

Computation-based Improvements for Hyperspectral Data Processing

Dissertation

zur

**Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)**

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

P. Jason Brazile

aus

den Vereinigten Staaten von Amerika

Promotionskomitee

Prof. Dr. Klaus I. Itten (Vorsitz)

Prof. Dr. Klaus Dittrich

Dr. Jens Nieke

Prof. Dr. Michael E. Schaepman

Zürich 2007

Remote Sensing Series Editorial Board:

Prof. Dr. K. I. Itten, Dr. E. Meier Dr. T. Kellenberger, Dr. M. Kneubühler, Dr. D. Small

Author:

P. Jason Brazile
Remote Sensing Laboratories
Department of Geography
University of Zurich
Winterthurerstrasse 190
CH-8057 Zurich
Switzerland

jbrazile@brazile.net
<http://www.geo.uzh.ch>

<p>Brazile, P. Jason: Computation-based Improvements for Hyperspectral Data Processing Remote Sensing Series Vol. 51 – Zurich: Remote Sensing Laboratories, 2007 ISBN 978-3-03703-017-2</p>

©Copyright 2007 by P. Jason Brazile. All rights reserved unless otherwise indicated.

Printed by the Druckerei der Zentralstelle der Studentenschaft der Universität Zürich.

Abstract

The University of Zurich's Remote Sensing Laboratories leads the development of the Swiss-Belgian APEX (Airborne Prism EXperiment) airborne imaging spectrometer project, sponsored in large part by the European Space Agency's PRODEX (PROgramme de Developpement d'Experiences) funding authority. The APEX project was conceived in 1996 and instrument construction began in 2002. Currently, the optical components are being integrated and the first test of the instrument is expected in October 2007.

The APEX project serves to further scientific research in several areas that could roughly be categorized into two: (A) the broad mapping of geoscientific data in general and (B) improvements in understanding geological, biological, chemical, and physical processes in particular. The data acquisition goals are intended to provide accurate and unique data for core earth science applications in studies involving limnology, soil, vegetation, mineralogy, snow, aerosols, and climate among others. Such core applications in turn contribute to improvements in the higher order retrieval of geological, biological, chemical or physical parameters such as the chlorophyll fluorescence of plants. Measurements are performed in the spectral range between 380 and 2500 nm. However in order to achieve quantitatively and qualitatively worthwhile results, it is important to better understand various areas of data acquisition (instrument construction) and data processing. This provided the APEX project's motivation for research into application-driven instrument design, trustworthy calibration/characterization processes and algorithms, and the consistent spatial/spectral uniformity of hyperspectral data products.

The breadth of the scientific areas involved and their resulting critical research questions led to the necessity that the APEX research team be put together in an interdisciplinary manner. As a specialist in computer science, the author of this thesis has focused on the application of software engineering principles and the use of distributed and parallel techniques for exploring novel instrument characterization processing algorithms in order to achieve the required quality goals of the overall project.

Thanks to the support of the APEX project, contributions in this thesis have produced advances in three particular areas. The reoccurring need for an easy-to-use, distributed batch processing scheduler led to the development of a batch scheduler web-service and accompanying processing client. This simple "grid tool" has already been successfully deployed in two projects peripheral to APEX. Exploration in scene-based instrument characterization led to important improvements in the scene-based retrieval of an instrument's spectral response function (SRF) shape, which can be used to more accurately derive surface reflectance products from at-sensor radiance measurements. And, finally, using a componentized architecture and prototyped examples, the case has been made for a loosely-coupled processing system based on modular reference implementations of published algorithms in order to better exploit economical "network effects" in furthering hyperspectral image processing research.

These contributions help to improve the quality of hyperspectral data products by complementing the focused research of other APEX team members, including rigorous forward- and inverse-modeling of the instrument, algorithms for the maximization of spectrally and spatially uniform data, and the application-driven definition of end-user data products.

Zusammenfassung

Die Abteilung Fernerkundung (Remote Sensing Laboratories) des Geographischen Instituts der Universität Zürichs führt ein Schweiz-Belgisches Konsortium an, das auf die Entwicklung und den Betrieb des flugzeuggestützten Bildspektrometers APEX (Airborne Prism EXperiment) abzielt. Dieses Projekt wird grösstenteils von dem PRODEX (PROgramme de Developpement d'Experiences) Programm der Europäischen Raumfahrtbehörde (ESA) finanziert. Das APEX Projekt wurde 1996 erstmalig konzeptionell vorgestellt und 2002 fing die Entwicklung des Geräts an. Zur Zeit werden die elektro-optischen Komponenten zusammengebaut und der erste Test des Instruments ist für Oktober 2007 vorgesehen.

Das APEX Projekt dient der Forschung in verschiedensten Bereichen, die grob in zwei Richtungen eingeteilt werden können: (A) die weitflächige Kartierung von geowissenschaftlichen Daten im allgemeinen und (B) die Verbesserung des Verständnisses der geo-, bio-, chemophysikalischen Prozesse im speziellen. Die Datenerfassung zielt darauf ab, eine exakte, zuverlässige Kartierung für verwandte Erd- und Atmosphärenwissenschaften (e.g., Mineralogie, ökologie, Limnologie, Schnee-, Aerosol- und Klimaforschung) zu ermöglichen. In diesen Wissenschaftsbereichen tragen die kartierten Daten dazu bei, die Herleitung von geo-, bio-, chemophysikalischen Parametern, wie etwa die Chlorophyllfluoreszenz von Pflanzen, besser oder einfacher flächenhaft zu erfassen. Die Messungen werden im Spektralbereich zwischen 380 und 2500 nm absolviert. Um jedoch quantitativ und qualitativ hochwertige Resultate zu erzielen, ist es notwendig verschiedene Bereiche der Datenerfassung (Gerätebau), und Datenaufbereitung (Prozessierung) besser zu verstehen. Dies führte dazu, dass sich das APEX Projekt mit dem anwendungsbasierten Gerätebau von Bildspektrometern befasst, wofür insbesondere die Motivation für die Bereiche Kalibrierung-Charakterisierung von Prozessen und Algorithmen, und die einheitliche räumlich-spektrale Gleichmässigkeit hyperspektraler Datenprodukte stellvertretend sind.

Die Auflistung dieser Forschungsschwerpunkte und der resultierenden wissenschaftlichen Fragestellungen bringt es mit sich, dass das APEX Forschungsteam sehr interdisziplinär zusammengesetzt sein muss. Als Informatikexperte, hat sich der Autor dieser Doktorarbeit an der Anwendung von Softwaretechnikgrundsätzen (Software Engineering Principles) und dem Gebrauch von verteilten und parallelen Charakterisierungsalgorithmen (Distributed and Parallel Characterization Processing Algorithms) fokussiert, um die geforderten hohen Qualitätsziele des Gesamtprojekts erreichen zu können.

Mit Unterstützung des APEX Projekts haben Beiträge in dieser Dissertation zu Fortschritten in drei Bereichen geführt: Die immer wieder auftauchende Notwendigkeit für bedienungsfreundliche, verteilte, Stapelverarbeitungssteuerprogramme (Easy-to-use, Distributed Batch Processing Scheduler), führte zur Entwicklung eines Batch Scheduler Web-Services und eines begleitenden Client-Programms. Dieses einfache "Grid Werkzeug" ist bereits in zwei APEX Nebenprojekten erfolgreich eingesetzt worden. Bei der Untersuchung von aufnahme-orientierten Gerätecharakterisierungen (Scene-based Instrument Characterization) ergaben sich wichtige Fortschritte bei der Herleitung der spektralen Antwortfunktionen (Spectral Response Functions) eines Spektrometers, die helfen, genauere Bodenreflektanzen aus der Strahlungsmes-

sungen am Flugzeug abzuleiten. Und schliesslich, durch die Anwendung von komponentenbasierter Architektur und Prototyping, wurde die Voraussetzung für ein loses-gekoppeltes Verarbeitungssystem (Loosely-coupled Processing System) geschaffen. Dieses Verarbeitungssystem basiert auf Referenzimplementationen veröffentlichter Algorithmen, um den ökonomischen "Netzwerk Effekt" (Network Effect) in der hyperspektralen Bildbearbeitung besser zu verstehen.

Die Beiträge der vorliegenden Dissertation helfen, die Qualität der hyperspektralen Datenprodukte zu verbessern, wodurch beispielsweise die Arbeit anderer APEX Teammitglieder im Bereich der Vorwärts- und Invers- Modellierung und, der Algorithmenentwicklung für die Gleichmässigkeit von spektralen und räumlichen Daten, substantiell unterstützt wurden.

Contents

Abstract	I
Zusammenfassung	III
Table of Contents	VII
List of Figures	X
List of Tables	XI
List of Abbreviations	XIII
1 Introduction	1
1.1 Rationale of Dissertation	2
1.2 Scientific Setting	3
1.2.1 The APEX Instrument	3
1.2.2 Scene-based spectrometer calibration/characterization	6
1.2.3 Hyperspectral Data Processing	6
1.3 Research Questions	7
1.3.1 Data Acquisition	7
1.3.2 Distributed Data Processing	8
1.3.3 Instrument Characterization	9
1.3.4 Componentization of Hyperspectral Processing Algorithms	10
1.4 Structure of Thesis	10
2 Software Architecture for Acquisition and Post-Processing of Hyperspectral Data	19
2.1 Introduction	20
2.2 In-Flight Data Acquisition	21
2.2.1 Bandwidth/Throughput	22
2.2.2 Device Interrupt Latency	24
2.2.3 Time Synchronization	24
2.2.4 Other features	26
2.2.5 Performance	26
2.3 Scientific Post-Processing	27
2.3.1 Key Algorithms in Foreign Language	28
2.3.2 Additional Components	28
2.3.3 Development Process	29
2.4 Conclusions and Outlook	30

3	A Domain-Independent RESTful Grid Service for Non-Programmers	35
3.1	Introduction	36
3.2	Implementation	37
3.2.1	Design Goals	37
3.2.2	Resource-Oriented vs Service-Oriented	38
3.2.3	Resource Oriented Service Interface	38
3.3	Example Deployments	40
3.3.1	Atmospheric Modeling	40
3.3.2	Satellite Mission Planning via Simulation	41
3.3.3	Render Farm	41
3.3.4	Document Digitizing	41
3.4	Related Work	42
3.5	Current Status and Future Work	42
3.5.1	Usability and Efficiency	42
3.5.2	Recommended Improvements	43
3.6	Summary	43
4	Cluster versus Grid for Generation of ATCOR's MODTRAN-based Look Up Tables	47
4.1	Introduction	48
4.2	Method	49
4.2.1	High-level granularity of MODTRAN processing	49
4.2.2	Parallel Decomposition	50
4.2.3	Numeric Consistency Evaluation	50
4.2.4	Runtime Performance Evaluation	52
4.2.5	Grid versus Cluster	52
4.2.6	Condor vs Low Fat Grid	53
4.2.7	Scheduling	54
4.3	Results	57
4.4	Discussion	57
4.4.1	High-level granularity of MODTRAN	57
4.4.2	Numeric Consistency Evaluation	58
4.4.3	The Role of Middleware	58
4.4.4	Runtime Performance Evaluation	60
4.4.5	Per node statistics	62
4.4.6	Per job statistics	62
4.4.7	Scalability	63
4.5	Conclusions	63
5	Scene-Based SRF Shape Discernibility for the APEX Imaging Spectrometer	69
5.1	Introduction	70
5.2	Method	71
5.3	Results	72
5.4	Discussion	72
5.5	Conclusions	74
6	Toward Scene-Based Retrieval of SRFs	79
6.1	Introduction	80
6.1.1	Motivation	80
6.1.2	Scene-based Retrieval	81
6.2	Method	83
6.3	Results	89
6.4	Discussion	93

6.4.1	Effectiveness of Method	93
6.5	Conclusions	95
6.6	Acknowledgments	96
7	Streaming, Language-Neutral, Hyperspectral Image Processing Components	101
7.1	Introduction	102
7.2	Design Goals For Promoting Longevity	103
7.3	Implementation	104
7.3.1	Streaming/Archiving	104
7.3.2	Data Structures	105
7.3.3	Algorithm Encapsulation by Callback	105
7.3.4	Parallelization	105
7.4	Examples	106
7.5	Related Work	108
7.6	Current Status and Future Work	109
8	Computation-based Improvements for Hyperspectral Data Processing - A Synthesis	113
8.1	Introduction	113
8.2	Synopsis	114
8.2.1	Data Acquisition	114
8.2.2	Distributed Data Processing	115
8.2.3	Scene-based Instrument Characterization	115
8.2.4	Componentization of Hyperspectral Processing Algorithms	115
8.3	Conclusions	116
8.4	Outlook	116
	Curriculum Vitae	121
	Bibliography	123



List of Figures

1.1	Simplified Overview of APEX Processing. Colored boxes represent processes that are both enhanced by distributed/parallel techniques and addressed in individual chapters in this thesis.	3
1.2	Selected specifications of the APEX instrument	4
1.3	Overview of APEX Level 2/3 processing	5
2.1	Data Acquisition Requirements	21
2.2	Subset of Data Acquisition Architecture	23
2.3	Time Synchronization	25
2.4	Post Processing Requirements	27
2.5	APEX Post Processing Architecture	29
3.1	Distribution of jobs on 82 heterogeneous worker machines (hostnames elided)	40
4.1	Stages of Processing	49
4.2	Numerics problems with case 4,332 of 45,056: circles/error bars show where any version differs 0.1% from mean of MODTRAN results produced from five common Fortran compiler/version combinations on an AMD Opteron. Note the negative values in Solar Scattering and Total Radiance (produced by two versions of a compiler from a single vendor)	51
4.3	Cluster vs. Grid program flow	54
4.4	Heterogeneous grid runtime estimation – relative CPU speeds of the 99 hosts are plotted along with an estimation of total runtime for a grid composed of increasingly more CPUs. It is estimated that 15 CPUs would run only 2 times slower than using all 99	55
4.5	Mean relative deviation of the per-case “profile” for each of the four platforms across all 45,056 MODTRAN executions – vertical lines denote cases with bad (e.g. negative) values	56
4.6	Mean and standard deviation of run times per node	58
4.7	Quadratic fit of node speeds (s) per job and extreme anomalies	59
4.8	Speedup/Efficiency of parallel MODTRAN jobs on cluster and grid	61
5.1	RMS (relative to gaussian) of radiances for varying target reflectance per SRF per solar feature for selected cases at 5 km visibility	73
6.1	Shape coverage of candidate SRF parameterization models.	81
6.2	Known SRF shapes and their commonly-used Gaussian approximations	82
6.3	Overview of experimental method	84
6.4	Varying instrument-specific LUT spectra for A(O ₂) feature window	87

6.5	Relative RMS (rRMS) versus case #. Relationship between case #'s and LUT input parameters	88
6.6	Evaluation of matching reference SRF(3,1.30) against LUT entries using rRMS .	90
6.7	Implied SNR requirements for two sub cases of SRF(3,1.30)	90
6.8	Same as Figure 6.6, but with varying simulated noise	94
7.1	APEX Forward instrument model (left) and inverse processing model (right) . . .	107

List of Tables

2.1	Selected APEX Specifications	20
2.2	Uses of embeddable/extendible scripting	30
3.1	Grid service resources and actions. PUT is implemented with POST for portability	38
3.2	Example: Message exchange for a batch containing a single job	39
4.1	Total running times in hours († See section 4.4.3)	57
5.1	Selected MODTRAN 4 input parameters	71
5.2	Instrument SNR requirements implied from relative RMS between candidate shape and gaussian (see Fig 5.1)	73
6.1	Selected MODTRAN 4 input parameters	85
6.2	Proposed instrument-specific LUT parameters for retrieval	86
6.3	Instrument specifications per feature window [nm]	86
6.4	Mean and standard deviation of rRMS for each reference spectra	91
6.5	Conservative SNR requirements (500, 1000, 5000, 10000, 50000, ∞) implied by simulation	92
7.1	Prototyped features deemed necessary for proving design goals	109



List of Abbreviations

AIS	Airborne Imaging Spectrometer
AMD	Advanced Micro Devices
ANSI	American National Standards Institute
AOC	APEX Operations Center
APEX	Airborne Prism EXperiment
API	Application Programming Interface
ASC	APEX Science Center
ASCII	American Standard Code for Information Interchange
ATCOR	ATmospheric CORrection
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
BEAM	Basic ERS and Envisat (A)ASTR and Meris Toolbox
BIL	Band Interleaved by Line
BIP	Band Interleaved by Pixel
BSQ	Band Sequential
CASI	Compact Airborne Spectrographic Imager
CCRS	Canada Centre for Remote Sensing
CD-ROM	Compact Disk Read Only Memory
CGI	Common Gateway Interface
CHB	Calibration Home Base
CHRIS	Compact High-Resolution Imaging Spectroscopy sensor
CJRS	Canadian Journal of Remote Sensing
CPU	Central Processing Unit
DAIS	Digital Airborne Imaging Spectrometer
DBMS	Database Management System
DDR	Double Data Rate dynamic random access memory
DISORT	DIScrete Ordinate Radiative Transfer
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DMA	Direct Memory Access
DVD	Digital Video Disk
EARSeL	European Association of Remote Sensing Laboratories
ENVI	Environment for Visualizing Images

ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
FMS	Flight Management System
FOV	Field of View
FPGA	Field Programmable Gate Array
FWHM	Full Width at Half Maximum
GNU	GNU's Not Unix
GPS	Global Positioning System
GUI	Graphical User Interface
HALO	High Altitude Long Range Research Aircraft
HITRAN	High resolution transmission molecular absorption
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HYDICE	Hyperspectral Digital Imagery Collection Experiment
IDL	Interactive Data Language
IEEE	Institute of Electrical and Electronics Engineers
IFOV	Instantaneous Field of View
INS	Inertial Navigation System
IP	Intellectual Property or Internet Protocol
ISDAS	Imaging Spectrometer Data Analysis System
ISP	Internet Service Provider
IT	Information Technology
JRE	Java Runtime Environment
LUT	Look up table
MD5	Message Digest algorithm 5
MERIS	Medium Resolution Imaging Spectrometer
MODIS	Moderate Resolution Imaging Spectrometer
MODTRAN	Moderate Resolution Transmittance Code
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
NFS	Network File System
NIR	Near-infrared range of the electromagnetic spectrum
NIST	National Institute of Standards and Technology
NPOC	National Point of Contact
OS	Operating System
OS	Optical Character Recognition
PAF	Processing and Archiving Facility
PAM	Portable Arbitrary Map (see PBM)
PBM	Portable Bit Map programs

PCI	Peripheral Component Interconnect
PC	Personal Computer
PDF	Portable Document Format
PI	Principal Investigator
POSIX	Portable Operating System Interface
PRODEX	PROgramme de Developpement d'Experiences
PVM	Parallel Virtual Machine
RAID	Redundant array of inexpensive disks
RAM	Random Access Memory
RDBMS	Relational Database Management System
REST	REpresentational State Transfer
RMS	Root Mean Square
RSL	Remote Sensing Laboratories
RT	Radiative Transfer
SIPS	Spectral Image Processing System
SI	Système International (d'Unités)
SLOC	Source lines of code
SMIRR	Shuttle Multispectral Infrared Radiometer
SNR	Signal-to-Noise Ratio
SPAM	SPectral Analysis Manager
SPARC	Scalable Processor ARChitecture
SPIE	Society for Photo-Optical Instrumentation Engineers
SQL	Structured Query Language
SRF	Spectral Response Function
SSI	Spectral Sampling Interval
SSL	Secure Sockets Layer
SWIR	Short-wave infrared range of the electromagnetic spectrum
TGARS	IEEE Transactions on Geoscience and Remote Sensing
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USA	United States of America
UTC	Coordinated Universal Time
VITO	Vlaamse Instelling Voor Technologisch Onderzoek
VNIR	Visible Near-infrared range of the electromagnetic spectrum
XML	eXtensible Markup Language
YAML	Yet Another Markup Language

Chapter 1

Introduction

Earth-observing imaging spectrometry concerns the simultaneous acquisition of spatially co-registered images in several spectrally contiguous bands measured in calibrated radiance units from a remotely operated platform [Goetz et al., 1985]. Its emergence as a separate endeavor from earlier multispectral investigations began in the late 1970s when feasibility was established for the exact identification of surface materials. This provided a marked improvement over the more basic classification made possible by multispectral imagers such as Landsat MSS and began with the experimentation with novel mobile ground based reflectance spectrometers. One of the first such instruments was used in 1974 to create a series of measurements in the Goldfield Mining District in Nevada that yielded identification results for iron minerals in the 800-1000 nm region and many OH-bearing minerals in the 2000-2500 nm region [Goetz et al., 1977]. Among other interesting results, such experiments implied that instruments providing 10nm resolution would be sufficient for allowing remote identification of many interesting materials.

One of the first scientific articles involving a remote spectrometer appeared in 1978, describing “a remote sensing system for use in light aircraft [based upon a] parallel electro-optical input spectroradiometer configuration with 500 channels operating in the 400-1100 nm region [at a] resolution of 18 meters square from an altitude of 600m at 200km/h [obtaining] 4-digit spectral radiance data at 2.5 spectra/sec on a 9-track tape in computer compatible format” [Chiu and Collins, 1978]. This system was later “improved by addition of a solid state silicon detector array [and] extended into the infrared by addition of a 64 element lead sulfide detector array” [Collins et al., 1981].

The first spaceborne experiment defined the next big step in instrument evolution. The Shuttle Multispectral Infrared Radiometer (SMIRR) was developed at the NASA/Caltech Jet Propulsion Laboratory to be deployed on the second flight of Shuttle (STS-2) in 1979, but was delayed until 1981. This experiment resulted in the first direct identification of soil minerals from orbit [Goetz et al., 1982]. At about this time, the appearance of the first commercially available hybrid focal plane arrays encouraged NASA/JPL’s development of the Airborne Imaging Spectrometer (AIS), first flown in 1982 [Vane et al., 1983]. This later evolved into the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) [Macenka and Chrisp, 1987], which through multiple upgrades over the years has maintained a central role in earth observing imaging spectroscopy – even as alternatives began to emerge, such as CASI [Babey and Anger, 1989], DAIS [Chang et al., 1993], HYDICE [Basedow et al., 1995] and HYMAP [Cocks et al., 1998].

However, the development of the optics, electronics, packaging and integration of the remote sensing instruments themselves has been a necessary but not sufficient condition for establishing hyperspectral imaging as a scientific endeavor. A large number of technical as well as

political supporting activities were needed to establish a scientific foothold for this burgeoning field.

From the start, applications were needed to motivate the political decisions required to support further expenditure on research in multiple costly emerging technologies. Fortunately, in addition to the originally proven geologic applications, promising studies in the use of remote imaging spectroscopy quickly emerged in a broad range of fields such as limnology, soil, vegetation, snow, aerosols and climate (for more detail, see [van der Meer and de Jong, 2002], [Chang, 2003], and [Schaepman, 2007]).

To establish that the imagery is well-founded in science and not simply *ad hoc* engineering, several supporting technical activities were needed. Laboratory instruments and techniques needed to be developed for calibrating/characterizing these new instruments, including appropriate modeling of all potential aberrations [Schläpfer et al., 2004b]. With proper instrument modeling [Lee et al., 2000], accurate inverse processing can be performed for mapping recorded digital numbers to corresponding at-sensor radiance in scientific units. Once imagery in scientific units had been derived, development of powerful statistical software was needed to provide basic consistency checks, support evaluations of data quality, and serve as a basis for producing higher-level products.

For validation purposes, appropriate portable field instruments needed to be developed along with corresponding algorithms and processes that would allow agreement to be established between overflights and simultaneous ground measurements [Secker et al., 2001]. Experiments involving these so-called vicarious calibration techniques revealed the importance of accurate atmospheric modeling software such as MODTRAN [Berk et al., 1998], together with the HITRAN database [Rothman et al., 2003], which provided the basis for corresponding “atmospheric correction” software such as ATREM [Boardman, 1998] or ATCOR [Richter, 1996] that allowed conversion from at-sensor radiance to surface reflectance. And finally, parametric geocoding needed to be performed [Schläpfer and Richter, 2002] to convert imagery into orthorectified imagery.

Only after this complex chain of processes and algorithms have been applied, can the surface reflectance data as derived from the digital number measurements from the instrument be used for end-user applications. From the point of view of systems developers, this imagery data is the end product – itself already the result of a substantial application of challenging science. However, in the bigger picture, this is only the starting point from which interesting earth science applications can begin. In fact, hyperspectral imagery may be merely one component to be assimilated with others [Dorigo et al., 2006] in the quest for algorithms and techniques for revealing ever more complex characteristics of the earth.

1.1 Rationale of Dissertation

The large body of work described in the previous section laid a solid foundation for earth-observing imaging spectrometry. Basic needs have been met for instrument design, data collection, instrument characterization, calibration/validation, image geocoding, and atmospheric correction. Additionally, on the application side, motivating research across a wide range of fields has been identified, with new discoveries emerging regularly.

Recent advances in earth observation imaging spectroscopy research [Plaza et al., 2007] have included evolutionary improvement in already existing problem areas, but have also attempted to expand in new areas that draw on interdisciplinary collaboration with research in neighboring fields. Collaborations with optical physicists led to improvements in laboratory radiometric

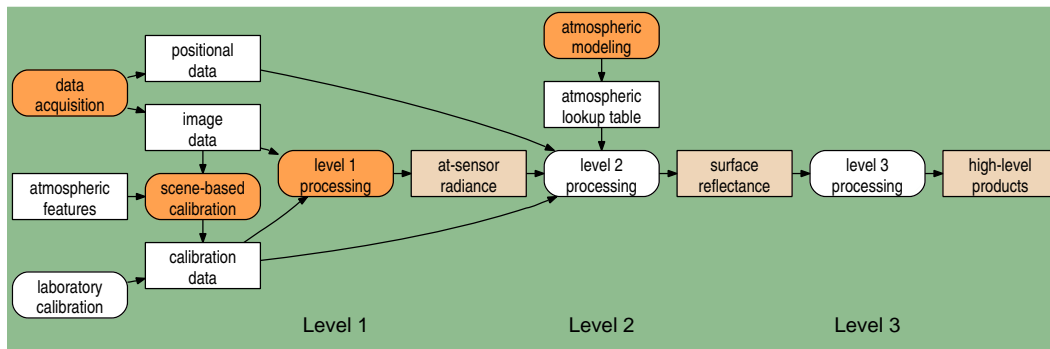


Figure 1.1: Simplified Overview of APEX Processing. Colored boxes represent processes that are both enhanced by distributed/parallel techniques and addressed in individual chapters in this thesis.

calibration [Schaeppman and Dangel, 2000] and in comparative goniometry for the support of spectrodirectional cross calibration [Dangel et al., 2005]. Collaborations with biophysicists investigated the supporting role that imaging spectroscopy can play in remotely estimating CO₂ flux [Gitelson et al., 2003]. Collaborations with computer scientists led to the use of mathematical image processing techniques toward improving unsupervised end-member extraction [Plaza et al., 2002] and the application of cluster and grid resources toward hyperspectral image processing [Brazile et al., 2004], [Plaza et al., 2006].

The unifying purpose of the work in this thesis has been to further maximize benefits made possible through interdisciplinary application of applied computer science toward advancing the state of the art in earth observing imaging spectroscopy. The first priority employed the use of finite state automata and bounded queuing theories to achieve data throughput needs arising in the engineering effort of assembling a new airborne instrument. The next priority involved developing an easy-to-use, firewall-transparent distributed processing tool to be used in several expected follow-on projects including atmospheric modeling, optimization (in the operations research sense), intellectual property blockage avoidance, and inverse modeling. The next priority involved using distributed processing toward the scene-based retrieval of a novel unknown – an instrument’s per-band spectral response function (SRF) shape. And the final problem addressed in this thesis is the application of the software engineering principles of loosely coupled, component-based, software architectures in addition to the economic principle of multiplicative network effects [Katz and Shapiro, 1994] to propose a way to improve quality and longevity while at the same time decreasing duplication of effort in the development of hyperspectral image processing systems.

1.2 Scientific Setting

1.2.1 The APEX Instrument

The topics in this thesis have arisen during the development of the ESA/Prodex-sponsored Airborne Prism EXperiment (APEX) pushbroom imaging spectrometer. A simplified overview of the data processing involved, highlighting the placement of work described in this thesis, is shown in Figure 1.1. The colored round boxes refer to steps addressed within specific chapters of this thesis. Real-time data acquisition (depicted in the left-most colored round box) involving the fusion of image data with corresponding temporal, positional, and environmental data

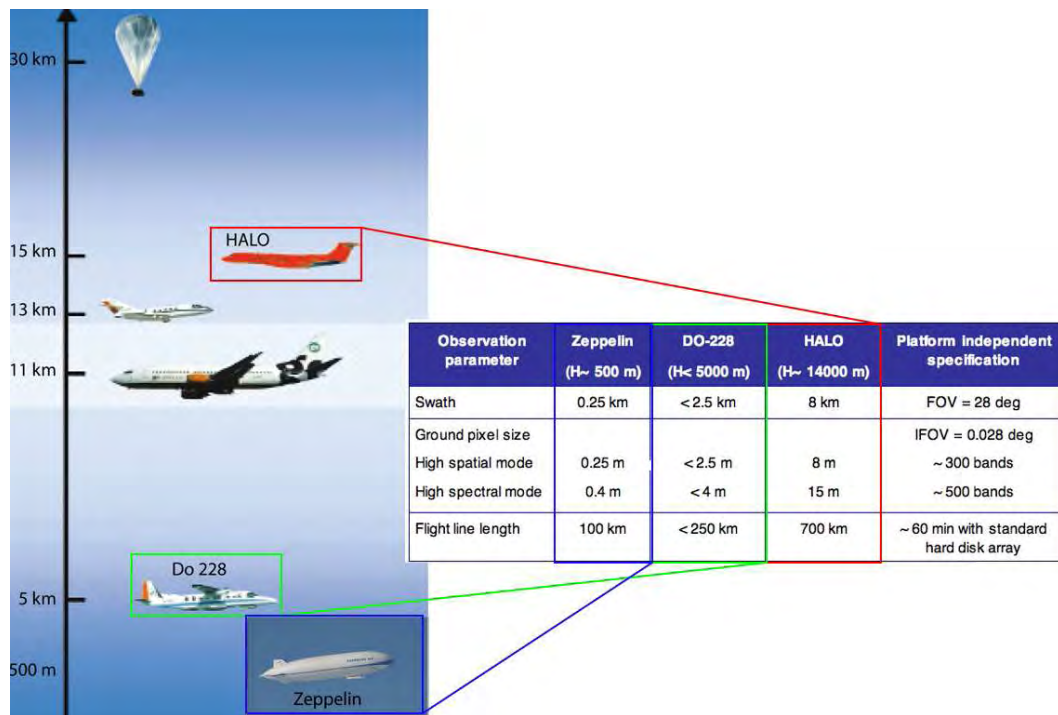


Figure 1.2: Selected specifications of the APEX instrument

from multiple sources using off-the-shelf hardware and software is likely a problem that has been solved many times before, but is difficult to find in the literature. Later chapters advocate distributed processing using parametric modeling toward advancing the case for scene-based calibration (depicted by the colored round box second to left in Figure 1.1) as a way to reduce down-time and labor costs incurred by the alternative of more frequent laboratory calibration. The feasibility of discernment of an instrument's Spectral Response Function (SRF) through inverse modeling is for the first time theoretically established given an instrument of state-of-the-art performance, then the concept is proven using existing instrument performances and more realistic SRF shapes. One chapter exhibits a case study comparing run-time performance and development issues in using a dedicated cluster vs a commodity grid for atmospheric modeling (depicted in the right-most colored round box) using the commonly-encountered workload of simulating a model over thousands of instances of varying parameters. The remaining chapters describe freely-distributable, open source tools for 1) an easy-to-use, cross-institutional computing grid and 2) a concept and prototype componentized framework intended to address some of the aforementioned deficiencies in prolific reusable, hyperspectral data processing software components (the remaining colored round box labeled "level 1 processing").

The APEX project started in 1997 with a feasibility study on the design of an imaging spectrometer [Itten et al., 1997] and led to a first performance definition [Schaepman et al., 1998], a subsequent design phase [Schaepman et al., 2000], prototyping for prediction of instrument performance [Schlöpfer et al., 2004a], and finally realization [Nieke et al., 2005]. First testing of the assembled instrument is scheduled for October 2007. Production acquisitions on the DO 228 airborne platform are planned from 2008 and the exploitation on the High Altitude Long Range Research Aircraft (HALO) is foreseen in 2010 [Nieke et al., 2007]. Selected specifications of the APEX instrument are shown in Figure 1.2.

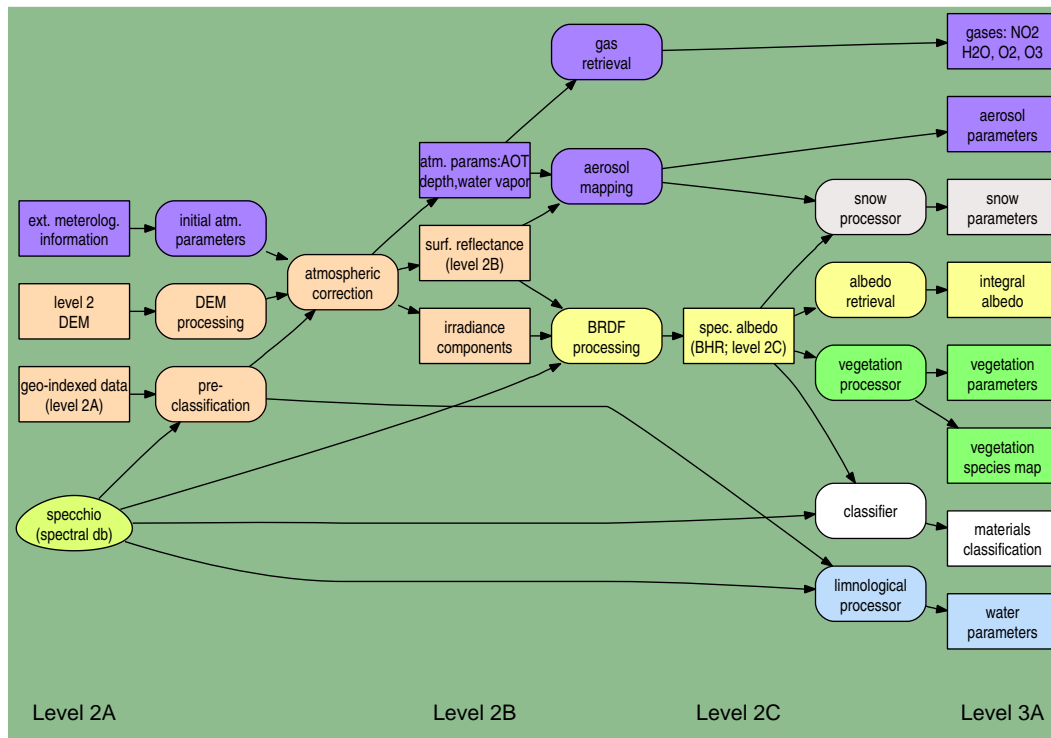


Figure 1.3: Overview of APEX Level 2/3 processing

The APEX instrument is designed as a dispersive pushbroom imaging spectrometer operating in the solar reflected wavelength range between 380 and 2500 nm. The spectral resolution is designed to be better than 10 nm in the SWIR and 5 nm in the VIS/NIR range of the spectrum. The total field of view (FOV) is on the order of $\pm 28^\circ$ recording 1000 pixels across track with a swath width of 2.5 – 8 km, depending on flight altitude and with a maximum of 511 spectral bands simultaneously [Nieke et al., 2005]. It is expected that individual flights will collect data on the order of hundreds of gigabytes that need to undergo data transformations using previously acquired as well as in-flight calibration data.

In the implementation phase, the data processing chain was well-specified in terms of the calibration and characterization pre-processing steps needed as well as the post-processing correction steps [Schäepman et al., 2002]. Later, during software prototyping, these conceptual plans were implemented and reworked, resulting in full descriptions and definitions of APEX data products [Schläpfer et al., 2004a], [Kaiser et al., 2004]. Consequently, a parallelization concept for this prototype was developed and prototyped on a small APEX processing cluster [Brazile et al., 2004]. This prototype is consistent with the recently updated APEX level 2/3 product workflow (see Figure 1.3), where hyperspectral application modules will be implemented for optimized interaction between the various processing modules. [Schläpfer et al., 2007].

The laboratory designated to handle APEX's dedicated full instrument calibration and characterization, known as the Calibration Home Base (CHB), is located at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. It will make use of that facility's 1.6 m integrating sphere and 7-ton granite optical bench, in addition to other equipment, for performing radiometric, spectral, and geometric calibration [Suhr et al., 2005].

For the upcoming exploitation phase, the APEX team has been organized into an APEX Sci-

ence Center (ASC) hosted at the Remote Sensing Laboratories (RSL) located at the University of Zurich, and an APEX Operations Center (AOC) hosted by the Flemish Institute for Technological Research (VITO) in Mol, Belgium.

1.2.2 Scene-based spectrometer calibration/characterization

For a spectrometer to be useful for making scientific measurements, it must be calibrated against a known standard and its error characteristics must be determined within a known range [Green, 1998]. These calibration and characterization tasks are usually performed during the instrument's downtime (e.g. off-season) by experts using resource intensive processes in a dedicated laboratory with specialized equipment [Schaepman and Dangel, 2000] and are also often simultaneously cross-checked either against other remote devices, or portable ground devices [Secker et al., 2001].

The idea eventually arose to retrieve and/or refine instrument characteristics by analyzing acquired (i.e., production) scene data as opposed to synthetic (i.e., non-production) laboratory data acquisitions [Barry et al., 2001]. This becomes increasingly more feasible through the development and maintenance of precise reference databases of known physical phenomena (such as atmospheric [Rothman et al., 2003] or solar [Thuillier et al., 2003] features); improved sophistication of engineering techniques for the juxtaposition of calibration references [Reuter and McCabe, 2002]; the decreasing effort needed to run scores of model simulations permuted over varying parameters [Brazile, 2007] for the purpose of inverse data fitting; and finally the sophistication of algorithms for retrieving various characteristics.

Among the first instances of scene-based hyperspectral instrument characterization was the use of a spectrum-matching technique for improving band center calibrations [Gao et al., 2002], that was later refined [Gao et al., 2004]. Similar techniques were performed for the spaceborne Medium Resolution Imaging Spectrometer (MERIS) based on an O₂ absorption feature [Ramon et al., 2003] and [Casadio and Colagrande, 2003]. Another method for detection of band center shift was later also independently developed [Guanter et al., 2005]. Shortly after Gao's earlier work, a similar but combined process that simultaneously detects band width changes as well as band center shifts was described [Neville et al., 2003] and detailed further [Neville et al., 2007]. These investigations inspired work described in this thesis for retrieval/refinement of spectral response function (SRF) shape in addition to the other two unknowns [Brazile et al., 2006], [Brazile et al., 2007]. Furthermore scene-based retrieval and refinement of spectral characteristics served as inspiration for a method for spatial characterization in the form of keystone aberration detection [Neville et al., 2004], which was subsequently formalized [Dell'Endice et al., 2007].

Currently, scene-based calibration/characterization methods cannot fully replace laboratory calibration, but they can refine and validate them as well as possibly increase instrument "up time" in between intensive full laboratory calibration/characterization campaigns. Continuing algorithm development and the decrease in cost of data processing imply that these trends will only increase over time.

1.2.3 Hyperspectral Data Processing

Software systems for visualization and interpretation of imaging spectroscopy data have been in development since the 1980's. Initially the SPectral Analysis Manager (SPAM) was able to perform primitive functions such as spectral arithmetic, clustering and matching [Mazer et al., 1988]. Later, the Spectral Image Processing System (SIPS) was developed specifically for viewing

analysis results from the Airborne Visible/Infrared Imaging Spectrometer and included functions for data formatting, correction to apparent reflectance, and various cosmetic processing [Kruse et al., 1993]. The Imaging Spectrometer Data Analysis System (ISDAS) included integrated tools in many categories including data pre-processing, data visualization and information extraction [Staenz et al., 1998]. ENVI was initially created by a five partner research spin-off to focus on visualization, processing and analysis using a platform with an integrated, high-level “Interactive Data Language”. More recently, the MATLAB hyperspectral image analysis toolbox has offered processing and information extraction software including various classification algorithms [Rosario-Torres et al., 2005]. The Basic ERS and Envisat (A)ATSR and Meris Toolbox (BEAM) is a Java-based software system that was commissioned primarily to handle ESA MERIS, (A)ATSR and ASAR data [Fomferra and Brockmann, 2005]. Readers desiring a good historical overview of various data processing toolkits are advised to read [Boardman et al., 2006].

All of these systems have some overlap in functionality and were developed independently (i.e. almost redundantly). They are implemented in different programming languages, hosted on different operating systems, and are only indirectly interoperable with each other – by way of support for mutually recognizable (and hopefully byte order neutral) storage file formats. It is not common practice to mix and match individual components of the various systems when implementing data novel data processing chains - even though this practice would be highly desirable if possible (e.g. ENVI’s spectral smoothing and MATLAB’s neural network toolbox). Most systems are not open source, so when the system is no longer officially supported (e.g. SPAM, SIPS), all effort is lost (as is the external imagery and processing configurations that relies on that system). Additionally, the general lack of source code visibility and open development process hinders oversight and quick defect turnaround.

1.3 Research Questions

During the development of the APEX imaging spectrometer, essential research questions arose in the areas of data acquisition, distributed processing, instrument characterization, and software componentization of hyperspectral processing algorithms. Of the many potential directions for study, the following questions were useful in targeting directed investigations.

1.3.1 Data Acquisition

Before beginning investigation into novel scientific applications, basic needs must first be covered: incremental improvement of the state-of-the-art of airborne and spaceborne imaging spectrometers. Each newly developed hyperspectral instrument aims to make the best use of its multiple state-of-the-art sensors. These probably offer one or more of: increased number of channels, greater dynamic range, reduced integration time and/or higher transmission rate. All of these imply pushing data acquisition throughput to the technical leading edge. While the capacity of solid state (optics and) electronics are continuously improving at a Moore’s Law rate [Moore, 1965], archival media (i.e. rotating platter) data rates are improving at a much lower rate. This means the data acquisition subsystems of new instruments will likely always remain challenging. If the data cannot be committed to stable storage in real time, one at least needs to record timestamped metadata along with the imagery and telemetry metadata to ensure that once everything does get committed to stable storage, it is accurate and synchronized. Additionally, given the overall budget of a new instrument, system designers would much rather allocate the largest portion of the budget to optics and sensors, rather than special purpose throughput

electronics. As much as possible, one would like to use commodity off-the-shelf components not only to support the hardware budget, but also the software development and maintenance budget.

For the data acquisition needs of a hyperspectral imager, what is a minimal set of special purpose hardware components required to allow high bandwidth, known-latency, data acquisition software to be written using open, commodity, software and hardware components? (Treated in Chapter 2)

1.3.2 Distributed Data Processing

In today's scientific research, there are an increasing number of sometimes even only auxiliary tasks to be performed that could benefit through the use of parallel or distributed processing. Perhaps during the design of an instrument, one has an optimization problem where a search through all permutations of the parameter space could produce an optimal answer. Or perhaps during calibration/characterization of a system, one has an inverse problem such as a recorded measurement filtered through a component with known parameterized characteristics but unknown values of those parameters. And suppose one also has an accurate mathematical model of the parameters. One might only need to run through permutations of those parameters' values to try to match the recorded measurement for the purpose of retrieving the unknown value of a component's modeled characteristic. Or perhaps the problem is more mundane. Maybe two groups would like to collaborate on a problem, but one group has intellectual property (IP) restricted software and the other group has IP restricted data and both groups would benefit if somehow the two could be brought together. What if it were legally possible to arrange for unencumbered input data to be moved to the site of the restricted software for processing and the results to be sent back. And at the other site, perhaps the unencumbered software can be sent to the location of the restricted data and the results be returned. Everyone could benefit while remaining within the legal constraints of the problem. Run time performance is not always the primary benefit of distributed data processing.

