ATLANTIS PRESS

Advances in Intelligent Systems Research (AISR), volume 160

3rd International Conference on Modelling, Simulation and Applied Mathematics (MSAM 2018)

A Fast Method for Hyperspectral Target Detection with Matrix Inverse Updating and Downdating

Xin Wu[1,*], Mizhen Wang[1] and Chen Yang[2]

[1]School of Physics and Optoelectronic Engineering, Xidian University, Shaanxi, Xi'an 710071, China

[2]Faculty of Automation and Information Engineering, Xi'an University of Technology, Shaanxi, Xi'an 710048, China

[*]Corresponding author

***Abstract*—The unsupervised nonnegative constrained least squares (UNCLS) method is a well-known method for hyperspectral subpixel target detection. Depending on the complexity and dimensionality of the hyperspectral data, it may be computationally expensive for large matrix inversion with the UNCLS method. This paper proposed a fast UNCLS algorithm to improve its computational performance. Matrix inverse updating and downdating theory is implemented to speed up the iterative least square estimation in the UNCLS method. Experiments on both simulated and real hyperspectral images show that the proposed fast UNCLS can efficiently detect subpixel targets with precision guaranteed. We have demonstrated that our proposed method improves performance of the UNCLS method, which achieves a >10x speedup for both the simulated and real scenes when large number of targets are detected.**

Keywords—remote sensing; matrix inverse; fast least squares; subpixel target detection

I. INTRODUCTION

Mixing the spectrum of a subpixel target with the spectra of the background results in a pixel with a combined spectral signature. Subpixel target detection in hyperspectral images involves with a separation of the constituent elements in the combined spectral signatures [1–4]. Several classical subpixel target detection algorithms have proposed, which are summarized in the book by Chang [5]. The unsupervised nonnegative constrained least squares (UNCLS) algorithm is one of the popularly-used methods. The UNCLS method iteratively applies the nonnegative constrained least squares (NCLS) algorithm with the signature matrix to estimate the abundance fractions [5]. It can automatically extract the target signatures with a stable result no matter how many times of execution without a priori [6]. Nevertheless, depending on the complexity and dimensionality of hyperspectral data, the UNCLS algorithm may be time-consuming involving with big matrix inversion for the covariance of the signature matrix.

Acceleration of hyperspectral image processing methods has been an active topic in recent years. With the rapid development of remote sensing techniques, more efficient computation methods are necessary for the unprecedented size of hyperspectral images. Researchers have proposed fast algorithms with different approaches [7–9], including adoption of hardware acceleration [10–12], in order to reduce the computational costs. Typically, Chang et al. [7] proposed a fast iterative method for the pixel purity index (PPI). Guerra et al. [8] applied the orthogonal subspace projection (OSP) method to accelerate the N-FINDR algorithm. López et al. [9] selected a subset of pixels to extract endmembers in a reduced time. In addition, implementations of parallel algorithms with high performance hardware are recently popular in hyperspectral image processing. PPI algorithm has been accelerated by a GPU-based massively parallel version for real-time applications [13]. The automatic target detection and classification algorithm (ATDCA) and the hyperspectral signal subspace identification by minimum error (HySime) method have also been implemented in parallel with GPUs [14,15]. Besides, field programmable gate arrays (FPGAs) and coprocessors (Intel Xeon Phi) are promising to adopted in this domain [11,12,16]. However, there is few acceleration of the UNCLS algorithm for hyperspectral subpixel detection.

In this paper, we propose a fast UNCLS method to accelerate the original one. Firstly, we incorporated the matrix inverse updating theory [17] to accelerate the inverse of the signature covariance matrix in estimating the abundance fractions. Secondly, we derived the matrix downdating scheme to invert the steering matrix in the NCLS algorithm. Given that we have the inverse of a matrix $R \in i$, it is possible that the inverse of the matrix $\mathbf{R}^{-1}$ can be obtained from $\mathbf{R}$ by appending a row or column ("update" or "upscale") or deleting a row or column ("downscale" or "downscale"). In this situation, it is much more efficient to update or downdate inverse of $\mathbf{R}$ other than to calculate $\mathbf{R}^{-1}$ from scratch. Theoretical time complexity of the proposed method has been made. Experiments have been conducted on a simulated hyperspectral image (HSI) based on the Cuprite AVIRIS data and a real HSI collected by NASA AVIRIS sensor over the World Trade Center area in New York City. Experimental results indicate that the proposed fast UNCLS method can achieve a better performance compared to the original algorithm with correct results guaranteed.

The rest of this paper is organized as follows. Section 2 briefly describes the UNCLS method and then describes the proposed fast UNCLS method and then our implementation in detail. Section 3 shows the experimental results. Section 4 concludes the paper with some remarks.

