Hyperspectral Unmixing on GPUs and Multi-Core Processors: A Comparison

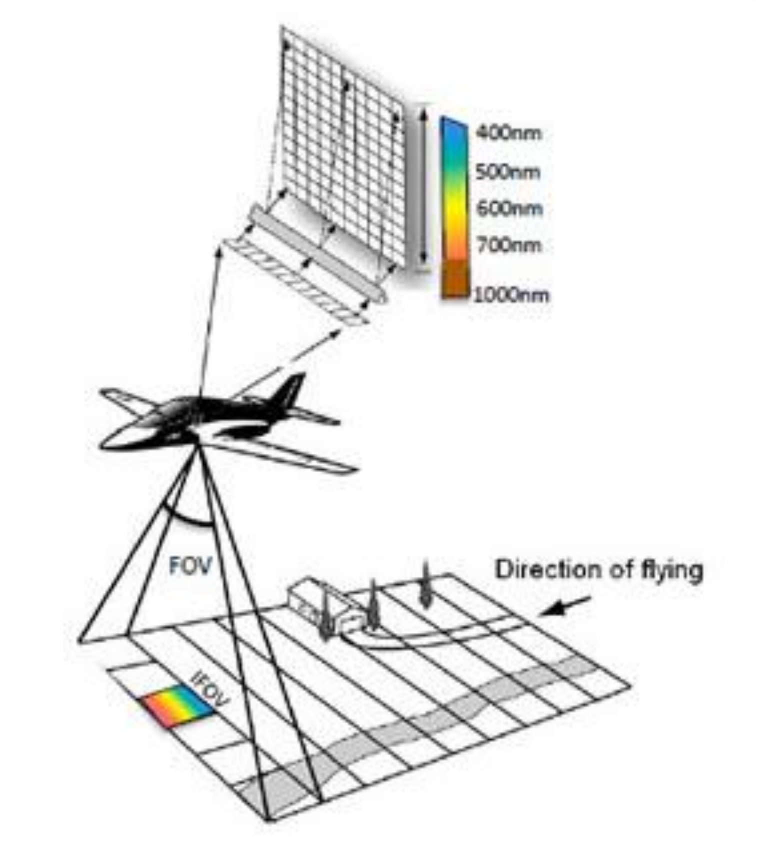
Dept. of Mechanical and Environmental Informatics
Kimura-Nakao lab.
Uehara Daiki

Today’s outline

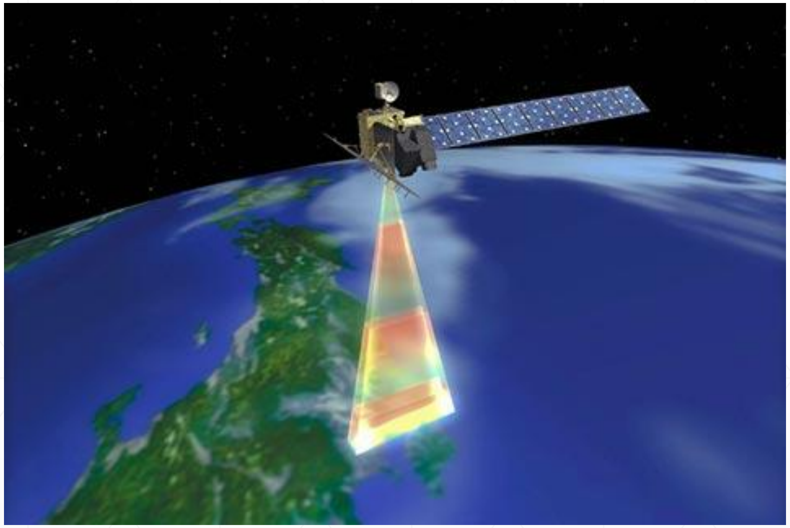
1. Self-introduction
2. Basics of hyperspectral imaging (Related p1386-1388)
3. Unmixing chain algorithm (Related p1388-1389)
4. GPU and multicore implementation (Related p1389-1391)
5. Experimental result (Related p1391-1396)
6. Conclusion

Today's outline

How to take a hyperspectral image



Reference 1



Reference 2

AVIRIS

AVIRIS

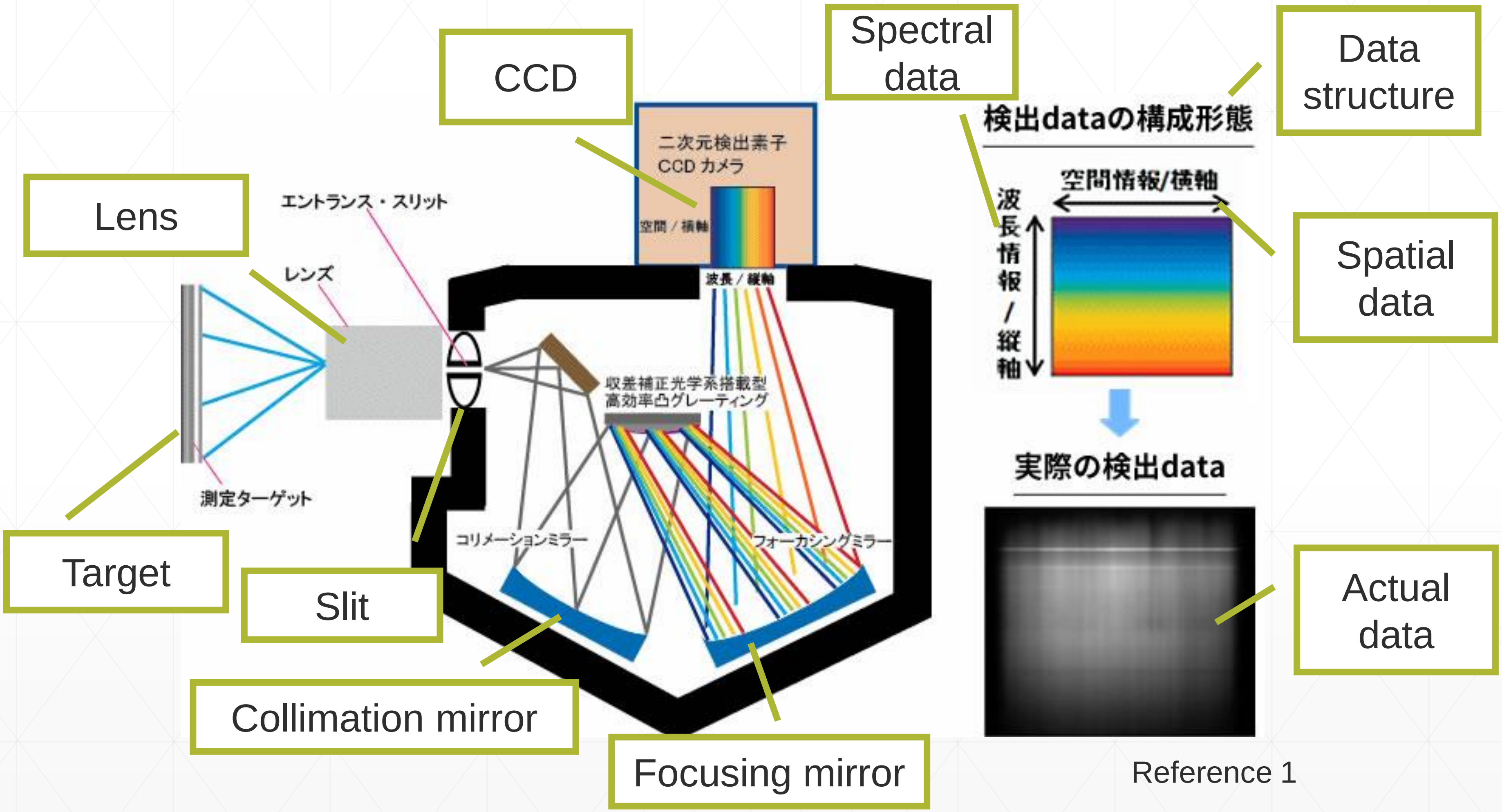
Airborne Visible / Infrared Imaging Spectrometer

