5M$
	$p_{3,21}^{5} = 0.5C + 0.5M$	$p_{3,22}^{5} = 0.5K + 0.5M$

Table 3 Subpixels in the 4th and 5th columns for simulations

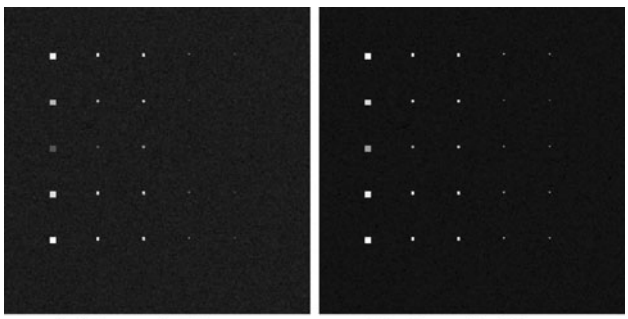
	50% Subpixel panels in fourth column	25% Subpixel panels in fifth column
Row 1	$p_{4,1}^{1} = 0.5A + 0.5\mathbf{b}$	$p_{5,1}^{1} = 0.25A + 0.75\mathbf{b}$
Row 2	$p_{4,1}^{2} = 0.5B + 0.5\mathbf{b}$	$p_{5,1}^{2} = 0.25B + 0.75\mathbf{b}$
Row 3	$p_{4,1}^{3} = 0.5C + 0.5\mathbf{b}$	$p_{5,1}^{3} = 0.25C + 0.75\mathbf{b}$
Row 4	$p_{4,1}^{4} = 0.5K + 0.5\mathbf{b}$	$p_{5,1}^{4} = 0.25K + 0.75\mathbf{b}$
Row 5	$p_{4,1}^{5} = 0.5M + 0.5\mathbf{b}$	$p_{5,1}^{5} = 0.25M + 0.75\mathbf{b}$

different spectral band images have various signal energies, in order for each spectral band image to achieve the same level of signal-to-noise ratio (SNR) defined as 50% signature (i.e., reflectance/radiance) divided by the standard deviation of the noise in [12], zero-mean Gaussian noises with different variances were used and added to different spectral band images for this purpose. Once target pixels and background are simulated, two types of target insertion, referred to as target implantation (TI) and target embeddedness (TE), can be designed to simulate experiments for various applications.

4.1 Target implantation (TI)

The first type of target insertion is referred to as target implantation (TI) which inserts the above 130 panel pixels into the image by replacing their corresponding background pixels. Therefore, the resulting synthetic image has clean panel pixels implanted in a noisy background with an additive Gaussian noise of SNR = 20:1 for this scenario as shown in Fig. 8a.

According to the ground truth, there have five pure distinct mineral signatures, A, B, C, K, and M used to simulate pure panel pixels in the first and second columns and one mixed background signature. Therefore, technically speaking, there are only five endmembers present in TI. But those pixels in the first two columns simulated by pure signatures can be considered as endmember pixels. In this case, there are a total of 100 endmember pixels with 80 endmember pixels in the first column and 20 endmember pixels in the second column. Since no prior knowledge about the scenario TI is provided, a new concept, Virtual Dimensionality (VD) in [7, 8] was used to estimate the number of endmembers that were present in the data where a noise-whitened HFC (NWHFC) method developed by Harsanyi et al. [13] was used for VD estimation for the simulated images. The VD estimated for TI was 5 as long as the false alarm probability $P_F \leq 10^{-1}$. Since the real time processing N-FINDR algorithms are a sequential process, we used $p = 5$ as the number of endmembers required for



(a) Scenario TI **(b)** Scenario TE

Fig. 8 Two scenarios for target insertion, TI and TE

N-FINDR to generate. Since the RT IN-FINDR implemented the RT SQ N-FINDR repeatedly, Fig. 9 shows the results of RT IN-FINDR where Fig. 9a–d dictates a progressive process of the first pass implemented by the RT SQ N-FINDR and Fig. 9e–f are results after the RT SQ N-FINDR completed 2 passes and 3 passes, respectively. The RT IN-FINDR was terminated when the results after 2 passes and 3 passes were identical in which case no more pass was required for the SQ N-FINDR to re-run again. As noted in the real-time progressive endmember extraction presented in Fig. 9a–c, the extracted endmembers kept changing until the first pass was completed in Fig. 9d where the first four extracted pixels were indeed endmember pixels. Now, the RT SQ N-FINDR was re-run again to determine whether or not the SQ N-FINDR must be terminated. Figure 9e shows the results after it completed two passes where one of third panel pixels was extracted as the fifth pixel which corresponded to the last pure material signature missed in the first pass. The RT SQ N-FINDR requires a re-run for a third pass where the same five endmember pixels were extracted. In this case, the RT IN-FINDR would have been terminated. In order to see the performance of the RT 5-pass Circular N-FINDR and RT 5-pass SC N-FINDR, Figs. 10, 11 show their 5-pass results, respectively, where both algorithms also extracted the five panel pixels as endmembers that specified five pure mineral signatures. It should be noted that the RT 5-pass Circular N-FINDR was implemented in this experiment to see how many passes would be required for implementing the RT multiple-pass circular N-FINDR before five passes. However, there is an interesting finding in Figs. 10, 11 where both actually generated all the desired five endmembers with only three passes for RT 5-pass Circular N-FINDR and five passes for RT 6-pass SC N-FINDR. Therefore, if the IN-FINDR was implemented as RT Multiple Circular N-FINDR, it could have been completed after four passes. Accordingly, in terms of the minimum number of passes required to extract the five endmembers, the RT Multiple Circular N-FINDR turned out to be the best with only three passes followed by the RT Multiple Circular N-FINDR implemented as an alternative IN-FINDR (4 passes) and RT 5-pass SC N-FINDR (5 passes).

If we further document computational cost required for each of the five algorithms, RT IN-FINDR implemented as multiple-pass SQ N-FINDR, RT IN-FINDR implemented as multiple-pass circular N-FINDR, RT SQ N-FINDR, RT 5-pass Circular N-FINDR and RT 5-pass SC N-FINDR, Table 4 tabulates their computing times in seconds which show that the best one was the RT Multiple Circular N-FINDR followed by the RT 5-pass SC N-FINDR which was very close to the RT 5-pass Circular N-FINDR. It should be noted that the computing time required for the SQ N-FINDR is nearly the same as that required by the RT

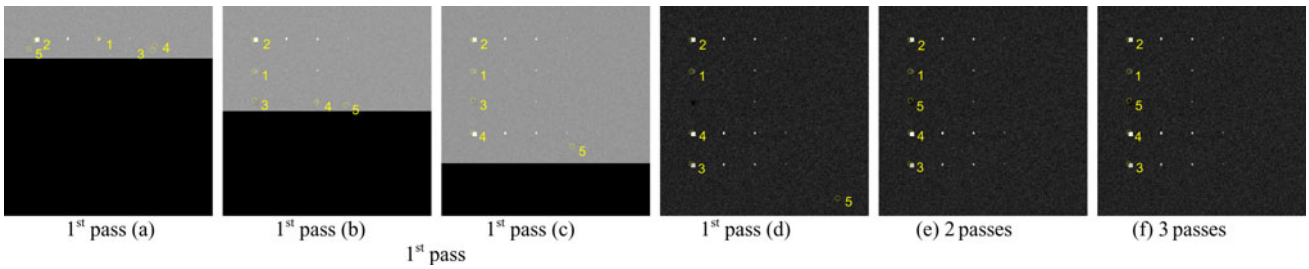


Fig. 9 Progress of five endmembers extracted by RT IN-FINDR on scenario TI: first pass of RT SQ N-FINDR (**a**–**d**); results of passes increased by one, **e** after 2 passes (**f**) after 3 passes

