# Sequential and Parallel Feature Extraction in Hyperspectral Data Using Nonnegative Matrix Factorization

Stefan A. Robila, *Member IEEE,* Lukasz G. Maciak

*Abstract*—**Feature extraction refers to the groups of techniques that, when applied to large dimensional and redundant data result in significant dimensionality reduction while preserving or even enhancing the information content. Among various techniques investigated for feature extraction, of new interest is Nonnegative Matrix Factorization (NMF). In NMF, it is assumed that the data is formed as a linear nonnegative combination of positive sources and the NMF solution recovers the original sources and the mixing matrix.**

**In this paper, we first look at ways NMF can be applied for feature extraction in hyperspectral imagery a data known for large sizes and redundancy. While some of the associations are natural to linear mixing model (LMM - that assumes that hyperspectral images are formed as a linear mixture of endmember information), we also show NMF to be a slow method. To counter this, we investigate alternative solutions such as projected NMF approaches and provide an insight to how parallel implementations would contribute to speedup. Experimental results on various data show projected NMF outperforming regular NMF with parallel implementations providing a promising speedup advantage.**

*Index Terms*—**remote sensing, hyperspectral data, linear unmixing, linear algorithms, Nonnegative Matrix Factorization.**

## I. Introduction

In remote sensing, few areas have seen such an accelerated development as it is the case with hyperspectral and multispectral imagery [1]. While spectral imaging has started as a basic evolution of the photogrammetry, the ability to increase the sensed information through the finer and finer separation of the light spectrum in different recorded images has resulted in tremendous advances in processing and extension of applicability that allow us to consider spectral imaging as a distinct field in its own.

Given any material, and using readily available lab equipment, one can measure the intensity of the light reflected by the material [2]. The resulting data are usually presented as a continuous graph of reflectance intensity versus light wavelength (often called *spectra*) [3]. Figure 1 displays the spectra for grass (solid line) and dry grass (dashed line) as provided by the United States Geological Survey Spectral Library, one of the largest sources of spectral information [3]. Note how the reflectance values differ significantly for the visible to infrared interval (400 nm to 1600 nm). This is due to the differences in both color (for the visible area) and chlorophyll content (for the infrared area).

When exact measurements are possible, the spectral characteristics of a material allow for accurate identification. Unfortunately, collecting spectral profiles similar to the ones provided by USGS is done using spectral radiometers able only to record information for only one distinct point. Compared to that, hyperspectral imaging sensors have the advantage of collecting data over larger areas by producing image based representation of the spectral characteristics. Figure 2 displays the parts of the data produced by such an imaging sensor (SOC 700). In this case, the spectral information corresponding to the wavelength intervals is colleted as image bands covering a relatively large area. Figure 2a shows the image of a ceramic plant pot placed on a large rock formation and having as background a brick wall. The plants arranged in the pot are both natural and artificial. Extracting a vector of pixels from the formed image cube is similar to producing the spectra for the material corresponding to the pixel location. Figure 2c shows spectra extracted in this way for fake and real (solid and dashed lines respectively) as well as for the ceramic pot (diamond markers) and the rock formation (dotted line). Figure 2d shows a vertical slice through the data cube where the spectra can be seen as rows.

Hyperspectral imagers are characterized by spectral and spatial resolution. Spectral resolution refers to the width of the spectral wavelength intervals associated to each of the bands and the spatial resolution refers to the surface (in square inches, feet, or miles) covered by each pixels. Both characteristics, together with the uncertainties of the atmospheric conditions and light illumination indicate that the spectra collected may not match the one recorded through regular spectrometers. Pixel values are not 'pure' spectrally, in the sense that the atmosphere, and the light do not offer perfect observation conditions and the spectral resolution may be

This work was supported in part by a 2005-2006 Sun Microsystems Academic Excellence Grant

S. A. Robila is with the Department of Computer Science at Montclair State University, Montclair, NJ 07043, USA, (phone: 973-655-4230, fax: 973-655-4164, email: robilas@mail.montclair.edu).

L. G. Maciak is with the Department of Computer Science at Montclair State University, Montclair, NJ 07043, USA, (email: maciakl1@mail.montclair.edu.).

1-4244-1302-8/07/$25.00 ©2007 IEEE

considerably coarser than the one obtained in a lab environment. Pixel values are neither 'pure' spatially since they most probably span over areas cover by a group of materials.

