

# DIFFICULTIES OF DEALING WITH LARGE IMAGE CUBES

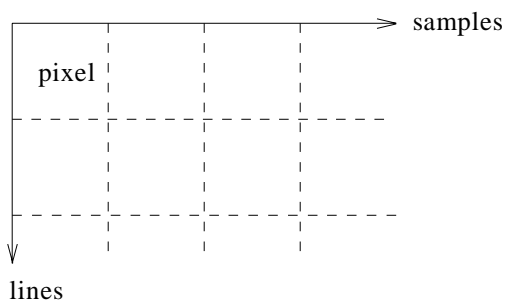
James Frew  
Computer Systems Laboratory  
University of California  
Santa Barbara, CA 93106  
frew@ucsb.edu

## Abstract

Image cubes are three-dimensional image datasets, in which the third dimension, typically spectral, is of the same order of magnitude as the line and sample dimensions. Image cubes are more difficult to manipulate than 2-dimensional images, both because of their increased data volume, and because of the significant additional information they contain. Conventional image processing systems often do not deal gracefully with images with a significant third dimension. Preprocessing to reduce data volume, concentrate on regions of interest, and accommodate existing systems occupies a considerable amount of effort in image cube analysis. With the exception of software and hardware for image display, current trends in the computing industry are optimistic for the future effective utilization of image cube datasets that will be available in the EOS era.

## 1. Introduction

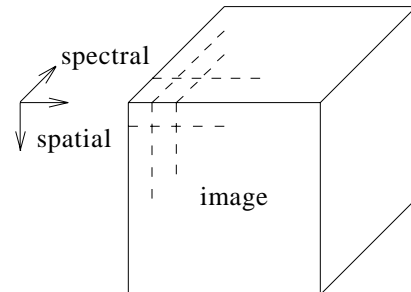
Digital images have traditionally been conceptualized and manipulated as 2-dimensional objects, usually as a rectangular grid of *pixel* (picture element) values. The most typical nomenclature derives from video displays: the image consists of *lines* ordered vertically, each line being subdivided into an equal number of *samples*:



There have been two widespread extensions to this model to accommodate a third dimension. The most obvious is color imagery, where each pixel is actually a (red,green,blue) tuple. Many hardware and software systems for image processing restrict the third dimension of the image to these three members,

plus perhaps an overlay or transparency value. The other way to simulate a third dimension is by logically grouping 2-dimensional images whose pixels are spatially coincident, and applying the same operation (or a component thereof) to each image in the group. Most 2-dimensional image processing systems, as well as raster-based geographic information systems allow operations of this kind<sup>1</sup>.

The advent of air- and spaceborne multispectral imaging systems began to reveal the shortcomings of these approaches to 3-dimensional image processing; for example, integrated hardware-software environments which use color frame buffers for processing as well as display cannot process all seven Landsat Thematic Mapper bands simultaneously. The EOS era will complete this “mono to many” transition, as currently experimental hyperspectral systems such as AVIRIS<sup>2</sup> (report 224 bands) are succeeded by HIRIS<sup>3</sup> (192 bands) and operational systems such as MODIS<sup>4</sup> (36-64 bands). Images from such datasets may be referred to as *image cubes*<sup>5</sup>, to emphasize the fact that the cardinalities of all 3 dimensions may be of the same order of magnitude:



## 2. Problems

The storage and processing of large image cubes will present formidable problems. The presence of a significant third dimension means that seemingly modest-sized datasets are in fact quite voluminous. For example, a standard AVIRIS image has the following dimensions:

2	bytes/band
224	bands/sample
614	samples/line
512	lines/image
140,836,864	bytes/image

Copyright © 1990 by James Frew. Published by the American Institute of Aeronautics and Astronautics with permission.

A 134 Mbyte\* file is large by the current standards of scientific computing, as the following example capacities suggest:

storage medium	# AVIRIS images
2400-foot 6250 bpi 9 track tape	1
540-Mbyte CD-ROM	4
1-Gbyte magnetic disk	7
2.3-Gbyte 8mm tape	17

Obviously an environment that will be dealing with multiple image cubes will require a substantial amount of online storage.

### I/O Throughput

In addition to the space requirements, image cubes will tax the I/O throughput limitations of most computing systems. Simply reading hundreds of megabytes can come to dominate the execution time of a program. The following timings were noted for programs doing nothing but reading an entire AVIRIS image. The “local” time was measured on a Sun-3/280 with a local SMD (storage module drive) disk, while the “remote” time was obtained on a Sun-4/65 accessing the same disk via NFS (Network File System) over a 10 Mbit/sec Ethernet:

disk	min:sec	Kbyte/sec
local	3:38	630
remote	4:18	533

The systems and network on which these timings were obtained were by no means optimized for I/O throughput; in this respect, they are probably representative of many current end-user scientific computing environments. Still, note that the measured throughput on the remote system is within a factor of 2 of the theoretical Ethernet maximum<sup>6</sup>.