II. PROPOSED APPROACH

A. The UNCLS Method Review

The UNCLS is a popular method for hyperspectral subpixel target detection. We briefly describe the UNCLS method here. More details can be found in the paper by Chang et al. [6]. The UNCLS method uses a linear mixture model, which assumes that $\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_p$ are p target signatures resident in a multispectral/hyperspectral image pixel vector $\mathbf{r}$.

$$\mathbf{r} = \mathbf{M}\alpha + \mathbf{n}, \tag{1}$$

where $\mathbf{n}$ denotes the noise from instruments or measurement. Denoting $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_p]$ an $L \times P$ target spectral signature matrix, the abundance column vector $\alpha = (\alpha_0, \alpha_1, ..., \alpha_p)^T$ can be estimated by a linear regression model as $\hat{\alpha}_{LS}$:

$$\hat{\alpha}_{LS} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{r} \tag{2}$$

To fulfill the non-negative constraint, $\forall j \in 1$, $p : \alpha_j > 0$ the method of Lagrange Multipliers is introduced to solve the optimization problem:

$$\text{Minimize LSE} = (\mathbf{M}\alpha - \mathbf{r})^{\mathrm{T}}(\mathbf{M}\alpha - \mathbf{r}) \quad \text{,subject to } \alpha \geq 0 \tag{3}$$

Equation (4) is adopted to estimate the non-negative abundance:

$$\hat{\alpha}_{NCLS} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{r} - (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\lambda, \tag{4}$$

where λ is the multiplier vector [18]. $\hat{\alpha}_{NCLS}$ is the abundance vector estimated by the NCLS algorithm. The NCLS algorithm is one of the key steps of the UNCLS method involving with matrix inversion $(\mathbf{M}^T\mathbf{M})^{-1}$ recursively. In the NCLS algorithm, two sets of indicators are introduced to decide which row/column to be deleted according to the Kuhn-Tucker conditions [18,19]. The passive set is initialized as an empty vector, and the active set is initialized with the indices from 1 to p. Indices in the passive and active sets will be changed according to the Lagrange multiplier vector λ, the complete scheme of the NCLS algorithm can be referred to reference [6].

Since the NCLS algorithm requires a complete knowledge of target signatures, the UNCLS method iterates the NCLS algorithm to achieve an unsupervised version. The UNCLS algorithm have two major components, namely two major loops. The first one is the implementation of the NCLS with an inner loop to form a steering matrix $\Phi_{\alpha}^{(k)}$. Each steering matrix has to be inversed for next calculation until the estimated abundance $\hat{\alpha}_{NCLS}$ fulfils the non-negative constrain. As the NCLS algorithm describes, the steering matrix is updated by deleting some rows and columns from the matrix $(\mathbf{M}^T\mathbf{M})^{-1}$ and then inverse the newly-formed one. More detailed steps can be found in reference [6].

Step 1. Initial condition: Select ε to be a prescribed error threshold and let $t_0 = \arg\{\max_r[\mathbf{r}^T\mathbf{r}]\}$, where $\mathbf{r}$ is run over all image pixel vectors. Let $k = 0$.

Step 2. Let $k \leftarrow k+1$, and apply the NCLS algorithm with the signature matrix $\mathbf{M} = [t_0 t_1 \ldots t_{k-1}]$ to estimate the abundance fractions of $t_0, t_1, \ldots, t_{k-1}$, $\hat{\alpha}_0^{(k)}(\mathbf{r}), \hat{\alpha}_1^{(k)}(\mathbf{r}), \ldots, \hat{\alpha}_{k-1}^{(k)}(\mathbf{r})$.

Step 3. Find the LSE defined by

$$LSE^{(k-1)}(\mathbf{r}) = \left(\mathbf{r} - \left[\sum_{i=0}^{k-1}\hat{\alpha}_i^{(k)}(\mathbf{r})t_i\right]\right)^T \cdot \left(\mathbf{r} - \left[\sum_{i=0}^{k-1}\hat{\alpha}_i^{(k)}(\mathbf{r})t_i\right]\right)$$

Check the error if $LSE^{(k-1)}(\mathbf{r}) < \varepsilon$ for all $\mathbf{r}$. If it is, the algorithm stops, otherwise continue.

Step 4. Find $t_k = \arg\{\max_r LSE^{(k-1)}(\mathbf{r})\}$. Go to step 2.

FIGURE I. THE ORIGINAL UNCLS ALGORITHM (CHANG ET AL. [6])

The second component of the UNCLS is the "unsupervised" part. NCLS algorithm is supervised, which needs subpixel target candidates as input. To achieve an unsupervised algorithm, the UNCLS introduces a try and error procedure which is implemented by another loop. Figure 1 shows the original UNCLS algorithm. As seen in Figure 1, steps 2, 3 and 4 form an outer loop with k times iteration. The inverse of the matrix $(\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}$ is calculated in an inner loop by NCLS algorithm repeatedly. $\mathbf{M}$ is modified by adding one column on each newly found target signature, and a new matrix inverse is calculated. During this process, the signature matrix is test as candidate target signatures and duplicated information will be eliminated by the NCLS inner loop. The two loops lead to a low computational efficiency. Therefore, we proposed to apply matrix inverse updating and downdating method by reusing the post-calculated data to improve the computational performance. It should be noted that the UNCLS method has already taken consideration to the linearity of the modified signature matrix. Since our proposed method takes advantage of the UNCLS method, the linearity problem is also concerned. Our proposed fast UNCLS algorithm is described in the next section.

