ACM Siggraph96 Course #16

Visualizing Scientific Data and Information:
Focusing on the Physical and Natural Sciences

Course Organizer:
Theresa Marie Rhyne
Lockheed Martin Services Group
U.S. EPA Scientific Visualization Center

Instructors:
Mike Botts
University of Alabama in Huntsville

Bill Hibbard
University of Wisconsin at Madison

Lloyd Treinish
IBM T.J. Watson Research Center

August 5, 1996

Abstract: This course focuses on the application of visualization tools and interactive techniques for the examination and interpretation of scientific data sets. Highly illustrative atmospheric, oceanographic and geographic examples are demonstrated in real time. The process of developing effective visualization paradigms for supporting high speed networking, database management, heterogenous computing platforms, user interface design, collaborative computing, science education and the implementation of animation techniques are highlighted. The convergence of visualization methods with the World Wide Web and the relationship between animation techniques and scientific information exploration are also discussed.
Introduction

The visual presentation and examination of physical and natural sciences data often require merging image processing methods with computer-generated color displays. Networking and World Wide Web tools also assist in this information exploration. Frequently there is a need for the integration of other computational technologies with visualization methods. These include:

a) the integration of terabyte or gigabyte distributed scientific databases and digital libraries with visualization systems;

b) the display and interpretation of data and information using statistical analyses, cartography, Computer Aided Design (CAD) and Geographic Information Systems (GIS) techniques in conjunction with visualization systems;

c) the design of visualization tools, user interfaces, and animations which support the specific needs of scientists, policy analysts, regulators, educators and the general public;

d) the development of collaborative computing tools which allow the integration of multi-disciplinary data and information (e.g. atmospheric, oceanographic, and geographic) into visualization systems to foster cross-disciplinary exploration and communications.

This full day course explores these issues with illustrative examples of visualization software and animations designed to support the examination of scientific data and information. Each of the instructors has prepared a visualization demonstration based on their own customized software tools which will (hopefully) be executed (in real time) on a workstation during the course.

We have divided these course notes into four sections and have included the demonstration discussions (case studies) at the end. During the delivery of the course, each section presented will be followed by a case study demonstration. To clarify this matter, we have included both a Table of Contents for the Course Notes and an Outline of the Presentation of the Course.

We hope that you enjoy attending this course and that these notes are insightful to you. Each of us learned a great deal and expanded our own horizons in collaborating together on this course.

Theresa Marie Rhyne
Bill Hibbard
Lloyd Treinish
Mike Botts
Table of Contents for the Course (#16) Notes

I. Classifying and Modeling Data in the Physical and Natural Sciences

II. Techniques for Examining Multiple Data Sets and Solutions for Data Management

III. Collaborative Computing and Integrated Decision Support Tools for Scientific Visualization

IV. Maintaining Interactivity in Visualizing Large Data Sets

Case Study #1 Visualizing Photochemical Model Data (Examining an Air Pollution Model)

Case Study #2 The InterUse Experiment: interactive tools for Geolocation and Visual Comparison

Case Study #3 Correlative Visualization Techniques for Disparate Data

Case Study #4 Examining Earth Sciences Data in Real Time (VIS-5D and VIS-AD for Steering Earth and Space Science Computations)
ACM Siggraph Course #16: Visualizing Scientific Data and Information:  
Focusing on the Physical and Natural Sciences  
Course Outline -- (August 5, 1996)

Introduction/Course Organization Remarks: Rhyne - 5 minutes

Topic #1: Classifying and Modeling Scientific Data  
(Bill Hibbard - 45 minutes)

Case Study/Hands-on Demo for Topic 1: (30 minutes)  
Photochemical Modeling (Air Pollution) - Rhyne  
(AVS & WWW demonstrations for examining air pollution control, policy analysis and decision making.)

Morning Break

Topic #2: Techniques for Examining Multiple Data Sets and Solutions for Data Management  
(Lloyd Treinish - 45 minutes)

Case Study/Hands-on Demo for Topic 2: (30 minutes)  
The InterUse Experiment: an interactive tool for Geolocation and Visually comparing data - Botts

Lunch

Topic #3: Collaborative Computing and Integrated Decision Support Tools for Scientific Visualization  
(Theresa Rhyne - 45 minutes)

Case Study/Hands-on Demo for Topic 3: (30 minutes)  
Correlative Visualization Techniques for Disparate Data - Treinish  
(IBM Data Explorer based demonstration for looking at ozone depletion and its relationship to other characteristics of the earth's atmosphere.)

Afternoon Break

Topic #4: Maintaining Interactivity in Visualizing Large Data Sets  
(Mike Botts - 45 minutes)

Case Study/Hands-on Demo for Topic 4: (30 minutes)  
Interactively Visualizing and Steering Computations - Hibbard  
(Vis5D and/or VisAD demonstration for examining atmospheric and oceanographic models and data sets.)

Wrap-Up Discussion: (All Instructors) (10 minutes)
Speakers' Biographical Information:

Mike Botts: University of Alabama in Huntsville
Senior Research Scientist
University of Alabama in Huntsville
postal address: Earth System Science Lab
Global Hydrology and Climate Center
University of Alabama in Huntsville
Huntsville, Alabama 35899
phone: 205-922-5760
fax: 205-922-5755
email: botts@atmos.uah.edu

Mike Botts is a Senior Research Scientist in the Earth System Science Laboratory at the University of Alabama in Huntsville. In 1992, he served on temporary assignment, at NASA Headquarters, to evaluate the state of scientific visualization for NASA's EOS Mission. Botts' report, entitled "The State of Scientific Visualization with Regard to the NASA EOS Mission to Planet Earth", established many of the guidelines for visualization requirements for NASA EOS. He is presently Principal Investigator on an EOS/Pathfinder grant (The Interuse Experiment) and is responsible for directing the interuse activities of three teams of developers at the University of Alabama in Huntsville and NASA/JPL. Botts is also Co-Investigator on a grant to extend the JPL LinkWinds visualization package to meet the demands of EOS, particularly with regard to geolocation of data and comparative analysis of disparate data.

Bill Hibbard: University of Wisconsin at Madison
Researcher
Space Science and Engineering Center
University of Wisconsin-Madison
postal address: SSEC, 1225 W. Dayton St., Madison, WI 53706
phone: 608-263-4427
fax: 608-263-6738
email: whibbard@macc.wisc.edu

Bill Hibbard is a Scientist at the Space Science and Engineering Center of the University of Wisconsin at Madison. He is the principal author of the 4-D McIDAS system, Vis5D and VisAD. The 4-D McIDAS system, begun in 1984, is an early effort to apply three-dimensional animated graphics to earth science data. The Vis5D system, available by anonymous ftp, extended this in 1988 to interactive 3-D animations of numerical weather simulations, using high-performance graphics workstations. The VisAD system provides interactive visualization of scientific algorithms, through a technique for deriving graphical depictions for algorithm data types. The Vis5D and VisAD systems have been adapted to run in distributed computing environments.
Lloyd A. Treinish is a research staff member in the Data Explorer group at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY. He works on techniques, architectures and applications of data visualization and methods of data management for a wide variety of scientific disciplines with a particular focus on earth, environmental and space sciences. His research interests range from computer graphics, data storage structures, data representation methodologies, data base management, computer user interfaces, and data analysis algorithms to middle atmosphere electrodynamics, planetary astronomy and climatology. In particular, he is interested in generic or discipline-independent techniques for the storage, manipulation, analysis and display of data, and has, for example, applied these ideas to the study of global atmospheric dynamics and ozone depletion.

Theresa Marie Rhyne is currently a Lead Scientific Visualization Researcher and is responsible for the Research and Development activities at the U.S. EPA Scientific Visualization Center. She is employed by Lockheed Martin Technical Services. From 1990 - 1992, she was the technical leader of the U.S. EPA Scientific Visualization Center and was responsible for building the Center since its founding in 1990. Her research interests include visualization toolkit development, collaborative computing in a networked environment and the integration of geographic information systems with scientific visualization methods. She is also a practicing fine artist and art educator in computer graphics. Rhyne authored the "Portrait of a Computer Artist" discussion which appears in the 1991 edition of the ACM/SIGGRAPH Computer Graphics Career Handbook.
Visualization in the Physical and Natural Sciences is distinguished by the need to deal with data sets that:

1. Are volumetric, time varying and multi-variant. Computer simulations typically sample physical space at a dense 3-D grid of points, sample time at a regular sequence, and include values for ten or more different fields (e.g., temperature, pressure, chemical concentrations, wind or current velocities). Some sensors are capable of sampling physical space at dense 3-D sets of points, scanning repeatedly over time, and measuring several different fields (e.g., radar or sonar reflectivity, radial velocity from Doppler shift, polarization).

2. Are produced by a large variety sources. Raw data are remotely sensed by a variety of satellite sensor types, radars, sonars, and lidars (laser radars). Direct measurements are taken from surface stations, airplanes, ships, buoys and balloons. Computers models simulate the environment, and data are also inferred from census reports and other geographical information sources. This variety of data dictates a need to deal with a wide variety of data geometries and data sampling and error characteristics. These issues are compounded when dealing with environmental science research problems which attempt to merge air, water and subsurface data into single visualization presentations.

3. Are very large. For example, the archive of GOES satellite data contains 120 terabytes, and scientists really do need access to this entire data set in order to understand climate change. NASA’s Earth Observing System is planned to generate one terabyte of data per day. Over half of all the 9 track magnetic tapes ever produced are used to store seismic data by oil companies.
Physical and chemical models of the atmosphere and oceans sample 3-D space on a grid that is topologically regular but geometrically irregular:

Models calculate many different fields (e.g., temperature, pressure, humidity) at each grid point. However, different fields may be on staggered grids. Here is a view looking down on a 2-D horizontal slice of a 3-D grid:

\[
\begin{array}{cccccccc}
T & U & T & U & T & U & T & T \\
V & V & V & V & V & V & V & V \\
T & U & T & U & T & U & T & T \\
V & V & V & V & V & V & V & V \\
T & U & T & U & T & U & T & T \\
V & V & V & V & V & V & V & V \\
T & U & T & U & T & U & T & T
\end{array}
\]

TEMPERATURE, PRESSURE, ETC AT GRID POINTS MARKED "T"

NORTH-SOUTH WIND AT GRID POINTS MARKED "V"

EAST-WEST WIND AT GRID POINTS MARKED "U"

Field values are averages of small volumes rather than point samples. Grid point locations are the centers of these volumes.

Spectral models do not sample fields spatially at all. Rather, they work with values that are spatial integrals, where each spectral component has a different spatial weighting function (e.g., spherical harmonics).
Models may nest small high-resolution grids inside large low resolution grids. Here is a view looking down on a 2-D horizontal slice of a 3-D grid:

A radar samples 3-D space on a set of concentric cones:

at each elevation angle, radar sweeps out a cone

each ray from radar is divided into samples along length

location of radar

Radar values are averages over small spatial regions.

An aircraft collects direct measurements (e.g., by thermometer) along a 1-D flight path embedded in 3-D space:

Aircraft values are averages along short segments of the flight path.
A satellite image is a 2-D array of radiances - each radiance is a complex integral of physical values along a vertical column of atmosphere:

Satellites measure radiances in several spectral channels. Each channel integrates vertical columns with a different weighting function:

Clouds are opaque to some channels, and complicate this picture.

Some satellite sensors are designed to see only radiation from cloud tops. The samples of these sensors lie on a 2-D cloud top terrain embedded in the 3-D atmosphere.

Some satellite sensors are designed to see the ground terrain, or even the subtle terrain defined by the ocean surface.
TIME SAMPLING

Model values are time averages. Note however, that for visualization we sub-sample model time steps, so:

1. Visualized values may be time averages over the entire intervals between time steps (if we average when we sub-sample), or
2. Visualized values may be closer to point samples in time (if we don't average when we sub-sample).

All grid points and all fields of a model are sampled at the same time or averaged over the same time interval.

Most observing systems measure spatial points one-by-one in sequence, so different points have different "valid times."

Observations are usually time averages over very short intervals (compared to the interval between entire volume scans).

However, some satellites take a snapshot in time of the radiances of all pixels and store them on a videcon - in this case all pixels have the same "valid time."

VALUE SAMPLING

Although models calculate with floating point numbers to minimize the accumulation of round-off errors, the actual precision of predicted values is much less than their 24 or 48 bit mantissas.

Some recent work with ensembles of weather forecasts (with slight random perturbations of their calculations) to estimate the accuracy of predictions.

Observations are often coded by 8 to 12 bit Analog-to-Digital converters.

Radiances vary between "identical" sensors. Need for calibration and verification against direct measurements with trusted instruments.
DATA PRODUCED BY COMPUTATIONS

Mathematical manipulations of observations produce new data in coordinate systems other than space and time. Histograms and scatter diagrams are examples:

Mathematical objects describing characteristics of observed data can themselves be handled as data, such as probability distributions of errors:

\[
\text{OBSERVED RADIANCE} \quad \downarrow \\
\text{DISTRIBUTION OF ACTUAL RADIANCE}
\]
DATA MODELS

The role of data in science:

Scientists develop mathematical models:
  to simulate nature
  to interpret and analyze observed data.

There is a two-way relation between mathematics and data:
  data (and algorithms) implement mathematical models of nature
  mathematical DATA MODELS interpret and characterize data.

A data model is part of an overall view of visualization [7] that includes:
  data model - how data are defined and organized
  computational model - how computations are expressed and executed
  display model - how data and information are communicated to a user
  user model - the tasks and capabilities (e.g., perceptual) of users
  hardware model - characteristics of hardware used to store, compute
  with and display data
THE FIELD DATA MODEL

The field data model [1] is a mathematical abstraction of some of the characteristics of the scientific data organizations that we have described. This data model was designed into the Data Explorer system, and many of its ideas are used by other visualization systems.

The field data model is about differentiable manifolds, which let scientists use calculus to define mathematical models of nature in spaces whose topologies differ from the usual flat Cartesian space.

For example, scientists study air flow around an airplane using differential equations. The space where the air flows is normal 3-D Cartesian space, minus the space inside the airplane. This is space-with-a-hole-in-it and its topology is not the same as the full Cartesian space. The field data model helps us describe data structures for storing the numerical solutions of differential equations in this space-with-a-hole-in-it.

The basic idea is to break the space-with-a-hole-in-it into pieces, each of which has a nice Cartesian topology.

Here's a Cartesian 2-D square with a hole in it. We have divided the square into four curvy rectangles. Each curvy rectangle has the simple topology of the original square. Together they make the complex topology of the square-with-a-hole-in-it.
After we divide space into a set of curvy pieces, we need a way to define coordinate systems on those pieces, so we can do calculus on them.

This diagram shows a curvy 2-D "base domain" embedded in a 3-D space. An invertible differentiable function $\Phi$ maps a rectangle in a 2-D "manifold coordinate system" to the "base domain." A differentiable function $\Psi$ maps the 2-D coordinate system to a 2-D "dependent variable space." Then $\Psi \circ \Phi^{-1}$ is a function from the curvy domain to the dependent variable. This function may be the solution of differential equations modeling a physical system.