Unfortunately, although such problems exist in probably every scientific field, there are not many easy-to-use, generally applicable services or tools to support solving these problems. On the easy-to-use side, Apple Computer offers a novice-friendly Xgrid [Apple Xgrid, 2004] distributed application tool, but it runs only on Apple hardware and does not work easily through firewalls which are ubiquitous today, thus reducing usability by users who are not all located at the same institution. Conversely, there are several very powerful and flexible distributed computing toolkits and frameworks such as Globus-based systems [Foster and Kesselman, 1997], boinc-based systems [Anderson, 2004], or the MPI programming framework [MPI, 2005], but these require above-average access to facilities (e.g. web servers, firewall/network configuration) and/or competence in programming and/or system integration.

If non-professional programming investigators located at competing institutions wish to pool their individual "everyman" computing resources toward collaborating on a problem that can be solved by distributing a medium to large number of independent computations, what service or tool could provide this, assuming they have modest means at their disposal? And if there is a choice between cluster and grid availability, what trade-offs can be expected with respect to development effort, result reproducibility, total job turnaround, and resource utilization over time? (Treated in Chapters 3 and 4)

1.3.3 Instrument Characterization

As hyperspectral instrument performance increases and data processing and applications become more sophisticated, the relative importance of characterization/calibration/validation increases. The validity of particular algorithms or techniques may even require a guaranteed minimum degree of aggregate systemic error. Laboratory instruments and techniques are becoming increasingly sophisticated to meet this need, but they are also becoming increasingly resource intensive. If characterization/calibration were a one-time or seldom-required cost, one would be tempted to merely accept the situation and learn to live with it. However, it is a recurring need – instrument characteristics change under differing operating conditions and individual components systematically deteriorate over time. Fortunately, there are multiple emerging techniques for alleviating the problem using indirect methods. The first line of attack on the problem involves the addition of auxiliary instrument components used for the purpose of on-board calibration. The inclusion of special filters, mirrors, and/or samples of fully characterized rare earth, has become fairly standard practice in all but the least expensive instruments [Conel et al., 1988]. A second way to mitigate the problem is through the use of vicarious or cross-calibration: comparison of measurements of the same targets at the same time using multiple instruments. In both of these cases, fairly standard procedures have been developed, and improvements are typical minor refinements.

An additional, and not yet entirely exploited, attack vector for the problem of increasing instrument characterization/calibration precision and/or accuracy in a less resource-intensive way involves scene-based techniques. If there are physical characteristics of objects present in a given scene that are known to a high degree of precision and/or accuracy, these can be used as reference points for comparison against instrument-measured observations of those same characteristics.

On first thought, one might respond, “but one can’t always rely on serendipitous presence of highly spectrally homogeneous snow, desert, water, or agricultural targets in any given scene”. This is true, however, there are some characteristics that *are* guaranteed to be included in every scene: atmospheric perturbations caused by e.g. water vapor and known trace gases. Therefore, the HITRAN molecular spectroscopic database [Rothman et al., 2003] provides a commonly-used set of reference data for various scene-based characterization/calibration techniques. In particular it has been used to successfully refine and/or retrieve instrument unknowns such as per-band spectral band center shifts [Neville et al., 2003], [Gao et al., 2004].

The first characteristics to be so observed were spectral band center shifts and band width increases/decreases. Of the potentially other refinable characteristics, one wonders if the spectral response function (SRF) *shape* is also a refinable characteristic.

Given today’s hyperspectral instrument performance and the known precision and accuracy of the molecular spectroscopic database of atmospheric phenomena, is it possible to retrieve or refine per-band spectral response function (SRF) differences between two otherwise identical instruments across a range of typical application scenarios based solely on the (SRF-filtered) measurements made by those instruments? If so, is it additionally possible to simultaneously retrieve or refine additional spectral unknowns such as band center and/or bandwidth shifts and if so, what parameters (SNR, prominence of atmospheric feature, elevation, etc.) are likely to constrain this feasibility? (Treated in Chapters 5 and 6)

1.3.4 Componentization of Hyperspectral Processing Algorithms

The urge to solve new scientific problems as opposed to making refinements to existing solutions is high. Once a potential new solution is found, it is quickly prototyped, proven, and published. In some cases it is even used somewhere during the production data processing for a given system. Assuming there are many rapid prototypers, eager to prove something to the point of publishability, but then possibly no further, it is easy to see why there is so much redundant effort in the development of hyperspectral image processing systems. Perhaps many years ago the mathematical description of an algorithm or method was small enough that merely publishing the math was sufficient for releasing the idea to the world in a way that could be quickly verified, duplicated and/or used. Nowadays it would probably be difficult to find programs small enough to be entirely publishable as a figure in a journal article. This complexity has a cost. There are potentially many defects present (or put differently – there may be an unrealized latent potential for enhancements/improvements). Even worse, there is now less of a guarantee that the description of the code in the paper matches the actual code. Nobody intends to misinform or mislead, but misunderstandings can quickly arise in common situations. Consider the professor who writes a paper about an idea she described to one of her students, who was the implementer of the idea. There are several potential sources of (unintentional) error – the communication flow from professor to student, the communication back from the student to the professor after she has implemented it, and the possibility that the student has made “minor” improvements to the method, perhaps even leading to different charts/graphs that get published together with a description of the originally conceived technique/algorithm, with the student neglecting to inform the professor of “little tweaks”.

The goal of science should be longevity. It is unfortunately rare to find any particular processing algorithm that has remained unchanged, yet popular for, say, 30 years. Yet if it is truly science, one would expect the algorithm to be useful from the time of its discovery until the end of human endeavor.

Additionally, it appears that computational speedup has somewhat leveled off, with any further gains to be provided through distributed and/or parallel processing rather than faster single processors. It would be prudent to take this into account while considering a solution to the problem.

Given the nature of the scientific software development process, the increasing complexity of processing algorithms, and the hardware trend of speedups only through multi-core/multi-node computing, what can be done from a software architecture perspective to increase collaboration, reduce duplicated effort and maximize software longevity? (Treated in Chapter 7)

1.4 Structure of Thesis

This thesis is based primarily upon published peer-reviewed scientific communications. In general, the engineering related topics are addressed in published abstract-reviewed proceedings (Chapters 2 and 3), and the more theoretical and/or in-depth studies are addressed in journal articles (Chapters 4, 5, 6, and 7).

Chapter 2 presents a technical note describing **APEX data acquisition from an engineering perspective**. The expected data rate of the image stream from the optical sub-system slightly exceeds the maximum-sustained write throughput for the selected data acquisition hardware

and operating system. To ensure the needed throughput even in the worst-case, an improvement technique involving parallel memory access and double buffering is implemented allowing data to be written to multiple raw disk devices (i.e., no file system) residing on separate buses in parallel. Since the initial design, improvements in commodity hardware (disk media, bus, and peripheral interface throughput) as well as software (software-RAID interfaces provided by the operating system kernel) have made it possible to reduce the complexity needed to implement these methods. However, since the hardware specification was frozen at the older technology, only operating system-related software improvements were allowed, which maintains the usefulness of the described implementation.

Chapter 3 describes a **grid service software component** that was developed to enable the work in the following chapters. This component enables computing resources from commodity workstations at diverse geographic locations to be voluntarily employed toward working on collective problems. Initial cross-institutional system set-up is achieved in minutes rather than days. Not only is its configuration uncomplicated, but for some sets of problems (e.g. numerous small-to-medium sized jobs, rather than few very large jobs) run-time efficiency can be double that of popular grid alternatives such as Condor [Litzkow et al., 1988].

Chapter 4 is an **atmospheric modeling cluster vs grid computing case study** showing how the computational running time of a large look-up table (LUT) used in atmospheric correction was reduced from requiring nearly a month to only a few hours. This was done using both cluster computing and grid computing in order to compare the advantages and disadvantages in accuracy, run-time performance, and development complexity of each. The resulting LUT is used as the core database for the widely-used ATCOR family of atmospheric correction software, that is also used during the generation of APEX data products.

Chapter 5 examines the **feasibility of scene-based retrieval of a novel calibration parameter – per-channel Spectral Response Function (SRF) shape**. By establishing discernibility of theoretical SRFs over a range of varying target observation parameters, retrieval is shown to be theoretically feasible for high resolution instruments such as APEX. This knowledge serves the basis for defining a particular retrieval method as further described in Chapter 6.

Chapter 6 details the requirements for and investigates **initial steps toward a specific method for retrieving an instrument's per-band spectral response function** to support scene-based calibration. The method is based on multiple simulated instruments modeled using real instrument specifications and real spectral response function shapes. Cluster and grid computing are instrumental in keeping the run-time within the overall processing budget allocated for operational product generation.

Chapter 7 outlines a concept for employing the Unix pipeline command paradigm in implementing **a set of loosely coupled, componentized hyperspectral image data processing utilities** for supporting the common but diverse needs of researchers in hyperspectral image data processing. Specific examples including APEX level 1 processing and Hyperion level 2 processing are described. The key to the concept is defining framework durable enough for its usefulness to outlast any language used in its initial implementation. Several beneficial properties of the envisioned system are explored, and prototypes of many difficult issues are described to help ensure that the concept is feasible and beneficial.

Finally, **Chapter 8** takes a high-level view at what earth observation imaging spectroscopy has achieved since its birth in the 1970's, **summarizes the step-wise contributions that have been made by publications in this thesis**, including what impact they are likely to have in the near to medium term and finally suggests how these contributions could and should be extended within the next 5-10 years.

References

- [Anderson, 2004] Anderson, D. P. (2004). The berkeley open infrastructure for network computing. [Online; accessed Jul-2004]. <http://boinc.berkeley.edu/>.
- [Apple Xgrid, 2004] Apple Xgrid (2004). Xgrid. [Online; accessed Jul-2004]. <http://www.apple.com/acg/xgrid>.
- [Babey and Anger, 1989] Babey, S. K. and Anger, C. D. (1989). A compact airborne spectrographic imager (CASI). In Proc. IGARSS 1989, volume II, pages 1028–1031, Vancouver, Canada.
- [Barry et al., 2001] Barry, P., Shepanski, J., and Segal, C. (2001). HYPERION on-orbit validation of spectral calibration using atmospheric lines and an on-board system. In Descour, M. and Shen, S., editors, SPIE Imaging Spectrometry VII, volume Vol. 4480, pages 231–241, San Diego.
- [Basedow et al., 1995] Basedow, R. W., Carmer, D. C., and Anderson, M. E. (1995). HYDICE system, implementation and performance. In Descour, M. R., Mooney, J. M., Perry, D. L., and Illing, L. R., editors, SPIE Imaging Spectrometry, volume 2480, pages 258–267.
- [Berk et al., 1998] Berk, A., Bernstein, L., Anderson, G., Acharya, P., Robertson, D., Chetwynd, J., and Adler-Golden, S. (1998). MODTRAN cloud and multiple scattering upgrades with applications to AVIRIS. Remote Sensing of Environment, 65(3):367–375.
- [Boardman, 1998] Boardman, J. W. (1998). Post-ATREM polishing of AVIRIS apparent reflectance data using EFFORT: a lesson in accuracy versus precision. Summaries of the Seventh JPL Airborne Earth Science Workshop, JPL Pub. 97-21:53.
- [Boardman et al., 2006] Boardman, J. W., Biehl, L. L., Kruse, F. A., Clark, R. N., Mazer, A. S., Torson, J., and Staenz, K. (2006). Development and implementation of software systems for imaging spectroscopy. In Proc. IGARSS 2006, pages 1969–1973, Denver, CO. IEEE.
- [Brazile, 2007] Brazile, J. (2007). Low Fat Grid: A RESTful Grid Service for Non-Programmers. In Jazoon '07 – The International Conference on Java Technology, Zurich.
- [Brazile et al., 2006] Brazile, J., Neville, R. A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K. I. (2006). Scene-based spectral response function shape discernibility for the APEX imaging spectrometer. IEEE Geoscience and Remote Sensing Letters, 3:414–418.
- [Brazile et al., 2007] Brazile, J., Neville, R. A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K. I. (2007). Toward scene-based retrieval of spectral response functions for hyperspectral imagers using fraunhofer features. Canadian Journal of Remote Sensing. in press.

- [Brazile et al., 2004] Brazile, J., Schläpfer, D., Kaiser, J., Schaepman, M. E., and Itten, K. I. (2004). Cluster versus grid for large-volume hyperspectral image preprocessing. In SPIE Atmospheric and Environmental Remote Sensing Data Processing and Utilization: an End-to-End System Perspective, volume 5548, pages 48–58.
- [Casadio and Colagrande, 2003] Casadio, S. and Colagrande, P. (2003). Meris O₂ calibration using sciamachy measurements. In Proceedings MERIS User Workshop, Frascati, Italy. ESA.
- [Chang, 2003] Chang, C.-I. (2003). Hyperspectral Imaging: Techniques for Spectral Detection and Classification. Springer.
- [Chang et al., 1993] Chang, S.-H., Westfield, M. J., Lehmann, F., Oertel, D., and Richter, R. (1993). A 79 channel airborne imaging spectrometer. In Vane, G., editor, SPIE Imaging Spectrometry of the Terrestrial Environment, volume 1937, pages 164–172.
- [Chiu and Collins, 1978] Chiu, H.-Y. and Collins, W. (1978). A spectroradiometer for airborne remote sensing. Photogrammetric Engineering and Remote Sensing, 44:507–517.
- [Cocks et al., 1998] Cocks, T., Jenssen, R., Stewart, A., Wilson, I., and Shields, T. (1998). The HyMap(TM) airborne hyperspectral sensor: The system, calibration and performance. In 1st EARSeL Workshop on Imaging Spectroscopy, pages 37–42, Zurich, Switzerland.
- [Collins et al., 1981] Collins, W., Chang, S.-H., Kuo, J. T., Douma, M., Marshall, S., and Murphy, P. (1981). High spectral resolution airborne spectrometry. In SPIE Imaging spectroscopy, pages 22–28, Los Angeles, CA.
- [Conel et al., 1988] Conel, J. E., Green, R. O., Alley, R. E., Bruegge, C. J., Carrere, V., Margolis, J. S., Vane, G., Chrien, T. G., Slater, P., Biggar, S., Teillet, P., Jackson, R., and Moran, M. (1988). In-flight radiometric calibration of the airborne visible/infrared imaging spectrometer (AVIRIS). In SPIE Recent advances in sensors, radiometry and data processing for remote sensing, volume 924, pages 179–195.
- [Dangel et al., 2005] Dangel, S., Verstraete, M. M., Schopfer, J., Kneubühler, M., Schaepman, M., and Itten, K. I. (2005). Toward a direct comparison of field and laboratory goniometer measurements. IEEE Transactions on Geoscience and Remote Sensing, 43(11):2666–2675.
- [Dell'Endice et al., 2007] Dell'Endice, F., Nieke, J., Schläpfer, D., and Itten, K. I. (2007). A scene based method for spatial misregistration detection in hyperspectral imagery. Applied Optics. in press.
- [Dorigo et al., 2006] Dorigo, W., Zurita-Milla, R., de Wit, A., Brazile, J., Singh, R., and Schaepman, M. (2006). A review on reflective remote sensing and data assimilation techniques for enhanced agroecosystem modeling. Journal of Applied Earth Observation and Geoinformation. in press.
- [Fomferra and Brockmann, 2005] Fomferra, N. and Brockmann, C. (2005). Beam – the ENVISAT MERIS and AATSR toolbox. In Proceedings MERIS User Workshop, Frascati, Italy. ESA.
- [Foster and Kesselman, 1997] Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing, 11(2):115–128.
- [Gao et al., 2002] Gao, B.-C., Montes, M. J., and Davis, C. O. (2002). A curve-fitting technique to improve wavelength calibrations of imaging spectrometer data. In Proceedings of the 11th JPL Airborne Earth Science Workshop, volume 03-4, pages 99–105.

- [Gao et al., 2004] Gao, B.-C., Montes, M. J., and Davis, C. O. (2004). Refinement of wavelength calibrations of hyperspectral imaging data using a spectrum-matching technique. *Remote Sensing of Environment*, 90:424–433.
- [Gitelson et al., 2003] Gitelson, A. A., Verma, S. B., Viña, A., Rundquist, D. C., Keydan, G., Leavitt, B., Arkebauer, T. J., Burba, G. G., and Suyker, A. E. (2003). Novel technique for remote estimation of CO_2 flux in maize. *Geophysical Research Letters*, 30(9):39/1–39/4.
- [Goetz et al., 1982] Goetz, A. F., Rowan, L. C., and Kingston, M. J. (1982). Mineral identification from orbit: Initial results from the shuttle multispectral infrared radiometer. *Science*, 218(4576):1020–1024.
- [Goetz et al., 1977] Goetz, A. F. H., Graham, R. A., and Ozawa, T. (1977). Portable reflectance spectrometer. Technical report, NASA.
- [Goetz et al., 1985] Goetz, A. F. H., Vane, G., Solomon, J. E., and Rock, B. N. (1985). Imaging spectrometry for earth remote sensing. *Science*, 228(4704):1147–1153.
- [Green, 1998] Green, R. O. (1998). Spectral calibration requirement for Earth-looking imaging spectrometers in the solar-reflected spectrum. *Applied Optics*, 37(4):683–690.
- [Guanter et al., 2005] Guanter, L., Richter, R., and Moreno, J. (2005). Spectral calibration of hyperspectral imagery using atmospheric absorption features. *Applied Optics*, 45(10):2360–2370.
- [Itten et al., 1997] Itten, K. I., Schaepman, M., De Vos, L., Hermans, L., Schläepfer, H., and Droz, F. (1997). APEX - airborne prism experiment: A new concept for an airborne imaging spectrometer. In *3rd Intl. Airborne Remote Sensing Conference and Exhibition*, volume 1, pages 181–188.
- [Kaiser et al., 2004] Kaiser, J. W., Schläpfer, D., Brazile, J., Strobl, P., Schaepman, M. E., and Itten, K. I. (2004). Assimilation of heterogeneous calibration measurements for the APEX spectrometer. In *SPIE Sensors, Systems, and Next Generation Satellites VII*, volume 5234, pages 211–220, Barcelona, Spain.
- [Katz and Shapiro, 1994] Katz, M. L. and Shapiro, C. (1994). Systems competition and network effects. *Journal of Economic Perspectives*, 8(2):93–115.
- [Kruse et al., 1993] Kruse, F. A., Lefkoff, A. B., Boardman, J. W., Heidebrecht, K. B., Shapiro, A. T., Barloon, P. J., and Goetz, A. F. H. (1993). The spectral image processing system (sips). *Remote Sensing of Environment*, 44(2-3):145–163.
- [Lee et al., 2000] Lee, C., Theiss, H., Bethel, J., and Mikhail, E. (2000). Rigorous mathematical modeling of airborne pushbroom imaging systems. *PE & RS*, 66(4):385–392.
- [Litzkow et al., 1988] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor—a hunter of idle workstations. In *The 8th International Conference on Distributed Computing Systems*, pages 104–111.
- [Macenka and Chrisp, 1987] Macenka, S. A. and Chrisp, M. P. (1987). Airborne visible/infrared imaging spectrometer (AVIRIS). In *SPIE Imaging Spectroscopy II*, volume 834, pages 32–43.
- [Mazer et al., 1988] Mazer, A. S., Martin, M., Lee, M., and Solomon, J. E. (1988). Image processing software for imaging spectrometry data analysis. *Remote Sensing of Environment*, 24(1):201–211.

- [Moore, 1965] Moore, G. E. (1965). Cramming more components onto integrated circuits. Electronics, 38(8):114–117.
- [MPI, 2005] MPI (2005). The MPI (Message Passing Interface) Forum. [Online; accessed Sep-2005]. <http://www.mpi-forum.org/>.
- [Neville et al., 2003] Neville, R. A., Sun, L., and Staenz, K. (2003). Detection of spectral line curvature in imaging spectrometer data. In Shen, S. and Lewis, P., editors, SPIE Algorithms and Technologies for Multispectral Hyperspectral and Ultraspectral Imagery IX, volume 5093, pages 144–154, Orlando, FL, USA.
- [Neville et al., 2004] Neville, R. A., Sun, L., and Staenz, K. (2004). Detection of keystone in imaging spectrometer data. In Shen, S. and Lewis, P., editors, SPIE Algorithms and Technologies for Multispectral Hyperspectral and Ultraspectral Imagery X, volume 5425, pages 208–217, Orlando, FL, USA.
- [Neville et al., 2007] Neville, R. A., Sun, L., and Staenz, K. (2007). Spectral calibration of imaging spectrometers by atmospheric absorption feature matching. Canadian Journal of Remote Sensing, in press.
- [Nieke et al., 2005] Nieke, J., Itten, K., Debruyn, W., Kaiser, J., Schläpfer, D., Brazile, J., Meuleman, K., Kempeneers, P., Neukom, A., Schilliger, T., De Vos, L., Piesbergen, J., Gege, P., Suhr, B., Schaepman, M., Gavira, J., Ulbrich, G., and Meynart, R. (2005). The airborne imaging spectrometer APEX: From concept to realization. In Zagojowski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 67–74, Warsaw, Poland.
- [Nieke et al., 2007] Nieke, J., Odermatt, D., Itten, K. I., Mauser, W., Oppelt, N., Ruhtz, T., Preusker, R., Fischer, J., Vohland, M., Hill, J., Kaufmann, H., Gege, T., Meuleman, K., Holzwarth, S., Müller, A., Puigdefabregas, J., Vellejo, R., Morales, J., and Ziereis, H. (2007). EO-HALO, An Earth-observation mission for regional studies in Europe. In Proc. EARSeL 5th Workshop on Imaging Spectroscopy.
- [Plaza et al., 2007] Plaza, A., Benediktsson, J. A., Boardman, J., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., Marconcini, M., Tilton, J. C., and Trianni, G. (2007). Recent advances in techniques for hyperspectral image processing. Remote Sensing of Environment. in review.
- [Plaza et al., 2002] Plaza, A., Martínez, P., Pérez, R., and Plaza, J. (2002). Spatial/spectral endmember extraction by multidimensional morphological operations. IEEE Transactions on Geoscience and Remote Sensing, 40(9):2025–2041.
- [Plaza et al., 2006] Plaza, A., Valencia, D., Plaza, J., and Martinez, P. (2006). Commodity cluster-based parallel processing of hyperspectral imagery. Journal of Parallel and Distributed Computing, 66(3):345–358.
- [Ramon et al., 2003] Ramon, D., Santer, R., and Dubuisson, P. (2003). MERIS in-flight spectral calibration in O₂ absorption using surface pressure retrieval. In Huang, H.-L., Lu, D., and Sasano, Y., editors, SPIE Optical Remote Sensing of the Atmosphere and Clouds III, volume 4891, pages 505–514.
- [Reuter and McCabe, 2002] Reuter, D. and McCabe, G. (2002). LEISA/atmopheric corrector (LAC) validation report. Technical report, NASA/GSFC.
- [Richter, 1996] Richter, R. (1996). A spatially adaptive fast atmospheric correction algorithm. International Journal of Remote Sensing, 17(6):1201–1214.

- [Rosario-Torres et al., 2005] Rosario-Torres, S., Arzuaga-Cruz, E., Velez-Reyes, M., and Jimenez-Rodriguez, L. O. (2005). An update on the MATLAB hyperspectral image analysis toolbox. In SPIE Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XI, volume 5806, pages 743–752.
- [Rothman et al., 2003] Rothman, L., Barbe, A., Benner, D. C., Brown, L., Camy-Peyret, C., Carleer, M., Chance, K., Clerbaux, C., Dana, V., Devi, V., Fayt, A., Flaud, J.-M., Gamache, R., Goldman, A., Jacquemart, D., Jucks, K., Lafferty, W., Mandin, J.-Y., Massie, S., Nemtchinov, V., Newnham, D., Perrin, A., Rinsland, C., Schroeder, J., Smith, K., Smith, M., Tang, K., Toth, R., Auwera, J. V., Varanasi, P., and Yoshino, K. (2003). The HITRAN molecular spectroscopic database: edition of 2000 including updates through 2001. Journal of Quantitative Spectroscopy and Radiative Transfer, 82:5–44.
- [Schaepman et al., 1998] Schaepman, M., De Vos, L., and Itten, K. (1998). APEX - airborne PRISM experiment: hyperspectral radiometric performance analysis for the simulation of the future ESA land surface processes earth explorer mission. In Shen, S. and Descour, M., editors, SPIE Imaging Spectrometry IV, volume 3438, pages 253–262.
- [Schaepman et al., 2002] Schaepman, M., Schläpfer, D., Brazile, J., and Bojinski, S. (2002). Processing of large-volume airborne imaging spectrometer data: the APEX approach. In SPIE Imaging Spectrometry VIII, volume 4816, pages 72–79, Bellingham, Washington, USA.
- [Schaepman et al., 2000] Schaepman, M., Schläpfer, D., and Itten, K. (2000). APEX - a new pushbroom imaging spectrometer for imaging spectroscopy applications: Current design and status. In Proc. IGARSS 2000, volume VII, pages 828–830, Hawaii.
- [Schaepman, 2007] Schaepman, M. E. (2007). Spectrodirectional remote sensing: from pixels to processes. International Journal of Applied Earth Observation and Geoinformation, 9(2):204–223.
- [Schaepman and Dangel, 2000] Schaepman, M. E. and Dangel, S. (2000). Solid laboratory calibration of a nonimaging spectroradiometer. Applied Optics, 39:3754–3764.
- [Schläpfer et al., 2004a] Schläpfer, D., Kaiser, J. W., Brazile, J., Schaepman, M. E., and Itten, K. I. (2004a). Calibration concept for potential optical aberrations of the APEX pushbroom imaging spectrometer. In SPIE Sensors, Systems, and Next Generation Satellites VII, volume 5234, pages 221–231.
- [Schläpfer et al., 2004b] Schläpfer, D., Kaiser, J. W., Nieke, J., Brazile, J., and Itten, K. I. (2004b). Modeling and correcting spatial non-uniformity of the APEX pushbroom imaging spectrometer. In 13th Annual JPL Airborne Earth Science Workshop, page 11.
- [Schläpfer et al., 2007] Schläpfer, D., Nieke, J., Dell’Endice, F., Hüni, A., Biesmans, J., Meuleman, K., and Itten, K. I. (2007). Optimized workflow for APEX level 2/3 processing. In Proc. EARSeL 5th Workshop on Imaging Spectroscopy.
- [Schläpfer and Richter, 2002] Schläpfer, D. and Richter, R. (2002). Geo-atmospheric processing of airborne imaging spectrometry data part 1: Parametric orthorectification. International Journal of Remote Sensing, 23(13):2609–2630.
- [Secker et al., 2001] Secker, J., Staenz, K., Gauthier, R., and Budkewitsch, P. (2001). Vicarious calibration of airborne hyperspectral sensors in operational environments. Remote Sens. Environ., 76:81–92.

- [Staenz et al., 1998] Staenz, K., Szeredi, T., and Schwarz, J. (1998). ISDAS - a system for processing/analyzing hyperspectral data. Canadian Journal of Remote Sensing, Vol. 24, No. 2:99–113.
- [Suhr et al., 2005] Suhr, B., Gege, P., Nieke, J., Itten, K., and Ulbrich, G. (2005). Calibration facility for airborne imaging spectrometers. In SPIE Sensors, Systems, and Next-Generation Satellites IX, volume 5978, Bruges, Belgium.
- [Thuillier et al., 2003] Thuillier, G., Hersé, M., Labs, D., Foujols, T., Peetermans, W., Gillotay, D., Simon, P. C., and Mandel, H. (2003). The solar spectral irradiance from 200 to 2400 nm as measured by the SOLSPEC spectrometer from the Atlas and Eureca missions. Solar Physics, 214:1–22.
- [van der Meer and de Jong, 2002] van der Meer, F. and de Jong, S. (2002). Imaging Spectroscopy: Basic Principles and Prospective Applications (Remote Sensing and Digital Image Processing). Springer.
- [Vane et al., 1983] Vane, G., Goetz, A. F. H., and Wellman, J. B. (1983). Airborne imaging spectrometer: A new tool for remote sensing. In Proc. IGARSS 1983. IEEE Cat. No. 83CH1837-4.

Chapter 2

A Software Architecture for In-flight Acquisition and Offline Scientific Post-Processing of Large Volume Hyperspectral Data

Brazile, J., Kohler, P., and Hefti, S., A Software Architecture for In-flight Acquisition and Offline Scientific Post-Processing of Large Volume Hyperspectral Data, Proceedings of the 10th Tcl/Tk Conference (Tcl/2003): July 29–Aug 2, 2003, Ann Arbor, Michigan, USA, Noumena, 2003, [CD-ROM].

Reprinted with Permission.

Abstract

The European Space Agency (ESA) is sponsoring a joint Swiss/Belgian/German initiative to design, build, and deploy an airborne dual prism dispersion pushbroom imaging spectrometer known as APEX (Airborne Prism EXperiment) to support earth observation applications at a local and regional scale. APEX has been designed to acquire 1000 pixels across track (covering 2.5-5 km, depending on flight altitude) with a maximum of 300 spectral bands simultaneously in the solar reflected wavelength range between 400 and 2500 nm [Schaepman et al., 2002]. This coverage of the Visible/Near Infrared (VNIR) and Shortwave Infrared (SWIR) spectrum in addition to a rigorous pre-flight and in-flight calibration and characterization process enable the resulting data to be used in a wide variety of applications including snow, vegetation, water, mineral, and atmospheric analysis and monitoring [Schaepman et al., 2003].

We discuss the design and ongoing implementation of the software architectures for both in-flight data acquisition and offline level 1 scientific data processing which are based in large part on the extendible and embeddable features of the Tcl scripting language to allow for rapid prototyping, hardware simulation and control, automated testing, and modularization and integration of proprietary software components as well as foreign domain-specific languages.

Specified Parameter	Value
Field of View (FOV)	$\pm 14^\circ$
Instantaneous Field of View (IFOV)	0.48 mrad
Flight Altitude	4,000 - 10,000 m.a.s.l.
Spectral channels	VNIR: ≈ 140 SWIR: ≈ 145
Spectral Range	400 - 2500 nm
Spectral Sampling Interval	400 - 1050 nm: < 5 nm 1050 - 2500 nm: < 10 nm
Spectral Sampling Width	$< 1.5 \times$ spectral sampling interval
Scanning Mechanism	Pushbroom
Storage Capacity On Board	> 300 GByte
Dynamic Range	12 . . . 16 bit
Positional Knowledge	20% of the ground sampling distance
Attitude Knowledge	20% of IFOV
Navigation system, flight line repeatability	$\pm 5\%$ of FOV

Table 2.1: Selected APEX Specifications

We additionally describe design trade-offs and architecture modifications that were made to better fit this programming model (event driven vs. threaded programming, “fat” calls vs. “thin” calls, etc.) in order to keep as much program logic as possible at a simpler higher level. In cases that require strict performance requirements, we describe how we built prototype architectural components and in some cases compared them with versions in C to help determine feasibility and ensure that memory and processing resource budgets could be met.

2.1 Introduction

The planning and specifications for the APEX instrument started in 1997 with a feasibility study [Itten et al., 1997] and proceeded through a scientific performance definition and an industrial design phase. The current construction phase of the instrument began in 2002 and is planned to be final in early 2005, when the first end-user flight campaigns are scheduled.

The core of the spectrometer consists of a beam splitter separating incoming light into the VNIR (380-1000nm) and SWIR (930-2500 nm) wavelength ranges where the beams are spatially and spectrally re-imaged on independent, co-registered detector arrays. The detectors both resolve 1000 spatial pixels across track and more than 150 spectral rows each, which are then summarized via reprogrammable on-chip binning into a maximum of 300 spectral rows for both detectors. A subset of the relevant specifications is shown in Table 2.1.

The data processing requirements of the APEX instrument can be logically separated into two somewhat independent components – on-board data acquisition, and offline scientific post-processing. The first section of this paper addresses the architecture requirements and subsequent design and implementation used for in-flight data acquisition, which mainly concerns itself with collection and integration of data from various sources and arranging for it to be stored in a safe and timely manner.

This is followed by a section on the requirements of scientific post processing which mainly involves how the collected raw data and previously acquired instrument characteristics can be correlated and calibrated into well-defined scientific units (i.e. at-sensor radiance in SI units, traceable to a certified standard (e.g. NIST, NPL)).

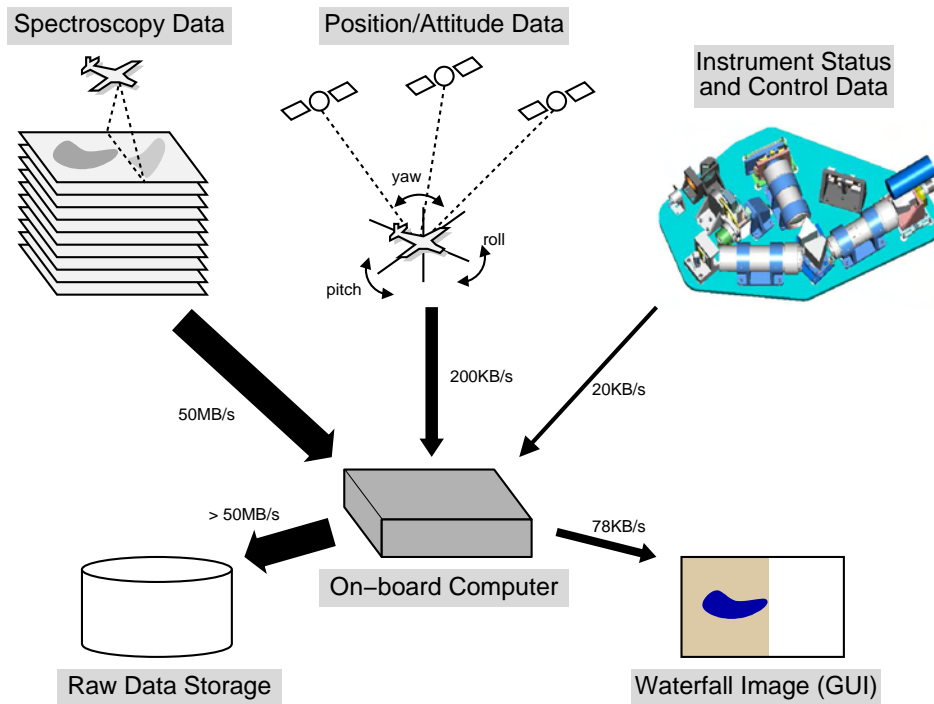


Figure 2.1: Data Acquisition Requirements

Then follows a section on the development process and how it has also influenced the design and implementation of the APEX system and application software. The final section summarizes and concludes with the current outlook.

2.2 In-Flight Data Acquisition

APEX in-flight data acquisition is somewhat challenging from a data processing point of view, as the collection of raw pixel data from the optical unit needs to be simultaneously correlated with high precision data coming from separate timing, positioning, and inertial data collection instruments and written in real time to on-board disk storage. Additionally, in-flight instrument calibration is performed between flight strips, requiring control of supporting hardware such as the calibration lamp, filter wheel, and mirror which are all to be centrally and automatically directed and monitored by the operator via an on-board computer. These data flow and bandwidth requirements are illustrated in Figure 2.1.

Initial system analysis indicated that there were 3 critical engineering problems that needed to be addressed to ensure successful data takes:

Data Throughput Optical data flows into the system at large volume and needs to be merged in memory by the processor with incoming “housekeeping” data flowing into the system from other I/O channels. This merged data needs to be written to disk in a safe and timely manner - a rate which exceeds single disk sustained write data rates.

Device Interrupt Latency Although incoming “housekeeping” data is not high volume, it does arrive at high frequency via a serial interface that is typically interrupt driven. The system can never be so busy with other tasks as to miss servicing one of these interrupts before the next interrupt arrives, otherwise data is lost.

Time Synchronization There are multiple independent devices (optics, GPS, orientation, etc.) taking data samples that need to be correlated and rectified with the notion of a common clock.

2.2.1 Bandwidth/Throughput

The data path between the optical unit and the on-board computer is the most difficult path to analyze/predict since it depends on many different factors. The following items have to be considered:

- Twice the optical data rate is needed on the PCI bus since the data is first transferred into on-board memory and then again from memory to the disk controller.
- In order to achieve PCI transfer rates close to the theoretical maximum, DMA (direct memory access) and long data bursts have to be implemented. This allows devices to read and/or write directly to on-board memory without needing control of the on-board CPU - allowing I/O operations to “overlap” with computation or with one another.
- DMA transactions on a PCI bus can be interrupted by I/O and event requests. To achieve maximum throughput, such interruptions have to be minimized e.g. by limiting the number of PCI devices on the same bus and/or by limiting the rate at which interruptions occur.
- A CPU board has a certain maximum I/O capacity which depends on the CPU, the I/O chipset and the type and speed of the main memory. All running applications as well as the OS (operating system) itself already use a part of that I/O capacity. If the OS and/or the applications generate heavy I/O or memory traffic, the throughput of the PCI subsystem might be affected.

In order to address the above items, a CPU board was chosen which includes dual processors, up to 4GB DDR RAM, two integrated ethernet interfaces, and - most critically - three independent PCI buses. This allows us to place the optic data PCI board and the PCI disk controller board each on a bus by themselves - using the third bus for all other peripheral I/O. The PCI hub of this board’s I/O chipset would theoretically allow 500MB/s bandwidth for each of 2 PCI buses during critical data transfer, which should satisfy the 65MB/s per device requirement with capacity to spare.

Next, we looked at the sustained write performance of our disk drives. It was determined that in order to achieve the target data rate, the video data stream needed to be split up and written to multiple drives in parallel on the raw devices (i.e. no filesystem).

Armed with this information, we devised a simple experiment to test data throughput along the complete data path – two small C programs (\approx 300 lines total) implementing a typical producer/consumer scenario. The producer DMA transferred data originating from an evaluation board (in place of the not yet completed optical board) on the first PCI bus to CPU shared memory. The consumer program consisted of multiple writer threads that concurrently read from the shared main memory buffer, then wrote the data through the disk controller on the second PCI bus to multiple drives in parallel. The CPU shared memory was partitioned into multiple semaphore protected buffers to allow overlapping (i.e. asynchronous) I/O.

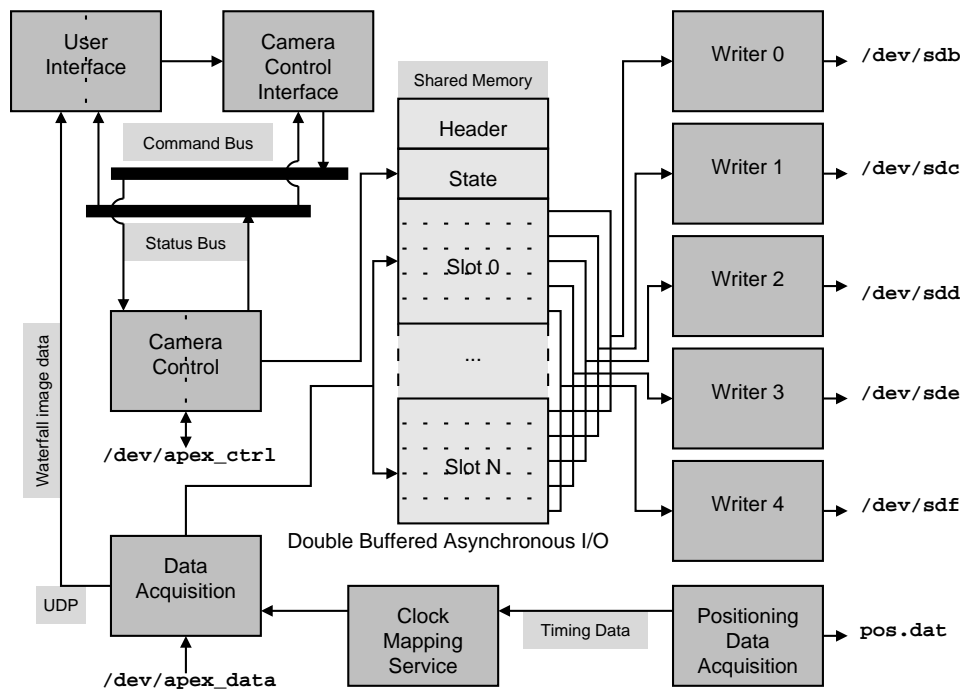


Figure 2.2: Subset of Data Acquisition Architecture

This test was run with 5 writers to 5 disks over the entire capacity of the disks, and the throughput was measured every 5 frames. We were surprised to see such a large variation in performance depending on which area of the disk was being written to. However, the maximum performance measured was 115.1 MB/s and the minimum was 85.7 MB/s – comfortably above our target of 50 MB/s. Even better, the CPU load during the test showed 65% idle time and 35% system time (and 0% user time – no other applications were running).

When this experiment proved successful, we proceeded to implement this concept as a Tcl-based prototype for the system - using the same asynchronous I/O strategy based on shared memory and semaphore (this time, through the `svipc.so` [Kelsey, 2003] Tcl extension). We kept the single producer process (labeled *Data Acquisition* in Figure 2.2) but used multiple processes for the N writers, instead of a single multi-threaded process. Since the N writers are created once and long-lived, there is no performance penalty in making this simplification. To complete the initial prototype, we needed only to develop our own Tcl extension (`veu.so`) for accessing the optical interface.

A final non-critical data throughput issue involved a component of the GUI. While the instrument is acquiring image data, it is required to simultaneously deliver what is known as a *waterfall image* to the operator console. This mechanism should select 3 of the 300 incoming spectral data channels and re-sample them to 8 bits per channel to represent a false color RGB (Red, Green, Blue) composite image. This allows the operator to use visual cues to verify that the mission is correctly following the intended flight line. Since this is a non-critical feature, it was determined that: (1) we can choose to only display every 1 of N lines, if desired and (2) we are allowed to use the unreliable UDP protocol (through the `udp.so` [Miller et al., 2003] Tcl extension) to transmit waterfall image data to the client since we don't care if image data is lost

before it is read by the operator's browser.

We initially saw some performance degradation with the Tcl version, but were later satisfied after modifying Kelsey's memory-to-disk write routine (`shmwrite`) to do a single large write instead of multiple 4K writes.

2.2.2 Device Interrupt Latency

A second critical engineering problem to be addressed was the servicing of high rate hardware device interrupts. The particular problem was that we expected high rate *housekeeping* data (25 times per second) to arrive over an RS-422 serial interface – a device typically serviced via an interrupt mechanism. There were two reasons to be wary of this issue. The first is mentioned above – we expected to make use of DMA and large burst I/O to fulfill our bandwidth/data throughput requirements and servicing device interrupts at a high rate is detrimental to this goal. The second reason to be concerned was our stated goal of trying as much as possible to keep all implementation in high level scripting code. One well-known scripting rule of thumb is to avoid scripted code in inner loops and critical performance sections - these things are better implemented in lower level code appearing as higher level primitives (i.e. "fat calls") at the scripting level.

