



Hyper-spectral Classification and Dimensionality Reduction Algorithms

Wilfredo Lugo – HP, Puerto Rico

Carmen Carvajal –UPRM

Wilson Rivera – UPRM

Gelato Federation Meeting

May 24-26, 2004



Agenda

- ❖ Hyperspectral Imaging (HSI)
- ❖ Dimensionality reduction and classification
- ❖ Performance of HSI algorithms on Itanium
- ❖ Grid level resources
- ❖ Summary

Enhancing Hyperspectral Analysis

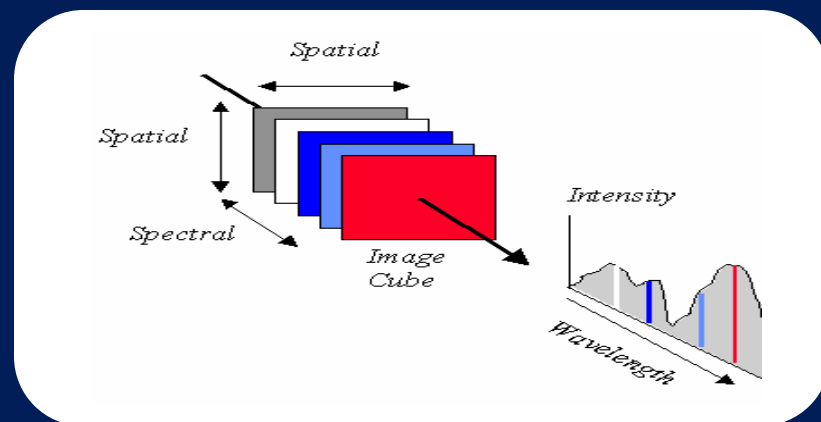
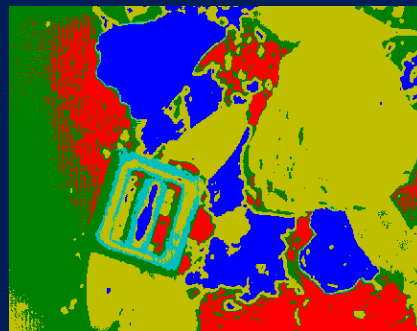
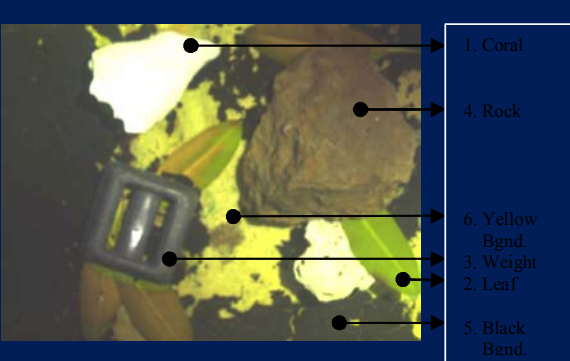
- ❖ HP Itanium 2 Academic Grant Program
 - 8 rx2600 (dual-processor) nodes & 16 port 1Gb switch.
 - \$151k + (\$126K; Matching from HP-PR).
- ❖ Hyper-spectral dimensionality reduction and classification algorithms.
 - **Phase 1**
 - Develop sequential implementations for the classifiers and feature reduction algorithms.
 - Benchmark sequential algorithms.
 - **Phase 2**
 - Port sequential algorithms to a parallel version.
 - Benchmark parallel algorithms.
 - **Phase 3**
 - ❖ Optimize parallel algorithms to Linux/IA64 architecture.
 - **Phase 4**
 - Make the parallel algorithms “GRID aware”.



Hyperspectral Imaging



- ❖ Hyperspectral Imaging (HSI) data contains high spectral resolution and spatial information (a picture of certain scene taken at different wavelengths)
- ❖ Each material has a unique spectrum signature data
- ❖ Tumor detection, land and underwater cover classification, underwater mine detection

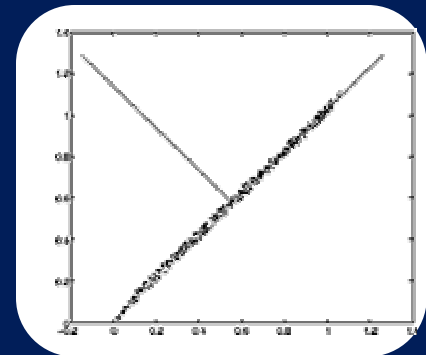


Hyperspectral Imaging

- ❖ New sensor concepts based on imaging spectrometry collect high spectral resolution data over a couple of hundred of wavelengths.
- ❖ High correlation between spectral bands cause data redundancy.
- ❖ Distorted version of the original signature due to scattering by the media or reflections from other objects in the field of view of the sensor.
- ❖ Spectral Data Example :
 - Spectral bands : 400
 - Image Resolution : 600x600
 - Bits/Pixel : 24
 - Data generated : 432 MB on a single image

