Spatial-Spectral Preprocessing for Endmember Extraction on GPU's

Luis I. Jiménez[a], Javier Plaza[a], Antonio Plaza[a], and Jun Li[b]

[a]Hyperspectral Computing Laboratory, Department of Computer Technology and Communications, University of Extremadura, E-10071, Cáceres, Spain
[b]Department of Geography, Sun Yat-Sen University, Guangzhou, China

ABSTRACT

Spectral unmixing is focused in the identification of spectrally pure signatures, called *endmembers*, and their corresponding *abundances* in each pixel of a hyperspectral image. Mainly focused on the spectral information contained in the hyperspectral images, endmember extraction techniques have recently included spatial information to achieve more accurate results. Several algorithms have been developed for automatic or semi-automatic identification of endmembers using spatial and spectral information, including the spectral-spatial endmember extraction (SSEE) where, within a preprocessing step in the technique, both sources of information are extracted from the hyperspectral image and equally used for this purpose. Previous works have implemented the SSEE technique in four main steps: 1) local eigenvectors calculation in each sub-region in which the original hyperspectral image is divided; 2) computation of the maxima and minima projection of all eigenvectors over the entire hyperspectral image in order to obtain a candidates pixels set; 3) expansion and averaging of the signatures of the candidate set; 4) ranking based on the spectral angle distance (SAD). The result of this method is a list of candidate signatures from which the endmembers can be extracted using various spectral-based techniques, such as orthogonal subspace projection (OSP), vertex component analysis (VCA) or N-FINDR. Considering the large volume of data and the complexity of the calculations, there is a need for efficient implementations. Latest-generation hardware accelerators such as commodity graphics processing units (GPUs) offer a good chance for improving the computational performance in this context. In this paper, we develop two different implementations of the SSEE algorithm using GPUs. Both are based on the eigenvectors computation within each sub-region of the first step, one using the singular value decomposition (SVD) and another one using principal component analysis (PCA). Based on our experiments with hyperspectral data sets, high computational performance is observed in both cases.

Keywords: Hyperspectral imaging, spatial-spectral endmember extraction (SSEE), graphics processing units (GPUs)

1. INTRODUCTION

Spectral unmixing[1] is a very important technique for the exploitation of images in remote sensing. In recent years, many methods have been developed with the goal to identify the pure spectral signatures within the image (called endmembers) and and its pixel level corresponding abundance.[2] Most of the endmember extraction methods are focused only in the spectral information, which involves susceptibility to noise, atypical values and anomalous endmembers.[3] Adding spatial information to the process will lead to the endmember identification to spatially homogeneous areas, which are expected to contain the most spectrally pure signatures within the scene.[4–6] With this goal, several spatial/spectral endmember identification methods have been developed for hyperspectral data.

Several algorithms have been developed with this purpose such as the *automatic morphological endmember extraction* (AMEE),[4] that uses morphological operators to discriminate the endmembers within an area, or the *spatial-spectral endmember extraction* (SSEE)[5] that uses the spatial information to increase the spectral contrast between spectrally similar endmembers but spatially independent.

Further author information: (Send correspondence to L.I.J.)
E-mail: L.I.J.: luijimenez@unex.es J.P.: jplaza@unex.es A.P.: aplaza@unex.es J.L.: jun@lx.it.pt

High-Performance Computing in Geoscience and Remote Sensing VI, edited by Bormin Huang, Sebastián López, Zhensen Wu, Jose M. Nascimento, Jun Li, Valeriy V. Strotov, Proc. of SPIE Vol. 10007, 100070K
 · doi: 10.1117/12.2241288

In the other hand, other spatial preprocessing methods have been used for endmember identification.[7–9] These methods have been thought to be included in the endmember extraction process when spectrally exclusive techniques are been used. Some of these preprocessing methods are the *Spatial preprocessing* (SPP),[7] which estimates, for each input scene pixel, a scalar value regarding the spatial similarity between that pixel and its neighborhood, defined by a spatial window around the pixel. An extension of this concept was presented in,[8] where the utilization of fixed spatial neighborhoods was replaced by the introduction of regions thought to characterize better the spatial context. More recent method is the *spatial and spectral preprocessing* (SSPP) developed in,[9] where the algorithm uses both, the spatial and spectral information, to locate a representative set of candidate pixels for each class within the image.