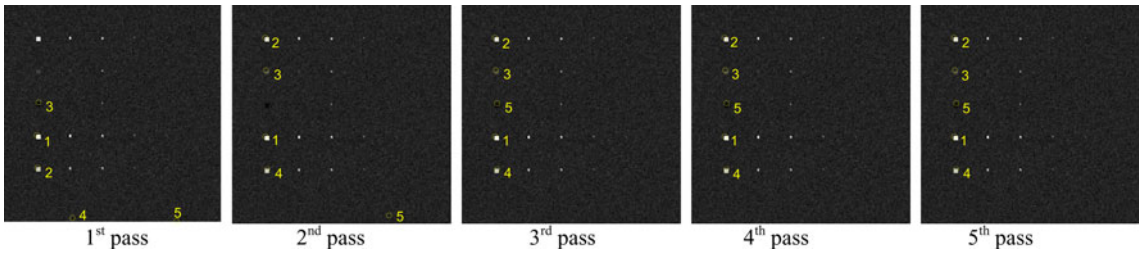


Fig. 10 Results of each pass of RT 5-Pass Circular N-FINDR

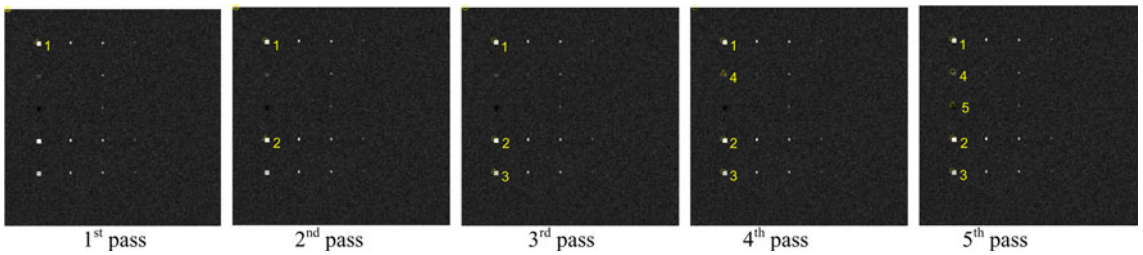


Fig. 11 Results of each pass of RT 5-Pass SC N-FINDR

5-pass SC N-FINDR due to the fact that the latter needs to implement 5 passes to accomplish what the 1-pass SQ N-FINDR does. For further comparison we also include results of running two non-real time sequential endmember extraction algorithms, simplex growing algorithm (SGA) [9] and vertex component analysis (VCA) [14] where VCA shows the best time since it only performs orthogonal projection which requires much less time than computing simplex volumes. In order to make fair comparison the SGA and VCA were also performed without dimensionality reduction. However, it should be noted that the VCA is not part of a family of the simplex-based algorithms and it also cannot be implemented as a real-time processing algorithm. Its results are simply included for reference. Interestingly, the results also show that except the IN-FINDR all other RT N-FINDR algorithms require less time than the SGA which computes volumes of a growing number of simplexes. This may be due to the fact that the SGA must find a set of growing simplexes with maximum volumes as the p grows compared to our proposed sequential RT N-FINDR algorithms which find a p-vertex simplex with maximum volume for a fixed p by replacing one vertex at a time while other $p-1$ vertices are fixed unchanged.

4.2 Target embeddedness (TE)

In analogy with scenario TI, another type of target insertion is referred to as target embeddedness (TE) is simulated except the way how the panel pixels were inserted. The background pixels were not removed to accommodate the inserted panel pixels as they were done in TI, but were rather superimposed with the inserted panel pixels. Therefore, in this case, the resulting synthetic image shown in Fig. 8b has clean panel pixels embedded in a noisy background. In this case, there were no pure signatures present in the TE because the panel pixels were superimposed on the background pixels. Therefore, technically

Table 4 Computing times of RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT 5-pass Circular N-FINDR and RT 5-pass SC N-FINDR

Algorithms	Computing time (s)
RT multiple-pass SQ N-FINDR implemented as RT IN-FINDR	39.6454 (3 passes)
RT multiple-pass circular N-FINDR implemented as RT IN-FINDR	10.9582 (4 passes)
RT SQ N-FINDR	13.2716
RT 6-Pass Circular N-FINDR	13.3979
RT 6-Pass SC N-FINDR	13.3614
SGA	20.43
VCA	2.42

speaking the number of endmembers should be zero. Nevertheless, theoretically speaking there are still five endmembers to be used plus the background signature to compose all data sample vectors to make up the data set. Under such circumstance, there are no endmember pixels. Therefore, what we would want to have is to extract pixels which are most likely pure pixels even they are not. This scenario commonly occurs in real world applications where there may not have any pure signatures in the data in which case the best we can do is to find most pure signatures resident in the data. The VD estimated by the noise-whitened HFC method for TE was also five with the false alarm probability $P_F \leq 10^{-1}$. In order to compare the results obtained for the scenario TI, the same experiments conducted for scenario TI were also performed for scenario TE. Figure 12 shows the results of RT IN-FINDR where Figure 12a–d dictate a real-time progressive process of the first pass implemented by the RT SQ N-FINDR and Figure 12e–f are results after the RT SQ N-FINDR completed two passes and three passes, respectively. The RT IN- FINDR was terminated when the results after two passes and three passes were identical in which case no more pass was required for the SQ N-FINDR to re-run again. Comparing the results in Figs. 12, 13, 14 to that in Figs. 9, 10, 11 all the four RT processing N-FINDR algorithms, RT IN-FINDR, RT SQ N-FINDR, RT 5-pass Circular N-FINDR and RT 5-pass SC N-FINDR were able to extract each panel pixel from five rows. The same phenomenon was also observed where RT 5-pass Circular N-FINDR extracted all the five endmembers at its second pass, while the RT 5-pass SC N-FINDR accomplished its task at the third pass. In other words, when the multiple-pass circular N-FINDR was implemented as alternative IN-FINDR, it would have completed its process after three passes. It is the same as 3 passes required for the IN-FINDR using the RT SQ N-FINDR.

Table 5 also tabulates computing times in seconds required by RT IN-FINDR implemented as multiple-pass SQ N-FINDR, RT IN-FINDR implemented as multiple-pass circular N-FINDR, RT SQ N-FINDR, RT 5-pass Circular N-FINDR and RT 5-pass SC N-FINDR along with SGA and VCA where the same conclusion drawn from Table 4 was also applied here and the best RT N-FINDR algorithm was still the RT Multiple Circular N-FINDR.

5 Real image experiments

In this section, two real data, HYperspectral Digital Imagery Collection Experiment (HYDICE) and Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) Cuprite data, were used to make a performance comparison among various real time N-FINDR processing algorithms proposed in this paper. All these algorithms were running on the original image without any data dimensionality reduction.

5.1 HYDICE data

The first image data to be studied is an image scene shown in Fig. 15a, which has a size of 64 × 64 pixel vectors with 15 panels in the scene and the ground truth map in Fig. 15b. It was acquired by 210 spectral bands with a spectral coverage from 0.4 to 2.5 μm. Low signal/high noise bands: bands 1–3 and bands 202–210; and water

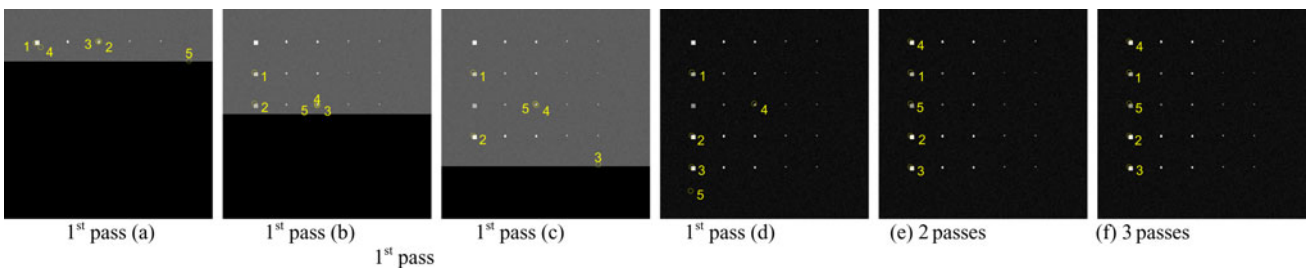


Fig. 12 Progress of 5 endmembers extracted by RT IN-FINDR: 1st pass of RT SQ N-FINDR (**a–d**); results of passes increased by one, **e** after two passes **f** after three passes

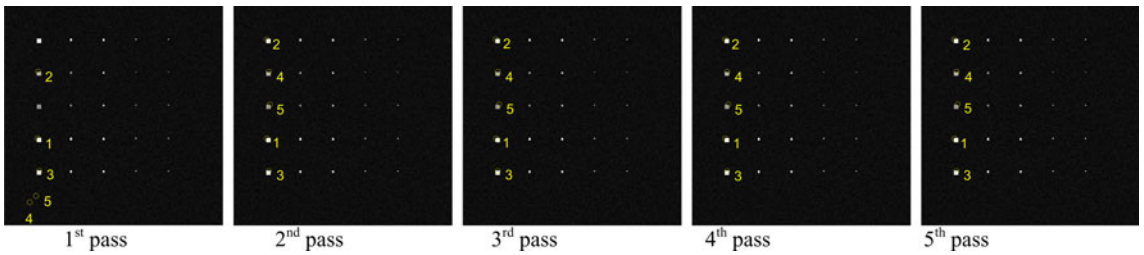


Fig. 13 Results of each pass of RT 5-Pass Circular N-FINDR

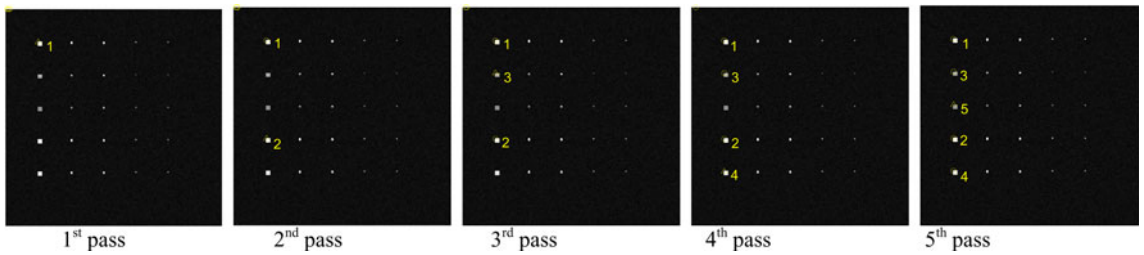


Fig. 14 Results of each pass of RT 5-Pass SC N-FINDR

Table 5 Computing times of RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT 5-pass Circular N-FINDR and RT 5-pass SC N-FINDR