To extract useful information from hyperspectral imagery, one must use methods that would reduce the spatial and spectral ambiguity and increase the relevance of the data. One traditional approach for processing hyperspectral data is feature extraction. Feature extraction is defined as the process of reducing the data to a lower dimension without significant information loss [5]. In our case, this is done by either selecting certain bands of by using a transform that produces the features as combinations of bands.

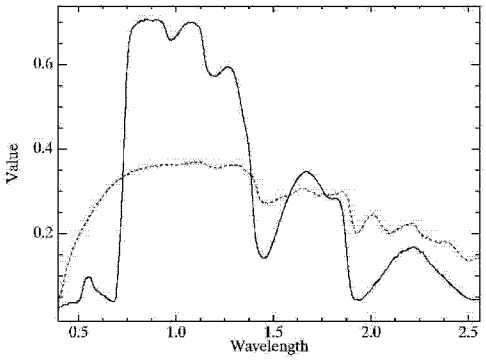


Fig. 1. Spectra for two different stages of vegetation growth measured over the 400-2500 nm spectral range. Healthy grass (solid) and dry grass (dashed) are plotted. Note the significant differences for the left side of the graph corresponding to visible and near infrared ranges. Wavelengths provided in micrometers (1μm = 1000nm).

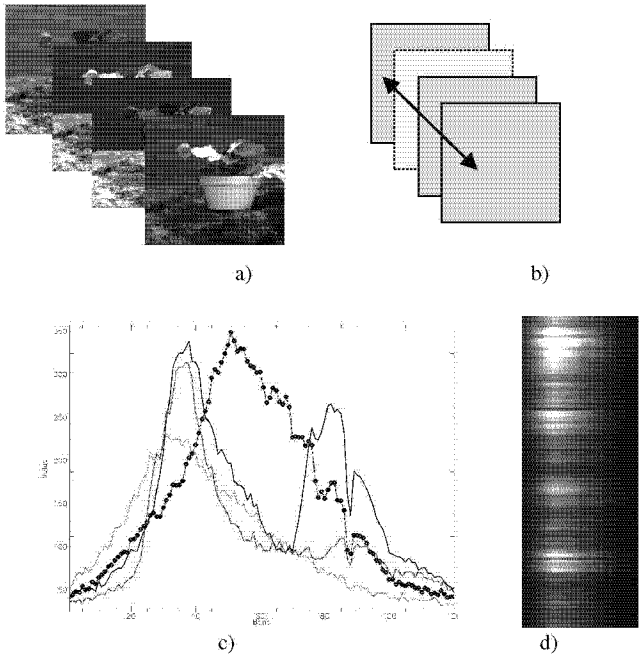


Fig. 2. Spectral bands provided by a hyperspectral imager corresponding to portions of the green, blue and red lights. (a) The images colleted by the sensor are grouped in (b) hyperspectral data cubes. (c) The spectral can now be collected as a vector through the cube. (d) A slice through the cube showing the various spectra, Data collected using a SOC 700 imaging system and recorded over the 400 nm – 900 nm wavelength interval in 120 bands.

While it is possible to envision supervised feature extraction where the bands are formed in order to increase separability of known materials, it is often the case that no prior information is available on the scene surveyed [6]. In this case (usually called unsupervised), the algorithms focus on the increase of the separation between classes within each feature. The separation is measured using class information such as distance between means, distance between probabilities, etc [3]. Previous efforts have focused on Principal Component Analysis (PCA) [7], Orthogonal Subspace Projection (OSP) [8], Minimum Noise Fraction (MNF) [9], Independent Component Analysis (ICA) [10], to name a few. Many of them however require significant constraints on the nature of the extracted features (such as decorrelation or independence), and may result in decreased relevance of the results.

In a series of recently published papers we have investigated how Nonnegative Matrix Factorization (NMF) can be employed for feature extraction [11, 12]. The studies showed NMF can have a natural interpretation in hyperspectral imagery and can be seen to closely match the linear mixing model used in spectra unmixing. However, they have also indicated that NMF is slow to converge. To counter this, in the current paper we survey a new NMF algorithm based on projected gradients. To increase the speedup we also look at a parallel implementation scenario. The paper is organized as follows. In the second section we provide a brief overview of NMF and its modeling for hyperspectral image processing. Section III describes the new projected gradient algorithm and discusses possible parallel implementations. Section IV provides a survey of our experimental results. The paper ends with Conclusions (section V) and References.