B. Proposed Fast UNCLS Method

We introduce matrix inverse updating and downdating theory to improve the UNCLS algorithm for a better computational performance. The comprehensive derivation of our fast UNCLS method is as follows. In the original UNCLS algorithm, each newly-detected target signature will be added to $\mathbf{M}$, the target signature matrix. In order to express the derivation clearly, we denote $\mathbf{R}_{\mathrm{k}} = \mathbf{M}_{\mathrm{k}}^{\mathrm{T}}\mathbf{M}_{\mathrm{k}}$ and its inverse as $\mathbf{R}_k^{-1}$, where the subscript k is the number of target

signatures and $\mathbf{e}_k$ is the k-th detected signature. Then $\mathbf{R}_k$ is expressed in Equation 5 and 6.

$$\mathbf{R}_k = \begin{bmatrix} \mathbf{M}_{k-1}^T \\ \mathbf{e}_k^T \end{bmatrix} [\mathbf{M}_{k-1} \quad \mathbf{e}_k], \tag{5}$$

$$\mathbf{R}_k = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c} & d \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{k-1}^T \mathbf{M}_{k-1} & \mathbf{M}_{k-1}^T \mathbf{e}_k \\ \mathbf{e}_k^T \mathbf{M}_{k-1} & \mathbf{e}_k^T \mathbf{e}_k \end{bmatrix}. \tag{6}$$

where A, b, c and d denote the block matrix elements. To solve the least squared estimation problem in Equation 3, matrix inverse updating is applied to the block-wise matrix inverse. The analytic block matrix inverse is given in Equation 7 and 8:

$$\mathbf{R}_k^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{c} & d \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{r}_{12} \\ \mathbf{r}_{21} & \mathrm{r}_{22} \end{bmatrix}, \tag{7}$$

$$\begin{aligned} \mathbf{R}_{11} &= \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{b}(d - \mathbf{c}\mathbf{A}^{-1}\mathbf{b})^{-1}\mathbf{c}\mathbf{A}^{-1}, \\ \mathbf{r}_{12} &= -\mathbf{A}^{-1}\mathbf{b}(d - \mathbf{c}\mathbf{A}^{-1}\mathbf{b})^{-1}, \\ \mathbf{r}_{21} &= -(d - \mathbf{c}\mathbf{A}^{-1}\mathbf{b})^{-1}\mathbf{c}\mathbf{A}^{-1}, \\ \mathrm{r}_{22} &= (d - \mathbf{c}\mathbf{A}^{-1}\mathbf{b})^{-1}. \end{aligned} \tag{8}$$

$\mathbf{A}^{-1}$ refers to the matrix inverse of the last iteration, which has already been calculated and can be reused. Therefore, we derived the corresponding formulas of our proposed method by substituting Equation 6 into Equation 8 to arrive at Equation 9 to 12.

$$\mathrm{r}_{22} = \frac{1}{\mathbf{e}_k^T\mathbf{e}_k - \mathbf{e}_k^T\mathbf{M}_{k-1}\mathbf{R}_{k-1}^{-1}\mathbf{M}_{k-1}^T\mathbf{e}_k}, \tag{9}$$

$$\mathbf{R}_{11} = \mathbf{R}_{k-1}^{-1} + \mathrm{r}_{22} \cdot \mathbf{R}_{k-1}^{-1}\mathbf{M}_{k-1}^T\mathbf{e}_k\mathbf{e}_k^T\mathbf{M}_{k-1}\mathbf{R}_{k-1}^{-1}, \tag{10}$$

$$\mathbf{r}_{12} = -\mathrm{r}_{22} \cdot \mathbf{R}_{k-1}^{-1}\mathbf{M}_{k-1}^T\mathbf{e}_k, \tag{11}$$

$$\mathbf{r}_{21} = \mathbf{r}_{12}^T. \tag{12}$$

Similar to the matrix inverse updating method, downdating method can be used to calculate $\mathbf{R}_{11}^{-1}$ according to the identity Equation 13.

$$\mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{b}(d - \mathbf{c}\mathbf{A}^{-1}\mathbf{b})^{-1}\mathbf{c}\mathbf{A}^{-1} = (\mathbf{A} - \mathbf{b}d^{-1}\mathbf{c})^{-1} \tag{13}$$

For instance, suppose that the last row and column is deleted from $\mathbf{R}_k^{-1}$. By inverting the first equation in Equation 8, we get:

$$\mathbf{R}_{11}^{-1} = \mathbf{A} - \mathbf{b}d^{-1}\mathbf{c}. \tag{14}$$

For block-wise matrix inverse, A and d must be square matrices and the sub-blocks of the b and c can be of arbitrary size [20,21], and yet for our fast UNCLS method b and c are vectors and d is a scalar. It is clear that r_{22} can be directly without a matrix inverse. Given the transpose relation between $\mathbf{r}_{21}$ and $\mathbf{r}_{12}$ as shown in Equations 11 and 12, we can also gain some improvement on the computational performance. Since the updating and downdating scheme has been derived above in Equations (8), (10) and (14), we modified the UNCLS algorithm and applied matrix inverse updating and downdating method to the fast UNCLS algorithm. The comprehensive algorithm of the proposed fast UNCLS method is shown in Figure 2. The pseudo code contains some symbols, which are used in the origin UNCLS algorithm by Chang [6]. According the pseudo code, we calculated the time complexity of the proposed method [22]. Firstly, let n be the number of pixels in one band of the HSI to be processed, which can be regarded as the scale of the UNCLS algorithm. As we applied the matrix inverse updating and downdating technique, the matrix inverse can be done outside the inner loop, thus the time complexity can be reduced from $T(n) = O(n^3)$ to $T(n) = O(n^2 + n\log_2 n)$. Theoretically, when the computing scale increases, the proposed algorithm can overcome the original one. However, the time complexity analysis is based on an abstract flow of the UNCLS algorithm, the computing performance evaluation with synthetic and real HSI are present in the next section.