While the software architects were considering how this tricky situation could be resolved – maybe by experimenting with a complicated polling device driver – an elegant solution was proposed by one of the hardware engineers. Apparently they had enough FPGA processing and memory budget remaining from the custom optical interface card that they could implement an RS-422 interface with a large on-card buffer and a programmable interrupt trigger. Even more importantly, the high rate housekeeping data could be rerouted through the optical interface masquerading as an additional fake spectral channel. This is beneficial not only for data latency and throughput reasons but also because it associates housekeeping meta data directly with the frame to which it belongs thereby reducing part of the clock synchronization problem. The only remaining data coming through the RS-422 to generate interrupts is now low rate control status i.e. command responses resulting from state change requests sent by the on-board computer to the instrument.

2.2.3 Time Synchronization

The third critical engineering problem to be addressed was how time synchronization should be handled between the various independent data delivery components. One of the synchronization problems was addressed above by implementing an intelligent RS-422 port. The two remaining timing-related components that need to be synchronized are the now unified optical data and the position and orientation data. In order to obtain meaningful and reproducible data, the spectral information and the position information have to be synchronized (and also serve as a necessary pre-condition for automated ortho-rectification during scientific post-processing).

In the optical component, the camera sends a line sync event to the optical interface card at the start of every image line acquired. It was determined that the best way to obtain low latency high resolution timing data was to tag the data directly in the optical interface card as it arrives from the camera (rather than via the on-board computer). This allows tagging of line sync events within a few nanoseconds. The implementation of this timer is kept simple by using a 64 bit free running counter driven by a quartz-stabilized oscillator. This clock tag is then transferred to the host computer along with the optical data.

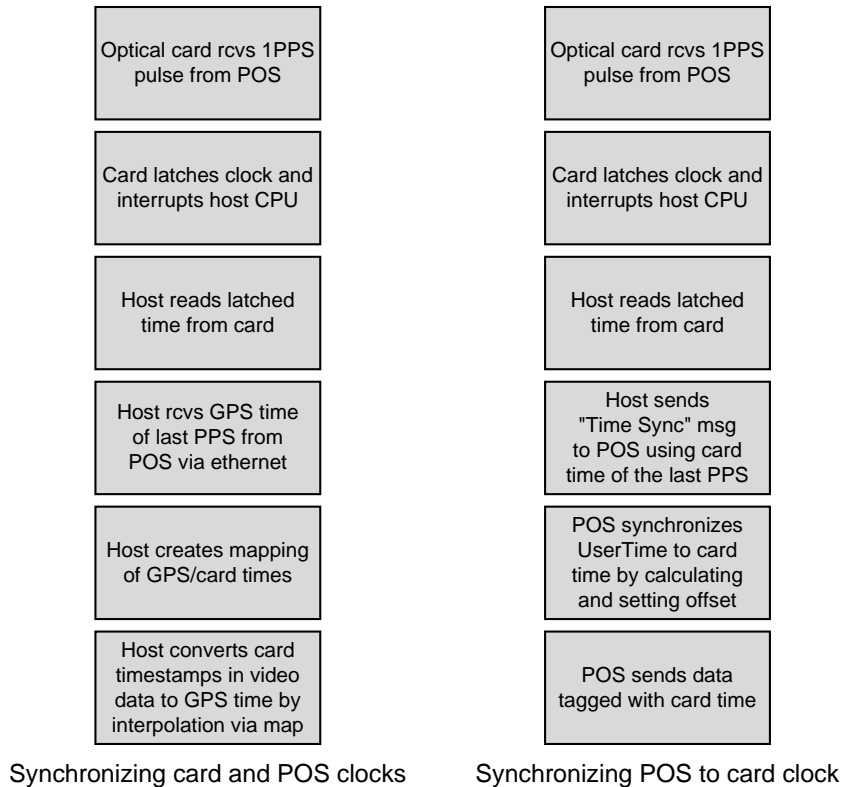


Figure 2.3: Time Synchronization

It should be noted, however, that there is a certain delay between the optical measurement on the sensor itself and the reception of the corresponding pixel on the optical interface card. This delay is caused by the exposure time, A/D (analog to digital) conversion, serializing and de-serializing of the pixel data, encoding and decoding for the physical layer of the optical link as well as the actual transmission time through the fiber. This delay, as well as its variation has to be characterized during inter-mission calibration and used in scientific post-processing.

The positioning component is connected via ethernet to the on-board computer. The data obtained from the positioning component is already tagged with two different timestamps which can be chosen from four different sources: internal clock, GPS time, UTC time, or *user time*. The latter is calculated by adding a fixed offset to the internal clock. This offset can be set by the on-board computer sending a `time sync` message to the positioning component.

To summarize, optical data is tagged by the optical interface card's clock whereas positional data is tagged with GPS or its own internal clock. It is these two clock bases which must be synchronized. Three different methods for doing this were investigated and in order to improve reliability, it was ultimately decided to implement two of them concurrently. This requires little additional effort since most of the steps of these two methods are the same. These processes are illustrated in Figure 2.3.

2.2.4 Other features

Once the above three critical performance problems were satisfactorily addressed, it was determined that the system could indeed be mostly implemented at the scripting level. It was at this time that some additional scripting components were defined:

Message Bus Since it was determined that multiple processes would be interested in sending commands to the instrument and multiple (possibly different) processes would be interested in receiving instrument status data from the instrument, the concept of a message bus was developed. Any process wishing to listen to messages on a bus `subscribe` to a particular topic and then receive (asynchronously) any messages `published` on that topic. The implementation makes use of event handlers to process asynchronous messages.

Runtime Configurability In order to maintain configuration flexibility, it was decided to implement instrument command sequences (e.g. `cool to 70deg K, set filter wheel to position 3, begin recording`) as scripts. The flight management system can for example automatically trigger these scripts when it reaches pre-programmed waypoints during a data take. By making these sequences scriptable, the full flexibility of the system is always available to the operator.

Regression Tests Regression tests have been implemented for individual components throughout the development of the system. Additionally, once camera operation became scriptable, it enabled system wide regression tests to also be scriptable.

2.2.5 Performance

Complete system wide throughput has not been measured since the initial scripted proof-of-concept was validated and put in place. It is planned that the regression testing framework can also be used to maintain a regularly runnable throughput performance test.

The goal of such a test would be to identify the upper limit of the data rate the final system is able to feed through from the detectors to the disks, while doing correctly all the modifications on the data block as described above (merging housekeeping data, rectifying timestamps, extracting waterfall images). Therefore, performance measurements have to determine that upper bound and its governing parameters, while ensuring that data integrity and order is guaranteed.

As outlined above, the on-board computer must buffer data coming from multiple sources and write them to disk via multiple writers. This is a data flow problem similar to determining how long it takes to fill a bathtub when it has a leak.

Note that both incoming and outgoing data rates may vary over time, e.g. due to user programmable frame rate changes or even differing disk zones during writing. The governing parameters therefore are:

incoming data rate governed by DMA block size and PCI bus business

outgoing data rate governed by bus speed and write capacity of the disks

CPU load governed by merging, extracting, rectification tasks on the buffer itself as well as other running applications and the operating system itself.

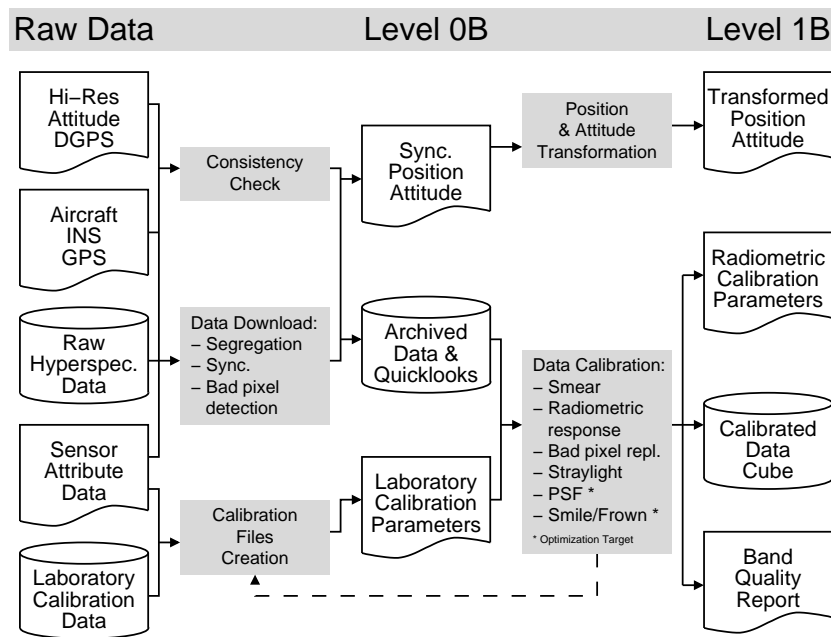


Figure 2.4: Post Processing Requirements

Therefore, the performance measurements should measure the incoming data rate and use that as the independent variable for all following measurements:

- Measure the outgoing data rate depending on:
 - the incoming data rate (i.e. by DMA buffer size, assuming that we can control the PCI load, since it is a dedicated PCI bus)
 - the size of the buffer (i.e. the number and sizes of slots in shared memory)
 - CPU load
- Determine the failure rate for that same parameter set, calculated from:
 - any deviations of data order on the final system
 - difference between sent-in and written-to-disk data blocks

2.3 Scientific Post-Processing

It is expected that individual flight campaigns will collect data on the order of 100s of GB that need to undergo an offline chain of data correction and characterization processes based on previously acquired and in-flight calibration parameters [Schaepman et al., 2000]. This processing chain includes conversion of raw data values into SI units, bad pixel replacement, and correction of smear, straylight, smile and frown anomalies. Higher-level processing is also planned, such as correction of at-sensor radiance values to ortho-rectified ground reflectance considering atmospheric and geometric effects [Schläpfer and Richter, 2002], [Richter and Schläpfer, 2002]. However, this higher level processing will be addressed in a later phase of the project and the

current phase needs only ensure that the parameters and data required for that level of processing are produced and made available.

A simplified block diagram of the planned processing is illustrated in Figure 2.4. The data acquisition process described in the previous section produces the top four components on the left side in the *Raw Data* column. The lower two components are produced during inter-mission calibration of the instrument which takes place in a laboratory known as the *Calibration Home Base*.

At this point all of the raw data is still present in the on-board computer and needs to be transferred to the off-line processing and archiving facility (PAF) computer. During this data transfer, quick consistency checks are made, and some simple constant-time operations can be performed such as bad pixel detection as well as generation of a high-resolution composite RGB pseudo-color *quicklook* image. During this data download phase, some intermediate files are created for use during scientific data calibration processing phase which ultimately produces what is known as the level 1B data product.

2.3.1 Key Algorithms in Foreign Language

Essential post-processing algorithms for hyperspectral data calibration and analysis are usually developed by scientists using special purpose high-level data modeling languages such as TMW's MATLAB and RSI's IDL. Because many of these correction and calibration algorithms are an active topic of research, it was strongly desired to leave scientifically sensitive processing algorithms in their original modeling language as much as possible to facilitate peer-reviewed validation as well as the ability to easily incorporate updates according to the latest advances. But while these languages are often ideal for their domain, they often don't provide convenient interfaces to interesting external software components such as relational database management systems, web servers, and cluster framework libraries.

However, these modeling languages do often provide a C programming interface to their internals allowing them to be embedded into other applications. This is certainly true in the cases of MATLAB and IDL.

A small experiment was developed to investigate the feasibility of embedding an IDL interpreter inside a Tcl interpreter in order to allow program logic to be developed in Tcl, while allowing scientific algorithms to be processed in IDL.

The experiment involved developing an `idl.so` Tcl extension which allowed creation of an interpreter for the invocation of commands as strings, and combining it with the `websh.so` [Vckovski et al., 2003] Tcl extension in order to implement an interactive web interface to an IDL command line. The user was able to enter arbitrary IDL commands into a form entry screen which was processed by the embedded IDL interpreter and the results formatted in HTML for display in a web browser.

When this experiment proved successful, additional commands for importing and exporting numerical arrays to/from IDL were added to the `idl.so` extension in order to be able to directly pass data to other Tcl extensions.

2.3.2 Additional Components

The proof-of-concept mentioned above was extended to allow access to an RDBMS (relational database management system) by way of the `mysql.so` [Soderlund et al., 2003] Tcl extension to allow for browsing and/or querying the hyperspectral data archive. It was then extended

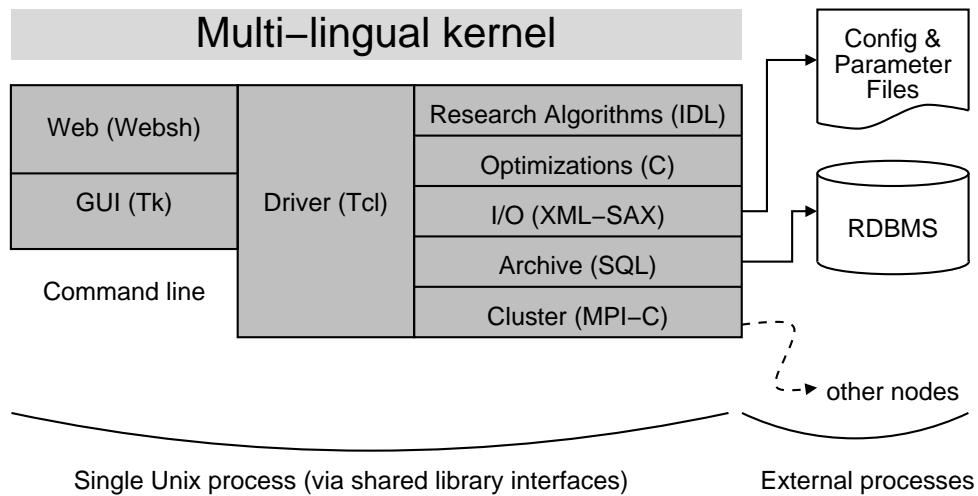


Figure 2.5: APEX Post Processing Architecture

with the `tdom.so` [Löwer and Ade, 2003] to allow for parsing, manipulation, and writing of XML based meta data for information that is expected to be shared with other (external) processing systems.

It has been determined that even with the large data volumes involved, most of the basic scientific level 1 processing can be performed within the allowed time constraints using a single server class computer. However, it is possible that smile/frown and point spread anomalies that need to be corrected by re-sampling in both the spatial and spectral dimensions simultaneously might require enough processing power to merit the use of cluster processing. Investigation of the particular problems and possible correction algorithms are currently underway. In the case that processing would benefit from cluster computing, experimentation with a Tcl extension to allow access to the MPI-C (Message Passing Interface) library is tentatively planned.

The current version of the prototype kernel is illustrated in Figure 2.5.

An additional ESA driven requirement of the processing system is that key calibration algorithms should be documented and that this documentation should be updated whenever the algorithm is updated. While some argue that IDL is a high enough level language to be self-documenting, this argument doesn't meet typical ESA definitions. Therefore, an IDL documentation system similar to Java's javadoc was developed to allow programmers to embed documentation inside comments in the code itself. This includes standard tags for things such as describing inputs, outputs, and side-effects. However, it additionally allows arbitrary \LaTeX code to be embedded in a description field so that accompanying mathematical formulas can be easily expressed. Automatically generated documentation from the code is then included into an overall algorithm description document that accompanies the processing software.

2.3.3 Development Process

Another factor involved in the development of a software architecture for scientific computing is that many of the developers are scientists. Ideally, all software development team members would be equally able to implement all tasks independently and asynchronously – and with roughly equivalent efficiency. However it is more common to find that decomposed subproblems

have an interdependence on each other, and that different scientific developers have different development strengths and weaknesses and levels of development efficiency.

One way to mitigate problems in this area and to ensure coherence in the overall design is to adopt a prototype-based iterative development model [Boehm, 1988]. The first iteration consists of simulating program flow using a high level prototyping paradigm and subsequent iterations involve refining the simulated steps by gradually replacing them with more realistic pieces.

This is the model we prefer and has driven the architecture and design of all the software described in this paper. Trouble spots are predicted, experiments are developed to investigate them, and then prototypes are initiated accommodating the solutions to the trouble spots while making way for the remaining tasks to be filled in.

Development efficiency, which is different from and often more important than run-time efficiency is also further improved by allowing continued use of multiple development environments from the prototype phase throughout the development process. The key enabling technology for this concept is attempting to arrange automatic inter-operability between these different runtime systems. This allows, for example, one scientist to develop a particular calibration algorithm independently from other team members using his most efficient language and development environment (e.g. IDL) while allowing another team member to independently develop the inner loop of a cubic convolution re-sampling algorithm in C or Fortran, and yet another developer to work on web-based form driven GUI code or SQL access to the database management system. If the resulting system has been architected to support interfaces between each of these pieces at runtime, the efficiency of the development and maintenance of the code should be high.

In this way, the development process itself can have a large effect on the software architecture design decisions that are made.

2.4 Conclusions and Outlook

We have discussed the analysis and requirements of system and application software for the APEX airborne pushbroom imaging spectrometer. We described how we analyzed these requirements to develop simple experiments to test critical components of a potential software architecture. Based on successful experimental results, we developed initial prototype software

scripts	
gencode	code generation from DB schema
idldoc	automatic documentation from code
testsuite	regression test suite
extended	
idl.so	IDL execution and data import/export
mysql.so [†]	RDBMS client
svipc.so [†]	Shared memory and semaphore interface
syslog.so	Interface to system error logger
tdom.so [†]	XML processing
udp.so [†]	UDP (for waterfall image server)
veu.so	data acquisition via PCI device driver
websh.so [†]	Web application framework
embedded	
apex	Hardware control command language
msgbus	Message bus library
[†] Already existing extensions	

Table 2.2: Uses of embeddable/extendible scripting

architectures for both on-board data acquisition as well as offline scientific post processing of hyperspectral data.

The initial prototyping phase resulted in the development of key components that are expected to be used without major interface changes throughout the development lifetime of the software. A summary of these components is given in Table 2.2.

The data acquisition software will soon be prototyped against a hardware simulator and full data acquisition system integration is planned for the end of 2003.

While the level 1 post-processor has enjoyed a more detailed design specification, its implementation is at a less finished stage of development than the data acquisition system. Many of its components are well-defined and just a simple matter of programming. However, some particular algorithms (e.g. for correction of smile/frown and point spread anomalies) are still in the investigative experimental and prototype phases. Nevertheless, the first functional delivery of the level 1 post processor is scheduled for the beginning of 2004. First end-user flight campaigns for the APEX spectrometer are scheduled for mid 2005.

References

- [Boehm, 1988] Boehm, B. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72.
- [Itten et al., 1997] Itten, K. I., Schaepman, M., De Vos, L., Hermans, L., Schläpfer, H., and Droz, F. (1997). APEX - airborne prism experiment: A new concept for an airborne imaging spectrometer. In *3rd Intl. Airborne Remote Sensing Conference and Exhibition*, volume 1, pages 181–188.
- [Kelsey, 2003] Kelsey, J. (2003). Tcl interface to System V IPC facilities. [Online; accessed Jul-2003]. <http://www.neosoft.com/tcl/ftparchive/sorted/net/svipc-2.2.0/>.
- [Löwer and Ade, 2003] Löwer, J. and Ade, R. (2003). tDOM - a fast XML/DOM/XPath package for Tcl written in C. [Online; accessed Jul-2003]. <http://www.tdom.org>.
- [Miller et al., 2003] Miller, M., Wu, X., and Thoyts, P. (2003). Tcl UDP extension. [Online; accessed Jul 2003]. <http://sourceforge.net/projects/tcludp>.
- [Richter and Schläpfer, 2002] Richter, R. and Schläpfer, D. (2002). Geo-atmospheric processing of airborne imaging spectrometry data. part 2: Atmospheric/topographic correction. *International Journal of Remote Sensing*, 23(13):2631–2649.
- [Schaepman et al., 2002] Schaepman, M., Schläpfer, D., Brazile, J., and Bojinski, S. (2002). Processing of large-volume airborne imaging spectrometer data: the APEX approach. In *SPIE Imaging Spectrometry VIII*, volume 4816, pages 72–79, Bellingham, Washington, USA.
- [Schaepman et al., 2000] Schaepman, M., Schläpfer, D., and Itten, K. (2000). APEX - a new pushbroom imaging spectrometer for imaging spectroscopy applications: Current design and status. In *Proc. IGARSS 2000*, volume VII, pages 828–830, Hawaii.
- [Schaepman et al., 2003] Schaepman, M., Schläpfer, D., Kaiser, J., Brazile, J., and Itten, K. (2003). APEX - airborne prism experiment: Dispersive pushbroom imaging spectrometer for environmental monitoring. In *Proc EARSel 3rd Workshop on Imaging Spectroscopy*.
- [Schläpfer and Richter, 2002] Schläpfer, D. and Richter, R. (2002). Geo-atmospheric processing of airborne imaging spectrometry data part 1: Parametric orthorectification. *International Journal of Remote Sensing*, 23(13):2609–2630.
- [Soderlund et al., 2003] Soderlund, H., Gulik, G., Ritzau, T., P.Brutti, and Trzewik, A. (2003). mysqltcl - Tcl Mysql interface. [Online accessed Jul 2003]. <http://www.xdobry.de/mysqltcl/>.
- [Vckovski et al., 2003] Vckovski, A., Brunner, R., and Hefti, S. (2003). Websh: The Tcl web application framework. [Online; accessed Jul-2003]. <http://tcl.apache.org/websh>.

Chapter 3

A Domain-Independent RESTful Grid Service for Non-Programmers

Brazile, J., Low Fat Grid: A RESTful Grid Service for Non-Programmers, Jazoon '07 – The International Conference on Java Technology, 2007, CD-ROM.

Reprinted with Permission.

Abstract

The design and implementation of a RESTful web service providing grid functionality is presented. This web service enables small, geographically dispersed groups of non-programmers to voluntarily share their computer resources to work on collective problems. Only three prerequisites must be met: 1) potential compute machines must have a Java runtime environment (JRE) installed and outgoing Internet web access, 2) at least one group member must be able to install a single, standalone machine-independent, unprivileged, perl CGI program on some Internet accessible server, and 3) at least one user is able to produce a tar format archive representing a batch of jobs – one directory per job, each containing relevant input files as well as a command file.

The client acts as a mini web browser: requesting jobs, executing them, then uploading the results using the same file upload mechanism used by web-based email clients. Job scheduling is self-balancing since clients pull jobs rather than the server pushing them – faster clients will request and process more jobs than slower clients.

A prototype of the system has been used for two scientific applications using spare resources on heterogeneous machines at institutions located in different countries. In one case, the application reproduced work previously performed on super computers. In the other case, distributed/remote execution offered a unique collaboration opportunity by working around IP restrictions – the release-restricted software located at one institution produced data that could be transferred and further processed at another institution, whose auxiliary data was release-restricted.

In addition to its unique ease-of-deployment, performance is shown to be favorable when compared to Condor, the leading distributed processing framework for non-programmers.

3.1 Introduction

There has been much interest in shared resource computing. However, except for file-sharing (movies, music, software) and a few special case applications (seti@home [Korpela et al., 2001], folding@home [Snow et al., 2002]) most grid frameworks have too many barriers to entry (e.g. programming expertise, access to infrastructure, etc) to support exploitation by non-programmers. There are many existing tools that try to address this problem, yet all fail to meet at least one criteria required by the author: ease-of-deployment and exploitation, cross-platform, and able to work across firewalls. Apple's Xgrid [Apple Xgrid, 2004] has a novice-friendly interface, but is limited to the MacOS X platform and WAN-usage requires port 4111 to not be blocked. Condor [Litzkow et al., 1988] also appeals to non-professional programmers but deployment and configuration is complicated. Also intentional disregard for low-latency [Palatin and Kliot, 2003] means that for our workload, total execution time could be up to twice as long as required. With Condor, cross-firewall operation is obscure at best [Beckles et al., 2005]. Boinc [Anderson, 2004b] is able to execute on the wider Internet, but application development is beyond the capabilities of most non-programmers. Furthermore, deployment is non-trivial and requires elevated privileges on an Internet server. The Globus framework [Foster and Kesselman, 1997] appears to offer the widest variety of services, but this richness also exposes a large level of complexity. Somewhat closer to meeting our non-local requirements is the experimental GFac2 project [Kandaswamy et al., 2006], however the list of runtime pre-requisites and deployment process place it beyond the goal of "simple for non-programmers" and therefore disqualify it as a *low fat* choice.

After the initial prototype was developed, it was determined that a web-service style interface could make the software more widely applicable by seamlessly supporting mash-ups. The first such grid-as-web-service known to the authors is the WSRF::Lite project [McKeown, 2004]. Unfortunately, it takes the more heavy-weight WS-style approach rather than the simpler Representational State Transfer (RESTful) [Fielding, 2000] approach to web services.

The *low fat grid* project [Brazile, 2006] attempts to solve the aforementioned problems by providing a two file solution implementing a RESTful web service and a client for that service, which are expected to be deployed and configured in a ready-to-use state within minutes. The CGI-based server is deployable on low-cost, commodity, web hosting accounts, and the client can run on any reasonable JRE-enabled compute host. Because its RESTful protocol is HTTP based, only typical web transactions are used – no firewall issues arise. As with many RESTful services, a standard web-browser could be used in place of a client, although that makes sense only for debugging or protocol investigation.

Job scheduling is self-balancing in that the clients pull jobs rather than the server pushing them. This means that faster clients will request and process more jobs than slower clients. Control actions provides monitoring of job execution and the ability to upload work packages as batches i.e, an archive of jobs, one per sub-directory.

A prototype of the system [Brazile, 2005] has so far been used for two real-world scientific applications using spare resources on heterogeneous (MacOS X, Linux, Solaris, and FreeBSD) machines at institutions located in different countries. In one case, the application reproduced work previously performed by super computers [Brazile et al., 2005]. In the other case, distributed/remote execution provided a unique opportunity in that for one of the collaborating institutions, critical processing software was not allowed to leave the premises, and in another, critical research data was not allowed to leave the premises – yet in both cases, data was allowed to be processed locally then sent on to be further processed at a remote location [Dangel et al., 2005].

As computer and network infrastructure increase in performance and decrease in price, it has

become common in even small companies to have more computers on the network than can be interactively used by personnel. The following table shows data from two departmental networks available to the author:

	# hosts on network	# active users	host/user ratio
Medium-sized IT Firm	667	≈ 70	≈ 9
University Department	372	≈ 150	≈ 2

Aside from more efficient use of spare computing resources, a carefully constructed distributed grid implementation could also offer a novel solution for sharing artificially constrained resources. Such artificial constraints might be imposed for many reasons ranging from intellectual property issues to simple licensing issues. These constraints might be imposed in critical or unique processing software, or critical or unique input data. In some fields, it might be more common to encounter seemingly insurmountable artificial constraints than technical constraints. However, sometimes such constraints can be worked around if processing can be decomposed into pieces allowing some sub-portion to be processed at one location and later processing elsewhere – each phase constructed to run in a location allowed by those constraints.

3.2 Implementation

3.2.1 Design Goals

The design of the *low fat grid* project was driven by the following requirements.

Cross-Institutional: Access to computing resources in geographically distinct locations should not only be possible but easy. Most workstations can browse the web (via proxy?), but no other ports.

Web-Service Philosophy: Provide wider appeal via potential mashups that follow naturally from a RESTful web-service approach.

Simple Server Deployment: Server deployment shouldn't be more difficult than copying a file to the CGI directory of a commodity web server installation. Neither java application servers nor servlet engines are typically available on commodity hosting accounts.

Simple Client Deployment: For deployment on machines without interactive users, a one-line shell for loop using ssh should suffice. Those with interactive users might receive an email or telephone call asking them to perform a reasonable procedure e.g. right-click on a URL in an open web browser and after launching, type in a project-specific URL.

Multi-Platform Clients: Linux, MacOS and win32 based clients would probably cover the majority of useful workstation configurations, but a generic solution that can cover these diverse platforms at the same time can probably trivially cover others.

Based on these goals, it was decided that a good solution would involve implementing a RESTful web service with a single CGI-file on the server side and a JRE-based client. The tar format [Stallman, 2006] was chosen as a platform-neutral archive container, with abundant user-friendly tool support, as well as being easy to implement for platforms that lack direct support (e.g. Microsoft-based web hosting platforms).

Resource	Action	Behavior
batches	GET	Retrieve list of batches with status
	POST	Request URI creation for next runnable job (any batch)
batches.tar	GET	Download archive of all batches
	POST	Request URI creation for new batch
batches/NNNNN	GET	Retrieve list of jobs for batch NNNNN with status
	POST	Request URI creation for next runnable job (for batch NNNNN)
batches/NNNNN.tar	GET	Download archive of batch NNNNN
batches/NNNNN/MMMMM	GET	getJobStatus()
batches/NNNNN/MMMMM.tar	GET	Download archive of job MMMMM
	PUT	Upload results for job MMMMM

Table 3.1: Grid service resources and actions. PUT is implemented with POST for portability

3.2.2 Resource-Oriented vs Service-Oriented

The WS-style or service-oriented approach to web services is a natural extension to remote procedure call (RPC) style programming. Individual operations are wrapped by what appear to be procedure calls. This approach can be straight-forward to implement, but because of the lack of constraints in the design of the interface, clients programmed to interact with one such service do not enjoy reduction in effort when needing to interact with another service. One of the few constraints is that all messages are to be encoded in XML. Not only does this add complexity when simpler data formats could suffice, but problems arise when wanting to stream binary data.

Contrarily, the RESTful or resource-oriented approach to web services is a natural extension to the web, or the HTTP protocol in particular. Instead of wrapping arbitrary procedure calls, key entities are identified and established as resources (URIs) and the concept of mapping is used to operate on the resources. In particular, typical Create/Read/Update/Delete (CRUD) operations are defined and mapped to the standard HTTP commands e.g. GET and POST. A RESTful approach is perhaps not always applicable, but when it can be used, it typically offers benefits over a WS approach.

The uniform operations expected in dealing with the HTTP protocol allows for reuse of logic and supports mash-ups more easily because once one has already coded a retrieval using HTTP GET and coded the handling of possible responses from that retrieval, much of this code is directly reusable on another such retrieval. Standard tools for operating on the web are often directly applicable. Tools such as `curl` can be used as simple test clients, spiders can be used for coverage testing, web proxies and caches can be used for security and scalability, etc).

3.2.3 Resource Oriented Service Interface

The *low fat grid* application is made up of two small components: 1) a CGI dispatcher with an embedded job scheduler and 2) a standalone client. The RESTful protocol [Fielding, 2000] used by the client and CGI dispatcher is described in Table 3.1 with an example message exchange provided in Table 3.2.

The client acts like a miniature web-browser and periodically polls the web-based server to see if there are jobs to be run. If so, the web server's response is a page containing a link

Intention	Request	Response
Upload new batch	POST batches.tar Content-Type: application/x-tar ...	202 Accepted Location: batches/19682 A new batch has been allocated
Alternate discovery	GET batches	200 OK Content-Type: text/html ... Batch Name
Request next job	POST batches/19682 Content-Type: text/plain ...	202 Accepted Location: batches/19682/00001 A new job has been allocated
Get it	GET batches/19682/00001.tar	200 OK Content-Type: application/x-tar ...
Upload processing results	POST batches/19682/00001.tar Content-Type: application/x-tar ...	202 Accepted Location: batches/19682/00001.tar Job has been committed
Get batch	GET batches/19682.tar	200 OK Content-Type: application/x-tar ...

Table 3.2: Example: Message exchange for a batch containing a single job

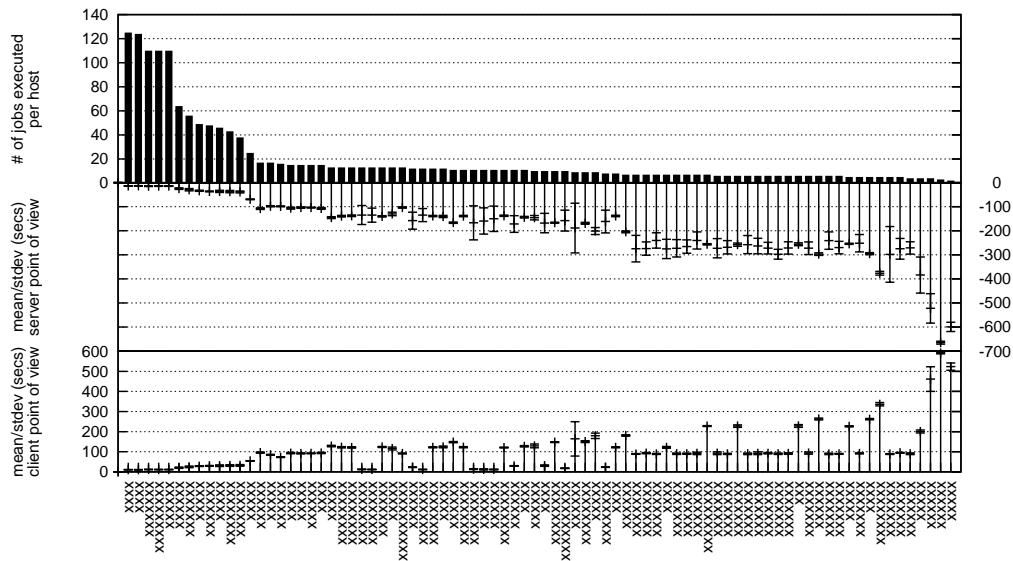


Figure 3.1: Distribution of jobs on 82 heterogeneous worker machines (hostnames elided)

corresponding to a job's input files and command file. If the client downloads a job, runs it and finishes it, it will contact the web server and CGI POST its output files as a file upload using the same protocol a browser-based email system uses to upload/include attachments.

On the server side, the CGI application fields HTTP requests from the client, queries the job scheduler, then formats the results as HTML output for the client. The job scheduler is logically separate from the CGI application since it needs to maintain state and provide serialized access to the jobs. While logically separate, it is contained physically within the single server application.

Job scheduling is self-balancing in that clients pull jobs rather than the server pushing them. This means faster clients will request and process more jobs than slower ones.

3.3 Example Deployments

Here are some specific use-cases for possible *low fat grid* deployments. The first two have been realized, while the remaining two are hypothetical.

3.3.1 Atmospheric Modeling

A scientist wants to model a complete set of earth atmospheric data representing situations resulting from variations in input parameters such as altitude, amount of water vapor, sun zenith angles, surface elevation, etc. The last time he did this, his personal workstation was tied up for a month performing the calculations. He knows of 10 idle workstations in his institution and his colleague at the university has access to 60 more. Together they conspire to run the jobs on all the spare machines over a weekend. He writes a small program to enumerate all the thousands of input files and write them each in a separate subdirectory along with a set of commands to run a single instance of the model. This input generation program takes 2 minutes

to run. He uploads the .tar archive of these directories to the *low fat grid* server running on his colleague's departmental web server at the university. He and his colleague both start the clients on their collective heterogeneous worker machines. At the end of the weekend, he collects all the results that had been automatically uploaded, verifies their correctness and together they publish a journal paper. An example time distribution of jobs can be seen in Figure 3.1). Some observations will be made about this in Section 3.5.1.

3.3.2 Satellite Mission Planning via Simulation

Research groups in different countries have been selected by the European Space Agency to submit a comprehensive mission plan for a new proposed Earth Observation satellite. Each group has been selected to participate on the grounds of that particular group's specialized expertise, which is different from that of the others. These specialties range from collection of ground truth databases, complex algorithms for atmospheric modelling, scientific instrument design and calibration, and system integration. The goal is to develop a mission simulator that validates the entire process of recording satellite data from observing a varied set of earth observation situations, transmitting the filtered data to a processing station, and applying inverse models to attempt to reconstruct reality based on the acquired observations. One of many problems is that one institute is not legally allowed to release its processing software or even details of its algorithms. Another institution prefers not to release its large database of ground truth data. Yet the end goal could still be realized, by showing a proof-of-concept based on the *low fat grid*, which would allow intermediately processed data to travel from institution to institution during the processing chain – permitting “local” use of a vast input database in one case, and “local” use of proprietary processing software in another.

3.3.3 Render Farm

An artist and new mother has recorded many hours of home movies covering the first year of her child's life. She decides to make a DVD, introducing each video chapter with an animated title sequence, where characters look like they are walking, taking a bath, eating etc.

She has created key frames for opening chapter sequences using some 3D modelling software, puts each resulting control/input file in a subdirectory along with a text file containing commands to run a rendering program like Pov-Ray [Orf, 2004] using the input file. She then copies the *low fat grid* server to the CGI directory of her ISP-provided home page. Then with her browser, she visits the server's URL where she is prompted to upload her .tar file. She calls two of her friends on the phone and asks them to visit her home page, download and start up a *low fat grid* client. They all watch the progress of the work over the day, by (re)-visiting her home page. A few hours later, all rendered sequences have been automatically uploaded back to web server and she can now download them to finish authoring her DVD.

3.3.4 Document Digitizing

As recommended by the tax bureau, a user has saved all his paper billing statements over the last seven years from his banks, insurance companies, investment brokers, etc. He wants to get rid of all this paper and possibly even streamline his budget by analyzing how the bills have changed over time. He has a scanner, but it is only a single sheet flat bed scanner, so instead he goes to the office where he can put a stack of papers in a batch scanner that can email the scanned bitmaps. He has a friend who has purchased a license to a high quality optical

character recognition (OCR) program that can convert files in bitmap format into Microsoft Word documents. He has another friend who has purchased a license to Acrobat writer that can convert Microsoft word documents to PDF. He has created a command text file for the batches of papers which run the OCR software, a set of commands for launching his friends Word/Acrobat program to print a .doc to a file [King, 2006]. He copies the *low fat grid* server to the CGI directory of his ISP-provided home page and asks his two friends to download and run the *low fat grid* client.

3.4 Related Work

The Boinc API and library [Anderson, 2004a] is the most visible generic grid framework, made popular via its SETI@home [Korpela et al., 2001] and Folding@home [Snow et al., 2002] applications. However, it is not simple to deploy and applications are restricted to those with a “Low data/compute ratio” i.e. those which “produce or consume more than a gigabyte of data per day of CPU time” [Anderson, 2004a]. Apple Computer’s Xgrid [Apple Xgrid, 2004] looks promising for its ease of use but is restricted to MacOS X users and local area networks. Condor [Litzkow et al., 1988] and PVM [Sunderam, 1990] are freely available and still widely used but are practically (though, not technically [Beckles et al., 2005]) limited to local area networks, and not user-friendly to deploy. Grid as a web service is also not new. For example, the WSRF::Lite project [McKeown, 2004] has seen multiple revisions and the GFac project [Kandaswamy et al., 2006] is in its second incarnation. The former is starting to evolve into a more RESTful approach but is not there yet. The latter provides a more general way to automatically *wrap* existing programs with a web service which may offer suitability for a wider range of applications than *low fat grid*, but its runtime requirements and deployment complexity are still beyond the realm of most non-programmers.

3.5 Current Status and Future Work

The prototype version of *low fat grid* has been successfully employed in two scientific applications involving 1) atmospheric modelling for the processing of satellite/airborne imaging data and 2) the validation of the design of an earth observation satellite and its data processing chain. The code for the current web service based on that prototype has been released under the GNU General Public License (GPL) version 2 [Stallman, 1989] and is available via Google’s project hosting service [Brazile, 2006].

3.5.1 Usability and Efficiency

Toward fulfilling its limited goals of providing an easy way to get a job done with little effort, the *low fat grid* has met all of its stated goals. However, one inherent usability issue that has not been addressed is: How many machines does it make sense to dedicate toward solving a problem? In the runtime distribution statistics of the example application shown in Figure 3.1, it is shown that 82 machines were employed toward running jobs. Yet from this plot, it is clear that using anything other than 12 fastest machines were mostly a waste of time. In fact, including the slowest 77 machines probably didn’t improve overall runtime by even a factor of two. It would improve usability if a user could get this feedback without having to run an entire batch before discovering it.

Another interesting feature from Figure 3.1 is the difference in mean/standard deviation of run times from the server's point of view in contrast to the client's point of view. Of course this reflects the network bandwidth and latency between the server and client while transferring input/output files. Some of this is bad luck due to congestion i.e. if job start/end happen to be somewhat synchronized. But if input/output files are non-trivial in size, then network performance between the client and server may begin to play just as significant a role in job throughput as raw client compute performance. Thus network performance between server and client is another piece of feedback that could potentially improve usability. On the other hand, the effect of this could be somewhat alleviated if the client were able to perform network I/O in parallel to computation. Such an improvement could greatly improve some classes of problems.

For our application, runtime performance compared favorably to implementation on a dedicated local cluster, when the number of compute nodes is small to moderate i.e., under 16 [Brazile et al., 2005]. Surprisingly, for that same application, the *low fat grid* prototype produced results in half the time of a local installation of Condor [Brazile et al., 2005]. This was determined to be due to Condor's deliberate pause between jobs (non-configurable), and our application profile of a larger number of jobs running on the order of minutes, rather than a small number of jobs running on the order of hours.

3.5.2 Recommended Improvements

To make the *low fat grid* more acceptable to a wider audience, it is believed that three essential features should be added – finish the job control/monitoring interface, improve the client, and address security.

The job progress/status interface is very primitive and canceling jobs and/or batches is not yet possible with the client or browser. For jobs running more than a few hours, this is not acceptable in a serious product.

To convince interactive users to share their machine's resources, they need to have capabilities such as the ability to temporarily disable grid jobs, to monitor the workstation resources that grid jobs are using and to get positive feedback, such as a personalized ranking or interesting report/visualization of the problem being solved. Local machine resource monitoring and control shouldn't be too difficult to add. It is not currently known how the *low fat grid* client could support generic graphical visualization beyond simple counters or progress bars.

In the interest of simplicity, security has been ignored up til now. There are at least two security issues that should be addressed: security against running undesired code (i.e. viruses) and batch application security against receiving deliberately incorrect results.

Currently, the only security feature designed in *low fat grid* is MD5 [Rivest, 1992] checksumming, which ensure that data transfers are reliable. However, one benefit of relying on a RESTful interface, is the ease of adopting an SSL-encrypted HTTPS channel in place of a clear text HTTP channel, when desired. The client itself could verify that the server delivers a valid server certificate. And since the connection would be authenticated at both ends, the application manager could keep better control of who is running worker clients.

3.6 Summary

The design and implementation of the *low fat grid* has been described. The application's utility has been illustrated in employing spare resources on typical computers from multiple locations

to collaborate on example applications that 1) have been in the past run on supercomputers and 2) provide a work-around for institutional intellectual property restrictions.

By adhering to clear usability design principles, the key requirements of simplicity of deployment, firewall-agnosticism, client heterogeneity and non-limited problem domain were met in a “low fat” web-service based application.

