PARALLEL DATA COMPRESSION FOR HYPERSPECTRAL IMAGERY

He Yang [1], *Qian Du* [1], *Wei Zhu* [1], *Ioana Banicescu* [2], *and James E. Fowler* [1]

[1] Department of Electrical and Computer Engineering, GeoResources Institute
[2] Department of Computer Science and Engineering
Mississippi State University, Mississippi State, MS 39762, USA

ABSTRACT

The high dimensionality of hyperspectral imagery challenges image processing and analysis. It has been shown that hyperspectral compression can be achieved by principal component analysis (PCA) for spectral decorrelation followed by the JPEG2000-based coding. This approach, referred to as PCA+JPEG2000, provides superior rate-distortion performance and can preserve useful data information. However, its main disadvantage is high computational complexity in the PCA process which entails the calculation of the data covariance matrix and its eigenvectors. Parallel processing is an appropriate approach to relieve the computation burden of such a PCA-based compression. In this paper, several parallel PCA implementations are proposed and their processing speed and resulting compression performance are investigated.

1. INTRODUCTION

A hyperspectral image is collected with hundreds of spectral channels for the same spatial area on the Earth. Due to its vast data volume, it is desirable to apply parallel processing to hyperspectral image processing and analysis when parallel computing facilities are available [1-4].

Since the data is highly correlated both spatially and spectrally, hyperspectral image-compression techniques are used to reduce data volume without losing information that is useful to interpret the scene imaged. Previously, it has been shown that the principal component analysis (PCA) in conjunction with JPEG2000 can provide superior rate-distortion performance, where the PCA is for spectral decorrelation and JPEG2000 is applied for the coding of the principal components (referred to as PCA+JPEG2000). In particular, PCA+JPEG2000 outperforms DWT+JPEG2000, the corresponding technique wherein a discrete wavelet transform (DWT) is adopted for spectral decorrelation. Spectral decorrelation is thus critical to hyperspectral compression, and PCA outperforms the DWT in this respect. We also found that the best rate-distortion performance is provided when a subset of PCs is maintained for compression rather than all the PCs [5]; we refer to this approach as SubPCA+JPEG2000.

However, the main disadvantage of PCA-based compression is the high computational complexity associated with the calculation of the covariance matrix, $\mathbf{\Sigma}$, and its eigenvectors. Some low-complexity schemes are proposed in [6-7]. In this paper, we develop several parallel algorithms for PCA+JPEG2000 and SubPCA+JPEG2000 to further increase the processing speed.

2. PARALLEL PROCESSING ALGORITHMS

Let a hyperspectral image be of size $L \times M \times N$, where L is the number of bands and M and N are the number of rows and columns in each band, respectively. To implement PCA+JPEG2000 with serial processing, the eigenvectors of $\mathbf{\Sigma}$ are computed and sent to the JPEG2000 encoder. The encoder then applies the spectral transform to the hyperspectral pixel vectors and embeds the spectral transform matrix and data mean vector $\mathbf{m}$ into the JPEG2000 bitstream. The JPEG2000 encoder will automatically allocate rate simultaneously to all codeblocks in all PCs by optimally truncating the bitstream for each codeblock. The decoder will extract the transform matrix and data mean vector $\mathbf{m}$ for the decoding process.

In the PCA transform, the cost of computing $\mathbf{\Sigma}$ is $O(MNL^2)$ and the cost of its eigendecomposition is $O(L^3)$. Since $MN >> L$, the computational burden is mainly from the calculation of $\mathbf{\Sigma}$.