### Memory

The amount of memory required by an image processing algorithm is governed by 2 factors:

**Random access.** Many algorithms access their input and/or output data in a non-sequential fashion, either systematically (e.g., the input to a convolution algorithm) or unpredictably (e.g., an output histogram array). Since external data are often accessed via strictly sequential media (e.g., tapes, interprocess communication channels, etc.), such algorithms are more generally useful if their implementation maintains the data’s domain of non-sequential access (e.g., convolution kernel, histogram array, etc.) in memory.

**Buffering.** Whereas random access may be a fundamental constraint imposed by the algorithm, buffering is usually a less rigid tradeoff of memory for processing speed. In situations where random access

\* As used in this paper, the prefixes K, M, G, and T indicate factors of  $2^{10}$ ,  $2^{20}$ ,  $2^{30}$ , and  $2^{40}$ , respectively.

to external data is possible, it is usually so much slower than memory access that implementations attempt to maintain as much of the data as possible in memory. Even strictly sequential algorithm are often implemented this way, on the assumption that memory is a cheaper system resource than I/O system calls.

Image cubes complicate the evaluation of these factors by adding potentially another order of magnitude to an algorithm’s memory requirements. Implementations that blithely assume that whole images will fit in memory may have to be rewritten to accommodate image cubes. In particular, applications whose usability is derived largely from the speed of memory-based processing may prove unsuitable for image cubes.

### Interleaving

The memory requirements of one- or two-dimensional algorithms may be reduced by appropriate prior *transposition* of the image cube axes; i.e., by changing the interleaving of the third (band) dimension with respect to the other (line, sample) dimensions, so that the dimensions in which the algorithm operates are presented contiguously. The three possibilities are *band sequential* (BSQ), *band interleaved by line* (BIL), and *band interleaved by pixel* (BIP)<sup>7</sup>:

interleaving	subscript order
BSQ	band, line, sample
BIL	line, band, sample
BIP	line, sample, band

where the subscript order is from most to least significant.

Unfortunately, no single interleaving strategy is optimal for all algorithms. For example, BSQ interleaving would pose the least memory requirements for neighborhood operations such as convolution, or for algorithms which must maintain image-wide data structures such as histograms. On the other hand, BIP interleaving is obviously the most appropriate for operations such as spectrum matching and classification. BIL interleaving is a popular if sub-optimal compromise between these extremes. Supporting all interleavings is not necessarily a solution, since the required cube transpose operations are themselves quite memory- and/or I/O- intensive.

### Data Compression

One way to deal with the large data volumes of image cubes is by some form of data compression. There are two basic approaches to this.

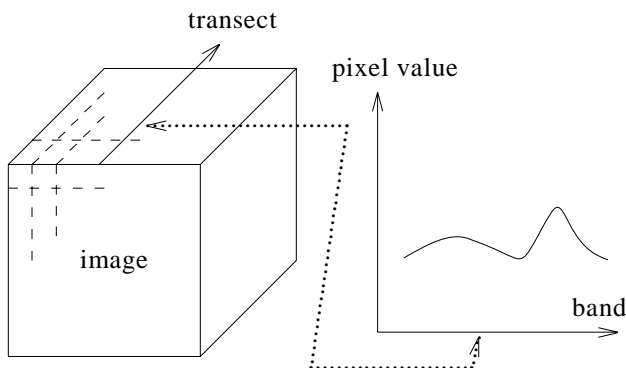
**Editing.** Prior to a processing sequence, an image cube may be edited to discard portions that are irrelevant to the particular application. Editing can in turn be implemented by subsampling, combination, or dimensionality reduction.

Subsampling involves discarding whole contiguous portions of the image cube. In the spatial domain, subsampling would be implemented by omitting certain ranges of lines and/or samples (i.e., extracting a *window* from the original cube). Similarly, spectral subsampling selects only those bands that are required by the current processing sequence. For example, most investigations of Earth surface processes could safely ignore spaceborne spectral data acquired in the atmospheric water vapor absorption bands.