References

- [Anderson, 2004a] Anderson, D. P. (2004a). The berkeley open infrastructure for network computing. [Online; accessed Jul-2004]. <http://boinc.berkeley.edu/>.
- [Anderson, 2004b] Anderson, D. P. (2004b). Boinc: A system for public-resource computing and storage. In Fifth IEEE/ACM International Workshop on Grid Computing, pages 4–10.
- [Apple Xgrid, 2004] Apple Xgrid (2004). Xgrid. [Online; accessed Jul-2004]. <http://www.apple.com/acg/xgrid>.
- [Beckles et al., 2005] Beckles, B., Son, S.-C., and Kewley, J. (2005). Current methods for negotiating firewalls for the Condor system. In Cox, S. and Walker, D. W., editors, Proc. 4th UK e-Science All Hands Meeting.
- [Brazile, 2005] Brazile, J. (2005). Grid hack. [Online; accessed Sep-2005]. <http://sourceforge.net/projects/ghack/>.
- [Brazile, 2006] Brazile, J. (2006). Low fat grid. [Online; accessed Nov-2006]. <http://code.google.com/p/lowfatgrid/>.
- [Brazile et al., 2005] Brazile, J., Richter, R., Schläpfer, D., Schaepman, M. E., and Itten, K. I. (2005). Cluster versus grid for operational generation of ATCOR's MODTRAN-based look up tables. Parallel Computing. in review.
- [Dangel et al., 2005] Dangel, S., Brazile, J., Petitcolin, F., Kneubuehler, M., Schaepman, M. E., and Itten, K. I. (2005). The design and prototyping of the SPECTRA simulator architecture. In Zagojowski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, Warsaw, Poland.
- [Fielding, 2000] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine.
- [Foster and Kesselman, 1997] Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing, 11(2):115–128.
- [Kandaswamy et al., 2006] Kandaswamy, G., Fang, L., Huang, Y., Shirasuna, S., Marru, S., and Gannon, D. (2006). Building web services for scientific grid applications. IBM Journal of Research and Development, 50(2/3):249–260.
- [King, 2006] King, G. (2006). Printing word and pdf files from python. [Online; accessed Dec-2006]. <http://www.darkcoding.net/software/printing-word-and-pdf-files-from-python/>.

- [Korpela et al., 2001] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Lebofsky, M. (2001). SETI@home – Massively Distributed Computing for SETI. Computing in Science and Engineering, 3(1):78–83.
- [Litzkow et al., 1988] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor-a hunter of idle workstations. In The 8th International Conference on Distributed Computing Systems, pages 104–111.
- [McKeown, 2004] McKeown, M. (2004). OGSi::Lite - grid services in perl. In Second Annual Reality Grid Workshop.
- [Orf, 2004] Orf, L. (2004). Scientific visualizations with Pov-Ray. Linux Journal, 2004(127):2.
- [Palatin and Kliot, 2003] Palatin, N. and Kliot, G. (2003). Low latency invocation in condor. Technical report, Technion Computer Science Department, Israel. 22 pages.
- [Rivest, 1992] Rivest, R. L. (1992). The MD5 message digest algorithm. [Online; accessed Dec-2006]. <http://www.ietf.org/rfc/rfc1321.txt>.
- [Snow et al., 2002] Snow, C. D., Nguyen, H., Pande, V. S., and Gruebele, M. (2002). Absolute comparison of simulated and experimental protein-folding dynamics. Nature, 420:102–106.
- [Stallman, 1989] Stallman, R. M. (1989). Gnu general public license. [Online; accessed Dec-2006]. <http://www.gnu.org/copyleft/gpl.html>.
- [Stallman, 2006] Stallman, R. M. (2006). Basic tar format. [Online; accessed Dec-2006]. http://www.gnu.org/software/tar/manual/html_node/tar_134.html.
- [Sunderam, 1990] Sunderam, V. S. (1990). PVM: a framework for parallel distributed computing. Concurrency, Practice and Experience, 2(4):315–340. <http://citeseer.nj.nec.com/sunderam90pvm.html>.

Chapter 4

Cluster versus Grid for Operational Generation of ATCOR's MODTRAN-based Look Up Tables

Brazile, J., Richter, R., Schläpfer, D., Schaepman, M.E., and Itten, K.I., Cluster versus Grid for Operational Generation of ATCOR's MODTRAN-based Look Up Tables, *Parallel Computing*, Elsevier, 2005, in review.

Abstract

A critical step in the product generation of satellite or airborne earth observation data is the correction of atmospheric features. Due to the complexity of the underlying physical model and the amount of coordinated effort required to provide, verify and maintain baseline atmospheric observations, one particular scientific modelling program, MODTRAN, whose ancestor was first released in 1972, has become a *de facto* basis for such processing. While this provides the basis of per-pixel physical modelling, higher-level algorithms, which rely on the output of potentially thousands of runs of MODTRAN are required for the processing of an entire scene. The widely-used ATCOR family of atmospheric correction software employs the commonly-used strategy of pre-computing a large look-up-table (LUT) of values, representing MODTRAN input parameter variation in multiple dimensions, to allow for reasonable running times in operation. The computation of this pre-computed lookup table has previously taken weeks to produce a DVD (about 4GB) of output. The motivation for quicker turnaround was introduced when researchers at multiple institutions began collaboration on extending ATCOR features into more specialized applications. In this setting, a parallel implementation is investigated with the primary goals of: the parallel execution of multiple instances of MODTRAN as opaque third-party software, the consistency of numeric results in a heterogeneous compute environment, the potential to make use of otherwise idle computing resources available to researchers located at multiple institutions, and acceptable total turnaround time. In both grid and cluster environments, parallel generation of a numerically consistent LUT is shown to be possible and reduce ten days of computation time on a single, high-end processor to under two days of processing time with as little as eight commodity CPUs. Runs on up to 64 processors are investigated and the advantages and disadvantages of clusters and grids are briefly explored in reference to their evaluation in a medium-sized collaborative project.

4.1 Introduction

The ancestor of the MODTRAN line of atmospheric modelling software was first released in 1972, based on band models developed in the 1950's and 1960's used to describe atmospheric transmission and absorption behaviour [Anderson et al., 1994]. Over time, not only has the software been continuously updated [Berk et al., 1998], [Berk et al., 2005], but also the underlying molecular spectroscopic database [Rothman et al., 2003]. Although competing models also exist, MODTRAN has become established as a *de facto* standard in fields related to atmospheric physics and remote sensing. While many applications employing MODTRAN have usage patterns that require only tens of executions, others require hundreds or thousands. Examples of this usage include sensitivity studies [Schläpfer and Schaepman, 2002], [Schläpfer and Nieke, 2005], and the radiometric and spectral calibration of imaging spectrometers using known characteristics of solar and atmospheric absorption features [Green et al., 2003], [Neville et al., 2003], [Brazile et al., 2006].

When MODTRAN became more widely used in operational, remotely sensed spectroscopy, invasive modifications of the typically end-user-compiled Fortran source code were developed by a group of users in order to introduce single-run parallelism, which resulted in efficient and scalable speedups [Wang et al., 2002]. For whatever reason, these code modifications were not officially incorporated into the standard MODTRAN release, which has grown to over 80,000 lines of code. The code base has moved on, producing multiple releases since then, and these parallel modifications have effectively been lost to the wider scientific community.

Meanwhile, further standard applications, including atmospheric correction software such as employed by ISDAS [Staenz et al., 1998] and the widely-disseminated ATCOR family of software [Richter and Schläpfer, 2002], [Schläpfer and Richter, 2002] were being built using MODTRAN calculations as the basis of their own algorithms. Atmospheric correction of multispectral/hyperspectral imagery involve calculations that depend on a number of varying MODTRAN-modifiable parameters defining atmospheric conditions (e.g., molecular absorber concentrations, aerosol scattering, optical depth) as well as observer and solar geometry, i.e., flight altitude, heading, ground elevation, view and solar zenith and azimuth angles. Therefore a large number of MODTRAN calls are required to process a single satellite or airborne scene. In software such as ISDAS [Staenz and Williams, 1997] and ATCOR [Richter, 2005], these computations are usually performed off-line, and stored in LUTs prior to actual atmospheric correction of a scene to enable reasonable operational hyperspectral data processing times. The use of LUTs support the processing of a large variety of unrelated scenes in a much shorter amount of time than would be needed for direct computation.

In the case of ATCOR, researchers from multiple institutions began to collaborate, in an informal way, on extending functionality into novel special-purpose areas, ranging from the haze removal of low-spectral, high-spatial IKONOS imagery to the processing of hyperspectral, wide field-of-view (FOV) imagery as obtained from airborne (as opposed to space-borne) instruments. This increased collaboration also increased the desire for higher turnaround on LUT generation and, in turn, created the opportunity to pool the collaborators computing resources, in an informal way, to speed up LUT generation.

In the LUT generation usage of MODTRAN, individual executions are independent of each other and input data parameters can be precomputed ahead of time, leading to a workload that is "embarrassingly parallel" – i.e. there is no particular effort needed to segment the problem into a large number of parallel tasks. It is a common pattern in the employment of parallel computing for embarrassingly parallel problems to run multiple instances of a particular program over varying input parameters within a given problem space. This usage is explored here in both grid and cluster processing environments.

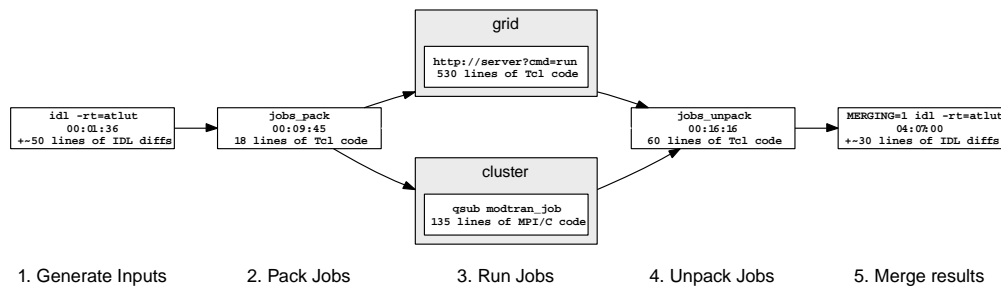


Figure 4.1: Stages of Processing

4.2 Method

The stated goals of this study include: the parallel execution of multiple instances of opaque third-party software, the consistency of numeric results in light of a heterogeneous compute environment, the ability to make use of otherwise idle computing resources available to researchers located at multiple institutions, and acceptable total turnaround time. These goals in addition to the parallel decomposition strategy are addressed individually.

4.2.1 High-level granularity of MODTRAN processing

It is tempting to consider making changes directly to the MODTRAN code itself, since only a minor number of input parameters are to be changed between executions. It is likely that common initialization code need be run only once and could thereafter be shared among multiple calls to only the subroutines affected by the changed parameter. However, in a multi-decade project like MODTRAN, the cost of software maintenance becomes a dominating factor to be considered in the development of any derivative solution. As mentioned in Section 4.1, this invasive approach was once taken [Wang et al., 2002], resulting in presumably positive short-term benefits, but of more questionable benefit in the long run, due to the changes not being merged into the official code. In this case, it can become infeasible to merge such changes after even a single follow-on release, especially when code restructuring is performed. Even the reformatting of comments, as occurred between two releases of the MODTRAN 4 series on its approximately 80,000 lines of code, can make externally tracked sets of changes very difficult to re-apply.

Therefore, the less invasive approach of treating “standard” MODTRAN as a “black box” and achieving parallelism at this higher level of granularity seems attractive – at least when large numbers of independent runs are expected. As mentioned in [Schläpfer and Nieke, 2005], there still exist complex usages of MODTRAN (e.g. high resolution at-sensor simulations with both DISORT and correlated-k features enabled), which lead to run times measured in hours for a single execution. This mode of operation could still benefit from the single-run parallelism described above in [Wang et al., 2002]. However, typical executions such as those used for the ATCOR LUT each require less than a minute on current hardware, making a single-execution parallel version less important than it once was.

4.2.2 Parallel Decomposition

The parameter space of the ATCOR LUT is chosen to adequately cover the problem domain at enough resolution such that interpolation of values between entry samples introduces limited error as compared with directly computed results. The six parameters that are varied are altitude, ranging from 1000 to 99000m; water vapour, from 0.4 to 2.9 g/cm²; 4 built-in MODTRAN aerosol models, from rural to desert; ground elevation, from 0 to 2.5 km; solar zenith angles, from 0 to 70 degrees; and visibility, from 5 to 120 km. These parameters are computed using 45,056 executions of MODTRAN – each performing six runs. The large number of data points is required to cover the wide range of geometry and weather conditions with a narrow grid enabling a linear interpolation in 6D space. This first-order interpolation is similar to the well-known bilinear interpolation in 2D, but neglects the mixed multilinear terms.

ATCOR's sequential Interactive Data Language (IDL) [Stern, 2005] code for generating the LUT consists of a single thread of execution that uses nested loops over the LUT dimensions to generate an input file, execute MODTRAN, parse the results and merge them into a data compressed binary data structure. Due to task independence, modifications for parallelizing this work involved only minor coding changes (see individual boxes in Figure 4.1) to split the single thread of execution into two separate passes – one for input file generation and one for output merging. The resulting two pass process is broken into five stages described below and summarized in Figure 4.1:

- 1. Generate Inputs** The original sequential code is run in two passes: a generation pass and a merging pass. This is done simply by guarding blocks of code with “if” statements. Additionally, a small amount of driver code was written to automatically permute all cases that were previously input from a graphical user interface.
- 2. Pack Jobs** A small script is used to bundle the generated input files into a collection of jobs organized for grid/cluster processing in a way that allows the data to be later re-arranged back to its original organization.
- 3. Run Jobs** The jobs are run in parallel either on a Message Passing Interface (MPI)-based [MPI, 2005] cluster or on the grid.
- 4. Unpack Jobs** A script is used to perform the re-arrangement as described in step 2 in preparation for the merging phase of the original sequential program.
- 5. Merge Results** In the 2nd pass, the original sequential code is re-run with a switch enabling the alternate (merging) blocks of code not taken during the step 1. A minor modification was also added to check for platform-related numerical problems discussed above.

4.2.3 Numeric Consistency Evaluation

Scientific programs such as MODTRAN are susceptible to numerical issues resulting from floating point calculations [Goldberg, 1991] and their inherent complexity [Barhen et al., 2004]. The additional problems introduced due to heterogeneous computing environments have also been well-reported [Blackford et al., 1997]. These include, but are not limited to, differences in: floating-point hardware parameters – which can influence results even in light of full IEEE-754 [IEEE 754, 1985] compliance; operating system, compiler and language runtime library interfaces, which have the potential both to degrade results by masking exceptions or neglecting to save/restore parameters across calls, or to improve results by working around hardware problems in software; issues with application algorithmic integrity, such as the absence of proper scaling to avoid harmful underflow or overflow conditions; and finally, the communication of floating-point values. At least on the latter point, MODTRAN produces its results as tables of

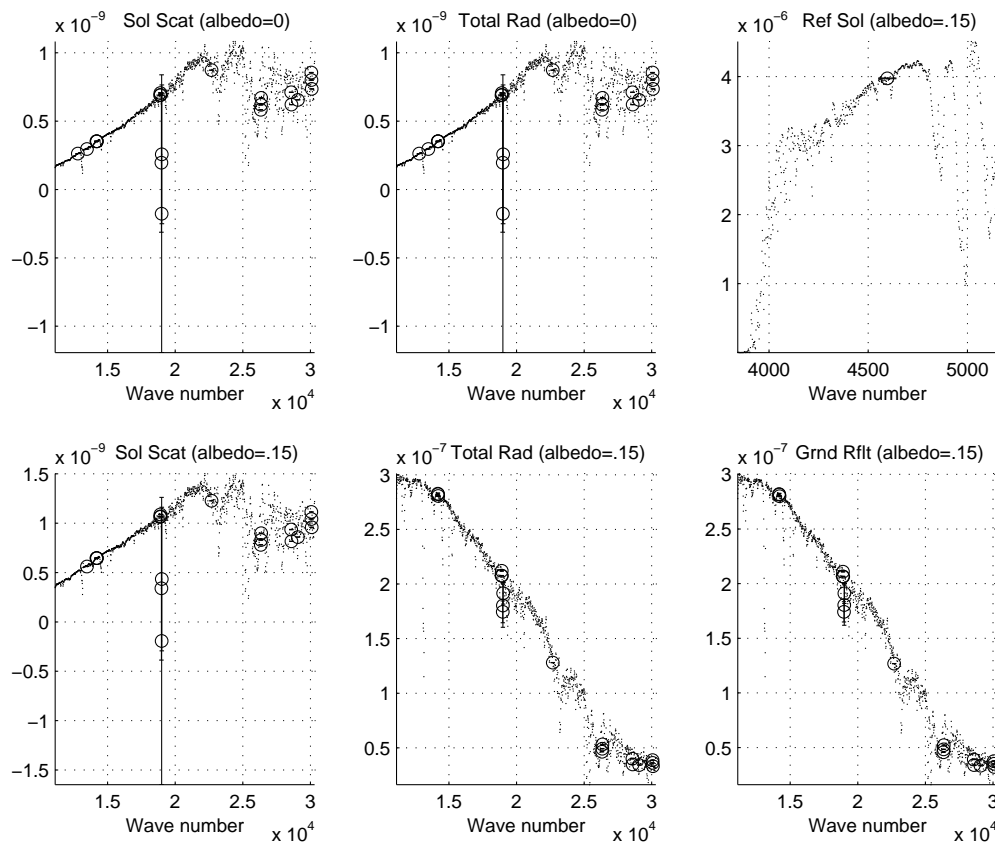


Figure 4.2: Numerics problems with case 4,332 of 45,056: circles/error bars show where any version differs 0.1% from mean of MODTRAN results produced from five common Fortran compiler/version combinations on an AMD Opteron. Note the negative values in Solar Scattering and Total Radiance (produced by two versions of a compiler from a single vendor)

floating point numbers in ASCII format – thereby reducing byte-ordering and binary floating-point representation issues that might arise as data pass through multiple systems.

The apparent effect of some of the aforementioned issues were encountered in the first test run even on the homogeneous cluster – in the case corresponding to input parameters altitude 2,000m, 0.4 g/cm² water vapour, urban aerosol model, visibility 120km, solar zenith angle 30° and ground elevation 1,900m. The computed results produced sporadic physically meaningless negative values for solar scattering and radiances. After this particular case was cross-checked on other hardware architectures without problem, it was decided to experiment with all five different compiler combinations (GNU's g77 3.3.4 and g77 3.4.4 [GNU Fortran, 2005], the Portland Group's pgf77 5.2-4 [Portland Group Fortran, 2005], and Intel's ifort 8.0, and ifort 8.1 [Intel Fortran, 2005]) available to the authors on the target platform, using only the default optimization (-O) configuration.

For this case, only MODTRAN executables compiled using the Intel compilers produced negative results, but other differences were also seen as shown in Figure 4.2.

Since no investigation was made into whether the problem is due to a compiler/run-time error or a MODTRAN coding error, it was decided to universally use the GNU Fortran compiler, since it produced usable results over all cases on the cluster platform and was assumed to produce the most consistent results across the heterogeneous grid platforms – even though GNU Fortran generated code was up to 15% slower than executables produced by other compilers.

Due to this experience, it was furthermore decided to make the extra effort to roughly quantify numerical consistency across the multiple platforms. Therefore all cases were eventually computed on all platforms. Since it is not known which one provides “true” results, relative deviations [Weisstein, 2005] for each of the 45,056 cases are computed across the four platforms.

4.2.4 Runtime Performance Evaluation

Although runtime performance is not the highest priority of this study, careful analysis of the results can still be constructive. Therefore, in order to support comparison of runtime execution, the entire ATCOR LUT as computed from 45,056 calls to MODTRAN 4 Version 3 Revision 1 was generated multiple times using both a homogeneous cluster and a heterogeneous grid, varying the number of CPUs in powers of two up to 64. The Linux-based Matterhorn cluster [Godknecht and Bolliger, 2004] consists of 522 AMD Opteron 244 (1.8Mhz) CPUs connected with 100 Mbit Ethernet or Myrinet. The grid was composed of the personal workstations of the authors and their colleagues and the public Internet was used for network transport. A total of 99 CPU grid resources were available to the authors, including Intel x86-compatible Linux nodes, AMD 64-based Linux nodes, Apple G5-based MacOSX nodes, and Sparc-based Solaris nodes. The slowest grid CPU (a Sun SPARCstation-5) runs MODTRAN 72.4 times slower (20 minutes, 30 seconds vs. 17 seconds) than the fastest grid CPU (an AMD Athlon 64 3500+) when running a particular single case (see Figure 4.4).

4.2.5 Grid versus Cluster

The clearest choice for new implementations of embarrassingly parallel problems intended for execution on a commodity cluster is the use of the MPI (Message Passing Interface) programming library [MPI, 2005] due to its ubiquity.

In contrast, for new grid implementations of embarrassingly parallel problems, there are several possibilities with varying features and trade-offs, but no clear choice for all cases has

yet emerged. Some environments focus on the client end-user experience ([Anderson, 2004], via FoldingHome and SETIHome), and others focus on integration into a fully global network [Foster and Kesselman, 1997]. Perhaps the most well known platform for the general scientific community is Condor [Litzkow et al., 1988]. Because of this, Condor was the initial selection for representing a typical grid implementation for this work.

4.2.6 Condor vs Low Fat Grid

Since the simple batched queuing of multiple executions of a single program with varying input parameters is a common use case, Condor covers this explicitly in its users manual, with examples using the `initialdir` and `queue` parameters. However, after initial Condor installation, setup, and tests were performed, the authors wished to collectively add all spare machines to which they had access to a common processing pool. This is where the first problem with Condor was encountered – that it doesn't directly support pooling processors across firewalls. After discovering only complicated workarounds [Beckles et al., 2005], which still didn't solve the problem that none of our potential servers allowed incoming connections on non-HTTP ports, and all of our potential compute nodes only allowed outgoing connections on the HTTP port, our query on the Condor-users mailing list confirmed that the only potential solutions involved complex tunnelling over ssh and/or requiring several intermediary Condor Generic Connection Broker (GCB—) nodes.

Additionally, a second problem with Condor was encountered. When tests were performed on a subset of the problem, overall run-time results were worse than expected. Investigation revealed that Condor doesn't hide that it is designed for high-throughput but not low-latency. For example, there is a hard-coded (not end-user configurable) pause between each job invocation, which in our case represents a non-trivial percentage of the run time of a single MODTRAN execution on our fastest processors. Although there exists a third-party extension to Condor [Palatin and Kliot, 2003] which aims to improve this, the authors determined that a simple HTTP-based [HTTP, 2005] grid could solve both problems that were encountered. When such a simple tool could not be found, we implemented our own.

The initial prototype, ghack [Brazile, 2005], consists of three short standalone programs (< 500 total source lines of code (SLOC)) capable of enabling geographically disperse groups, with only web browsing privileges and access to a single commodity ISP account, to pool their machines toward working on collective problems. The two primary design goals were ease-of-use (e.g. non-professional programmers) and ease-of-deployment (e.g. aside from write access to a CGI-script capable directory on a web server, no system administrator support, elevated system privileges, or firewall configuration is necessary.) A zip archive containing a collection of per-job directories of input files is HTTP file-uploaded to an Internet accessible web server from which worker nodes (possibly at different institutions) voluntarily request jobs when they are idle. The worker nodes use the same protocols as web-based email programs to download input files and, after processing, upload results. End-users can visit a simple status page to monitor operation. This software enables volunteers to informally collaborate in an ad-hoc resource-sharing grid that can be realized on the order of hours rather than the more typical weeks or months required when more formal coordination is involved.

Since performance wasn't a primary objective for our HTTP-based tool, it was surprising to observe that that in a test within a single institution (the only possibility for standard Condor) involving 1/16th of the data set, a grid of 16 compute nodes ran in half the time needed for Condor.

This success of the initial prototype prompted its re-implementation reduced to two standalone programs and the use of the Representational State Transfer (REST) [Fielding, 2000] web ser-

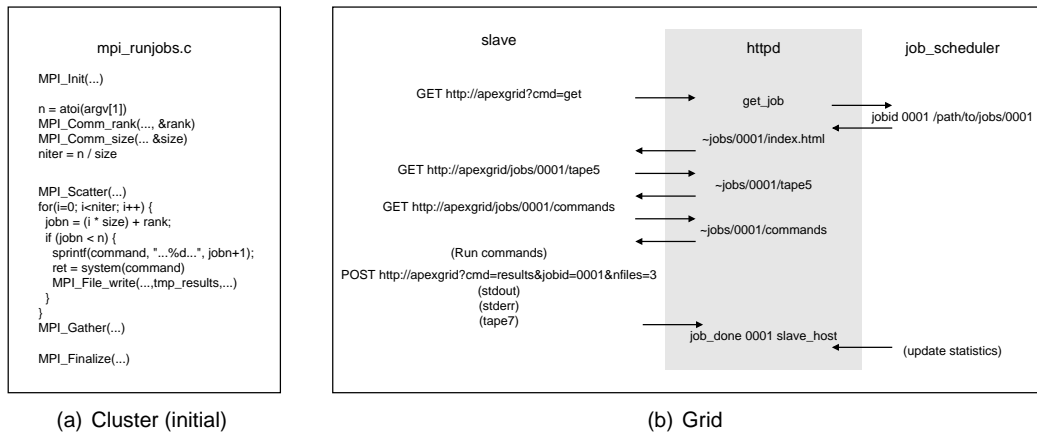


Figure 4.3: Cluster vs. Grid program flow

vice architecture, for an even simpler-to-deploy, more flexible solution known as the *low fat grid* project [Brazile, 2007], [Brazile, 2006].

4.2.7 Scheduling

The scheduling of nearly identical jobs on a homogeneous cluster is not a difficult problem and initially a simple algorithm (shown in Figure 4.3(a)) was chosen to evenly distribute jobs among all processors.

Scheduling nearly identical jobs for a heterogeneous grid is not as trivial. Not only do times vary due to CPU capacity but also due to disk and network latency and bandwidth, especially when using HTTP (or HTTPS when security is desired or required) over the public Internet.

For the grid, a simple scheduling algorithm was chosen whereby the nodes themselves request jobs from the job server. This automatically distributes more jobs to faster nodes. The resulting initial program flow for both cases can be seen in Figure 4.3.

After the first cluster run, it was noted that the homogeneous compute nodes finished at unexpectedly different times – with the “fastest” node finishing several hours earlier than the “slowest” – amounting to about 10% of the total execution time in this case. After searching for and not finding any coding errors, it was decided to simply use the same scheduling algorithm as performed with the low fat grid. The cluster nodes request jobs when they are ready, leading to efficient self-adjusting job distribution.

In order to obtain a rough estimation of total run time on the heterogeneous grid, a single run of MODTRAN was timed on all available grid hosts and a speedup estimate was made by cumulatively adding nodes ranked from the fastest to the slowest. The results are shown in Figure 4.4. For this problem and set of grid nodes, it was estimated that in the best case, using only 15 grid nodes would be only 2 times slower than using all nodes. Using more of the slow but numerous additionally available CPUs produce ever-dwindling returns.

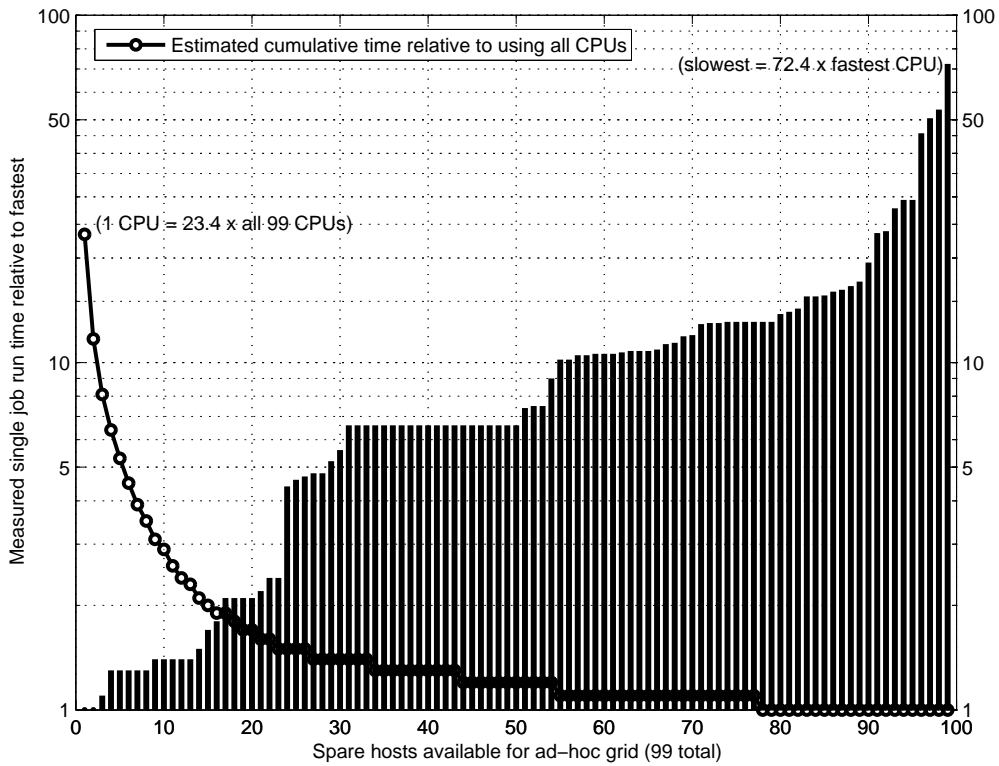


Figure 4.4: Heterogeneous grid runtime estimation – relative CPU speeds of the 99 hosts are plotted along with an estimation of total runtime for a grid composed of increasingly more CPUs. It is estimated that 15 CPUs would run only 2 times slower than using all 99

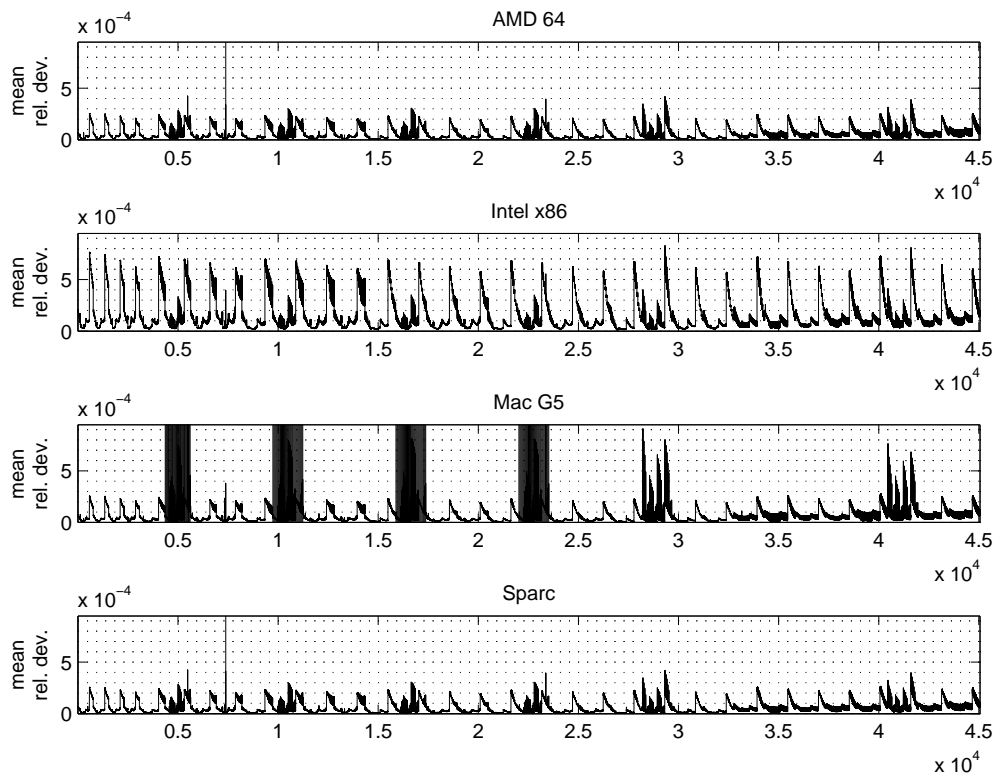


Figure 4.5: Mean relative deviation of the per-case “profile” for each of the four platforms across all 45,056 MODTRAN executions – vertical lines denote cases with bad (e.g. negative) values

4.3 Results

Numerical differences of the GNU-compiled MODTRAN-generated results on the four different platforms are characterized in Figure 4.5 by mean relative deviations [Weisstein, 2005] of a per-case “profile” across all 45,056 cases. The “profile” for a case, which is only used for comparison of that case with the other platforms, is defined as the mean of the subset of only those component radiance values that are eventually recorded in the resulting LUT. All cases which produced bad i.e., negative, results (see Section 4.2.3) are marked by vertical lines.

Total running times for the full computation are summarized in Table 4.1.

For characterizing per-node statistics, the mean and standard deviation of all MODTRAN run times (as measured from the node's point of view) were computed for each node that was used in any of the cluster or grid runs. These are shown in Figure 4.6. Cluster nodes are named with the pattern `nodeNNNN` – the others are grid nodes.

Additionally, for analyzing per-case statistics, minimum, mean and maximum run times (measured from the node's point of view) were computed for each of the 45,056 MODTRAN cases. In Figure 4.7, a quadratic fit of both the grid node and cluster node mean times are plotted, as well as the particular minimum, mean, and maximum times of the .05% most anomalous cases. A quadratic fit was deemed appropriate enough to show the trend, while reducing plot clutter enough to allow displaying of anomalous cases. Showing more anomalous cases would be difficult to read in the plot and would not reveal additional relevant information.

The speedup and efficiency of the parallel portion of the code (e.g. Stage 3 described in Figure 4.1) is plotted in Figure 4.8 according to a standard algorithm [Demmel, 2005].

4.4 Discussion

4.4.1 High-level granularity of MODTRAN

Although it cannot be known what performance effect would have resulted from applying parallelization at a lower level of granularity i.e., within MODTRAN itself, according to Table 4.1, a sufficiently fast turnaround time is clearly achievable using either of the aforementioned grid or cluster approaches on current hardware. With continuous hardware improvements, this situation is only expected to improve.

(a) Sequential Stages			(b) Parallel Stage (Stage 3)		
Stage #	Time	Description	# CPUs	Grid	Cluster
1	0.03	Generate Inputs	2	113.87	125.09 †
2	0.16	Pack Jobs	4	58.70	59.29 †
3	<i>see (b)</i>	Run Jobs	8	36.64	31.26
4	0.27	Unpack Jobs	16	21.53	16.19
5	4.12	Merge results	32	15.33	7.68
			64	12.95	3.93

Table 4.1: Total running times in hours († See section 4.4.3)

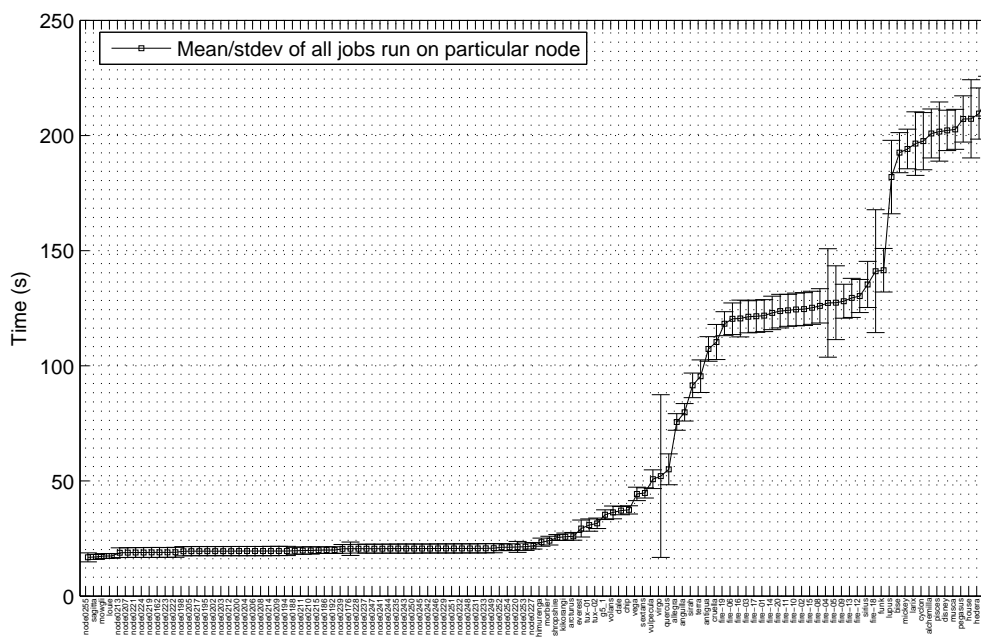


Figure 4.6: Mean and standard deviation of run times per node

4.4.2 Numeric Consistency Evaluation

The most important information depicted in Figure 4.5 is that other than the large number of cases where the Mac G5 produces bad (negative) values, the numerical results do not differ unreasonably from one another. The single 32-bit platform (Intel x86 compatible) was the only platform to produce no negative results, yet its results deviated the most from the other (64-bit) platforms, which showed consistent agreement. It is not known why the Mac G5 resulted in so many bad cases, but an attempt to use a more recent version of the compiler (GNU Fortran version 4.0) resulted in a compiler crash when attempting to build MODTRAN.

The most comforting outcome is that the AMD 64 and SPARC results are almost identical. Even the single “bad” case (7,386 – corresponding to altitude 2,000m, 2.9 g/cm² water vapour, maritime aerosol model, visibility 15km, solar zenith angle 60° and ground elevation 1,500m) for both platforms shows agreement where the AMD 64 produces only one negative value (at wavenumber 17,115 cm⁻¹) and the SPARC produced three (the same as the AMD 64 case but additionally at neighbouring wavenumbers 17,110 cm⁻¹ and 17,130 cm⁻¹). Yet, for unknown reasons, neither the Intel 86 nor the Mac G5 produced bad values for this case.

4.4.3 The Role of Middleware

Interesting questions arise regarding the role of distributed and/or parallel computing middleware for implementing embarrassingly parallel problems. A large amount of services are offered, including: support for problem decomposition, problem characterization analysis, fine-grained monitoring, ease of deployment, ability to access large pools of resources, maximum configurability, exotic scheduling/rescheduling possibilities, high-throughput, low-latency, etc.

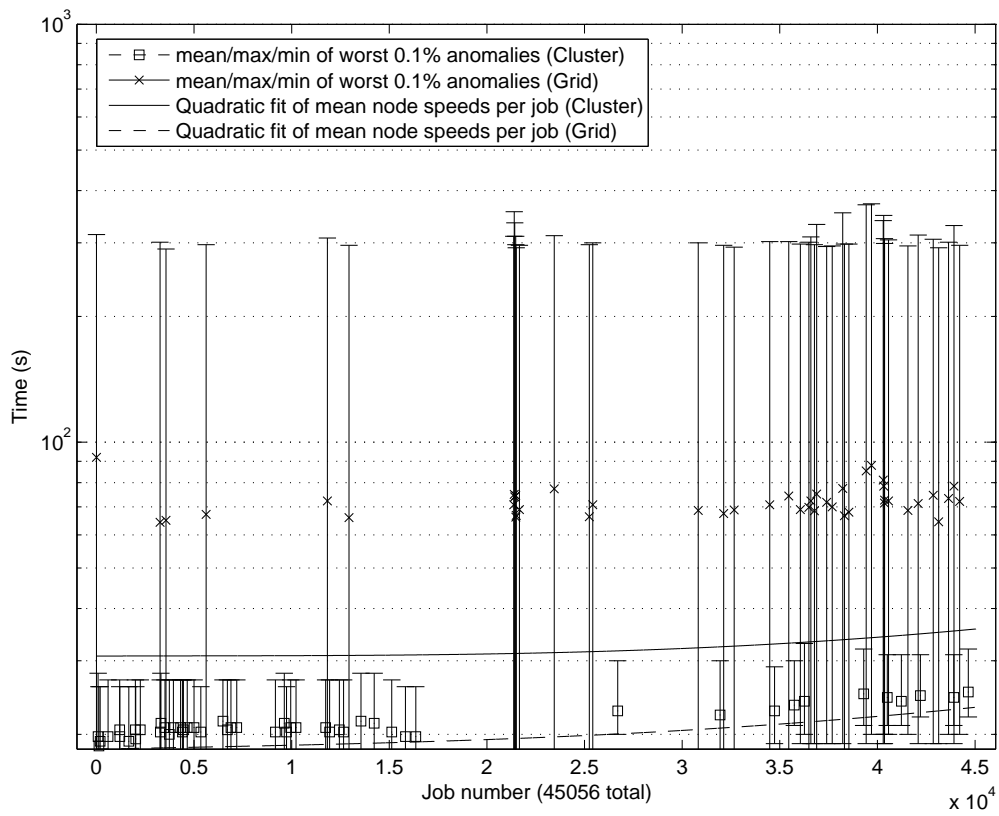


Figure 4.7: Quadratic fit of node speeds (s) per job and extreme anomalies

In this study, which we consider to be a medium sized problem, we were primarily interested in two orthogonal questions: how does one achieve the fastest overall turnaround and how does one get the task implemented with the least amount of effort? To artificially allow for the comparison between cluster and grid, we performed our own problem decomposition and merging of jobs and expected to only make use of queuing/scheduling/execution functionality. However, this effort would not have been much different if we needed to implement only one of the two solutions.

The answers we obtained to these two questions were surprising. Although Table 4.1 shows total run times which imply clearly higher potential turnaround on the cluster, it does not show *effective* turnaround times. In the cluster case, our centralized computing center's job queuing system is configured in a way that penalizes two and four CPU cases due to expected run time requiring more than the 48-hour short-job queue time cut-off. The 32 and 64 CPU cases are penalized due to needing so many CPUs simultaneously. During this study, such jobs sat in the waiting queue for up to two months on multiple occasions before being killed due to cluster maintenance or system failure. In practice, end-users learn to get maximum job throughput by manually partitioning problems into single CPU jobs running just under the 48-hour priority queue cut-off time and manually scheduling these themselves in ways that avoid job quantity penalties imposed by the queuing system. This causes the effect described above of starving jobs that weren't so optimized to the queuing/scheduling middleware. Although such optimizations could potentially have been performed because our problem is embarrassingly parallel, for this study it was desired to accurately characterize cluster operation at specific CPU sizes, and therefore these techniques were avoided in running ATCOR LUT jobs. Over the course of a year, two cluster maintenance days were encountered that serendipitously allowed the author to be among the first to submit new jobs after a cluster reboot, allowing the 32 and 64 CPU jobs to finally run. This was unfortunately not possible for the two and four CPU cluster runs, which is why the timing results for these cases are simulated by concatenating the results of the entire job split into eight equally sized partitions.

In contrast, none of the *low fat grid* cases ever required waiting longer than the upcoming weekend, which was done as a courtesy to the workstation owners.

Therefore, since cluster utilization was high and the cluster middleware was centrally tuned for a workload different from the natural workload of our problem, the fastest overall turnaround for ATCOR LUT generation was achieved using the grid.

Additionally, not including implementing the grid middleware ourselves in < 500 source lines of code (SLOC), the least amount of realization effort was needed for the grid since implementation merely involved pre-generating varying input parameters to be placed one per directory in a zip archive, and convincing colleagues to run a small standalone client program on otherwise idle workstations.

4.4.4 Runtime Performance Evaluation

The speedup/efficiency plot in Figure 4.8 for the parallel-only portion of the cluster runs is as expected for an "embarrassingly parallel" task. The plot of the grid result is more interesting, showing that speedup clearly diverges from ideal starting with 8 CPUs, and that in this setup, there is no good reason to use 32 CPUs over 16 – which largely validates the grid estimation predicted in Figure 4.4.