2.1. Data Partitioning Strategies

A hyperspectral image cube can be partitioned either spatially or spectrally in order to reduce computation. Since the high computational complexity is due mainly to large spatial size (MN), it is intuitive to apply spatial partitioning. Since most of our data-processing and analysis algorithms are based on spectral information, this partitioning strategy provides complete spectral information (and no communication spectrally is needed) during processing, a characteristic that is important to most spectral-analysis methods [1]. For parallel processing under spatial partitioning, a hyperspectral image is divided evenly into several three-dimensional (3D) blocks and each block is

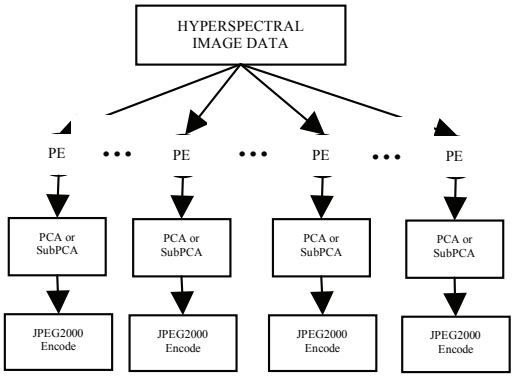

Figure 1. Parallel Algorithm 1.

distributed to a separate processing element (PE) for processing.

2.2. Parallel Processing Algorithms

Algorithm 1 — Block-based Parallel Compression

In this approach, the entire compression algorithm is parallelized; i.e., PCA+JPEG2000 and SubPCA+JPEG2000 are performed in their entirety in each block using a separate PE. Fig. 1 illustrates the processing arrangement for this approach.

Algorithm 2 — Parallel $\boldsymbol{\Sigma}$ *Computation*

In this approach, only the computation of $\boldsymbol{\Sigma}$ is parallelized, a strategy which is similar to that of [8]. However, in [8], local means must be computed first in PEs to determine the global mean. Afterwards, the local covariance matrix $\boldsymbol{\Sigma}'$ of each block is calculated in the PEs to find the global $\boldsymbol{\Sigma}$. To simplify the computing process, we use the fact that $\boldsymbol{\Sigma} = \mathbf{R} - \mathbf{m}\mathbf{m}^T$, where $\mathbf{R}$ is the data correlation matrix without mean removal. As illustrated in Fig. 2, local correlation matrix $\mathbf{R}'$ and local mean $\mathbf{m}'$ of each block are calculated, then the global mean $\mathbf{m}$ and global correlation matrix $\mathbf{R}$ can be found by merging the local results. Finally, $\mathbf{m}$ and $\mathbf{R}$ are used to determine the overall covariance matrix $\boldsymbol{\Sigma}$. Eigendecomposition of $\boldsymbol{\Sigma}$ is conducted followed by the other compression steps.

Algorithm 3 —Parallel Eigenvector Computation

In this approach, only the computation of $\boldsymbol{\Sigma}$ and its eigendecomposition are parallelized. As illustrated in Fig. 3, the covariance matrix $\boldsymbol{\Sigma}'$ of each block is calculated using a PE, then the eigendecomposition is performed to find all the eigenvectors for each $\boldsymbol{\Sigma}'$. Finally, eigenspace merging [9] is employed to determine the eigenvectors of the entire data covariance matrix $\boldsymbol{\Sigma}$ followed by compression of the entire dataset.

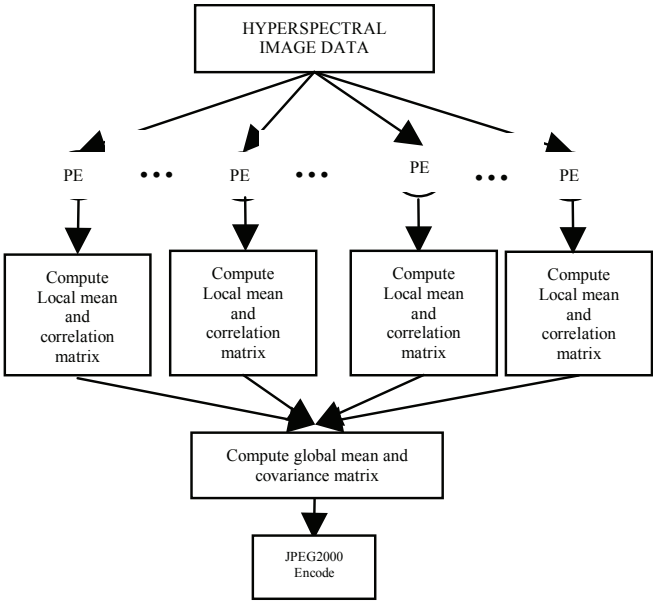

Figure 2. Parallel Algorithm 2.