NASA

Jet Propulsion Laboratory
California Institute of Technology

Reference 3

A kind of hyperspectral image sensor

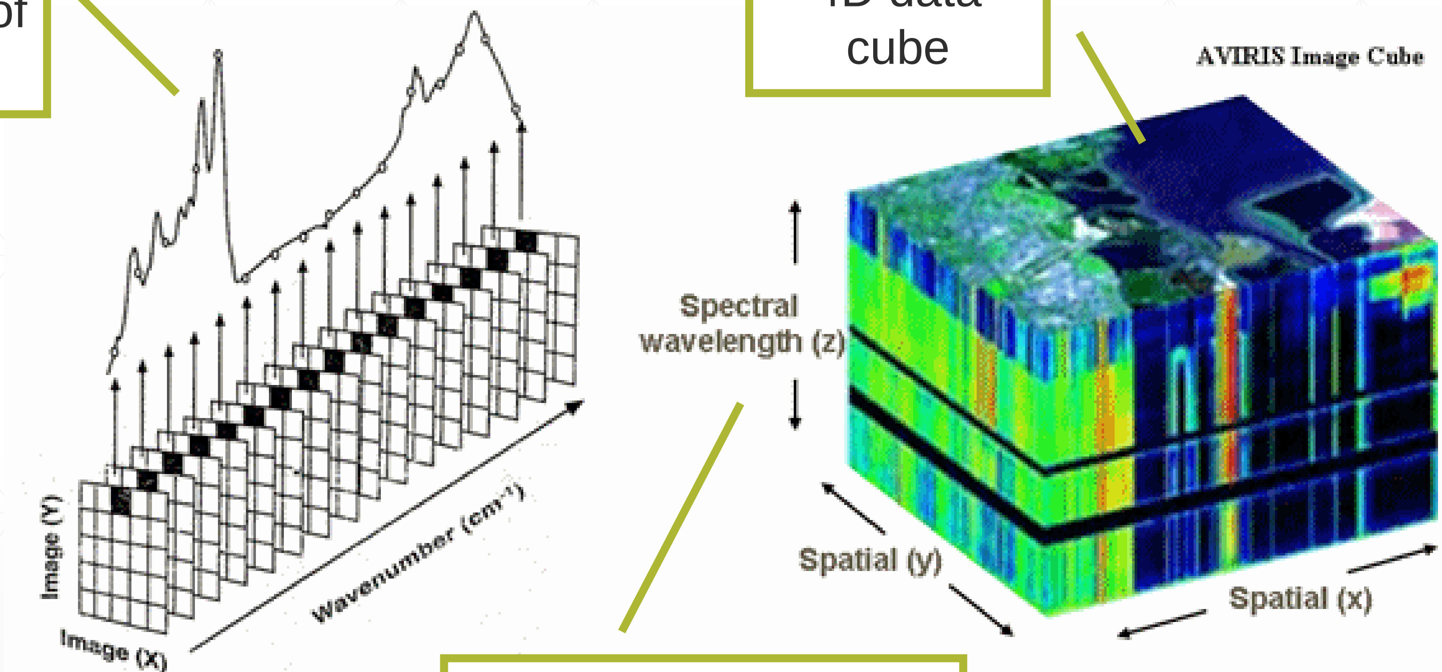


Reference 1

Hyperspectral image

Spectrum of a pixel

4D data cube

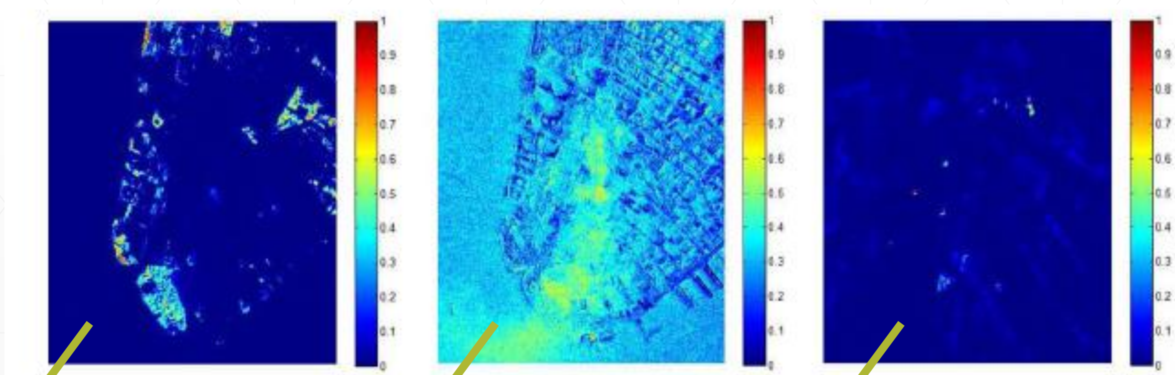


224 bands
0.4 to 2.5 μm
1band 9.375 nm width

Reference 1

Hyperspectral image

False color composition of an AVIRIS WTC scene



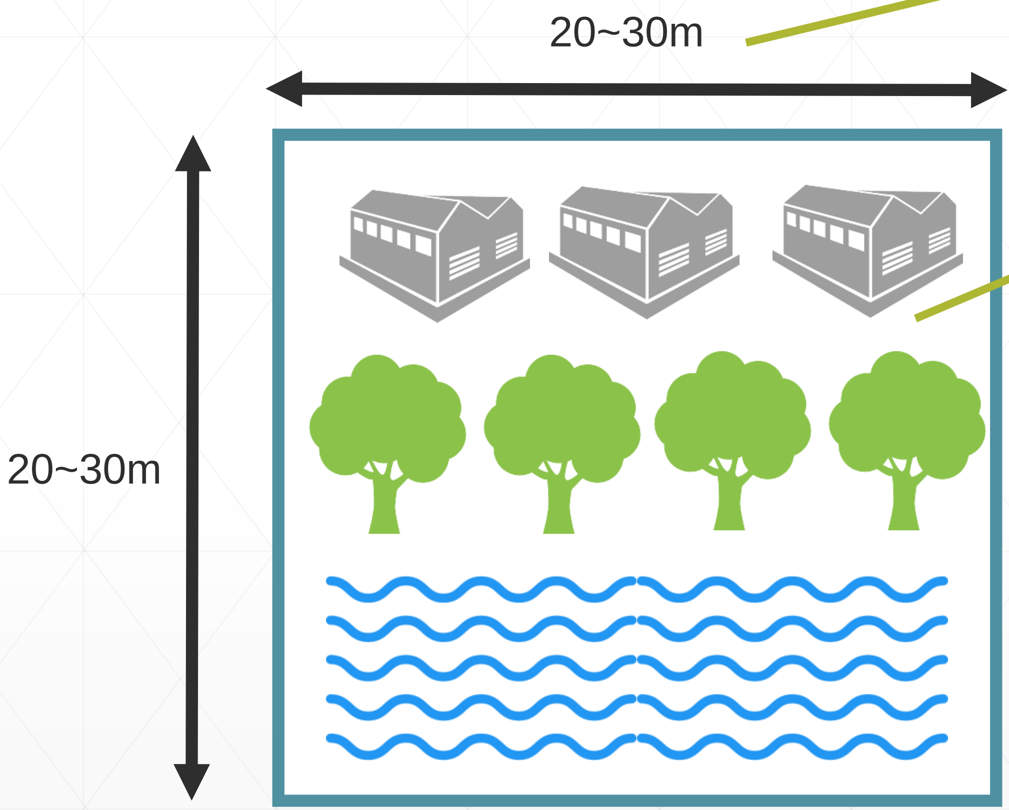
Vegetation

Smoke

Fire

Reference 4

Mixture problem



A pixel

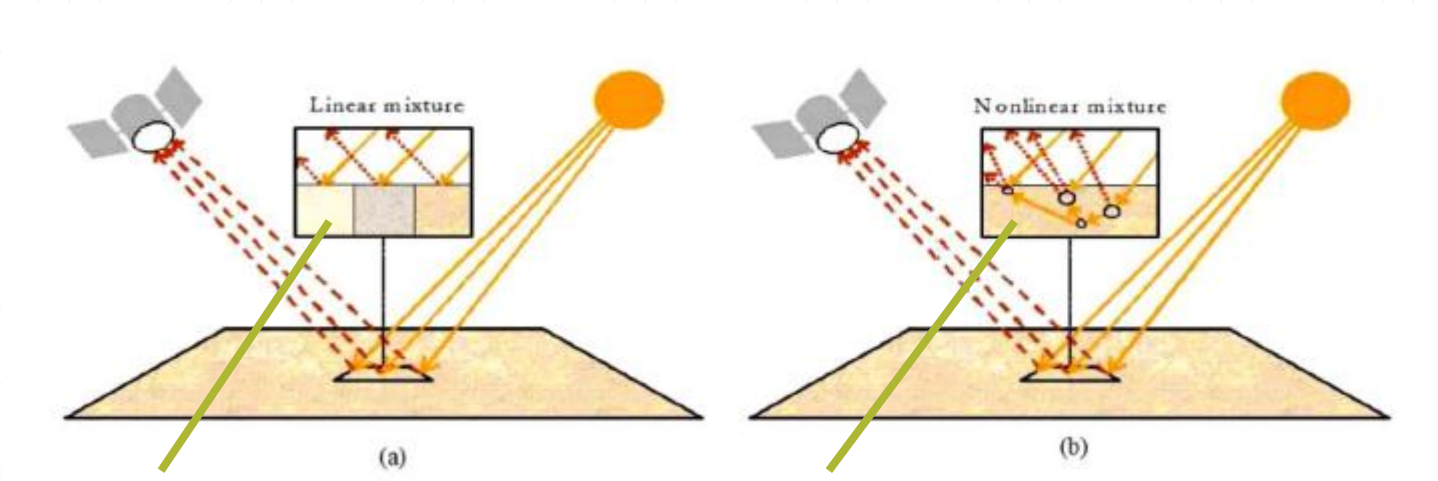
Low spatial resolution

Several kind of endmembers in a pixel

[Endmember]
Buildings
Vegetation
Water
Fire
etc.