The total running times in Table 4.1 show that for small numbers of CPUs (up to four), the grid can outperform the cluster. All of the grid CPUs used in these runs were newer than the CPUs in the homogeneous cluster, so this merely reflects Moore's law [Moore, 1965], which implies

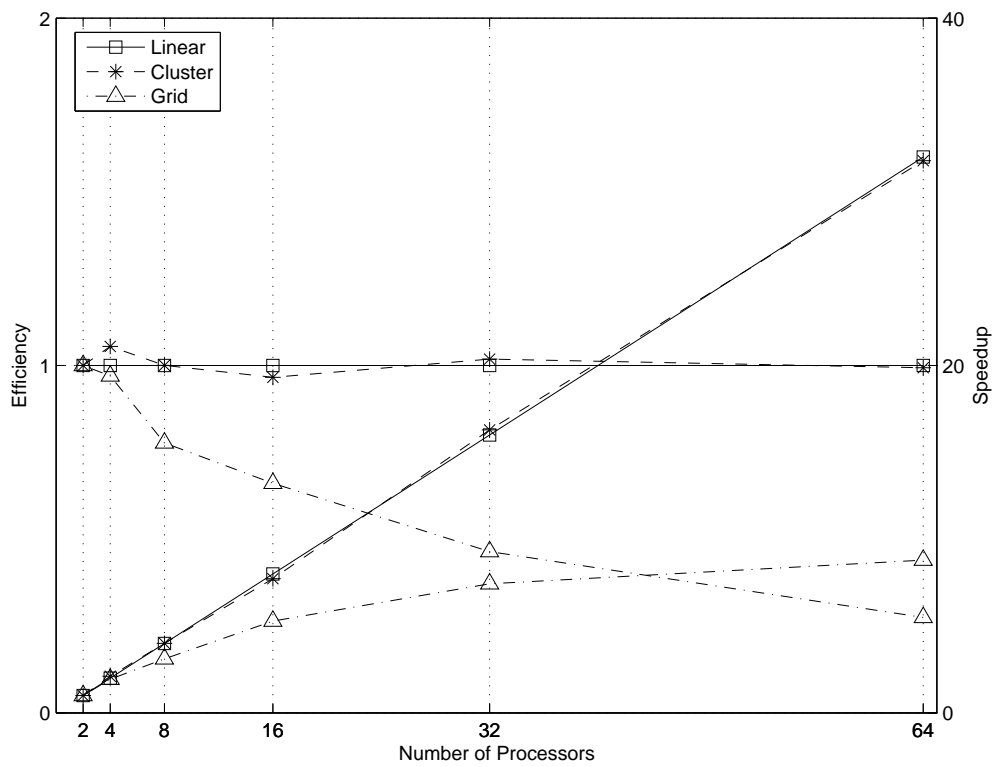


Figure 4.8: Speedup/Efficiency of parallel MODTRAN jobs on cluster and grid

that commodity computational power doubles every 18 months. The difference between the two systems starts to clearly diverge after eight nodes. This is mostly due to there being fewer "new" CPUs available for a grid but it is also due to the increasing affect of the inefficiency of HTTP over the public Internet as the grid's network transfer protocol. In an after-the-fact investigation of possibilities for minimizing the effects of this inefficiency, a few experimental modifications were explored. First, a test run was performed using a 16 (homogeneous) node grid against a server running each of four different freely available web server programs in their standard configurations. On a subset of the parallel-only portion of the computation, this showed average multiple-trial run times ranging from 3.7 to 4.5 hours, with the ubiquitous Apache software performing best. It was observed that at least one of the competing offerings didn't support concurrent CGI processes in its standard configuration, which could explain the larger than expected spread. Another possibility for reducing the inefficiency of HTTP over the public Internet was investigated through the use of data compression. Again, a subset of only the parallel portion of the computation was run on a 16 homogeneous node grid comparing the difference between applying data compression on the output data before transferring the results over the network. In this case, running time was reduced from the 3.7-hour average to an average of 1.3 hours, where the per-job output data size was reduced from about 1MB to about 300KB. Even with only 16 nodes, it appears that reducing the data bandwidth requirement also reduces strains due to concurrency on the server. In our case, input data is small compared to output data. However if the opposite were true, data compression could also be applied to the input. Likewise, if input were to consist of several small files, it is expected that improvement could be seen by using pipelined HTTP download [Percival, 2005] from the server to the grid nodes. In short, there is often potential for reducing the effect of inefficiencies inherent in using HTTP over the public Internet as a grid network transfer protocol.

4.4.5 Per node statistics

There are three noteworthy features in the per node graph (Figure 4.6). First, the single cluster node, named `node0255`, seems to clearly be faster than the others. It has been confirmed by the cluster administrator that this node in fact contains faster CPUs and could be one reason why the original scheduling algorithm described in Section 4.2.7 was anomalous. Second, there is extreme variation shown by the grid node named `virgo`. A closer look at the data revealed the pattern that the slow run times occurred at clusters in time. After consulting the owner of the workstation it was revealed that he also often ran long-running jobs at night and on the weekends, implying that the variation was due to CPU contingency between multiple simultaneous CPU-bound processes. The final striking feature is that the slower grid nodes were not only clearly slower, but often had much larger variation. The fact that they are slower is due to their being SPARC-based CPUs as opposed to the others based on commodity PC hardware. The larger variation is explained in that these are nodes at a remote institution and the inefficiency of the public Internet network transport contributed more to their total run time.

4.4.6 Per job statistics

The quadratic fit of the mean running times per case reveals two items of information. First, there is a consistent per job difference between the mean grid and the mean cluster fitting across the entire workload, as expected. Second, and unexpectedly, the per case running times consistently increase as the case number increases. Examination revealed that the two slowest changing of the six varying MODTRAN parameters are flight altitude and amount of

water vapour. It then makes sense that calculations such as multiple scattering take more computation at higher altitudes when there are more atmospheric layers to be considered.

When examining the extreme anomalies in Figure 4.7, it is first noticeable that the grid variations are so much larger than the cluster variations. This is certainly expected due to the cluster being a closed non-interactive system with dedicated network. In fact, more interesting is that such large variations are possible in a closed-network cluster. This also probably contributed to the anomalous results of the original simple scheduling algorithm described in Section 4.2.7. The biggest variation was due to a Network File System (NFS) server reboot during the running of a job and the smaller variations are also due to NFS contingency during initial input and final output of data files.

The grid anomalies show the extreme effects of intermittent network partitioning on the public Internet - the largest grouping of effects appearing in the early morning as people begin to arrive at work.

4.4.7 Scalability

Assuming that all compute nodes are homogeneous, total job time is dominated by processing time (as opposed to I/O time), all jobs involve roughly the same amount of computation, and that the non-parallel portion of the problem (merging results) is relatively constant and fixed, then scalability is effectively linear since this is an embarrassingly parallel problem.

These assumptions are essentially true in the cluster case but certainly not with the grid, where compute nodes are heterogeneous, total time for any one job might be dominated by I/O for nodes that are far-away or behind slow-haul links. Because of the variability in bandwidth and latency of the internet transport and the differences in profile of fast/slow machines available at any one time for computation, it is impossible to give general scalability results for the grid case. However, for the grid case performed in this study, the level of scalability can be seen in Figures 4.4 and 4.8 and Table 4.1.

4.5 Conclusions

The application of grid and cluster parallelization in executing thousands of runs of third party software with varying input parameters has been investigated, specifically for the generation of ATCOR's LUT, widely used in earth observation applications. The importance of validating the numerical consistency of the results for numerically intensive scientific software such as MODTRAN, especially in light of heterogeneous computing environments, has been shown. In this case study, the potential was also shown for an ad-hoc, voluntary grid to provide faster overall-turnaround time than a better-equipped but highly-utilized and inconveniently-configured cluster.

It has been shown that for producing the MODTRAN based ATCOR LUT, both grid and cluster implementations can be used to reduce runtime from ten days on a single CPU, down to under two days using modest computing resources. When measuring only compute time (i.e. not wall-clock time) cluster-based runs sized above the final merging step runtime threshold, scaled linearly. In a heterogeneous grid using HTTP transport over the public Internet, run-time scalability is far from linear, however the detrimental additional I/O cost can be limited by proper use of web server software and the use of data compression on large input and/or output files.

High quality, free grid and cluster middleware is available, but for a particular set of uses (e.g. implementation by non-programmers, no access to privileged servers or services, execution

across firewalls, etc) the simple *low fat grid* has advantages over the well-known Condor middleware.

When both grid and cluster implementations solve a given embarrassingly parallel problem relatively well, it is worthwhile considering the non-runtime advantages and disadvantages that the two alternatives provide. Clusters allow for higher and more predictable I/O throughput, while grid environments may provide access to (fewer instances of) more recent and therefore higher performing processors. Cluster environments typically offer high quality software compilers and tools affecting numeric accuracy and/or aid in development time, but grid environments, composed of the resources of a circle of collaborative researchers may provide more immediate turnaround of small-to-medium-sized jobs due to lack of resource contingency. Clusters are typically homogeneous which can be an important characteristic for some problems, but for others the variation in platforms usually inherent in a grid can be a benefit, such as in the cross-validation of numeric results.

Also important is the correct granularity of problem decomposition. In this case study, critical third-party core software was treated as a “black box” – even though source code was available. Software evolves over time and it is likely that the cost of maintaining and re-applying locally produced code changes is too high, not because of the cost of initial development but either because the core software might change its structure too much between releases, or because the developers of the local changes are no longer available to re-apply them to a later release.

For prolific software such as the Fortran-based MODTRAN modeler, which may remain in wide use over spans of decades, longevity of parallel/distributed implementation is improved by allowing such decoupled control over job granularity. If the absolute performance of the core software stays roughly the same, the number of jobs that can be run could possibly be doubled on the same number of processors merely by upgrading the hardware. By building this insight into the software driving the “black box” executions, one can increase the useful longevity of the overall system.

Another possible dimension of scalability lies in the number of results generated, which in this case was optimized to fill the capacity of a DVD. As DVD capacities increase, higher numbers of samples can be generated to provide less average error due to inter-sample interpolation. Furthermore, additional dimensions (such as sensor viewing angle, in our case) could be added to the parameter space, in order to enable novel applications.

Acknowledgements

This work was supported in part by ESA/ESTEC contracts 16298/02/NL/US and 15449/01/NL/Sfe. The authors wish to acknowledge the support of the University of Zurich IT Services Department for access to the Matterhorn cluster [Godknecht and Bolliger, 2004]. J. Brazile acknowledges the support of Netcetera AG throughout the period of this work.

References

- [Anderson, 2004] Anderson, D. P. (2004). Boinc: A system for public-resource computing and storage. In Fifth IEEE/ACM International Workshop on Grid Computing, pages 4–10.
- [Anderson et al., 1994] Anderson, G. P., Wang, J., Hoke, M. L., Kneizys, F. X., Chetwynd Jr., J. H., Rothman, L. S., Kimball, L. M., McClatchey, R. A., Shettle, E. P., Clough, S., Gallery, W. O., Abreu, L. W., and Selby, J. E. A. (1994). History of one family of atmospheric radiative transfer codes. In Lynch, D. K., editor, SPIE Passive Infrared Remote Sensing of Clouds and the Atmosphere II, volume 2309, pages 170–183.
- [Barhen et al., 2004] Barhen, J., Protopopescu, V., and Reister, D. B. (2004). Consistent uncertainty reduction in modeling nonlinear systems. SIAM Journal on Scientific Computing, 26(2):653–665.
- [Beckles et al., 2005] Beckles, B., Son, S.-C., and Kewley, J. (2005). Current methods for negotiating firewalls for the Condor system. In Cox, S. and Walker, D. W., editors, Proc. 4th UK e-Science All Hands Meeting.
- [Berk et al., 2005] Berk, A., Anderson, G. P., Acharya, P. K., Bernstein, L. S., Muratov, L., Lee, J., Fox, M., Adler-Golden, S. M., Chetwynd, J. H., Hoke, M. L., Lockwood, R. B., Gardner, J. A., Cooley, T. W., Borel, C. C., and Lewis, P. E. (2005). MODTRAN5: a reformulated atmospheric band model with auxiliary species and practical multiple scattering options. In Shen, S. S. and Lewis, P. E., editors, SPIE Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XI, volume 5806, pages 662–667.
- [Berk et al., 1998] Berk, A., Bernstein, L., Anderson, G., Acharya, P., Robertson, D., Chetwynd, J., and Adler-Golden, S. (1998). MODTRAN cloud and multiple scattering upgrades with applications to AVIRIS. Remote Sensing of Environment, 65(3):367–375.
- [Blackford et al., 1997] Blackford, L., Cleary, A., Petitet, A., Whaley, R., Demmel, J., Dhillon, I., Ren, H., Stanley, K., Dongarra, J., and Hammarling, S. (1997). Practical experience in the numerical dangers of heterogeneous computing. ACM Transactions on Mathematical Software, 23(2):133–147.
- [Brazile, 2005] Brazile, J. (2005). Grid hack. [Online; accessed Sep-2005]. <http://sourceforge.net/projects/ghack/>.
- [Brazile, 2006] Brazile, J. (2006). Low fat grid. [Online; accessed Nov-2006]. <http://code.google.com/p/lowfatgrid/>.
- [Brazile, 2007] Brazile, J. (2007). Low Fat Grid: A RESTful Grid Service for Non-Programmers. In Jazoon '07 – The International Conference on Java Technology, Zurich.

- [Brazile et al., 2006] Brazile, J., Neville, R. A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K. I. (2006). Scene-based spectral response function shape discernibility for the APEX imaging spectrometer. IEEE Geoscience and Remote Sensing Letters, 3:414–418.
- [Demmel, 2005] Demmel, J. (2005). Speedup. [Online; accessed Sep-2005]. <http://www.eecs.berkeley.edu/~demmel/cs267-1995/disc2.html>.
- [Fielding, 2000] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine.
- [Foster and Kesselman, 1997] Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing, 11(2):115–128.
- [GNU Fortran, 2005] GNU Fortran (2005). The GNU Fortran compiler. [Online; accessed Sep-2005]. <http://www.gnu.org/software/fortran/fortran.html>.
- [Godknecht and Bolliger, 2004] Godknecht, A. J. and Bolliger, C. (2004). University of Zurich Matterhorn cluster. [Online; accessed Sep-2004]. <http://www.matterhorn.unizh.ch>.
- [Goldberg, 1991] Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys, 23(1):5–48. <http://citeseer.ist.psu.edu/goldberg91what.html>.
- [Green et al., 2003] Green, R. O., Pavri, B. E., and Chrien, T. G. (2003). On-orbit radiometric and spectral calibration characteristics of EO-1 Hyperion derived with an underflight of AVIRIS and in-situ measurements at Salar de Arizaro, Argentina. IEEE Transactions on Geoscience and Remote Sensing, 41(6):1194–1203.
- [HTTP, 2005] HTTP (2005). Hypertext transfer protocol. [Online; accessed Sep-2005]. <http://www.w3.org/Protocols/>.
- [IEEE 754, 1985] IEEE 754 (1985). ANSI/IEEE standard for binary floating point arithmetic: Standard 754-1985.
- [Intel Fortran, 2005] Intel Fortran (2005). Intel Fortran compiler for Linux. [Online; accessed Sep-2005]. <http://www.intel.com/cd/software/products/asmo-na/eng/compilers/flin/>.
- [Litzkow et al., 1988] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor—a hunter of idle workstations. In The 8th International Conference on Distributed Computing Systems, pages 104–111.
- [Moore, 1965] Moore, G. E. (1965). Cramming more components onto integrated circuits. Electronics, 38(8):114–117.
- [MPI, 2005] MPI (2005). The MPI (Message Passing Interface) Forum. [Online; accessed Sep-2005]. <http://www.mpi-forum.org/>.
- [Neville et al., 2003] Neville, R. A., Sun, L., and Staenz, K. (2003). Detection of spectral line curvature in imaging spectrometer data. In Shen, S. and Lewis, P., editors, SPIE Algorithms and Technologies for Multispectral Hyperspectral and Ultraspectral Imagery IX, volume 5093, pages 144–154, Orlando, FL, USA.
- [Palatin and Kliot, 2003] Palatin, N. and Kliot, G. (2003). Low latency invocation in condor. Technical report, Technion Computer Science Department, Israel. 22 pages.

- [Percival, 2005] Percival, C. (2005). Pipelined http get utility. [Online; accessed Sep-2005]. <http://www.daemonology.net/phttpget/>.
- [Portland Group Fortran, 2005] Portland Group Fortran (2005). The Portland Group compiler products. [Online; accessed Sep-2005]. <http://www.pgroup.com/products/index.htm>.
- [Richter, 2005] Richter, R. (2005). Atmospheric / topographic correction for satellite imagery. Technical Report DLR-IB 565-01/05, DLR, Wessling, Germany.
- [Richter and Schläpfer, 2002] Richter, R. and Schläpfer, D. (2002). Geo-atmospheric processing of airborne imaging spectrometry data. part 2: Atmospheric/topographic correction. International Journal of Remote Sensing, 23(13):2631–2649.
- [Rothman et al., 2003] Rothman, L., Barbe, A., Benner, D. C., Brown, L., Camy-Peyret, C., Carleer, M., Chance, K., Clerbaux, C., Dana, V., Devi, V., Fayt, A., Flaud, J.-M., Gamache, R., Goldman, A., Jacquemart, D., Jucks, K., Lafferty, W., Mandin, J.-Y., Massie, S., Nemtchinov, V., Newnham, D., Perrin, A., Rinsland, C., Schroeder, J., Smith, K., Smith, M., Tang, K., Toth, R., Auwera, J. V., Varanasi, P., and Yoshino, K. (2003). The HITRAN molecular spectroscopic database: edition of 2000 including updates through 2001. Journal of Quantitative Spectroscopy and Radiative Transfer, 82:5–44.
- [Schläpfer and Nieke, 2005] Schläpfer, D. and Nieke, J. (2005). Operational simulation of at sensor radiance sensitivity using the MODO/MODTRAN environment. In Zagojewski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 611–619, Warsaw, Poland.
- [Schläpfer and Richter, 2002] Schläpfer, D. and Richter, R. (2002). Geo-atmospheric processing of airborne imaging spectrometry data part 1: Parametric orthorectification. International Journal of Remote Sensing, 23(13):2609–2630.
- [Schläpfer and Schaepman, 2002] Schläpfer, D. and Schaepman, M. E. (2002). Modeling the noise equivalent radiance requirements of imaging spectrometers based on scientific applications. OSA Journal of Applied Optics, 41:5691–5701.
- [Staenz et al., 1998] Staenz, K., Szeredi, T., and Schwarz, J. (1998). ISDAS - a system for processing/analyzing hyperspectral data. Canadian Journal of Remote Sensing, Vol. 24, No. 2:99–113.
- [Staenz and Williams, 1997] Staenz, K. and Williams, D. J. (1997). Retrieval of surface reflectance from hyperspectral data using a look-up table approach. Can. J. of R. S., Vol. 23, No. 4:354–368.
- [Stern, 2005] Stern, D. (2005). Interactive Data Language. [Online; accessed Sep-2005]. <http://www.rsinc.com>.
- [Wang et al., 2002] Wang, P., Liu, K. Y., Cwik, T., and Green, R. (2002). MODTRAN on supercomputers and parallel computers. Parallel Computing, 28:53–64.
- [Weisstein, 2005] Weisstein, E. W. (2005). Relative deviation. [Online; accessed Sep-2005]. <http://mathworld.wolfram.com/RelativeDeviation.html>.

Chapter 5

Scene-Based Spectral Response Function Shape Discernibility for the APEX Imaging Spectrometer

Brazile, J., Neville, R.A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K.I., Scene-Based Spectral Response Function Shape Discernibility for the APEX Imaging Spectrometer, IEEE Geoscience and Remote Sensing Letters, IEEE, 3, 414–418, 2006.

Reprinted with Permission.

Abstract

Scene-based spectrometer calibration is becoming increasingly interesting due to the decreasing cost of computing resources as compared to laboratory calibration costs. Three of the most important instrument parameters needed for deriving surface reflectance products are per-band bandwidths i.e. full width at half maximum (FWHM), band centers, and spectral response function (SRF) shape. Methods for scene-based bandwidth and band center retrieval based on curve matching in the spectral regions near well-known solar and atmospheric absorption features have been investigated with satisfying results [Gao et al., 2004], [Neville et al., 2003]. The goal of this work is to establish the feasibility of per-band SRF shape discernibility.

To this end, at-sensor radiances in multiple application configurations have been modeled using MODTRAN 4 configured for the currently being built APEX airborne imaging spectrometer [Nieve et al., 2005] in its un-binned configuration (i.e. optimized for spectral resolution). To establish SRF shape discernment feasibility, per-band MODTRAN 4 spectral “filter response function” files have been generated for five common theoretical shapes using APEX nominal bandwidth and band center specifications and are provided as MODTRAN 4 input for the instrument model.

In several application configurations, the typically used gaussian SRF is used as reference and compared with radiances resulting from hypothetical instruments based on the four other shapes in order to detect differences in selected spectral subsets or “windows” near well-known Fraunhofer features. A relative RMS metric is used to show that discernment in some cases is

directly feasible, and in others, feasible if noise reduction techniques (e.g. along track averaging of homogeneous targets) are possible.

5.1 Introduction

For many earth observation applications, remotely sensed spectral imagery is only useful after the derivation of surface reflectance from a given airborne or space borne instrument's radiance measurements. The atmospheric correction involved in this derivation is delicate and greatly affects the accuracy of the resulting spectral reflectance data [Green, 1998]. Often, additional post-processing techniques such as spectral smoothing i.e. spectral polishing [Boardman, 1998] or spatial averaging are used (if possible due to target homogeneity) to improve spectral accuracy at the cost of spatial resolution.

However, investigations have shown that it is possible to improve the original derivation of surface reflectance by improving the accuracy of the instrument characteristics given as input to this process [Neville et al., 2003].

Among instrument parameters used as input for this calculation are detector bandwidths (given as FWHM) and band centers. Typically, the values fed as input are those that were determined based upon the most recent (or pre-launch) laboratory calibration of the instrument (e.g. via monochromator [Cocks et al., 1998], etalon [Sinclair et al., 2002], or low pressure gas lamps [Milton and Choi, 2004]). However, Gao [Gao et al., 2002] introduced and later enhanced [Gao et al., 2004] an atmospheric/solar feature curve-fitting technique which allows for the refinement of band centers derived from a particular scene recorded by the instrument. Ramon [Ramon et al., 2003] and Casadio [Casadio and Colagrande, 2003] performed similar MERIS calibration based on the O₂ absorption feature. Neville [Neville et al., 2003] also uses a feature-based method that is additionally able to refine bandwidth information and further shows that the improvement in resulting surface reflectance can remove the need for additional spectral smoothing. While the accuracy of these feature matching techniques rely on the correctness of MODTRAN 4 Version 3 Revision 1 [Berk et al., 1998] and its HITRAN-based [Rothman et al., 2003] feature database, so does the atmospheric correction of the scene in general – any underlying inaccuracies would anyway negatively affect the derived surface reflectance.

While accurate band center and bandwidth data are assumed to be the most important of the instrument input characteristics in the surface reflectance derivation process, an additional parameter that can be given are the per-band SRF shapes – in place of the standard practice of assuming strictly gaussian shapes.

The goal of this work is to establish the feasibility of retrieving certain per-band SRF shapes directly from a scene by measuring the effect of notable hypothetical SRFs on radiances modeled under typical situations. The example instrument chosen for this study is the currently being built APEX airborne imaging spectrometer developed within the framework of the European Space Agency's (ESA) funding scheme PRODEX. The aim of APEX is to present an Earth observation platform that enables the reproducible measurement of the radiance field of the terrestrial surface at a local and regional scale as well as acting as simulator, calibrator, and validation experiment fostering imaging spectroscopy application development [Itten et al., 1997], [Schaeppman and Itten, 1998], [Nieke et al., 2005].

Table 5.1: Selected MODTRAN 4 input parameters

target refl.	0.05, 0.1, 0.2, 0.4, 0.8
aerosol model	rural extinction
visibility	5 km, 23 km, 100 km
atmos. model	mid-latitude summer
surface alt.	400 m
sensor alt.	5 km
solar zenith angle	180°
scattering alg.	DISORT (16 iterations)
solar data	Thuillier 2002 [Thuillier et al., 2003]
filter resp. func.	gaussian, bartlett, cosine, welch, box [Weisstein, 2005]

5.2 Method

In order to establish discernment feasibility for typical applications, a range of model parameters were selected to cover multiple target types and multiple aerosol visibilities. To allow the simplification of studying a single surface and target altitude, only Fraunhofer lines were used as the basis for selecting comparative feature windows (i.e. radiance values for bands on either side of a particular feature). Windows surrounding atmospheric (as opposed to solar) features should also be feasible, but results are suspected to be more susceptible to scene-based variation (e.g. in sensor altitude, water vapor content, etc.). Selected MODTRAN 4 input parameters are shown in Table 5.1.

Feature windows for comparison of the 75 MODTRAN 4 cases (5 target reflectances \times 3 visibilities \times 5 SRF shapes) were selected from among the Fraunhofer lines with prominent enough features determined to be useful with current instruments [Neville, 2003]. For each candidate feature, all window sizes ranging from 2–5 bands on each side of the feature were iteratively evaluated in order to choose the “best” window. The window size is then fixed for that particular feature. Iterative window selection allows for tuning the selection of features most suitable for a particular instrument.

Reference filter response functions were arbitrarily selected from standard theoretical models [Weisstein, 2005], and generated by a MATLAB-based implementation [Brazile, 2005] at fixed nominal band centers and FWHMs specified for the APEX instrument. The gaussian shape of SRF filter generator was verified by both reproducing exactly the example `DATA/aviris.flt` AVIRIS filter response file delivered with MODTRAN 4 [Berk et al., 2003] for use as the `CARD 1A3 FLTNAM` parameter and measurement via an independently developed gaussian fitting routine [Blake, 2005].

The bartlett and box functions were chosen for being extreme cases, and the cosine and welch functions were chosen for being similar to each other in order to estimate discernment sensitivity.

The chosen evaluation metric for discernment is relative RMS calculated as follows for a gaussian-based window of radiances, L_G , and a second SRF-based window, L_X :

$$\text{relative RMS (\%)} = \frac{\sqrt{\frac{\sum_{i=1}^n (L_{Gi} - L_{Xi})^2}{n}}}{\overline{L_{Gi}}} \times 100.$$

This metric has the merit of directly implying nominal ¹ SNR values (i.e. $\text{SNR} = 100 / \text{rRMS}$) needed by an instrument to achieve discernment, although in practical application other metrics may have more desirable qualities.

5.3 Results

The results are visualized in Figure 5.1 (for simplicity, at only one of the three visibility values), but the implied nominal instrument SNR requirements for all measurements are shown in Table 5.2.

In the figure, the top row shows six of the well-performing Fraunhofer feature windows (for simplicity, the four candidate SRF-convolved spectra are plotted only at the highest target reflectance of 0.8, whereas the internal plots refer to the entire range), while the left column shows the four SRFs that were compared against the standard gaussian SRF. The internal plots then show the relative RMS at five target reflectances (0.05, 0.1, 0.2, 0.4, and 0.8) for the corresponding feature window and SRF.

In Table 5.2, the information in the rows and columns correspond to the same information in the figure, with the addition of the extra visibility dimension (5, 23, and 100 km). Table entries which do not differ from the one directly above it are denoted with *ditto*.

The most interesting general observations of the results are: the bartlett SRF is generally the least discernible from the gaussian SRF; the **A(O₂)** and **B(O₂)** features seem to have the lowest SNR requirements for discernment; the seemingly very similar cosine and welch SRFs appear to be easily discernible when compared against the gaussian; differing visibility and target reflectance values have mostly minor influences on discernibility; for the APEX instrument most of the best performing feature windows contained only four bands (although some features not shown here did best with five and one with eight bands); and most importantly, SNR requirements for discernment in some cases lie directly within current typical instrument specifications (i.e. without the need to employ signal enhancement post-processing).

5.4 Discussion

When selecting theoretical SRFs for this study, it was assumed that the cosine and welch functions would most closely match the gaussian, due to visual inspection of their shape. Since the bartlett function matches gaussian the most and the box the least, the spectral coverage of the tails of the SRF have proven to be more important for the resulting signal than the area under the curve above the FWHM. Equally surprising was the ease of discernibility between the cosine and welch functions with respect to the gaussian-convolved result. This could also be a result of the same effect - i.e. even though their shapes are similar, the coverage of their tails are clearly different.

Unsurprisingly, the good results obtained from the **A(O₂)** and **B(O₂)** features validate the popularity of these choices for feature-based calibration in the prevailing literature.

It was unknown what effect differing visibility and target reflectance cases would have on SRF discernibility, so the resulting relative invariance shown by the results to these variations have good implications on the simplicity of possible retrieval methods.

¹A sensor-specific noise model is not addressed in this work.

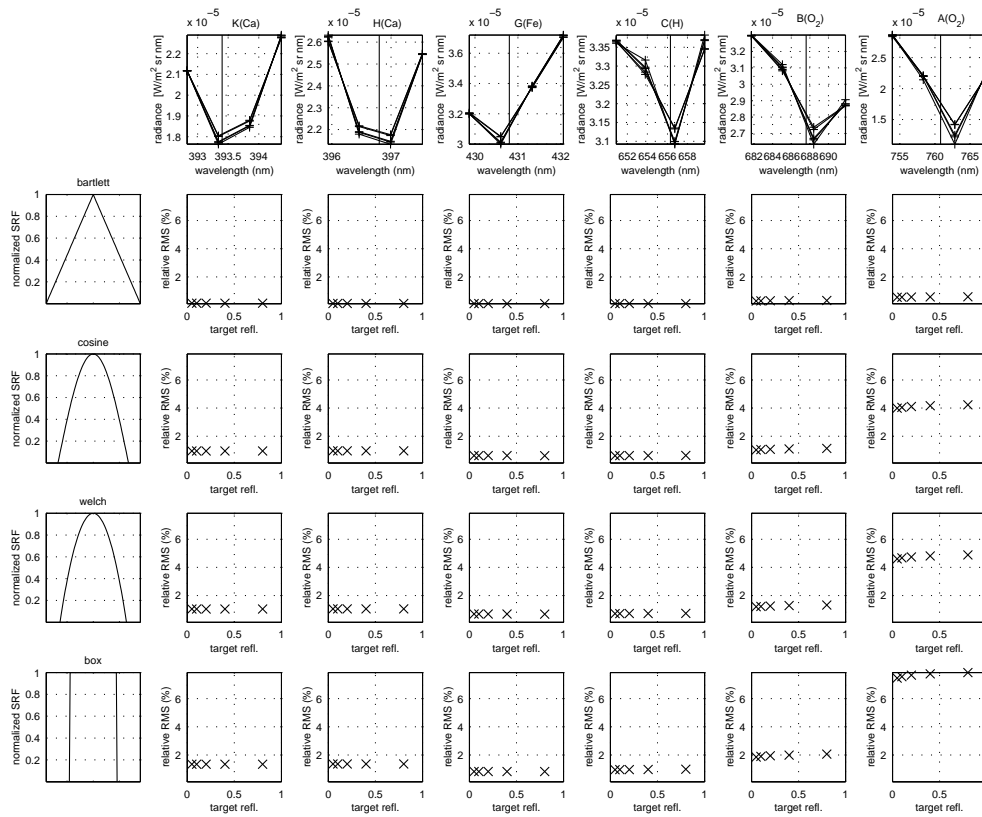


Figure 5.1: RMS (relative to gaussian) of radiances for varying target reflectance per SRF per solar feature for selected cases at 5 km visibility

Table 5.2: Instrument SNR requirements implied from relative RMS between candidate shape and gaussian (see Fig 5.1)

feature	K(Ca)			H(Ca)			G(Fe)			C(H)			B(O ₂)			A(O ₂)			mean	
	5	23	100	5	23	100	5	23	100	5	23	100	5	23	100	5	23	100		
bartlett	0.05	778	778	778	849	849	849	992	992	992	957	960	959	347	378	395	180	187	189	684
	0.1	do.	do.	do.	do.	do.	do.	993	do.	993	951	952	953	338	368	387	178	184	186	
	0.2	do.	do.	do.	do.	do.	848	do.	993	do.	943	945	948	329	360	381	175	182	185	
	0.4	do.	do.	do.	do.	do.	do.	994	994	do.	935	939	943	320	352	375	172	180	184	
cosine	0.05	105	105	105	104	104	104	161	161	161	162	162	162	97	108	114	25	25	25	110
	0.1	do.	do.	do.	do.	do.	do.	do.	do.	do.	161	161	do.	95	105	112	do.	do.	do.	
	0.2	do.	do.	do.	do.	do.	do.	do.	do.	do.	160	do.	161	93	103	111	24	do.	do.	
	0.4	do.	do.	do.	do.	do.	do.	do.	do.	do.	159	160	do.	91	102	109	do.	do.	do.	
welch	0.05	95	95	95	95	95	146	146	146	140	141	141	83	92	97	22	22	22	97	
	0.1	do.	do.	do.	do.	do.	do.	do.	do.	do.	139	140	140	81	90	96	21	do.		do.
	0.2	do.	do.	do.	do.	do.	do.	do.	do.	do.	139	do.	do.	80	88	94	do.	21		do.
	0.4	do.	do.	do.	do.	do.	do.	do.	do.	do.	138	138	139	78	87	93	do.	do.		do.
box	0.05	74	74	74	73	73	121	121	121	105	106	106	54	60	63	13	14	14	74	
	0.1	do.	do.	do.	do.	do.	do.	do.	do.	do.	105	105	do.	53	58	62	do.	do.		do.
	0.2	do.	do.	do.	do.	do.	do.	do.	do.	do.	104	do.	do.	52	57	61	do.	do.		do.
	0.4	do.	do.	do.	do.	do.	do.	do.	do.	do.	103	do.	do.	50	56	60	do.	do.		do.
mean	263			280			355			337			151			60				

Also unpredicted beforehand were the optimal spectral ranges of the feature windows. Since most of them (18 out of 21 analyzed) performed best with the smallest window size – two bands on each side of the window – one can be somewhat confident that the resulting differences were most influenced by the features in question, rather than neighboring features or unknown sources.

The most satisfying results are the implied SNR requirements. None of the SRF's required an SNR more than 1000:1 and most applications (e.g. snow, agriculture, mining, etc.) deal with target scenes that contain large enough homogeneous areas that along-track averaging can be used to easily achieve this performance.

There are, however, a few questions raised by seemingly anomalous results in Table 5.2. The most interesting, from a shape discernment point of view, arises from the A(O₂) column. Why is it that SRF shape discernment is affected both by allowing visibility to vary (up to 8% between the 5 km and 100 km extremes at 0.8 target reflectance) and allowing reflectance to vary (up to 6% between the 0.5 and .8 extremes at 5 km visibility) – but only in the bartlett case and not for any of the other shapes? To investigate this, the full resolution MODTRAN 4 (i.e. `tape7`) output was consulted. As expected, for every group of five cases where the only varying input parameter was the SRF filter, the full resolution `tape7` files were identical – only the channel-specific post-convolution `channels.out` files differed. So for any of these cases, if the comparative difference between gaussian-convolved spectra is greater for one shape than it is for another, then it can only be due to either the MODTRAN 4 implementation of this convolution or, more likely, simply a data-dependent mathematical artifact of either the convolution or the use of the relative RMS as the spectra discernment metric.

Finally, how can it be that for features K(Ca), H(Ca) and G(Fe), relative discernibility between SRF shapes is unaffected when visibility and target reflectance are allowed to vary, yet in the case of the B(O₂) feature at 5 km visibility, bartlett shape discernibility differed by 13% when reflectance was allowed to vary between the .05 and 0.8 extremes? Again, this outcome is directly due to the results produced by the underlying MODTRAN 4 [Berk et al., 1998] and HITRAN [Rothman et al., 2003] models and are beyond this scope of this paper. However, we feel that a maximum deviation in SNR of 13% for a 1600% change in reflectance doesn't present a major hurdle in the development of a useful SRF shape retrieval algorithm.

5.5 Conclusions

The discernibility of four theoretical SRFs from the typically used gaussian, and their respective SNR requirements at various Fraunhofer feature windows has been shown for an instrument modeled on the currently being built APEX [Nieke et al., 2005] imaging spectrometer.

For some feature windows, these SNR values are already within specification of current typical instruments such as APEX. For other feature windows, the required SNR can often be achieved with signal enhancement techniques such as along-track averaging of homogeneous targets.

It is suggested that since two of the examined SRFs were so similar, sensitivity of discernment may yield retrieval of realistic SRFs within a useful resolution. Additionally, iterative feature window analysis allows for tuning the selection of features that are most suitable for a particular instrument.

Ultimately, a band-by-band SRF retrieval method would involve finding as many useful features as possible over an instrument's entire spectral range. It is suggested that in addition to the Fraunhofer features shown here, atmospheric feature windows could be used for this purpose, but their use is more susceptible to per-scene variation [Schlöpfer and Nieke, 2005] which adds

more complexity to such a method. Finally, to cover those bands which are not able to be retrieved directly via Fraunhofer or atmospheric features, per-detector interpolated fitting would be needed.

A spectrum matching-based SRF retrieval method would minimally use a program such as [Brazile, 2005], together with a particular instrument's band center and bandwidth characteristics to pre-generate MODTRAN 4 inputs for a slowly varying look-up table of radiances, indexed by parameterized SRF shapes. Spectrum matching would be performed using some to-be-determined appropriate metric and retrieval of the shape could be performed by reverse mapping the best matching look-up table entry index to its shape. If the parameterized shapes chosen have appropriate characteristics (e.g. continuity), iterative searching might be possible for increasing the accuracy of the retrieved shape. In any case, a shape retrieval method would need to be integrated with band center and bandwidth retrieval methods, as these iterative matching methods are not likely to be strictly independent.

Finally, to compliment investigation on SRF retrieval methods, it is recommended to perform a sensitivity study on the effects of using incorrect SRF shapes in surface reflectance product generation to ensure that such a retrieval is practically worthwhile.

Acknowledgment

J. Brazile thanks the University of Zurich Geography Department and Netcetera AG for supporting the sabbatical allowing on-site collaboration with the Canadian Centre of Remote Sensing, and Felix Seidel, member of the Swiss National Point of Contact (NPOC) team for satellite imagery, for practical tips on modeling with MODTRAN 4.

References

- [Berk et al., 2003] Berk, A., Anderson, G., Acharya, P., Hoke, M., Chetwynd, J., Bernstein, L., Shettle, E., Matthew, M., and Adler-Golden, S. (2003). MODTRAN 4 version 3 revision 1 USER'S MANUAL.
- [Berk et al., 1998] Berk, A., Bernstein, L., Anderson, G., Acharya, P., Robertson, D., Chetwynd, J., and Adler-Golden, S. (1998). MODTRAN cloud and multiple scattering upgrades with applications to AVIRIS. Remote Sensing of Environment, 65(3):367–375.
- [Blake, 2005] Blake, J. (2005). fitgauss. [Online; accessed Jan-2005]. <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=7489>.
- [Boardman, 1998] Boardman, J. W. (1998). Post-ATREM polishing of AVIRIS apparent reflectance data using EFFORT: a lesson in accuracy versus precision. Summaries of the Seventh JPL Airborne Earth Science Workshop, JPL Pub. 97-21:53.
- [Brazile, 2005] Brazile, J. (2005). MODTRAN-compatible SRF generator. [Online; accessed Jan-2005]. <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8723>.
- [Casadio and Colagrande, 2003] Casadio, S. and Colagrande, P. (2003). Meris O₂ calibration using sciamachy measurements. In Proceedings MERIS User Workshop, Frascati, Italy. ESA.
- [Cocks et al., 1998] Cocks, T., Jenssen, R., Stewart, A., Wilson, I., and Shields, T. (1998). The HyMap(TM) airborne hyperspectral sensor: The system, calibration and performance. In 1st EARSeL Workshop on Imaging Spectroscopy, pages 37–42, Zurich, Switzerland.
- [Gao et al., 2002] Gao, B.-C., Montes, M. J., and Davis, C. O. (2002). A curve-fitting technique to improve wavelength calibrations of imaging spectrometer data. In Proceedings of the 11th JPL Airborne Earth Science Workshop, volume 03-4, pages 99–105.
- [Gao et al., 2004] Gao, B.-C., Montes, M. J., and Davis, C. O. (2004). Refinement of wavelength calibrations of hyperspectral imaging data using a spectrum-matching technique. Remote Sensing of Environment, 90:424–433.
- [Green, 1998] Green, R. O. (1998). Spectral calibration requirement for Earth-looking imaging spectrometers in the solar-reflected spectrum. Applied Optics, 37(4):683–690.
- [Itten et al., 1997] Itten, K. I., Schaepman, M., De Vos, L., Hermans, L., Schläepfer, H., and Droz, F. (1997). APEX - airborne prism experiment: A new concept for an airborne imaging spectrometer. In 3rd Intl. Airborne Remote Sensing Conference and Exhibition, volume 1, pages 181–188.

- [Milton and Choi, 2004] Milton, E. J. and Choi, K. Y. (2004). Estimating the spectral response function of the CASI-2. In Annual Conference of the Remote Sensing and Photogrammetry Society, Aberdeen, Scotland. Remote Sensing and Photogrammetry Society, Nottingham, UK.
- [Neville, 2003] Neville, R. A. (2003). Spectral absorption features for use in calibrating imaging spectrometers. Technical Report CHTN_2003_002, Canada Centre for Remote Sensing, Ottawa, Ontario, Canada. 11 pages.
- [Neville et al., 2003] Neville, R. A., Sun, L., and Staenz, K. (2003). Detection of spectral line curvature in imaging spectrometer data. In Shen, S. and Lewis, P., editors, SPIE Algorithms and Technologies for Multispectral Hyperspectral and Ultraspectral Imagery IX, volume 5093, pages 144–154, Orlando, FL, USA.
- [Nieke et al., 2005] Nieke, J., Itten, K., Debruyn, W., Kaiser, J., Schläpfer, D., Brazile, J., Meuleman, K., Kempeneers, P., Neukom, A., Schilliger, T., De Vos, L., Piesbergen, J., Gege, P., Suhr, B., Schaepman, M., Gavira, J., Ulbrich, G., and Meynart, R. (2005). The airborne imaging spectrometer APEX: From concept to realization. In Zagojowski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 67–74, Warsaw, Poland.
- [Ramon et al., 2003] Ramon, D., Santer, R., and Dubuisson, P. (2003). MERIS in-flight spectral calibration in O₂ absorption using surface pressure retrieval. In Huang, H.-L., Lu, D., and Sasano, Y., editors, SPIE Optical Remote Sensing of the Atmosphere and Clouds III, volume 4891, pages 505–514.
- [Rothman et al., 2003] Rothman, L., Barbe, A., Benner, D. C., Brown, L., Camy-Peyret, C., Carleer, M., Chance, K., Clerbaux, C., Dana, V., Devi, V., Fayt, A., Flaud, J.-M., Gamache, R., Goldman, A., Jacquemart, D., Jucks, K., Lafferty, W., Mandin, J.-Y., Massie, S., Nemtchinov, V., Newnham, D., Perrin, A., Rinsland, C., Schroeder, J., Smith, K., Smith, M., Tang, K., Toth, R., Auwera, J. V., Varanasi, P., and Yoshino, K. (2003). The HITRAN molecular spectroscopic database: edition of 2000 including updates through 2001. Journal of Quantitative Spectroscopy and Radiative Transfer, 82:5–44.
- [Schaepman and Itten, 1998] Schaepman, M. and Itten, K. (1998). APEX-airborne PRISM experiment: An airborne imaging spectrometer serving as a precursor instrument of the future ESA land surface processes and interactions mission. In ISPRS Commission VII Symposium on Resource and Environmental Monitoring, volume 22(7), pages 31–37, Budapest, Hungary.
- [Schläpfer and Nieke, 2005] Schläpfer, D. and Nieke, J. (2005). Operational simulation of at sensor radiance sensitivity using the MODIS/MODTRAN environment. In Zagojowski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 611–619, Warsaw, Poland.
- [Sinclair et al., 2002] Sinclair, P., Berman, R., Hersom, C., and Hollinger, A. (2002). Spectral calibration of hyperspectral imagers using a white light etalon. In Proceedings of the 12th CASI Conference on Astronautics (ASTRO 2002), Ottawa, Ontario, Canada.
- [Thuillier et al., 2003] Thuillier, G., Hersé, M., Labs, D., Foujols, T., Peetermans, W., Gillotay, D., Simon, P. C., and Mandel, H. (2003). The solar spectral irradiance from 200 to 2400 nm as measured by the SOLSPEC spectrometer from the Atlas and Eureca missions. Solar Physics, 214:1–22.
- [Weisstein, 2005] Weisstein, E. W. (2005). Full width at half maximum. [Online; accessed Jan-2005]. <http://mathworld.wolfram.com/FullWidthatHalfMaximum.html>.