The techniques mentioned above for endmember identification based on spectral and spatial information are characterized by a high computational cost, due to the need to process both types of information contained in the hyperspectral images. However, because the calculations are similar, many of these techniques have been efficiently implemented using graphic processing units or GPU's. For instance the SPP algorithm was implemented for GPU's in.[10] The SSPP also has been implemented for GPU's in,[11] meanwhile the RBSPP is also susceptible to be parallelized (including the segmentation process that usually results in an irregular distribution in parallel implementations). The AMEE algorithm has been also implemented for GPU's in.[12] Although, the SSEE (a highly representative and successful method for spatial/spectral endmember extraction) have not been developed for graphics accelerators yet.

In this work an implementation of the SSEE algorithm[5] for GPU's has been developed. In this method, the hyperspectral image is processed looking for a set of signatures which can be considered candidates to be endmembers, among which, it will be performed the finally pure spectral signatures extraction. As consequence, the SSEE can be considered a spatial preprocessing algorithm. Decreasing the number of candidates to be endmembers the computational time consume for the subsequent endmember extraction algorithm can be reduce. A set of optimizations have been implemented in the SSEE algorithm which allow a efficient execution of the algorithm in GPU's using NVidia's *Compute Unified Device Architecture* (CUDA)* due to its high computational cost. Comparisons of the new proposed methods against already available parallel implementations of SPP and SSPP algorithms have been performed, revealing that those methods can be efficiently exploited in GPU's when the spatial information is included in the endmember extraction process.

The paper is structured as follows. Section 2 describes the SSEE algorithm, along with the different optimized versions. Section 3 presents the GPU implementation of the SSEE algorithm. The experiments performed to evaluate the accuracy of the signatures extracted and the performance obtained by the proposed parallel implementations are specified in Section 4. Finally, Section 5 ends this work with some remarks and possible future research lines.

2. SPATIAL-SPECTRAL ENDMEMBER EXTRACTION (SSEE)

The original SSEE algorithm can be resume as follows.[5] In first place, the original hyperspectral image is splitted in smaller pieces and the singular values decomposition (SVD) is used in order to obtain the eigenvectors that describe the spectral variance on each subset. This step of the algorithm is the most expensive in terms of computational time. The SSEE original implementation defines that the image subsets can not overlap each other.

After that, the whole image is projected over the aforementioned eigenvectors obtaining the maximal and minimal projection values which determines the initial candidate pixels.

To continue the candidate pixels set obtained in the previous step is extended based on the spectral similarity between the candidates within a window around each candidate and this step is followed by an averaging process using another window of the same size as the first. This process divides the endmembers by class in the spectral space. At this point, the SSEE algorithm has obtained a set of candidate pixels where the final endmembers are. As consequence, the SSEE algorithm produces similar outputs as the ones produces by a preprocessing method like the SPP or the SSPP.

*https://developer.nvidia.com/cuda-zone

In the last step, the SSEE algorithm lists all the candidate pixels sorted by the spectral distance to the first of the list. As is mentioned in the original article,[5] after this point it can be used a manual process or classical endmember extraction techniques based on spectral information like *orthogonal subspace projection* (OSP).[13]

As we have said before, the SSEE algorithm is characterized by a high computational cost. Th SVD process and the projections performed using the entire image represent most of this computational cost. Some optimizations have been focused in this part of the algorithm, specifically the one which tries to replace the SVD by the PCA[14] which many available parallel versions are highly efficient.

3. GPU IMPLEMENTATION

The SSEE algorithm GPU implementation has been developed using the NVidia CUDA programming language, with some calls to functions available in cuBLAS[†] and BLAS/LAPACK[‡] libraries. Coming up next GPU's general architecture and our specific proposed implementation are described.