For sensors with large dynamic ranges, it may also be appropriate to subsample the image cube radiometrically; i.e., to extract only a subset of the bits for each pixel<sup>3</sup>. For example, visible reflectance data acquired over bodies of water may well contain information only in the low-order bits. For sensors such as AVIRIS that yield 10-bit pixels, ignoring the 2 high-order bits can result in a factor of 2 reduction in data volume (since 10-bit pixels are usually stored in 2-byte integers).

Combination is useful when the entire image cube must be processed, but a reduced resolution in one or more dimensions is acceptable. The simplest form of combination is averaging. Spatial averaging or “smoothing” is a familiar and well-understood image processing operation. Spectral averaging is less ubiquitous, since until recently there was no data available on which it would make sense. Spectral averaging is also more likely to be implemented with a nonlinear smoothing function, either to approximate the spectral response of particular sensor systems (e.g., simulating Thematic Mapper spectral response from HIRIS data), or to guarantee that full spectral resolution is preserved for certain sharply-defined features (e.g., mineral absorption).

Dimensionality reduction is achieved by holding one or two cube dimensions constant, and extracting the for the remaining dimensions. If two dimensions are held constant, the result is a 1-dimensional *transect*. Transects are typically interpreted graphically; for example, in a hyperspectral image, a transect the bands at a given line aand sample yields a graph of that pixel’s spectrum:



If one dimension is held constant, the result is a 2-dimensional *plane*. Planes are typically interpreted as monochrome images; groups of 3 planes may be viewed as color images.

The ability to interactively select transects and planes, and to display them in real time, has proven to be an extremely effective tool for “browsing” image cubes<sup>8</sup>.

**Compression.** Data compression is usually thought of as converting a data stream to a more compact but equivalent representation. The key lies in the definition of “equivalent”. Lossless compression algorithms by definition may be inverted to yield the *exact* input, while lossy algorithms may not be fully invertible, yet may yield a usable enough representation to be considered *functionally* equivalent. Editing and combination are, in one sense, “lossy” compression algorithms that irreversibly eliminate some features of the original data, yet preserve all information required by a particular application.

Lossless compression algorithms are attractive because they do not require any irreversible decisions to be made about what features of the image cube are important. A typical lossless algorithm is the “Lempel-Ziv-Welch” algorithm<sup>9</sup>, which is available on most current computer systems. Applying this algorithm to a typical AVIRIS scene yielded the following result:

status	Mbytes	generation time (mins)
un-compressed	134	3.8
compressed	66	15.5

The times are CPU-only measurements on a Sun-4/65 workstation. Note that while compressing the image is quite slow, subsequent un-compressing is relatively fast. This would tend to encourage compressed storage of image cubes, especially if the same asymmetry carries over into hardware implementations of the LZW algorithm.

The LZW algorithm is generic; although it adapts itself to some extent to the characteristics of the input data, it cannot perform as well as a compression scheme that takes account of the specific statistical properties of a given image cube. Studies of existing multispectral image datasets<sup>10</sup> indicate that compression ratios of 3:1 may be achievable with simple predictive coding algorithms. It remains to be seen if these observations will carry over to hyperspectral data.

Lossy compression algorithms can achieve substantial increases in compression over lossless algorithms, if some irreversible data loss can be tolerated. Vector quantization algorithms achieving 20:1 compression ratios have been shown to have minimal impacts on certain analyses (spectrum matching; mixture component analysis) commonly performed on

hyperspectral data<sup>11</sup>. It has been proposed to place optional real-time lossy compression facilities on HIRIS so that the usefulness of these compression schemes may be evaluated in the widest variety of contexts.

### Ancillary Data

In discussing the data volumes implied by image cubes, the proliferation of ancillary data should not be overlooked. Ancillary data is that information required to interpret an image cube, in addition to the actual pixel values. While the volume of such data should always remain much less than that of the image cube itself, it will not be negligible.

A common form of ancillary data is radiometric calibration information. Each detector in an imaging sensor must be calibrated; since these calibrations drift over time (perhaps even over a single scene), calibration coefficients for each detector are usually provided with each image. Hyperspectral sensors will have an enormous number of detectors; this is due both to the number of spectral bands, and the replacement of mechanical scanning mechanisms with ‘pushbroom’ detector arrays. For example, the proposed configuration for HIRIS has 1000 detectors for each of 192 bands<sup>12</sup>.

## 3. Future Directions

Many of the difficulties of dealing with image cubes will be ameliorated to some extent by future developments in computer hardware and software that may be readily extrapolated from current trends. Some of the more significant trends in scientific computing are summarized in this section.

### Distributed Computing

The overall trend in modern computing towards decentralization will allow more investigators access to image cube processing, both by providing remote access to image cube datasets and processing facilities, and by making it easier for investigators to construct and optimize their own processing environments.