Firstly, the HSI data is loaded to the memory and the number of subpixel targets is declared as initial condition. The first target signature candidate is selected by finding the maximum power through the whole HIS, then enter the inner loop with a minimum error threshold is set as 1.0e-24. When the number of the detected subpixel targets is greater than two, matrix inverse updating and downdating was applied to the inner loop of the original NCLS algorithm (In Figure 2, from Step 2 to Step 9). In the modified NCLS algorithm, downdating scheme is applied to build a steering matrix $\mathrm{F}_a^{(k)}$, all rows and columns specified in the negative indices are deleted from the matrix $(\mathbf{M}^T\mathbf{M})^{-1}$. To calculate inverse of the newly formed matrix $(\mathrm{F}_a^{(k)})^{-1}$, parts of $\mathbf{M}^T\mathbf{M}$ can be reused for simple combination according to the block-wise matrix inverse theory. Suppose that the i-th row and column of $(\mathbf{M}^T\mathbf{M})^{-1}$ is to be deleted, the new matrix can be formed by the other four block-wise matrices. By moving the deleted row and column to the bottom and the left respectively as shown in Figure 3, inverse of the new matrix can be calculated by Equation 14 from the block element A, b, c, d in $\mathbf{M}^T\mathbf{M}$.

Step 1. Load HSI and initialize the condition, start the outer loop.

Step 2. Find the first target signature candidate, enter the inner loop.

Step 3. Initial two sets of indices, then check the estimated abundant factors, denote all indices correspond to negative components as same as the original NCLS algorithm. If any component is negative, continue; or, get out of the inner loop.

Step 4. Compute a new set of abundant factors $\hat{\alpha}_i^{(k)}$ by least square method.

Step 5. Form a steering matrix $\Phi_{\alpha}^{(k)}$ by deleting all rows and columns in the matrix $(\mathbf{M}^T\mathbf{M})^{-1}$, where the negative indices are selected by Step 3.

Step 6. If the number of rows and columns deleted from matrix $(\mathbf{M}^T\mathbf{M})^{-1}$ is larger than the predetermined threshold ϖ, form the matrices **A, B, C,** and **D** in the matrix inverse downdating equations.

Step 7. Calculate the inverse of the steering matrix.

Step 8. Calculate the Lagrange multipliers and update the negative indices according to the Kunh-Tucker conditions.

Step 9. Update the estimated abundant factors, go to step 3.

Step 10. In the outer loop check the LSE by the predefined error (1.0e-24) to decide whether to append the vector to the signature matrix.

Step 11. Check the number of found subpixel targets. If it reaches the expected value, the algorithm stops, otherwise continue.

FIGURE II. THE PSEUDO CODE OF OUR FAST UNCLS ALGORITHM

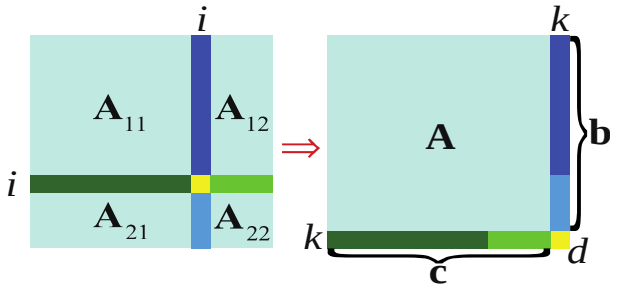


FIGURE III. FAST MATRIX INVERSE DOWNDATING BY MOVING THE SPECIFIED COLUMN AND ROW TO THE EDGES

Besides, negative indices may include more than one index for the rows and columns to be deleted in the NCLS algorithm, so B, C and D can be block-wise matrices instead of vectors or a scalar. Hence, we can express matrix inverse downdating in a block-wise format:

$$\mathbf{R}_{11}^{-1} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}. \tag{15}$$

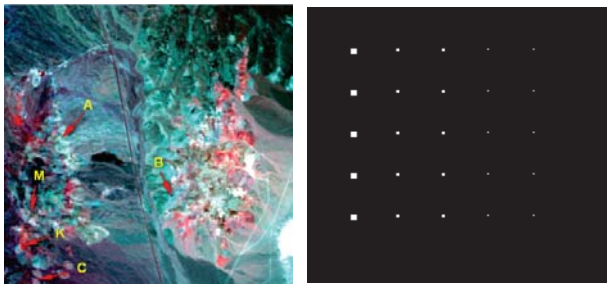
Note that Equation 15 can be more efficient only when $\mathbf{D}^{-1}$ is a small size matrix. When the sizes of $\mathbf{R}_{11}$ and D are close in the fast UNCLS algorithm, the downdating scheme cannot make an obvious improvement on the performance. If matrices $\mathbf{R}_{11}$ and D are close in size, the reusable matrix block is small, the downdating operation will consume almost the same computing time as inversing matrix D directly. Therefore, a predetermined threshold ϖ is defined to make a limitation on the size of downdating matrix scale in Step 6. The threshold is necessary since computing performance may be reduced if too many columns and rows are deleted from the steering matrix, which is also observed in our experiments. It means that the threshold constrains the maximum size of matrix D in Equation 15. The value of the threshold ϖ is important for the overall efficiency of the proposed fast UNCLS method. It is related to the size of the target signature matrix with respect to the number of signatures, once the number of signatures is determined the value of threshold can be predefined accordingly. Details about how we set the value of ϖ in the experiments are present in the next section.

