Proceedings of the 16th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2016
4–8 July, 2016.

Multi-core Implementation of Spatial-Spectral Preprocessing for Hyperspectral Unmixing

Luis Ignacio Jiménez[1], Sergio Bernabé[2], Carlos García[2], Javier Plaza[1], Gabriel Martín[3], Sergio Sánchez[1] and Antonio Plaza[1]

[1] *Department of Computer Technology and Communications, University of Extremadura*

[2] *Department of Computer Architecture and Automation, Complutense University of Madrid*

[3] *Instituto de Telecomunicacões, Lisbon*

emails: luijimenez@unex.es, sebernab@ucm.es, garsanca@ucm.es, jplaza@unex.es, gabriel.hernandez@lx.it.pt, sersanmar@unex.es, aplaza@unex.es

Abstract

Spectral unmixing pursues the identification of spectrally pure constituents, called *endmembers*, and their corresponding *abundances* in each pixel of a hyperspectral image. Most unmixing techniques have focused on the exploitation of spectral information alone. Recently, some techniques have been developed to take advantage of the complementary information provided by the spatial correlation of the pixels in the image. Computational complexity represents a major problem in these spatial-spectral techniques, as hyperspectral images contain very rich information in both the spatial and the spectral domains. In this letter, we develop a computationally efficient implementation of a spatial-spectral processing (SSPP) algorithm that has been successfully applied prior to spectral unmixing of hyperspectral data. Our implementation has been optimized for multi-core processors, and is evaluated (using both synthetic and real data) using an 2×Intel Xeon processor E5-2670 at 2.60GHz. Significant speedups can be achieved when processing hyperspectral images of different sizes. This allows for the inclusion of the proposed parallel preprocessing module in a full hyperspectral unmixing chain able to operate in real time.

Key words: Hyperspectral unmixing, spatial-spectral preprocessing (SSPP), OpenMP, multi-core processors.

 ISBN: 978-84-608-6082-2

1 Introduction

In hyperspectral unmixing, endmember extraction is the process of collecting pure signature spectra of the materials present in a remotely sensed hyperspectral scene. These pure signatures are then used to decompose the scene into a set of so-called abundance fractions representing the coverage of each endmember in each image pixel.

Several algorithms have been developed for automatic or semi-automatic identification of endmembers over the last decade [1] which majority have been developed under the pure pixel assumption, i.e., they assume that the remotely sensed data contain one pure observation for each different material in the scene [2]. Most of these algorithms rely exclusively on the exploitation of spectral information in order to select the final set of endmembers.

In order to include also the spatial information, several spatial preprocessing algorithms have been developed that can be applied prior to any spectral-based endmember extraction technique. Techniques include the spatial preprocessing (SPP) [3], region-based spatial preprocessing (RBSPP) [4], and spatial-spectral preprocessing (SSPP) [5]. The goal of these preprocessing methods is to guide the search for endmembers using not only spectral but also spatial information, which greatly assists in the selection of more spatially representative endmembers without the need to modify the endmember identification algorithm (the preprocessing can be applied as an optional step). As consequence, the spatial preprocessing increase the computational cost to the full spectral unmixing chainmaking efficient implementations for spatial preprocessing techniques an important goal.

In this work, we present a new parallel implementation of the SSPP algorithm, which has been shown as one of the most successful spatial preprocessing techniques available in the literature [5]. Our implementation has been developed for multi-core architectures where real and synthetic scenes are used to validate the efficacy of the implementation.

The remainder of this manuscript is organized as follows. Section 2 enumerates and describes the different steps of the SSPP method. Section 3 describes the proposed parallel implementation for multi-core processors. Section 4 describes the experiments conducted using real and synthetic data sets intended to evaluate the acceleration achieved by our parallel implementation. Section 5 concludes the paper with some remarks and hints at plausible future research lines.