Hyperspectral unmixing is needed for accurate analysis

Mixture model

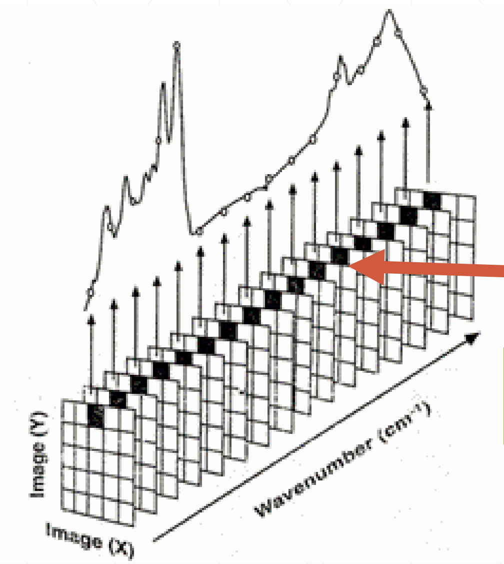


Linear mixture model

Non-linear mixture model

Reference 4

Linear mixture model



Number of endmembers

Abundance of i-th endmember

Noise vector
L dimensions

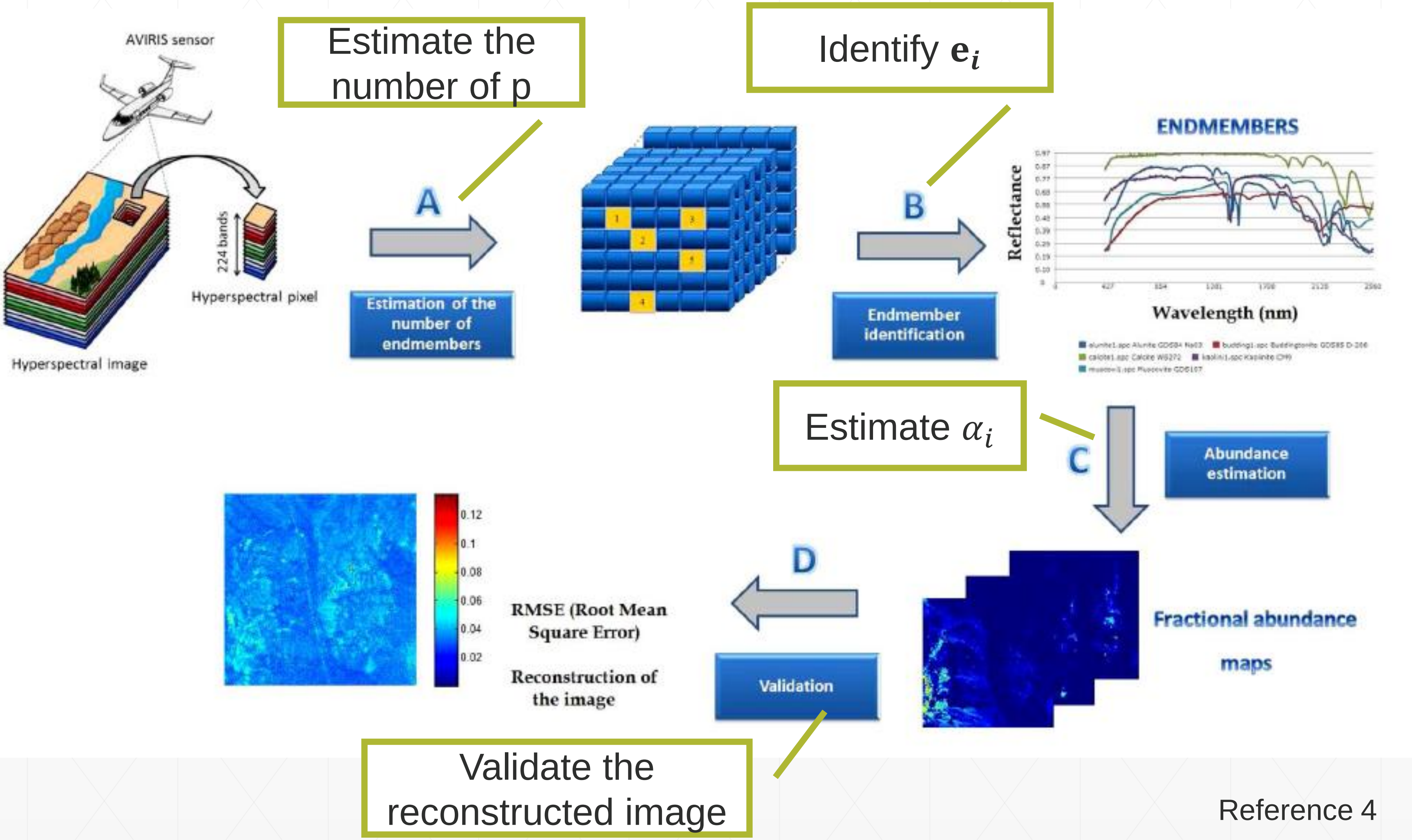
$$\boldsymbol{y} \approx \mathbf{M}\alpha + \mathbf{n} = \sum_{i=1}^{p} \mathbf{e}_i \alpha_i + \mathbf{n}$$

Pixel vector
L dimensions

Endmember signature vector
L dimensions

Note: L is the number of spectral bands.

Full hyperspectral unmixing chain



Reference 4

Today’s outline

1. Self-introduction
2. Basics of hyperspectral imaging (Related p1386-1388)
3. **Unmixing chain algorithm (Related p1388-1389)**
4. GPU and multicore implementation (Related p1389-1391)
5. Experimental result (Related p1391-1396)
6. Conclusion

Estimation of the number of endmembers

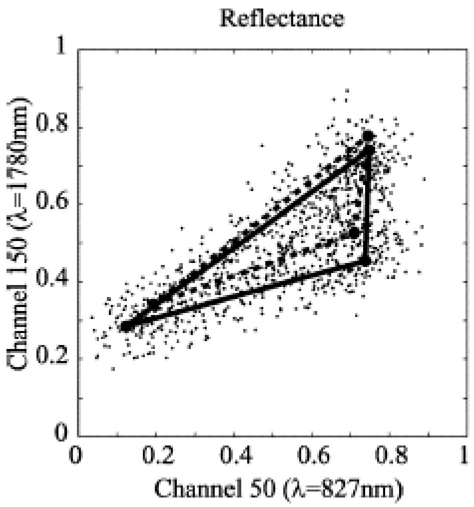
- Virtual dimensionality algorithm

1. Calculate the covariance matrix.
   $$\mathbf{K}_{L\times L} = \frac{1}{N}(\mathbf{Y}-\overline{\mathbf{Y}})^T(\mathbf{Y}-\overline{\mathbf{Y}})$$
2. Calculate the correlation matrix.
   $$\mathbf{R}_{L\times L} = \mathbf{K}_{L\times L} + \overline{\mathbf{Y}\mathbf{Y}}^T$$
3. Calculate covariance eigenvalues $\{\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_L\}$ and correlation eigenvalues $\{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L\}$.
4. If $\hat{\lambda}_l - \lambda_l > 0$ for $l = 1,2,\ldots,L$ then $p \mathrel{+}= 1$
   - p is the number of endmembers.
   - Neyman-Pearson test is used for estimation of the number of endmembers.

Basic idea of Virtual Dimensionality

- Assuming the hyperspectral signatures are unknown nonrandom and deterministic signal sources.
- Assuming noise is white with zero mean.
- Auto-covariance
  - $\mathrm{K_{XX}}(\tau) = \mathrm{E}[(X(t) - \mu)(X(t+\tau) - \mu)]$
    $= \mathrm{E}[X(t) \cdot X(t+\tau)] - \mu^2 = \mathrm{R_{XX}}(\tau) - \mu^2$
  - If $\mathrm{K_{XX}}(\tau) = \mathrm{R_{XX}}(\tau)$, $\mu^2 = 0$. This means that there is only noise.