The objects of calculus are functions. Inside computers we usually implement functions as data objects that sample the functions at finite numbers of points. The field data model describes how we organize sample points in "base domains."

A set of lines may connect pairs of points (these may be the pairs of points used to calculate finite difference approximations to differential equations). The relation between points defined by these lines defines a sort of topology for the point set. The topology may be regular or irregular.
DATA MODEL ISSUES

We have seen several examples of scientific data, and we have seen an example of a scientific data model. A data model defines and organizes a set of data objects. Here we describe some general issues that data models need to address:

1. The types of **primitive values** occurring in data objects. Primitive values are atoms of data, such as integers, real numbers and text characters. Types of primitive values may be classified according to the kinds of primitive operations that may be applied to values. There may be an order relation defined on values (ordinal), there may be a topology defined on values (discrete or continuous), there may be a distance relation between values (metric), and there may be arithmetical operations defined on values.

2. The ways that primitive values are **aggregated** into data objects. There are a variety of approaches to defining data aggregates. Common programming languages allow data to be aggregated as tuples (e.g., structures in the C programming language) and arrays, and provide pointers for defining more complex aggregates. The field model aggregates values in terms of differentiable manifolds. Object-oriented programming languages broaden the notion of aggregates to include more general relations between data objects as defined by functions.

3. **Metadata** about the relation between data and the things that they represent. For example, given a meteorological temperature value, metadata includes its spatial and temporal location (and the extents and weighting of any averaging) in the Earth's atmosphere. The fact that the value is a temperature, and its scale (Fahrenheit, Kelvin, etc), are also forms of metadata. Metadata includes an estimate of the accuracy of the temperature (whether sensed by an instrument or computed by a weather model). Because instruments and observing systems are fallible, an expected data value may not be defined at all, so missing data indicators are a form of metadata. If a temperature is observed by an instrument, then there may be metadata about the instrument (e.g., aperture, pointing direction, filters, etc.). If a temperature is computed, then there may be metadata about the algorithm used to compute the temperature and the source of the algorithm's inputs.
THE LATTICE DATA MODEL

Mathematical models define infinite precision real numbers and functions with infinite domains, whereas computer data objects contain finite amounts of information and must therefore be approximations to the mathematical objects that they represent.

Several forms of metadata are intended to document the precision with which data approximate the things that they represent:

1. Accuracy estimates attached to values. This may be expressed as a number of digits of precision, as lower and upper bounds for a value, or as distributions of the expected errors of values.
2. Missing data codes (i.e., totally imprecise representations).
3. The locations and times of samples (i.e., what is the set of samples used to represent a continuous field), and accuracy estimates for those sample locations.

We propose a data model based on lattices that integrates these forms of metadata.

Data objects in the lattice data model represent mathematical objects.

Data objects are classified according to their data types.

Data objects of **scalar** data types are primitive values of the data model, and represent the primitive variables of mathematical models.

Primitive values are aggregated into complex data objects by tuple and array type constructors.

**Array** data types represent functions. The index of an array represents the domain of the function (so the index of an array is a scalar type). The values in an array represent the range of the function.

**Tuple** data types represent tuples of mathematical objects.

The lattice data model is concerned with how precisely data objects approximate mathematical objects.
Scalar data types represent primitive variables of mathematical models.

A continuous scalar type defines a set of closed real intervals (i.e., \([a, b]\) where \(a\) and \(b\) are real numbers) as values. Data objects of continuous scalar types represent real variables. An interval \([a, b]\) is an approximation of a real number \(c\) if \(c\) is in the interval \([a, b]\) (i.e., if \(a \leq c \leq b\)).

We define an order relation between real intervals by the inverse of set containment. That is, smaller intervals are "larger" since they are more precise:

\[
\begin{align*}
[0.0, 0.0] & \quad [0.01, 0.01] & \quad [0.5, 0.5] & \quad [0.945, 0.945] \\
\quad [0.0, 0.01] & \quad & \quad & \quad \\
\quad & \quad [0.0, 0.1] & \quad & \quad \\
\quad & \quad & \quad [0.0, 1.0] & \quad \\
\quad & \quad & \quad & \quad \perp
\end{align*}
\]

A discrete scalar object defines a countable set of values. Data objects of discrete scalar types represent mathematical variables over the same value set. For example, a discrete scalar type may define the set of integers, or the set of finite text strings, as values. Such objects are only approximations to themselves.

There is no order relation between objects of discrete scalar types. For example, no integer is more precise than any other integer:

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \ldots \\
\quad & \quad & \quad & \quad & \quad & \quad & \quad & \perp
\end{align*}
\]

The value sets of continuous and discrete scalar types include the missing value denoted by \(\perp\). This value is less precise than any other value.
A data object of an array data type is a finite set of samples of a function.

For example, a function of a real variable is represented by a set of 2-tuples that are (domain, range) pairs. The set \{([1.1, 1.6], [3.1, 3.4]), ([3.6, 4.1], [5.0, 5.2]), ([6.1, 6.4], [6.2, 6.5])\} contains three samples of a function. The domain value of a sample lies in the first interval of a pair and the range values lies in the second interval of a pair.

Adding more samples, or replacing the second sample by the pair ([3.81, 3.85], [5.1, 5.13]), will create a more precise approximation to the function. This diagram shows the order relation between a few array data objects.
Tuple data types define sets of objects of simpler types. They are the same as structures in the C programming language.

Tuples are ordered element-wise, so that \((u, v, w) \leq (x, y, z)\) if \(u \leq x\), \(v \leq y\), and \(w \leq z\). This diagram shows the order relation among a few tuple data objects.

\[
\begin{array}{c}
(A, B, E) \\
| & | & | \\
(A, \bot, E) & (A, B, \bot) & (\bot, B, E) \\
| & | & | \\
(A, \bot, \bot) & (\bot, \bot, E) & (\bot, B, \bot) \\
| & | & | \\
(\bot, \bot, \bot)
\end{array}
\]

The lattice data model defines a set \(T\) of data types by:

1. Define a finite set \(S\) of scalar types (continuous and discrete). \(S \subseteq T\).

2. If \(s\) is a scalar type and \(r\) is any type that is not defined in terms of \(s\), then \(t = \text{array}[s]\) of \(r\) \(\in T\) is an array type with domain type \(s\) and range type \(r\).

3. If \(t_1, t_2, \ldots, t_n\) are data types defined from disjoint sets of scalar types, then \(t = \text{struct}\{t_1; t_2; \ldots; t_n\} \in T\) is a tuple data type with element types \(t_1, t_2, \ldots, t_n\).

Embed objects of all types in \(T\) in one big lattice \(U\).

For each scalar type \(s \in S\), define \(I_s\) as the set of data objects of type \(s\).

Define \(X = \prod_{s \in S} I_s\) (this is a cross product).

Members of \(X\) are tuples with one element for each scalar type in \(S\).

Define the lattice \(U\) as the set of all sets of tuples in \(X\). These sets are ordered by containment. (Well, almost. See reference [3] for the details).
Objects of all types in T can be naturally embedded in the lattice U.

For example, define U in terms of three scalars \textit{time}, \textit{temperature} and \textit{pressure}. Objects in the lattice U are sets of tuple of the form \((\text{time}, \text{temperature}, \text{pressure})\). Define a tuple data type \text{struct}\{\text{temperature}; \text{pressure}\}. A data object of this type can be modeled as a set of tuples (actually, it is a set consisting of one tuple) in U with the form \{\((\bot, \text{temperature}, \text{pressure})\)}). This embeds the tuple data type in the lattice U, as shown in this diagram.

\[
\text{embedding of a tuple type into a lattice}
\begin{array}{c}
\text{(temp1, pres1)}
\end{array}
\rightarrow
\begin{array}{c}
\{\bot, \text{temp1}, \text{pres1}\}
\end{array}
\]

\[
\text{an element of the tuple type}
\text{(temperature, pressure)}
\rightarrow
\text{a member of the lattice of sets of tuples of the form (time, temperature, pressure)}
\]

Define an array data type \textit{(array [time] of temperature)}. A data object of this type consists of a set of pairs of \((\text{time}, \text{temperature})\). This array data object can be embedded in U as a set of tuples of the form \((\text{time}, \text{temperature}, \bot)\), as shown in this diagram.

\[
\begin{array}{c}
time1: \text{temp1}
time2: \text{temp2}
time3: \text{temp3}
\vdots
\end{array}
\rightarrow
\begin{array}{c}
\{(\text{time1}, \text{temp1}, \bot),
(\text{time2}, \text{temp2}, \bot),
(\text{time3}, \text{temp3}, \bot),
\ldots
(\text{timeN}, \text{tempN}, \bot)\}
\end{array}
\]

\[
\text{array of temperature values indexed by time values}
\rightarrow
\text{set of tuples with \bot pressure values and with no two time values equal}
\]

These basic embedding ideas can be combined to embed complex types in U.

The lattice data model is implemented by the VisAD system [2]. See Appendix A for information about how to get VisAD by anonymous ftp.
VOXEL-BASED DISPLAY MODEL:

COLUMNS INDEXED BY COMBINATIONS OF selectors

select a set of data objects to display

define mappings:

for example:

region → selector-1
lat_lon → x, z
time → animation
ir → y
vis → color
count → x

adjust color tables

choose ranges of values for selectors

control animation

rotate and zoom 3-D voxel volume

adjust contour levels and intervals

USER INTERFACE FOR CONTROLLING DATA DISPLAY:

adjust color tables

choose ranges of values for selectors

control animation

rotate and zoom 3-D voxel volume

adjust contour levels and intervals

VOXEL-BASED DISPLAY MODEL:

Objects have complex types:

goes_partition:

array indexed by [region]

array indexed by [lat_lon]

structure {ir; vis; variance; texture;}

ir vis variance texture

histogram_partition:

array indexed by [region]

structure {array ... ; lat_lon;}

lat_lon

array indexed by [ir]

count

Examples of data types that users can define in the VisAD data model, a diagram of VisAD's voxel-based display model, and examples of the scalar mappings that VisAD users define to control how data are depicted in the display model.
THE SEMANTICS OF DATA FOR COMPUTATION AND DISPLAY

The lattice data model integrates metadata about data precision and this information can be integrated into the computational and display semantics of data.

APPROXIMATING REAL ARITHMETIC IN THE LATTICE

\[[a, b] + [c, d] = [a+c, b+d]\]
\[[a, b] - [c, d] = [a-d, b-c]\]
\[[a, b] * [c, d] = [\min\{a*c, a*d, b*c, b*d\}, \max\{a*c, a*d, b*c, b*d\}]\]
\[[a, b] / [c, d] =
    \begin{align*}
        & \text{if } c \leq 0 \leq d \text{ then missing} \\
        & \text{else } [\min\{a/c, a/d, b/c, b/d\}, \\
                      & \max\{a/c, a/d, b/c, b/d\}] 
    \end{align*}\]

\[[a, b] \text{ op missing} = \text{missing}\]
\[\text{missing op } [a, b] = \text{missing}\]
where \(\text{op} = +, -, *, /\)

APPROXIMATING FUNCTION OPERATIONS IN THE LATTICE

Functions approximated by finite samplings in arrays for example, for functions \(f, g: \mathbb{R}^2 \rightarrow \mathbb{R}\)

samples of \(f\) marked by: \(\circ\)
samples of \(g\) marked by: \(\times\)

approximate \(g + f\) by first resampling \(f\) at samples of \(g\)

where a sample of \(g\) (an \(\times\) ) falls outside range of \(f\)
(area covered by \(\circ\) 's)
value of \(f\) is \(\text{missing}\)
Analyzing Visualization in the Lattice

Lattice-structured Data Model \( U \)
Lattice-structured Display Model \( V \)
Visualization Mapping \( D : U \rightarrow V \)

Study visualization as a function, according to mathematical structures of \( U \) and \( V \). Rather than as a program.

Assume \( D : U \rightarrow V \) satisfies expressiveness conditions:

Displays express all facts about data objects and express only those facts.

Rigorously interpret the expressiveness conditions in the lattice. Then:

\( D : U \rightarrow V \) satisfies the expressiveness conditions if and only if \( D \) is a lattice isomorphism

i.e., \( D \) maps the lattice structure of \( U \) onto the lattice structure of \( V \)

Given data and display lattices where reals are approximated by intervals and missing, and functions are approximated by arrays, \( D \) is a lattice isomorphism if and only if:

Mappings from data aggregates to display aggregates can be factored into mappings from data primitives to display primitives.
OBJECT-ORIENTED IS NATURAL FOR LATTICE DATA MODEL

Mathematical objects are superclasses
   for example: real number, vector, function

Data object approximations are subclasses of these
   for example:

mathematical superclass:

   real number (virtual operations:
              add, subtract, multiply, divide)

   data subclasses:
   floating point number
   rational interval
      (virtual ops \rightarrow
      IEEE 754-1985)
      (virtual ops \rightarrow
      interval arithmetic)

RESEARCH AREA

Analyze visualization mappings $D : U \rightarrow V$ that preserve various
mathematical structures on $U$ and $V$.

<table>
<thead>
<tr>
<th>structure on U and V</th>
<th>mapping D : U \rightarrow V</th>
</tr>
</thead>
<tbody>
<tr>
<td>lattice</td>
<td>isomorphism</td>
</tr>
<tr>
<td>metric</td>
<td>isometric</td>
</tr>
<tr>
<td>topology</td>
<td>continuous, homeomorphism</td>
</tr>
<tr>
<td>arithmetic</td>
<td>linear</td>
</tr>
<tr>
<td>symmetry*</td>
<td>groups of mappings**</td>
</tr>
</tbody>
</table>

* an example of symmetry is that perception is invariant to translating a display in time or space
** the set of valid visualization mappings is invariant to composition with a group of symmetry transformations of $V$

See [5] for more information about this area.
Appendix A. How to get Vis5D and VisAD by anonymous ftp

Vis5D is our system for interactive visualization of simulations of the Earth's atmosphere and oceans. Vis5D is free and available by anonymous ftp over the Internet. To get it, do the following:

\[
% \texttt{ftp iris.ssec.wisc.edu} \\
\text{(or }% \texttt{ftp 144.92.108.63}) \\
\text{login: } \texttt{anonymous} \\
\text{password: } \texttt{myname@mylocation} \\
\texttt{ftp> cd pub/vis5d} \\
\texttt{ftp> ascii} \\
\texttt{ftp> get README} \\
\texttt{ftp> bye}
\]

See Section 2 of the README file for complete installation instructions.

VisAD is our system for interactively visualizing and steering scientific computations. VisAD is free and available by anonymous ftp over the Internet. To get it, do the following:

\[
% \texttt{ftp iris.ssec.wisc.edu} \\
\text{(or }% \texttt{ftp 144.92.108.63}) \\
\text{login: } \texttt{anonymous} \\
\text{password: } \texttt{myname@mylocation} \\
\texttt{ftp> cd pub/visad} \\
\texttt{ftp> ascii} \\
\texttt{ftp> get README} \\
\texttt{ftp> bye}
\]

See Section 2 of the README file for complete installation instructions.