Chapter 6

Toward Scene-Based Retrieval of Spectral Response for Hyperspectral Imagers Using Fraunhofer Features

Brazile, J., Neville, R.A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K.I., Toward Scene-Based Retrieval of Spectral Response Functions for Hyperspectral Imagers Using Fraunhofer Features, Canadian Journal of Remote Sensing, Canadian Aeronautics and Space Institute, 2007, in press.

Abstract

Initial steps are proposed and tested in the development of a method for retrieving/refining instrument spectral characteristics for dispersive hyperspectral imagers such as the Airborne Visible/Infrared Imaging Spectrometer - AVIRIS [Green et al., 1988], CASI [Anger et al., 1996], HyMap [Cocks et al., 1998], Hyperion [Pearlman et al., 2003] and CHRIS [Barnsley et al., 2004] based upon data acquired by the instrument in operation using statistical spectrum matching with MODTRAN-modeled instrument results in the vicinity of reference Fraunhofer feature windows.

Until now, such scene-based retrieval has focused primarily on refining spectral band centre shifts while assuming that spectral response function (SRF) parameters remain static. In particular, most methods assume that the SRF is of a Gaussian shape. As a consequence of recent investigations showing that scene-based discernment of SRF shape should be feasible given current typical instrument performance, this paper explores algorithmic components deemed necessary for the development of a look-up table (LUT) based retrieval method for obtaining SRF parameters on a band-by-band basis, even in the presence of minor band centre or band width deviations from nominal instrument specifications. The proposed method employing these components is appropriate for dispersive hyperspectral imagers but not for others, for example Fourier transform hyperspectral imagers.

In experiments using nominal implementations of the proposed components, reference spectra match expected LUT spectra in nearly all cases, even when band centre and band width deviations are considered. This holds true for all three modelled instruments and nearly all of the six selected Fraunhofer windows. Expected signal-to-noise requirements are in many cases challenging, yet feasible, using signal enhancement techniques such as along-track averaging.

6.1 Introduction

6.1.1 Motivation

As the use of hyperspectral imaging data for a wide range of applications becomes more widespread, expectations increase with regard to the resolution and accuracy of such data. In particular, erroneous spectral calibration – the assignment of spectral wavelengths to the recorded across-track pixel number – can lead to large errors in the resulting generated surface reflectance products required by a majority of earth observation applications. Spectral calibration is performed in the laboratory either pre-launch as with the Hyperion spaceborne instrument [Pearlman et al., 2003], or in the case of airborne instruments such as the Airborne Visible/Infrared Imaging Spectrometer - AVIRIS [Green et al., 1988], multiple calibrations are possible, for example between flight seasons. Laboratory calibration typically involves using a monochromator to scan across the focal plane in sub-nanometer steps [Cocks et al., 1998]. Another method involves using a tunable etalon filter allowing multiple measurements to be taken simultaneously, enabling calibration over larger portions of the detector array and with different look angles [Sinclair et al., 2002]. More recently, a laboratory-based SRF estimation method is described using the same low pressure gas lamps used in the instrument's band centre calibration [Milton and Choi, 2004].

There are various reasons why scene-based calibration is desirable. When used to refine traditional laboratory characterization, it can help determine conditions when re-characterization or re-calibration is called for, such as instrument deterioration over time or deployment in new environments. For some characterizations where results are deemed effective enough, it could be used in lieu of more traditional methods. Laboratory calibration is very resource intensive. Some instruments, such as the Compact Airborne Spectrographic Imager (casi), allow multiple configurations [Milton, 2006], each preferably with its own characterization. In extreme cases, such as the Airborne Prism EXperiment (APEX) spectrometer [Itten et al., 1997], summing of the instrument's entire set of (511) sub-channels is fully programmable, allowing arbitrary data-take specific configurations [Schaepman and Itten, 1998]. Additionally, increased interest in spectro-directional spectroscopy implies that multi-angular instruments such as the Compact High Resolution Imaging Spectrometer - CHRIS [Barnsley et al., 2004] might become more common, multiplying calibration complexity for each view angle. Finally, perhaps the most attractive argument for scene-based characterization is that it can be performed by anyone in possession of a suitable data set. One can attempt to retrieve or refine instrument characteristics at the time of any particular data-take.

Scene-based detection of instrument characteristics also has disadvantages. The methods employed are typically complicated and might be fragile in untested situations. They can also be computationally intensive, though this may be less costly than deployment and maintenance of traditional laboratory retrieval methods, especially when initial software development and hardware costs are amortized over time. Most critically, scene-based methods have been suspected of producing results inferior to traditional methods. However, the expected trend is that growing confidence as techniques mature will cause this perception to diminish over time.

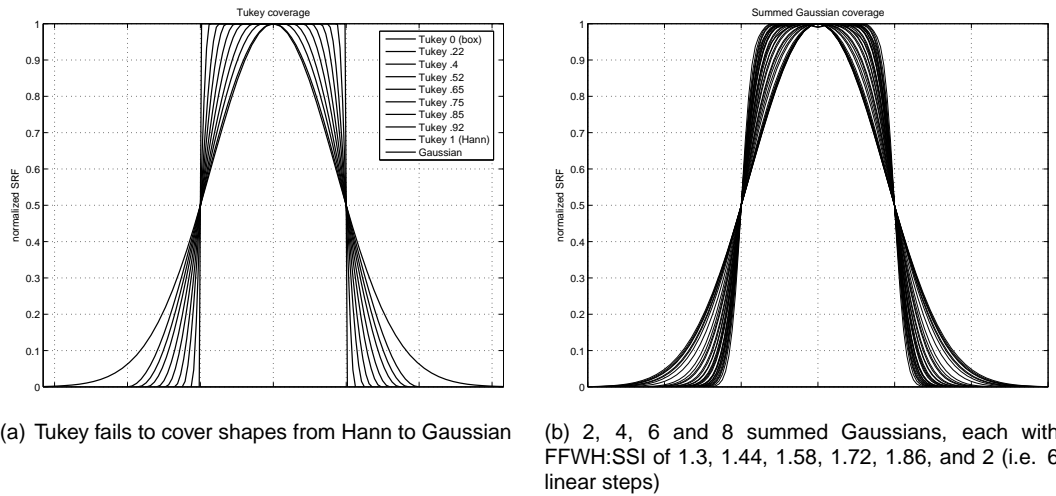


Figure 6.1: Shape coverage of candidate SRF parameterization models.

6.1.2 Scene-based Retrieval

It is becoming increasingly more feasible to retrieve and/or refine instrument characteristics in hyperspectral imaging data by analyzing the data themselves. [Gao et al., 2002] introduced a spectrum matching technique to improve data calibration which was later refined [Gao et al., 2004]. [Ramon et al., 2003] and [Casadio and Colagrande, 2003] performed similar calibration for the spaceborne Medium Resolution Imaging Spectrometer (MERIS), based on the O₂ absorption feature. [Neville et al., 2003] also used a feature-based method specifically for the detection of spectral line curvature, which subsequently inspired a method for the scene-based detection of keystone aberrations [Neville et al., 2004]. While the accuracy of these spectrum matching techniques rely on the correctness of a trusted model, such as that provided by the atmospheric radiative transfer (RT) code MODTRAN 4 [Berk et al., 1998] and its HITRAN-based feature database [Rothman et al., 2003], so does the atmospheric correction of the scene and, therefore, the end product itself. Accordingly, consistency is maintained provided the same trusted model is used for generation of LUT entries for both the atmospheric correction and the characterization via spectrum matching. Further, it is argued that as the trusted model becomes more accurate, so do both instrument characterization and the resulting surface reflectance products.

The most difficult and error prone process in deriving end-user products from remotely sensed earth observation data involves atmospheric correction. Three of the most important instrument parameters used as inputs for atmospheric correction, i.e. for generating surface reflectance products from at-sensor radiance, are per-band band centres, bandwidths (typically characterized as full-width at half-maximum or FWHM) and SRF shape. The aforementioned studies have explored retrieval of band centre and bandwidth parameters. Therefore, the next step is to investigate methods for the scene-based retrieval of per-band SRF shapes.

The theoretic feasibility of discernment of a set of given SRFs from the typically-used Gaussian has been recently established [Brazile et al., 2006] specifically for the case of the APEX imaging spectrometer currently being built [Nieke et al., 2005]. Band-by-band SRF retrieval feasibility was addressed by examining sensitivity of discernment using spectrum matching of several different Fraunhofer feature windows across a wide spectral range, and under varying target reflectances and aerosol visibility conditions. However, this work stopped short of defining a general scene-based per-band SRF retrieval method.

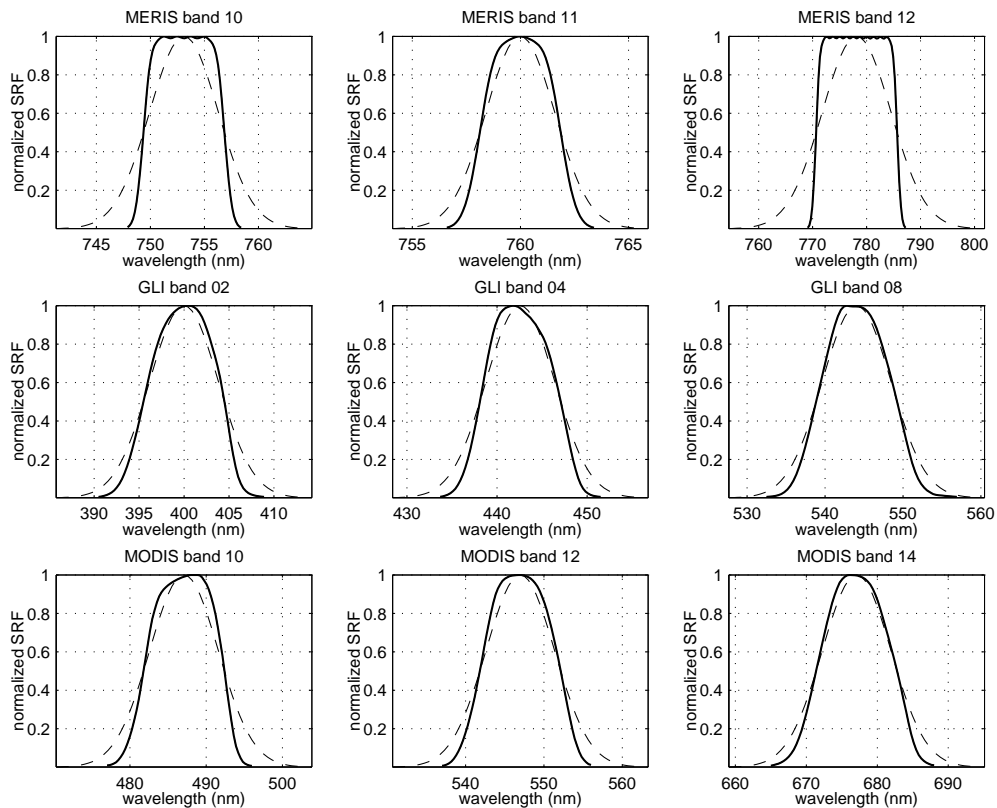


Figure 6.2: Known SRF shapes and their commonly-used Gaussian approximations

The goal of this paper is to extend this work by exploring and validating specific components required for a generally applicable scene-based SRF retrieval method. In particular, these components should include the following:

An SRF parameterization that generates realistic shapes, which is continuous enough to allow retrieval via spectrum matching against discrete LUT entries, should be established.

A robust spectrum matching metric needs to be possible even in the presence of minor shifts in nominal band centre and/or changes in band width specifications.

Spectral Coverage should include a large enough set of Fraunhofer features that are prominent enough to reveal discernable signal differences in the resulting at-sensor radiances throughout an instrument's spectral range.

General Applicability should be established by testing for multiple instruments.

Statistical Consistency should reveal trends in neighboring bands allowing the identification and rejection of outliers.

6.2 Method

The proposed method, consisting of nominal implementations of the proposed components, attempts to simulate the steps needed to match the reference spectra produced from a specific data-take by a specific instrument against a large pre-generated set of trial spectra modeled from that instrument's specifications, but allowing its nominal spectral characteristics to vary. The assumption is that when a good match is found between the reference spectra and one of the pre-generated spectra, then chances are good that the spectral characteristics of the instrument that produced the reference spectra are reflected by the input parameters used for generating the matching LUT entry. It is assumed that interpolation can be used to fill in the gaps between LUT entries, at least in mathematically well-behaved situations.

The process is centered around the characteristic issue appearing both during the forward modeling of the instrument under varying spectral characteristics and indirectly in the inverse modeling of the matching spectra. This is the question of how to model the SRF. First, an attempt was made to define the properties desirable for an SRF model. Initially, it was thought that functions with well-defined formulas for determining properties, such as area or FWHM, would be useful for ease in generation and inter-comparison of SRFs. An earlier study [Brazile et al., 2006] proposed apodization functions [Weisstein, 2005] such as Welch or cosine to fulfill this requirement. However, it became clear that for retrieval purposes, it is more important for an SRF to have a parameterized, smoothly changing nature, which preferably covers a variety of known shapes.

A search of the literature revealed the Tukey function [Harris, 1978], which in its parameterization ranges from equivalence with the boxcar to equivalence with the Hann function (Figure 6.1(a)). Although its continuous nature seems ideal, initial investigation indicated that its Hann extreme did not go far enough toward the typically-used Gaussian shape to cover cases that appear in practice (Figure 6.2).

It was finally decided that an appropriate parameterization can be obtained by summing a number of Gaussians at varying ratios of sub-channel FWHM to spectral sampling interval (SSI). Physically, the summing of Gaussians corresponds to the common practice of summing instrument sub-channels to increase per-band signal-to-noise (SNR) performance. Through examination of the resulting shapes, it was decided to allow the number of summed Gaussians to vary among 1, 2, 4, 6, and 8; and the sub-channel FWHM/SSI ratios to vary in six linear steps: 1.3, 1.44, 1.58, 1.72, 1.86, and 2 (Figure 6.1(b)).

If the SRF of the sub-channels covered by a summed band can be assumed to be a Gaussian function (g_i), then the SRF of the summed band can be expressed by summing up the SRFs of all the covered sub-channels as follows:

$$SRF_{sum}(\lambda) = \sum_{i=1}^N g_i(\lambda) \quad (6.1)$$

$$g_i(\lambda) = e^{-\frac{4 \ln(2) (\lambda - \mu_i^{sub})^2}{FWHM_{sub}^2}} \quad (6.2)$$

where $FWHM_{sub}$ is the FWHM of the sub-channels (assumed to be the same over the given span), μ_i^{sub} is the centre wavelength position of the i^{th} sub-channel relative to the centre wavelength of its corresponding summed band, λ is the sample wavelength, and N is the number of sub-channels. $FWHM_{sub}$ and μ_i^{sub} can be calculated using the following formulas:

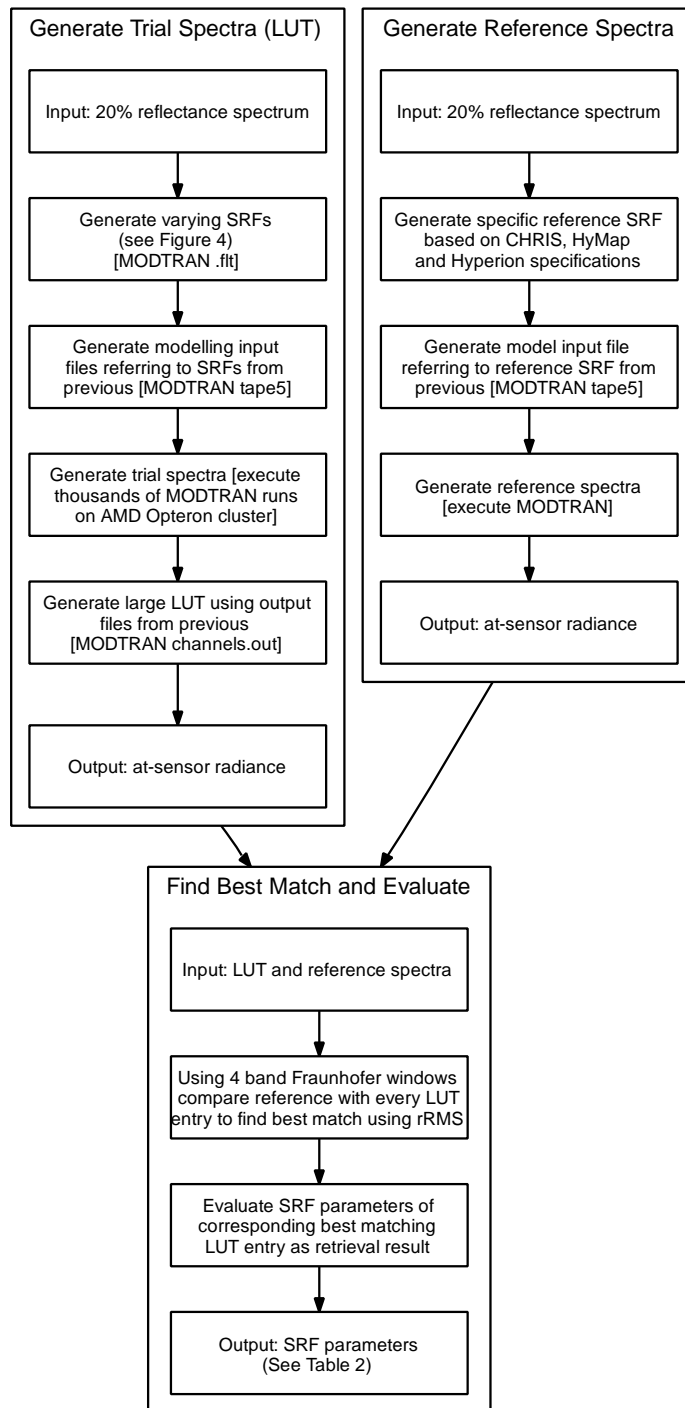


Figure 6.3: Overview of experimental method

Table 6.1: Selected MODTRAN 4 input parameters

target refl.	0.2
aerosol model	rural extinction
visibility	100 km
atmos. model	mid-latitude summer
surface alt.	400 m
solar zenith angle	180°
scattering alg.	ISAAC's two-stream
solar data	Thuillier [Thuillier et al., 2003]
filter resp. func.	see text

$$FWHM_{sub} = R \cdot SSI_{sub} \quad (6.3)$$

$$\mu_i^{sub} = (C_{sum} - ((N - 1) \cdot SSI_{sub}))(i - 1) \cdot SSI_{sub} \quad (6.4)$$

where $SSI_{sub} = SSI_{sum}/N$ (if the SSI between two adjacent summed bands is known or can be assumed), R is the ratio of $FWHM_{sub}$ to SSI_{sub} , SSI_{sub} is the spectral sampling interval between the centres of two adjacent sub-channels, SSI_{sum} is the spectral sampling interval of the summed band, and C_{sum} is the centre wavelength of a given summed band.

The resulting implementation of Equation 6.1 is normalized, ensured to fit the specific band centre location of the summed band, and down-sampled to 65 samples to conform to MODTRAN's input filter convention. In essence, this routine is similar to the previously-used filter generator [Brazile, 2005], but augmented to allow for the additional summing and ratio parameters.

It is now possible to examine the experimental method in detail. An overview of the process is visualized in the flow diagram in Figure 6.3. There are three general phases: 1) pre-generation of instrument-specific trial spectra allowing the instrument's spectral characteristics to vary 2) generation of reference spectra simulating a particular data-take and 3) finding the pre-generated spectra that best matches the reference and evaluating the match for retrieval purposes.

Since all modelled at-sensor radiance spectra are produced by the MODTRAN 4 Radiative Transfer code from the 20% spectrally flat target reflectance input, the ordering of some of the steps in the procedure are determined by the requirements of this software. For example, MODTRAN 4 allows the input of an arbitrary SRF by setting the `CARD 1A3 FLTNAM` parameter [Berk et al., 2003] which references an external ASCII file (with the `.flt` suffix, by convention) containing tables of floating point values representing per-band SRFs. Therefore, in the setup of MODTRAN 4 input, the various SRF `.flt` files must first be generated. To this end, a multiply-nested loop "wrapper" around the function corresponding to Equation 6.1 is used to generate SRFs with instrument-specific but varying spectral characteristics.

Next, individual input files (MODTRAN `tape5` files) for all runs are generated referring to the previously-created external `.flt` files but otherwise with base input parameters as shown in Table 6.1. To reduce the size of the LUT for this experiment, visibility and target reflectance are not allowed to vary since a previous study [Brazile et al., 2006] implies that outcomes due to variations in these parameters do not cause substantial changes in the sensitivity of the result and the primary focus of this study is to examine varying SRFs.

The individual simulations are then run on a cluster of AMD (Advanced Micro Devices) Opteron-based compute nodes [Godknecht and Bolliger, 2004] executing all of the thousands of runs.

Table 6.2: Proposed instrument-specific LUT parameters for retrieval

target refl.
aerosol model
visibility
atmos. model
surface alt.
sensor alt.
solar zenith angle
number of summed sub-channels †
ratio of sub-channel FWHM to SSI †
band centre shift from nominal †
fwhm shift from nominal †

† Only these parameters varied in this experiment

Upon completion of the MODTRAN jobs, the instrument-specific at-sensor radiance spectra in individual `channels.out` files are collected and composed into a single LUT whose dimensions correspond to the retrieval parameters shown in Table 6.2

A second phase of the experiment is to produce reference at-sensor radiance spectra simulating an instrument data-take. The spaceborne sensors Hyperion and CHRIS and the airborne HyMap instrument are considered for this study. This is done by again using MODTRAN 4 to generate the reference spectra. The aforementioned formula (Equation 6.1) is re-used to produce reference SRFs. A benefit of using even-numbered summed Gaussians for SRF parameterization during generation of the LUT allows the use of odd-numbered summed Gaussians to generate similar but differing reference cases for validation. For each of the three instruments, 18 validating reference spectra are generated and evaluated at each of six Fraunhofer feature windows yielding a total of 324 samples for valuation. For generating the reference spectra, non-shifted SRFs are modelled varying over the same six SSIs in the LUT but with summed Gaussians of three, five, and seven rather than the even summed Gaussians in the LUT. It was decided that these combinations tested on each of the three instruments for different feature windows are minimally required to evaluate retrieval feasibility. A more complete validation of the SRF parameterization and the method itself would involve testing less quantized samples over the entire LUT parameter space.

The final phase of the experiment involves the spectrum matching itself as well as evaluation

Table 6.3: Instrument specifications per feature window [nm]

	CHRIS			HyMap			Hyperion		
	$\Delta\lambda$	\overline{SSI}	\overline{FWHM}	$\Delta\lambda$	\overline{SSI}	\overline{FWHM}	$\Delta\lambda$	\overline{SSI}	\overline{FWHM}
d(Fe)	29.60	9.87	9.68	44.10	14.70	15.42	30.52	10.17	11.39
D1/D2(Na)	31.00	10.33	10.62	46.00	15.33	16.05	30.53	10.18	10.70
a(O ₂)	27.70	9.23	9.20	46.00	15.33	16.15	30.53	10.18	10.39
C(H)	30.90	10.30	10.30	45.60	15.20	16.05	30.52	10.17	10.30
B(O ₂)	17.30	5.77	5.75	45.90	15.30	16.32	30.53	10.18	10.37
A(O ₂)	21.30	7.10	7.08	45.10	15.03	16.45	30.53	10.18	10.73

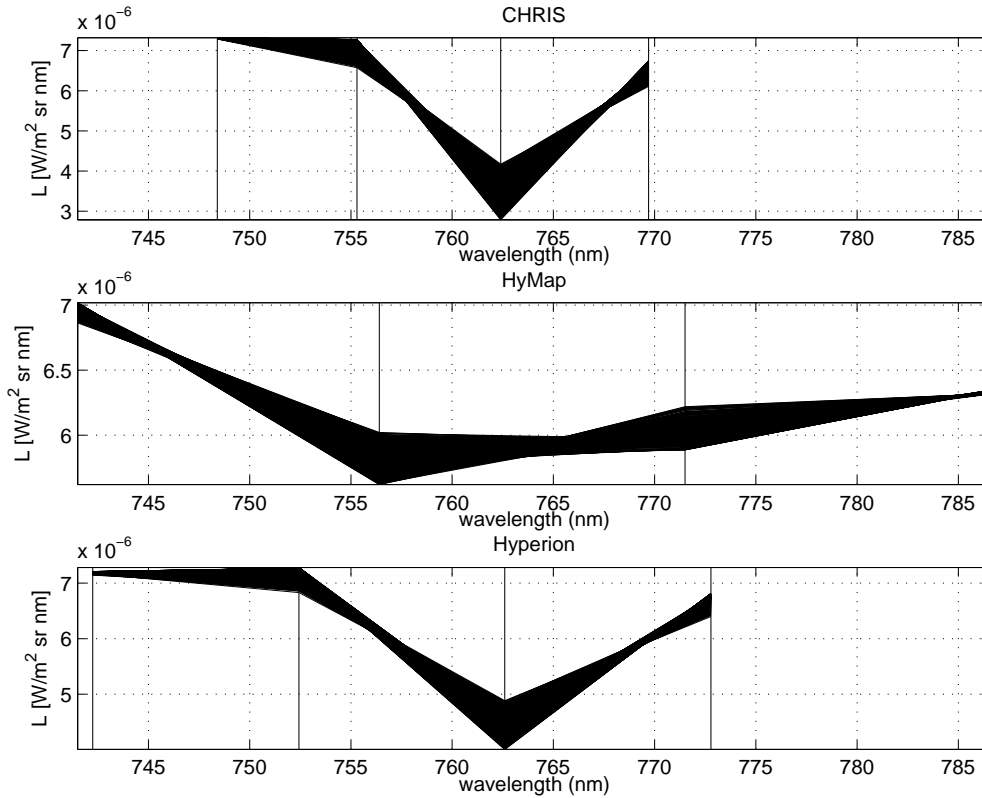


Figure 6.4: Varying instrument-specific LUT spectra for A(O₂) feature window

of the matches for the purpose of retrieval. Spectrum matching is performed using Fraunhofer feature windows of four bands (two on each side of the feature) with the reasoning that perturbations caused by the absorption features increase the likelihood of discerning differences in the SRF-convolved spectra. Instrument specifications in the vicinity of these features are shown in Table 6.3. In practice, sometimes more than four bands are necessary to cover wide absorption features (e.g., 940 nm H₂O). Additionally, absorption features other than these six are usable, and possibly even more appropriate for a particular instrument, but these are the only six that can be uniquely covered by individual windows of four bands for each of the three instruments chosen – allowing for inter-comparison.

The evaluation metric for spectrum matching chosen for this experiment is the relative root mean square (rRMS), calculated as follows for a reference window of radiances, L_R , and a second window of radiances L_X :

$$rRMS(\%) = \frac{\sqrt{\frac{\sum_{i=1}^n (L_{Ri} - L_{Xi})^2}{n}}}{L_{Ri}} \times 100. \tag{6.5}$$

This metric has the merit of directly implying nominal SNR values (i.e., SNR = 100 / rRMS) needed to achieve discernment, as well as allowing direct comparison with previous studies also using this metric [Brazile et al., 2006]. However, to provide an additional estimation of SNR requirements, multiple amounts of simulated noise are added to the signal in discrete steps corresponding to SNR values of 500:1, 1000:1, 5000:1, 10000:1, and 50000:1. In particular, a

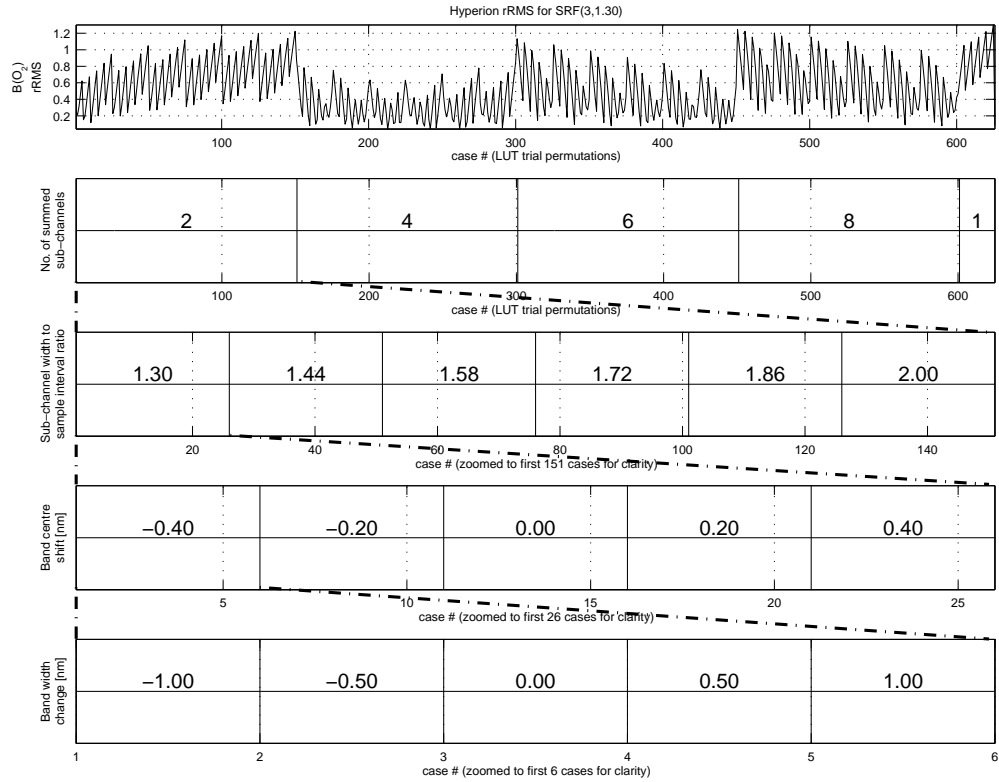


Figure 6.5: Relative RMS (rRMS) versus case #. Relationship between case #'s and LUT input parameters

vector of uniform random noise N at a given signal-to-noise ratio r was added to the modeled radiance spectra S as described by:

$$\vec{S}_n = \vec{S} + \left(\frac{\vec{S}}{r} \times \vec{N} \right) \quad (6.6)$$

It should be noted that each of the goals stated in the previous section These noisy spectra are used to determine conservative requirements in the following manner: the differences between rRMS results with noisy spectra and those with no noise are calculated, then characterized by mean and standard deviation. These characterizations are then compared with the difference between the best rRMS and the next best rRMS result (as calculated without noise). If the difference between these two values is greater than the characterized difference with the given simulated SNR, then it is assumed that SNR performance is sufficient for retrieval. It is argued that the chosen SNR values are challenging but achievable. Since signal-to-noise typically improves by the square root of the number of samples taken [Smith, 1997], one can double the SNR of a single sample by averaging four samples when viewing a relatively homogeneous target.

Given that the target in a scene is varying, the atmosphere is relatively uniform so the shape of the absorption features remains relatively constant throughout the scene provided there are no large topographical variations, which might result in significant variability in atmospheric depth. Thus, scene variability provides (to a good approximation) a varying gain factor to all

the spectral bands that span the absorption feature. One can therefore gain SNR by averaging increasingly more along-track samples. If the single pixel SNR is 200:1, and one needs 8000:1, then $(8000/200)^2 = 1600$ along track samples are needed. For calibration purposes, one should easily be able to obtain 6000 image lines in a scene, which for this example would give an SNR of $> 15,000:1$. And if more is needed, a special calibration data acquisition (assuming a satellite sensor) can be performed over a large desert area (this would give a bright target which itself improves the SNR of each single sample) to give 50,000 image lines. This could translate to an SNR of $\approx 45,000:1$ if each single sample SNR is 200:1.

are addressed by the experimental method. A suitable SRF parameterization is introduced that is able to generate realistic shapes and is at least somewhat continuous throughout its range. To address the goal of spectrum matching robustness in the presence of band centre or band width deviations, i.e., to enable retrieval/refinement of an instrument's spectral line curvature, two dimensions are present in the LUT – one for allowing up to two discrete band centre shifts to the left or right in steps of 0.2 nm, and the other for allowing up to two discrete smaller or larger band width changes in steps of 0.5 nm. The experiment attempts to address the spectral coverage issue by testing a selection of Fraunhofer features likely to be prominent enough to be useful with current instruments [Neville, 2003]. Interpolation of neighboring windows could be used to cover gaps in an instrument's spectral range, but certainly more than a few band windows should be directly covered.

For addressing the goal of general applicability, the experiment is performed on models of three existing instruments with different spectral characteristics – the CHRIS and Hyperion space borne instruments, and the HyMap airborne instrument. Each instrument is modelled using known characteristics such as nominal band centres, sensor altitude, etc. These three instruments are chosen primarily because of the availability of their band specifications, but additionally because their per-band SRFs are not provided – i.e. these are the kinds of instruments where such a retrieval would be desired. The final goal of statistical consistency is intended to be shown by the results of the experiment.

The modelled trial at-sensor radiance spectra LUT entries resulting from the 625 per-instrument permutations ($((4 \text{ summings} \times 6 \text{ sub-channel FWHM/SSI ratios}) + \text{Gaussian}) \times 5 \text{ band centre positions} \times 5 \text{ band width values}$) can be seen for the Fraunhofer A(O₂) feature in Figure 6.4. Finally, Figure 6.5 shows the relationship between the 625 permutations (bottom row) and the varying LUT input parameters (top four rows). For example, it is clear from the fourth row showing the span of summing cases, that the summing parameter is the slowest varying parameter (i.e., the outermost loop) during generation of the LUT. The five summing cases (the smallest sum of one, which has no sub-channel ratio variation, is on the right) are easily recognizable.

6.3 Results

The spectrum matching result across all instruments and feature windows for one of the least successfully matching 18 SRF cases is shown in Figure 6.6 and statistics for the complete set of 324 evaluation samples is characterized in Table 6.4. The figure plots the spectra matching metric, i.e. rRMS for all LUT permutations for a given instrument (columns) and Fraunhofer feature window (rows). Figure 6.7 shows SNR requirements implied by the rRMS for two extreme cases shown in Figure 6.6 and Table 6.5 shows less precise and more conservatively calculated SNR requirements based on characterized differences when adding simulated noise in discrete steps. The effect of this simulated noise upon rRMS can be seen visually in Figure 6.8.

In Figure 6.6, clear patterns can be seen, as entries are allowed to vary in parameters such as band centre or FWHM shifts, number of summed Gaussians, or sub-channel FWHM to SSI ratio

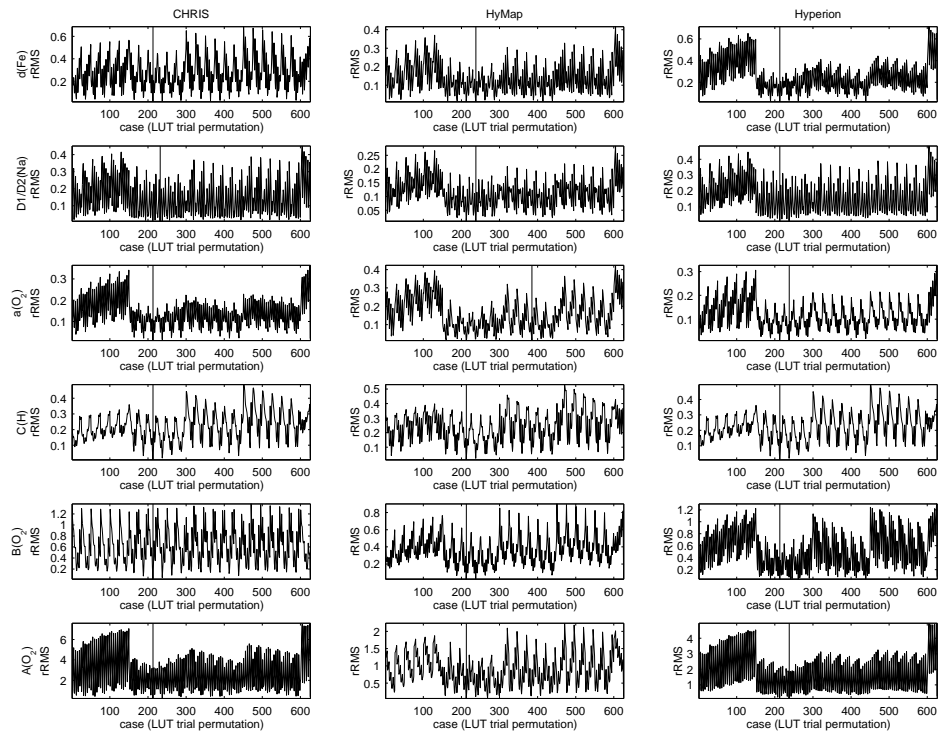


Figure 6.6: Evaluation of matching reference SRF(3,1.30) against LUT entries using rRMS

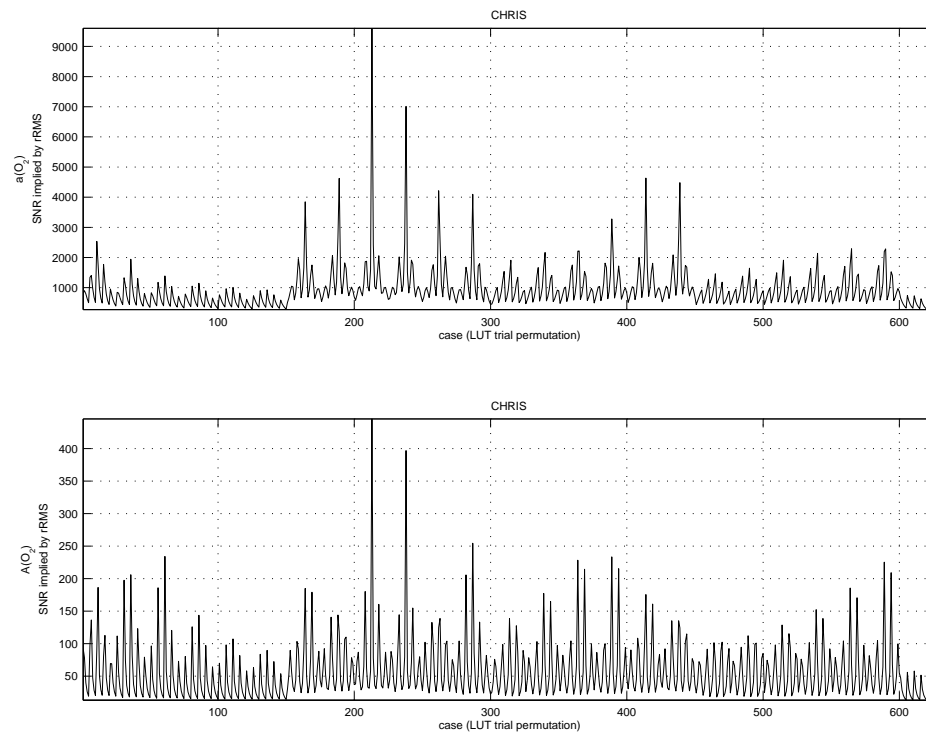


Figure 6.7: Implied SNR requirements for two sub cases of SRF(3,1.30)