## II. Nonnegative Matrix Factorization (NMF)

Given the observed data **x**, the goal of NMF is to find **s** and a linear mixing transform **W** both positively defined such that [13]:

$$\mathbf{x}=\mathbf{W}\mathbf{s} \tag{1}$$

This approach can be understood as factorizing a data matrix subject to positive constraints. Solutions to NMF are based on constraining positivity and the gradient optimization (minimizing the distance between x and iterations of Ws). The optimization is done by repeatedly updating **W** and **s** using [13]

$$\mathbf{W} = \mathbf{W} - \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{W}} \tag{2}$$

$$\mathbf{s} = \mathbf{s} - \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} \tag{3}$$

where:

$$f(\mathbf{W},\mathbf{s}) = \left\|\mathbf{x} - \mathbf{W}\mathbf{s}\right\|_F^2 \tag{4}$$

and $\|.\|_F$ designates the Frobenius (or Euclidean) norm. At each step we also ensure that **W** and **s** are positive and s is normalized.

An algorithm including the positivity restrictions is presented in [13] and was used to separate a limited number of hyperspectral spectra in [14]. In this case, the equations 2 and 3 are substituted by:

$$s_{ij} = s_{ij} \frac{\left(\mathbf{W}^{\mathrm{T}}\mathbf{x}\right)_{ij}}{\left(\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{s}\right)_{ij} + \varepsilon} \tag{5}$$

$$W_{ij} = W_{ij} \frac{\left(\mathbf{x}\mathbf{s}^{\mathrm{T}}\right)_{ij}}{\left(\mathbf{W}\mathbf{s}\mathbf{s}^{\mathrm{T}}\right)_{ij} + \varepsilon} \tag{6}$$

respectively.

In [11] we present a similar approach. Compared with NMF algorithms described in the literature our change was to enforce the restrictions on **s** such that they will satisfy the linear mixing model (LMM). In LMM, each observed spectra **x** can be expressed as [11]:

$$\mathbf{x} = \sum_{i=1}^{m} a_i \mathbf{s_i} + \mathbf{w} = \mathbf{S}\mathbf{a} + \mathbf{w} \tag{7}$$

where **S** is an *nxm* matrix of spectra ($\mathbf{s_1}$, .., $\mathbf{s_m}$) of the individual composing materials (also called *endmembers*), **a** is an *m*-dimensional vector describing the fractional abundances of the endmembers in the mixture (*abundance vector*) and **w** is the additive noise vector. The elements of the abundance vector are assumed to be positive and with unit sum:

$$a_i \geq 0, i = 1, ...., m \tag{8}$$

$$\sum_{i=1}^{m} a_i = 1 \tag{9}$$

Identifying various materials in the image means finding the endmembers and their abundances. The endmember linear unmixing problem plays an important role in hyperspectral image analysis [11]. Feature extraction can be seen as performing unmixing. The abundance determination is done at the same time with endmember selection. In this case, the endmembers and abundances are assumed deterministic and unknown and a maximum likelihood estimation approach is employed to determine them.

Most of the feature extraction techniques cannot be directly applied for endmember extraction since they do not verify the equations 7 and 8. In addition, techniques such as PCA and ICA further restrict the endmembers to be orthogonal or independent. In this view, NMF is less restrictive and does not require any significant modification for applying it for unmixing.

A quick analysis of the update steps in (4) and (5) reveals that while they ensure that the matrices remain positively defined, they also require significant computation times. If we assume that **W** is *n* x *m*, **x** is *n* x *p*, and **s** is *m* x *p*, computing the equation 4 requires O(*nmp*) operations, while the equation 5 requires O($nmp+p^2m$). Assuming large values for *p* since it relates to the number of pixels in the image and the fact that the iterative algorithms will require several iterations, NMF seems to be a computationally intensive method.

One possible problem is that the update step does not involve any fine tuning. When the new values for **x** and **W** are computed, no control is provided to ensure that the update leads to the optimum values or we are in fact 'overshooting'. To counter this, we investigated a new direction, recently introduced in the literature and based directly on the gradient steps.

## III. Adaptive Projected Nonnegative Matrix Factorization (APNMF)

Given the original update steps (2) and (3), there is no warrantee that the resulting values will not include negative components. To counter this, we modify these steps to enforce positivity:

$$\mathbf{W} = P\left(\mathbf{W} - \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{W}}\right) \tag{10}$$

$$\mathbf{s} = P\left(\mathbf{s} - \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right) \tag{11}$$

where:

$$P(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{12}$$

In other words, the update will occur only when the positivity is ensured. In this case, it is possible to build an NMF algorithm that tries to compute the best update step for each iteration.