For more information via the World Wide Web, see our home page at:

\[\texttt{http://www.ssec.wisc.edu/~billh/vis.html}\]
References


Techniques for Examining Multiple Data Sets and Solutions for Data Management

A Tutorial for

ACM SIGGRAPH '96 Course #16: Visualizing Scientific Data and Information: Focusing on the Physical and Natural Sciences

Lloyd A. Treinish
IBM Thomas J. Watson Research Center
Post Office Box 704
Yorktown Heights, NY 10598 USA
914-784-5038 (voice)
914-784-5130 (facsimile)
lloydt@watson.ibm.com

August 1996
# Techniques for Examining Multiple Data Sets and Solutions for Data Management

## Table of Contents

1. Introduction to Visualization and Analysis of Scientific Data Sets........................II-1

2. Unifying Principles of Data Management for Scientific Visualization..................II-5

3. Interactive Archives for Scientific Data.....................................................................II-35

4. A Rule-based Tool for Assisting Colormap Selection..............................................II-51
1. Techniques for Examining Multiple Data Sets and Solutions for Data Management

The visualization and analysis of large scientific data represents a very challenging task, especially in the earth, space and environmental sciences. The myriad of earth science data, for example, from observation (in situ and remotely sensed) and computation (simulations and empirical models) are complex and very large in volume. These data are multidimensional (typically two or three spatial dimensions, perhaps one or more non-spatial dimensions, e.g., energy), dynamic (time-varying in data and dimensionality) and consist of many parameters. There is enormous variation in the instrumentation used to observe the Earth that have consequences in the data geometries, sampling and error characteristics. Such variation is often compounded by inconsistencies in the data gathering process, especially for instruments that perform long-term monitoring of the Earth. These measurements each relate some aspect of the physical phenomena under observation. Typically they must be combined in order to glean some knowledge of the data. Furthermore, they are often used in conjunction with simulations to verify theory or as initial or boundary conditions for empirical models. A long term goal of such on-going activities as well as planned data acquisition and computational efforts is to view the Earth as an integrated system. This would merge and define the interactions between the near-space environment, the atmosphere, the oceans, the land (both surface and subsurface), etc. An additional aspect of such work is the evaluation of the environmental effects of anthropomorphic activities. Greater cognition of data characteristics and handling of the diversity of earth, space and environmental data can lead to effective solutions to their visualization and analysis. In that regard, a few of a plethora of important topics are discussed herein, such as data management (mechanisms for handling assorted kinds of data); data archiving and browsing (as an aid in identifying data of interest to analyze); and human perception (consideration of how images are interpreted). Abstracts for these topics follow.

2. Unifying Principles of Data Management for Scientific Visualization

The instrumentation technology behind data generators in a myriad of disciplines, whether observational or computational, is rapidly improving, typically much faster than the techniques available to manage and use the resultant data. In fact, an onslaught of orders of magnitude more data from such sources is expected over the next several years, especially in the earth and space sciences. Hence, greater cognizance of the impact of these data volumes is required. If adequate mechanisms for use of the actual data that meet the requirements and expectations of the scientific users of such data are not available, then the efforts to generate and archive the data will be for nought. Within the context of scientific visualization, the role of data management can be expressed by the need for a class of data models that is matched to the structure of scientific data as well how such data may be used. There are many approaches to data management within visualization systems. An understanding of the these mechanisms yields important insight to the strengths and weaknesses of various visualization architectures, including scalability in data set(s) size, complexity and diversity as well as ease of initial use. From such a survey, recommendations for future research and enhancements of current systems can be derived.

3. Interactive Archives for Scientific Data

Appropriate data management techniques coupled with the process or methods of scientific visualization, or at least the technologies that support them show promise in helping to address some of the problems of utilizing large data archives. Consider browsing, for example, in a role for feature identification by the scientist/user to serve as guide in the data selection process. To date, most efforts associated with data browsing have focused on simple images with image data. Unfortunately, these techniques are not applicable to many classes of data or when more than one data set is to be considered. By recalling that browsing is more of a subjective process involving the human visual system and that this is one of the origins of the notion of scientific visualization as a method of computing, then the utilization of visualization strategies for qualitative presentation of data becomes a viable approach. For browsing to be effective it must be interactive with near-real-time system response. With data sets of interesting size, e.g., ≥ O(1 GB), immediate interaction cannot take place on current conventional systems (i.e., high-end graphics workstations). Even though a 1 GB data set is admittedly modest
by today's standards, the access and visualization of the entire data set or even a large fraction of it place significant burdens on the floating point and bandwidth capacities of the computer system being employed. This idea can also be extended to environments without high-bandwidth access to an interactive system by considering the distribution of compressed visualizations instead of data for predefined access and browsing scenarios.

4. A Rule-based Tool for Assisting Colormap Selection

Visualization software maps data to a visual form, creating a pictorial representation of numerical data. Current software systems support a plethora of mapping choices and tools for manipulating and rendering data, but provide little guidance in their application to specific tasks. The user must define the mapping(s) between data and pictures via the tools in such systems, which are driven by the structure of the data, not the task of the problem. Hence, the visualization process is time-consuming and often in the domain of computer graphics laboratories not the desks of scientists. A framework for rule-based guidance may improve the effectiveness of systems by assisting the user in making two types of appropriate representation choices. One concerns domain-independent factors, such as ensuring that data content is reflected in images and that perceptual artifacts are not erroneously interpreted as data features. Thus, the notion of mapping data structures onto perceptual structures is introduced. A second type concerns task-dependent factors. For example, different advice on representation is required depending on whether the goal of visualization is exploration or presentation. In either case, rule-based rather than tool-based systems can help users make good decisions about the visualization of their data without requiring them to become experts in human vision, data structures, visualization algorithms and color theory.

Example -- Global Topography

Figure 1-1 contains western and eastern hemisphere views of the Earth's terrain, including the topography of its surface as well as bathymetry (the topography of the ocean floor) derived from a simple rectilinear grid. The surface is mapped to a gray scale so that the region above a nominal sea level has the appearance of a topographic map while that below is gray but increasing in darkness with depth. In addition to the familiar landmasses, this depiction illustrates undersea mountain ranges and crustal plates. The original topographic grid has been warped onto a radially deformed sphere at 5-minute resolution. This transformation was applied to the base geometry only. Hence, the data were not altered in any fashion by this operation. This approach to preserving data fidelity during visualization not only illustrates the characteristics of the data but also regions of poor coverage, artifacts of the gridding process and supersampling, errors in height values and registration, etc. The entire globe contains 9,331,200 polygons (quads) with about half being visible in each of the views in the image. The deformed spheres were Gouraud-shaded and rendered with annotation as a 4000 x 3200 pixel-resolution gray-scale image in PostScript. The data are available courtesy of the National Space Science Data Center at NASA/Goddard Space Flight Center, Greenbelt, MD.

Copyright © IBM Corporation 1993. All Rights Reserved.

Figure Caption

1-1. Two different geographic views of pseudo-colored and radially deformed topographic globes of the earth.
Spherical Projections of Earth's Terrain (km above and below sea level)
2. Unifying Principles of Data Management for Scientific Visualization

Background

Obviously, data management is critical for appropriate and effective utilization of large volumes of data. For example, such data may come from sources as diverse as

- remotely-sensed or in-situ observations in the earth and space sciences
- seismic sounding of the earth for petroleum geophysics (or similar signal processing endeavors in acoustics/oceanography, radio astronomy, nuclear magnetic resonance, synthetic aperture radar, etc.)
- large-scale supercomputer-based models in computational fluid dynamics (e.g., aerospace, meteorology, geophysics, astrophysics), quantum physics and chemistry, etc.
- medical (tomographic) imaging (e.g., CAT, PET, MRI)
- computational chemistry
- genetic sequence mapping
- intelligence gathering
- geographic mapping and cartography
- census, financial and other "statistical" data

The instrumentation technology behind these data generators is rapidly improving, typically much faster than the techniques available to manage and use the resultant data. In fact, an onslaught of orders of magnitude more data from these and other sources is expected over the next several years. Hence, greater cognizance of the impact of these data volumes is required. For example, NASA’s Earth Observing System (Eos), which is planned for deployment in the late 1990s, will have to receive, process and store up to ten TB (10^{13} bytes) of complex, interdisciplinary, multidimensional earth sciences data per day, for over a decade. Given the fact that such data sets are or will be generated, what can be done to cope with this deluge from the perspective of their access, management and utilization?

The ability to generate pictorial representations of data is currently in vogue as being the answer. This concept of (scientific) data visualization really implies a method of computing that gives visual form to complex data using graphics and imaging technology. It is based upon the notion that the human visual system has an enormous capacity for receiving and interpreting data efficiently. Despite the advancement of visualization techniques for scientific data over the last several years, there are still significant problems in bringing today’s hardware and software technology into the hands of the typical scientist. Some of these same problems do occur in more general processing and analysis of scientific data in many disciplines. For example, data management is required to make such computing effective, which can be expressed by the need for a class of data models that is matched to the structure of scientific data as well how such data may be used. The critical component of data management is typically missing in many such computing efforts. When this concept is scaled to support large data sets (e.g., a few GB), several critical problems emerge in the access and use of such data, which brings the problem back to the data level. This circular reasoning does not imply that applications such as visualization are not relevant, but the challenge is more complex than simply flowing data from storage and looking at the pictures or even "pointing" back to the numbers behind the pictures. This requirement for scientific data models extends beyond the definition and support for well-defined physical formats. It must include the logical specification of self-documenting (including semantics) scientific data to be studied via a visualization system and data derived through the operation of the tools in such a system.
Recently, considerable attention has also been placed on metadata support for the management of current and past large data streams because through it a scientist would be able to select data of interest for analysis. However, if adequate mechanisms for use of the actual data that meet the requirements and expectations of the scientific users of such data are not available, then the efforts to generate and archive the data and supporting metadata management will be for nought. Although it is beyond the scope of this discussion, advances in visualization and data structures can also be applied to metadata management in the form of browsing, support of spatial search and selection criteria, etc.

What can be done? A place to begin is a discussion about solutions for how data should be organized, managed and accessed, which is often not adequately addressed in visualization and related software. The following is a survey of some of the methods and requirements for data management in visualization. It is hardly meant to be either exhaustive or definitive, but merely as an introduction to an important topic.

Data Characteristics

There are a tremendous number of sources of scientific data, be they computed or measured. Even from a single source there can be a wide variety of data sets. Each such data set typically contains several independent variables such as time, one or more spatial, spectral, etc. variables, and of course, many dependent variables, where the interesting science is stored. There can be a bewildering range of underlying formats, structures, arrangements and access methods for these data. To attempt to bring some simplifying order to this chaos, consider six key attributes of data: dimensionality, parameters, data type, rank, mesh structure, and aggregation.

Any data set may be considered as a function(s) of independent variable(s). These independent variables may be called dimensions. The number of independent variables may be called the dimensionality of the data. It is the fundamental characteristic of the data. Such dimensions may be space (length, width, height), time, energy, etc. For example, zero-dimensional data are just numbers such as sales, while two-dimensional data could depend on an area such as barometric pressure over a state. Some complex data may have five or more dimensions.

The function(s) composing a data set really are dependent variable(s) -- the data themselves, which may be called parameters. They are dependent on the dimensions, such as sales or temperature. Thus, data implies a parameter or field of one or more (dependent) values that is a function of one or more (independent) variables,

\[ \{y_1, y_2, \ldots, y_n\} = \{f_1(x_1, x_2, \ldots, x_n) \}
\]

\[ f_2(x_1, x_2, \ldots, x_n) \]
\[ \vdots \]
\[ f_n(x_1, x_2, \ldots, x_n) \]  \hspace{1cm} (1)

The data type includes the physical primitive, which describes how data values are stored on some medium (e.g., byte, int, float, etc.). It can include machine representations (e.g., little endian vs. big endian, IEEE vs. VAX, etc.). In addition, there can be a category of such types, i.e., real, complex or quaternion.

A parameter may have more than one value, which is characterized by tensor rank. Rank 0 is a scalar (one value), such as temperature (a magnitude -- a single-valued function). Rank 1 is a vector such as wind velocity (a magnitude and a direction; two values in two dimensions, three values in three dimensions). Vectors of size, \( n \), are \( n \)-valued functions. Rank 2 is a tensor such as stress on an airframe (four values in two dimensions, nine values in three dimensions). A rank 2 tensor in \( n \)-dimensional space is a \( n \times n \) matrix of functions (e.g., stress). Dimensionality and rank are thus, related. The number of elements in a particular parameter is \( d^n \), where \( d \) is the dimensionality and \( n \) is the rank. A distinction can be made between a 2-vector, 3-vector and two scalars in 2-space vs. a 2-vector, 3-vector and three scalars in 3-space independently of data type primitive or underlying mesh structure, if any. As with dimensionality, rank may be large for very complex data.
Such a taxonomy typically does not include those data directly associated with graphics, which is beyond the scope of this discussion. Table 2-4, at the end of this paper, outlines the extent of support of these characteristics by various implementations. An important point to consider is that these characteristics should be considered more or less independently of each other to maximize flexibility in actual visualization software.

As with data, the techniques used to visualize may be similarly classified by their dimensionality. Because visualization implies creating a pictorial form for data, there is an implied geometrical relationship. Table 2-1 summarizes the dimensionality of typical visualization geometries.

**Table 2-1. Dimensionality of Visualization Geometry**

<table>
<thead>
<tr>
<th>Dimensionality</th>
<th>Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Point</td>
</tr>
<tr>
<td>1</td>
<td>Line</td>
</tr>
<tr>
<td>2</td>
<td>Polygon</td>
</tr>
<tr>
<td>3</td>
<td>Volume</td>
</tr>
</tbody>
</table>

Of course, a geometric primitive of lower dimensionality may be imbedded in a higher dimensional space such as a line on a plane or in a volume or a surface in a volume.

However, the dimensionality of visual representation may be different than that of the data, depending on the specific visualization task. In addition, these geometries may be used in different ways to create a specific visual representation.

Table 2-2 illustrates some of the diversity of visualization techniques that may be required in a system when only considering the earth and space sciences. Examples are listed with different dimensionalities and rank. Again, the number of elements in a particular quantity is \( d_r \), where \( d \) is the dimensionality and \( r \) is the rank. The suite of techniques should accommodate time-dependent quantities as having another scalar dimension, which can be mapped into a specific visualization primitive or "axis." In a complementary fashion, animation should be treated as an additional scalar "axis" for sequencing of other visualization strategies in either a discrete or continuous fashion, which may be hardware dependent. It should be noted that the example data types are not intended to be all encompassing and the specific visualization methods are not meant to be definitive mappings for specific data types nor exhaustive – but only to impart the notion of the amount of diversity.
The following are a few cases of commonly used mesh structures:

1. Regular grid with regular positions and regular connectivity.

   In this case, the two-dimensional primitive is a rectangle while the three-dimensional primitive is a parallelepiped. Usually, this case would imply cartesian coordinates. The mesh may be implicitly stored in a compact fashion via a specification of an ordered pair for each dimension (i.e., the origin and spacing in some coordinate system). The entire mesh may be formed by taking a product of the specifications across each dimension. An example would be a grid of temperatures over a rectilinear map of the earth's surface.