Convex cones of hyperspectral image

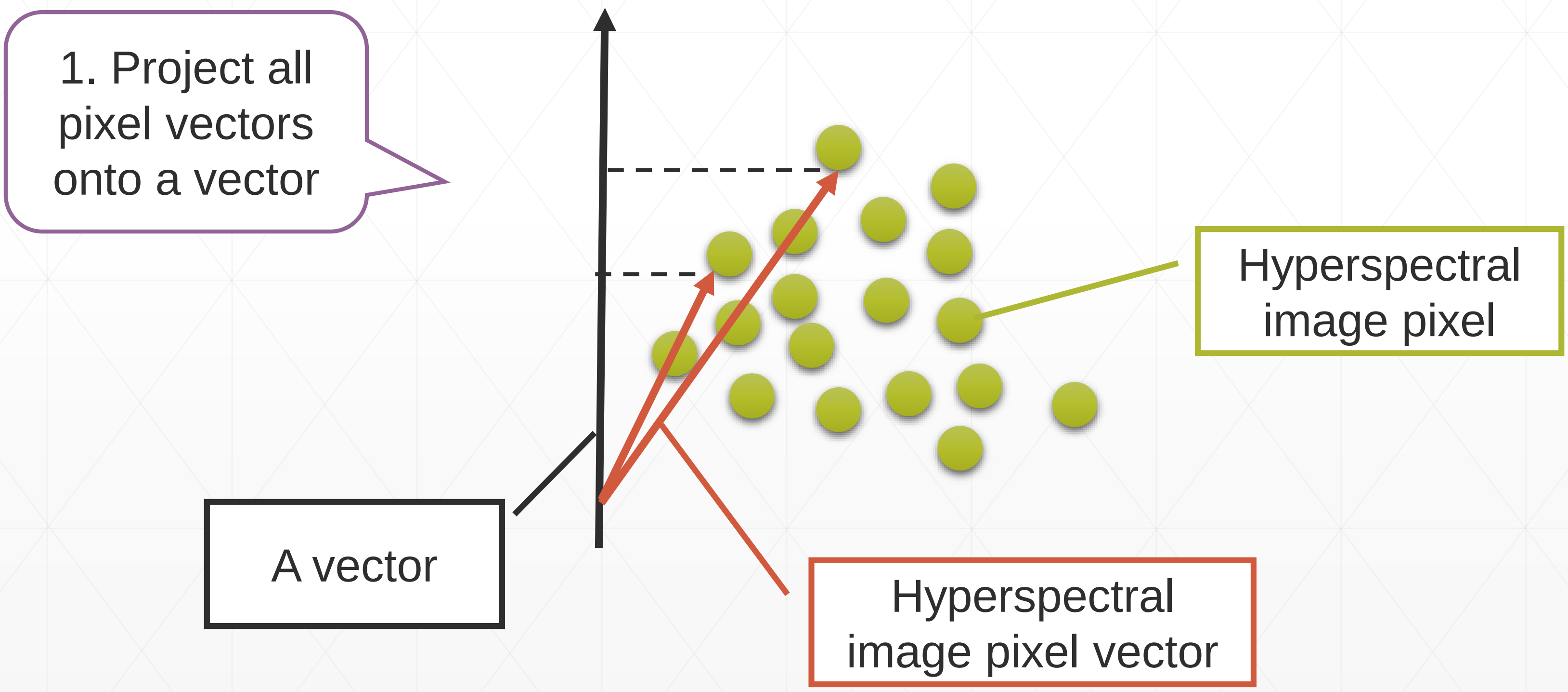


Reflectance convex cone
Reference 5

- Reflectance (and radiance) is strictly non-negative.
- Reflectance (or radiance) spectrum vectors lie inside a convex region [Ref 6].
- Vertices of the convex can be used as endmember spectra [ref 6].
  - If the mixture model is the linear mixture model.

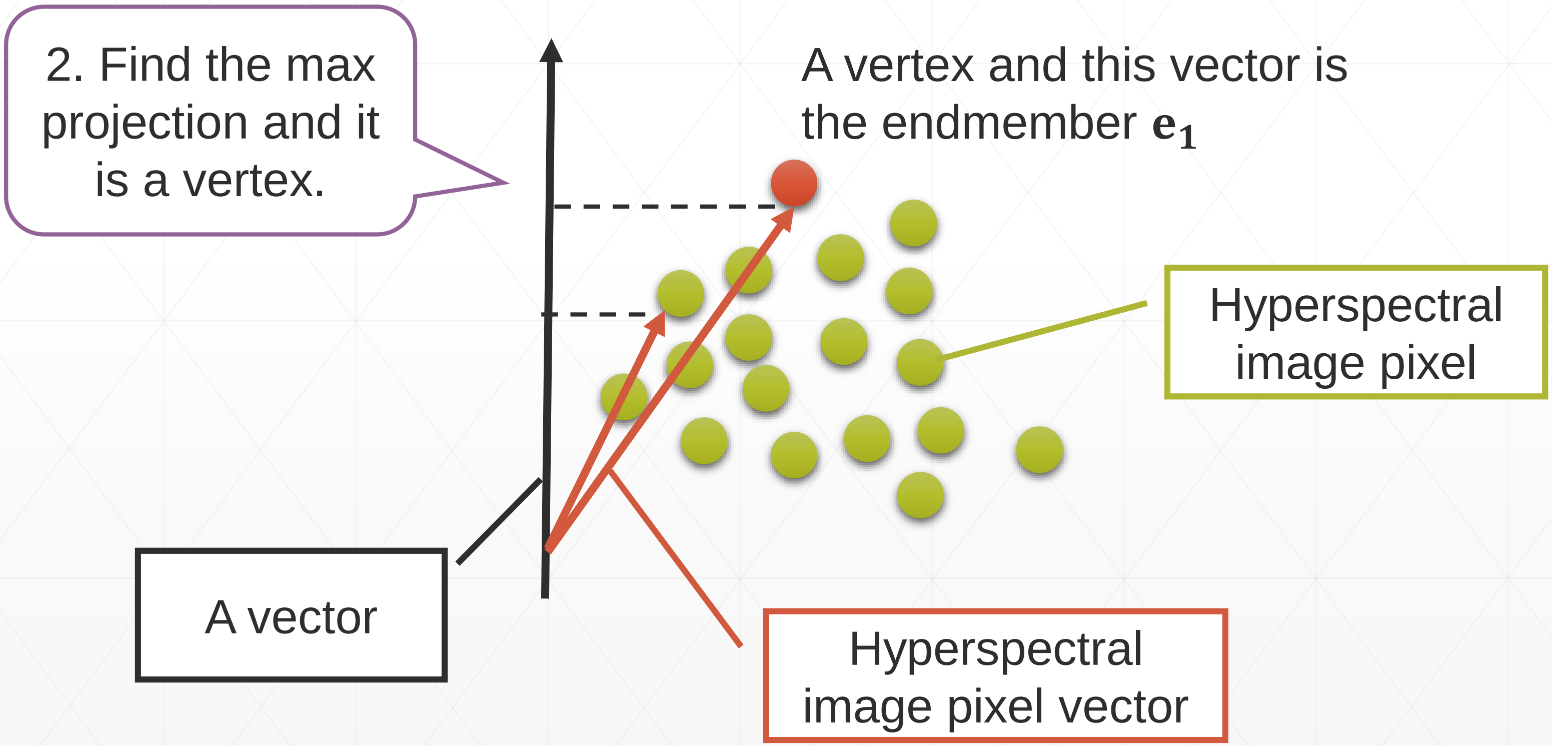
Endmember extraction

- Orthogonal Subspace Projection
  - The objective of this algorithm is to find vertex of the convex cone. (Ref 7)