The algorithm is described in Figure 3 and constitutes a simplified version of the one presented in [15]. There are several significant differences compared to the original NMF algorithms. First, we note that in the steps 7 and 10, **W** and **s** respectively are constants. This means that we can compute the derivative on **s** (and on **W** respectively) without worrying about the other component of the NMF problem. Second, the direct use of the gradient formula allows for the introduction of an adaptive factor α that that will lead to an update on **s**, and **W** with higher precision than the regular gradient approach. Because of this, the algorithm is also often described as *alternative projection optimization* [15]. Previous studies indicate that using alternative projection steps still allows for a determination of the optimum value [16,17]. Finally, the algorithm now has a clear stop condition (see step 12). We will stop when the change between two consecutive iterations has not produced a significant improvement (expressed as a fraction ε from the previous value of the function *f* described in equation 3). In our experiments, ε had the value 0.001 – i.e. one thousandth.

Of particular interest is the choice for α in the steps 6 and 9. Since in both cases we can come up with similar strategies, we will focus on only step 6. Given a generic current solution **W** and **s**, the goal is to find α such that:

$$\alpha = \arg\min_{\alpha}\left(f\left(\mathbf{W}, P\left(\mathbf{s}_k - \alpha\frac{\partial f(\mathbf{W}_k,\mathbf{s}_k)}{\partial \mathbf{s}_k}\right)\right)\right) \tag{13}$$

In other words, we must find the update parameter that optimally minimizes the distance (based on **s**) to the solution. [15] and [16] discuss an iterative process that can produce an approximation of such α. In this case, we start with a fixed value (say 0.001) and we increase it or decrease it until the optimal value is reached. An efficient way of the update is given in Figure 4. In short, if the update provided by the current α is too small (the test in step 4) we will increase alpha

by ten (i.e. divide by β=0.1). We will continue increasing as long as the update is still small (step 5) and will stop at the largest value that ensured that the update is not overshooting. In the case that the update is already overshooting in step 4, we will start a decrease process for α by repeatedly dividing it by 10 (i.e. multiply with β=0.1) and stop once we find a value that does not overshoot the optimal solution. In all our experiments we followed the suggestion that σ = 0.01.

We note that we have extensively used the term of overshooting. In our understanding, overshooting means that the update value for **s** will not provide a sufficient decrease of the optimization function. The condition:

$$f(\mathbf{W},\bar{\mathbf{s}}) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\bar{\mathbf{s}} - \mathbf{s}) \quad (14)$$

1. Randomly initialize **W** and **s** to positive values
2. Scale the columns of **s** to sum up to one
3. $\mathbf{W_0} = \mathbf{W}$ and $\mathbf{s_0}=\mathbf{s}$
4. Compute $d_0=f(\mathbf{W_0},\mathbf{s_0})$
5. Repeat for $k$=0,1,…:
   6. Find optimum α for 7.
   7. $\mathbf{s}_{k+1} = P\left(\mathbf{s}_k - \alpha \frac{\partial f(\mathbf{W}_k,\mathbf{s}_k)}{\partial \mathbf{s}_k}\right)$
   8. Scale the columns of **s** to sum up to one
   9. Find optimum α for 10.
   10. $\mathbf{W}_{k+1} = P\left(\mathbf{W}_k - \alpha \frac{\partial f(\mathbf{W}_k,\mathbf{s}_{k+1})}{\partial \mathbf{W}_k}\right)$
   11. Compute $d_{k+1}=f(\mathbf{W_{k+1}},\mathbf{s_{k+1}})$
12. while $(d_k - d_{k+1}) < \varepsilon d_k$

Fig 3. Adaptive Projected Nonnegative Matrix Factorization algorithm (APNMF)

1. Start with current **W** and **s**
2. Start with α=0.001, β = 0.1, and σ=0.01
3. Compute $\bar{\mathbf{s}} = P\left(\mathbf{s} - \alpha \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right)$
4. If $f(\mathbf{W},\bar{\mathbf{s}}) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\bar{\mathbf{s}} - \mathbf{s})$
   do steps 5 and 6
   Else
   do step 7
5. while $f(\mathbf{W},\bar{\mathbf{s}}) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\bar{\mathbf{s}} - \mathbf{s})$
   5.1. α=α / β
   5.2. $\bar{\mathbf{s}} = P\left(\mathbf{s} - \alpha \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right)$
6. α=α * β
7. while $f(\mathbf{W},\bar{\mathbf{s}}) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\bar{\mathbf{s}} - \mathbf{s})$
   6.1. α=α * β
   6.2. $\bar{\mathbf{s}} = P\left(\mathbf{s} - \alpha \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right)$

Fig 4. Computation of the optimum update value for APMNF.

has been shown to ensure that a sufficient decrease will occur and that an optimum solution will be reached [15].