2 Spatial-Spectral Preprocessing

This section briefly outlines the SSPP algorithm in [5]. As shown in the flowchart given in Fig. 1, the SSPP method consists of the following steps:

Multi-scale Gaussian filtering. This step takes as input the original hyperspectral image $\mathbf{Y}$ and returns a filtered version of the image. To perform this step, we first apply Gaussian filtering to each of the B spectral bands of the hyperspectral image. This

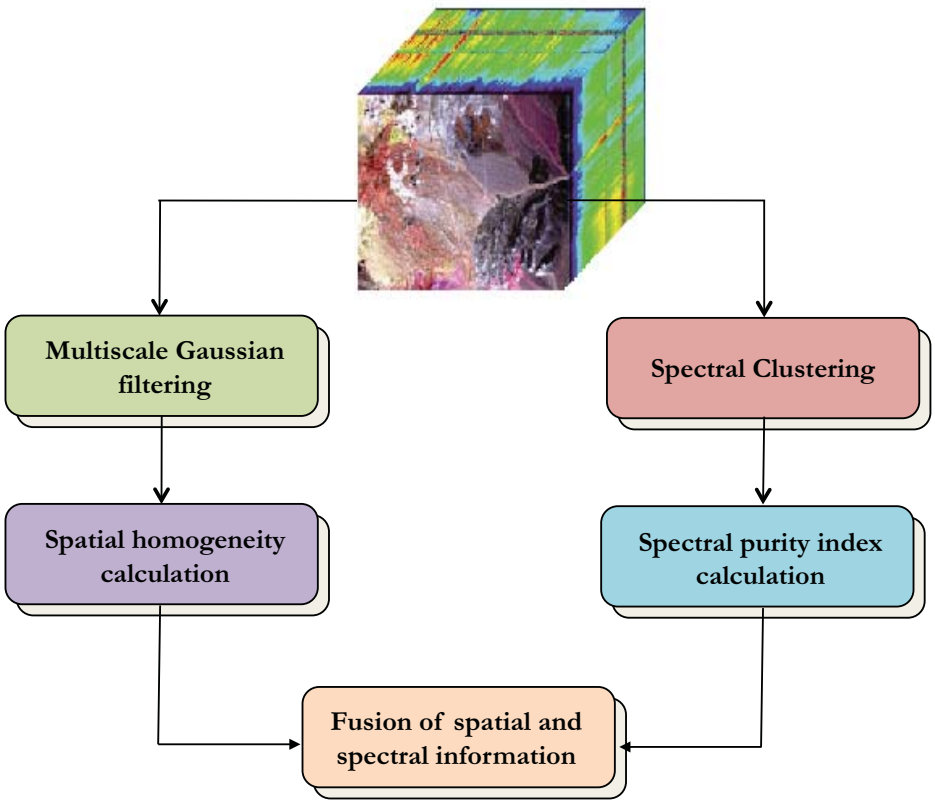

Figure 1: Block diagram illustrating the spatial-spectral preprocessing (SSPP) method.

 ISBN: 978-84-608-6082-2

results in a filtered version $\mathbf{Y}_F$ of the original hyperspectral image. Let us denote by $\mathbf{y}(i,j) = [y_1(i,j), y_2(i,j), \cdots, y_B(i,j)]$ the B-dimensional pixel vector at spatial coordinates (i,j) of the hyperspectral image $\mathbf{Y}$, which can now be defined as a set $\mathbf{Y} = \{\mathbf{y}(i,j)\}_{i \in 1,\cdots,r}$. Eq. (1) shows the pixel-level operation that we perform for each k-th spectral band of the hyperspectral image, with $1 \leq k \leq B$:

$$F_k[\mathbf{y}(i,j)] = \sum_{i'=1}^{r} \sum_{j'=1}^{c} G(i-i', j-j') \cdot y_k(i',j'), \text{ with } G(i',j') = \frac{1}{2\pi\sigma^2} e^{-\frac{i'^2+j'^2}{2\sigma^2}}. \quad (1)$$