2. Deformed regular or curvilinear or structured grid with irregular positions and regular connectivity.

   In this case the primitives are the same as in case 1. Non-cartesian coordinates may be implicitly supported through such a deformed structure. Generally, the mesh is explicitly stored via a specification of a node position along each dimension (axis) in some coordinate system. The entire mesh may be formed by taking a product of the specifications across each dimension. An example would be a grid of pressures over an airframe.

   A variation in this case and in case 1 would be for a partially regular grid, which has one or more dimensions being regular (i.e., case 1.) and the balance being irregular.

3. Irregular "regular" or structured grid with irregular positions and regular connectivity.

   In this case the primitives are the same as in case 1 and 2, but with irregular sizing and spacing, which may include holes. Positions are specified explicitly. Sparse matrices, grids or meshes can also be considered in this category. However, indexing schemes can be introduced to eliminate wasted storage. An example would be several satellite images with gaps in coverage.

4. Unstructured or irregular grid with regular or irregular connectivity.

   In this case, the two-dimensional primitive is typically a triangle while the three-dimensional primitive is typically a tetrahedron. However, it may be prismatic, icosahedral, hexahedral, etc. or even variable. Positions of each node are specified explicitly while a connections list identifying the relationship and order of each node on each primitive is generally required. An example would be a finite element mesh of the structural integrity of the frame of a car.

5. No grid with irregular positions and no connectivity.

   In this case the data are scattered or available on specific points, which are explicitly identified. An example would be measured rainfall in specific towns.

The notion of taking a product to form a mesh from vectors of positions can be applied to connections between positions. For example, regular connections of n points is defined by a set of n-1 line segments. A product of two sets of connections is a set of points obtained by summing one point from each of the terms in all possible combinations. Hence, a product of two regular connections composed of line segments would be a set of squares. A product of a regular connections of line segments and an irregular one of triangles would be a partially regular set of prisms. Figure 2-1 shows a potential taxonomy for these different mesh classes with respect to the functional description in equation (1).
The following are a few cases of commonly used mesh structures:

1. **Regular grid with regular positions and regular connectivity.**

   In this case, the two-dimensional primitive is a rectangle while the three-dimensional primitive is a parallelepiped. Usually, this case would imply cartesian coordinates. The mesh may be implicitly stored in a compact fashion via a specification of an ordered pair for each dimension (i.e., the origin and spacing in some coordinate system). The entire mesh may be formed by taking a product of the specifications across each dimension. An example would be a grid of temperatures over a rectilinear map of the earth's surface.

2. **Deformed regular or curvilinear or structured grid with irregular positions and regular connectivity.**

   In this case the primitives are the same as in case 1. Non-cartesian coordinates may be implicitly supported through such a deformed structure. Generally, the mesh is explicitly stored via a specification of a node position along each dimension (axis) in some coordinate system. The entire mesh may be formed by taking a product of the specifications across each dimension. An example would be a grid of pressures over an airframe.

   A variation in this case and in case 1 would be for a partially regular grid, which has one or more dimensions being regular (i.e., case 1.) and the balance being irregular.

3. **Irregular "regular" or structured grid with irregular positions and regular connectivity.**

   In this case the primitives are the same as in case 1 and 2. but with irregular sizing and spacing, which may include holes. Positions are specified explicitly. Sparse matrices, grids or meshes can also be considered in this category. However, indexing schemes can be introduced to eliminate wasted storage. An example would be several satellite images with gaps in coverage.

4. **Unstructured or irregular grid with regular or irregular connectivity.**

   In this case, the two-dimensional primitive is typically a triangle while the three-dimensional primitive is typically a tetrahedron. However, it may be prismatic, icosahedral, hexahedral, etc. or even variable. Positions of each node are specified explicitly while a connections list identifying the relationship and order of each node on each primitive is generally required. An example would be a finite element mesh of the structural integrity of the frame of a car.

5. **No grid with irregular positions and no connectivity.**

   In this case the data are scattered or available on specific points, which are explicitly identified. An example would be measured rainfall in specific towns.

The notion of taking a product to form a mesh from vectors of positions can be applied to connections between positions. For example, regular connections of n points is defined by a set of n-1 line segments. A product of two sets of connections is a set of points obtained by summing one point from each of the terms in all possible combinations. Hence, a product of two regular connections composed of line segments would be a set of squares. A product of a regular connections of line segments and an irregular one of triangles would be a partially regular set of prisms. Figure 2-1 shows a potential taxonomy for these different mesh classes with respect to the functional description in equation (1).
Functions of Several Variables
(Discrete Sampling of Continuous Data)

Explicitly Positioned
(e.g., irregular)

Implicitly Positioned
(e.g., regular)

Independent
Variable
Values

Dependent
Variable
Values

Neighbors

Dependent
Variable
Values

Figure 2-1. Example Taxonomy of Data Properties and Meshes

Some of the two-dimensional mesh structures are shown in Figure 2-2. Figures 2-3, 2-4, 2-5, 2-6 and 2-7 show similar mesh structures, but in three dimensions. However, conventions are required for the specification of where the data (function is applied) with respect to the connection elements or topological cell primitives. The data may apply to each node or position of a grid. Alternatively, the data may apply to a cell center or face, or even an edge. Although many cases usually apply only one convention for an entire mesh, it is not always true. Some finite element meshes may use more than one type of primitive and convention.

Figure 2-2. Example Two-Dimensional Mesh Types
Figure 2-3. Example Three-Dimensional Mesh Type: Partially Regular Mesh of Parallepipeds

Figure 2-4. Example Three-Dimensional Mesh Type: Irregular Mesh of Parallepipeds
Figure 2-5. Example Three-Dimensional Mesh Type: Deformed Curvilinear or Structured Mesh of Parallepipeds

Figure 2-6. Example Three-Dimensional Mesh Type: Unstructured Mesh of Tetrahedra
Data Aggregates

Often there is a need to form aggregates of data. In dependent-variable space, one could have a collection of parameters or fields over the same or different grids that could be treated as single entity. In that sense, an aggregate or group could be composed of members that are either a single field or another group. This mechanism can be used to define simple tree structures. In independent-variable space, one could have a collection of meshes. For example, in aerospace fluid dynamics simulations, a computation is often performed over several intersecting grids. Such a multigrid solution permits the definition of variable grid resolution and regularity around airframe structures such as an engine nacelle or a wing. In this case, one could treat the collection of meshes as a single entity, although a mechanism must be defined for accommodating regions of invalidity where grids may intersect. A similar problem occurs with observational data, where there may be no data for some grid nodes. Another type of mesh aggregate can result from a hybrid collection of meshes of different cell primitives, where within each mesh, the cell is the same.

A special case of aggregates can be called series, where the tree structure has only one level of children. The classic example is a time series, where there is multiple instances of some field or aggregate over a constant or changing mesh. Generically, such a series does not have to depend on time, but could apply to any sequencing of events.

Temporal Data

The support of temporal data as time series is an interesting matter. How should time be handled? There are two flavors of temporal support: as a running clock and epoch-based. In either case, time tags could be viewed like an additional dimension in the position component of a field with "linear" connectivity. One could then use a compact representation for constant $\Delta t$. Otherwise, each time tag (position) could be specified individually for irregular spaced time steps. Obviously, the other method of supporting time tags would be associating each tag with a member of a (time) series.
The running clock approach supports elapsed time from some base (i.e., such as the origin of a position component dimension) like time of launch or start of simulation. The simplest implementation would be the use of a floating point representation of linear time in user-defined units (e.g., milliseconds, seconds, minutes, hours, days, years). Optionally, the elapsed time could be in a clock or calendar representation as a float (e.g., HH:MM:SS.s, YY:DD:DD.d, where HH = hour [0-23], MM = minute [0-59], SS = second [0-59], s = fractional second, YY = year, DDD = day of year [0-365 or 366], and d = fractional day).

Epoch-based time tags represent an "absolute" standard time from a standard reference epoch in a linear, clock or calendar representation. Such tags are more common with observational than simulated data since the former often require a link to the real world as well as registration with other observations. Generally, GMT is used as the standard time frame (at least for the earth). Example reference epochs often start exactly at midnight on January 1 of some specific year (e.g., 0, 1900, 1950) according to an accepted (i.e., European) calendar. Some example conventions for representation include the following:

- milliseconds since midnight, 1 January 0 AD (double)
- year/month/day since midnight, 1 January 0 AD: YYYYMMDD (long)
- time of day: HH:MM:SS (long)
- year/month/day since midnight, 1 January 1900: YYMMDDD.d (float)
- year/day of year since midnight, 1 January 1900: YYDDD.d (float)
- year/month/day/hour/minute/second/msec since midnight, 1 January 0 AD: YYYYMMDDHH:MM:SS:SSS (long)
- year/month/day/hour/minute/second/msec since midnight, 1 January 0 AD: YYYYMMDDHH:MM:SS:SSS (string)

Obviously, one can generate any number of acceptable and useful conventions ad nauseam. However, in almost all such cases there are significant problems in providing a 32-bit floating point representation with sufficient precision to handle both clock and calendar data together. In addition, for whatever conventions are established, efficient algorithms for calendric and temporal manipulation will be required (e.g., unit and scale conversion, leap year calculations) to keep these conventions reasonably transparent.

**Implementations and Techniques**

Traditional methods of handling scientific data such as flat sequential files are generally inefficient in storage, access or ease of use for large complex data sets particularly for input/output and floating-point-intensive applications like signal processing (e.g., inverse problems) and visualization. Modern, commercial relational data management systems do not offer an effective solution because they are more oriented to business applications. The relational model does not accommodate multidimensional, irregular or hierarchical structures often found in scientific data sets. In addition, relational systems do not provide sufficient performance for the size, complexity and type of access dictated by current and future data sets and their potential usage. In contrast, these data base management systems have been quite viable for a class of non-spatial metadata management (e.g., in the NASA earth and space science community). Hence, current data base systems are not yet up to the challenge of supporting very large data sets.

Therefore, there is a need for some type of data (base) model(s) that possesses elements of a modern data base management system but is oriented toward scientific data sets and applications. This intermediate approach should be easy to use, support large disk-based (perhaps other media as well) data sets and accommodate multiple scientific data structures in a uniform fashion. In the process of providing simple access to self-describing data, such a mechanism should match applications requirements for visualization as well as data analysis and management, and be independent of any specific discipline or source or visualization technique. Hence, data management as embodied as a data model(s) is as important a component of a data visualization system as underlying graphics and imaging technology. Its implementation, the management of and access to the data, should be decoupled from the actual visualization software. Conceptually, Figure 2-8 illustrates this notion, where the data model serves as a bridge between visual and data media. In other words - borrowing from the
Many software packages touted as supporting visualization ignore the issue of data management. Access to data is often provided via a simple flat file structure that may or may not be proprietary or left as an exercise to the user. Systems that support different classes of grids (e.g., regular/structured and unstructured) typically do not provide uniform mechanisms for their use. In any of these cases, the extensibility of such software to many, potentially very large data sets for disparate applications is extremely limited. Any process that accesses data implies the development of software that can manage arbitrary data sets and possesses different tools for displaying (or working with) data. There must be a clean interface between the data and the display of the data, so that arbitrary data can be accessed by the visualization software. As a consequence of such an approach, a software system of this design has an open framework. It can ingest arbitrary data objects for visualization, and other visualization techniques can be added independent of the application. This implies that a significant reduction in long-term software development costs can be realized because new data sets do not require new display software, and new display techniques do not require new data access software. A recommended initial approach is to attempt to characterize and categorize the data in the domain of interest according to the taxonomy outlined in previous section.
Implementation Genre

These classic limitations have been recognized by a few groups in the support of a number of scientific applications. As a result, several data models have been defined, some within very domain-specific contexts while others being more general. Recently, a few of these models have been associated with software, including visualization and have associated language bindings that represent the implementation of one or more abstract data types. These implementations can be characterized in the following manner:

- Unified -- utilizes a single, generalized data model
- Diffuse -- utilizes multiple data models
- Focused -- utilizes a single, domain-specific data model

A unified implementation includes an Applications Programming Interface (API) and/or language bindings, which defines operations to be done with data. This implementation is shown schematically in Figure 2-10. Such an interface may be "object-oriented", where such operations imply methods that are associated with a class of objects. However, the notion of an abstract data type is sufficient for such an implementation. Hence, the API hides the details of the physical form(s) of storage for data. In general, such implementations permit easy incorporation of new data and applications and offer the potential to develop highly optimized performance for generic applications. The primary limitation of this approach relates to the data model itself. Such implementations are usually hard to extend for new classes of data not considered in the original data model definition.
A diffuse implementation includes multiple APIs, which define underlying data structures. Each of these structures and hence, APIs, are associated with a single physical form of storage. This implementation is shown schematically in Figure 2-11. Usually, separate implementations of specific applications are required for each interface, which may be integrated in some fashion by a higher-level application. Such implementations permit easy incorporation of new data because a new application, interface and structure is added. Hence, it becomes difficult to operate with more than one of the types of data simultaneously. As a result, these implementations are usually difficult to optimize or extend for generalized applications.

A focused implementation usually does not include an API for the data model. This implementation is shown schematically in Figure 2-12. Any interface to data in such a system defines the underlying physical data format. Therefore, it becomes easy to optimize the performance for very specialized applications and quite difficult to extend to other data and applications.
To illustrate this taxonomy, a few examples of each type of implementation for data management or applications enabling and visualization are discussed briefly below. These examples are either public domain or commercial software that are used in the scientific community. The list is meant to be illustrative, not exhaustive or comprehensive. Additional details concerning these implementations are given in the final section of this paper.

**Implementations for Data Management or Applications Enabling**

Common Data Format (CDF), developed at NASA/Goddard Space Flight Center, was one of the first implementations of a scientific data model. It is based upon the concept of providing abstract support for a class of scientific data that can be described by a multidimensional block structure. Although all data do not fit within this framework, a large variety of scientific data does. From the CDF effort spawned the Unidata Program Center’s netCDF, which is more focused on issues of uniform data transport. Hence, CDF and netCDF can individually be characterized as being distinct unified implementations.

Both netCDF and CDF support the same data model -- the idea of a data abstraction for supporting multidimensional blocks of data -- since netCDF is a separate and more recent implementation of the ideas that were developed in the original VAX/VMS FORTRAN version of CDF in the mid-1980s. Although the model is the same, the interfaces are quite different. The current release of CDF is much newer than that of netCDF, where the latter has only had incremental improvements and new ports since its initial implementation. These systems are extensible by the user, and conventions have been established in some organizations to ensure proper interpretation when data are exchanged.

NetCDF has only one physical form -- a single file written according to the IEEE standard via the eXternal Data Representation (XDR) protocol developed by Sun Microsystems and placed in the public domain. The multi-dimensional arrays are written by C convention (last dimension varies fastest). This implementation has proven to be very convenient and easy to use. As a result, the software has been ported to a number of platforms, is widely used in the atmospheric sciences and other research communities and can be employed with some independent visualization software. However, this approach does have its performance limits when scaled to large data sets, used by FORTRAN programmers or operated on non-IEEE machines. Many operating systems have relatively small limits (either actual or practical) on the size of physical files, which would require an user to implement a multiple file solution. In addition, the current software supports only limited direct editing or other transactions on the files in place (i.e., without copying). Currently, Unidata has only provided very limited generic utilities for data in netCDF. However, they are developing a number of new tools (netCDF operators) and a C++ interface.