Algorithms	Computing time (s)
RT multiple-pass SQ N-FINDR implemented as RT IN-FINDR	40.2761 (3 passes)
RT multiple-pass circular N-FINDR implemented as RT IN-FINDR	7.9062 (3 passes)
RT SQ N-FINDR	13.3364
RT 6-Pass Circular N-FINDR	13.3841
RT 6-Pass SC N-FINDR	13.3838
SGA	20.28
VCA	2.54

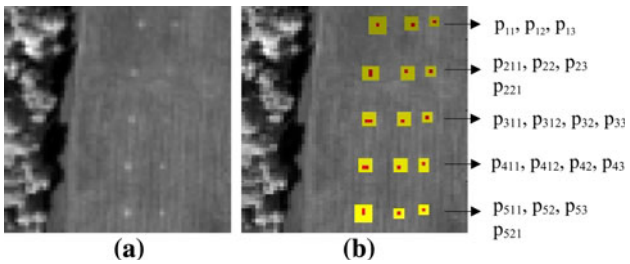


Fig. 15 **a** A HYDICE panel scene which contains 15 panels; **b** Ground truth map of spatial locations of the 15 panels

vapor absorption bands: bands 101–112 and bands 137–153 were removed. Therefore, a total of 169 bands were used in experiments. The spatial resolution and spectral resolution of this image scene are 1.56 m and 10 nm, respectively.

Within the scene in Fig. 15a 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 $i = 1, 2,\ldots,5$, there are three panels painted by the same paint but with three different sizes. The sizes of the panels in the first, second and third columns are 3 m $\times$ 3 m and 2 m $\times$ 2 m and 1 m $\times$ 1 m, respectively. Since the size of the panels in the third column is 1 m $\times$ 1 m, they cannot be seen visually from Fig. 15a due to the fact that its size is less than the 1.56 m pixel resolution. For each column $j = 1, 2, 3$, the 5 panels 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 3 (row 5) may also have effect on detection of panels in row 2 (row 4). The 1.56 m-spatial resolution of the image scene suggests that most of the 15 panels are one pixel in size except that the panels in the 1st column with the 2nd, 3rd, 4th, 5th rows which are two-pixel panels, denoted by p_{211}, p_{221}, p_{311}, p_{312}, p_{411}, p_{412}, p_{511}, p_{521}. Figure 15b 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. For our experiments for this scene, VD was

estimated to be 9 with the false alarm fixed at probabilities $P_F = 10^{-3}, 10^{-4}$.

First, we demonstrate the utility of the RT IN-FINDR which implemented the SM N-FINDR in real time processing. Since the RT IN-FINDR can be implemented as the RT multiple-pass SQ N-FINDR, Fig. 16a–d illustrates a progressive real time processing of one single pass (i.e., 1st pass) executed by the RT SQ N-FINDR where three endmembers were already extracted at the end of the pass in Fig. 16d. The nine endmembers found in the first pass were then used as the initial endmembers of the second pass for the RT SQ N-FINDR with final nine endmembers shown in Fig. 16e. By comparing the nine endmembers obtained in Figs. 16d and 12e, the only difference between them was the seventh endmember extracted from the background. Since both sets of nine endmembers were not identical, a third pass is needed to run the RT SQ N-FINDR. The final nine endmember extracted at the end of the fourth pass were shown in Fig. 16g which were identical to the nine endmember extracted in the third pass in which case the RT IN-FINDR is terminated where the nine endmembers extracted in the fourth pass were the final desired endmembers. It should be noted that according to reports in [9] the best performance produced by the N-FINDR could only extract three endmembers from the 15-panel HYDICE scene in Fig. 15a rather than five endmembers as expected. The reason for this is because the material made for the panels in second and third rows was the same fabric and the two panel signatures, p_2 and p_3 used to specify panel pixels in these two rows were very similar and considered as the same endmember. As a result, only one endmember in the third row was extracted to represent these two panel signatures. Similarly, an endmember in the fifth row was extracted to represent the two panel signatures, p_4 and p_5 that were used to specify panel pixels in the fourth and fifth rows. It has also been shown in [4–6, 9, 15–17] that in order for the N-FINDR along with its variants to be able to extract five panels was to use the independent component analysis (ICA) to perform data dimensionality because the main strength of the ICA lies in blind source separation. As a consequence of using the ICA to perform dimensionality reduction for the N-FINDR, the ICA works exactly what it is designed to make the N-FINDR capable of finding five endmembers to specify all the five panel signatures, p_1, p_2, p_3, p_4 and p_5. However, implementing a dimensionality reduction as a pre-processing step such as ICA makes real time processing impossible. Nevertheless, when data dimensionality becomes necessary, the RT N-FINDR can be always implemented after data dimensionality as a follow-up real time endmember extraction processing algorithm as shown in Fig. 16.

The above example shown in Fig. 16 sheds light on how the inner and outer loops are executed by the RT IN-FINDR. Most importantly, it also shows that the RT SQ

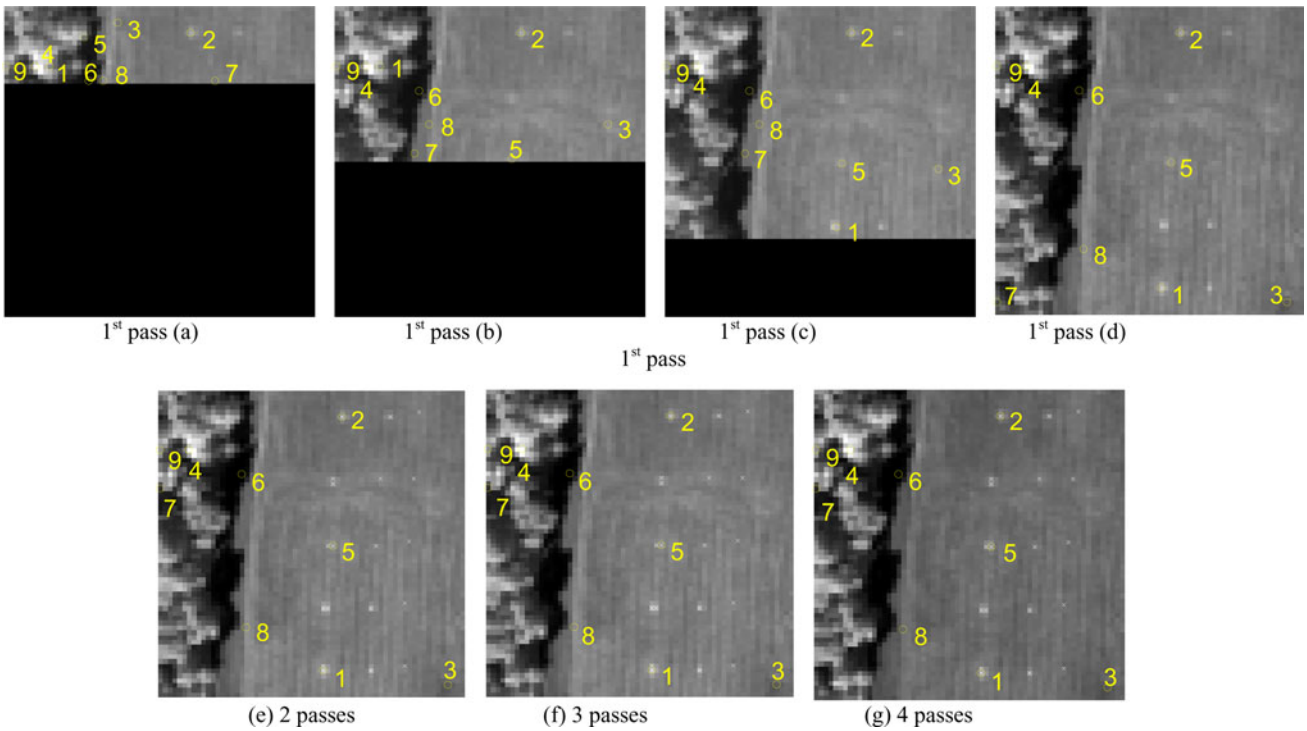


Fig. 16 Progress of nine endmembers extracted by RT IN-FINDR: first pass of RT SQ N-FINDR (**a**–**d**); Results of passes increased by one, **e** after two passes (**f**) after three passes (**g**) after four passes

N-FINDR which only implements the inner loop is sufficiently enough to be used for endmember extraction because it already extracted the three endmembers after its first pass in Fig. 16a–d. This evidence further provides the utility of 9-pass Circular N-FINDR and 9-pass SC N-FINDR. Figures 17, 18 show the nine endmembers extracted by the RT 9-pass Circular N-FINDR and RT 9-pass SC N-FINDR where the RT Circular N-FINDR actually completed its process after seven passes. In other words, if the RT IN-FINDR was implemented as RT multiple-pass circular N-FINDR, it would have been terminated after seven passes which were actually three more passes than only four passes required by the IN-FINDR implemented as the RT multiple-pass SQ N-FINDR. This conclusion was consistent with that made for the scenarios TI and TE. These experiments may suggest that the RT Multiple SQ N-FINDR would actually perform better than the RT multiple-pass circular N-FINDR in real applications in terms of fewer passes. However, from a computational point of view, the RT multiple-pass circular N-FINDR was still the best even though it required more passes to complete its process as shown in Table 6. These conclusions also confirmed by the following Cuprite data experiments.