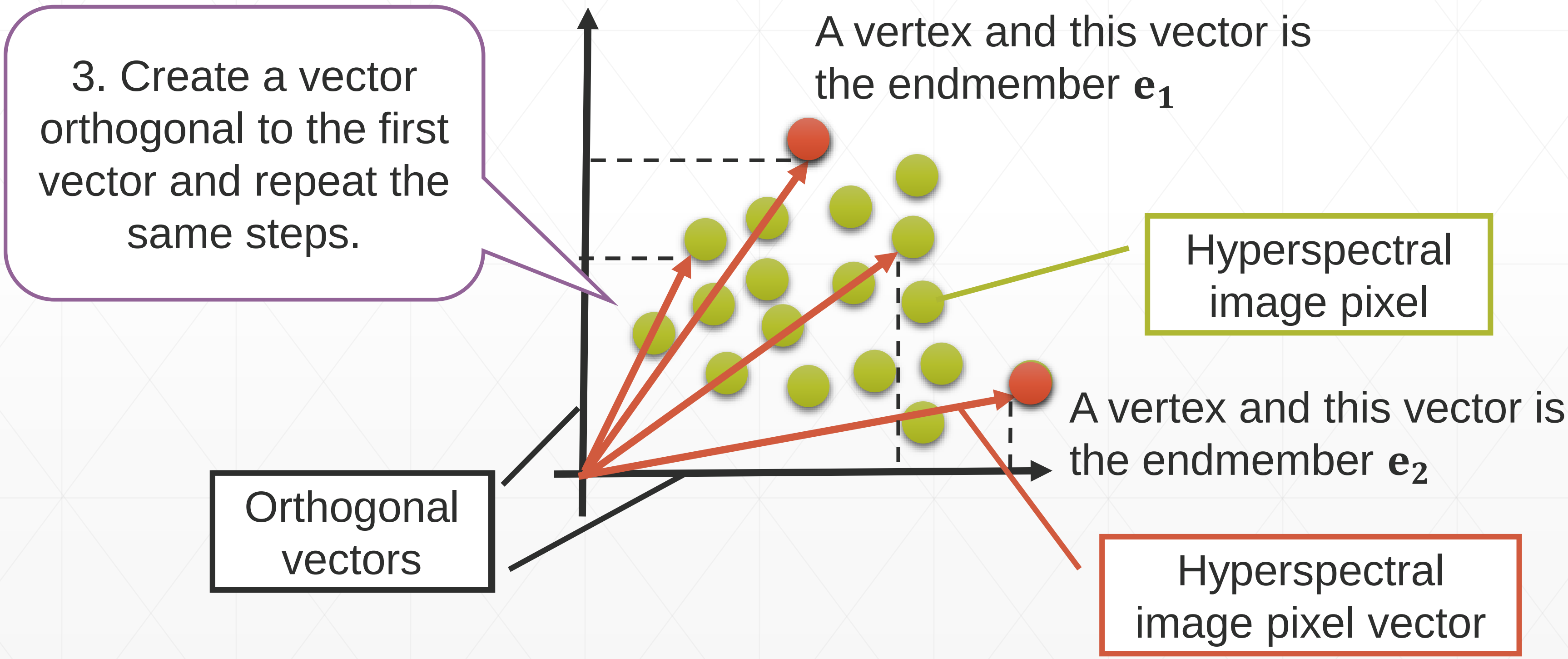
Endmember extraction

- Orthogonal Subspace Projection
  - The objective of this algorithm is to find vertex of the convex cone. (Ref 7)



Endmember extraction

- Orthogonal Subspace Projection
  - The objective of this algorithm is to find vertex of the convex cone. (Ref 7)



Unconstrained least squares algorithm for abundance estimation

- Endmembers $\mathbf{M} = \{\mathbf{e}_i\}_{i=1}^{p}$.
- Abundance fractions $\alpha = [\alpha_1, \alpha_2, \dots \alpha_p]$.
- α can be estimated by the following expression in least squares sense.

$$\alpha = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y}$$

Today’s outline

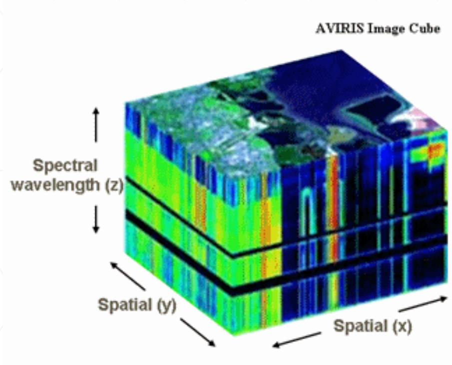
1. Self-introduction
2. Basics of hyperspectral imaging (Related p1386-1388)
3. Unmixing chain algorithm (Related p1388-1389)
4. **GPU and multicore implementation (Related p1389-1391)**
5. Experimental result (Related p1391-1396)
6. Conclusion

Review the VD algorithm

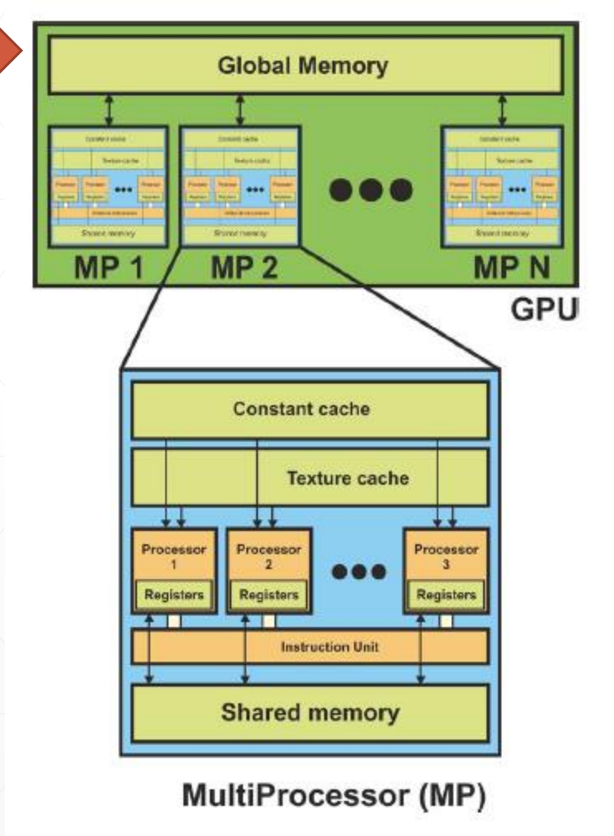
- Virtual dimensionality algorithm
  1. Calculate the covariance matrix.
     $\mathbf{K}_{L\times L} = \frac{1}{N}(\mathbf{Y}-\overline{\mathbf{Y}})^T(\mathbf{Y}-\overline{\mathbf{Y}})$
  2. Calculate the correlation matrix.
     $\mathbf{R}_{L\times L} = \mathbf{K}_{L\times L} + \overline{\mathbf{Y}\mathbf{Y}^T}$
  3. Calculate covariance eigenvalues $\{\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_L\}$ and correlation eigenvalues $\{\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L\}$.
  4. If $\hat{\lambda}_l - \lambda_l > 0$ for $l = 1,2,\ldots,L$ then $p += 1$
     - p is the number of endmembers.
     - Neyman-Pearson test is used for estimation of the number of endmembers.