In contrast, CDF supports multiple physical forms: XDR or native, single or multiple file (one header file for each variable), row (i.e., C) or column (i.e., FORTRAN) major organization and the ability to interoperate among them, which includes compatibility with the original VMS FORTRAN implementation. This implies that for performance-critical applications, the appropriate physical form can be chosen. For example, where data portability is not critical and absolute performance is of greater importance (e.g., VMS), the support of a native physical format is critical. The CDF software has been ported to a number of platforms and is most often used in the space physics and climate research communities. The CDF software supports caching and direct utilization of the file system to provide rapid access and in-place updates. Data in CDF are supported by a large and growing collection of both utilities and sophisticated general-purpose applications (some portable and some VMS-specific).

Another important effort has been the Hierarchical Data Format (HDF) developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. This activity evolved from the need to move files of scientific data among heterogeneous machines, which grew out of the requirement to look at images and other data on personal computers, workstations, etc. that were generated on a supercomputer. HDF, which is also self-describing, uses an extensible tagged file organization to provide access to basic data types like a raster image (and an associated palette), a multidimensional block, etc. In this sense, HDF provides access (i.e., via its C and FORTRAN bindings) to a number (six) of different data file organizations. Hence, HDF can be characterized as being a diffuse implementation.
The HDF software has been ported to a wide variety of platforms from personal computers to supercomputers. Currently, all of HDF's data structures are memory resident. Thus, the aforementioned discussion about the relative physical organizations of CDF and netCDF and performance tradeoffs do not apply to HDF, because that level of functionality is not provided. This limits the ability of an application using HDF software to effectively utilize or even randomly access large disk-resident data sets. The myriad of data types that HDF supports essentially through separate interfaces is through central registration at NCSA in contrast to the CDF/netCDF approach. There are obvious advantages and disadvantages of leaving full authority with the user or with the implementer. A particular choice of approach is highly application dependent. Thus, HDF is extensible to support additional models and formats but only at NCSA (e.g., recent efforts to support more complex data structures). HDF has been an extremely successful vehicle for creating portable data sets and driving a number of popular and powerful visualization packages in the Apple Macintosh environment and more recently under XWindow, which offers further evidence as to the importance of codified data access techniques.

HDF has been extended with a new storage scheme called Vset, that is compatible with the extant HDF structures. It attempts to supersede some of the limitations inherent in the CDF/netCDF and the conventional HDF data models. It supports regular and irregular data and the ability to form hierarchical groupings. Another extension to HDF currently in development is to embody the netCDF and model and interface as an additional HDF object. This idea may be extended to the CDF model and interface as well.

The Flexible Image Transport System (FITS) developed by the National Radio Astronomy Observatory is the standard interchange mechanism for astronomical (except for planetary) imagery and related data. It is well defined, self-describing but has no official software interface or language bindings. Hence, portability is implied at the physical byte level. Hence, FITS can be characterized as being a focused implementation.

Most databases for spherically distributed (e.g., astronomical, cartographic, climatological) data are not structured in a manner consistent with their geometry. As a result, such databases possess undesirable artifacts, including the introduction of "tears" in the data when they are mapped onto a flat file system. Furthermore, it is difficult to make queries about the topological relationship among the data components (e.g., adjacency of connected regions) using simple and purely symbolic means, that is, without performing real arithmetic. For example, hierarchical data structures are appropriate to support spatial data. A potential solution to these problems has been developed at NASA/Goddard Space Flight Center by applying recursive subdivision of spherical triangles obtained by projecting the faces of an icosahedron onto a sphere. The collection of these spherical triangles are managed with quadtrees. These sphere quadtrees are insensitive to the familiar distortions suffered in planar representations far away from the equator. Such distortions arise from the need to properly project data (i.e., flatten the earth) by preserving shape or distance, for example, in two dimensional visualizations. This data structure allows the representation of data at multiple resolutions and arbitrary levels and is supported with efficient searching mechanisms. It provides the ability to correlate geographic data by providing a consistent reference among data sets of different resolutions or data that are not geographically registered. The implementation of this idea represents a hybrid of the aforementioned approaches because it is very domain-specific, but possesses a well-defined data structure and interface as well as operations.

**Implementations for Visualization**

The Visualization Systems Group at IBM Thomas J. Watson Research Center has developed a data format, which is based upon a more comprehensive data model that includes curvilinear and irregular meshes and hierarchies (e.g., trees, series, composites), vector and tensor data, etc. in addition to the class of scalar, multidimensional blocks supported by the aforementioned implementations. Currently, the physical disk-based format, called dx, is complete and has been published, but it provides only simple sequential access. A portable, application-independent software interface for it is not yet available, and hence support for disk-based structures and database-type updates in place are not supported. The IBM Visualization Data Explorer software, developed by this same group, utilizes this format as one import/export mechanism. It is a client-server data-flow system for general visualization applications. More importantly, however, is the implementation of the data model in this software package via an object-oriented approach to management of data in memory for computation. All operations on data within this software, independent of a role in generating pictures, work with shared data structures in memory via an uniform interface. The model utilizes the fact that researchers at Sandia National Laboratories observed that the mathematical notion of fiber bundles provides
a useful abstraction for scientific data management and applications. In the Visualization Data Explorer, this idea is specialized and extended to incorporate localized, piecewise field descriptions. This permits the same consistent access to data independent of its underlying grid, type or hierarchical structure(s) via a uniform abstraction. Data communication among subsequent operations is accomplished by passing pointers. In addition, sharing of these structures among such operations is supported. For example, when an image is rendered, or for that matter any computation in this system, the data structures can be linked (i.e., hierarchically) so that a path from the data to the pictures and back can be identified. Hence, IBM Visualization Data Explorer can be characterized as being an unified implementation.

The Application Visualization System (AVS) from Advanced Visual Systems, Incorporated and IRIS Explorer or simply Explorer from Silicon Graphics Incorporated are both distributed data-flow software packages for general visualization applications. From a data management perspective, both systems are conceptually quite similar. They incorporate separate underlying interfaces for different classes of data (e.g., structured vs. unstructured). As a result they often provide separate applications of similar functionality for different data (i.e., modules). Both packages also provide higher-level applications that integrate some of this functionality. Hence, AVS and Explorer can individually be characterized as being distinct diffuse implementations.

The Flow Analysis Software Toolkit (FAST) developed by Sterling Federal Systems, Incorporated for NASA Ames Research Center is an integrated collection of tools designed for the visualization and analysis of the results of computational fluid dynamics (CFD) simulations. The performance of the tools and their interface are optimized toward researchers working on CFD problems. Hence, FAST can be characterized as being a focused implementation. More recently, researchers there have also utilized the idea of a fiber bundle to serve as an abstraction for CFD data to be visualized, and hence simplify the manipulation of large data sets. Their system, called SuperGlue, uses an interpretative language, based upon C and Scheme (a dialect of Lisp) for rapid prototyping of complex visualizations and user interfaces while effectively supporting code reuse. Therefore, SuperGlue can be considered a hybrid implementation because it uses a unified approach for handling data but within a very specific domain.

Metadata

Inherent in data models for visualization should be the ability to support self-describing scientific data structures. Generically, such self-descriptions are loosely referred to as metadata or information/data about data. Recently, considerable attention has been placed on metadata support for the management of current and past large data streams because through it a scientist would be able to select data of interest for analysis. For example, the National Space Science Data Center (NSSDC) at NASA's Goddard Space Flight Center among others has been instrumental in the implementation of several systems that provide metadata management services to the earth and space science research communities. Such support has primarily been for characteristic (i.e., directories and catalogs), physical inventory and temporal information with more recent efforts looking at problems in spatial (e.g., geographic) information. Unfortunately, most efforts at metadata management, including many of those within NASA have been divorced from the data themselves. An exception to this was the development of an information/analysis system for atmospheric research in the early 1980s. An attempt to bridge this gap between metadata management and data was one of the factors that lead to the initial development of the CDF, from which spawned (for some similar reasons) netCDF. It is in such a context that this discussion about metadata - that is for the data themselves, what information is required to make the data sufficiently scientifically self-descriptive for "visualization."

In addition to self-describing data structures with well-defined software interfaces such as CDF, netCDF, and HDF that support metadata at some level, there are a number of more traditional approaches bundling useful scientific information with data. Typically this has implied sequential (i.e., magnetic tape) formats whose data contents are preceded by formatted text headers with little or no associated software. Among the better examples are ones that have been developed in the earth sciences in Europe (e.g., GF3, BUFR/CIRI) and the astronomical imaging community in the US (e.g., FITS). Recently, on-line versions of such formats have appeared as well as reading/writing FORTRAN programs and interesting applications. Although these formats were originally designed for the convenience of data producers trying to create magnetic tapes rather than more challenging applications like visualization, their developers and subsequent users have developed a rich lexicon of metadata that should not be ignored.
Data base management systems (DBMS) provide a capability to find, sort, merge, organize, update, and output diverse data types. However, most DBMSs have been designed primarily for archiving and managing data for a specific domain by developers with a background in computer science or related fields rather than a (physical) science discipline. The result is that these systems suffer from the intrinsic flaw of not effectively providing the capabilities needed by a casual or new user in that scientific discipline. These databases restrict the capabilities for managing the syntax of a domain as part of its data structure, have limited data structures which cannot represent explicit relationships between data classes and demand precise, mathematical query formulation for database interactions (i.e., SQL). In addition, such DBMS often exclude many of the data objects used in the scientific domain and do not efficiently store, index, or retrieve (i.e., in relational systems) image or spatial data. Hence, such database bases are difficult to design for scientific data and the users of existing database systems require an in-depth understanding of the database architecture, data content, location, and query language in order to use the data effectively. When data structures exist in a large database (i.e., schemata), this problem is exacerbated, often making the database unusable in an operational environment. These limitations can apply for large scientific metadata bases as well as for the data themselves.

Classes of Metadata

To help in the evaluation of what metadata and how it should be supported, the notion of metadata must be more clearly specified. Hence, four classes of metadata are defined: (1) what is needed to access the data; (2) what information is associated with or describes the data; (3) what data are associated with or define the data; and (4) what other documentation categorizes the data.

"Data Base" Metadata

The first class -- what is needed to access the data -- is essentially what is referred to as metadata in the data base sense. Generically this metadata class includes data type primitives (e.g., byte, short, long, float, double, string) and structural information for each enumerated data object. The latter can include dimensionality (size, shape, [in]dependency), rank, positions, connections, and any referencing or indexing to other objects. For group objects, global information on the type of group would be necessary while similar information on individual members would by definition already be supported.

Attribute Metadata

The second class of metadata is various types of information that is associated with data, generally at the level of an individual data object. Consistent with the CDF/netCDF parlance this class is called attributes. The attribute class of metadata should be extensible by the user. In other words, the user is free to define new attributes, even though some application software should not be expected to operate on such user-specific information except to acknowledge its existence and be able to "list" it. A discussion of individual examples of such metadata can be seen in the user's manuals for CDF and netCDF.

Ancillary (Meta)data

In addition, to information about data there may be other data associated with data that are required for a complete definition. Such (meta)data is clearly numeric and could typically include spatial, temporal and spectral tags or locations associated with data, which can be accommodated via mesh specification.

Other (Documentation) Metadata

The last class of metadata helps the user to identify qualitative facts about the data. Such information is typically textual and probably is best supported by free text. There are several types of documentation in this category, which include laboratory notes or logs to identify what was seen, to identify what was done, etc. Another category would include a specification of what should be done with the data by the user or someone else. A third variety would include user and data set information to identify saved images or hardcopy, place data in the proper context, etc. These metadata are textual and could be supported in the same manner as text/string attributes. Examples of such metadata are user name, organization, the date of creation of a data set, image, script or log, and the source, name/title, version and history (e.g., audit trail of what has been done to the data.
from its initial raw form) of a data set. Without conventions for nomenclature, this category is unsupported in context — application software can only record and play back the text, not act on the stored information.

**Evaluation Criteria**

As implementations of specific data models have matured and been used, experience in their use have led to a host of issues concerning actual applications. In this sense, an enumeration of criteria to evaluate extant implementations can be helpful. Much of the underlying issues were discussed in a workshop on *Data Structures and Access Software for Scientific Visualization* conducted at the ACM SIGGRAPH conference in Dallas, TX in 1990. From the report of that workshop these issues can be divided into two categories, access and implementation. In summary, the access issues relate to:

- How should data be brought into a visualization system?
- What does a system let a scientist do with the data?
- What does a system let an application programmer do with the data?
- How should data be described (i.e., metadata and data attributes as discussed in the previous section)?
- How does a system preserve the fidelity of the data?

Given implementation approaches as outlined in the Implementations and Techniques section of this paper, the issues that arise are:

- How are code AND data to be portable across multiple platforms?
- How is interoperability between structures to be maintained?
- At what (low-level) should interfaces be independent of the user?

The participants in the aforementioned workshop generally agreed that to address these issues, the following is required:

- Utilize well-defined layering
- Employ C for coding and provide bindings to other languages as required (e.g., FORTRAN)
- Maintain consistent terminology in interfaces and structures and use them to define a context for functionality

**Scaling**

A major concern for extant systems is in their practical application to a number of large data sets. These issues often relate to scaling to even modest data sets by today's standards independent of addressing raw bandwidth. The data effectiveness of a system can be measured by its ability to handle multiple data sets simultaneously of various sizes, types, structures, etc. without forcing artificial constraints that disrupt the fidelity of the original data. Systems that support different classes of data separately will have difficulty scaling to support disparate data properly at the same time. Systems that support different classes of data uniformly do not because they effectively decouple the management of and access to the data from the actual visualization software.

Another area of scaling is in performance related to aggregate data size and visual complexity (e.g., numbers of polygons, pixels, voxels, etc.). Given current and planned data rates for scientific investigations, whether computational or observational, workstation demonstrations with a few MB of data are hardly relevant. For real science problems, an effective system should be able to scale up to the storage capacity of a given platform (e.g., memory + swap) independent of the CPU performance. Therefore, the software should be able to generate im-
ages, for example, slowly on a small workstation, for what would seem to be big data sets for that platform while operations on small data sets would be more interactive. Higher performance should be achievable via increased aggregate CPU performance, which should include parallelism.

Therefore, one should consider the following:

- **Data structure residency**

  Are the data structures in an implementation on secondary storage (i.e., disk) and/or in primary storage (i.e., memory)?

- **Transaction-like processing**

  What types of access methods for data are supported? Is the ability to do DBMS-like operations on data provided, which can be important for data sets that are too expensive to even partially reproduce?

- **Physical format and file structure**

  How does an implementation compensate for typical limitations in conventional file systems (e.g., blocking) or operating systems (e.g., paging) for bulk data access?

- **Distributed access to data within visualization systems** (see next subsection)

The implementations discussed in the Implementations and Techniques section of this paper address perhaps only a few of these points and are areas for which further research is required.