Spatial homogeneity calculation. This step takes as input the filtered hyperspectral image obtained in the previous step and produces a spatial homogeneity index for each pixel in the original image $\mathbf{Y}$. To perform this step, we first calculate the root mean square error (RMSE) [6] between the original hyperspectral image and the filtered image. Eq. (2) indicates the operation to calculate the RMSE between the pixel $\mathbf{y}(i,j)$ in the original image and the pixel at the same spatial coordinates, $\mathbf{y}_F(i,j)$, in the filtered image:

$$\text{RMSE}[\mathbf{y}(i,j), \mathbf{y}_F(i,j)] = \left(\frac{1}{B} \sum_{k=1}^{B} (y_k(i,j) - y_{F_k}(i,j))^2 \right)^{\frac{1}{2}}. \quad (2)$$

The lower the RMSE score, the higher the similarity between the pixel in the original image and its neighbors. Quite opposite, the higher the RMSE, the lower the similarity of the pixel in the original image with regards to its neighbours. As a result, the RMSE in Eq. (2) can be used as a spatial homogeneity index for each pixel $\mathbf{y}(i,j)$ in the hyperspectral image $\mathbf{Y}$.

In the *spectral purity index calculation* step, we first use principal component analysis (PCA) [7] to reduce the dimensionality of the hyperspectral image, retaining the first p principal components (PCs) containing most of the variance in the data. Then, we use the first PCs as the skewers for which we identify the pixels with maxima and minima projection values, following a procedure similar to the one adopted by the pixel purity index (PPI) algorithm in [8]. The pixels with maxima and minima projection values are assigned a weight of 1. The weight of the mean value between the maxima and minima projection value is 0. A threshold value is also applied so that the weights lower than this threshold are assigned the value 0. Finally the spectral purity is calculated as the sum of all the weights over the first p PCs.

In the *spectral clustering*, we perform a spectral-based unsupervised clustering of the original hyperspectral image. This step, which is applied separately from the previous steps, uses the K-Means algorithm [9] in order to identify p clusters in the hyperspectral image.

Fusion of spatial and spectral information is a step that takes as input the spatial homogeneity index calculated in the second step and the clusters calculated in fourth step,

and returns a subset of candidate pixels in the original hyperspectral image which will be used for endmember identification purposes. For each cluster, a subset of spatially homogeneous and spectrally pure pixels is selected. To do so, pixels in each cluster are ranked according to increasing values of their spatial homogeneity and spectral purity.

Finally, an endmember extraction algorithm can be applied to the pixels retained after the procedure above. The outcome of the process is a set of p endmembers and their corresponding fractional abundance maps (one per endmember).

3 Parallel Implementation

The parallel implementation of SSPP has been developed using OpenMP which is an API used to explicity address multithreaded, shared-memory parallelism. In OpenMP the users specify the region in the code that are suitable for parallel implementation using pragmas and clauses supported in the gnu or Intel compilers. In the following, we briefly summarize the main techniques used in the multi-core implementation of the considered algorithm:

1. *Multi-scale Gaussian filtering.* We have developed an optimization where the most consuming part using the convolution of two 1-dimensional filters is the central part. This optimization consists on declare a *#pragma omp parallel for schedule (static, 32)* where the loop is divided into 32 equal-size chunks and also, an unrolling is developed to vectorize this central part to calculate the gaussian filtering. We have empirically tested that, if the size of the secene is largest, this value should be increased.

2. *Spatial homogeneity calculation.* The RMSE is required to calculate the spatial homogeneity index for each pixel in the data set. A reduction process for each pixel is computed where *#pragma omp simd* is applied to vectorize the reduction whose main loop is based on each spectral band. Finally, for each pixel we have used a *#pragma omp parallel for* to divide the loop iterations between the spawned threads and compute the square for each pixel in the scene.

3. *Spectral purity index calculation.* The PCA operation is required to calculate the spectral purity index. First of all, we need to calculate the normalized image obtained by substracting the average of all pixels in the scene to each pixel in the original image. For this purpose, the reduction process is performed using the *#pragma omp parallel for private (mean)*, where *mean* is the average of all pixels in the scene and later. For this calculation, *#pragma omp simd* is used to vectorize the loop. The same strategy is applied to substract the *mean* value to each pixel in the data set. After that, *mkl and lapack* are used to compute matrix multiplications to obatin the reduced image.