Moreover, the performance of matrix inverse updating and downdating is compared with the built-in matrix inverse function of Matlab in Section 3.1. The proposed fast UNCLS method is evaluated with both simulated and real HSI data in the following section in Section 3.2.

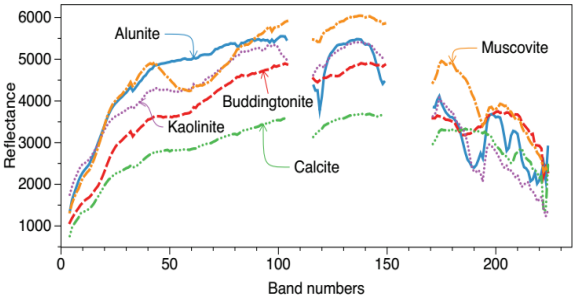
III. EXPERIMENTS

A. *Simulated Hyperspectral Image Evaluation*

We conducted simulated hyperspectral image experiments for two reasons: one is to verify the correctness of the proposed fast UNCLS algorithm; the other one is to evaluate the performance of the matrix inverse with updating/downdating versus Matlab built-in inv function. As illustrated in reference [5], TE (Target Embeddedness) is a simulated HSI primarily designed for target detection. Five pure spectra were extracted from the public Cuprite AVIRS hyperspectral image, which are alunite, buddingtonite, kaolinite, calcite, and muscovite as shown in Figure 4(a) and (c). There are 25 target panels in the simulation as shown in Figure 4(b). Each row is simulated by the same mineral signature and each column of five targets has the same pixel size. The first and second columns are pure spectral targets with 4x4 and 2x2 pixels, respectively. The third column is also in a 2x2 pixel size but the spectra is a mixture of the five pure signatures. The last two columns contain ten 1x1 pixel targets with a mixture of the five mineral signatures. The mixture rate is different between the fourth (1:1) and fifth column (1 3), which is of same with the TE2 scenario in reference [5]. Thus, we can simulate ten different subpixel targets in the last two columns. Besides, the targets pixels are embed into the background with a SNR = 20:1 Gaussian noise, which is simulated by the sample mean of the real Cuprite image. Figure 5(a) shows the embed targets as false color image with background noise, bands 40, 80, and 189 are used. For clarity, we have removed the noisy background for illustration of target detection in the following.



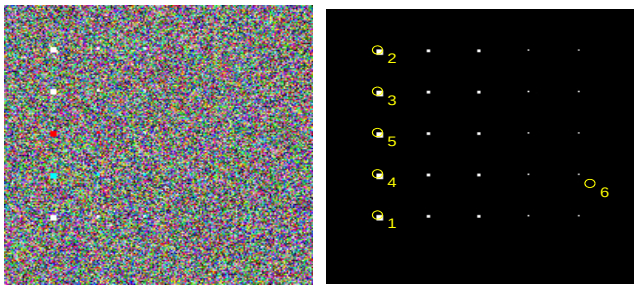
(a) Cuprite AVIRIS image (b) Simulated target



(c) Five selected mineral signatures

FIGURE IV. SIMULATED HSI SCENARIO WITH 25 TARGET PANELS

The hyperspectral target signature detection results by the original and out fast UNCLS method are identical as shown in Figure 5. The six signatures (the background was detected as one target signature) were marked with yellow circles and the numbers besides indicating the detection order.

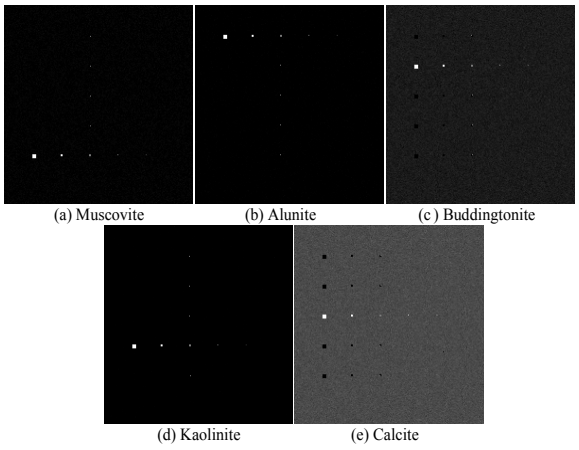


(a) Simulated HSI with noises. (b) Pure spectral signatures.

FIGURE V. TARGET SIGNATURES EXTRACTED FROM SIMULATED HSI BY THE PROPOSED FAST UNCLS

We also verified the abundance of the 5 detected target signatures, and illustrated the detection result in Figure 6. As seen from the figures, all the hyperspectral subpixel targets located in the last two columns were detected.

Comparing with the "ground truth" data [5,7], we calculated the RMSE (root mean square errors) of the detection results obtained by our fast UNCLS and the original one, as shown in Table I.