*Distributed Access*

Developing machine and media-independent distribution and network services for well-structured and supported science data presents a number of difficult technical problems. In the scientific community today and especially in the future there will be strong demands for the ability to easily access and share large volumes of complex data among computational facilities, network and data servers, and user/scientific workstations. Typically data must be physically moved by low-level network transfer or via off-line media, despite the capabilities of today’s computer systems. Such techniques are slow and clumsy, and will be increasingly inadequate for large data volumes.

Current Unix workstations from a number of manufacturers lend themselves ideally to the implementation of the aforementioned software. However, to be truly effective in the growing communications needs among heterogeneous scientific computing environments, other classes of systems must be accommodated. The mechanism for sharing should be driven by the scientific requirements of the applications and the structure of the data, not by the conventional limitations dictated by the architecture of the respective computer hardware and operating systems. The key to the development is the use of appropriate generic and portable techniques so that the software can operate on a wide variety of platforms that the ultimate users may have at their disposal. It is assumed that the software must utilize ANSI C as a programming language and the XDR protocol would appear to be a likely portable physical data protocol, but TCP/IP must be used as a low-level network protocol. (Many computer manufacturers, typically of Unix-based systems, have adopted XDR as their native protocol. Specific data models like netCDF and CDF utilize XDR.) However, the use of XDR does create additional overhead as the price for portability.

Work on defining the protocols for interprocess communication among heterogeneous platforms is needed. Unfortunately, many interprocess protocols are inefficient for migration of large data sets or data sets with large internal structures. For example, the Network File System (NFS) developed by Sun Microsystems and placed in the public domain has been adopted essentially by all workstation, minicomputer, mainframe and supercomputer manufacturers. Since NFS provides access at the file level not the science data object level, it would have limited performance to drive data-intensive applications.
Other mechanisms under consideration would include, for example, Unix (TCP/IP) sockets, the Remote Procedure Call protocol developed by Sun Microsystems and the Network Computing System (NCS) developed by (the former) Apollo Computer. All of these mechanisms are geared toward low bandwidth communications with small packet sizes. Although NCS is relatively new and it has only been adopted by a few computer manufacturers and it is still proprietary, NCS appears to offer better performance and more flexibility than NFS and XDR, but it may still be inefficient for shared, distributed access to large data sets. In addition, with the absorption of Apollo by Hewlett-Packard, there has been little new in NCS development. Hence, there is a need for further research to select an appropriate protocol.

Such a network data access protocol would be incorporated within a network shell to provide transparent access to distributed data. In this scenario, a visualization application running a workstation or a background server could have transparent access to large data sets stored on a data server, for example, at the applications programmer's level. For environments where task-to-task networking is neither available nor feasible (e.g., wide-area and other low bandwidth networks, non-networked systems), transport distribution services are required to provide computer-independent mechanisms for "shipping" data through a variety of media. Currently, visualization software systems such as AVS, IRIS Explorer and apE have a distributed execution model, in which computational tasks embodied as modules reside in separate processes, which may be on different machines. In this case, all communication among such processes is via sockets. Unfortunately, typical networking infrastructures place practical limits on this approach to relatively small data sets.

For example, NCSA has developed a Data Transfer Mechanism (DTM) as a simplified, RPC but more robust than the standard Unix RPC. It is a message passing system to support building of distributed applications. These messages are classes of self-describing data types, which obviously can include scientific fields. Such messages are at least 1 MB in size. NCSA provides an extensible software interface with access to the messages at multiple levels. This approach has shown some promise in on-going experiments at NCSA.

In parallel, NASA/Ames Research Center, has developed Distributed Library (dlib) as a tool to enable software developers to implement procedures on heterogeneous systems that communicate as part of distributed processing over a network. Employing only standard RPC, it takes the approach of maintaining state about the on-going processes and allocating memory for long-term communication among a client and server, for example. Dlib has been used for integrating visualization on a graphics workstation with computation on a supercomputer with packages like FAST. Hence, like DTM, one goal of dlib is to enable computational steering.

Conclusions

An effort to create data models and associated access software, especially to support scientific visualization involves the integration of various physical science disciplines and the computational sciences. It emphasizes the development of capabilities for a researcher to concentrate on doing science, freeing him or her from the mechanism of working with specialized data structures or formats. What is important is not the details of the technology, but what that technology can easily and inexpensively provide to promote science. Such activities will further the use of computer systems to support the management, analysis and display of any scientific data of interest under the control of the scientist doing research.

Additional Reading

The following is a brief list of selected papers, documents, and other material related to scientific data structures, formats, access software and systems for data management and visualization. Apologies are given for any documentation, software or other material inadvertently omitted.


Selected Representative Scientific Data Structures, Formats and Access Software

Table 2-4 compares the characteristics of several fairly generic structures and software that are used to access and utilize scientific data in applications such as visualization. This list covers examples that provide a uniform data model, access mechanism or both, which have been used in visualization systems. Neither the characteristics nor the list is meant to be complete. It is meant to be a "living" document that serves as a point of reference on data structures, formats and access software for software developers as well as users and generators of data. Therefore, some of the information is likely to be already out of date. Hence, anonymous ftp addresses on the internet are provided for access to the latest material. There are innumerable omissions to keep the information succinct and useful, for which apologies are given. Some examples that are very domain-specific or no longer in widespread use or not readily available have been left out. Others are visualization or application environments/software, which may support many of the characteristics enumerated in the table. They have been omitted for either a lack of information from the developing organization or the support mechanism is not uniform. With regard to the latter, in other words, the software provides multiple paths for access to distinct data types rather than a single one. Many of the references cited in the previous section will address most of these examples.

Public Domain Contacts for Information about Data Structures, Formats, and Access Software:

<table>
<thead>
<tr>
<th>Name</th>
<th>Contact</th>
<th>Address</th>
<th>Anonymous ftp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Data Format (CDF)</td>
<td>Greg Goucher</td>
<td><a href="mailto:goucher@nssdc.gsfc.nasa.gov">goucher@nssdc.gsfc.nasa.gov</a></td>
<td>nssdca.gsfc.nasa.gov</td>
</tr>
<tr>
<td><a href="http://nsadc.gsfc.nasa.gov/cdf/cdf_home.html">http://nsadc.gsfc.nasa.gov/cdf/cdf_home.html</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexible Image Transport System (FITS)</td>
<td>Don Wells</td>
<td><a href="mailto:dwells@nrao.edu">dwells@nrao.edu</a></td>
<td>nssdca.gsfc.nasa.gov</td>
</tr>
<tr>
<td><a href="http://fits.cv.nrao.edu/">http://fits.cv.nrao.edu/</a></td>
<td></td>
<td>fits.cv.nrao.edu</td>
<td>tetra.gsfc.nasa.gov</td>
</tr>
<tr>
<td>Hierarchical Data Format (HDF)</td>
<td>Mike Folk</td>
<td><a href="mailto:mfolk@ncsa.uiuc.edu">mfolk@ncsa.uiuc.edu</a></td>
<td>ftp.ncsa.uiuc.edu</td>
</tr>
<tr>
<td><a href="http://hdf.ncsa.uiuc.edu:8001">http://hdf.ncsa.uiuc.edu:8001</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Common Data Form (netCDF)</td>
<td>Russ Rew</td>
<td><a href="mailto:rew@unidata.ucar.edu">rew@unidata.ucar.edu</a></td>
<td>unidata.ucar.edu</td>
</tr>
<tr>
<td><a href="http://www.unidata.ucar.edu/packages/netcdf/index.html">http://www.unidata.ucar.edu/packages/netcdf/index.html</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portable Data Base (PDBlib)</td>
<td>Stewart Brown</td>
<td><a href="mailto:sabrown@phoenix.ocf.llnl.gov">sabrown@phoenix.ocf.llnl.gov</a></td>
<td>phoenix.ocf.llnl.gov</td>
</tr>
</tbody>
</table>

World-Wide-Web Pages for Information about Visualization Environments and Associated Data Structures, Formats and Access Software:

<table>
<thead>
<tr>
<th>Name</th>
<th>Contact</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVS</td>
<td>AVS</td>
<td><a href="http://www.avs.com">http://www.avs.com</a></td>
</tr>
<tr>
<td>Data Explorer</td>
<td>IBM</td>
<td><a href="http://www.almaden.ibm.com/dx/">http://www.almaden.ibm.com/dx/</a></td>
</tr>
<tr>
<td>FAST</td>
<td>NASA/Ames Research Center</td>
<td><a href="http://www.nas.nasa.gov/FAST/fast.html">http://www.nas.nasa.gov/FAST/fast.html</a></td>
</tr>
<tr>
<td>IRIS Explorer</td>
<td>NAG</td>
<td><a href="http://www.nag.co.uk/Welcome_IEC.html">http://www.nag.co.uk/Welcome_IEC.html</a></td>
</tr>
<tr>
<td>Khoros</td>
<td>University of New Mexico</td>
<td><a href="http://www.khoros.unm.edu/">http://www.khoros.unm.edu/</a></td>
</tr>
<tr>
<td>SciAn</td>
<td>Florida State University</td>
<td><a href="http://www.scri.fsu.edu/~lyons/scian/">http://www.scri.fsu.edu/~lyons/scian/</a></td>
</tr>
<tr>
<td>VIS-5D</td>
<td>University of Wisconsin</td>
<td>file://www.ssec.wisc.edu/~billh/vis5d.html</td>
</tr>
</tbody>
</table>
# Table 2-4. Selected Representative Scientific Data Structures, Formats and Access Software

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name/Characteristic</td>
<td>CDF</td>
<td>dx</td>
<td>FITS</td>
<td>Flux</td>
<td>HDF</td>
<td>netCDF</td>
<td>PDB</td>
<td>PLOT3D</td>
</tr>
<tr>
<td>2</td>
<td>Type</td>
<td>structure</td>
<td>format</td>
<td>format</td>
<td>structure/format</td>
<td>structure</td>
<td>structure</td>
<td>structure</td>
<td>format</td>
</tr>
<tr>
<td>3</td>
<td>Host Language Interface</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>no</td>
<td>&quot;4GL&quot;</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>Language Bindings</td>
<td>FORTRAN, C, IDL</td>
<td>C (via Data Explorer)</td>
<td>no</td>
<td>no</td>
<td>FORTRAN, C, C++</td>
<td>FORTRAN, C</td>
<td>FORTRAN, C, Scheme</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>Applications/Utilities</td>
<td>yes</td>
<td>yes (Data Explorer)</td>
<td>yes</td>
<td>yes (apE)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>Translation Tools</td>
<td>yes</td>
<td>yes</td>
<td>limited</td>
<td>?</td>
<td>limited</td>
<td>limited</td>
<td>yes</td>
<td>no?</td>
</tr>
<tr>
<td>7</td>
<td>Apple Macintosh</td>
<td>yes</td>
<td>N/A</td>
<td>via apE (AVUX)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
<td>via PLOT3D</td>
</tr>
<tr>
<td>8</td>
<td>Convex Unix</td>
<td>no</td>
<td>N/A</td>
<td>N/A</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
<td>via PLOT3D</td>
</tr>
<tr>
<td>9</td>
<td>Cray Unixos</td>
<td>no</td>
<td>N/A</td>
<td>N/A</td>
<td>via apE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>via PLOT3D</td>
</tr>
<tr>
<td>10</td>
<td>Digital Unix</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>via apE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>DEC VMS</td>
<td>yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>via PLOT3D</td>
</tr>
<tr>
<td>12</td>
<td>HP-UX</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>via apE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>13</td>
<td>IBM AIX</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>14</td>
<td>IBM MVS</td>
<td>no</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>IBM OS/2</td>
<td>no</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>N/A</td>
<td>no</td>
<td>yes</td>
<td>?</td>
<td>N/A</td>
</tr>
<tr>
<td>16</td>
<td>Linux</td>
<td>yes</td>
<td>no</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
<td>no</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>17</td>
<td>MS-DOS</td>
<td>yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>N/A</td>
</tr>
<tr>
<td>18</td>
<td>Silicon Graphics IRIX</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>via apE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>via FAST</td>
</tr>
<tr>
<td>19</td>
<td>SunOS, Solaris</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>via apE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>via PLOT3D</td>
</tr>
<tr>
<td>20</td>
<td>MS Windows NT</td>
<td>no</td>
<td>via Data Explorer</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
<td>no</td>
<td>?</td>
<td>N/A</td>
</tr>
<tr>
<td>21</td>
<td>MS Windows 3.1/95</td>
<td>yes</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
<td>N/A</td>
</tr>
<tr>
<td>22</td>
<td>Network Computing</td>
<td>NeXT</td>
<td>DG (Data Explorer)</td>
<td>N/A</td>
<td>Stardent, NeXT</td>
<td>CM-5, CTSS, ...</td>
<td>NeXT</td>
<td>MIPS, Intel</td>
<td>N/A</td>
</tr>
<tr>
<td>23</td>
<td>Other Operating Systems</td>
<td>yes, implicit</td>
<td>yes, implicit</td>
<td>yes, implicit</td>
<td>yes</td>
<td>yes, implicit</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>24</td>
<td>Point/Scattered Data</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>25</td>
<td>Curvilinear Meshes</td>
<td>indirect</td>
<td>yes</td>
<td>limited</td>
<td>limited</td>
<td>via VSET</td>
<td>indirect</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>26</td>
<td>Irregular Meshes</td>
<td>indirect</td>
<td>yes</td>
<td>no</td>
<td>?</td>
<td>via VSET</td>
<td>no</td>
<td>?</td>
<td>no</td>
</tr>
<tr>
<td>27</td>
<td>Unstructured Grids</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
<td>via VSET</td>
<td>no</td>
<td>?</td>
<td>no</td>
</tr>
<tr>
<td>28</td>
<td>Sparse Matrices/Invalid Data</td>
<td>indirect</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>indirect</td>
<td>?</td>
<td>no</td>
</tr>
<tr>
<td>29</td>
<td>Time and Other Series</td>
<td>yes</td>
<td>yes</td>
<td>limited</td>
<td>?</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>30</td>
<td>Heterogeneous, Composites</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>limited</td>
<td>via VSET</td>
<td>no</td>
<td>?</td>
<td>no</td>
</tr>
<tr>
<td>31</td>
<td>Multizone Grids</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Dimensionality</td>
<td>no limit (def. $10$)</td>
<td>no limit</td>
<td>999</td>
<td>?</td>
<td>SDS: $7$</td>
<td>no limit (def. $32$)</td>
<td>no limit</td>
<td>3</td>
</tr>
<tr>
<td>33</td>
<td>Scalar</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>34</td>
<td>Vector</td>
<td>as scalars</td>
<td>yes</td>
<td>as scalars</td>
<td>yes</td>
<td>as scalars</td>
<td>no</td>
<td>?</td>
<td>no</td>
</tr>
<tr>
<td>35</td>
<td>Tensor</td>
<td>as scalars</td>
<td>yes</td>
<td>as scalars</td>
<td>no</td>
<td>?</td>
<td>as scalars</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>36</td>
<td>Variables, Fields, etc.</td>
<td>no limit (def. $128$)</td>
<td>no limit</td>
<td>no limit</td>
<td>?</td>
<td>?</td>
<td>no limit (def. $512$)</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Table 2-4. Selected Representative Scientific Data Structures, Formats and Access Software