4. *Spectral clustering and fusion of spatial and spectral information.* For both steps, we have not applied any parallel technique to accelerate the process because the

 ISBN: 978-84-608-6082-2

processing time is lower.

4 Experimental validation

The experiments are carried out using a collection of 24 synthetic hyperspectral images simulated with different sizes (10000 to 200000 pixels) and number of endmembers (10 to 30). The signatures are obtained from the USGS library and the scenes are generated using the procedure described in [10] to simulate natural spatial patterns. These images comprise 224 narrow spectral bands between 0.4 to 2.5 μm. On the other hand, we have used the well-known AVIRIS Cuprite scene, collected by the Airborne Visible Infra-Red Imaging Specrometer (AVIRIS) in the summer of 1997 and available online in reflectance units after atmospheric correction. The portion comprises a relatively large area (350 lines by 350 samples and 20-m pixels) and 224 spectral bands between 0.4 to 2.5 μm and a total size of around 46 MB. Bands 1-3, 105-115, and 150-170 were removed prior to the analysis due to water absorption and low SNR in those bands prior to the analysis.

In order to evalute the performance, the proposed multi-core implementation has been tested on the following platform: 2×Intel Xeon processor E5-2670 with 8 cores each, at 2.60 GHz and 32 GB of DDR3 RAM memory. The Figs. 2 and 3 show the processing times considering different number of endmembers and execution threads. As can be seen, increasing the number of threads allows a better parallel performance and speedup respect to our single-threaded optimized implementation. On the other hand, Fig. 4 shows the execution times obtained for the AVIRIS Cuprite scene.

For illustrative purposes, Table 1 shows the timming results and speedups for each data set used in the experiments. As shown in Table 1, our data sets could be processed with significant speedup factor using the Intel Xeon, up to 3 times.

Table 1: Mean execution times (in seconds) and speedups (in the parentheses) for the best multi-core setting using real and synthetic scenes with different sizes and endmembers.

Image	**10 endmembers**	**20 endmembers**	**30 endmembers**
100×100 - 8 threads	0.0697 (1.68)	0.0697 (1.68)	0.0704 (1.72)
200×100 - 8 threads	0.1136 (1.81)	0.1101 (1.81)	0.1120 (1.82)
300×100 - 8 threads	0.1576 (1.85)	0.1490 (1.93)	0.1483 (1.94)
400×100 - 8 threads	0.1805 (2.02)	0.1830 (1.89)	0.1801 (2.06)
100×500 - 8 threads	0.2112 (2.13)	0.2027 (2.12)	0.2034 (2.07)
200×100 - 8 threads	0.3649 (2.25)	0.3582 (2.22)	0.3591 (2.20)
300×100 - 8 threads	0.5104 (2.26)	0.5028 (2.18)	0.5115 (2.22))
400×100 - 8 threads	0.6494 (2.39)	0.6568 (2.38)	0.6729 (2.39)
AVIRIS Cuprite - 14 threads	0.3579 (2.72) - 19 endmembers		