(a) Muscovite (b) Alunite (c) Buddingtonite

(d) Kaolinite (e) Calcite

FIGURE VI. ABUNDANCE FOR EACH TARGET SIGNATURE

TABLE I. RMSE OF THE TWO UNCLS METHODS WITH GROUND TRUTH DATA

Targets	T1	T2	T3	T4	T5
Original	0.0072	0.11	0.2969	0.0064	0.0383
Proposed	0.0072	0.11	0.2969	0.0064	0.0383

Identical RMSE of the original and our fast UNCLS verify the correctness of the proposed method tested on the simulated HSI. Taking advantage of the simulation of a hyperspectral image, we can create synthetic HSI with different spatial size, spectral signatures, and subpixel targets for experiments. Thus, we also evaluated the performance of the matrix inverse updating comparing with the traditional inv Matlab build-in function. Since we set the number of subpixel targets from 1 to 100, which means a set of 1×1 to 100×100 square matrices were inverted by the two methods, we simulated a set of different spatial size HSI with 224 bands. To guarantee an adequate data, we selected a 200 (samples) x 200 (lines) x 224 (bands) with 100 different signatures as research object. And all the tests were iterated by 100 times to retrieve the average computation time. The computing performance result is shown in Figure 7.

It is straightforward that matrix inverse updating method overcomes built-in inv function of Matlab as the size of the HSI matrix getting larger. In matrix inverse updating, each iteration of the calculation reuses the last inversion result, which gains a significant time saver. From the experimental results, we can conclude that the matrix inverse updating scheme is better at calculating an iterative inverse than the built-in inv function, and the speedup will go higher when the number of subpixel targets increases.

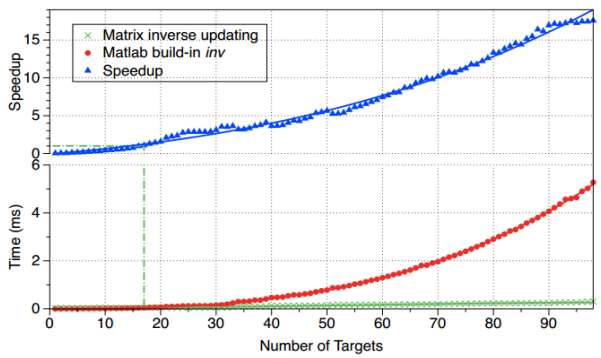


FIGURE VII. EVALUATION OF MATRIX INVERSE UPDATING AND BUILT-IN INV METHOD

In a similar way, we conducted a test for matrix inverse downdating method over built-in inv function of MATLAB. Figure 8 shows the computing time and speedup for the two methods. It can be seen that the speedup gets reduced as the deleted target signatures increased, which means downdating is suitable for inverting a block matrix only when deleting a small number of rows and columns. That is why a threshold ϖ is set in our fast algorithm. Besides, after testing various sizes of HSI with different number of subpixel targets (here a 200 x 200 with 60 signatures HSI is present), we found that the performance of matrix inverse downdating is better than the built-in inv function when the number of rows/columns to be deleted is less than half of the whole targets. Inspired by this observation, we can predetermine the value of threshold ϖ as half of the input number of expected target signatures in our real HSI experiments. The detailed experimental results are shown in the next section.

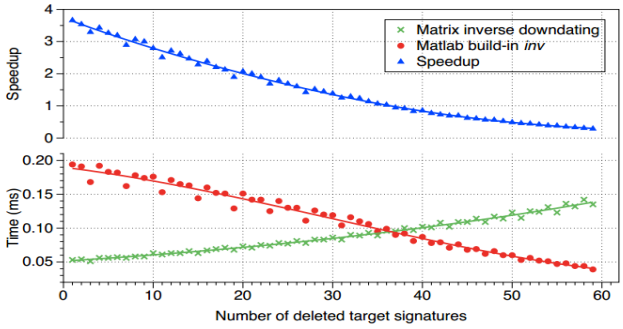


FIGURE VIII. EVALUATION OF MATRIX INVERSE DOWNDATING AND INV

It should also be noted that the performance will improved by matrix inverse updating and downdating only when we need to calculate matrix inverse iteratively and the matrix size is big enough. Fortunately, usage of the fast UNCLS algorithm fulfils the above two requirements in hyperspectral subpixel target detection. There are two reasons why updating and downdating scheme gains performance improvement with the proposed fast UNCLS method: one is that a hyperspectral data is usually with massive data in spatial resulting in a large matrix $\mathbf{M}^T\mathbf{M}$; the other one is that the UNCLS calculated the inverse of $\mathbf{M}^T\mathbf{M}$ iteratively by least square optimization.