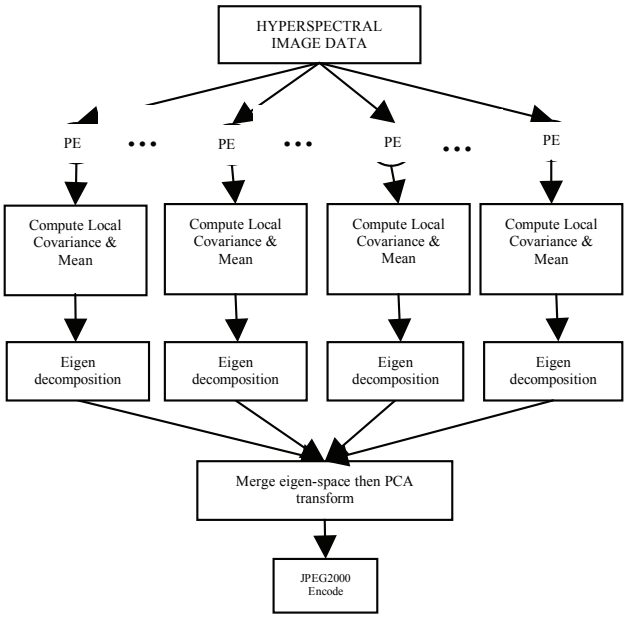

Figure 3. Parallel Algorithm 3.

Algorithm 4 —Parallel Partial Eigenvector Computation

As a final parallelization strategy, consider Fig. 4 wherein the computation of $\boldsymbol{\Sigma}$ and its partial eigendecomposition are parallelized. In other words, the $\boldsymbol{\Sigma}'$ of each block is calculated using a PE, then the eigendecomposition is performed to find the major eigenvectors for each $\boldsymbol{\Sigma}'$ (for SubPCA+JPEG2000). Afterwards, eigenspace merging [9] is employed to find the major eigenvectors for the global covariance matrix $\boldsymbol{\Sigma}$ followed by the other compression steps.

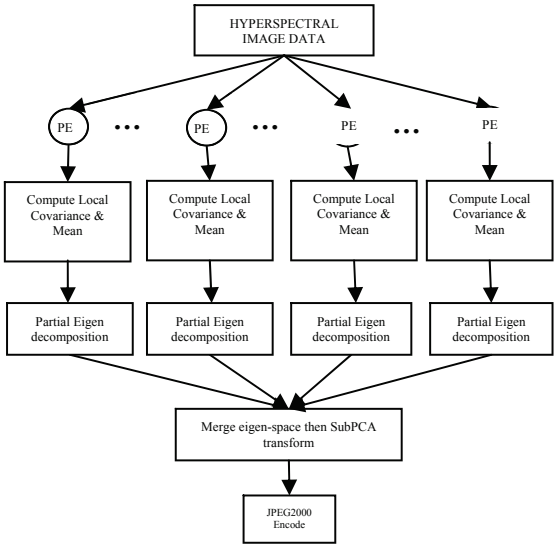

Figure 4. Parallel Algorithm 4.

4. EXPERIMENTAL RESULTS

4.1. Parallel Computer and Dataset

The parallel computer used in the experiment is a 2048-processor cluster based on 2.6-GHz AMD Opteron 2218 dual-core processors running the Linux operating system. Each node has 8 GB of RAM, and the system is constructed with Sun Microsystems SunFire X2200 M2 servers. All the computing nodes are diskless and connected with gigabit Ethernet between the 32 nodes in each rack, with 10-gigabit Ethernet between each of the 16 racks.

We use the Jasper Ridge dataset which was collected by the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). This 16-bit radiance dataset has been cropped spatially to a size of 512 × 512, and it is composed of 224 spectral bands.

The parallel algorithms were implemented in the C++ programming language using the message passing interface (MPI) [10]. Note that Algorithms 1 and 2 apply to both PCA+JPEG2000 and SubPCA+JPEG2000, while Algorithm 3 is applicable to PCA+JPEG2000 only and Algorithm 4 to SubPCA+JPEG2000 only. Therefore, there are six cases to be compared to measure parallel speedup and compression performance. We use Kakadu Version 5.2 as the implementation of JPEG2000.