To obtain the update step for W we note that we can apply the same algorithm as Figure 4 since:

$$\mathbf{x} = \mathbf{W}\mathbf{s} \Leftrightarrow \mathbf{x}^T = \left(\mathbf{W}\mathbf{s}\right)^T \Leftrightarrow \mathbf{x}^T = \mathbf{s}^T\mathbf{W}^T \quad (15)$$

Finally, we note that the condition 14 is not well formed from the point of view of vector arithmetic. An alternative condition is provided in [Ref]:

$$(1-\sigma)\left\langle \frac{\partial f(W,\mathbf{s})}{\partial \mathbf{s}}, \bar{\mathbf{s}} - \mathbf{s} \right\rangle + \frac{1}{2}\left\langle \bar{\mathbf{s}} - \mathbf{s}, \mathbf{W}^{\mathrm{T}}\mathbf{W}\left(\bar{\mathbf{s}} - \mathbf{s}\right)\right\rangle \le 0 \quad (16)$$

$$\frac{\partial f(W,\mathbf{s})}{\partial \mathbf{s}} = \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{s} - \mathbf{W}^{\mathrm{T}}\mathbf{x} \quad (17)$$

The developed algorithm has a complexity higher than the regular NMF one. At each iteration, to compute the formula in equation 17 we would need $O(m^2n^2p)$. Then, to compute the new **s** we need $O(mp)$ followed by the check for condition 16. This is done in $O(m^2n^2p)$. Overall, to compute the new updated **s** the algorithm in Figure 4 needs $O(km^2n^2p)$ where $k$ is the number of rounds the steps 5 or 7 are needed to be run.

To speedup the processing, we suggest a parallelization of the algorithm in Figure 4. Figure 5 shows such parallelization based on a generic number of $t$ threads. The algorithm starts by deciding which direction the search for α should be taken. From this, we will compute in parallel several new possible update steps and chose the best one.

1. Start with current **W** and **s**
2. Start with α=0.001, β = 0.1, and σ=0.01, and $t$ parallel processes.
3. Compute $\bar{\mathbf{s}} = P\left(\mathbf{s} - \alpha \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right)$
4. If $f(\mathbf{W},\bar{\mathbf{s}}) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\bar{\mathbf{s}} - \mathbf{s})$
   do step 5
   Else
   do step 6
5. Compute in parallel for $\alpha_1=\alpha/\beta$, ..,$\alpha_1=\alpha/\beta^t$
   $\bar{\mathbf{s}}_k = P\left(\mathbf{s} - \alpha_k \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right)$
   5.1. Find largest $k$ s.t. $f(\mathbf{W},\mathbf{s}_k) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\mathbf{s}_k - \mathbf{s})$
   5.2. If $k$ is equal $t$, repeat 5 with $\alpha=\alpha_t$
   If $k$ not found, use the original α
   Otherwise $\alpha=\alpha_k$ and stop
6. Compute in parallel for $\alpha_1=\alpha^*\beta$, ..,$\alpha_1=\alpha^*\beta^t$
   $\bar{\mathbf{s}}_k = P\left(\mathbf{s} - \alpha_k \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}}\right)$
   6.1. Find smallest $k$ s.t. $f(\mathbf{W},\mathbf{s}_k) - f(\mathbf{W},\mathbf{s}) \le \sigma \frac{\partial f(\mathbf{W},\mathbf{s})}{\partial \mathbf{s}} (\mathbf{s}_k - \mathbf{s})$
   6.2. If $k$ not found, repeat 6 with $\alpha=\alpha_t$
   Otherwise $\alpha=\alpha_k$ and stop

Fig 5. Parallel computation of the optimum update value for APMNF.

The PAPNMF (Parallel Adaptive Projected Nonnegative Matrix Factorization) algorithm that results has the advantage of providing a faster iterative step, closer to $O((k/t)m^2n^2p)$. It is interesting to note that if $\alpha$ is very close to the original starting value, the parallel version may not provide significantly better results.

## IV. Experimental Results

The sequential and parallel APNMF algorithms were implemented in Matlab 7.3 and run a on a Dell Latitude system with Intel Pentium 4 at 2.4GHz processor and 512MB of RAM. In order to avoid repletion, we refer the reader to our previous work in [11] for an explanation on the relvance of NMF for feature extraction. In this paper, we have focused the practical experiments on the comparative analysis of accuracy between regular NMF and APNMF as well as performed a preliminary investigation on the speedup provided by the PAPNMF.