B. *Real Image Evaluation*

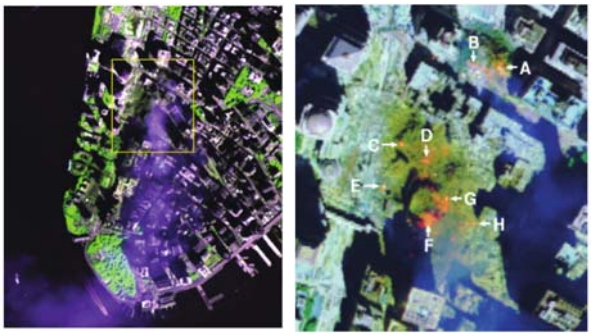


FIGURE IX. REAL HYPERSPECTRAL IMAGES CAPTURED BY AVIRIS.(A) FALSE COLOR IMAGE OF WTC REAL HSI. (B)THERMAL HOT SPOTS UNDER THE SMOKE

The WTC real HSI collected by NASA AVIRIS over the World Trade Center on 16 September 2001 was used in our experiment; it comprises 614 x 512 pixels with 224 spectral bands making about 140 MB of binary data. Since the spatial resolution is 1.7 meters, there exist many subpixel targets in this complex urban scene. The RGB (red, green, blue) false color image of WTC is formed using the 577.37, 836.27 and 451.72 nm channels, as shown in Figure 9(a). The green area delineates vegetation, while the purple smoke was over the burned area, and from the detailed center area (in the yellow rectangle) some thermal hot spots caused by the fire can be observed and are denoted from 'A' to 'H' in Figure 9(b). The hot spots are subpixel targets to be detected, the original data was collected by U.S. Geological Survey (USGS) and made available online (http://speclab.cr.usgs.gov/wtc/).

Both the original and our fast UNCLS algorithms were executed on the WTC real HSI. Firstly, we checked the correctness of the detection results, and we got the identical positions of signatures, as shown in Table II.

TABLE II. SIGNATURE POSITIONS EXTRACTED BY THE ORIGINAL AND PROPOSED UNCLS ALGORITHM

Sig#	UNCLS	Proposed	Signature#	UNCLS	Proposed
1	(145,468)	(145,468)	16	(57, 256)	(57, 256)
2	(68,155)	(68, 155)	17	(145,469)	(145,469)
3	(182,255)	(182,255)	18	(168,508)	(168,508)
4	(9,174)	(9, 174)	19	(467,497)	(467,497)
5	(444,484)	(444,484)	20	(356,439)	(356,439)
6	(356,438)	(356,438)	21	(449,488)	(449,488)
7	(488,441)	(488,441)	22	(143,464)	(143,464)
8	(67,384)	(67, 384)	23	(145,466)	(145,466)
9	(143,466)	(143,466)	24	(138,458)	(138,458)
10	(137,226)	(137,226)	25	(143,463)	(143,463)
11	(453,490)	(453,490)	26	(140,457)	(140,457)
12	(105,441)	(105,441)	27	(106,444)	(106,444)
13	(81, 483)	(81, 483)	28	(36, 389)	(36, 389)
14	(138,459)	(138,459)	29	(216,282)	(216,282)
15	(144,464)	(144,464)	30	(170,507)	(170,507)

And we marked all the signatures on the false color WTC image as shown in Figure 10, where the number of target signatures varies from 1 to 30. The positions of the detected signatures have been marked with red circles in the figure. For the fire and smoke area, signatures 3, 10 and 29 can be used to extract all subpixel targets in the WTC data.

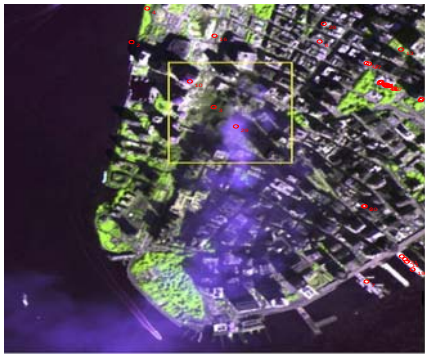


FIGURE X. THE 30 TARGET SIGNATURES WERE DETECTED ON THE WTC REAL HSI

We made further performance comparison of the proposed fast UNCLS with the original one on the real WTC HSI scene. The experimental platform is a computer with an Intel® Core™ i5-3210M @2.5GHz CPU and 4 GB RAM. Matlab R2013b was used on a Windows 7 64-bits operating system. For both the original and our fast UNCLS implementations, we ran 10 times to obtain a reliable result. The running times of the two methods and the speedups are plotted in Figure 11.

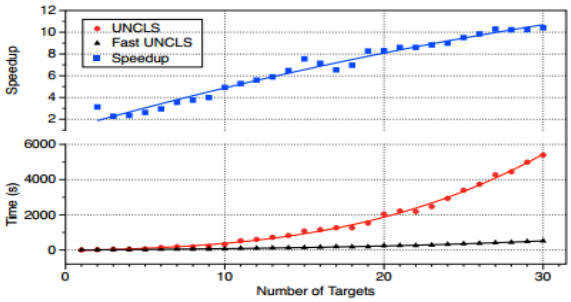


FIGURE XI. PERFORMANCE OF THE FAST AND ORIGINAL UNCLS WITH WTC HSI

From the speedup (blue-square line) shown in in Figure 11, we can see that our fast UNCLS algorithm gains a better performance as compared to the original one for each target signature detected. Note that the original UNCLS algorithm would take as much as 5396.8 seconds (about 1.5 hours) to finished detecting 30 target signatures from the WTC real HSI data, while our fast version reduced that to 518.5 seconds (about 8.6 minutes), a ~10x speedup achieved.