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Name/Characteristic</td>
<td>CDF</td>
<td>dx</td>
<td>FITS</td>
<td>Flux</td>
<td>HDF</td>
<td>netCDF</td>
<td>PDB</td>
<td>PLOT3D</td>
</tr>
<tr>
<td>41</td>
<td>Data Type Support</td>
<td>byte</td>
<td>byte</td>
<td>byte</td>
<td>byte</td>
<td>byte</td>
<td>byte</td>
<td>byte</td>
<td>byte</td>
</tr>
<tr>
<td>42</td>
<td>Physical Data Level</td>
<td>XDR, native, ASCII</td>
<td>IEEE, ASCII</td>
<td>XDR, native, ASCII</td>
<td>multiple</td>
<td>XDR</td>
<td>custom, portable</td>
<td>native, ASCII</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Byte/Logical/char</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>44</td>
<td>Integer*2/short</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>45</td>
<td>Integer*4/long</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>46</td>
<td>Integer*8/hyper</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>47</td>
<td>Signed &amp; unsigned integers</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>48</td>
<td>Real*4/float</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>49</td>
<td>Real*8/Double</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>50</td>
<td>Complex*8/complex</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>51</td>
<td>Complex*16</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>52</td>
<td>Quaternion (int, float, …)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>53</td>
<td>Pointer</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
<td>no</td>
<td>y</td>
<td>no</td>
</tr>
<tr>
<td>54</td>
<td>Character Strings</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>55</td>
<td>Imbedded Metadata</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>56</td>
<td>High-Level I/O Access</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>57</td>
<td>Sequential Access</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>58</td>
<td>Random Access</td>
<td>yes</td>
<td>via Data Explorer</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>59</td>
<td>Physical Files</td>
<td>single or multiple</td>
<td>single</td>
<td>single</td>
<td>single</td>
<td>single</td>
<td>single</td>
<td>single</td>
<td>single</td>
</tr>
<tr>
<td>60</td>
<td>Compression</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>61</td>
<td>Array Majority</td>
<td>row or column</td>
<td>row</td>
<td>column</td>
<td>row</td>
<td>row</td>
<td>row</td>
<td>row or column</td>
<td>column</td>
</tr>
<tr>
<td>62</td>
<td>Structure Driver</td>
<td>data</td>
<td>data</td>
<td>physical data</td>
<td>data</td>
<td>data</td>
<td>data</td>
<td>data</td>
<td>data</td>
</tr>
<tr>
<td>63</td>
<td>In-Place Edits/Transactions</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>limited</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>64</td>
<td>Data Structure Residency</td>
<td>disk</td>
<td>memory (DX)</td>
<td>N/A</td>
<td>memory (apE)</td>
<td>memory</td>
<td>disk</td>
<td>?</td>
<td>N/A</td>
</tr>
<tr>
<td>65</td>
<td>Primary Orientation</td>
<td>application</td>
<td>application</td>
<td>transport</td>
<td>application</td>
<td>transport</td>
<td>transport</td>
<td>application</td>
<td>application</td>
</tr>
<tr>
<td>66</td>
<td>Transport Mechanism</td>
<td>copy</td>
<td>copy</td>
<td>copy</td>
<td>copy</td>
<td>copy</td>
<td>copy</td>
<td>copy</td>
<td>copy</td>
</tr>
<tr>
<td>67</td>
<td>Astronomy/Astrophysics</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>68</td>
<td>CF (a.g., aerospace)</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>69</td>
<td>Chemistry</td>
<td>limited yes</td>
<td>no</td>
<td>yes</td>
<td>limited</td>
<td>limited</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>70</td>
<td>Meteorology</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>71</td>
<td>Oceanography</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td>Oceanography</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>73</td>
<td>Planetary Sciences</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>74</td>
<td>Space Physics</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>75</td>
<td>Other Disciplines</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>applicable</td>
</tr>
</tbody>
</table>

II-30
| Cell: B1 | Note: NSSDC Common Data Format. CDF was developed by the NSSOC and NASA/Goddard Space Flight Center and has become a standard in the NASA space and earth sciences community for storage and applications systems. |
| Cell: C1 | Note: Data Explorer (format). D* is the external disk representation of the data model of the IBM Visualization Data Explorer developed by IBM T. J. Watson Research Center. |
| Cell: D1 | Note: Flexible Image Transport System. FITS, developed by the National Radio Astronomy Observatory, is the primary standard interchange format for the astronomical community and is a driver for some powerful image processing applications. |
| Cell: E1 | Note: Flux. Flux is the internal general data format, which supports the animation production environment, apE, a visualization system developed by the Ohio Supercomputer Center and now owned by TaraVisuals, Inc. |
| Cell: F1 | Note: Hierarchical Data Format. The National Center for Supercomputing Applications initially developed HDF to facilitate the transport of data generated by supercomputer simulations to other computer systems. It is supported by a number of useful and popular scientific tools and has become widely available. |
| Cell: G1 | Note: Network Common Data Form. Unidata developed its own implementation of the CDF data model for the machine-independent access and transport of multidimensional data, which has become widely available. |
| Cell: H1 | Note: PDB. PDB was developed at Lawrence Livermore National Laboratory to facilitate the storage and retrieval of a variety of data structures in a portable fashion. |
| Cell: I1 | Note: PLOT3D. The PLOT3D format was developed at NASA/Ames Research Center to support a number of CFD visualization and analysis applications, including PLOT3D and more recently, FAST. |
| Cell: A2 | Note: Type. Is the standard a format or a data structure? |
| Cell: A3 | Note: Host Language Interface (HLI) or Applications Programming Interface (API). Does the standard support a software package or other mechanism that provides access to its data structure(s) such as a subroutine package or library? |
| Cell: A4 | Note: Language Bindings. If the standard has an HLI or API, does that interface support extensions to a host language’s data types and operations as abstractions? |
Table 2-4. Selected Representative Scientific Data Structures, Formats and Access Software

| Cell: A5 | Note: Applications. Are there application software systems or utilities available that operate on the standard (e.g., visualization, manipulation)? Standards that are currently in design or development may lack such applications today, but they may be planned. |
| Cell: A6 | Note: Translation Tools. Are there software packages or utilities that enable different classes of data to be recast, filtered, modelled or translated into the standard? |
| Cell: A7 | Note: For what computer systems is the standard implemented? For formats as opposed to structures with access software, N/A (not applicable) is generally indicated. If there is applications software that operates on a particular system that utilizes the format directly, it is indicated. |
| Cell: A25 | Note: Regular Grids. Does the standard support the definition and access to regular (rectilinear) grids typically as multidimensional parameters? Is the grid structure implicit (e.g., via product or other specification) or explicit? |
| Cell: A26 | Note: Point/Scattered Data. Does the standard support the definition and access to simple point data? |
| Cell: A27 | Note: Curvilinear Meshes. Does the standard support the definition and access to complex grids or meshes such as deformed or irregular grids on non-cartesian topologies? |
| Cell: A28 | Note: Irregular Meshes. Does the standard support the definition and access to complex grids or meshes that are fully irregular? |
| Cell: A29 | Note: Unstructured Grids. Does the standard support the definition and access to complex grids or meshes that are unstructured on constant or irregular topologies (e.g., finite element)? |
| Cell: A30 | Note: Sparse Matrices/Invalid Data. Does the standard support the definition, access and compact storage of sparse matrices or other representations of regions of invalid or "missing" data? |
| Cell: A31 | Note: Does the standard support the definition and access to series of parameters, such as a time dependency? |
| Cell: A32 | Note: Hierarchies. Does the standard support the definition and access to groups, hierarchical or collections of parameters such as aggregates, geographic, tree, etc. structures? |
| Cell: A33 | Note: Multizone Grids. Does the standard support multizone or structures represented by multiple grids. Any arbitrary limit or default on the number of grids is indicated. |
| Cell: A35 | Note: Does the standard support multidimensional parameters or fields. Any arbitrary limit or default on the number of dimensions is indicated. |
| Cell: A36 | Note: Scalar. Does the standard support the definition and access of scalar fields? |
| Cell: A37 | Note: Vector. Does the standard support the definition and access of vector fields? Explicit support by components as scalars is indicated. |
| Cell: A38 | Note: Tensor. Does the standard support the definition and access of tensor fields? Explicit support of components as scalars is indicated. |
| Cell: A39 | Note: Variable, Fields, Parameters, etc. Any limit on the number of distinct items supported in the standard are indicated. |
| Cell: A41 | Note: Data Type Support. At what level does the standard provide or define access to data elements? What primitive data types are supported by the standard for the definition of data elements? |
| Cell: D41 | Note: Metadata can include an algorithm for conversion to real for data and grid structure. |
| Cell: A56 | Note: Imbedded Metadata. Does the standard support self-describing information on the data objects that it contains? |
| Cell: A57 | Note: High-Level I/O Access. Does the standard provide access to data elements at an object level? |
| Cell: A58 | Note: Sequential Access. Does the standard support sequential access to the data elements that it contains? |
| Cell: A59 | Note: Random Access. Does the standard support random access to the data elements that it contains? |
| Cell: A60 | Note: Physical Files. Does the standard support each instance of itself via one or more physical files in a file system? |
| Cell: A61 | Note: Array Majority. How does the standard store array elements, either row major (last dimension varies fastest, the convention in the C programming language) or column major (first dimension varies fastest, the convention in the FORTRAN programming language). |
| Cell: A62 | Note: Structure Driver. Is the driver of the structure of data objects in the standard the data themselves or the physical organization of the data? |
| Cell: A63 | Note: In-Place Edit/Transactions. Once data are defined in terms of the standard can an application edit, extend or delete the metadata or the data elements in-place without violating the integrity of the standard or copying physical files (e.g., through the host language interface)? |
| Cell: A64 | Note: Data Structure Residency. For standards that are associated with or only accessible via software, are the data resident in memory or on disk potentially with associated caching? |
| Cell: A65 | Note: Primary Orientation. Is the driver of the structure and design of the standard for building applications systems or for the transportation or interchange of data? |
| Cell: A66 | Note: Transport Mechanism. What the procedure for moving data in the standard between different computer systems (e.g., off-line (tape) or on-line (ftp) copy, remote procedure call)? |
| Cell: A67 | Note: For what scientific or engineering discipline(s) is the standard designed to support? |
3. Interactive Archives for Scientific Data

Introduction

The instrumentation technology behind data generators in a myriad of disciplines, whether observational or computational, is rapidly improving, especially in the earth and space sciences. The capability to produce data is typically growing much faster than the techniques available to manage and use them. Traditional bulk access to data archives do not scale well to the volume and complexity of current or planned data streams nor to demanding applications of such data, such as analysis and modeling. A fundamental change in the modes of accessing archives, from static, batch systems to dynamic, interactive systems, is required. As a first step in enabling this paradigm shift, consider appropriate data management techniques coupled with the process or methods of scientific visualization. The technologies that support visualization show promise in helping to address some of these issues.

Data management

Data management issues relevant to the access and utilization of large scientific data sets (e.g., locating, retrieving, and using data of interest) may be classified into four levels:

I. At the back-end are warehousing issues related to problems of media, bandwidth, protocols, formats.

II. Above that are data models, access techniques, data base systems, query languages, and programming interfaces.

III. As a medium between the access of data and the user, consider browsing as an aid for feature identification, which serves as a guide in the data selection process.

IV. At the front-end are human factors issues (e.g., system ergonomics, human perception) and the incorporation of domain-specific knowledge to permit a system to be usable and useful (e.g., domain-driven task-based interfaces and tools).

At level I, high-speed and high-capacity storage and networking are either available now or will be common in the near future. A chief concern is not the volume of data nor the signaling speeds of these devices, but the effective capacity and bandwidth that can be utilized by applications that require warehousing (i.e., above level I). Typical storage and communications protocols are not a match for GB archives, yet current archives are in the TB (230 bytes) range, while those being planned are measured in PB (250 bytes). In addition, accepted data distribution mechanisms such as CD-ROMs hardly scale to data rates planned for projects like NASA's Earth Observing System, which are measured in TB/day (1 TB/day is about 2000 CD-ROMs/day).

At level II and to a limited extent level I, consider what is stored and how it is stored. There are many interfaces and structures developed by or used in the scientific community (Treinish, 1992b; Brown et al, 1993), which are driven by application, politics, tradition and requirements. For the kind of data rates that need to be supported for specific computational codes, applications, etc., it is NOT sufficient to provide only bulk access. The software that accesses the data must be able to do so in a meaningful way. Self-describing physical storage formats and structures should be used. The self-describing formats must have associated disk-based structures and software interfaces (e.g., at least abstract data types for programmers and applications, and simple utilities for those who do not wish to program). They must be transparently accessible at the desk of an end-user. These structures and interfaces must be consistent with the high-speed access to be provided. This must also include task-to-task communication (e.g., transparent workstation access to high-speed storage).

Such requirements demand the utilization of a generalized data model as a mechanism to classify and access data as well as to map data to operations. Further, it defines and organizes a set of data objects (Treinish, 1992b). The model (e.g., Haber et al. (1991)) must provide a self-describing

- representation of the physical storage structure (e.g., format)
- structural representation of the data (e.g., data base schema)
- higher-level logical structure

Since access to data organized via such a model is described in terms of a logical structure, a generalized data model provides the foundation for implementing a data access server, as shown in Figure 1. Without such a data model, performance of high-speed devices will not be realized. High data-rate computations like signal processing, visualization, and some classes of modeling also require such support.

Level II requires enabling tools for data/information systems. Having efficient storage and access are critical for driving applications, but will not be directly useful for helping find data of interest. At the very least, there is a need for low-level metadata management that enables both content and context for the warehousing information. Traditionally, a relational data base management system (RDBMS) could be used for its implementation. Although this concept was first prototyped over 10 years ago, a RDBMS cannot handle semantics, spatial information, or the data volume (Treinish and Ray, 1985). The RDBMS would not have any data per se, but would have pointers to the bulk storage enabling simple querying of what is there and where it is stored by characteristics such as spacecraft, instrument, mission, code, date/time, investigator, and owner. It would have to be supplemented with a non-relational system to adequately support spatial metadata (e.g., Fekete (1990)), as shown in Figure 1.

![Figure 1. Data management for interactive archives.](image)

There are many challenges in the implementation of an effective data system at level II providing complete access to data of interest simply and easily. A key role for data systems is as a means of efficient and intelligent querying and searching for relevant data based upon an assumption that the data volume and complexity is sufficiently large that practical examination of more than a tiny fraction of archive contents is prohibitively expensive. Therefore, the implementation of searches as meta-operations on abstractions of the archive (e.g., contextual, domain-driven, spatial, user-driven, and visual), though difficult, would be highly beneficial. Visual searches or browsing imply the perusal of pictorial representations of data or their abstractions with sufficient content for a user to determine the utility for further, more detailed examination. Practical methods of scientific visualization developed fairly recently show promise in the implementation of visual searches. Hence, consider a very simple notion: looking at data of interest in an "appropriate" fashion to determine the merit of accessing such data for further study. Thus, visualization can be an adjunct to archive management.