### A. HYDICE Data

The hyperspectral image set (see Figure 6) was provided from the Hyperspectral Digital Imagery Collection Experiment (HYDICE) by the Spectral Information Technology Application Center (SITAC). It corresponds to a foliage scene taken from with a spatial resolution of 1.5m at wavelengths between 400nm and 2.5 micron part of the Forrest Radiance set. Various panels are present in the scene organized on eight rows and of different sizes (3m x 3m, 2m x 2m and 1m x 1m from left to right respectively).

A subset of 85 bands uniformly extracted from the data was used as input to the NMF algorithm. Starting with 10 different initializations for **W** and **s** we have run NMF, APNMF, and PAPNMF (varying the number of parallel processes from 2 to 8). Figure 8 shows the progression of accuracy (based of the function $f$ described in equation 4) vs the number of rounds. We note that while both NMF and APNMF have stopped within 20 rounds for $\varepsilon = 0.01$, neither have stopped after 200 rounds for $\varepsilon = 0.01$. For clarity, we have plotted only the first 25 rounds, with the mention that, after 200 rounds APNMF has reached a value approximately half than the one reaced by NMF at the same number of rounds. We conjecture that APNMF, through its fine tuning algorithm is able to match closer the optimal solution than the original multiplicative NMF method.

We did not expect the PAPNMF to provide any changes in the accuracy progress when compared with APNMF. Indeed, a review of the algorithm presented in Figure 5 convinces the reader that at each step PAPNMF will produce the same update coefficient as APNMF. A cursory look at the resulted experimental data verified out theoretical conclusion.

When looking at the average number of iterations needed for the APNMF to provide the next $\alpha$ compared to PAPNMF we notice a remarkable reduction (see Figure 9). While the sequential solution requires on average three iterations, the number decreases to close to one once four or more parallel processes are employed. Since only three values are checked on average to yield the new $\alpha$, it is clear that increasing the parallelization beyond four will not help.

### B. SOC Data

The second experiment uses data produced using a SOC 700 hyperspectral sensor currently available in our lab. The camera is able to produce 640x640 pixel images on 120 bands equally spaced within the 400nm and 900nm (i.e. visible to near-infrared range). Forty bands uniformly extracted from the image cube were used for the experiment.

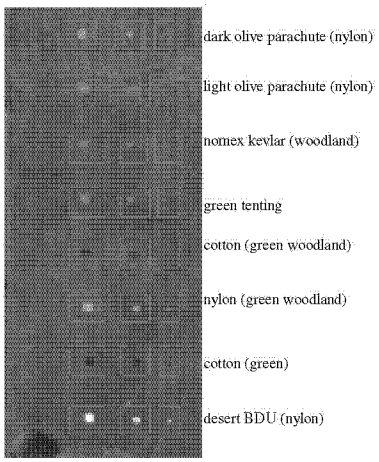


Fig 6. Hydice data scene with the panels highlighted.

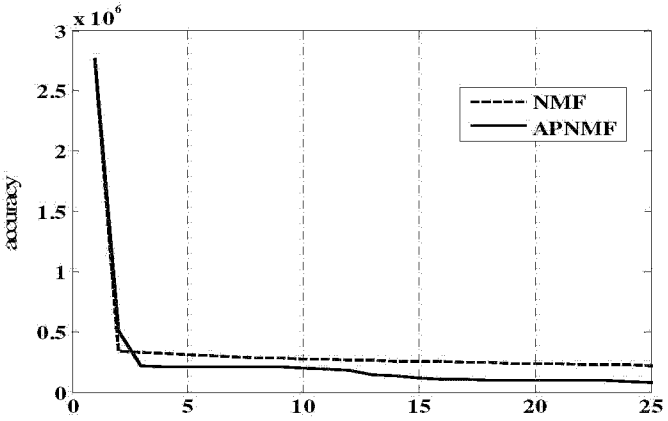


Fig 8. Accuracy graph for MNF and APNMF applied to the Hydice data for the first 25 iterations.