In order to compare computational efficiency, Table 6 documents the computing times of the five real time processing N-FINDR algorithms, RT IN-FINDR implemented as multiple-pass SQ N-FINDR, RT IN-FINDR implemented as multiple-pass circular N-FINDR, RT SQ N-FINDR, RT 9-pass Circular N-FINDR and RT 9-pass SC N-FINDR along with SGA and VCA.

As shown in the table, the VCA had the best time since it only performs orthogonal projections and did not compute simplex volumes which require significant amount of computing time. However, among all simplex volume computed-based algorithms the RT 9-pass SC N-FINDR yielded the best computing time and the RT multiple-pass SQ N-FINDR was the slowest followed by the second slowest, the non-real time SGA. Therefore, by taking into account both the computational cost in Table 6 and the results in Figs. 16, 17, 18 for real-time processing, the RT 9-pass multiple-pass circular N-FINDR was still the best RT N-FNDR algorithm.

Table 6 Computing times of RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT 9-pass Circular N-FINDR and RT 9-pass SC N-FINDR

	Computing time (s)
RT multiple-pass SQ N-FINDR implemented as RT IN-FINDR	16.2234 (4 passes)
RT multiple-pass circular N-FINDR implemented as RT IN-FINDR	3.1846 (7 passes)
RT SQ N-FINDR	4.0463
RT 9-pass Circular N-FINDR	4.1655
RT 9-pass SC N-FINDR	4.0060
SGA	5.04
VCA	0.5

5.2 Cuprite data

The Cuprite shown in Fig. 6a was used to design synthetic image experiments conducted in Sect. 4. This section we perform experiments based on this real image scene. The VD estimated for this scene was 22 based on the false alarm fixed at probability $P_F = 10^{-4}$. Similar experiments done for HYDICE experiments were also performed for this Cuprite data. Figure 19a–c shows real-time processing of progressive results of the RT IN-FINDR with completion of first pass in Fig. 19d. Since the RT IN-FINDR can be implemented as a multiple-pass RT SQ N-FINDR, the SQ N-FINDR was re-run again using the final 22 endmembers found in the 1st pass as its new initial endmembers and result is shown in Fig. 19e. If two sets of the final 22 endmembers produced in the two consecutive passes do not agree, the SQ N-FINDR is repeated over and over again to extract another new set of 22 endmembers in the 3rd pass in Fig. 19f, 4th pass in Fig. 19g until it completed 8 passes where the two sets of 22 endmembers extracted in the 7th and 8th passes were identical shown in Fig. 19j–k and the algorithm was terminated. Since the pixels extracted by the algorithm were generally not identical to ground truth pixels, a spectral similarity measure such as spectral angle mapper (SAM) was used for endmember identification. The number within parentheses under each sub-figure after Fig. 19d indicates the number of extracted materials. The results show that the RT 1-pass SQ N-FINDR could successfully extract five pixels identified as four mineral signatures, in Fig. 19d, while the RT IN-FINDR implemented as the RT multiple-pass SQ N-FINDR could only extracted four mineral endmembers after 6 passes in Fig. 19i–k. Figure 20 also shows the results of the RT 22-pass Circular N-FINDR where all the five materials were extracted at 16th pass. This indicated that if the IN-FINDR implemented as multiple-pass circular N-FINDR, it would have been terminated after 17 passes. Similarly, Fig. 21 shows the results produced by the 22-pass SC N-FINDR which found the five materials after 14 passes.

As for computational cost, Table 7 tabulates computing times in seconds for the five real time N-FINDR processing algorithms, RT IN-FINDR implemented as multiple-pass SQ N-FINDR, RT IN-FINDR implemented as multiple-pass circular N-FINDR, RT 1-pass SQ N-FINDR, RT 22-pass Circular N-FINDR and RT 22-pass SC N-FINDR along with the SGA and VCA where the multiple-pass circular N-FINDR once again is the fastest algorithm among all the simplex volume computed-based algorithms.