The GPU implementation for the different stages of the SSEE method can be divided in four steps. The eigenvector calculation is where most of execution time spent by the SSEE algorithm is used . This step can be performed using the SVD or the PCA. The SVD has been implemented by means of the function *dgesvd* available in the BLAS/LAPACK library, which has been evaluated to be very fast in terms of computational time. In case of the GPU implementation of the PCA, some steps are performed. First, a kernel normalizes the image subset subtracting the average pixel in that subset. This kernel, called *avgXCUDA*, establishes a one-dimensional grid which size is set to the number of bands present in the image and the block size to the number of available threads in the GPU. To continue the cuBLAS library function *cublasDgemm* is called, and performs the product of the image subset and its transpose. Finally the SVD transformation is calculated in the CPU, because was empirically proved that the parallel implementation of this point is not needed to achieve better performance results.

Once the eigenvectors are calculated and transferred to the GPU memory, the function *cublasDgemm* is used again to perform the image projection. This step is very important in terms of computational cost mostly due to the complexity raise when the window used is bigger. For this step, a kernel called *maxminProjection* is used, being designed to select the maximal and minimal projections on each band in the image, using a reduction operation based on the percent of the total number of bands defined by a threshold value, *svd_th*, introduced as parameter. Here the number of blocks is equal to the number of bands and the number of threads is the maximum available in the device.

To perform this stage first to do is to calculate the euclidean norm of each pixel using a kernel called *euclideanNorm*. The number of thread dedicated to this kernel is the maximum available in the GPU, and the number of blocks is set as the ratio between the number of pixel and block size plus one. A second kernel called *expandCandidatePixels* has been developed in order to perform the expansion of the candidates set within the range that is set by the size of a window *ws*. Finally another kernel called *averageStep* calculates the spectral average of the candidates within the same range defined by a window around them. Both kernels set the a two-dimensional grid the rows and columns are dined by the rows and columns of the original image, and the number of threads is the maximum allowed by the GPU.

Last step is to list the candidate pixels base on the distance between them to the first of the set. In this work we have used a kernel called *BitonicSort* to sort this set using a bitonic sorting algorithm for GPUs. The number of threads is empirically set to 256 and the number of blocks is obtained by the next expression:

$$n\_blocks = \exp(\log_2 \lfloor \frac{n\_pixels}{n\_threads} \rfloor),$$

where n_blocks, n_pixels and $n\_threads$ denote respectively the number of blocks, the number of pixels and the number of threads. This last steps is optional and can be replaced by a classical endmember extraction method. Is also an interesting point that this last stage means no difference in terms of computational time if is compared to the total, even when the candidate set is very large.

[†]https://developer.nvidia.com/cublas
[‡]http://www.netlib.org/lapack/

4. EXPERIMENTAL RESULTS

The experiments were done using a subset of the image obtained by *Airborne Visible Infra-Red Imaging Spectrometer* (AVIRIS), operated by *NASA's Jet Propulsion Laboratory*, over the region of Cuprite in Nevada during the summer of 1997 (available online[§]). The portion used in the experiments corresponds to a 350×350 pixels subset, which comprises 188 spectral bands in the range between 400-2500 nm and 50MB. The mineral present in the image are well known and the reference spectral signatures are available in the library (*United States Geological Survey (USGS) library* [¶]) which is used in this work for an accuracy evaluation of the algorithms.

Two different sets of experiments were performed in this work in order to evaluate the quality of the candidate pixels set and the performance of the SSEE parallel implementations for GPUs.

4.1 Candidate Pixels Quality

To analyze the quality of the candidates selected by different methods (not only the SSEE, also the SPP and the SSPP) we have compared them against a set of spectral signatures extracted from the USGS library, using the spectral angle distance (SAD) as quantitative measure. It is recalled that the SSEE algorithm can be seen as a spatial preprocessing method, very similar to the SPP and SSPP. As consequence, a comparison between these methods is done in order to measure the quality of the selected candidates. With this purpose 25 signatures have been selected within the USGS library, which are shown in the Figure 1.