 ISBN: 978-84-608-6082-2

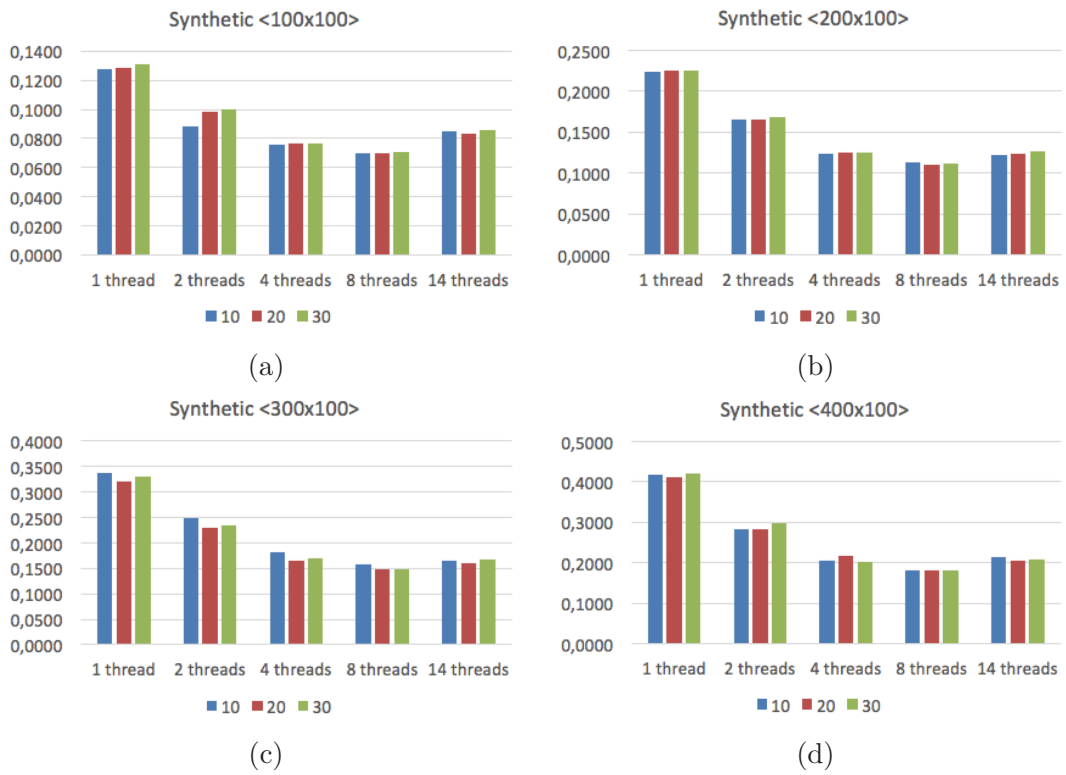

Figure 2: Execution times (seconds) for the multi-core version of the SSPP algorithm, considering different number of endmembers and sizes: (a) 100×100 pixels. (b) 200×100 pixels. (c) 300×100 pixels. (d) 400×100 pixels.