4.2. Results

Parallel Performance for PCA only

Fig. 5 shows the speedup performance without including the Kakadu JPEG2000 encoder, where the speedup factor is defined as the ratio between the time spent on solving a problem with a single PE and the time spent with p PEs.

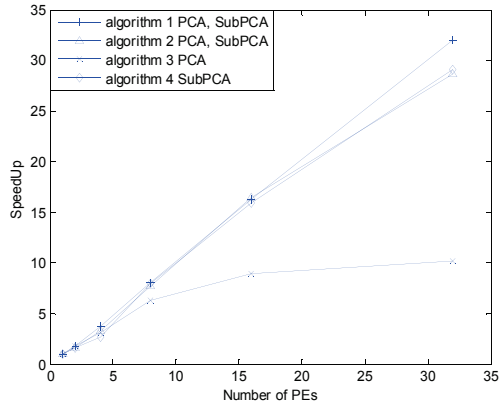

Figure 5. Performance of parallel PCA.

We can see that Algorithm 1 achieves the best speedup, obtaining linearly increasing speedup with increasing number of PEs; the performances of Algorithms 2 and 4 are similar and better than that of Algorithm 3. Since Algorithm 3 features fully parallelized eigendecomposition while Algorithm 4 performs only partial eigendecomposition, Algorithm 3 shows inferior speedup performance due to the cost of communication as needed for the global eigenspace merging.

Parallel Performance for the Entire Compression Process

To investigate the scaling properties of the parallelization including the JPEG2000 encoder, Fig. 6 plots the speedup factors versus the number of PEs. As the encoding time is stable at different bitrates (expressed in terms of bits per pixel per band (bpppb)), only the curve for 0.1 bpppb is shown here.

We see that Algorithm 1 achieves the best speedup performance while Algorithms 2 and 3 almost have no speedup when applied for PCA+JPEG2000. Algorithms 2 and 4 have medium performance when applied for SubPCA+JPEG2000, and when a parallel algorithm is used for SubPCA, the parallel speedup performance is always better than that of PCA.

The SNR performance on compression is not changed in Algorithms 2, 3, and 4 as compared to their serial counterparts, since the same transform matrices are obtained in each case. However, Algorithm 1 shows good speedup performance, but the true scalability is limited in the compression performance, since SNR is decreased as the number of PEs are increased as illustrated in Figs. 7 and 8. When using Algorithm 1, SubPCA has better parallel performance than PCA. As shown in Fig. 7, SNR for PCA+JPEG2000 decreases dramatically when the block size is small (i.e., a larger number of PEs), since the encoder embeds the transform matrix (with an unchanged size) in the bitstream, occupying the overhead bits whose impact is not negligible at low bitrate for small data size. As shown in

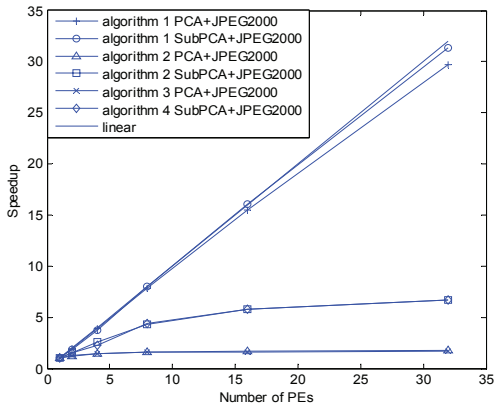

Figure 6. Performance of PCA plus JPEG2000 compression.

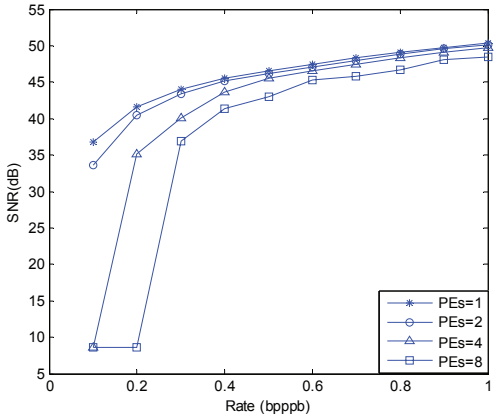

Figure 7. Rate-distortion performance of PCA+JPEG2000 for Algorithm 1.