GPU implementation of VD

1. Load the full hyperspectral image **Y** to the main memory of the GPU.



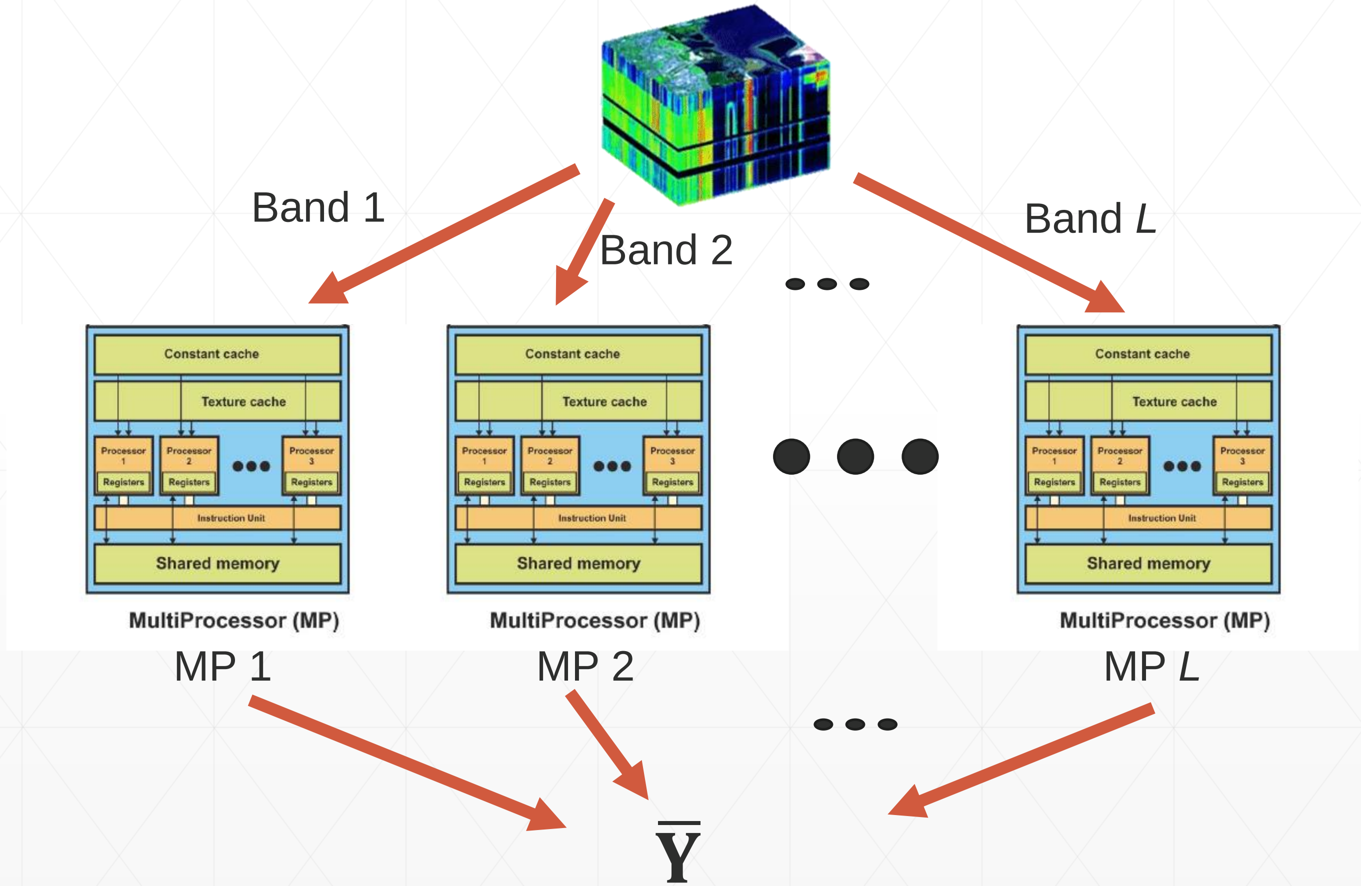
Reference 1



Reference 4

GPU implementation of VD

1. Calculate $\bar{\mathbf{Y}}$ using L blocks.
   - L is the number of bands.



GPU implementation of VD

2. Calculate the covariance matrix.

$$\mathbf{K}_{L\times L} = \frac{1}{N}(\mathbf{Y} - \bar{\mathbf{Y}})^T(\mathbf{Y} - \bar{\mathbf{Y}})$$

- *cublassSgemm* function (a cuBLAS library function) is used for the parallel matrix multiplication above.

3. Calculate the autocorrelation matrix.

$$\mathbf{R}_{L\times L} = \mathbf{K}_{L\times L} + \overline{\mathbf{Y}\mathbf{Y}}^T$$

- Calculate $L \times L$ components using $L \times L$ threads as follows:

$$\mathbf{R}_{ij} = \mathbf{K}_{ij} + \bar{\mathbf{Y}}_i\bar{\mathbf{Y}}_j$$

4. Calculate autocorrelation and covariance eigenvalues using host CPU.

GPU implementation of VD

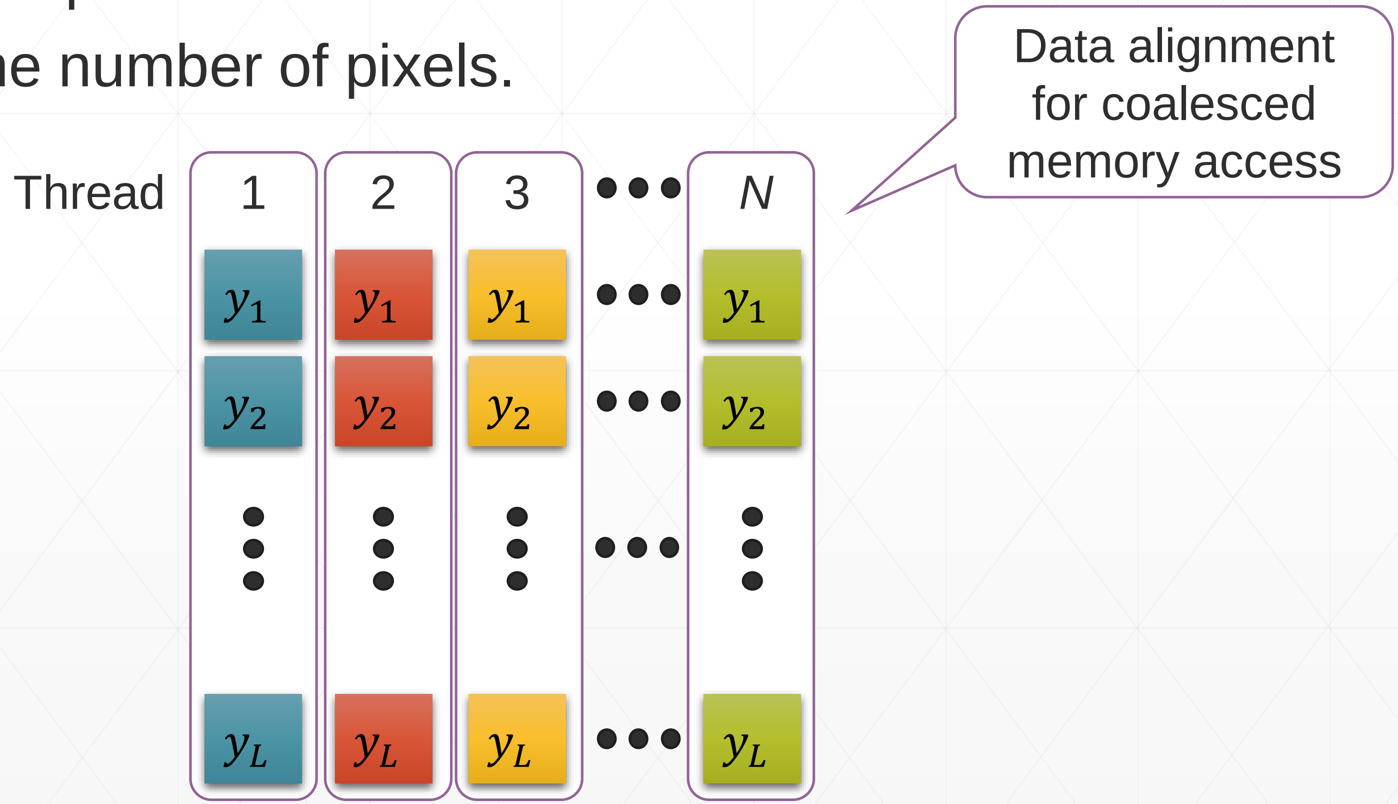
5. If $\hat{\lambda}_l - \lambda_l > 0$ for $l = 1,2, \ldots, L$ then $p \mathrel{+}= 1$
   - p is the number of endmembers.
   - Neyman-Pearson test is used for estimation of the number of endmembers.
   - This step processed in host CPU.

GPU implementation of OSP

- Orthogonal Subspace Projection with Gram-Schmidt orthogonalization (OSP-GS)
  - By using Gram-Schmidt orthogonalization, OSP can be parallelized.

GPU implementation of OSP

1. Store the pixel vector by columns in GPU global memory.
   - y_i: i is the band number.
   - Color: a pixel vector.
   - N is the number of pixels.

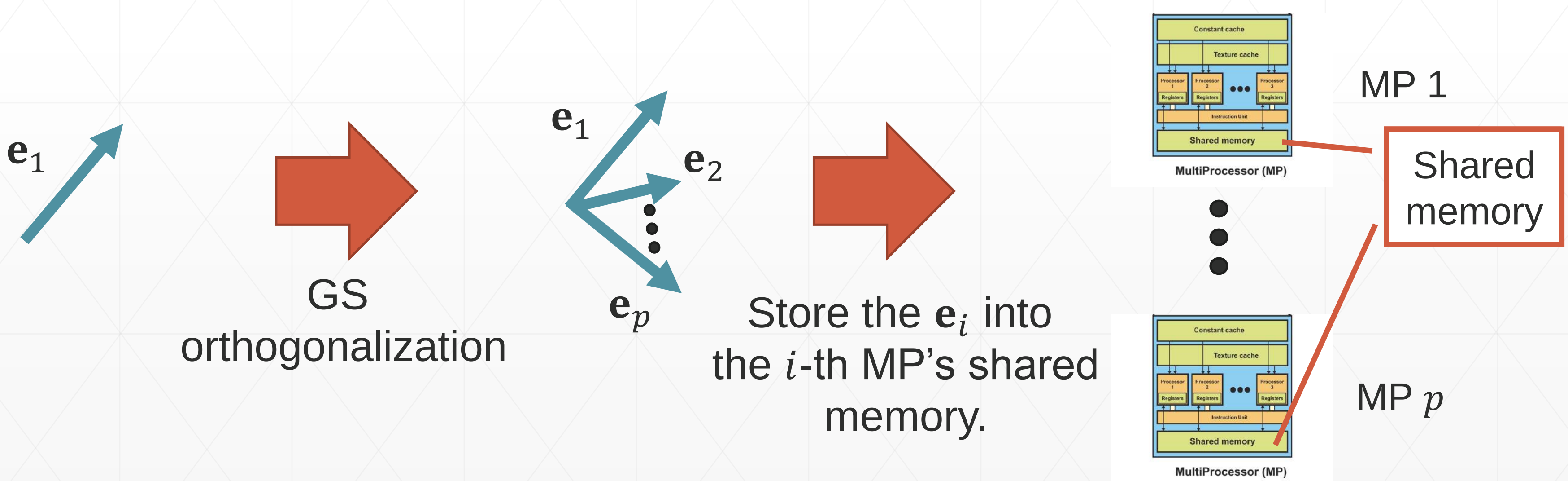


GPU implementation of OSP

2. Calculate the brightest pixel $\mathbf{e}_1$ in $\mathbf{Y}$.
   - Calculate the dot product between each pixel and its transposed version in parallel.
   - A pixel vector which has the maximum projection value will be the $\mathbf{e}_1$.