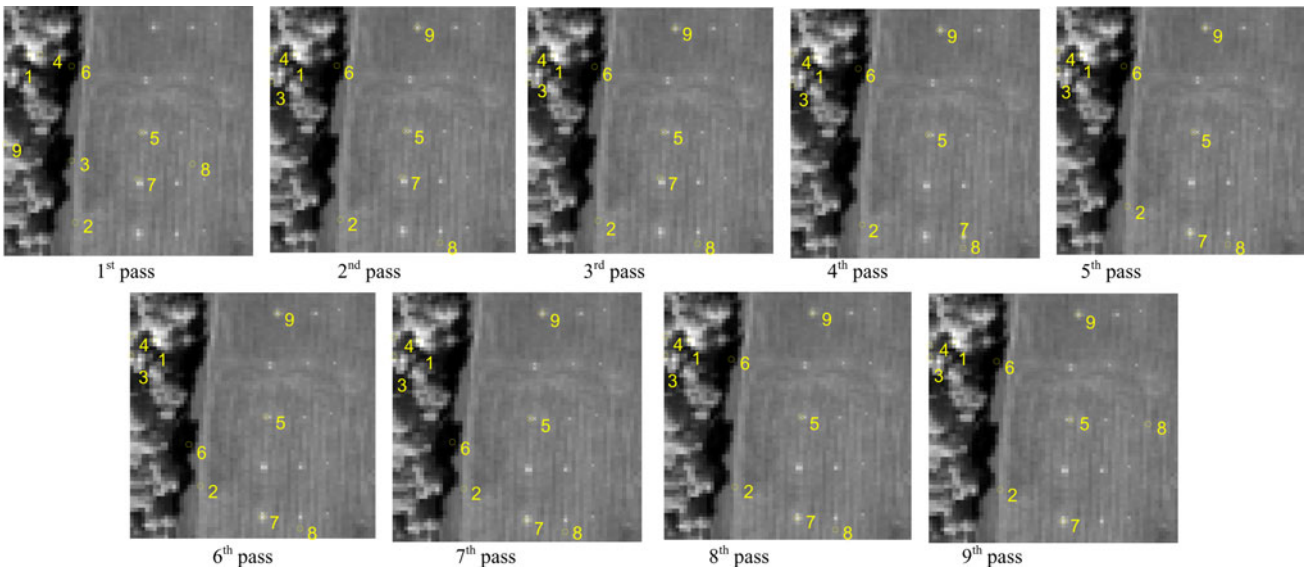


Fig. 17 Results of each pass of RT 9-Pass Circular N-FINDR

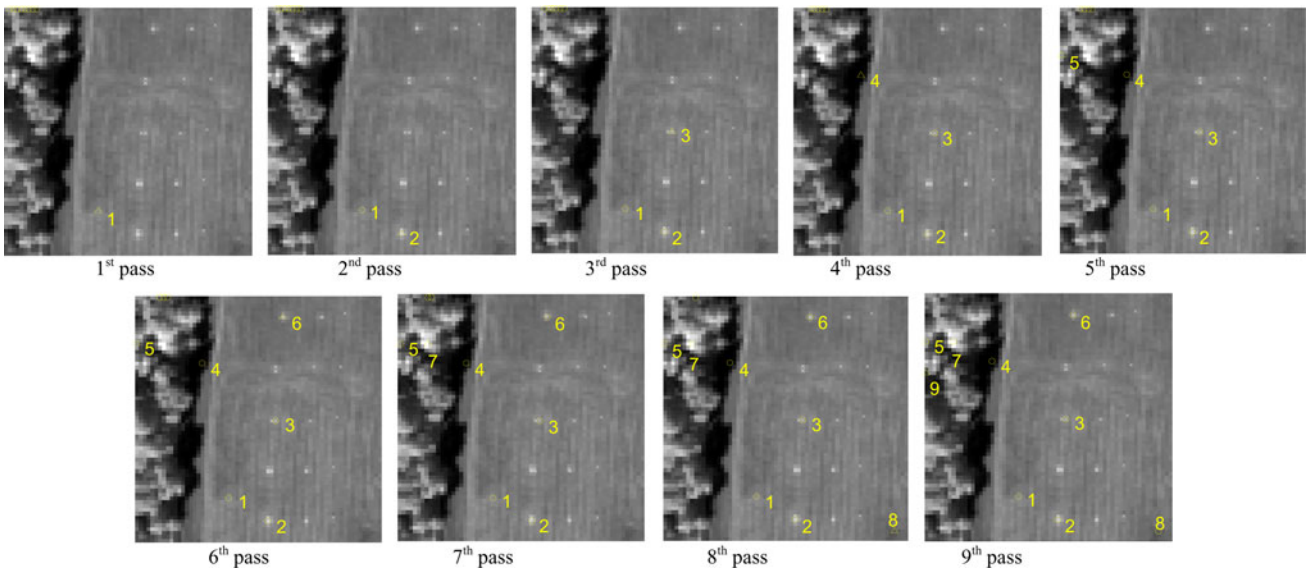


Fig. 18 Results of each pass of RT 9-Pass SC N-FINDR

Because the VCA only used inner products to perform orthogonal projection compared to the computation of matrix determinants required by N-FINDR, it required least computing time as expected. Nevertheless, its orthogonal projection-based performance was not as good as the simplex volume-computer based performance as shown in [9]. Most importantly, the current version of the VCA cannot be implemented in real time. In doing so, two major issues need to be addressed. One is that the VCA uses a random Gaussian variable to produce an initial endmember to generate a new endmember and such a Gaussian process cannot be implemented in real time. It seems that this issue can be resolved by using the first incoming data sample vector as its initial endmember as the way used in our RT N-FINDR. The other is how to implement the VCA in real-time processing as the number of endmembers increases. This is an interesting problem for a further investigation.

In addition to the computational costs tabulated in Table 7 the results in Figs. 19, 20, 21 are further used for performance evaluation, the winner for the best RT N-FINDR was the multiple-pass circular N-FINDR which also performed the best for the HYDICE data experiments.

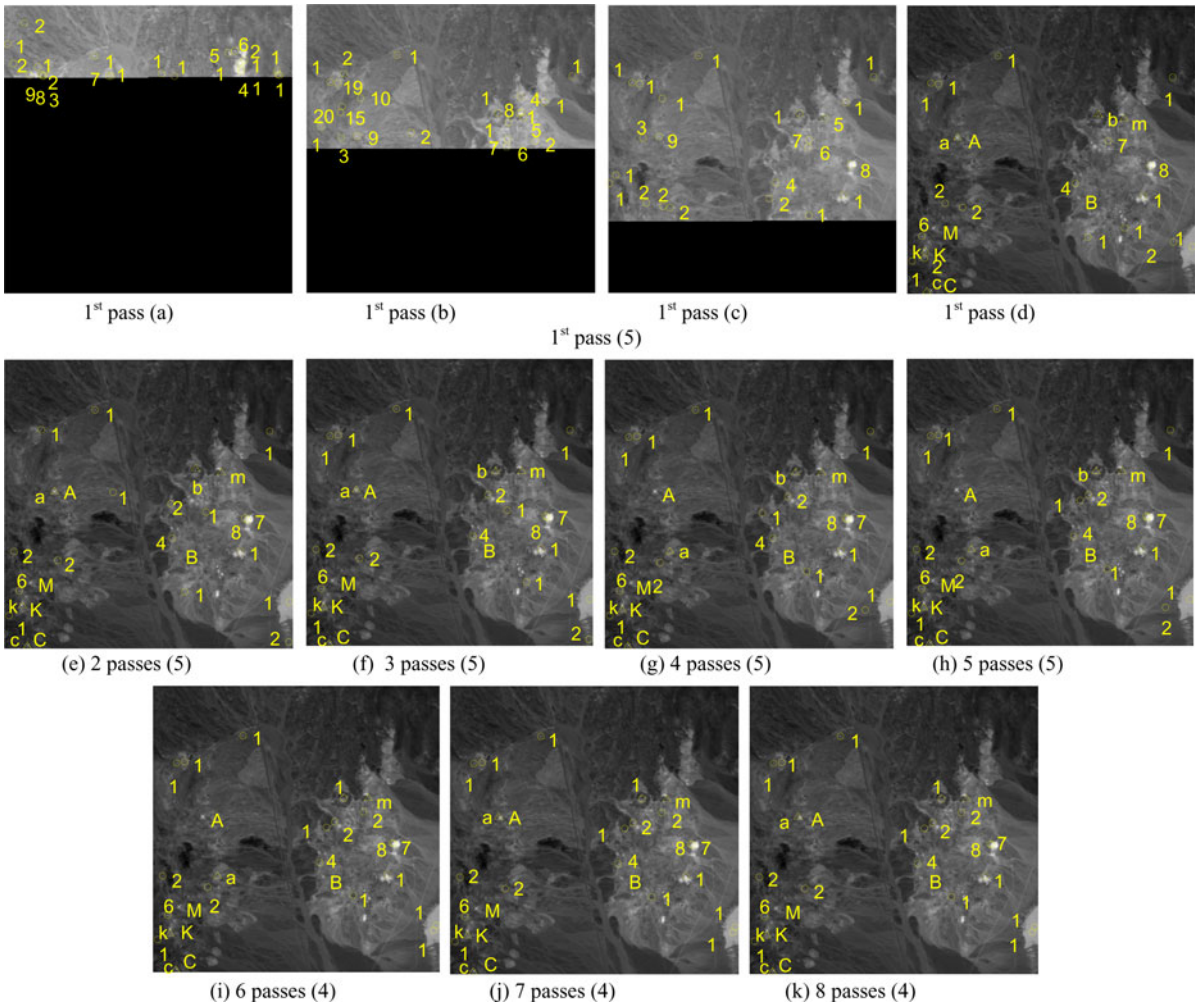


Fig. 19 Progress of 22 endmembers extracted by RT IN-FINDR: first pass of RT IN-FINDR (**a**–**d**); results of passes increased by one, **e** after 2 passes (**f**) after 3 passes (**g**) after 4 passes (**h**) after 5 passes (**i**) after 6 passes (**j**) after 7 passes (**k**) after 8 passes

6 Real-time demonstration

This section presents real-time demonstration of the three real-time N-FINDR processing algorithms, RT IN-FINDR, implemented as multiple-pass SQ N-FINDR, RT 9-pass Circular N-FINDR and RT 9-pass SC N-FINDR using the HYDICE image scene in Fig. 15a as a test data. Since the RT SC N-FINDR requires the least amount of computing time according to Table 6, it was used as a benchmark for compariosn. Figure 22a–c shows progressive processes of RT multiple-pass SQ N-FINDR, RT 9-pass Circular N-FINDR and RT 9-pass SC N-FINDR where each of 9 passes completed by the RT SC N-FINDR was used as a baseline for comparison.

For example, in the first column in Fig. 21 shows the first pass progressive processing by the RT Circular N-FINDR and the RT SQ N-FINDR after the completion of the first pass by the SC N-FINDR. From Table 4, the RT multiple-pass SQ N-FINDR required approximate 16.22 s to complete its process with 4 passes executed by the RT SQ N-FINDR, each of 4 passes required about 4 s. This is clearly shown in Fig. 21 where after the RT SC N-FINDR completed its 9 passes, the RT 9-Pass Circular N-FINDR was nearly done, while the RT IN-FINDR was just about to complete its first pass.

7 Analysis of comparative performance

This section compares the relative performance in endmember extraction among five endmember extraction algorithms without real-time processing, IN-FINDR, SQ N-FINDR, SC N-FINDR, SGA, VCA and five real time processing versions of N-FINDR, RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT Circular N-FINDR and RT SC N-FINDR. Table 8 tallies endmembers extracted by individual algorithms where the endmembers are tabulated in the order that endmembers