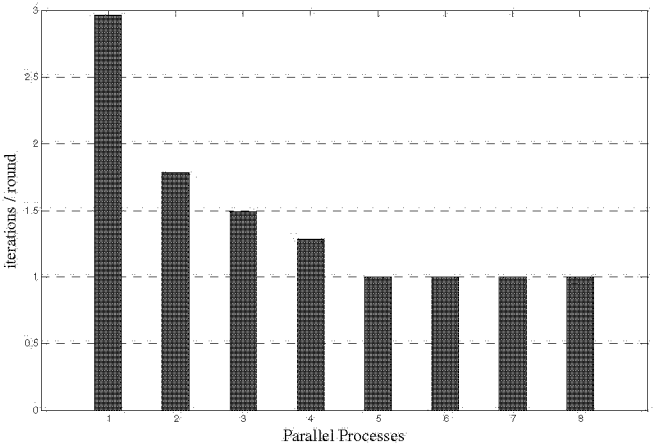


Fig 9. Average number of iterations for each round versus the number of parallel processes used in the case of the Hydice data.

Figure 10 shows the selected image. The set-up is formed of an artificial plant arranged in a light brown ceramic pot. Several real leaves (shown in enhanced green in the picture) were placed between the artificial leaves (left side, top and lower right side of the green area). To benefit from full spectrum illumination, the arrangement was placed outside on a large rock formation. The background is formed of a brick wall.

We have followed experimental scheme similar to the one for Hydice (10 runs, each including NMF, APNMF and PAPNMF with the number of parallel processes ranging from 2 to 8). Figure 11 shows the average progression of accuracy (based of the function *f* described in equation 4) vs the number of rounds. For clarity, we have plotted only the first 25 rounds, with the mention that, again, after 200 rounds APNMF has reached a value significantly lower than the one reached by NMF at the same number of rounds.

When looking at the average number of iterations needed for the APNMF to provide the next $\alpha$ compared to PAPNMF we notice a remarkable reduction (see Figure 12). While the sequential solution requires on average three iterations, the number decreases to close to one once four or more parallel processes are employed. Since only three values are checked on average to yield the new $\alpha$, it is clear again that increasing the parallelization beyond four will not help. While not plotted, we also note that our empirical observations lead us to believe that the number of iterations for $\mathbf{s}$ has been significantly lower (usually 1 increase from 0.0001) compared to the number of iteraions for $\mathbf{W}$ (usually 4-6 decreases from 0.0001). While we noted consistent behavior among the two data sets, the only viable explanation would be that $\mathbf{s}$ is a significantly larger data compared to $\mathbf{W}$ and this, in turn will minimize any differences when computing the formula 16.

## V. Conclusion

We have investigated a new approach to solving the Nonnegative Matrix Factorization problem when employed for feature extraction in hyperspectral imagery. While previiusl work has suggested NMF as a viable tool in providing improved data representation (a key component of feature extraction), questions remain related to the convergence of the method. I our new investigations we have analyzed a newly proposed algorithm based on adaptive projective gradient and modified it to fit the hyperspectral linear mixing model. The algorithm is further improved by an elegant parallelization of the most time consuming component, the search for the optimum update coefficient.

Our experiment results support the claim that APNMF outperforms NMF by resulting into a lower valued accuracy. Furthermore, when we employ parallelization, we discover a reduction of up to three fold in the computation time. Given the current programming environment that was based on a sequential implementation and a single processor machine, our results are mostly of theoretical interest. Current work in developing an efficient parallel processing module for NMF are underway and will be reported in future communications.

Fig 10. SOC 700 image used in the experiments

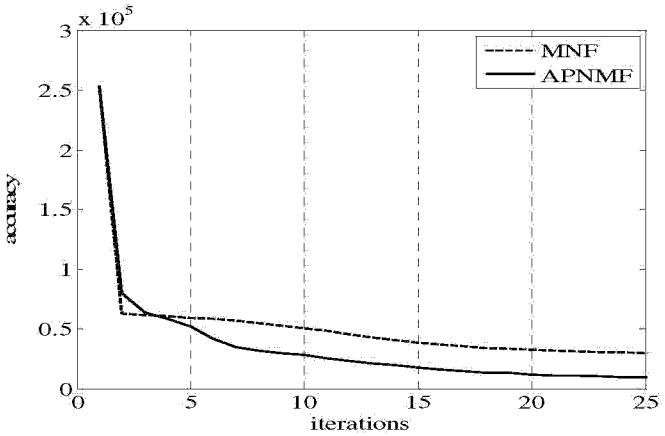


Fig 11. Accuracy graph for MNF and APNMF for the SOC data for the first 25 iterations.