**Networks.** The high-bandwidth wide-area networks expected in the EOS era<sup>13</sup> will facilitate the sharing of image cube datasets, something that can only be accomplished today by exchanging offline media. At the same time, local-area network technology is fast approaching, and even exceeding, the transfer rates normally associated with intra-system buses<sup>14</sup>. Just as fast wide-area networks will blur the distinction between remote sites, fast local-area networks will help complete the illusion that a group of computers is a single coherent system.

**Servers.** In order to fully exploit increasing network speeds, the provision of specialized functionality as identified network *services* will continue

to evolve. In a wide-area context, services can be envisioned which provide standardized access to image cube datasets (image servers), application processing (compute servers), and spectral libraries (database servers).

**Operating Systems.** The true promise of distributed computing will not be realized without fully distributed operating systems<sup>15</sup>. These will allow compute-intensive tasks to both connect themselves automatically to the appropriate network services, and to migrate between equivalent servers as changing utilization dictates<sup>16</sup>.

### Mass Storage

Current trends in mass storage will address the capacity, speed, and availability limitations currently affecting image cubes. For absolute maximization of offline storage volumes, new tape technologies, both optical and magnetic, promise to make multiple-Tbyte archives available to individual investigators. Optical disks will make GBytes to TBytes of data accessible online, although their access times and transfer rates are expected to significantly lag magnetic disks for the foreseeable future. Magnetic disk technology is now concentrating maximizing throughput, especially by arranging large arrays of relatively inexpensive disks in parallel<sup>17</sup>.

Of special interest is the emergence of hierarchical storage systems which integrate these various levels of storage into a single network file service, transparently migrating files between high-speed and high-volume storage according to frequency of use. This approach should be of great value to investigator computing environments, which typically lack the expertise to manage a collection of discrete heterogeneous storage systems<sup>18</sup>.

### Workstations

The emergence of the personal workstation as the primary computing resource of most Earth science investigators has profound implications for image cube applications. Three trends in workstation evolution are noteworthy:

**Processors.** The CPUs in workstations have been doubling in instruction execution speeds for several years, and this trend shows no sign of abating. Coupled with the migration of multiprocessing and massive parallelism to the workstation (or at least the local network) level, the increase in processing power will effectively allow the most complex image cube analyses to be executed under the control of the investigator. In particular, the projected widespread availability of parallel processing in the coming decade should have substantial impact on image cube analyses, since many image processing algorithms lend themselves easily to parallel implementations<sup>19</sup>.

**Memory.** The increasing density, and decreasing price, of computer memory should lead to the routine presence of image analysis workstations with hundreds of Mbytes, or even Gbytes, of main memory. This should eliminate many of the current memory restrictions on image cube processing. In addition, huge local memories will serve as effective caches for data accessed remotely<sup>20</sup>, which will overcome most of the bottlenecks in existing implementations of wide-area distributed file systems<sup>21</sup>.

**Image Displays.** Color displays with at least 24 bits of radiometric resolution (8 bits each for red, green, and blue) are becoming more readily available. This is one area, however, where the short term prognosis is not favorable for image cube manipulation, or for Earth-resources-oriented image processing in general. The reason is that most current workstation displays are oriented towards graphics rather than image processing. In graphics, a relatively terse high-level description must be rendered into a displayable image, while in image processing, the image is a given, and the high-level description is usually the goal of the processing sequence<sup>22</sup>. Workstation vendors have devoted considerable effort to graphics-related problems but relatively little to image processing. Image cube analysis would benefit greatly from a “dumb” frame buffer whose sole special feature was extremely rapid access to the computer’s main memory.

### Software

Advances in applications software will have subtler but no less important impacts on image cube processing.

**Visualization.** The emergent field of scientific visualization, concerned as it is with producing image from complex multidimensional data<sup>23</sup>, is likely to yield spinoff of value to image cube analysis. It is easy to imagine applying tools for volume visualization to, for example, display as complex solids the spectra within an image cube meeting certain criteria. Indeed, subtle spectral-spatial relationships may be impossible to discover by non-visual means.

**Standards.** As with display hardware, the state of the art in software is far more advanced for graphics than for imaging. In particular, several accepted standards exist for graphics interfaces, while there are as yet no accepted standards for image processing, although a few have been proposed<sup>24</sup>. Network protocols for image display are badly needed; current work on imaging extensions to the X window system is encouraging<sup>25</sup>.