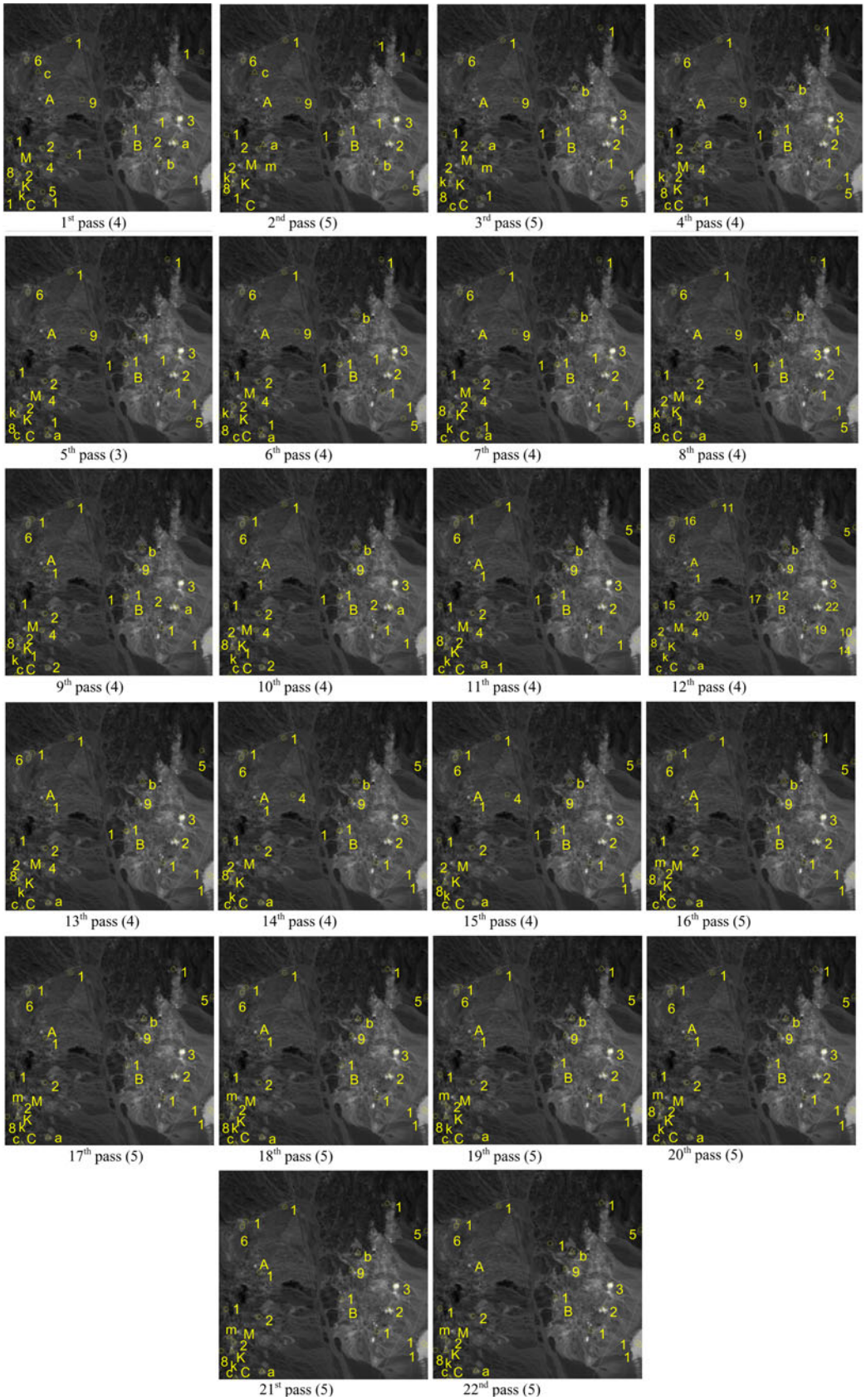


Fig. 20 Result of each pass of RT 22-Pass Circular N-FINDR

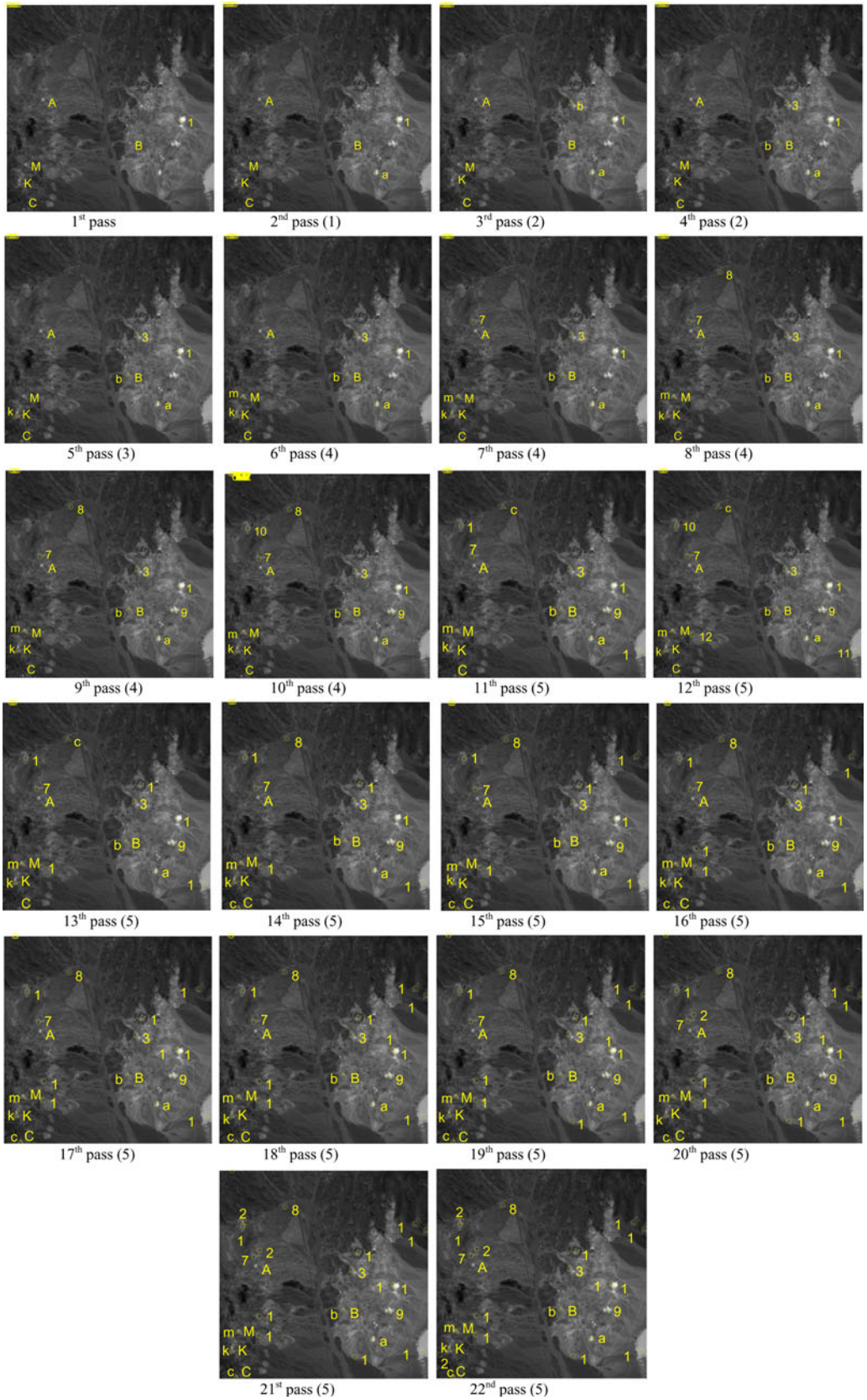


Fig. 21 Result of each pass of RT 22-Pass SC N-FINDR

Table 7 Computing times of RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT Circular N-FINDR and RT SC N-FINDR

Algorithms	Computing time (s)
RT multiple-pass SQ N-FINDR implemented as RT IN-FINDR	7,742.2 (8 passes)
RT multiple-pass circular N-FINDR implemented as RT IN-FINDR	737.6612 (17 passes)
RT 1-pass SQ N-FINDR	956.5172
RT 22-Pass Circular N-FINDR	954.0581
RT22-Pass SC N-FINDR	933.0949
SGA	947.94
VCA	9.38

were extracted in a sequential order except those extracted by IN FINDR.

It should be noted that the results by RT multiple-pass circular N-FINDR are not included in the table. This is because the number of passes run by RT multiple-pass circular N-FINDR is determined by VD. As a result, in many cases it requires more passes than it should. For example, in the TI scenario RT multiple-pass circular N-FINDR required five passes to complete its process. As a matter of fact, if it was implemented as IN-FINDR as shown in the table, the five mineral signatures could have been extracted in three passes, i.e., K, M, C (1st pass), BKMA (2nd pass), C (3rd pass). This is also true for the TE scenario, HYDICE and Cuprite data. Additionally, the results by 1-pass SQ N-FINDR are not included since they are already produced in the 1st pass by the Rt multiple-pass SQ N-FINDR implemented as IN-FINDR. According to Table 8 the real time processing N-FINDR algorithms performed as well as their counterparts without real time processing. Unlike the HYDICE experiments where the extracted panel pixels were exactly the same as true endmembers, the Cuprite data has many pixels whose spectral signatures are also specified by the five mineral signatures. In this case, the pixels extracted in Fig. 22 may not be the same pixels identified in Fig. 6(b). In this case, the SAM and mean squared error (MSE) were used as criteria to identify desired pixels that can be also used as endmember pixels. For this purpose, Table 9 documents the SAM values of endmembers extracted in Fig. 22 between their true spectral signatures along with their corresponding MSEs.