Table 6.4: Mean and standard deviation of rRMS for each reference spectra

CHRIS													
		3,1.30		3,1.44		3,1.58		3,1.72		3,1.86		3,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)		0.27	0.14	0.27	0.14	0.28	0.15	0.29	0.16			0.31	0.18
D1/D2(Na)		-	-	0.15	0.09	0.15	0.09	-	-			0.18	0.11
a(O ₂)		0.14	0.06	0.14	0.06	0.15	0.06	0.16	0.07			0.18	0.08
C(H)		0.22	0.09	0.22	0.09	0.22	0.10	0.22	0.10			0.23	0.11
B(O ₂)		0.61	0.31	0.61	0.31	0.62	0.31	0.62	0.32			0.63	0.33
A(O ₂)		2.69	1.61	2.71	1.57	2.81	1.61	-	-			3.39	1.93
		5,1.30		5,1.44		5,1.58		5,1.72		5,1.86		5,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)		-	-	0.29	0.15	0.28	0.14	0.27	0.14	0.27	0.14	0.27	0.14
D1/D2(Na)		0.16	0.11	0.16	0.11	0.16	0.10	0.15	0.10	0.15	0.10	-	-
a(O ₂)		0.16	0.09	0.15	0.09	0.14	0.08	0.14	0.08	0.14	0.07	0.14	0.06
C(H)		-	-	0.23	0.09	0.23	0.09	0.22	0.09	0.22	0.09	0.22	0.09
B(O ₂)		0.62	0.35	0.62	0.33	0.61	0.32	0.61	0.32	0.61	0.31	0.61	0.31
A(O ₂)		3.07	2.04	2.97	1.95	2.87	1.85	2.78	1.75	2.72	1.67	2.69	1.61
		7,1.30		7,1.44		7,1.58		7,1.72		7,1.86		7,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)		0.33	0.17	0.32	0.17	0.31	0.16	0.30	0.15	0.29	0.15	0.28	0.14
D1/D2(Na)		0.18	0.12	0.17	0.11	0.17	0.11	-	-	0.16	0.11	0.16	0.10
a(O ₂)		0.17	0.10	0.17	0.10	-	-	-	-	0.15	0.09	0.15	0.09
C(H)		0.26	0.11	0.25	0.11	0.24	0.10	0.24	0.10	0.23	0.09	0.23	0.09
B(O ₂)		0.64	0.36	0.63	0.36	0.63	0.35	0.62	0.35	0.62	0.34	0.61	0.33
A(O ₂)		3.27	2.19	3.22	2.16	3.15	2.11	3.08	2.05	3.00	1.98	2.92	1.91
HyMap													
		3,1.30		3,1.44		3,1.58		3,1.72		3,1.86		3,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)				0.14	0.07	0.14	0.08	0.15	0.08	-	-	0.17	0.09
D1/D2(Na)				0.11	0.05	0.12	0.05	0.13	0.05	0.13	0.06	0.14	0.06
a(O ₂)				0.17	0.08	0.18	0.09	-	-	-	-	0.23	0.12
C(H)				0.24	0.11	0.25	0.12	0.26	0.13	0.27	0.13	0.28	0.14
B(O ₂)				0.36	0.16	0.37	0.17	0.38	0.18	0.40	0.19	0.42	0.21
A(O ₂)				0.90	0.48	0.94	0.52	1.00	0.57	1.05	0.61	1.11	0.65
		5,1.30		5,1.44		5,1.58		5,1.72		5,1.86		5,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)				0.15	0.09	0.14	0.08	0.14	0.08	0.14	0.08	0.14	0.08
D1/D2(Na)				0.12	0.06	0.11	0.06	-	-	0.11	0.05	0.11	0.05
a(O ₂)				0.18	0.12	0.17	0.11	0.16	0.10	0.16	0.09	-	-
C(H)				0.25	0.11	0.25	0.10	0.24	0.10	0.24	0.10	0.24	0.10
B(O ₂)				0.39	0.18	0.38	0.17	0.37	0.17	0.37	0.16	0.36	0.16
A(O ₂)				0.98	0.54	0.94	0.51	0.91	0.48	0.89	0.46	0.89	0.46
		7,1.30		7,1.44		7,1.58		7,1.72		7,1.86		7,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)		0.17	0.10	0.16	0.10	0.15	0.10			0.15	0.09	0.14	0.09
D1/D2(Na)		0.13	0.07	0.13	0.07	0.12	0.07			0.12	0.06	0.12	0.06
a(O ₂)		0.22	0.15	-	-	0.20	0.14			0.18	0.13	0.17	0.12
C(H)		0.31	0.14	0.29	0.13	0.28	0.13			0.26	0.11	0.25	0.11
B(O ₂)		0.42	0.18	0.42	0.18	0.41	0.18			0.39	0.18	0.38	0.17
A(O ₂)		1.14	0.66	1.10	0.63	1.06	0.61			0.99	0.55	0.96	0.52
Hyperion													
		3,1.30		3,1.44		3,1.58		3,1.72		3,1.86		3,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)		0.26	0.13	0.27	0.12	0.29	0.12	-	-	-	-	0.36	0.17
D1/D2(Na)		0.16	0.10	0.16	0.09	0.17	0.09	0.18	0.09	0.19	0.09	0.20	0.09
a(O ₂)		-	-	0.12	0.05	0.12	0.06	0.13	0.06	0.14	0.07	0.14	0.07
C(H)		0.22	0.09	0.22	0.09	0.22	0.10	0.23	0.10	0.23	0.11	0.24	0.11
B(O ₂)		0.53	0.29	0.55	0.30	0.58	0.33	0.63	0.37	-	-	0.71	0.43
A(O ₂)		-	-	1.71	0.97	1.79	0.96	1.91	1.01	2.06	1.09	2.22	1.18
		5,1.30		5,1.44		5,1.58		5,1.72		5,1.86		5,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)				0.27	0.20	0.26	0.18	0.26	0.16	0.25	0.15	0.26	0.13
D1/D2(Na)				0.17	0.11	0.17	0.10	0.17	0.10	0.17	0.10	-	-
a(O ₂)				0.12	0.07	0.12	0.07	0.12	0.06	0.12	0.06	0.12	0.06
C(H)				-	-	0.23	0.09	0.23	0.09	0.22	0.09	0.22	0.09
B(O ₂)				0.59	0.39	0.55	0.35	0.53	0.32	0.52	0.30	-	-
A(O ₂)				1.82	1.27	1.78	1.22	1.73	1.16	1.70	1.10	1.69	1.04
		7,1.30		7,1.44		7,1.58		7,1.72		7,1.86		7,2.00	
		\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ	\bar{x}	σ
d(Fe)		0.33	0.23	0.32	0.23	0.30	0.22			0.28	0.20	0.27	0.19
D1/D2(Na)		0.18	0.11	0.18	0.11	0.18	0.11			-	-	0.17	0.11
a(O ₂)		0.14	0.08	0.14	0.08	0.13	0.08			0.13	0.08	0.12	0.07
C(H)		0.27	0.11	0.26	0.11	-	-			0.24	0.09	0.23	0.09
B(O ₂)		0.79	0.52	0.73	0.49	0.67	0.46			-	-	0.57	0.37
A(O ₂)		1.94	1.34	1.92	1.34	1.90	1.32			1.84	1.28	1.80	1.25

Table 6.5: Conservative SNR requirements (500, 1000, 5000, 10000, 50000, ∞) implied by simulation

		CHRIS					
		3,1.30	3,1.44	3,1.58	3,1.72	3,1.86	3,2.00
d(Fe)		10000	5000	5000	10000		5000
D1/D2(Na)		-	5000	5000	-		5000
a(O ₂)		50000	5000	5000	10000		5000
C(H)		5000	5000	5000	5000		5000
B(O ₂)		5000	5000	1000	1000		1000
A(O ₂)		5000	500	500	-		500
		5,1.30	5,1.44	5,1.58	5,1.72	5,1.86	5,2.00
d(Fe)		-	5000	10000	50000	5000	50000
D1/D2(Na)		50000	50000	50000	50000	50000	-
a(O ₂)		10000	10000	10000	50000	10000	50000
C(H)		-	10000	5000	∞	10000	50000
B(O ₂)		5000	5000	5000	10000	5000	10000
A(O ₂)		5000	500	5000	5000	500	5000
		7,1.30	7,1.44	7,1.58	7,1.72	7,1.86	7,2.00
d(Fe)		5000	50000	50000	∞	50000	10000
D1/D2(Na)		10000	50000	50000	-	50000	10000
a(O ₂)		5000	50000	-	-	50000	10000
C(H)		5000	50000	50000	50000	50000	10000
B(O ₂)		5000	50000	5000	50000	10000	5000
A(O ₂)		1000	5000	5000	50000	5000	500
		HyMap					
		3,1.30	3,1.44	3,1.58	3,1.72	3,1.86	3,2.00
d(Fe)			5000	5000	10000	-	5000
D1/D2(Na)			5000	5000	10000	5000	5000
a(O ₂)			5000	5000	-	-	5000
C(H)			5000	5000	5000	5000	5000
B(O ₂)			5000	1000	1000	5000	1000
A(O ₂)			1000	500	1000	5000	500
		5,1.30	5,1.44	5,1.58	5,1.72	5,1.86	5,2.00
d(Fe)			5000	10000	50000	5000	50000
D1/D2(Na)			50000	50000	-	50000	50000
a(O ₂)			10000	10000	50000	10000	-
C(H)			10000	5000	∞	10000	50000
B(O ₂)			5000	5000	10000	5000	10000
A(O ₂)			5000	5000	10000	5000	10000
		7,1.30	7,1.44	7,1.58	7,1.72	7,1.86	7,2.00
d(Fe)		5000	50000	50000		50000	10000
D1/D2(Na)		10000	50000	50000		50000	10000
a(O ₂)		5000	-	50000		50000	10000
C(H)		5000	50000	50000		50000	10000
B(O ₂)		5000	50000	5000		10000	5000
A(O ₂)		1000	10000	5000		5000	5000
		Hyperion					
		3,1.30	3,1.44	3,1.58	3,1.72	3,1.86	3,2.00
d(Fe)		10000	5000	5000	-	-	5000
D1/D2(Na)		50000	5000	5000	10000	5000	5000
a(O ₂)		-	5000	5000	10000	50000	5000
C(H)		5000	5000	5000	5000	5000	5000
B(O ₂)		5000	5000	1000	1000	-	1000
A(O ₂)		-	500	500	500	500	500
		5,1.30	5,1.44	5,1.58	5,1.72	5,1.86	5,2.00
d(Fe)			5000	10000	50000	5000	50000
D1/D2(Na)			50000	50000	50000	50000	-
a(O ₂)			10000	10000	50000	10000	50000
C(H)			-	5000	∞	10000	50000
B(O ₂)			5000	5000	10000	5000	-
A(O ₂)			1000	5000	5000	1000	50000
		7,1.30	7,1.44	7,1.58	7,1.72	7,1.86	7,2.00
d(Fe)		5000	50000	50000		50000	10000
D1/D2(Na)		10000	50000	50000		-	10000
a(O ₂)		5000	50000	50000		50000	10000
C(H)		5000	50000	-		50000	10000
B(O ₂)		5000	50000	5000		-	5000
A(O ₂)		5000	5000	5000		5000	1000

variations. The best matching entry in each plot is marked with a vertical bar. It can be seen that in the CHRIS column, the row corresponding to the D1/D2(Na) feature window disagrees with all the others. Similarly, in the HyMap column, the results for the best match are split, with only the bottom three features agreeing on the best match.

It is also apparent from the figure that some features provide more clear discernment than others – the C(H) feature shows prominent spikes in each varying group, and the A(O₂) feature provides the largest rRMS magnitudes.

Again, Table 6.4 characterizes the matching statistics for the set of 324 evaluation samples. In the table, the result for a particular sample is not shown if it falls into one of two situations. First, if it not clear from the data what the best match should be because the matching results are split, then the entire instrument-specific column for that reference point is left blank. An example of this situation appears in the first HyMap column for the reference corresponding to the SRF generated from 3 binned Gaussians and a FWHM/SSI ratio of 1.30. This can be visually verified by looking at the HyMap column in Figure 6.6. In this column, only half of the features (i.e., the bottom three) agree on the match. Therefore the entire column in the corresponding table (column 1, row 4) is left blank.

The other situation occurs when the best match is not the one agreed upon by the other features. In this case, the data for the given evaluation sample is replaced by a dash '-'. An example of this case occurs with the D1/D2(Na) feature of the first case (3,1.30) in the CHRIS result. Again this can be visually verified in Figure 6.6 by looking at the CHRIS column and noticing that the best match in the 2nd row disagrees with the best match of all the other features. The corresponding table entries for the mean and standard deviation of that case (column 1, row 1, D1/D2(Na)) are replaced with a dash '-'. Entries in Table 6.5 corresponding to entries in Table 6.4 are also treated the same way with respect to missing entries.

Table 6.4 reveals that many of the characteristics present in the example graph also hold for the other trials. For example, the A(O₂) feature often provides the largest variance and mean rRMS values in all trials and the D1/D2(Na) feature could cause discernment difficulties, especially in the presence of noise.

Given the reference SRFs selected for this experiment, 11% (6 out of 54) of the SRF matches are inconclusive because the selection of best match is split among the feature windows tested. In 44% of the SRF matches at least one result did not agree with the majority.

6.4 Discussion

6.4.1 Effectiveness of Method

It is both positive and surprising that a majority of the 54 evaluation samples unanimously agree upon the best matching result. Yet the negative results show how precarious this success might really be in some cases. If one were presented only with graphs of the differences in spectra matching (Figure 6.6), it would be difficult to predict that the a(O₂) feature would be the case to consistently cause trouble. Intuitively, it would seem the best situation when groups of results steadily rise or fall, as in d(Fe) cases 1-150 for HyMap and Hyperion – one would expect that the trend implies where the best match is to be found. In the worst case, all groups hover in flat trends such as in the D1/D2(Na) CHRIS and Hyperion cases 200-600. Not only are the groups flat, but it appears that about 20% of the values could compete with each other for the best match, especially when there is unpredictable noise. On the other hand, it is not required that every Fraunhofer feature yield a clearly successful match. Statistical methods might be

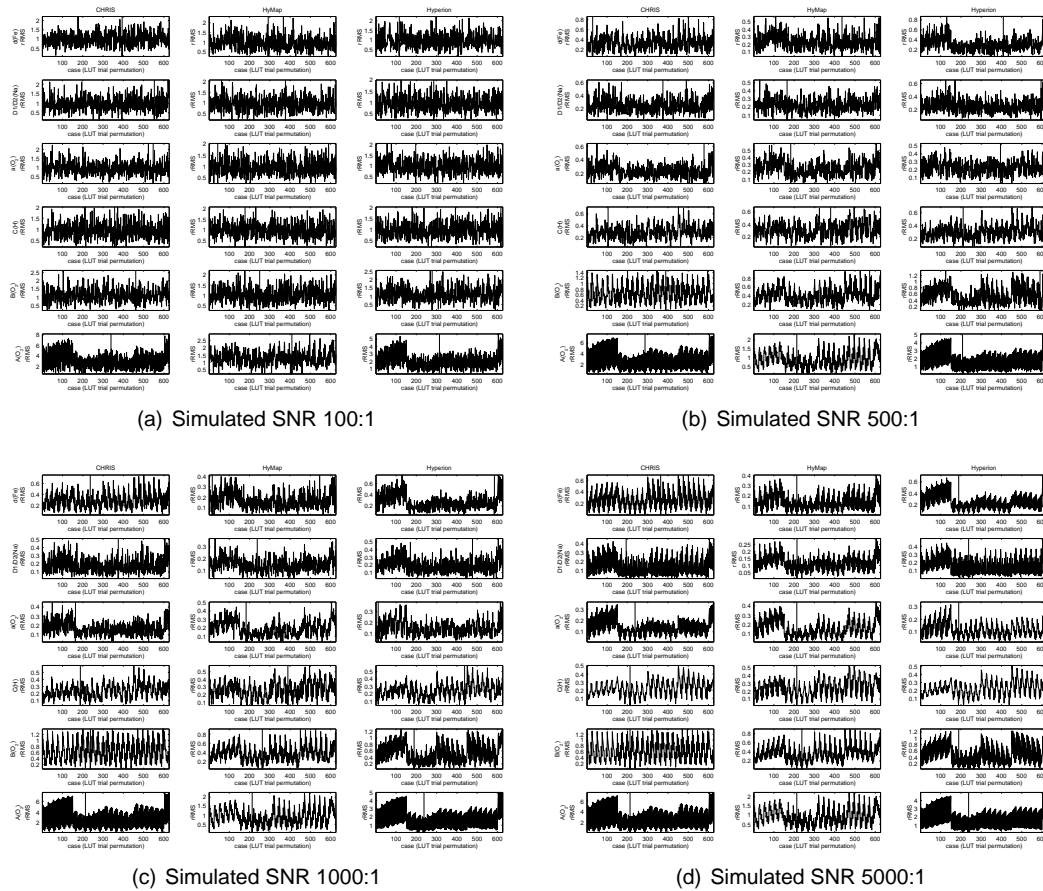


Figure 6.8: Same as Figure 6.6, but with varying simulated noise

applicable for allowing stronger neighboring windows to compensate for the results of the less prominent matches. It could also be that partial knowledge of the instrument (e.g., if it is likely to be summed) or knowledge of the underlying sensors reveals some sub-channel insight that might play a role in dismissing unlikely combinations in order to help reduce the search space.

One very promising result is revealed by examining the $A(O_2)$ case across all instruments. Upon revisiting Figure 6.4, it is interesting to see that for CHRIS and Hyperion, the 3rd channel in the two instrument-specific $A(O_2)$ windows line up almost exactly and seem to be quite near to the trough of the feature between 760nm and 765nm. However, HyMap not only doesn't line up with the other two instruments, it seems not to have any band that contains the trough as its primary constituent, as can be seen by the relative similarity of the value ranges of the 2nd and 3rd channel in the feature. In spite of this, the retrieval method seems to do well enough at providing a clearly matching spectra. The positive conclusion from this is that, at least in some cases, the width and prominence of a feature can be more important in deciding discernability than the placement and interval of a given instrument's band. Another interesting observation from the same feature is to compare Figure 6.4 with the bottom row of Figure 6.6. CHRIS and Hyperion produce similar signatures in the LUT which is seemingly due to the similar shape of the spectra in the window even though the SSI of the two instruments are clearly different. The signature in the LUT for HyMap for this case is the other extreme.

Integration Within Processing Chain

In addition to validation of the method and iteratively refining its currently nominal components, the method and its components must also integrate well within a complete processing chain. Such a processing chain has been refined over many years by three of the authors for preprocessing hyperspectral data such as acquired by EO-1 Hyperion [Khurshid et al., 2006]. In that chain, a noise-reduction step involving ISDAS' `average-smooth` tool [Staenz et al., 1998] is already performed in preparation for an existing spectral smile detection step. The results of this noise reduction could be used unaltered as input to a revised spectral smile and SRF detection module. In fact, the existing smile detection module already not only detects smile but also band center and band width shifts as well as gain/offset detection in an iterative feedback loop coupled with atmospheric correction since these need to all be performed simultaneously. It is this module that would need to incorporate the additional SRF parameterization and window based spectrum matching components proposed here. Broad coverage over the instrument's range as well as statistical consistency tests are already implemented by the current module. The newly retrieved SRFs would then be added to the already retrieved band center and band width parameters as input for the immediately following atmospheric correction module. Smile correction, if needed, would occur following atmospheric correction.

6.5 Conclusions

The theoretical discernability of SRF shapes has been confirmed using more practical shapes than the abstract functions of the previous study [Brazile et al., 2006]. In particular, it appears feasible to fully cover a continuous range of symmetric shapes from Gaussian to boxcar. Although the Tukey SRF parameterization appears interesting because of its smoothness, it has been rejected for its inability to cover shapes closer to the often specified Gaussian. The feasibility of using a summed Gaussian with sub-channel FWHM to SSI ratio parameterization shows promise in its shape coverage in the limited trials presented here. The robustness of this parameterization together with the use of rRMS as spectrum matching metric are suggested as nominal implementations of proposed SRF retrieval components because of the ability to match even in the presence of band centre and band width deviations. However, less smooth parameterization in two dimensions may prove to be a detriment when attempting to approach higher levels of retrieval accuracy. An additional promising result with this nominal implementation is that all but 4% of the combinations of sensor and single feature Fraunhofer windows presented here provide a statistically successful outcome. This holds promise for supporting SRF retrieval throughout a nontrivial subset of an instrument's entire spectral range. Clearly, some features reveal more prominent signal discernment than others, but assuming the components of the proposed method improve over time, along with the use of signal-to-noise enhancement such as along-track averaging, the basic method is expected to be applicable even in weaker cases.

Finally, the general applicability of the method of spectrum matching for SRF characterization is promising, as three clearly different instruments (CHRIS, HyMap, and Hyperion) yielded uniformly positive results even though the feature window sizes and locations relative to the feature centres varied greatly.

It is suggested that the method, even in its currently primitive form, could be used to obtain SRF estimates better than Gaussian for the not-uncommon case in which bands are created by summing up to tens of sub-channels.

At the same time, it is noted that these conclusions have been based upon a limited set of trials and that further research is required to substantiate them.

6.6 Acknowledgments

This work was supported in part by ESA/ESTEC contracts 16298/02/NL/US and 15449/01/NL/Sfe. The authors wish to acknowledge the generous support of the University of Zurich IT Services Department for use of the Matterhorn cluster. J. Brazile thanks Netcetera AG for allowing the sabbatical enabling this collaboration with the Canada Centre for Remote Sensing.

Appendix: Computational Requirements

For reasons of scientific rigor and reproducibility, all SRF filtering in this experiment is performed directly by MODTRAN 4 using the `FLTNAM` parameter. However, for generating a LUT, this method results in a large amount of wasted effort, since the same base spectrum is recomputed for each of the hundreds of varying SRFs. In practice, a spectrum needs only to be modeled by MODTRAN once at high resolution for each base case (example as summarized in Table 6.2) and a separate, post-processing convolution can be run to achieve the simulated instrument-filtered result for each of the varying SRFs.

Implementing this shortcut would lead to a more manageable operational process, especially when considering the number of MODTRAN executions that are required to cover common cases. Real cases require multiple step variation of several base parameters e.g. multiple target reflectances, aerosol models, visibilities, atmospheric models, surface altitudes, solar zenith angles, etc.

A common solution for the batch operation of multiple executions of a single program with varying input parameters is the use of queuing software such as Condor [Litzkow et al., 1988], whose manual excerpt describing use of the `initialdir` and `queue` parameters is directly applicable to this situation. Unfortunately, Condor is not intended for low latency invocation, although experimental patches are available which provide this capability [Palatin and Kliot, 2003]. In the MODTRAN configurations used in this study, roughly the same amount of time is spent in between executions as during the executions themselves – effectively doubling the total run time from the expected run time. For this reason, as well as the difficulty in operating across firewalls [Beckles et al., 2005], a simpler, HTTP-based, “low fat grid” [Brazile, 2006] application is used for performing the cooperative sharing of computing resources at the disperse institutions.

References

- [Anger et al., 1996] Anger, C., Achal, S., Ivanco, T., Mah, S., Price, R., and Busler, J. (1996). Extended operational capabilities of casi. In Proceedings of the Second International Airborne Remote Sensing Conference, pages 124–133, San Francisco, CA. EOS.
- [Barnsley et al., 2004] Barnsley, M. J., Settle, J., Cutter, M., Lobb, D., and Teston, F. (2004). The PROBA/CHRIS mission: A low-cost smallsat for hyperspectral, multi-angle observations of the earth surface and atmosphere. IEEE Transactions on Geoscience and Remote Sensing, 42:1512–1520.
- [Beckles et al., 2005] Beckles, B., Son, S.-C., and Kewley, J. (2005). Current methods for negotiating firewalls for the Condor system. In Cox, S. and Walker, D. W., editors, Proc. 4th UK e-Science All Hands Meeting.
- [Berk et al., 2003] Berk, A., Anderson, G., Acharay, P., Hoke, M., Chetwynd, J., Bernstein, L., Shettle, E., Matthew, M., and Adler-Golden, S. (2003). MODTRAN 4 version 3 revision 1 USER'S MANUAL.
- [Berk et al., 1998] Berk, A., Bernstein, L., Anderson, G., Acharya, P., Robertson, D., Chetwynd, J., and Adler-Golden, S. (1998). MODTRAN cloud and multiple scattering upgrades with applications to AVIRIS. Remote Sensing of Environment, 65(3):367–375.
- [Brazile, 2005] Brazile, J. (2005). MODTRAN-compatible SRF generator. [Online; accessed Jan-2005]. <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8723>.
- [Brazile, 2006] Brazile, J. (2006). Low fat grid. [Online; accessed Nov-2006]. <http://code.google.com/p/lowfatgrid/>.
- [Brazile et al., 2006] Brazile, J., Neville, R. A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K. I. (2006). Scene-based spectral response function shape discernibility for the APEX imaging spectrometer. IEEE Geoscience and Remote Sensing Letters, 3:414–418.
- [Casadio and Colagrande, 2003] Casadio, S. and Colagrande, P. (2003). Meris O₂ calibration using sciamachy measurements. In Proceedings MERIS User Workshop, Frascati, Italy. ESA.
- [Cocks et al., 1998] Cocks, T., Jenssen, R., Stewart, A., Wilson, I., and Shields, T. (1998). The HyMap(TM) airborne hyperspectral sensor: The system, calibration and performance. In 1st EARSeL Workshop on Imaging Spectroscopy, pages 37–42, Zurich, Switzerland.
- [Gao et al., 2002] Gao, B.-C., Montes, M. J., and Davis, C. O. (2002). A curve-fitting technique to improve wavelength calibrations of imaging spectrometer data. In Proceedings of the 11th JPL Airborne Earth Science Workshop, volume 03-4, pages 99–105.

- [Gao et al., 2004] Gao, B.-C., Montes, M. J., and Davis, C. O. (2004). Refinement of wavelength calibrations of hyperspectral imaging data using a spectrum-matching technique. Remote Sensing of Environment, 90:424–433.
- [Godknecht and Bolliger, 2004] Godknecht, A. J. and Bolliger, C. (2004). University of Zurich Matterhorn cluster. [Online; accessed Sep-2004]. <http://www.matterhorn.unizh.ch>.
- [Green et al., 1988] Green, R. O., Eastwood, M. L., Sarture, C. M., Chrien, T., Aronsson, M., Chippendale, B., Faust, J., Pavri, B., Chovit, C., Solis, M., Olah, M., and Williams, O. (1988). Imaging spectrometry and the airborne visible/infrared imaging spectrometer (AVIRIS). Remote Sensing of Environment, 65:227–248.
- [Harris, 1978] Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. Proceedings of the IEEE, 66(1):51–83.
- [Itten et al., 1997] Itten, K. I., Schaepman, M., De Vos, L., Hermans, L., Schläepfer, H., and Droz, F. (1997). APEX - airborne prism experiment: A new concept for an airborne imaging spectrometer. In 3rd Intl. Airborne Remote Sensing Conference and Exhibition, volume 1, pages 181–188.
- [Khurshid et al., 2006] Khurshid, K. S., Staenz, K., Sun, L., Neville, R., White, H. P., Bannari, A., Champagne, C. M., and Hitchcock, R. (2006). Preprocessing of EO-1 Hyperion data. Canadian Journal of Remote Sensing, 32(2):84–97.
- [Litzkow et al., 1988] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor-a hunter of idle workstations. In The 8th International Conference on Distributed Computing Systems, pages 104–111.
- [Milton, 2006] Milton, E. J. (2006). CASI-2 sensor data acquisition modes. [Online; accessed Nov-2006]. <http://arsf.nerc.ac.uk/documents/casi2.pdf>.
- [Milton and Choi, 2004] Milton, E. J. and Choi, K. Y. (2004). Estimating the spectral response function of the CASI-2. In Annual Conference of the Remote Sensing and Photogrammetry Society, Aberdeen, Scotland. Remote Sensing and Photogrammetry Society, Nottingham, UK.
- [Neville, 2003] Neville, R. A. (2003). Spectral absorption features for use in calibrating imaging spectrometers. Technical Report CHTN_2003_002, Canada Centre for Remote Sensing, Ottawa, Ontario, Canada. 11 pages.
- [Neville et al., 2003] Neville, R. A., Sun, L., and Staenz, K. (2003). Detection of spectral line curvature in imaging spectrometer data. In Shen, S. and Lewis, P., editors, SPIE Algorithms and Technologies for Multispectral Hyperspectral and Ultraspectral Imagery IX, volume 5093, pages 144–154, Orlando, FL, USA.
- [Neville et al., 2004] Neville, R. A., Sun, L., and Staenz, K. (2004). Detection of keystone in imaging spectrometer data. In Shen, S. and Lewis, P., editors, SPIE Algorithms and Technologies for Multispectral Hyperspectral and Ultraspectral Imagery X, volume 5425, pages 208–217, Orlando, FL, USA.
- [Nieke et al., 2005] Nieke, J., Itten, K., Debruyn, W., Kaiser, J., Schläpfer, D., Brazile, J., Meuleman, K., Kempeneers, P., Neukom, A., Schilliger, T., De Vos, L., Piesbergen, J., Gege, P., Suhr, B., Schaepman, M., Gavira, J., Ulbrich, G., and Meynart, R. (2005). The airborne imaging spectrometer APEX: From concept to realization. In Zagojowski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 67–74, Warsaw, Poland.

- [Palatin and Kliot, 2003] Palatin, N. and Kliot, G. (2003). Low latency invocation in condor. Technical report, Technion Computer Science Department, Israel. 22 pages.
- [Pearlman et al., 2003] Pearlman, J. P., Barry, P., Segal, C., Shepanski, J., Beiso, D., and Carman, S. (2003). Hyperion, a space-based imaging spectrometer. IEEE Transactions on Geoscience and Remote Sensing, 41(6):1160–1173.
- [Ramon et al., 2003] Ramon, D., Santer, R., and Dubuisson, P. (2003). MERIS in-flight spectral calibration in O₂ absorption using surface pressure retrieval. In Huang, H.-L., Lu, D., and Sasano, Y., editors, SPIE Optical Remote Sensing of the Atmosphere and Clouds III, volume 4891, pages 505–514.
- [Rothman et al., 2003] Rothman, L., Barbe, A., Benner, D. C., Brown, L., Camy-Peyret, C., Carleer, M., Chance, K., Clerbaux, C., Dana, V., Devi, V., Fayt, A., Flaud, J.-M., Gamache, R., Goldman, A., Jacquemart, D., Jucks, K., Lafferty, W., Mandin, J.-Y., Massie, S., Nemtchinov, V., Newnham, D., Perrin, A., Rinsland, C., Schroeder, J., Smith, K., Smith, M., Tang, K., Toth, R., Auwera, J. V., Varanasi, P., and Yoshino, K. (2003). The HITRAN molecular spectroscopic database: edition of 2000 including updates through 2001. Journal of Quantitative Spectroscopy and Radiative Transfer, 82:5–44.
- [Schaepman and Itten, 1998] Schaepman, M. and Itten, K. (1998). APEX-airborne PRISM experiment: An airborne imaging spectrometer serving as a precursor instrument of the future ESA land surface processes and interactions mission. In ISPRS Commission VII Symposium on Resource and Environmental Monitoring, volume 22(7), pages 31–37, Budapest, Hungary.
- [Sinclair et al., 2002] Sinclair, P., Berman, R., Hersom, C., and Hollinger, A. (2002). Spectral calibration of hyperspectral imagers using a white light etalon. In Proceedings of the 12th CASI Conference on Astronautics (ASTRO 2002), Ottawa, Ontario, Canada.
- [Smith, 1997] Smith, S. W. (1997). The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishers.
- [Staenz et al., 1998] Staenz, K., Szeredi, T., and Schwarz, J. (1998). ISDAS - a system for processing/analyzing hyperspectral data. Canadian Journal of Remote Sensing, Vol. 24, No. 2:99–113.
- [Thuillier et al., 2003] Thuillier, G., Hersé, M., Labs, D., Foujols, T., Peetermans, W., Gillotay, D., Simon, P. C., and Mandel, H. (2003). The solar spectral irradiance from 200 to 2400 nm as measured by the SOLSPEC spectrometer from the Atlas and Eureca missions. Solar Physics, 214:1–22.
- [Weisstein, 2005] Weisstein, E. W. (2005). Full width at half maximum. [Online; accessed Jan-2005]. <http://mathworld.wolfram.com/FullWidthatHalfMaximum.html>.

Chapter 7

Streaming, Language-Neutral, Hyperspectral Image Processing Components

Brazile, J., Plaza, A., Schläpfer, D., Nieke, J., and Itten, K.I., Streaming, Language-Neutral, Hyperspectral Image Processing Components, Software Practice and Experience, Wiley, 2007, in preparation.

Abstract

An image format, data stream, and a set of generic, stand-alone, composable utility programs are proposed for the streaming “pipeline” processing of hyperspectral imaging data. The compositability of the stand-alone utilities or “stages”, implemented using the venerable Unix pipe paradigm, allow novel uses via mix and match of components, and per-component parameterization via command-line flags for the development of complete data processing chains.

A standardized data stream interface allows insertion anywhere within the processing chain of additional experimental “stages” by users themselves – supporting specialized research and possibly with the help of language-specific libraries, allowing ignorance of non-related framework-provided detail. Since all stages are stand-alone programs, they can be written in any programming language that can implement the data stream format.

It is claimed that such support for end-users as well as component developers is necessary for promoting greater longevity – a desirable trait for scientific software. The development of prolific software is more achievable now as the concept and acceptance of open source software development becomes more widespread.

As a proof of concept, a Level-1 processor is (re)-implemented using this paradigm for the inverse processing of synthetic Airborne Prism EXperiment (APEX) pushbroom imaging spectrometer data from raw digital numbers to at-sensor radiance in scientific units.

Additionally, to show the advantage in adaptability of such a component architecture, a scatter/gather stage is proposed for the support of data-parallel distributed processing in a modular and run-time configurable manner.

Example processing stages are provided in the C and Java system languages as well as the dynamic languages Tcl, and Perl, and data modelling languages Matlab and IDL.

7.1 Introduction

The processing of hyperspectral imaging data is becoming as common as two dimensional bit-mapped raster processing has been in the past. In the earth observation scientific community, there have been many image processing systems written in different programming languages: AVIRIS (Fortran) [Aronsson, 1998], ISDAS (C) [Staenz et al., 1998], Beam (Java) [Fomferra and Brockmann, 2005], and APEX-PAF (IDL) [Schläpfer et al., 2004] to list a few. The market also offers many, increasingly high quality products such as IDL/ENVI, ESRI/ARC, and PCI/Geomatica. Yet the pattern of duplicated effort producing less-than-interoperable, inflexible, solutions shows little promise of change over the long run.

The idea that software should be componentized – built from powerful, primitive components, has been around at least as long as Douglas McIlroy's presentation at a conference on software engineering in 1968 [McIlroy, 1968]. Except for specifics (e.g. programming languages), many of the complaints made then are the same as those today:

[We have systems] from a dozen manufacturers. Even though many are dedicated to special applications, a tremendous amount of similar software must be written for each. All need input-output conversion, sometimes only [...] characters and [...] numbers, [other times] full-blown Fortran style I/O.

On the question of why the market can't provide these components:

Software houses [...] work must be financed, and large financing can usually only be obtained for large products. So we see the software houses purveying systems, or very big programs[.] The manufacturers produce unbelievable amounts of software. Generally, as this is the stuff that gets used most heavily it is all pretty reliable, a good conservative grey, that doesn't include the best routine for anything, but that is better than the average programmer is likely to make. [...] manufacturers tend to be rather pragmatic in their choice of methods. They strike largely reasonable balances between generality and specificity and seldom use absolutely inappropriate approaches in any individual software component. But the profit motive wherefrom springs these virtues also begets their prime hangup – systems now. The system comes first; components are merely annoying incidentals. Out of these treadmills I don't expect to see high class components of general utility appear.

Shortly thereafter, McIlroy's invention of the pipe concept as implemented in the Unix operating system in 1972 [Ritchie, 1979] is thought by many to be the most practically useful implementation of a componentized software framework of the many attempts over the intervening decades.

A highly relevant example in the history of the development of raster image processing algorithms is a common portable interchange format and pipelined processing concept first released as the portable bitmap programs (PBM) [Poskanzer, 1998]. This concept and concrete implementation supported the digital raster image processing research community in establishing a framework for sharing code and programs to handle tedious tasks while allowing researchers to focus on specific algorithms advancing the state-of-the-art.

The present work proposes employing the same paradigm for supporting the field of hyperspectral image processing. Currently, the interchange of raw and processed hyperspectral imaging data is supported via file format standards such as NCSA's HDF, ESRI's ArcView format, or ITT's ENVI format, but the sharing and exchange of experimental component algorithms during the stages of processing is not common. Therefore, a composable, streaming image processing concept is proposed with the goal of sharing programs and code for established tedious tasks, while fostering research on state-of-the-art specialized algorithms for less well-established stages.

Using the pipeline model of composable stand-alone tasks, users can prepend, rearrange, remove, append, or interject pipeline stages at run-time, while developing and/or tuning a particular processing chain. Once a desirable sequence of stages is discovered, the concept shows how meta-components such as a scatter/gather stage could be used to distribute processing among multiple workstations for parallel operation. Because each pipeline stage is a stand-alone program, it can be written in any programming language, which potentially expands value through the **network effect** [Katz and Shapiro, 1994].

Example stages are provided in multiple system programming languages (often used for performance) as well as multiple high-level data modeling languages (often used for rapid prototyping).

To demonstrate the realistic usability of this toolkit, a functional Level-1 (digital number to at-sensor radiance) hyperspectral image processing chain for the APEX airborne pushbroom spectrometer [Nieke et al., 2005] is developed using a pipeline of multiple stages of the toolkit .

7.2 Design Goals For Promoting Longevity

The primary objective of this work is a framework design that can outlive the programming language used to develop it. Given that the oldest high level programming languages are 50 years old, this can either seem not stringent enough, when compared with the longevity of most of the other mathematical and scientific concepts behind imaging spectroscopy or too grandiose, when compared to the typical lifespan of popular programming languages. The current popular winner, Java, has had a good decade but seems to be on a downward trend according to an index based on job postings and Internet searches [tiobe, 2006] (Fortran just misses the top 20) as well as a separate metric based on multi-year technical book sales [O'Reilly, 2006].

Yet, Unix outlived its first implementation language and prospered in multiple independent re-implementations since then. PBM was almost exclusively C, but later saw many components (re)written in other, emerging languages (including Java). Based on the characteristics of prolific components up to now, a subset of design goals have been derived – some primarily for the benefit of the end-user, others primarily for the benefit of the component developer.

stand-alone component-based (user) Making the components stand-alone helps enforce language independence. The typical motto is “do one thing, but do it well”. The key is finding the most appropriately powerful “one thing” operators.

streaming (user) Complex data structures seem to call for multiple files and file formats, yet the pipeline paradigm requires streaming in order to allow flexible, novel uses. Also, the stream should be capable of arbitrary length – physical headers (if any) cannot be required to contain information that requires prescience.

parallelizable (user) Multi core machines are now mainstream and this trend will only increase. The most flexible approach is to allow the end user to experiment both with data-driven and code-driven parallel pipeline formations.

multi-lingual (programmer) Domain-specific languages are often an excellent way to focus primarily on the problem at hand. But this leads to different languages for different domains. A multi-lingual system is not only more inclusive, it allows for the most appropriate language to be selected for any given task.

algorithm-centric (programmer) The hope is that implementation of important components can be contributed by 3rd parties in two pages or less of code, with 90% of that code concerning itself only with the algorithm and not the interface to the framework.

limitation-averse (programmer) Language-specific object serialization in the data stream or image formats is detrimental. On defining internal formats, hard-coding a particular byte-ordering would be a minor sin, but hard-coding a particular band interleave would be a major sin. Likewise, limiting to only code-parallel or only data-parallel options for distributed processing is undesirable.

7.3 Implementation

7.3.1 Streaming/Archiving

In address the common case first, the largest amount of data is expected to be image data itself. This is likely to be the easiest to process as a stream as long as the number format, byte ordering and band interleaving is known. If this were the only type of data, the PAM extension could be made to suffice. The primary difficulty is the variability and large amount of possibly associated meta-data (perhaps even per-pixel). This implies that the meta-data should be out-of-band rather than as an embedded header. Yet, out-of-band data implies multiple files, which at first seems contradictory to a stream-of-bytes approach.

However, if the data file, together with associated meta-data files were to be packaged together, say as an archive, then the archive could be streamed. With the proper choice of archive format, data could be processed/transformed on-the-fly as the data archive streams past. Minimally that means that meta-data files should appear in the archive before their associated image files. As with the corresponding Unix paradigm, there can still be unlucky cases where a component (e.g. sort) needs to view the entire stream before it can begin emitting its own stream. In practice, this just means sometimes you might have to make (a constant) multiple passes through the data. The paradigm has proven to still be highly useful in spite of such setbacks.

An obvious choice of data stream would be the ZIP format, since it is so common. However, this has a few problematic limitations. First, it requires the size and the contents of the archive to be known in advance since it is written in the archive header. We would like to reserve the right to decide to add, delete or modify files in the archive on-the-fly as we are processing the data that streams past. Also, strict conformance to the ZIP specification means that the uncompressed size of a single file can be a maximum of 4 GB.

An alternative exists that avoids these two limitations – the tar file format. The tar format contains a header for every file entry in the archive. If the POSIX (IEEE P1003.1) variant is used, then extended header information is possible, allowing for example virtually unlimited file sizes. Another benefit of the tar format is that the headers are simple and in ASCII, allowing compact implementations for languages that don't have built-in tar support.

7.3.2 Data Structures

For universal access to non-trivial amounts of possibly hierarchical meta-data, human and machine readable data structures are desired for ancillary meta data. It should be conceptually and programmably easy for those data structures to be mapped directly (or deserialized) to internal language-specific data structures for a given component's implementation language.

A minimal requirement for data structures would include scalars (e.g., strings, numbers) and vectors (e.g., arrays, lists) of those scalars. It would also be nice to have associative arrays (also known as hash tables) – still better would be arbitrary recursive composition of scalars, vectors, and associative arrays.

One heavy-weight serialization possibility would be XML. However even for languages that have built-in XML support, additional code would need to be written for translating to and from XML to the domain-specific meta-data data structure that the component developer needs to work with. XML should not be ruled out, since there is much to be gained by third party tool support such as editors, validators, transformation engines, etc.

However, there are also simpler, lighter-weight alternatives, which could be attractive in the case that the component developer must implement parsing and formatting herself (i.e. because she is using an unsupported language). Some possible examples include JSON [Crockford, 2006], YAML [Ben-Kiki et al., 2004], or S-expressions [Rivest, 1997]. Often, sequences needed for escaping non-ASCII characters are different in various languages – these kinds of details should be well-defined for the chosen serialized meta-data format.

7.3.3 Algorithm Encapsulation by Callback

One of the primary goals of the design is to try to limit the irrelevant system knowledge that a component developer must learn before she can develop a component. It is hoped that the vast majority of the code she writes concerns itself with the specialized processing handled by that component - rather than I/O, serialization/de-serialization, format conversion, bookkeeping, or other meta-data maintenance tasks. One method for achieving this could be for the framework to provide callback hooks in strategic framework locations. For example, in the framework code, every time a frame in a running stream is decoded, it might try to call a user-defined frame callback, passing as a parameter the already-deserialized, currently-accumulated, meta-data that might be helpful while processing that frame. Additionally a long-lived “scratchpad” might be made available, so that the component author can maintain state between calls.

7.3.4 Parallelization

The flexibility of a serial pipeline while designing and experimenting with processing chains should not be allowed to preclude parallelization. The most straightforward way to achieve this happens automatically on multi-core machines – each component stage in the processing chain will likely map to a different CPU. On single CPUs, this could be achieved transparently by using a secure remote execution tool (`ssh`) to execute certain pipeline stages remotely. However there are many problems with this approach: there is likely to be a slowest stage in the chain causing all components to wait on it. Also, the cost of transferring data off-machine and back between stages is likely to be inefficient.

Another approach would be to partition the data and use the same processing chain on each subset in its own CPU. This could be handled semi-automatically by a scatter/gather [Hohpe and Woolf, 2003]

meta-component. A scatter component could partition the data and transfer it to compute nodes along with (possibly a subset of) a full command pipeline. Logically, the gather component is responsible for gathering all of the scattered processing/data back into a single result, although practically, this functionality is likely best implemented directly in the scatter component.

7.4 Examples

The APEX airborne imaging spectrometer, developed within the framework of the European Space Agency's (ESA) funding scheme PRODEX, is currently undergoing construction and system integration [Nieke et al., 2004]. In addition to serving as an airborne surface radiance measurement device, APEX is also intended to act as a simulator, calibrator, and validation experiment supporting imaging varied spectroscopy application development [Itten et al., 1997], [Nieke et al., 2005].

A model of the instrument has been defined and a corresponding inverse model data processor has been implemented (see Figure 7.1) for producing at-sensor radiance [Schläpfer et al., 2004], [Kaiser et al., 2004]. This processor has been run against simulated data based on the instrument's currently known characteristics – the unknowns provided by specifications. However this initial processing prototype has been constructed in the typical monolithic way.

If the proposed hyperspectral streaming toolkit were available today, the processor could be decomposed into separate standalone components operated as stages in a pipeline (alternatively known as *filters*). Data processing could then be configured at run time using a paradigmatic command pipeline:

```
tar -cf - frames.meta frames.data | unbin | unsmear | unintegrate
  | darkcurrent | gain | badpix | resample | rebin > cube-rad.tar
```

This would enable desirable features such as the following:

- Data converters into and out of native format could be pre-pended/appended

```
... | convert -from hdf frames | ... |
  convert -to envi frames | ...
```

- “Quirks”¹ for different instruments could be encapsulated (and contributed by others)

```
... | hyperion_destripe | hyperion_rotate 2.1 | ...
```

- Generic GUIs for feedback could be placed at any stage in the pipeline

```
... | waterfall -rgb 4,10,30 frames | ... |
  waterfall -rgb 21,43,65 frames | ...
```

- Since each stage is standalone, any stage could be written in any computer language, increasing the chances of external contributions. As an example, someone might notice that many of the transformations are instances of generic affine transformations, and for example darkcurrent and gain transformations could be replaced by a single generic affine stage (by composing two transformation matrices):

```
... | affine -keys darkcurrent,gain frames.meta | ...
```

¹Quirks in this sense refers to machine-encoded information that shows how to work around unusual behavior for particular products in order to make their results closer to an expected form.