**Feature Libraries.** The precision with which image cubes, especially hyperspectral ones, can characterize observed feature, can only be exploited by rapid access to libraries of reference features for comparison. Current image cube processing environ-

ments typically provide access to local libraries tailored for particular investigations (e.g., laboratory mineral spectra<sup>8</sup>). In the EOS era, these libraries will need to be maintained as a wide-area network service.

### References

1. J. Westervelt, *An Introduction to GRASS*, U.S. Army Construction Engineering Research Laboratory, Champaign, IL, August 1988. (technical report).
2. W. Porter and H. Enmark, A system overview of the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS), in *Imaging Spectroscopy II*, G. Vane (ed.), Proc. SPIE 834, 1987, 22-31.
3. A. Goetz, *HIRIS : High-Resolution Imaging Spectrometer : Science Opportunities for the 1990s*, Earth Observing System Instrument Panel Report, vol. IIc, National Aeronautics and Space Administration, Washington, DC, 1987.
4. W. Esaias, *MODIS : Moderate-Resolution Imaging Spectrometer*, Earth Observing System Instrument Panel Report, vol. IIb, National Aeronautics and Space Administration, Washington, DC, 1986.
5. T. Martin, M. Martin, R. Davis, R. Mehlman, M. Braun and M. Johnson, *Planetary Data System Standards for the Preparation and Interchange of Data Sets, Version 1.1*, Jet Propulsion Laboratory, Pasadena, CA, 03 October 1988. (document D-4683).
6. M. Nemzow, *Keeping the Link : Ethernet Installation and Management*, McGraw-Hill, New York, 1988.
7. D. Simonett, T. Smith, W. Tobler, D. Marks, J. Frew and J. Dozier, *Geobase Information System Impacts on Space Image Formats*, Remote Sensing Unit, Department of Geography, University of California, Santa Barbara, CA, April 1978. (SBRUS Technical Report 3).
8. A. Mazer, M. Martin, M. Lee and J. Solomon, Image processing software for imaging spectrometry, in *Imaging Spectroscopy II*, G. Vane (ed.), Proc. SPIE 834, 1987, 136-139.
9. T. A. Welch, A technique for high-performance data compression, *IEEE Computer* 17, 6 (1984), 8-19.
10. T. M. Chen, D. H. Staelin and R. B. Arps, Information content analysis of Landsat image data for compression, *IEEE Transactions on Geoscience and Remote Sensing GE-25*, 4 (1987), 499-501.

11. R. L. Baker and Y. T. Tse, Compression of high spectral resolution imagery, in *Applications of Digital Image Processing XI*, A. Teschner (ed.), Proc. SPIE 974, 1988, 255-264.
12. J. Conley, M. Herring and D. Norris, High resolution imaging spectrometer (HIRIS), in *Imaging Spectroscopy II*, G. Vane (ed.), Proc. SPIE 834, 1987, 188-195.
13. National Research Network Review Committee, *Toward a National Research Network*, National Academy Press, Washington, DC, 1988.
14. S. P. Joshi, High-performance networks : a focus on the fiber distributed data interface (FDDI) standard, *IEEE Micro* ??(1986), 8-14.
15. S. Mullender, G. van Rossum, A. Tanenbaum, R. van Renesse and H. van Staveren, Amoeba - a distributed operating system for the 1990s, *IEEE Computer*, May 1990.
16. M. Litzkow, M. Livny and M. Mutka, *Condor - a hunter of idle workstations*, University of Wisconsin, Madison, WI, 1987. (Computer Sciences Technical Report #730).
17. R. Katz, Winged memory, *UNIX Review* 7, 9 (1989), 59-66.
18. T. Taylor and R. Fortier, *Using optical disks to extend the capacity of magnetic disks through hierarchical storage*, Epoch Systems, Inc., Marlborough, MA. (technical report).
19. E. Davies, Image Processing - its Milieu, its Nature, and Constraints on the Design of Special Architectures for its Implementation, in *Computing Structures for Image Processing*, M. Duff (ed.), Academic Press, London, 1983, 57-76.
20. D. Chandler, Interview with Wayne Rosing, *UNIX Review* 8, 1 (1990), 82-89.
21. M. Satyanarayanan, Scalable, secure, and highly available distributed file access, *IEEE Computer*, May 1990, 9-21.
22. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, MD, 1982.
23. K. A. Frenkel, The art and science of visualizing data, *Communications of the ACM* 31, 2 (1988), 111-121.
24. P. Krolak, R. Coelho and R. Miner, A declaration of device independence, *Electronic System Design*, May 1988, 53-56.
25. J. Mauro, *X Image Extension Concepts, version 2.4*, Digital Equipment Corp., Merrimack, NH, 10 June 1988. (technical report).