8 Conclusion

The N-FINDR has been widely used in endmember extraction for hyperspectral imagery. Despite the fact that the N-FINDR has shown great promise and potential in

Table 8 Comparative performance of IN-FINDR, SQ N-FINDR, SC N-FINDR, RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT Circular N-FINDR and RT SC N-FINDR in terms of endmember extraction

Algorithms	TI (VD = 5)	TE (VD = 5)	HYDICE (VD = 9)	Cuprite (VD = 22)
IN-FINDR	K, A, B, M, C	M, A, B, K, C	p_{521}, p_{312}, p_{11}	A, B, C, K, M
SQ N-FINDR	K, M, C, B	B, C, K, M	p_{312}, p_{521}	A, B, C, K
SC N-FINDR	A, K, M, B, C	A, K, M, B, C	p_{521}, p_{312}	A, B, C, K, M
SGA	A, M, B, K, C	K, C, A, B, M	p_{521}, p_{312}, p_{11}	A, B, C, K, M
VCA	B, K, C, A	A, C, K, B	p_{312}, p_{521}	A, C, M
RT multiple-pass SQ N-FINDR implemented as RT IN-FINDR	B, A, M, K (1th pass) C (2nd pass) (completed in 3 passes)	B, K, M (1st pass) B, C (2nd pass) (completed in 3 passes)	p_{512}, p_{312}, p_{11} (1st pass) (completed in 4 passes)	A, B, C, K, M (1st pass) (completed in 8 passes)
RT multiple-pass circular N-FINDR implemented as RT IN-FINDR	K, M, C (1st pass), B, K, M, A (2nd pass), C (3rd pass) (completed in 4 passes)	K, B, M (1st pass) K, A, M, B, C (2nd pass) (completed in 3 passes)	p_{312}(1st pass), p_{11} (2nd pass), p_{521} (5th pass) (completed in 6 passes)	A, B, C, K (1st pass), M(2nd pass) (completed in 10 passes)
RT Mutiple-Pass SC N-FINDR	A, K, M, B, C (completed in 5 passes)	A, K, B, M, C (completed in 5 passes)	p_{512} (2nd pass), p_{312}(3rd pass), p_{11}(6th pass) (completed in 9 passes)	A (2nd pass), B (3rd pass), K (5th pass), M (6th pass), C (11th pass) (completed in 22 passes)

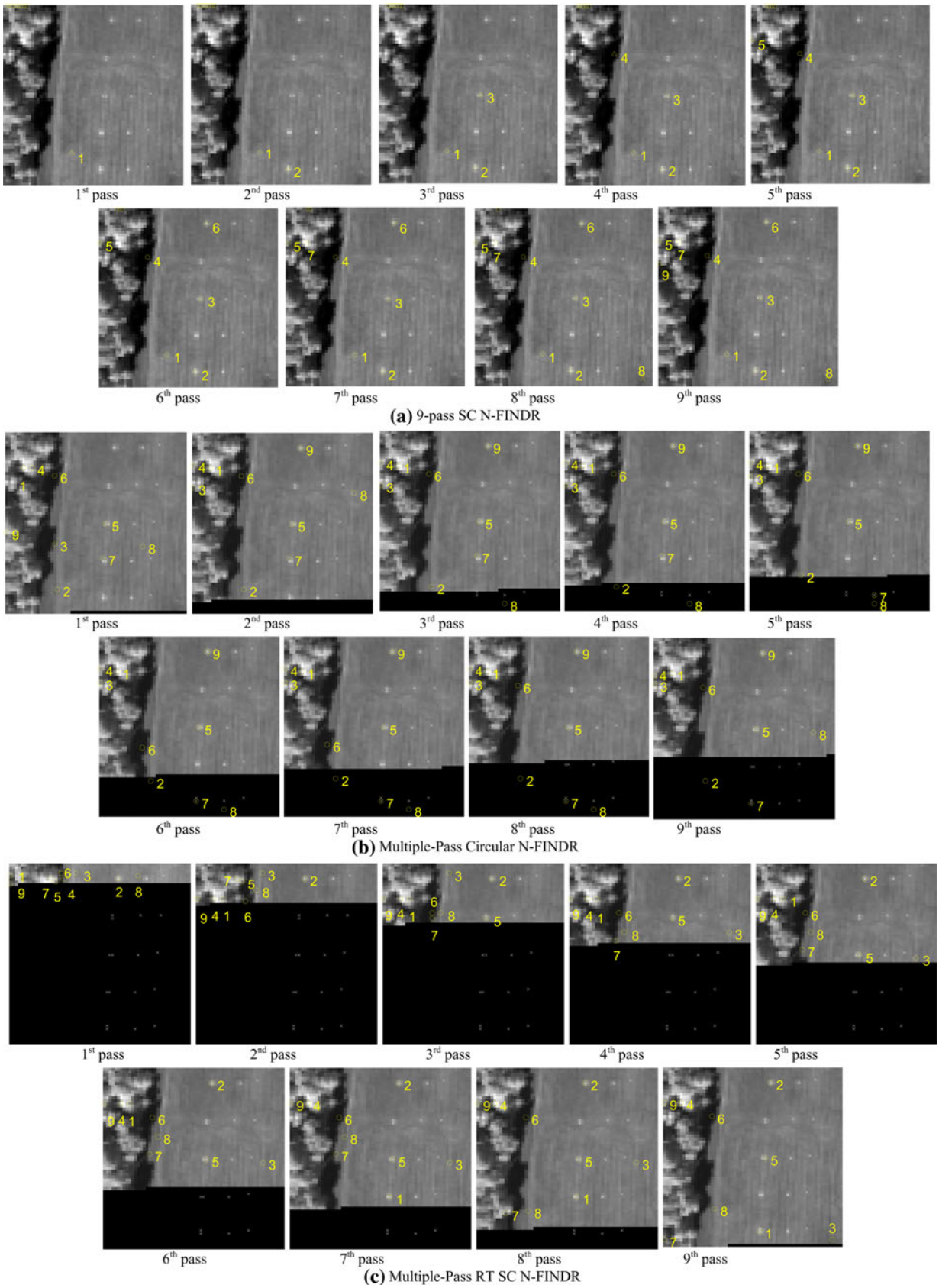


Fig. 22 Progressive process of RT IN-FINDR/multiple-pass RT SC N-FINDR, RT 9-pass Circular N-FINDR and RT 9-pass SC N-FINDR

Table 9 Comparative performance of IN-FINDR, SQ N-FINDR, SC N-FINDR, RT multiple-pass SQ N-FINDR, RT multiple-pass circular N-FINDR, RT Circular N-FINDR and RT SC N-FINDR in terms of SAM and MSE

	IN-FINDR	SQ N-FINDR	SC N-FINDR	SGA	VCA	RT multiple-pass SQ N-FINDR implemented as RT IN-FINDR	RT multiple-pass circular N-FINDR implemented as RT IN-FINDR	RT mutiple-pass SC N-FINDR
SAM								
A	0.0755	0.0489	0.0172	0.0167	0.0679	0	0.0717	0.0943
B	0.0671	0.0726	0.0750	0.0334		0.0730	0.0750	0.0383
C	0.0362	0.0516	0.0516	0.0516	0.0374	0.0362	0.0516	0.0516
K	0.0342	0.0300	0.0341	0.0613		0.0342	0.0348	0.03
M	0.0706		0.0706	0	0.0710	0.0692	0.0809	0.0264
MSE								
A	1.0512×10^6	0.3141×10^6	0.1161×10^5	0.1247×10^5	0.2921×10^6	0	3.8144×10^5	5.7110×10^6
B	0.2836×10^6	0.1181×10^6	2.5406×10^5	0.4635×10^5		0.8759×10^5	2.5406×10^5	0.3439×10^6
C	0.0831×10^6	1.5355×10^6	1.1085×10^5	1.1085×10^5	0.0145×10^6	0.8312×10^5	1.1085×10^5	0.1108×10^6
K	0.1035×10^6	0.4934×10^6	0.6794×10^5	0.7204×10^5		0.7090×10^5	0.3878×10^5	0.0354×10^6
M	0.1403×10^6		1.4034×10^5	0	1.2149×10^6	2.6450×10^5	1.5742×10^5	0.0200×10^6

data exploitation, its computational un-implementability has prevented it from being considered in many practical applications. This paper looks into the design rationale of the N-FINDR and re-invents the wheel by re-deriving the N-FINDR as a real time iterative N-FINDR (RT IN-FINDR) which can be implemented as a real-time processing algorithm. The need of real time processing was also reported in [18] where the N-FINDR was included as an endmember extraction algorithm for spectral anomaly detection. Unfortunately, no detail of its real time implementation was documented in [18]. This paper materializes this idea by breaking up the N-FINDR into two executable real time processing loops, an inner loop called real time sequential N-FINDR (RT SQ N-FINDR) which finds final endmembers from a specific initial condition and an outer loop, called pass which re-runs the inner loop to eliminate the inner loop's dependency of initial conditions. As a result, the RT IN-FINDR can be implemented as a real time multiple-pass SQ N-FINDR. Moreover, to further reduce computational complexity of the RT SQ N-FINDR in real time implementation, two new versions, referred to as real time Circular N-FINDR (RT Circular N-FINDR) and real time successive N-FINDR (RT SC N-FNDR) are also developed as alternatives to replace the RT SQ N-FINDR implemented in the IN-FINDR. There are several advantages and benefits resulting from implementing the N-FINDR as real time processing algorithms. Most important and foremost is elimination of random initial conditions commonly used in endmember extraction algorithms such as PPI, N-FINDR, VCA that generally result in inconsistent final extracted endmembers. Second, there is no need of data dimensionality reduction which is generally required by many endmember extraction algorithms, e.g., PPI, N-FINDR, VCA, etc. Third, the significant reduction of computational complexity makes real-time N-FINDR processing algorithms attractive in real applications. Finally, the nature in algorithmic structure in processing data sample vectors sequentially, circularly and successively facilitates the hardware design such as field programmable gate array (FPGA) for chip design. By concluding this paper, one final comment on implementation of real-time N-FINDR processing algorithms is noteworthy. Except the PPI most endmember extraction algorithms developed in the literature need to know the value of the p, the number of endmembers prior to its processing. This is also true for our proposed real-time N-FINDR processing algorithms. In order to resolve this issue, the algorithms use the HFC/NWHFC method in [7, 8, 13] to estimate the VD and set VD $= p$. Since the HFC/NWHFC requires calculation of sample correlation and covariance matrices which must know the entire data before the calculation can take place. In this case, the proposed real-time N-FINDR processing algorithms only

need one more pass to accomplish the task. The sample correlation and covariance matrices can be calculated and updated in a causal manner by including incoming data sample vectors in which case no storage is required to store the entire complete data. Moreover, calculating the inverse of these matrices can be also updated causally in real time by the well-known Woodburry identity formula and its FPGA implementation can be also found in [19].