To evaluate the quality of the candidate pixels selected by each parallel implementation in relation to the reference set of signatures extracted from the USGS library, two matching algorithms are considered.[9] The first one obtains the average of the best matching between the two sets randomly sorted in each iteration for a total of 100, and its results are shown in the Table 1. This table includes the results for each version of the SSEE algorithm, the SPP and the SSPP. The second algorithm matches the signatures using the minimum spectral angle value between all possible combinations (see Table 2). In both cases the 25 mineral signatures are used as a reference set. With the intention of establishing a fair comparison is requires to compare the number of candidates that match with the reference set. May be difficult to decide when two signatures are well matched. In this work is considered a good match when the spectral angle distance value between two signatures is below 10° (where the worst case is when two signatures obtain 90°). Based on that, we used the next metric to represent the rate of success S:

$$S(\%) = \frac{\frac{m}{P} + [1 - \frac{n}{N}]}{2} * 100,$$

where n is the number of extracted candidates by each method, N is the total number of pixels present in the image, m is the number of signatures well matched and P is the total number of reference signatures. The Tables 1 and 2 include the results of this metric. Unlike the SSPP and the SSEE, the SPP do not reduce the number of candidate pixels to be endmembers whence this metric is not apply to it.

As is shown in Tables 1 and 2, several reference signatures are mismatched by some methods, for instance *Alunite HS295.3B* or *Chlorite SMR-13.a 104-150*. This is because the difficulty of ensuring that these same signatures would be present in the original image, taking into account that some minerals present in the image could not be in the reference set. Many algorithms are able to give some candidates that can be very similar, spectrally speaking, to the USGS library reference signatures, even considering that the threshold is 10°. We can conclude based on the information given by the Tables 1 and 2 that the SSEE algorithm gives good results in most cases, being slightly better the PCA version. Considering all methods the SPP are the best results but taking in account the smaller set of candidates which is provide by the SSEE when the PCA is used, would be interesting to check its impact to the performance what is spoken in the following section.

[§]http://aviris.jpl.nasa.gov
[¶]http://speclab.cr.usgs.gov

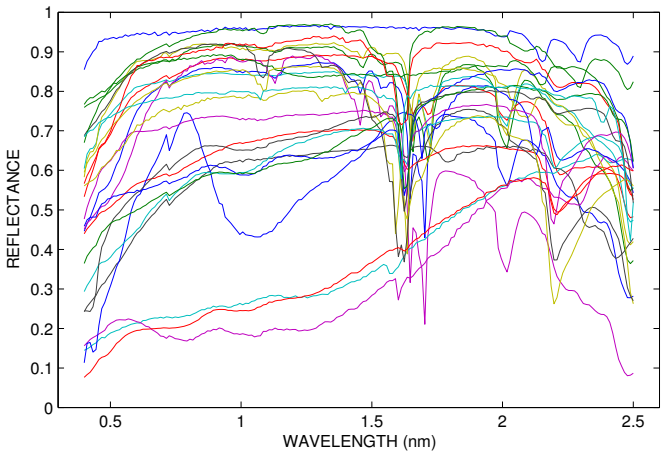

Figure 1. Set of 25 spectral signatures selected from the USGS library for the experiments: Alunite GDS84 Na03, Alunite GDS83 Na63, Alunite GDS82 Na82, Alunite AL706 Na, Alunite HS295.3B, Alunite SUSTDA-20, Buddingtonite GDS85 D-206, Calcite WS272, Calcite HS48.3B, Chalcedony CU91-6A, Chlorite HS179.3B, Chlorite SMR-13.a 104-150, Dickite NMNH106242, Halloysite NMNH106236, Jarosite GDS99 K-y 200C, Kaolinite KGa-1 (wxyl), Kaolinite KGa-2 (pxyl), Kaolin/Smect KLF506 95%K, Montmorillonite SWy-1, Montmorillonite SAz-1, Muscovite GDS107, Muscovite HS146.3B, Muscovite HS24.3, Nontronite GDS41, Pyrophyllite PYS1A fine g.