 ISBN: 978-84-608-6082-2

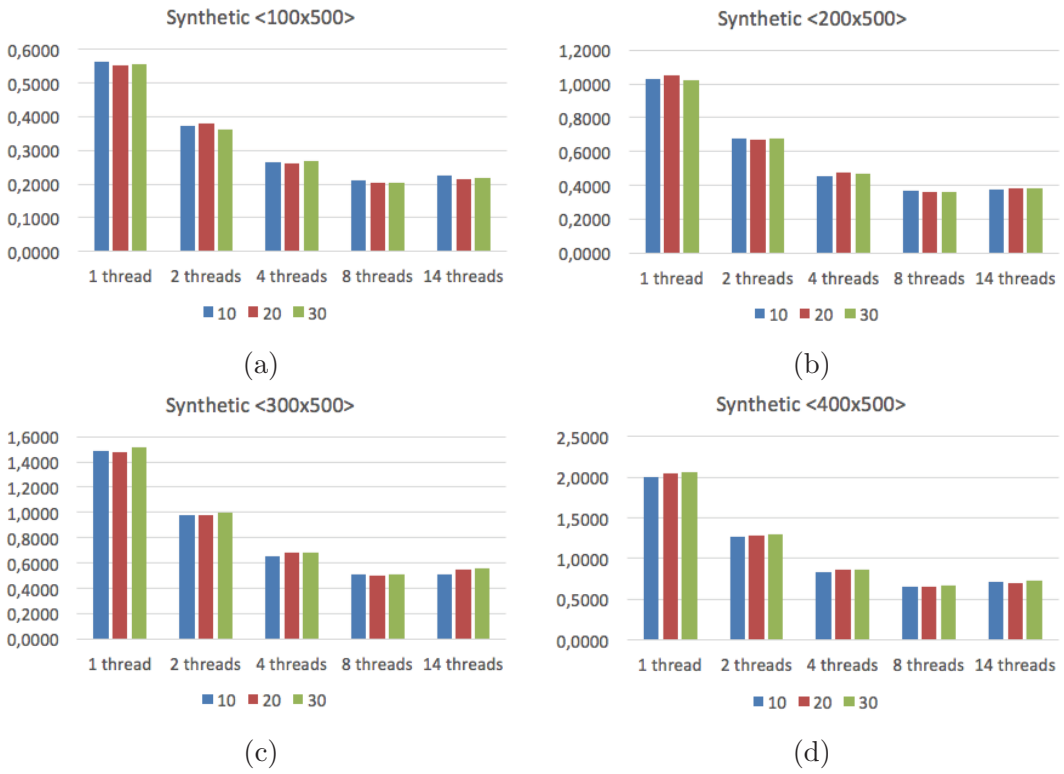

Figure 3: Execution times (seconds) for the multi-core version of the SSPP algorithm, considering different number of endmembers and sizes: (a) 100×500 pixels. (b) 200×500 pixels. (c) 300×500 pixels. (d) 400×500 pixels.

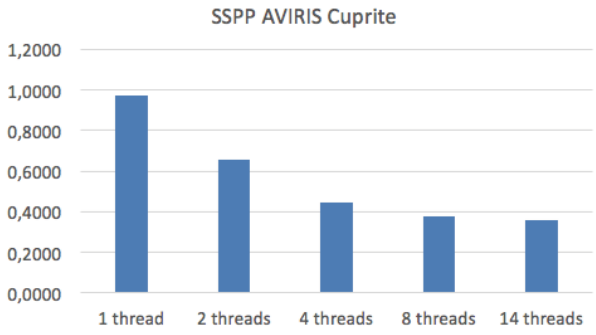

Figure 4: Execution times (seconds) for the multi-core version of the SSPP algorithm, considering the AVIRIS Cuprite scene.

5 Conclusions

A multi-core implementation of the SSPP algorithm for spectral unmixing has been proposed. The optimized version exploits different strategies using OpenMP: vectorization and shared memory between threads. The obtained results indicate that significant performance could be obtained using an Intel Xeon processor E5-2670 platform. Further experimentation with additional real scenes and a comparison with another programming languages are desirable in future research developments.

Acknowledgements

This work has been supported by Junta de Extremadura (decreto 297/2014, ayudas para la realización de actividades de investigación y desarrollo tecnológico, de divulgación y de transferencia de conocimiento por los Grupos de Investigación de Extremadura, Ref. GR15005) and by the Formación Posdoctoral programme (FPDI-2013-16280). This work was partially supported by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). CETA-CIEMAT belongs to CIEMAT and the Government of Spain. Funding from the Spanish Ministry of Economy and Competitiveness (MINECO) through the research contracts TIN2012-32180 and TIN2015-65277-R are also gratefully acknowledged.

References

[1] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 5, no. 2, pp. 354–379, 2012.

[2] J. Plaza, E. M. T. Hendrix, I. Garcia, G. Martin, and A. Plaza, "On endmember identification in hyperspectral images without pure pixels: A comparison of algorithms," *Journal of Mathematical Imaging and Vision*, vol. 42, no. 2-3, pp. 163–175, 2012.

[3] M. Zortea and A. Plaza, "Spatial preprocessing for endmember extraction," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 8, pp. 2679–2693, 2009.

[4] G. Martin and A. Plaza, "Region-based spatial preprocessing for endmember extraction and spectral unmixing," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 745–749, 2011.

[5] G. Martin and A. Plaza., "Spatial-spectral preprocessing prior to endmember identification and unmixing of remotely sensed hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 5, no. 2, pp. 380–395, 2012.

[6] N. Keshava and J. F. Mustard, "Spectral unmixing," *Signal Processing Magazine, IEEE*, vol. 19, no. 1, pp. 44–57, 2002.

[7] J. A. R. y X. Jia, "Remote Sensing Digital Image Analysis," in *Springer- Verlag*, 1999.

[8] J. W. Boardman, F. A. Kruse, and R. O. Green, "Mapping Target Signatures Via Partial Unmixing of Aviris Data," *Proc. JPL Airborne Earth Sci. Workshop*, pp. 23–26, 1995.

[9] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.

[10] G. S. Miller, "The definition and rendering of terrain maps," in *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4. ACM, 1986, pp. 39–48.

 ISBN: 978-84-608-6082-2