Acknowledgment C.-I Chang would like to thank for support received from the National Science Council in Taiwan under NSC 98-2811-E-005-024 and NSC 98-2221-E-005-096.

References

1. Winter, M. E.: N-finder: an algorithm for fast autonomous spectral endmember determination in hyperspectral data. In: Image Spectrometry V, Proceedings of SPIE, vol. 3753, pp. 266–277 (1999)
2. Boardman, J.W.: Geometric mixture analysis of imaging spectrometery data. Proc. Int. Geosci. Remote Sens. Symp. **4**, 2369–2371 (1994)
3. Chaudhry, F., Wu, C., Liu, W., Chang, C.-I., Plaza, A.: Pixel purity index-based algorithms for endmember extraction from hyperspectral imagery, chap. 3. In: Chang, C.-I. (ed.) Recent Advances in Hyperspectral Signal and Image Processing. Research Signpost, Trivandrum (2006)
4. Chang, C.-I., Plaza, A.: Fast iterative algorithm for implementation of pixel purity index. IEEE Geosci. Remote Sens. Lett. **3**(1), 63–67 (2006)
5. Plaza, A., Chang, C.-I.: Impact of initialization on design of endmember extraction algorithms. IEEE Trans. Geosci. Remote Sens. **44**(11), 3397–3407 (2006)
6. Chang, C.-I., Wu, C.-C.: Random pixel purity index algorithm. IEEE Trans. Geosci. Remote Sens. Lett. (to appear)
7. Chang, C.-I.: Hyperspectral Imaging: Techniques for Spectral Detection and Classification. Dordrecht: Kluwer Academic/Plenum Publishers (2003)
8. Chang, C.-I., Du, Q.: Estimation of number of spectrally distinct signal sources in hyperspectral imagery. IEEE Trans. Geosci. Remote Sens. **42**(3), 608–619 (2004)
9. Chang, C.-I., Wu, C., Liu, W., Ouyang, Y.C.: A growing method for simplex-based endmember extraction algorithms. IEEE Trans. Geosci. Remote Sens. **44**(10), 2804–2819 (2006)
10. Reed, M., Simons, B. Functional Analysis, Academic Press, New York (1972)
11. http://speclab.cr.usgs.gov/cuprite.html
12. Harsanyi, J.C., Chang, C.-I.: Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach. IEEE Trans. Geosci. Remote Sens. **32**(4), 779–785 (1994)
13. Harsanyi, J.C., Farrand, W., Chang, C.-I.: Detection of subpixel spectral signatures in hyperspectral image sequences. In: Annual Meeting, Proceedings of American Society of Photogrammetry and Remote Sensing, Reno, pp. 236–247 (1994)
14. Nascimento, J.M.P., Dias, J.M.: Vertex component analysis: a fast algorithm to unmix hyperspectral data. IEEE Trans. Geosci. Remote Sens. **43**(4), 898–910 (2005)
15. Wang, J., Chang, C.-I.: Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis. IEEE Trans. Geosci. Remote Sens. **44**(6), 1586–1600 (2006)
16. Wu, C.C., Lo, C.S., Chang, C.-I.: Improved process for use of a simplex growing algorithm for endmember extraction. IEEE Trans. Geosci. Remote Sens. Lett. **6**(3), 523–527 (2009)
17. Wang, J., Chang, C.-I.: Applications of independent component analysis in endmember extraction and abundance quantification for hyperspectral imagery. IEEE Trans. Geosci Remote Sens. **44**(9), 2601–2616
18. Winter, E.M., Schlangen, M.J., Hill, A.B., Simi, C.G., Winter, Winter, M.E.: Tradeoffs for real-time hyperspectral analysis. In: Proceedings of SPIE, vol. 4725, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery VIII, pp. 366–371 (2002)
19. Wang, J., Chang, C.-I.: FPGA design for real-time implementation of hyperspectral target detection and classification algorithms. In: Plaza, A., Chang, C.-I., (eds.) High-Performance Computing in Remote Sensing, chap. 16, pp. 379–395, Boca Raton: CRC Press (2007)

Author Biographies

Chao-Cheng Wu received the BS degree in electrical engineering from the Tamkang University, Taipei, Taiwan in 2002 and the MS and PhD degree in electrical engineering from the University of Maryland, Baltimore County in 2006 and 2009. His research interests include endmember extraction, pattern recognition, and hyperspectral image processing.

Hsian-Min Chen received his BS, MS degrees from Huafan University, Taipei, Taiwan in 1999, 2001 and PhD in Electrical Engineering from National Chung Hsing University, Taichung, Taiwan, 2007. He is currently with the Department of Radiology, China Medical University Hospital, Taichung, Taiwan, ROC. His research interests include digital image processing, and biomedical image processing.

Chein-I Chang received his BS degree from Soochow University, Taipei, Taiwan, MS degree from the Institute of Mathematics at National Tsing Hua University, Hsinchu, Taiwan and MA degree from the State University of New York at Stony Brook, all in mathematics. He also received his MS, MSEE degrees from the University of Illinois at Urbana-Champaign and PhD degree in electrical engineering from the University of Maryland, College Park. Dr. Chang has been with the University of Maryland, Baltimore County (UMBC) since 1987 and is currently a professor in the Department of Computer Science and Electrical Engineering. He was a visiting research specialist in the Institute of Information Engineering at the National Cheng Kung University, Tainan, Taiwan, from 1994–1995. He received an NRC (National Research Council) senior research associateship award from 2002–2003 sponsored by the US Army Soldier and Biological Chemical Command, Edgewood Chemical and Biological Center, Aberdeen Proving Ground, Maryland. Additionally, Dr. Chang was a distinguished lecturer chair at the National Chung Hsing University sponsored by the Department of Education in Taiwan, ROC from 2005 to 2006 and is a chair professor in Department of Electrical Engineering, National Chung Hsing University from 2006–2012, Taichung, Taiwan, ROC and is currently a distinguished visiting fellow/fellow professor sponsored by National Science Council in Taiwan from 2009–2010. He was also a keynote speaker for the 2008 International Symposium on Spectral Sensing Research (ISSSR) in 2008 and will be a plenary spekaer for SPIE Optics + Applications, Remote Sensing Symposium, 2009. He has four patents and several pending on hyperspectral image processing. He was the guest editor of a special issue of the Journal of High Speed Networks on Telemedicine and Applications (April 2000) and

co-guest editor of another special issue of the same journal on Broadband Multimedia Sensor Networks in Healthcare Applications, April 2007. His is also co-guest editor of a special issue on High Performance Computing of Hyperspectral Imaging for International Journal of High Performance Computing Applications, December 2007 and special issue on Signal Processing and System Design in Health Care Applications for EURASIP Journal on Advanced in Signal Processing, 2009. Dr. Chang has authored a book, Hyperspectral Imaging: Techniques for Spectral Detection and Classification published by Kluwer Academic Publishers in 2003 and edited two books, Recent Advances in Hyperspectral Signal and Image Processing, Trivandrum, Kerala: Research Signpost, Trasworld Research Network, India, 2006 and Hyperspectral Data Exploitation: Theory and Applications, John Wiley & Sons, 2007 and co-edited with A. Plaza a book on High Performance Computing in Remote Sensing, CRC Press, 2007. He is currently working on a second book, Hyperspectral Data Processing: Signal Processing Algorithm Design and Analysis, John Wiley & Sons, 2010 and a third book, Real Time Hyperspectral Image Processing: Algorithm Architecture and Implementation, Springer-Verlag, 2012. Dr. Chang was an Associate Editor in the area of hyperspectral signal processing for IEEE Transaction on Geoscience and Remote Sensing 2001-2007 and a Fellow of IEEE and SPIE and a member of Phi Kappa Phi and Eta Kappa Nu.