Table 1. Average spectral angle distance (in degrees) between the set of candidate pixels obtained by different spatial preprocessing methods (SSEE, SPP and SSPP) and the USGS reference signatures for the AVIRIS Cuprite image. In the case of SSEE and SPP, several window sizes are considered. The reported results correspond to ones obtained after 100 executions of the matching algorithm. The best results (lower angles) are highlighted in bold typeface.

	SSEE-SVD				**SSEE-PCA**				**SPP**				**SSPP**
Window Size	35x35	50x50	70x70	175x175	35x35	50x50	70x70	175x175	3x3	5x5	7x7	9x9	
Candidates	2416	1795	1310	403	204	140	77	28	122500	122500	122500	122500	3653
Average	8,138	8,184	8,930	10,928	7,144	7,756	8,734	11,279	**7,009**	7,098	7,170	7,171	8,255
S(%)	89,014	89,267	89,465	79,836	91,917	**91,943**	89,969	75,989	——	——	——	——	88,509

Table 2. Spectral angle distances (in degrees) between the set of candidate pixels obtained by different spatial preprocessing methods (SSEE, SPP and SSPP) and the USGS reference signatures for the AVIRIS Cuprite image. In the case of SSEE and SPP, several window sizes are considered. The reported results correspond to the matching resulting in minimum spectral angle. The best results (lower angles) are highlighted in bold typeface.

	SSEE-SVD				**SSEE-PCA**				**SPP**				**SSPP**
Window Size	35x35	50x50	70x70	175x175	35x35	50x50	70x70	175x175	3x3	5x5	7x7	9x9	
Candidates	2416	1795	1310	403	204	140	77	28	122500	122500	122500	122500	3653
Average	8,148	8,199	8,918	10,928	7,140	7,768	9,054	11,173	**7,013**	7,113	7,176	7,176	8,268
S(%)	89,014	89,267	89,465	79,836	**91,917**	89,943	89,969	81,989	——	——	——	——	88,509

4.2 Computational Performance

With the intention to evaluate the computational performance, the experiments were done in a desktop computer (Intel core i7 920 CPU a 2.67 GHz y 6 GB of RAM) with a NVidia GTX 580, with 512 cores working at 1.54 GHz. Serial versions were executed in one CPU core available, and the parallel times were measured based on each architecture resources. The average speedup has been calculated between each pair CPU/GPU after 10 executions. All GPU implementations mentioned above in this section obtain the same exact results as in serial versions.

Table 3. Execution times (seconds) and parallel performance (speedup) for the parallel versions of the different SSEE approaches, considering different window sizes, after executing the algorithm on NVidia GTX 580 after 10 Monte Carlo runs.

SSEE-SVD	**STEP1**	**STEP2**	**STEP3**	**STEP4**	**TOTAL**	**SPEEDUP**
35x35	6,5675	0,1060	0,5728	0,0067	7,2530	15,1616
50x50	4,4758	0,1140	0,6188	0,0051	5,2137	11,4076
70x70	3,2228	0,1158	0,6992	0,0044	4,0422	9,1843
175x175	3,4194	0,1095	0,9062	0,0020	4,4371	5,1112
SSEE-PCA	**STEP1**	**STEP2**	**STEP3**	**STEP4**	**TOTAL**	**SPEEDUP**
35x35	8,5853	0,0228	0,1381	0,0011	8,7473	11,4655
50x50	4,1857	0,0303	0,1395	0,0010	4,3565	11,4519
70x70	2,2115	0,0320	0,1335	0,0008	2,3778	11,0915
175x175	0,4972	0,0310	0,1584	0,0010	0,6876	8,9735

The Table 3 shows the performance obtained with the SSEE parallel implementations considering the aforementioned architecture (for SPP and SSPP results, we refer to previous work[10,11] respectively where these methods are better described). We can conclude that the PCA version of the SSEE algorithm is faster when the window size used is smaller, which in fact is more important because the candidates obtained are more spatially independent. The SVD version is limited due is executed in the CPU and its parallel version does not worth unless the data volume would be very large.[15] If the accuracy results of subsection 4.1 are taken in account, we can say that the PCA version clearly improves the SVD version results.