IV. CONCLUSION

The unsupervised nonnegative constrained least squares (UNCLS) method is a popular method for hyperspectral subpixel target detection. Depending on the complexity and dimensionality of the hyperspectral data, it may be time consuming for large matrix inversion iteratively in the original UNCLS algorithm. In this paper, we developed a fast UNCLS algorithm with matrix inverse updating and downdating theory. The correctness and performance are evaluated and analyzed with both the simulated and real HSI scenes. The proposed method improves the computational performance, which achieves a >10x speedup for both the simulated Cuprite and real WTC scenes when large number of targets are detected.

REFERENCES

[1] Manolakis, D.; Siracusa, C.; Shaw, G. Hyperspectral subpixel target detection using the linear mixing model. IEEE Trans. Geosci. Remote Sens. 2001, 39, 1392–1409.
[2] Du, B.; Zhang, Y.; Zhang, L.; Zhang, L. A hypothesis independent subpixel target detector for hyperspectral Images. Signal Process. 2015, 110, 244–249.
[3] Zhang, Y.; Du, B.; Zhang, L. A sparse representation-based binary hypothesis model for target detection in hyperspectral images. IEEE Trans. Geosci. Remote Sens. 2015, 53, 1346–1354.
[4] Liang, Y.; Markopoulos, P. P.; Saber, E. S. Subpixel target detection in hyperspectral images from superpixel background statistics. In Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International; IEEE, 2016; pp. 7018–7021.
[5] Chang, C.-I. Hyperspectral Data Processing: Algorithm Design and Analysis; 1 edition.; Wiley: Hoboken, NJ, 2013.
[6] Chang, C.-I.; Heinz, D. C. Constrained subpixel target detection for remotely sensed imagery. IEEE Trans. Geosci. Remote Sens. 2000, 38, 1144–1159.
[7] Chang, C.-I.; Plaza, A. A fast iterative algorithm for implementation of pixel purity index. IEEE Geosci. Remote Sens. Lett. 2006, 3, 63–67.
[8] Guerra, R.; López, S.; Callicó, G. M.; Lopez, J. F.; Sarmiento, R. On the acceleration of the N-FINDER algorithm for hyperspectral endmembers extraction. In Proc. SPIE 9124; Huang, B.; Chang, C.-I.; López, J. F., Eds.; Baltimore, Maryland, USA, 2014; Vol. 9124, p. 91240H.
[9] Lopez, S.; Moure, J. F.; Plaza, A.; Callico, G. M.; Lopez, J. F.; Sarmiento, R. A New Preprocessing Technique for Fast Hyperspectral Endmember Extraction. IEEE Geosci. Remote Sens. Lett. 2013, 10, 1070–1074.
[10] Jiménez, L. I.; Martin, G.; Sanchez, S.; García, C.; Bernabe, S.; Plaza, J.; Plaza, A. GPU Implementation of Spatial–Spectral Preprocessing for Hyperspectral Unmixing. IEEE Geosci. Remote Sens. Lett. 2016, 13, 1671.
[11] Santos, L.; Berrojo, L.; Moreno, J.; López, J. F.; Sarmiento, R. Multispectral and Hyperspectral Lossless Compressor for Space Applications (HyLoC): A Low-Complexity FPGA Implementation of the CCSDS 123 Standard. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2016, 9, 757–770.
[12] Bernabé, S.; Botella, G.; Navarro, J. M.; Orueta, C.; Prieto-Matias, M.; Plaza, A. Parallel implementation of the simplex growing algorithm for hyperspectral unmixing using OpenCL. In Geoscience and Remote Sensing Symposium (IGARSS), 2016 IEEE International; IEEE, 2016; pp. 6153–6156.
[13] Wu, X.; Huang, B.; Plaza, A.; Li, Y.; Wu, C. Real-Time Implementation of the Pixel Purity Index Algorithm for Endmember Identification on GPUs. IEEE Geosci. Remote Sens. Lett. 2014, 11, 955–959.
[14] Wu, X.; Huang, B.; Wang, L.; Zhang, J. GPU-Based Parallel Design of the Hyperspectral Signal Subspace Identification by Minimum Error (HySime). 2016.
[15] Bernabe, S.; Lopez, S.; Plaza, A.; Sarmiento, R. GPU Implementation of an Automatic Target Detection and Classification Algorithm for Hyperspectral Image Analysis. IEEE Geosci. Remote Sens. Lett. 2013, 10, 221–225.
[16] Gonzalez, C.; Resano, J.; Plaza, A.; Mozos, D. FPGA Implementation of Abundance Estimation for Spectral Unmixing of Hyperspectral Data Using the Image Space Reconstruction Algorithm. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2012, 5, 248–261.
[17] Hager, W. Updating the Inverse of a Matrix. SIAM Rev. 1989, 31, 221–239.
[18] Lawson, C. L.; Hanson, R. J. Solving Least Squares Problems; Society for Industrial and Applied Mathematics: Philadelphia, 1987.
[19] Bro, R.; De Jong, S. A fast non-negativity-constrained least squares algorithm. J. Chemom. 1997, 11, 393–401.
[20] Sherman, J.; Morrison, W. J. Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. Ann. Math. Stat. 1950, 21, 124–127.
[21] Golub, G. H.; Loan, C. F. V. Matrix Computations; fourth edition edition.; Johns Hopkins University Press: Baltimore, 2012.
[22] Skiena, S. S. The Algorithm Design Manual; 2nd edition.; Springer: London, 2008.