GPU implementation of OSP

3. Calculate the vectors orthogonal to the $\mathbf{e}_1$ by using Gram-Schmidt orthogonalization.
   - **Note:** $\mathbf{e}_i$ vectors in the eq.(2) of today's paper is ***not*** an endmember.
   - The number of orthogonal vectors is p.
     - p is the number of endmembers calculated in VD steps.
   - This step is processed in the host CPU.



GPU implementation of OSP

Those steps below are processed using p blocks.

4. Project all pixel vectors onto the orthogonal vectors.
   - The orthogonal vector is stored in the shared memory for fast access.
5. Find the vector which has the maximum projection value.
6. Store the vector found in the previous step in the shared memory. This vector is an endmember vector $\mathbf{e}_i$.
7. Pass the endmember vector to the endmember matrix $\mathbf{M}$.

GPU implementation of UCLS

1. Calculate $\mathbf{M}^T\mathbf{M}$ in the GPU.
2. Calculate $(\mathbf{M}^T\mathbf{M})^{-\mathbf{1}}$ in the host CPU.
3. Multiply the inverse by $\mathbf{M}$ in the GPU.
4. Multiply the result by each pixel $\mathbf{y}$.

Multicore implementation of the unmixing chain

1. Matrix multiplication
   - OpenMP+BLAS
   - Divide the matrix into some OpenMP threads.
   - In each thread, invoke *dgemm* function.
   - *dgemm* is a BLAS function. This function is optimized for matrix multiplication.
2. Use parallel for directive.

Today's outline

1. Self-introduction
2. Basics of hyperspectral imaging (Related p1386-1388)
3. Unmixing chain algorithm (Related p1388-1389)
4. GPU and multicore implementation (Related p1389-1391)
5. **Experimental result (Related p1391-1396)**
6. Conclusion

Hyperspectral image data

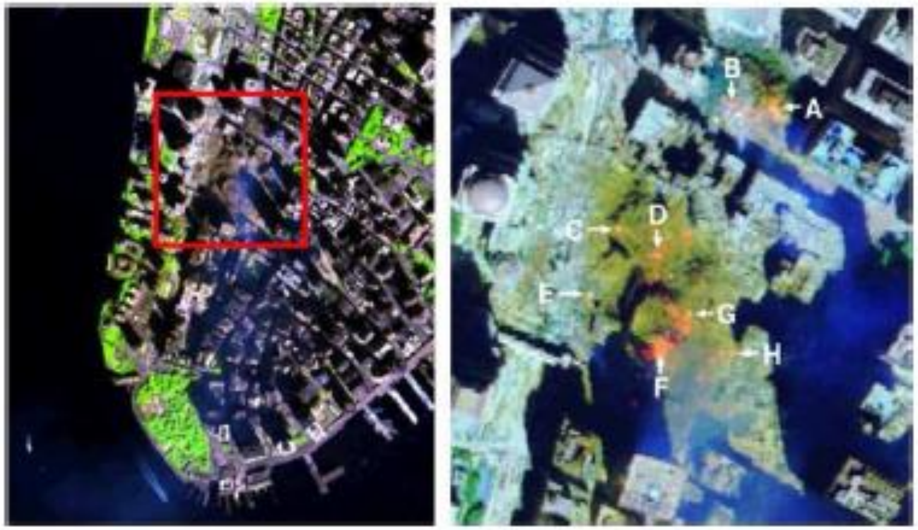


Fig. 6. False color composition of an AVIRIS hyperspectral image collected by NASAs Jet Propulsion Laboratory over lower Manhattan on Sept. 16, 2001 (left). Location of thermal hot spots in the fires observed in World Trade Center area, available online: http://pubs.usgs.gov/of/2001/ofr-01-0429/hotspot.key.tgif.gif (right).

TABLE II
SPECTRAL ANGLE VALUES (IN DEGREES) BETWEEN THE TARGET PIXELS EXTRACTED BY OSP-GS ALGORITHM AND THE KNOWN GROUND TARGETS IN THE AVIRIS WORLD TRADE CENTER SCENE

A	B	C	D	E	F	G	H
0.00°	27.16°	0.00°	15.62°	27.81°	3.98°	2.72°	24.26°

614x512 pixels and 224 bands
Size: 140MB

Reference 4

Hyperspectral image data

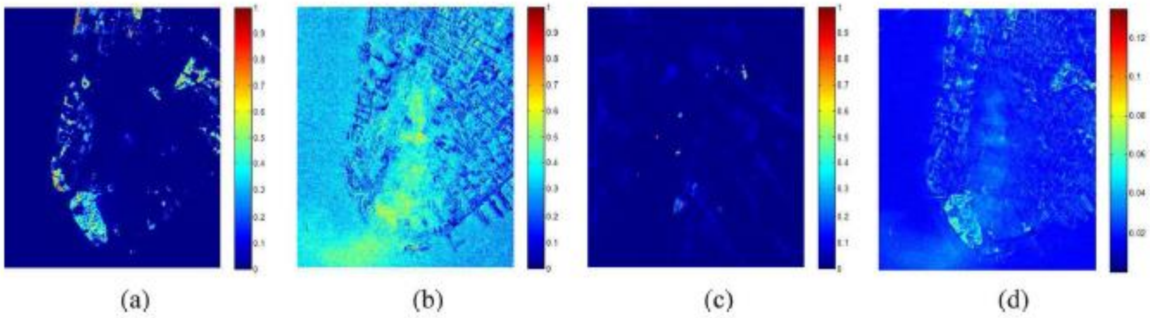


Fig. 9. Abundance maps extracted from the WTC scene for different targets: (a) Vegetation. (b) Smoke. (c) Fire. (d) Per-pixel RMSE obtained in the reconstruction process of the AVIRIS WTC scene using $p = 31$ endmembers (the overall RMSE in this case was 0.0216).

Reference 4

Hyperspectral image data

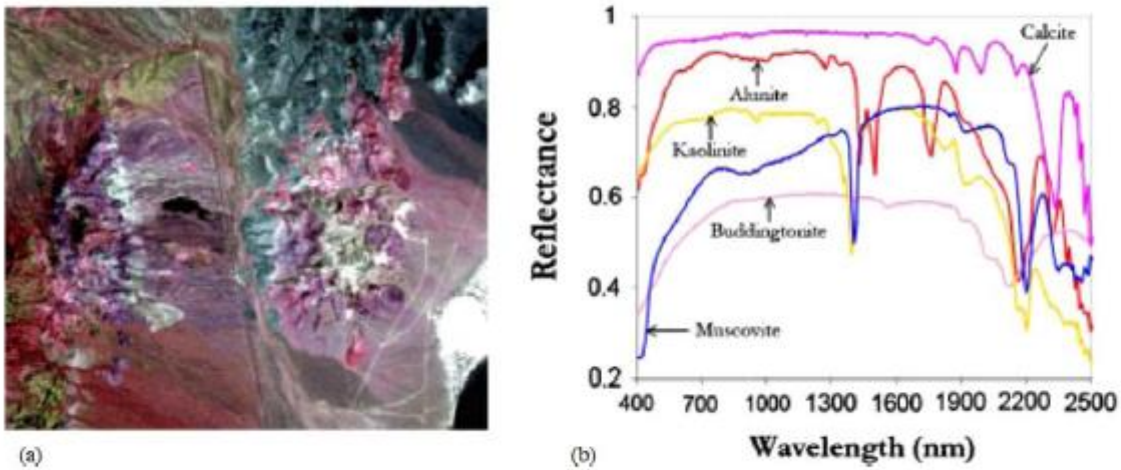


Fig. 7. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada and (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

TABLE I
SPECTRAL ANGLE VALUES (IN DEGREES) BETWEEN THE TARGET PIXELS EXTRACTED BY THE OSP-GS ALGORITHM AND THE REFERENCE USGS MINERAL SIGNATURES FOR THE AVIRIS CUPRITE SCENE

Alunite	Buddingtonite	Calcite	Kaolinite	Muscovite	Average
5.48°	4.08°	5.87°	11.14°	5.68°	6.45°