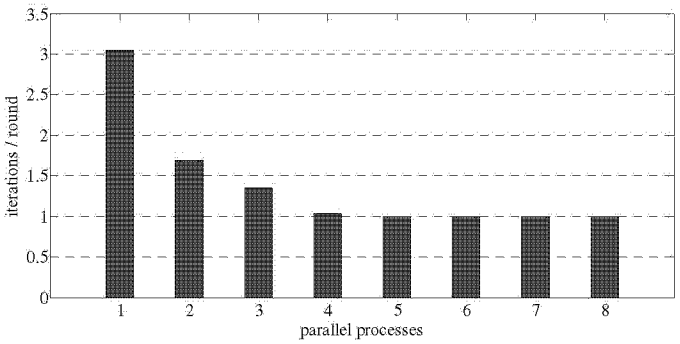


Fig 12. Average number of iterations for each round versus the number of parallel processes used in the case of the SOC data.

## References


[1] David Landgrebe, Signal Theory Methods in Multispectral Remote Sensing, John Wiley and Sons, 2003.

[2] T. M. Lillesand, R.W. Kiefer, Remote sensing and image interpretation, John Wiley and Sons, New York, 2000

[3] J. A. Richards, X. Jia. Remote Sensing Digital Image Analysis, Springer, New York, 1999

[4] R.N. Clark, G.A. Swayze, A.J. Gallagher, T.V.V. King, and W.M. Calvin, 1993, The U. S. Geological Survey, Digital Spectral Library: Version 1: 0.2 to 3.0 microns, U.S. Geological Survey Open File Report, 93-592, 1340 pages, http://speclab.cr.usgs.gov. (accessed Jan 1, 2007)

[5] S. A. Robila, P. K. Varshney, "Extracting Features from Hyperspectral Data Using ICA", in P.K. Varshney, M.K. Arora editors. Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data, Springer, New York, 2004, pp. 199 - 216.

[6] P.M. Mather, Computer processing of remotely-sensed images, John Wiley & Sons, 1987

[7] P.J. Ready, and P.A. Wintz, "Information extraction, SNR improvement, and data compression in multispectral imagery", IEEE Transactions on Communications, Vol. Com-21, 1973, 1123-1130.

[8] J.C. Harsanyi, and C.-I Chang, "Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach", IEEE Transactions on Geoscience and Remote Sensing, vol 32, no 4, pp: 779-785, 1994

[9] A.A. Green, M. Berman, P. Switzer, and M.D. Craig, "A transformation for ordering multispectral data in terms of imagequality with implications for noise removal", IEEE Transactions on Geoscience and Remote Sensing, vol 26, no 1, pp. 65-74, 1988

[10] S. A. Robila, "Independent Component Analysis (ICA)", in P.K. Varshney, M.K. Arora editors. Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data, Springer, New York, 2004, pp. 109 - 132.

[11] S. A. Robila, L. Maciak, "Novel Approaches for Feature Extraction in Hyperspectral Images", Proceedings IEEE LISAT, 2006, 7 pgs. on CD.

[12] S. A. Robila, L. Maciak, "Parallel Unmixing Algorithms for Hyperspectral Image Processing", SPIE Intelligent Robots and Computer Vision XXIV, vol. 6384, 2006, 10pgs., in print

[13] D. Lee and H. Seung. "Algorithms for Non-Negative Matrix Factorization", Advances in Neural Processing, 2000

[14] P. Pauca, J. Piper, and R. Plemmons, "Nonnegative Matrix Factorization for Spectral Data Analysis", to appear in Linear Algebra and Applications, 2006

[15] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. To appear in Neural Computation, 2007

[16] D. P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. IEEE Transations on Automatic Control, 21:174–184, 1976.

[17] D. P. Bertsekas. Nonlinear Programming. Athena Scientific, Belmont, MA, second edition, 1999

Dr. **Stefan A. Robila** (StudM'01-M'03) is Assistant Professor of Computer Science and Director of the Center for Imaging and Optics at Montclair State University, Montclair, NJ. Stefan received a B.S. in Computer Science from University of Iasi in 1997, followed by an M.S. in Computer Science and a Ph.D. in Computer Information Science in 2000 and 2002 respectively, both from Syracuse University. His current research interests are remote sensing, signal and image processing, multispectral / hyperspectral imagery, data mining and pattern recognition.

**Lukasz G. Maciak** is a graduate student at Montclair State University, working towards a M.S. in Computer Science. Lukasz received a B.S. in Computer Science from Montclair State University in 2004. He is currently doing research in hyperspectral imaging and Feature Extraction.