5. CONCLUSIONS

In this work a new implementation of the *spatial-spectral endmember extraction* (SSEE) algorithm has been presented as a spatial preprocessing method capable to select candidate pixels for a subsequent endmember extraction process. Two different implementations have been considered, based on the method used to calculate the eigenvectors. The proposed GPU implementation has been compared against other spatial preprocessing methods, such as, the SPP and the SSPP in terms of accuracy and computational performance. Based on the obtained results, we can conclude that the SSEE implementations can compete with the other preprocessing algorithms without lose quality in the candidate pixels set where the final endmembers are going to be extracted. Future works will be focused in to include new spatial preprocessing algorithms to the comparison and the development of new parallel strategies using GPUs.

ACKNOWLEDGMENTS

This work has been supported by *Junta de Extremadura* (through Decree 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).

REFERENCES

[1] Keshava, N. and Mustard, J. F., "Spectral unmixing," *IEEE Signal Process. Mag.* **19**(1), 44–57 (2002).

[2] Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., and Chanussot, J., "Hyperspectral unmixing overview: Geometrical, statistical and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.* **5**(2), 354–379 (2012).

[3] Plaza, A., Martinez, P., Perez, R., and Plaza, J., "A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data," *IEEE Trans. Geosci. Remote Sens.* **42**(3), 650–663 (2004).

[4] Plaza, A., Martinez, P., Pérez, R., and Plaza, J., "Spatial/spectral endmember extraction by multidimensional morphological operations," *Geoscience and Remote Sensing, IEEE Transactions on* **40**(9), 2025–2041 (2002).

[5] Rogge, D. M., Rivard, B., Zhang, J., Sanchez, A., Harris, J., and Feng, J., "Integration of spatial–spectral information for the improved extraction of endmembers," *Remote Sens. Environ.* **110**(3), 287–303 (2007).

[6] Lopez, S., Moure, J. F., Plaza, A., Callico, G. M., Lopez, J. F., and Sarmiento, R., "A new preprocessing technique for fast hyperspectral endmember extraction," *IEEE Geoscience and Remote Sensing Letters* **10**, 1070–1074 (September 2013).

[7] Zortea, M. and Plaza, A., "Spatial preprocessing for endmember extraction," *IEEE Trans. Geosci. Remote Sens.* **47**, 2679–2693 (2009).

[8] Martin, G. and Plaza, A., "Region-based spatial preprocessing for endmember extraction and spectral unmixing," *IEEE Geosci. Remote Sens. Lett.* **8**(4), 745–749 (2011).

[9] Martin, G. and Plaza., A., "Spatial-spectral preprocessing prior to endmember identification and unmixing of remotely sensed hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.* **5**(2), 380–395 (2012).

[10] Delgado, J., Martin, G., Plaza, J., Jimenez, L. I., and Plaza, A., "Fast spatial preprocessing for spectral unmixing of hyperspectral data on graphics processing units," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **9**(2), 952–961 (2016).

[11] Jimenez, L. I., Martin, G., Sanchez, S., Plaza, J., and Plaza, A., "GPU implementation of spatial-spectral preprocessing for hyperspectral unmixing," *IEEE Geoscience and Remote Sensing Letters* (submitted, 2016).

[12] Setoain, J., Prieto, M., Tenllado, C., Plaza, A., and Tirado, F., "Parallel morphological endmember extraction using commodity graphics hardware," *IEEE Geoscience and Remote Sensing Letters* **43**(3), 441–445 (2007).

[13] Harsanyi, J. C. and Chang, C.-I., "Hyperspectral image classification and dimensionality reduction: An orthogonal subspace projection approach," *IEEE Trans. Geosci. Remote Sens.* **32**, 779–785 (1994).

[14] Jolliffe, I., [*Principal component analysis*], Wiley Online Library (2002).

[15] Lahabar, S. and Narayanan, P. J., "Singular value decomposition on GPU using CUDA," *Proceedings of the IEEE Parallel and Distributed Processing Symposium* , 1–10 (2009).