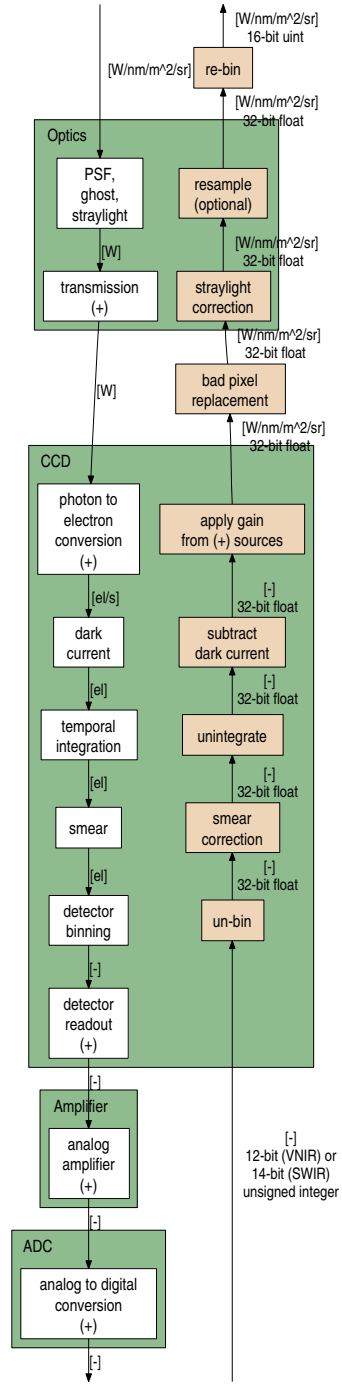


Figure 7.1: APEX Forward instrument model (left) and inverse processing model (right)

- Multiple interchangeable implementations of the same stage could be chosen at run-time e.g. `isdas_smile_detect` vs `gao_smile_detect`.
- It would be possible to mix and match binary-only proprietary stages (perhaps for intellectual property reasons) with open source stages.
- Code parallelism could be dynamically introduced via tools such as `ssh` or `socketpipe` [Spinellis, 2005].

```
... | ssh node1 stage1 | ssh node2 stage2 | ...
```

- Data parallelism for more efficient distributed operation could be dynamically introduced using proposed `scatter(/gather)` meta stage.

```
scatter -byframe -n 400 image.tar -h {host1 host2 host3}
-l { ssh } -r { unbin | cholcol [...] | affine [...] |
badpixel | affine [...] | resample | rebin | affine [...] |
convert -to uint } -o image-lvl.tar
```

7.5 Related Work

Two comparably relevant systems are similar to the concept presented in this work. The most structurally similar is PBM [Poskanzer, 1998], and in particular its successor, NetPBM [Henderson, 1993], which introduced a higher order PAM file format that seems suited to supporting three dimensional data. However, there are four primary difficulties that disqualify this specific approach for dealing with hyperspectral data: (1) in the hyperspectral world, it is expected to be able to arbitrarily use at least three of possible three dimensional band interleavings – BSQ, BIL, or BIP (2) the careful tracking of scientific units should be made whenever data is transformed through a stage (3) there might be geocoding at the pixel level and (4) a potentially large amount of other associated meta-data (band centers, bad pixel maps, etc) precludes its inclusion in a fixed-sized inline header such as inherent in all PBM formats.

Another highly relevant system is the ISDAS toolkit [Staenz et al., 1998]. It took the Unix pipeline paradigm a little less strictly, and instead used it primarily as a runtime configuration metaphor. The end-user is presented with a graphical canvas and can add processing components as graph nodes. One then connects these nodes together with directed edges and, crucially, is not limited to only one input or output edge. Arbitrary directed acyclic graphs are allowed, although limited checking (e.g. vector vs scalar mismatches) is enforced on the component interfaces. After interactive experimentation, a particular graph can be saved as a template that can be re-loaded/re-run on demand. Many facets of this system meet our stated design goals. However, other than it being founded on top of a proprietary product near its end-of-life, three specific characteristics of this system reveal its unsuitability: (1) the system is tightly integrated into a GUI toolkit. Ignoring the quirks of this particular toolkit, it suffices to say that GUI toolkits have even shorter lifespans than programming languages. (2) component authors needed to know large parts of the system irrelevant to the implementation of their algorithm, (3) it is only possible to write components in one (no longer very popular) programming language. In spite of these shortcomings, the system was very innovative: the ability to visualize the resulting processing data flow remains a feature missing from most other solutions including the current proposal.

Feature	Remarks
in-line stream processing	The zip format has been evaluated and rejected because archive size must be known in advance and individual files in the archive are limited to 4GB. The tar format meets all requirements and additionally proves easier to implement.
multi-lingual components	The APEX prototype processor (written in IDL) has been adapted for streaming operation. Prototype code in Matlab, C, Perl and Java are able to decode the stream.
algorithm-centric	A functional <code>waterfall</code> component has been implemented in less than 100 lines of (Tcl) code.
parallel/distributed capability	Local parallelism: stand-alone stages are confirmed to be distributed automatically across multiple CPUs by the operating system. Code parallelism is possible using <code>ssh</code> and <code>socketpipe</code> but unlikely to provide much gain due to high volume data. Data parallelism concept is introduced using a <code>scatter/gather</code> meta-stage.

Table 7.1: Prototyped features deemed necessary for proving design goals

7.6 Current Status and Future Work

The concept as constrained by the design goals led to the prototyping of the critical features and is summarized in Table 7.1.

Consistent with the component-oriented paradigm first popularized by the Unix operating system, and later supported by its implementation in the PBM image processing tool kit, only a few specific components are implemented initially. These have been chosen to illustrate expected classes of possible components to come. Administrative classes of components are expected for mundane tasks such as conversion to and from existing data formats. Operational classes of components are expected to help with parallelization, reporting, and archiving tasks. However, the most important class of components are the actual processing stages. It is expected that initially, specialized components will be produced for individual instruments. It is hoped that over time similarities will be noticed and consolidated into generic stages, such as the suggested `affine` stage for an entire class of linear transformations, which can not only reduce code, but reduce processing by taking advantage of the insight that transformation matrices can be composed requiring only one pass over the data to effect multiple logical transformations. It is further hoped that as stages begin to be shared among multiple projects for multiple instruments, consensus and sets of best-practices might form to work toward consistent processing pipelines and the ordering of stages e.g. does spectral smile get corrected before keystone?

In addition to the specification of specific components, another conspicuously absent detail is the specific requirements (as opposed to the format) for meta-data. It is believed that this should be left open until experience is gained and feedback obtained from initial component implementers and end-users during practical use of the system.

Finally, the obvious development model for such an undertaking is the distributed open source model. The code for the current implementation, including code revision history is hosted as a google project [Brazile, 2007] which also supports issue tracking and an associated mailing list.

Acknowledgements

This work was supported in part by ESA/ESTEC contracts 16298/02/NL/US and 15449/01/NL/Sfe.

References

- [Aronsson, 1998] Aronsson, M. (1998). A review of the new AVIRIS data processing system. In 1998 AVIRIS Workshop.
- [Ben-Kiki et al., 2004] Ben-Kiki, O., Evans, C., and Ingerson, B. (2004). Yaml ain't markup language (yaml). [Online; accessed Dec-2006]. <http://yaml.org/spec/current.html>.
- [Brazile, 2007] Brazile, J. (2007). Portable hyperspectral image processing components. [Online; accessed Jan-2007]. <http://code.google.com/p/phm/>.
- [Brazile et al., 2005] Brazile, J., Kaiser, J. W., Schläpfer, D., Nieke, J., Schaepman, M. E., and Itten, K. I. (2005). Parallelization of APEX imaging spectrometer product generation. In Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 103–111, Warsaw, Poland.
- [Crockford, 2006] Crockford, D. (2006). The application/json media type for JavaScript Object Notation (JSON). [Online; accessed Dec-2006]. <http://www.ietf.org/rfc/rfc4627.txt>.
- [Fomferra and Brockmann, 2005] Fomferra, N. and Brockmann, C. (2005). Beam – the ENVISAT MERIS and AATSR toolbox. In Proceedings MERIS User Workshop, Frascati, Italy, ESA.
- [Henderson, 1993] Henderson, B. (1993). netpbm. [Online; accessed Sep-2005]. <http://sourceforge.net/projects/netpbm/>.
- [Hohpe and Woolf, 2003] Hohpe, G. and Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.
- [Itten et al., 1997] Itten, K. I., Schaepman, M., De Vos, L., Hermans, L., Schläpfer, H., and Droz, F. (1997). APEX - airborne prism experiment: A new concept for an airborne imaging spectrometer. In 3rd Intl. Airborne Remote Sensing Conference and Exhibition, volume 1, pages 181–188.
- [Kaiser et al., 2004] Kaiser, J. W., Schläpfer, D., Brazile, J., Strobl, P., Schaepman, M. E., and Itten, K. I. (2004). Assimilation of heterogeneous calibration measurements for the APEX spectrometer. In SPIE Sensors, Systems, and Next Generation Satellites VII, volume 5234, pages 211–220, Barcelona, Spain.
- [Katz and Shapiro, 1994] Katz, M. L. and Shapiro, C. (1994). Systems competition and network effects. Journal of Economic Perspectives, 8(2):93–115.
- [McIlroy, 1968] McIlroy, M. D. (1968). Mass produced software components. In Proc. NATO Software Engineering Conference, pages 138–155, Garmisch, Germany.

- [Nieke et al., 2005] Nieke, J., Itten, K., Debruyn, W., Kaiser, J., Schläpfer, D., Brazile, J., Meuleman, K., Kempeneers, P., Neukom, A., Schilliger, T., De Vos, L., Piesbergen, J., Gege, P., Suhr, B., Schaepman, M., Gavira, J., Ulbrich, G., and Meynart, R. (2005). The airborne imaging spectrometer APEX: From concept to realization. In Zagojiewski, editor, Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 67–74, Warsaw, Poland.
- [Nieke et al., 2004] Nieke, J., Itten, K. I., Kaiser, J. W., Schläpfer, D., Brazile, J., Debruyn, W., Meuleman, K., Kempeneers, P., Neukom, A., Feusi, H., Adolph, P., Moser, R., Schilliger, T., van Quickelberghe, M., Alder, J., Mollet, D., De Vos, L., Kohler, P., Meng, M., Piesbergen, J., Strobl, P., Schaepman, M. E., Gavira, J., Ulbrich, G., and Meynart, R. (2004). APEX: Current status of the airborne dispersive pushbroom imaging spectrometer. In Barnes, W. and Butler, J., editors, SPIE Earth Observing Systems IX, volume 5542, pages 109–116.
- [O'Reilly, 2006] O'Reilly, T. (2006). [Online; accessed Dec-2006]. http://radar.oreilly.com/archives/2006/08/programming_language_trends.html.
- [Poskanzer, 1998] Poskanzer, J. (1998). pbm – Portable Bitmap programs. [USENET; comp.sources.misc]. Volume 2, Issue 83.
- [Ritchie, 1979] Ritchie, D. (1979). The evolution of the Unix time-sharing system. In Language Design and Programming Methodology, pages 25–36.
- [Rivest, 1997] Rivest, R. L. (1997). S-Expressions. [Online; accessed 19-January-2007]. <http://theory.lcs.mit.edu/~rivest/sexp.txt>.
- [Schläpfer et al., 2004] Schläpfer, D., Kaiser, J. W., Brazile, J., Schaepman, M. E., and Itten, K. I. (2004). Calibration concept for potential optical aberrations of the APEX pushbroom imaging spectrometer. In SPIE Sensors, Systems, and Next Generation Satellites VII, volume 5234, pages 221–231.
- [Spinellis, 2005] Spinellis, D. (2005). Socketpipe. [Online; accessed Sep-2005]. <http://www.spinellis.gr/sw/unix/socketpipe/>.
- [Staenz et al., 1998] Staenz, K., Szeredi, T., and Schwarz, J. (1998). ISDAS - a system for processing/analyzing hyperspectral data. Canadian Journal of Remote Sensing, Vol. 24, No. 2:99–113.
- [tiobe, 2006] tiobe (2006). Programming community index. [Online; accessed Dec-2006]. <http://www.tiobe.com/tpci.htm>.

Chapter 8

Computation-based Improvements for Hyperspectral Data Processing - A Synthesis

8.1 Introduction

A good foundation has been established for earth-observing imaging spectroscopy over the last three decades. Basic needs have been met for instrument design, data collection, instrument characterization, calibration/validation, image geocoding, and atmospheric correction. Additionally, on the application side, motivating research across a wide range of fields has been identified, with new discoveries emerging regularly.

In addition to steady evolutionary improvement in already existing problem areas, there are also attempts to expand in new areas that draw on interdisciplinary collaboration with research in neighboring fields – such as computer science. To list a few examples, there have been investigations incorporating computer science research from sub-fields such as database modeling [Bojinski et al., 2003], neural networks [Benediktsson and Kanellopoulos, 1999], computer vision [Huertas et al., 1999], genetic algorithms [Siedlecki and Sklansky, 1989], computer graphics [Setoain et al., 2007], computational geometry [Bachmann et al., 2005], and distributed processing [Brazile et al., 2004].

In fact, the unifying purpose of the work in this thesis has been to further contribute to the interdisciplinary application of applied computer science toward advancing the state of the art in earth observing imaging spectroscopy. This has included computational improvements to a data acquisition throughput engineering problem, simplified distributed processing across the internet, SRF parameterization and distributed inverse modeling for supporting instrument characterization, and the design of a loosely-coupled, componentized set of utilities to promote longevity and easier collaboration in general purpose hyperspectral image processing. In particular, this thesis has investigated the problems which either had the highest priority at the time or were deemed likely to gain the largest benefit from the author's experience.

The above mentioned computer science sub-fields are by no means exhaustive. This borrowing from other vocations is a welcome trend. One could imagine consciously selecting new candidate fields for collaboration based on: potential political benefits (e.g. gateway to the

mainstream via support for monitoring of global warming), scientific-process benefits (e.g. engineering models replaced by physical-based models, more rigorously-defensible mathematical transformations), process benefits (e.g. tool-supported reproducible processes), and economic benefits (e.g. improved efficiency through better collaboration, more open and community-based models of development, and better knowledge and tool sharing of common solutions with neighboring fields).

However, what is probably the most important problem to focus on at this particular point in time is the opening up of and preservation of our current body of knowledge. Before the age of computers, most important theories, techniques and algorithms were by necessity describable in compact ways: papers with a few formulas, algorithms consisting a few tens of steps, processes described by a few tens of diagrams. However as many techniques and algorithms have come to be developed inseparably from the computer software that implements them, they have started to become complex enough that the only adequate documentation/specification is the software (source code) itself. This means it is increasingly more important that source code begins to make its way into scientific archives.

As the first generation of pioneers begin to make way for the next generation, now is the time to survey the current state of the art, consolidate collective knowledge, and most importantly, ensure that no processes, algorithms, tools or techniques retire with their corresponding original author/maintainers. In the publish or perish academic world, there is a danger of continuously adopting the “new” thing because it is new and therefore publishable. The danger of losing the old to obscurity is quite high especially if the “old” thing is only available “upon request” or on a proprietary basis. For the sake of science, all previously investigated methods (especially those that are effective under some circumstance) should remain available “forever” alongside new ones.

8.2 Synopsis

The research questions raised at the beginning of the implementation phase of the APEX project in the areas of data acquisition, distributed data processing, instrument characterization and software componentization of hyperspectral algorithms have been addressed in this thesis with the following results:

8.2.1 Data Acquisition

Analysis and design of the data throughput requirements and hardware capabilities led to prototypes that proved that the high data bandwidth requirements could be met through application of well-known parallel processing techniques such as asynchronous direct memory access (DMA) used by input and output media and double buffering in Input/Output subsystems to smooth out the expected bursts of data. A minimalistic custom-developed hardware interface card implementing a finite state machine, bounded buffering, and communication components was designed and implemented for interfacing with the image data acquisition component as well as ancillary sensor data component, which allowed aggregate data rates using software acquisition to fall within the range of feasibility on standard commodity hardware [Brazile et al., 2003]. Some of the functionality needed (e.g. collection of positioning and environmental meta data) were similar to those reported in other airborne hyperspectral imaging projects [Hansen et al., 1992], [Aronsson, 1998], but other needs were unique to the APEX design.

8.2.2 Distributed Data Processing

A minimalist set of required functionality was defined for a distributed processing “grid” tool that proved simple enough for non-specialists to setup and use with almost no configuration and little software or hardware requirements [Brazile, 2007]. This tool was deployed for solving multiple distributed computing problems during the life of the APEX project.

Although the “grid” tool focused on ease-of-use, in one common usage pattern (thousands of multi-minute job runs) its run-time efficiency measured favorably when compared to **Condor** [Litzkow et al., 1988] – the most popular grid tool for non-specialist scientific computing users. Fortunately for comparison purposes, it was possible to design the distributed MODTRAN runs needed for the look-up table needs of ATCOR [Richter, 2005b], [Richter, 2005a] in such a way that execution was easy to implement using a grid or a cluster. There were many insights to be gained from comparing the development, deployment and scientific results of the two approaches. Of the many potentially important evaluation characteristics used in the comparison of the grid vs cluster implementation of ATCOR look-up table generation, run-time performance was not as important as ease-of-development, accuracy of results, and availability of distributed resources/infrastructure and wall-clock turnaround. It is interesting to following more recent parallel and distributed implementations [Plaza et al., 2006], [Plaza et al., 2007], [Valencia et al., 2007].

8.2.3 Scene-based Instrument Characterization

An instrument’s Spectral Response Function (SRF) as distinct from the typically-used Gaussian model, was identified as a potential novel scene-based instrument characterization retrieval candidate. Through permutation-driven statistical modeling of an instrument with APEX performance, it was furthermore established that the discernibility of several theoretical SRF shapes is feasible over a varying range of targets, visibility, altitude, water vapor and aerosol models making use of perturbations caused by prominent, well-known atmospheric features [Brazile et al., 2006].

A further study showed that using statistical methods, SRF retrieval should be possible even when simultaneously attempting to retrieve related instrument parameters such as band center and bandwidth shifts [Brazile et al., 2007a]. Additionally, SRF retrieval method sensitivity was investigated using existing instrument models of lesser performance than APEX and an estimate of what degree of performance might be needed to retrieve such parameters in realistic situations was shown.

Finally, a particular SRF parameterization based on number of binned channels and a ratio of sub channel FWHM to spectral sampling interval has been introduced as a potentially useful parameterization for retrieval purposes because of continuity and shape coverage characteristics.

Based on these two investigations, it remains to be seen how well this additional retrieval will function when it is operationally added to the already simultaneous retrieval of bandwidth and band center shifts [Neville et al., 2007] in the ISDAS [Staenz et al., 1998] processing system.

8.2.4 Componentization of Hyperspectral Processing Algorithms

Based on the experience gained during hyperspectral data processing over multiple projects using multiple programming systems, an analysis and synthesis was attempted to distinguish

common shared needs of hyperspectral data processing software developers. A componentized processing framework and set of utilities (similar to a highly successful existing system for raster processing [Poskanzer, 1998]) was conceived for providing these perceived needs in a system-agnostic and therefore highly re-usable manner. Use cases of the proposed componentized framework were exercised toward development of processing chains for typical hyperspectral processing applications [Brazile et al., 2005]. As an additional system motivation, further components (inspired by similar existing tools e.g. [Spinellis, 2005], [Percival, 2005]) were conceived of for the distributed and/or parallel processing of the envisioned complete processing chains. Finally, prototype components of the envisioned system were implemented in multiple programming languages to prove the concept's feasibility [Brazile et al., 2007b].

8.3 Conclusions

The research performed within the scope of this thesis has made three potentially lasting contributions. The most general and therefore potentially widest reaching is the simplified tool for supporting distributed computing. With the proper exposure to potential users, it is felt that this contribution offers real benefits (at least within an important subclass of distributed computing) over the popular alternative – Condor. Potential users include not only scientists and researchers, but any group who wish to pool their geographically disperse computing resources toward working on a common problem. The technology used (HTTP, Java, REST) is likely to remain applicable within the next 10-15 years.

This leads to the second contribution – the concept for prolific hyperspectral processing components. By design, this contribution has a potentially longer lifespan, but the target audience – users of hyperspectral imaging data – is much smaller. It is hoped that by applying the common software engineering principle of decoupling, more components will be able to interoperate, algorithm developers will be free to use their language and operating system of choice, end-users will be able to compose components in novel ways, and components may outlive the language of their initial implementation. As with the first contribution, the effect of this contribution will rely on the exposure of these principles to the proper audience and the eventual acknowledgment of its value.

The third potentially lasting contribution of this thesis is possibly the longest lasting, but has the smallest audience. The potential scene-based retrieval of the spectral response function (SRF) shape has been introduced and explored for the first time in the scientific community by two journal papers included in this thesis. The impact of the precise knowledge of this particular instrument parameter on the quality of end-user data products is not yet as well known as with other parameters retrievable from scene-based imagery. The value of this work is significant in 1) extending the scope of retrievable parameters beyond bandwidth and band center calibration, 2) exploring useful continuous parameterizations of spectral response functions in terms of instrument hardware characteristics and 3) defining acceptable sets of parameters and their varying ranges for covering arguments about addressing typical targets.

8.4 Outlook

The value of trading processing time for gaining independence from labor-intensive calibration/characterization cycles is becoming increasingly convincing. The method outlined in this thesis for retrieving the SRF is novel and compelling, but it is possible that a method based on more rigorous modeling could yield more mathematically defensible results than the empirical

approach taken here. If the evaluation of the Tukey function had yielded better results, it would have been in this sense a better candidate.

Also, in addition to the known retrievable parameters such as band center, band width, and now SRF shape, other relevant instrument parameters should be investigated for possible scene-based retrieval. Largely unexplored possibilities include aberrations stemming from instrument-specific optical and electrical characteristics. In addition to investigating the potential for such characterizations merely because they may be retrievable, it would be worthwhile to attempt to quantify the potentially cumulative affects of classes of various correctable calibration errors across multiple instruments, in an attempt to prioritize research on the problems that are most likely to cause the greatest impact on the quality of the end product.

In another possible direction of investigation, instead of focusing on the instrument side of the problem, it may also be worthwhile to explore the reference side of the problem i.e., other than solar and molecular atmospheric features, what other measurable physical phenomena could potentially serve as a reference against which instrument characteristics could be deduced? It could be worthwhile investigating inherent sonic, gravitational, or magnetic information known to be emitted from or affected by the earth.

As argued in the later chapters of this thesis, almost as important as any algorithm/method itself, is an open, sharable implementation of that method, which can be evaluated and improved upon, independently from its published high-level algorithmic description. Much of the software produced in this thesis is published as open source in various venues. Yet even that is not enough. Many projects, even successful open source projects, languish from lack of awareness of them or from their implementation languages or operating systems going out of favor. If the language-neutral component-based framework described in Chapter 7 were already available, much of the software described in this thesis could be more properly offered as components that could better support science in the traditional sense – advancing the state of the art incrementally without redundant, duplicated effort, or the loss of valuable information solely because of its complex and/or proprietary nature.

Further work in this direction should be considered a near term priority (≤ 5 years) since it could immediately provide multiplicative computational improvements for hyperspectral data processing, just as a similar increase in productivity in recent years has occurred in the business world, largely attributed to the more efficient use of Information Technology exploiting the economic principle of the network effect [Katz and Shapiro, 1994].

References

- [Aronsson, 1998] Aronsson, M. (1998). A review of the new AVIRIS data processing system. In 1998 AVIRIS Workshop.
- [Bachmann et al., 2005] Bachmann, C., Ainsworth, T., and Fusina, R. (2005). Exploiting manifold geometry in hyperspectral imagery. IEEE Transactions on Geoscience and Remote Sensing, 43(3):441–454.
- [Benediktsson and Kanellopoulos, 1999] Benediktsson, J. and Kanellopoulos, I. (1999). Classification of multisource and hyperspectral data based on decision fusion. IEEE Transactions on Geoscience and Remote Sensing, 37(3):1367–1377.
- [Bojinski et al., 2003] Bojinski, S., Schaepman, M. E., Schläpfer, D., and Itten, K. I. (2003). SPECCHIO: a spectrum database for remote sensing applications. Computers & Geosciences, 29(1):27–38.
- [Brazile, 2007] Brazile, J. (2007). Low Fat Grid: A RESTful Grid Service for Non-Programmers. In Jazoon '07 – The International Conference on Java Technology, Zurich.
- [Brazile et al., 2005] Brazile, J., Kaiser, J. W., Schläpfer, D., Nieke, J., Schaepman, M. E., and Itten, K. I. (2005). Parallelization of APEX imaging spectrometer product generation. In Proceedings EARSeL 4th Workshop on Imaging Spectroscopy, pages 103–111, Warsaw, Poland.
- [Brazile et al., 2003] Brazile, J., Kohler, P., and Hefti, S. (2003). A software architecture for in-flight acquisition and offline scientific post-processing of large volume hyperspectral data. In Noumena, editor, Proceedings of the 10th Tcl/Tk Conference (Tcl/2003): July 29–Aug 2, 2003, Ann Arbor, Michigan, USA, Dexter, MI, USA. Noumena.
- [Brazile et al., 2006] Brazile, J., Neville, R. A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K. I. (2006). Scene-based spectral response function shape discernibility for the APEX imaging spectrometer. IEEE Geoscience and Remote Sensing Letters, 3:414–418.
- [Brazile et al., 2007a] Brazile, J., Neville, R. A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K. I. (2007a). Toward scene-based retrieval of spectral response functions for hyperspectral imagers using fraunhofer features. Canadian Journal of Remote Sensing. in press.
- [Brazile et al., 2007b] Brazile, J., Plaza, A., Schläpfer, D., Nieke, J., and Itten, K. I. (2007b). Streaming, language-neutral, hyperspectral image processing components. Software Practice and Experience. in preparation.
- [Brazile et al., 2004] Brazile, J., Schläpfer, D., Kaiser, J., Schaepman, M. E., and Itten, K. I. (2004). Cluster versus grid for large-volume hyperspectral image preprocessing. In SPIE Atmospheric and Environmental Remote Sensing Data Processing and Utilization: an End-to-End System Perspective, volume 5548, pages 48–58.

- [Hansen et al., 1992] Hansen, E., Larson, S., Novack, H., and Bennett, R. (1992). AVIRIS ground data processing system. In Third Annual JPL Airborne Geoscience Workshop, volume 1.
- [Huertas et al., 1999] Huertas, A., Nevatia, R., and Landgrebe, D. (1999). Use of hyperspectral data with intensity images for automatic building modeling. In Proc 2nd International Conference on Information Fusion, pages 680–687.
- [Katz and Shapiro, 1994] Katz, M. L. and Shapiro, C. (1994). Systems competition and network effects. Journal of Economic Perspectives, 8(2):93–115.
- [Litzkow et al., 1988] Litzkow, M., Livny, M., and Mutka, M. (1988). Condor—a hunter of idle workstations. In The 8th International Conference on Distributed Computing Systems, pages 104–111.
- [Neville et al., 2007] Neville, R. A., Sun, L., and Staenz, K. (2007). Spectral calibration of imaging spectrometers by atmospheric absorption feature matching. Canadian Journal of Remote Sensing, in press.
- [Percival, 2005] Percival, C. (2005). Pipelined http get utility. [Online; accessed Sep-2005]. <http://www.daemonology.net/phttpget/>.
- [Plaza et al., 2007] Plaza, A., Plaza, J., and Valencia, D. (2007). Impact of platform heterogeneity on the design of parallel algorithms for morphological processing of high dimensional image data. Journal of Supercomputing, 40(1):81–107.
- [Plaza et al., 2006] Plaza, A., Valencia, D., Plaza, J., and Martinez, P. (2006). Commodity cluster-based parallel processing of hyperspectral imagery. Journal of Parallel and Distributed Computing, 66(3):345–358.
- [Poskanzer, 1998] Poskanzer, J. (1998). pbm – Portable Bitmap programs. [USENET; comp.sources.misc]. Volume 2, Issue 83.
- [Richter, 2005a] Richter, R. (2005a). Atmospheric / topographic correction for airborne imagery. Technical Report DLR-IB 565-02/05, DLR, Wessling, Germany.
- [Richter, 2005b] Richter, R. (2005b). Atmospheric / topographic correction for satellite imagery. Technical Report DLR-IB 565-01/05, DLR, Wessling, Germany.
- [Setoain et al., 2007] Setoain, J., Prieto, M., Tenllado, C., Plaza, A., and Tirado, F. (2007). Parallel morphological endmember extraction using commodity graphics hardware. IEEE Geoscience and Remote Sensing Letters, page in press.
- [Siedlecki and Sklansky, 1989] Siedlecki, W. and Sklansky, J. (1989). A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters, 10(5):335–347.
- [Spinellis, 2005] Spinellis, D. (2005). Socketpipe. [Online; accessed Sep-2005]. <http://www.spinellis.gr/sw/unix/socketpipe/>.
- [Staenz et al., 1998] Staenz, K., Szeredi, T., and Schwarz, J. (1998). ISDAS - a system for processing/analyzing hyperspectral data. Canadian Journal of Remote Sensing, Vol. 24, No. 2:99–113.
- [Valencia et al., 2007] Valencia, D., Plaza, A., Martinez, P., and Plaza, J. (2007). Parallel processing of high-dimensional remote sensing images using cluster computer architectures. International Journal of Computers and their Applications, 14(1):23–34.



Curriculum Vitae

SURNAME Brazile
GIVEN NAME Paul Jason
DATE OF BIRTH 14 February 1968
PLACE OF BIRTH Dallas, Texas, USA
NATIONALITY USA

EDUCATION
1983 – 1986 DENTON HIGH SCHOOL (Denton, Texas) High school diploma.
1986 – 1991 UNIVERSITY OF NORTH TEXAS (Denton, Texas) Bachelor of Arts in
Computer Science (cum laude), minor in Mathematics.
1991 – 1994 UNIVERSITY OF TEXAS (Austin, Texas) Master of Science in Computer
Science, minor in Geography.
2002 – 2007 UNIVERSITY OF ZURICH (Zurich, Switzerland) Doctorate of Natural
Sciences, Supervision by Prof. Dr. Klaus I. Itten, Remote Sensing Lab-
oratories, Department of Geography, University of Zurich (Switzerland).

WORK EXPERIENCE
1988 – 1989 UNIVERSITY OF NORTH TEXAS, DEPT. OF COMPUTER SCIENCE
(Denton, Texas) *Systems administrator*. Ported and maintained re-
search software on various BSD and System V UNIX machines.
1989 – 1990 IEX CORPORATION (Richardson, Texas) *Programmer*. Developed se-
rial port performance evaluation software for newly developed AT&T
computer hardware and maintained a locally developed SCCS-based
configuration management system.
1991 LOCKHEED MISSILES AND SPACE COMPANY, INC (Austin, Texas) *Pro-
grammer*. Introduced configuration management and automatic docu-
mentation tools for an X/Motif cartographic charting and geopositioning
application.
1991 – 1994 UNIVERSITY OF TEXAS, DEPT. OF MATHEMATICS (Austin, Texas) *Op-
erating Systems Specialist*. Led a small team of system administrators
for the maintenance of the departmental UNIX network of 75 Sun and
NeXT workstations.
1994 – 1996 DSC COMMUNICATIONS CORPORATION (Plano, Texas) *Software En-
gineer*. Designed and implemented operating system specific locking,
error detection, and auditing primitives for an embedded telecommuni-
cations product.
1996 – 1998 AMPERSAND INC (Boston, Massachusetts) *Software Engineer*. Con-
tract tasks including distributed operating system kernel development,
UNIX device driver maintenance, a port of the MATLAB runtime to an
embedded supercomputer, and GIS database query middleware.
1998 – Present NETCETERA AG (Zurich, Switzerland) *Senior Software Engineer*. De-
sign and implementation of web-based and other network applications
and services.

Bibliography

The following list summarizes all communications by Jason Brazile published in the framework of this thesis.

Peer Reviewed Articles

Brazile, J., Plaza, A., Schläpfer, D., Nieke, J., and Itten, K.I., Streaming, Language-Neutral, Hyperspectral Image Processing Components, *Software Practice and Experience*, Wiley, 2007, in preparation.

Brazile, J., Richter, R., Schläpfer, D., Schaepman, M.E., and Itten, K.I., Cluster versus Grid for Operational Generation of ATCOR's MODTRAN-based Look Up Tables, *Parallel Computing*, Elsevier, 2005, in review.

Brazile, J., Neville, R.A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K.I., Toward Scene-Based Retrieval of Spectral Response Functions for Hyperspectral Imagers Using Fraunhofer Features, *Canadian Journal of Remote Sensing*, Canadian Aeronautics and Space Institute, 2007, in press.

Brazile, J., Neville, R.A., Staenz, K., Schläpfer, D., Sun, L., and Itten, K.I., Scene-Based Spectral Response Function Shape Discernibility for the APEX Imaging Spectrometer, *IEEE Geoscience and Remote Sensing Letters*, IEEE, 3, 414–418, 2006.

Plaza, A., Benediktsson, J.A., Boardman, J., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., Marconcini, M., Tilton, J.C., and Trianni, G., *Recent Advances in Techniques for Hyperspectral Image Processing*, Remote Sensing of Environment, 2007, in review.

Nieke, J., Schläpfer, D., Dell'Endice, F., Brazile, J., and Itten, K., Uniformity of imaging spectrometry data products - the APEX approach, *IEEE Transactions on Geoscience and Remote Sensing*, 2007, in review.

Painter, T.H., Schaepman, M.E., Schweitzer, W., and Brazile, J., Recent Advances in Reflectance Spectroscopy-based Differentiation of Natural and Artificial Man-Made Substances, *Journal of Improbable Research*, in press, 2007.

Dorigo, W., Zurita-Milla, R., de Wit, A., Brazile, J., Singh, R., and Schaepman, M., A review on reflective remote sensing and data assimilation techniques for enhanced agroecosystem modeling, *Journal of Applied Earth Observation and Geoinformation*, 2006, in press.

Conference Contributions

Brazile, J., Low Fat Grid: A RESTful Grid Service for Non-Programmers, Jazoon '07 – The International Conference on Java Technology, 2007, CD-ROM.

Brazile, J., Kaiser, J.W., Schläpfer, D., Nieke, J., Schaepman, M.E., and Itten, K.I., Parallelization of APEX imaging spectrometer product generation, Proceedings EARSel 4th Workshop on Imaging Spectroscopy, 103–111, 2005.

Brazile, J., Schläpfer, D., Kaiser, J., Schaepman, M.E., and Itten, K.I., Cluster versus Grid for large-volume hyperspectral image preprocessing, SPIE Atmospheric and Environmental Remote Sensing Data Processing and Utilization: an End-to-End System Perspective, 5548, 48–58, 2004.

Brazile, J., Kohler, P., and Hefti, S., A Software Architecture for In-flight Acquisition and Offline Scientific Post-Processing of Large Volume Hyperspectral Data, Proceedings of the 10th Tcl/Tk Conference (Tcl/2003): July 29–Aug 2, 2003, Ann Arbor, Michigan, USA, Noumena, 2003, [CD-ROM].

Brazile, J., Schaepman, M., Schläpfer, D., Kaiser, J., and Itten, K., A Beowulf-Style Cluster Processing Concept for Large Volume Imaging Spectroscopy Data, 7th World Multiconference on Systemics, Cybernetics and Informatics, IIS, X, 17–20, 2003.

Dangel, S., Schaepman, M., Brazile, J., Petitcolin, F., Su, B., Briottet, X., Gloor, M., Moreno, J., and Itten, K., System architecture and design for a SPECTRA mission end-to-end simulator, ESA/ESTEC, WPP-225, 2004, [CD-ROM].

Dangel, S., Brazile, J., Petitcolin, F., Kneubuehler, M., Schaepman, M.E., and Itten, K.I., The design and prototyping of the SPECTRA simulator architecture, Proceedings EARSel 4th Workshop on Imaging Spectroscopy, 2005.

Debruyn, W., Reusen, I., Kempeneers, P., Deronde, B., Itten, K., Schaepman, M., Schläpfer, D., Brazile, J.K.J., and Meuleman, K., The airborne imaging spectrometer APEX (Airborne PRISM Experiment): Overview and status questions, 23rd EARSel Symposium, Mill Press, xiv, 546, 2003, [CD-ROM].

Kaiser, J.W., Nieke, J., Schläpfer, D., Brazile, J., and Itten, K.I., Sensitivity study for atmospheric applications with APEX, Proceedings EARSel 4th Workshop on Imaging Spectroscopy, 2005.

Kaiser, J.W., Schläpfer, D., Brazile, J., Strobl, P., Schaepman, M.E., and Itten, K.I., Assimilation of Heterogeneous Calibration Measurements for the APEX Spectrometer, SPIE Sensors, Systems, and Next Generation Satellites VII, 5234, 211–220, 2004.

Nieke, J., Itten, K., Debruyn, W., Kaiser, J., Schläpfer, D., Brazile, J., Meuleman, K., Kempeneers, P., Neukom, A., Schilliger, T., De Vos, L., Piesbergen, J., Gege, P., Suhr, B., Schaepman, M., Gavira, J., Ulbrich, G., and Meynart, R., The Airborne Imaging Spectrometer APEX: From Concept to Realization, Proceedings EARSel 4th Workshop on Imaging Spectroscopy, 67–74, 2005.

Nieke, J., Itten, K.I., Kaiser, J.W., Schläpfer, D., Brazile, J., Debruyn, W., Meuleman, K., Kempeneers, P., Neukom, A., Feusi, H., Adolph, P., Moser, R., Schilliger, T., van Quickelberghe, M., Alder, J., Mollet, D., De Vos, L., Kohler, P., Meng, M., Piesbergen, J., Strobl, P., Schaepman, M.E., Gavira, J., Ulbrich, G., and Meynart, R., APEX: Current status of the airborne dispersive pushbroom imaging spectrometer, SPIE Earth Observing Systems IX, 5542, 109–116, 2004.

Nieke, J., Itten, K.I., Kaiser, J.W., Schläpfer, D., Brazile, J., Debruyn, W., Meuleman, K., Kempeneers, P., Neukom, A., Feusi, H., Adolph, P., Moser, R., Schilliger, T., van Quickelberghe,

M., Alder, J., Mollet, D., De Vos, L., Kohler, P., Meng, M., Piesbergen, J., Strobl, P., Schaepman, M.E., Gavira, J., Ulbrich, G., and Meynart, R., APEX: Status of the airborne dispersive pushbroom imaging spectrometer, Proc. DGPF Jahrestagung, 2004, [CD-ROM].

Nieke, J., Kaiser, J.W., Schläpfer, D., Brazile, J., Itten, K.I., Strobl, P., Schaepman, M.E., and Ulbrich, G.J., Calibration methodology for the airborne dispersive pushbroom imaging spectrometer (APEX), SPIE Sensors, Systems, and Next-Generation Satellites VIII, 5570, 445-452, 2004.

Plaza, A., Benediktsson, J.A., Boardman, J., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., Marconcini, M., Tilton, J.C., and Trianni, G., Processing of Hyperspectral Image Data: Advanced Technologies, Proc. IGARSS 2006, IEEE, 1974-1978, 2006.

Schaepman, M., Dangel, S., Kneubühler, M., Schläpfer, D., Bojinski, S., Brazile, J., Kötz, B., Strub, G., Kohler, R., Popp, C., Schopfer, J., and Itten, K., Quantitative Field Spectroscopic Measurement Instrumentation and Techniques, Proceedings 1st EPFS Workshop on Field Spectrometry, 2002, 13 pp., [CD-ROM].

Schaepman, M., Schläpfer, D., Brazile, J., and Bojinski, S., Processing of large-volume airborne imaging spectrometer data: the APEX approach, SPIE Imaging Spectrometry VIII, 4816, 72-79, 2002.

Schaepman, M., Schläpfer, D., Kaiser, J., Brazile, J., and Itten, K., Land Application Driven Performance Requirements for Airborne Imaging Spectroscopy, Geophys. Res. Abstracts, 5, 12386, 2003.

Schaepman, M., Schläpfer, D., Kaiser, J., Brazile, J., and Itten, K., APEX - Airborne Prism Experiment: Dispersive Pushbroom Imaging Spectrometer for Environmental Monitoring, Proc EARSeI 3rd Workshop on Imaging Spectroscopy, 2003.

Schaepman, M., Itten, K., Schläpfer, D., Brazile, J.K.J., Debruyn, W., Neukom, A., Feusi, H., Adolph, P., Moser, R., Schilliger, T., De Vos, L., Brandt, G., Kohler, P., Meng, M., Piesbergen, J., Strobl, P., Gavira, J., Ulbrich, G., and Meynart, R., APEX: Current Status of the Airborne Dispersive Pushbroom Imaging Spectrometer, SPIE Sensors Systems and Next-Generation Satellites VII, 5234, 202-210, 2004.

Schläpfer, D., Kaiser, J.W., Brazile, J., Schaepman, M.E., and Itten, K.I., Calibration concept for potential optical aberrations of the APEX pushbroom imaging spectrometer, SPIE Sensors, Systems, and Next Generation Satellites VII, 5234, 221-231, 2004.

Schläpfer, D., Kaiser, J.W., Nieke, J., Brazile, J., and Itten, K.I., Modeling and Correcting Spatial Non-Uniformity of the APEX Pushbroom Imaging Spectrometer, 13th Annual JPL Airborne Earth Science Workshop, 11, 2004.

Ulbrich, G., Meynart, R., Nieke, J., Itten, K., Kaiser, J., Schläpfer, D., Brazile, J., Debruyn, W., Meuleman, K., Kempeneers, P., Neukom, A., Feusi, H., Adolph, P., Moser, R., Schilliger, T., van Quickelberghe, M., Alder, J., Mollet, D., De Vos, L., Kohler, P., Meng, M., Piesbergen, J., Strobl, P., Schaepman, M., and Gavira, J., APEX - Airborne Prism Experiment: The realization phase of an airborne imaging spectrometer, SPIE Sensors, Systems, and Next-Generation Satellites VIII, 5570, 2004.

Teillet, P., Fedosejevs, G., Gauthier, R., Gibson, J., Hawkins, R., Lukowski, T., Neville, R., Staenz, K., Toutin, T., Touzi, R., White, H., Wolfe, J., Brazile, J., Carbonneau, Y., Chenier, R., Filfil, R., Murnaghan, K., Nedelcu, S., Short, N., Sun, L., and Yue, B., Recent advances in data calibration and standardisation in support of sustainable development of natural resources, Proc. IGARSS 2005, 6, 4160-4163, 2005, <http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=10226>.

Reports

Dangel, S., Petitcolin, F., Brazile, J., Miesch, C., Poutier, L., Jia, L., Moreno, J., Gloor, M., Kneubuehler, M., Schaepman, M., and Itten, K., SPECTRA end-to-end simulator : Final Report ESA, RSL, 2005, [CD-ROM], http://www.esa.int/esapub/bulletin/bulletin125/bull125_publicat.pdf.

Published Software

Brazile, J., Low Fat Grid, Google, 2006, [Online; accessed Nov-2006], <http://code.google.com/p/lowfatgrid/>.

Brazile, J., Grid Hack, OSTG, 2005, [Online; accessed Sep-2005], <http://sourceforge.net/projects/ghack/>.

Brazile, J., MODTRAN-compatible SRF Generator, The Mathworks Inc., 2005, [Online; accessed Jan-2005], <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8723>.

Brazile, J., and Dell'Endice, F., APEX Imaging Spectrometer Tools, OSTG, 2006, [Online; accessed Dec-2006], <http://sourceforge.net/projects/apex-esa/>.