Reference 4

Hyperspectral image data

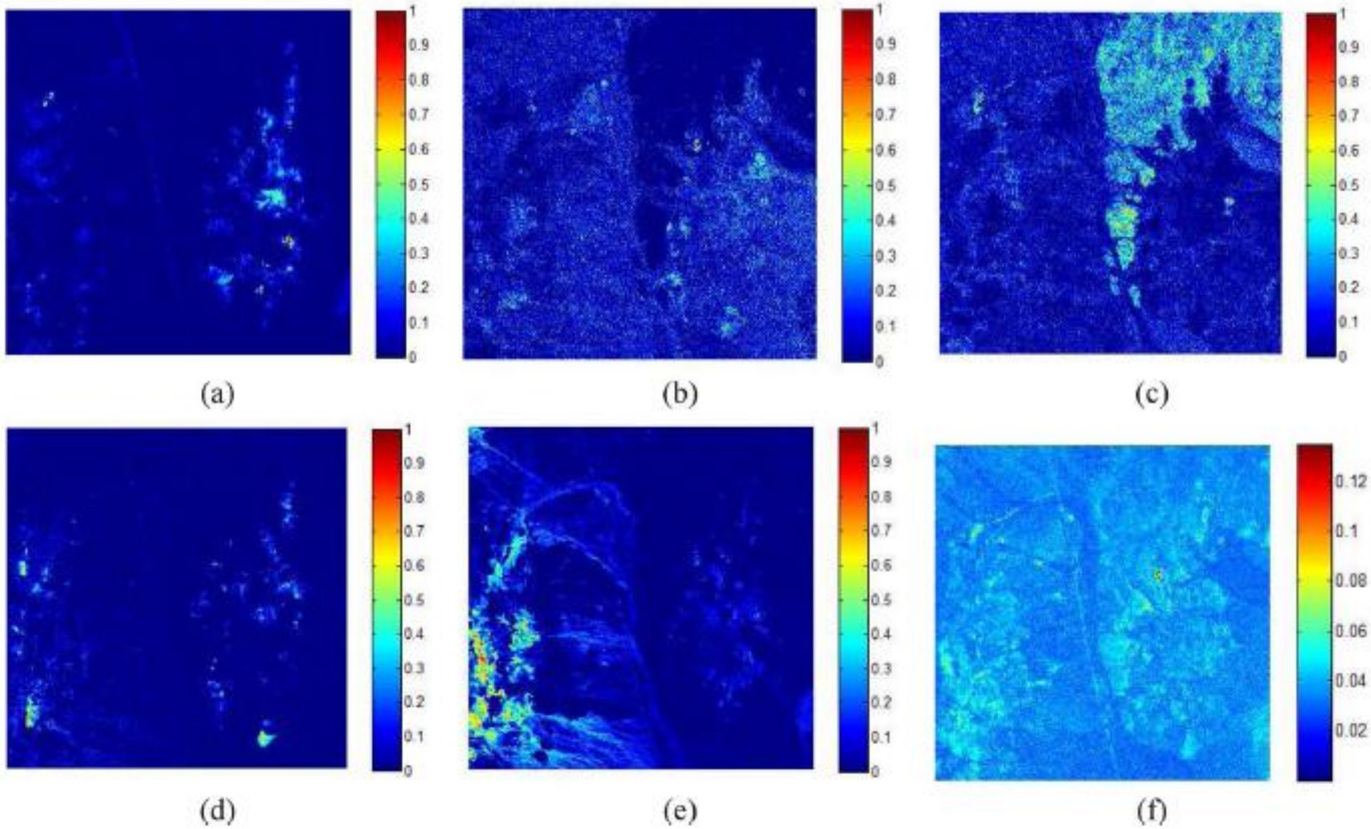


Fig. 8. Abundance maps extracted from the Cuprite scene for different minerals: (a) Alunite. (b) Budinggtonite. (c) Calcite. (d) Kaolinite. (e) Muscovite. (f) Per-pixel RMSE obtained in the reconstruction process of the AVIRIS Cuprite scene using $p = 19$ endmembers (the overall RMSE in this case was 0.0361).

GPU and multicore CPU specs

1. GPU1
   - NVidia Tesla C1060, 240 cores, 1.296GHz, 4GB total dedicated memory, 800MHz memory, 102GB/s
2. GPU2
   - NVidia GeForce GTX 580, 512 cores, 1.544GHz, 1,536MB total dedicated memory, 2,004MHz memory, 192.4GB/s
3. MC1
   - Intel i7 920, 2.67GHz, 4 cores, 6GB DDR3 RAM, host of GPU1 and GPU2
4. MC2
   - Intel Xeon, 2.53GHz, 12 cores, 24GB DDR3 RAM

Processing times and speedups

TABLE IV
PROCESSING TIMES (IN SECONDS) AND SPEEDUPS ACHIEVED FOR THE PARALLEL UNMIXING CHAIN IN TWO DIFFERENT PLATFORMS: MULTI-CORE AND GPU, TESTED WITH THE AVIRIS CUPRITE SCENE

	Initialization	VD	OSP-GS	UCLS	Writing of final results	Total
Serial time	0.121	5.541	1.331	1.051	0.009	8.053
Parallel time GPU1	0.269	0.246	0.049	0.067	0.009	0.640
Parallel time GPU2	0.281	0.241	0.024	0.034	0.011	0.590
Parallel time MC1	0.126	0.924	0.516	0.277	0.010	1.853
Parallel time MC2	0.098	1.066	1.055	0.197	0.053	2.468
Speedup (GPU1)	–	22.48	27.26	15.74	–	12.58
Speedup (GPU2)	–	23.00	55.28	30.62	–	13.64
Speedup (MC1)	–	6.00	2.58	3.79	–	4.35
Speedup (MC2)	–	5.20	1.26	5.35	–	3.26

Not real time

This cuprite scene was took in 1.985 sec.
Processing time must be less than 1.985 sec for real time processing.

Processing times and speedups

TABLE V
PROCESSING TIMES (IN SECONDS) AND SPEEDUPS ACHIEVED FOR THE PARALLEL UNMIXING CHAIN IN TWO DIFFERENT PLATFORMS: MULTI-CORE AND GPU, TESTED WITH THE AVIRIS WTC SCENE

	Initialization	VD	OSP-GS	UCLS	Writing of final results	Total
Serial time	0.364	20.149	9.979	10.314	0.036	$\boxed{40.842}$
Parallel time GPU1	0.522	0.711	0.202	0.280	0.039	1.755
Parallel time GPU2	0.535	0.499	0.109	0.133	0.037	1.313
Parallel time MC1	0.370	3.258	3.549	2.759	0.037	$\boxed{9.973}$
Parallel time MC2	0.281	3.782	4.612	1.620	0.206	$\boxed{10.501}$
Speedup (GPU1)	–	28.34	49.28	36.79	–	23.28
Speedup (GPU2)	–	40.37	91.49	77.66	–	31.12
Speedup (MC1)	–	6.18	2.81	3.74	–	4.10
Speedup (MC2)	–	5.33	2.16	6.37	–	3.89

Not real time

This WTC scene was took in 5.096 sec.
Processing time must be less than 5.096 sec for real time processing.

Today's outline

1. Self-introduction
2. Basics of hyperspectral imaging (Related p1386-1388)
3. Unmixing chain algorithm (Related p1388-1389)
4. GPU and multicore implementation (Related p1389-1391)
5. Experimental result (Related p1391-1396)
6. **Conclusion**

Conclusion

1. Hyperspectral imaging can benefit from GPU and multicore processors.
2. GPUs and multicore processors are still rarely exploited in real missions due to power consumption and radiation tolerance issue.

References

1. HyperSpec ハイパースペクトルセンサー・イメージ分光カメラ原理/ハイパースペクトルイメージングとは，ARGO，2009，http://www.argocorp.com/cam/special/HeadWall/how_it_works.html
2. 陸域観測技術衛星「だいち」（ALOS），JAXA，2015，http://www.jaxa.jp/projects/sat/alos/index_j.html
3. AVIRIS, NASA JPL, http://aviris.jpl.nasa.gov/
4. Sergio Bernabe, et al., “Hyperspectral Unmixing on GPUs and Multi-Core Processors: A Comparison”, *IEEE Journal of selected topics in applied earth observations and remote sensing,* vol.6, No.3, pp. 1386-1398, 2013
5. Jose M.P. Nasciment, Jose M. Bioucas Dias, “Vertex Component Analysis: A Fast Algorithm to Unmix Hyperspectral Data”, *IEEE Transactions on geoscience and remote sensing,* vol. 43, No.4, pp.898-910, 2005

References

6. Agustin Ifarranguerri, Chein-I Chang, “Multispectral and Hyperspectral Image Analysis with Convex Cones”, *IEEE Transactions on geoscience and remote sensing,* vol. 37, No. 2, pp.756-770, 1999
7. Sebastian Lopez, et al., “A Low-Computational-Complexity Algorithm for Hyperspectral Endmember Extraction: Modified Vertex Component Analysis”, *IEEE geoscience and remote sensing letters,* vol.9, No. 3, pp.502-506, 2012

Acknowledgements

1. Icon made by Freepik from www.flaticon.com is licensed under CC BY 3.0