Dimensionality Reduction: Principal Components

- ❖ Provides a way to compress data without losing much information
- ❖ Produce a smaller set of uncorrelated variables from a correlated data (the new variables are orthogonal to each other)
- ❖ First components has the most significant amount of energy of the data.
- ❖ Steps:
 - Calculate Covariance Matrix
 - Calculate Eigen Values and Vectors
 - Multiply Eigen Vectors and Spectral Image



Dimensionality Reduction : Feedback Iterative Method

❖ Select the subset of bands that best separates the centroids of a given number of classes

❖ Possible number of combinations :

$$\binom{N}{m} = \frac{N!}{(N-m)!m!}$$

❖ Steps

- Perform initial classification (Euclidean or ML)
- For each band combination (Eg 3 2 1)
 - Calculate centroid distances (Eg. 5 1; 5 2; 5 3; 5 4; etc)
 - Calculate centroid average distance for the set
- Selects set with the largest average distance

Classifier : Euclidian Distance

❖ Calculates the distance between each pixel and each class centroid

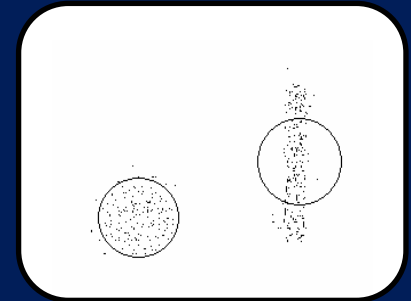
$$g_i(X) = (X - M_i)^T (X - M_i)$$

❖ Assigns each pixel as a member of the class with the closest centroid

❖ Pixel membership changes by iterations

❖ Steps:

- Get C initial means
- For each iteration
 - Calculates pixel distances and assign membership
 - Calculates new means based on the class members
- Returns a classification vector



Classifier : Maximum Likelihood

❖ Calculates the distance between each pixel and each class centroid taking into account the data variance

$$g_i(X) = \frac{1}{2} \ln |\Sigma^{-1}| - \frac{1}{2} (X - M_i)^T \Sigma^{-1} (X - M_i)$$

❖ Assigns each pixel as a member of the class with the closest centroid

❖ Steps:

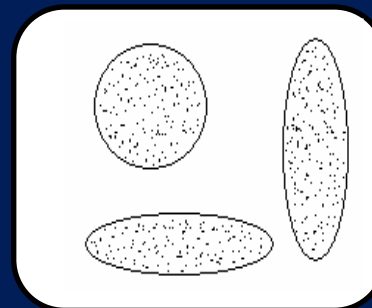
- Get C initial means

- For each iteration

- Calculates pixel distances and assign membership

- Calculates new means based on the class members

- Returns a classification vector



Algorithm Implementations

- Based on BLAS
- IA-32: ATLAS + CLAPACK
 - PIII 1Ghz machine running Red Hat 7.3 with 512 MB of RAM
 - gcc 3.3-3 compiler
- IA-64: HP- MLIB (LAPACK, VECILB, SCILIB)
 - HP rx4640 machine with one IA64 Madison processor 1.5Ghz and 6MB of cache running Red Hat Advanced Server 2.1 with 1GB of RAM
 - Intel 8.0 non commercial compiler

Gathered Benchmarks (Sequential)

Algorithm	Matlab	IA32 Standard Implementation	IA32 (ATLAS + CLAPACK)	IA64 (HP-MLIB)
PCA	5m15s	4m35s	7.09s	5.10s
FIM	3 weeks	3h46m	3h10m	3h32m
ML	38h44m	6h17m	1h4m68s	18m14s
CMeans	15h	21m50s	4m55s	13.91s

Matrix Multiplication Issues

- Common matrix multiplication $O(n^3)$:

```
for (i = 0; i < m; i++)
  for (j = 0; j < p; j++){
    for (k = 0; k < n; k++)
      D[i][j] += A[i][k]*B[k][j];
```

- VERY BAD for any cache processor!!

- Our approach :

Unrolling by 5

```
for (i = 0; i < m; i++)
  for (j = 0; j < p; j++){
    for(k=0;k<n;k+=5){
      t1+=A[i][k ]*B[k ][j];
      t2+=A[i][k+1]*B[k+1][j];
      t3+=A[i][k+2]*B[k+2][j];
      t4+=A[i][k+3]*B[k+3][j];
      t5+=A[i][k+4]*B[k+4][j];
    }
    D[i][j]+=(t1+t2+t3+t4+t5);
```

- Matrices on ML and FIM not satisfies the 5 multiple dimension rule

My Matmat() vs LAPACK sgemm()

- `matmat(float **A, int rows_A, int cols_A (rows of B), float **B, int cols_B, float **C)`
- `sgemm(char *transa, char *transb, int *n, int *m, int *k, float *alpha, double a[], int *lda, double b[], int *ldb, float *beta, double c[], int ldc)`

$$C = \alpha(A) * B + (\beta)C$$

- For a common generic matrix-matrix multiplication
 - `C = zeros(rows_A, cols_B)`
 - `alpha = 1.0`
 - `Beta = 0.0` or `1.0`

float * vs float ** issue

- float ** vs float *
- Common dynamic memory allocation for a NxM array

```
float **A = NULL;
/*Allocates the rows*/
A = malloc(N*sizeof(float *));
/*Allocates the cols*/
for(i = 0; i < N; i++)
    A[i] = malloc(M*sizeof(float));
```
- Problem : MEMORY IS NOT CONTIGOUS!!!