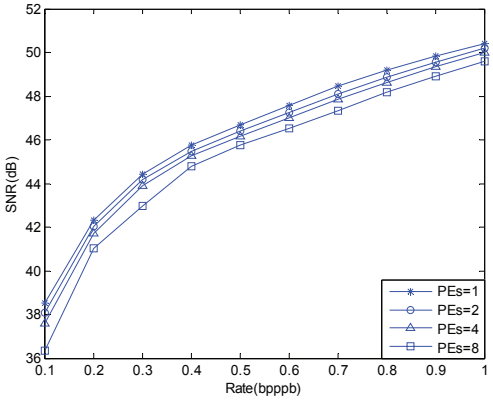

Figure 8. Rate-distortion performance of SubPCA+JPEG2000 for Algorithm 1.

Fig. 8, this may not be a problem for SubPCA since its transform matrix has a much smaller size.

5. CONCLUSIONS

We compared four parallel implementations for hyperspectral image compression using PCA+JPEG2000 and SubPCA+JPEG2000. Since the computation of the PCA is parallelized, the scalability of the implementations is limited by the JPEG2000 encoder. When a parallel encoder is not available, SubPCA parallel implementation achieves greater reduction of the overall compression time and provides a better parallel performance.

Parallel implementations using eigenspace merging have lower speedup performance than that of covariance-matrix merging. But when eigenspace merging is applied to SubPCA+JPEG2000, the speedup performance is excellent with no degradation in compression performance. If the entire compression algorithm is parallelized, the speedup is the best but compression performance may be degraded, particularly for PCA+JPEG2000. Thus, there is a tradeoff between speedup and data-compression performance. In this case, the optimal number of PEs depends on the spatial and spectral sizes of the dataset, an issue which needs further investigation.

6. REFERENCES

[1] A. Plaza, D. Valencia, J. Plaza, and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *J. Parallel Distributed Comp.*, vol. 66, pp. 345-358, 2006.

[2] A. Plaza, D. Valencia, J. Plaza and C.-I Chang, "Parallel Implementation of Endmember Extraction Algorithms from Hyperspectral Data," *IEEE Geo. Rem. Sens. Let.*, vol. 3, pp. 334-338, July 2006.

[3] A. Plaza, "Parallel Techniques for Information Extraction from Hyperspectral Imagery Using Heterogeneous Networks of Workstations," *J. Parallel Distributed Comp.,* vol. 68, pp. 93-111, 2008.

[4] A. Plaza, D. Valencia and J. Plaza, "An Experimental Comparison of Parallel Algorithms for Hyperspectral Analysis Using Homogeneous and Heterogeneous Networks of Workstations," *Parallel Computing*, vol. 34, no. 2, pp. 92-114, 2008.

[5] Q. Du and J. E. Fowler, "Hyperspectral image compression using JPEG2000 and principal components analysis," *IEEE Geo. Rem. Sens. Let.*, vol. 4, pp. 201-205, Apr. 2007.

[6] B. Penna, T. Tillo, E. Magli, and G. Olmo, "Progressive 3-D coding of hyperspectral images based on JPEG 2000," *IEEE Geo. Rem. Sens. Let.*, vol. 3, pp. 125-129, Jan. 2006.

[7] Q. Du and J. E. Fowler, "Low-complexity principal component analysis for hyperspectral image compression," *International Journal of High Performance Computing Applications* (in press).

[8] T. El-ghazawi, S. Kaewpijit, and J. Le Moigne, "Parallel and adaptive reduction of hyperspectral data to intrinsic dimensionality," *Proceedings of the IEEE International Conference on Cluster Computing*, pp. 102 – 109, 2001.

[9] A. Franco, A. Lumini, and D. Maio, "Eigenspace merging for model updating," *Proceedings of International Conference on Pattern Recognition*, 2002.

[10] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, 2nd ed., Addison Wesley, 2003.