Contiguous Allocation

- Contiguous Allocation of 2D pointers

```
float **A = NULL;
A = malloc(N * sizeof(float ));
A[0] = malloc(N*M*sizeof(float))
for(i = 1; i < N; i++)
    A[i] = A[0] + i*M;
```

- LAPACK call will become:

```
float alpha = 1.0, beta = 1.0;
int rows_a, cols_a (or rows_b) , cols_b, lda, ldb, ldc;
sgemm("TRANS", "TRANS", &rows_a, &cols_a, &cols_b, &A[0][0],
    &lda, &B[0][0], &ldb, &beta, &C[0][0], &ldc);
```

Parallel Benchmark

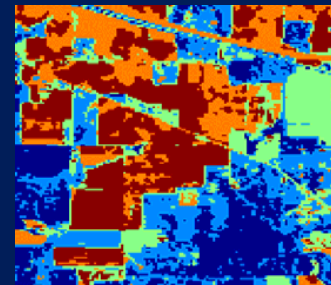
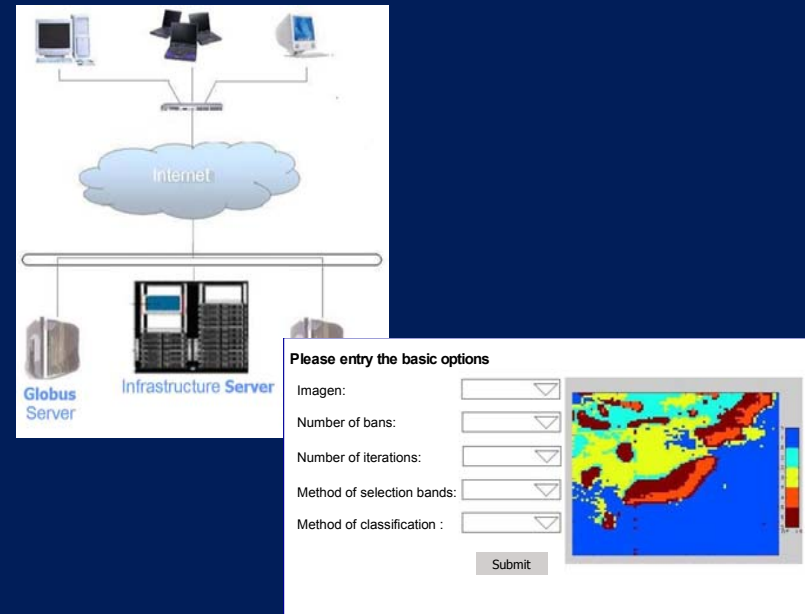
Algorithm	Matlab	IA64 (1.5GHz/3Mb Cache)	16 IA32 nodes
PCA	5h15s	1m9s	13s

Impact on HSI applications:

- Reduced the runtime of a single-body Steepest Descent Fast Multipole Method (SDFMM) application by **74%** on a 32-node cluster
- Reduced the runtime FIM on a **38%** on 16-node Itanium cluster

Grid Resources

- ❖ Continue parallelization/acceleration work in hyperspectral dimensionality reduction and classification algorithm.
 - ❖ Incorporate new resources and scale the Grid-HIS testbed at UPRM.
 - ❖ Design and implement computational methods for ensembles of classifiers reducing dramatically execution time and classification error.
- Development of adaptive middleware
 - **Profile-guided parallelization/optimization**
 - **Multi-language support**
 - **Profile-guided I/O partitioning**
 - **Interaction with distributed image database resources**



Software installed

Machine	Package/Tool	Version	Note
Server/Node	Red Hat Workstation Package	9.0	* Infrastructure Server (server). ** Installing GTP Source Installation package (node).
Server/Node	Java SDK	1.4.1	The underlying code of GT3 is written in Java.
Server	CAMgr	N/A	Managed a simple certificates authority base on openssl.
Node	Apache Ant	1.5.3	Used to execute an Ant-based build script that automates the build process.
Node	Junit	3.8.1	Java-based testing framework that facilitates regression tests.
Node	PostgreSQL	7.3.3	Open source relational database management system.
Server/Node	Globus Toolkit	3.0	

Summary

- Sequential Implementations
- Sequential Port to IA64
 - Load proper libraries and compilers
- Sequential optimization on IA64
- Parallel implementations
 - Optimization on IA-64 cluster
- Grid level resources