**History**

The idea of visually browsing data is hardly new. Scientists have always used visual abstractions (on paper) of their experimental results for communication. Often these same visualizations en masse were filed and later perused to help recall findings from earlier work. The mechanism for storing, scanning, and distribution of such visualizations was the same as for text -- initially shelves and files of paper followed by boxes of microfiche. With the advent of digital data collection instrumentation (e.g., seismic soundings for oil prospecting, study of the earth's atmosphere from spacecraft), this same paradigm was adopted with computer-generated paper-

11-36
and later microfiche-based visualizations. These visualizations were based upon the graphics technology of the era and the traditions in these fields (e.g., line drawings of iso-contours). Monochromatic microfiche became an effective means of storing visual information compactly and were relatively inexpensive to generate after the cost of the production equipment was amortized. Further, they were easily distributed, were cheap to examine remotely, and were archiveable. These are important virtues to consider in a modern visualization approach to browsing that goes beyond the relatively limited medium of microfiche.

For example, figure 2 is a photographic reproduction of a microfiche (approximately three inches by five inches in size) showing contour maps of daily total column ozone observed by a NASA spacecraft over Antarctica during two months of 1986. Although the system that generated the microfiche is primarily intended for interactive use in the analysis of many kinds of earth and space science data (Trenish, 1989), the needs of a low-cost browsing medium could only be met until fairly recently by employing its tools in the batch production of microfiche.

The advent of digital instrumentation, especially in remote sensing, created another role for browsing that is based upon the assumption that there is insufficient computing resources to support the analysis of all acquired data from all experiments of a specific project or mission. Hence, visual abstractions were created to help identify what subset of data should actually be fully processed to support scientific investigations. These graphical representations, known as summary plots, were generated on microfiche and distributed to all participants in a project. They usually contained a time sequence of simple visualizations of instrument outputs in specific modes at sufficient resolution to suitably identify a set of "events" as interesting, and thus warrant further processing. The particular presentations were chosen to highlight the signatures of such events within the limited graphical techniques that could be supported on monochromatic microfiche (line drawing, simple gray-scale polygon fill). Users of such a mechanism would receive a stack of microfiche each week, for example, and visually browse them, looking for patterns in the plots that would be indicative of something interesting (Trenish, 1982).

With improvements in technology to support interactive computing, most efforts associated with data browsing at level III in the aforementioned hierarchy of archive data management and access, have focused on simple images with image data. Furthermore, implementations are generally confined to one data set or a small number of similar data sets (cf, Simpson and Harkins, 1993; Oleson, 1992). This has been a conscious choice based upon the need for supporting only a limited domain and/or for driving interactivity. Unfortunately, these techniques are not applicable to many classes of data or when multiple disparate data sets are to be considered simultaneously, which are the characteristics of most current or planned archives.

Approach

There are four issues associated with interactive data browsing using visualization from archives of scientific data:

A. What are the visual abstractions for data presentation and interaction?

B. How are the browsing products distributed?

C. How is interactivity achieved?

D. What is the mechanism for integrating browsing into a data system?

Figure 3 is a schematic of a simple interactive archive system. Each of the aforementioned levels of data management are shown, such that level I services are provided via archive, warehouse, and data distribution servers. Level II services are within the gray box: data and information server, and metadata server. Figure 1 illustrates the relationship between these components and specific data management technologies.
Browsing (level III) is provided via the browse server and local and remote clients. At this level of detail, any level IV services would be embedded within the local and remote clients. The schematic only indicates the major interaction paths as arrows between each component. The arrow thickness corresponds to relative bandwidth of communications, where thickest arrows would imply the greatest bandwidth (e.g., ANSI High-Performance Parallel Interface or HIPPI), and the thinnest arrow would imply the least (e.g., Ethernet). A jagged arrow refers to traditional remote network communications. In addition, remote communications could be disconnected by distribution of products to a remote site via an alternate medium (e.g., CD-ROM) to be utilized with a local client at the remote site.

**Abstraction**

Efforts in scientific visualization typically focus on the analysis of data in a post-processing mode. This is still the emphasis even with the potential advent of computational steering or telescience, the remote sensing equivalent. Browsing is a subjective process involving the human visual system, which is consistent with one of the origins of the notion of scientific visualization as a method of computing. Consider what happens when an individual as an observer walks into a room. That person scans the room and identifies and categorizes the contents (objects) in the room. From the classification or browsing process, further interaction with those contents may take place. Hence, the requirements for qualitative browsing presentation are not the same as the requirements for analysis.
As with primitive summary plots, browsing visualization methods should intentionally transform the structure of the data to highlight features to attract attention, scale dimensions to exaggerate details, or segment the data into regions (Rogowitz and Treinish, 1993a). Traditional image browsing is very limited in this regard. Highlighting can only be achieved by modifying a pseudo-color or false-color presentation, which may not be sufficient even for image data or data than can be represented by an image. Of course, this method cannot even be considered for data that cannot be visualized as a simple image (e.g., any non-scalar data, any data of more than two dimensions).

Visual highlighting is often ill-advised for analysis because of the intentional distortion. In general, the browsing visualization should show data in the proper context (e.g., spatial or temporal). This is especially true in the earth and space sciences, where acquired data need to be registered in some (geographic) coordinate system for viewing. If the browsing visualization is to show multiple parameters, then the highlighting techniques must preserve the fidelity of the data (e.g., regions of missing data, original mesh structure) well enough, so that the artifacts in the presentation are not erroneously interpreted as potential (correlative) features in the data. Although the focus of such techniques for browsing is not for quantitative study, they may have a role in such tasks (Treinish, 1993b).

**Distribution**

One potential problem with the visual browsing of large data archives is the need to be in relatively close proximity to either the data archives or a facility that can generate the browse products (i.e., high-bandwidth access to an interactive system). In general, high-bandwidth communications between an archive and its users is not always practical, given typical geographic dispersal of scientists and their need for utilizing more than one archive.

Since interactive browsing has usually considered only image data, the distribution of browse products has focused on the distribution of these same images in some form, usually lossy compressed, because of the typical size of image archives (Simpson and Harkin, 1993). In this sense, the browsing and visualization media and the data are all the same. Hence, the compression is of the original data. The compressed data will be easier to distribute than the original data, but compression is applicable only to a limited class of data (i.e., images). Furthermore, the resultant compressed form may not be suitable for browsing, and the quality may be poor.

Alternatively, consider the distribution of compressed visualizations instead of compressed data. These visualizations would be represented as images, so that available image compression technology could be utilized. As with image data, uncompressed visualizations would be of high quality but of sufficient volume to be impractical to distribute to remote sites. The compressed visualization images would be easy to distribute, as with compressed image data. The key difference is that the compressed visualization could apply to any collection of data and would be of high quality. Hence, one could consider the compressed visualizations as the summary plots of the 1990s, because the lack of resources to access or distribute all archived data or their abstractions is similar to the lack of resources to process or analyze all acquired bits, which motivated the generation of summary plots in the past.

Such an approach could be extended for predefined access and browsing scenarios, where the compressed visualizations are available for on-line remote access or via CD-ROM. In this case, the viewing of such compressed visualizations would require relatively simple, low-cost display software and hardware, which is becoming more readily available in desktop environments. In general, the access and distribution costs associated with compressed visualizations will be similar to those associated with image data, but will, of course, be significantly less than those associated with the original data. Generating the visual browse products obviates the need to distribute the actual data, but entails an additional cost, which would be justified given the added value.

**Interactivity**

For visual browsing to be effective, it must be interactive. Otherwise, it is little different than watching television or traditional image browsing. One aspect of the interaction is in terms of data management: selection of and access to data of potential interest and metadata to help in guiding that process. In terms of visualization,
the ability to interact with the browser "objects" (e.g., spatially, temporally) in near-real-time is critical. This requires rapid access to the data and rapid generation of the browser products.

Implementation

Abstraction

The requirements to create qualitative visualizations that are effective as browser products do have implications on the software used to create them. Registration of data into an appropriate coordinate system for viewing requires the support of user-defined coordinate systems. To be able to properly show more than one data set simultaneously requires the ability to operate on different grid structures simultaneously and to transform grid geometry independent of data. Depending on the visualization strategies being used, rendered images may need to contain different geometries (e.g., points, lines, surfaces, and volumes independent of color or opacity or of original grid structure). (See Trenish (1993b) for a discussion of these ideas with respect to data analysis.)

A commercial scientific visualization environment (IBM Visualization Data Explorer or DX) has been used to experimentally implement the aforementioned browsing techniques. DX is a general-purpose software package for scientific data visualization. It employs a data-flow-driven client-server execution model and is currently available on Unix workstation platforms manufactured by Sun, SGI, IBM, HP, DG and DEC (Lucas et al, 1992). It is also available on a parallel supercomputer, IBM POWER Visualization System.

Distribution

Near-real-time browsing of visualizations at sufficient resolution to enable the user to see relevant features requires the distribution of a large number of images. Clearly, lossy compression is necessary to drive viewing with update rates near the refresh rate of display controllers. For utilization remote from the archive or browser server, low-cost, simple display software and hardware are needed as well. There is much literature on data compression strategies and algorithms, but these will not be discussed herein. (A few hundred citations are reported over the last decade by the National Technical Information Service alone, (NTIS, 1992).) This notion of visualization distribution is built upon the extant and growing body of implementations of compression algorithms, which are being utilized in scientific, multimedia, and entertainment applications.

The idea of distributing imagery for visualization is not new. For example, Johnston et al (1989) experimented with both block-truncation and Lempel-Ziv compression for the distribution of visualization animation. Rombach et al (1991) discussed the Joint Photographic Experts Group (JPEG) compression scheme for the distribution of cardiographic imagery from different sources like ultrasound, magnetic resonance imagery, and angiography. In these and other cases, the authors considered a low-cost viewing environment on the desktop as being critical, especially if the expense of generating the images to be distributed is high. Therefore, in this initial implementation, block-truncation (lossy, i.e., reduces the number of colors to represent a full-color pixel), modified Lempel-Ziv (lossless, e.g., like Unix compress), and temporal coherence between animation frames (lossy, e.g., the Moving Picture Experts Group, MPEG (LeGall, 1991)) will be considered.

To illustrate the viability of this approach, consider a modest data set, composed of a rectilinear scalar field of 32-bit floating-point numbers (e.g., atmospheric temperature) at one-degree of geographic resolution at seven levels in the earth's atmosphere. Each time step would require about 1.7 MB (1 MB = \(2^{20}\) bytes). If these are daily data, then less than a year would fit on a single CD-ROM, uncompressed. This does not include ancillary data required for annotation, such as coastline maps, topography, other reference material. Lossy compression would not be relevant since the data are not imagery. Lossy compression could be applied to each layer of the atmosphere individually. However, the results would be rather poor (i.e., the two-dimensional spatial resolution is already low, 180 x 360), and spatial coherence for the entire volume could not be maintained. If lossless compressed, decompression could be expensive (e.g., using Lempel-Ziv) or inconvenient (e.g., using scaled/encoded 12- or 16-bit integers). Either compression approach is highly sensitive to the contents of the data set.

Alternatively, visualization compression is independent of data characteristics, and only the resolution of the visualization image(s) drives the compression/distribution/decompression cost. Of course, the distribution of
uncompressed browse visualizations is expensive, potentially more than that of the uncompressed data. Lossless compression, although cheaper to distribute, would still require the decompression process. The decompression of losslessly compressed images could also be more expensive than the decompression of losslessly compressed data. Hence, the lossy compression of the visualization imagery is the best approach both from a cost perspective as well as from the perspective of image quality. Hence, for sequence of 640x480 24-bit image representations of the simple volumetric data set, over 14 years' worth of frames for such daily data could be stored using a simple 8:1 block truncation compression (i.e., each 24-bit pixel is represented by three bits) on a single CD-ROM. Using 3:1 JPEG compression, a sequence of over two years of these images at hourly resolution could be stored on a single CD-ROM.

Interactivity

For browsing to be effective it must be interactive with near-real-time system response. With data sets of interesting size, e.g., ≥ O(1 GB), immediate interaction cannot take place on current conventional systems (i.e., high-end graphics workstations). Even though a 1 GB data set is admittedly modest by today's standards for data generation, the access and visualization of an entire data set or even a large fraction of it for browsing, may place significant burdens on the floating point and bandwidth capacities of the computer system being employed. The bandwidth requirements are derived from the bulk access speeds of large data sets and the transmission of images sufficiently fast to be interactive. The floating point requirements stem from three classes of computation for visualization:

1. Transformation (e.g., warping, registration)
2. Realization (e.g., contouring, color mapping, surface deformation)
3. Rendering (i.e., creating images)

Although the visualization requirements are different, the computational needs of interactive browsing are very similar to those of visualization in a virtual world environment (Bryson and Levit, 1991).

Experimentation with a commercial parallel supercomputer (IBM POWER Visualization System) and the aforementioned DX environment has shown the viability of such interactive visual browsing, even with multiple data sets. In this effort, the PVS and DX combination has been used as the browse server with local and remote clients on workstations as indicated in figure 3. The PVS functions as an archive server in this context. Figure 4 shows the relationship between the browse server and the data and information server and its components in the interactive archive system shown in figure 3.

The IBM POWER Visualization System (PVS), which was introduced in 1991, is a medium-grain, coherent shared-memory parallel supercomputer with the interactivity of a workstation. This has been achieved not via special-purpose hardware but instead via a programmable (general-purpose) approach that maintains balance among floating point performance via moderate parallelism, large physical memory, and high-speed external and internal bandwidth. The PVS consists of three major hardware components: server, disk array, and video controller. The server is a symmetric multi-processor with up to 32 processors (40 MHz Intel i860XR or 44 MHz Intel i860XP), a 1.28 GB/sec (at 40 MHz) internal backplane supporting a hierarchical bus structure, hierarchical memory (16 MB local memory per processor and up to 2 GB global/shared memory), ANSI HiPPI communications, fast and wide SCSI-2, and an IBM RISC System/6000 support processor (for local area network and storage access). The server supports parallelized computations for visualization via DX. The disk array is a HiPPI-attached RAID-3 device with either 50 MB/sec or 95 MB/sec sustained access speeds, or a fast and wide SCSI-2 four-bank RAID-3 device with 76 MB/sec sustained access speeds. It provides access to archived data to be browsed. The video controller is a programmable 24-bit double-buffered frame buffer with 8-bit al-
pha overlay (for custom XWindow server) attached to an IBM RISC System/6000 workstation. It receives images from the PVS server via HiPPI (either compressed or uncompressed) at resolutions up to 1920x1536, including HD1V, for real-time image updates. The video controller provides an interface for interaction with and viewing of the browsing visualization at speeds up to 95 MB/sec.

Results

Abstraction

Figure 5 shows a traditional two-dimensional visualization of ozone data similar to those shown in figure 2. The data are realized with a pseudo-color map and iso-contour lines for September 30, 1992. The rectangular presentation of the data is consistent with the provided mesh in that it is torn at the poles and at a nominal International Date Line. The ozone data are overlaid with a map of world coastlines and national boundaries in magenta as well as fiducial lines (of latitude and longitude) in white.

To provide a qualitative impression for browsing, the data are transformed to a three-dimensional continuous spherical surface in figure 6. The ozone is triply redundantly mapped to radial deformation, color, and opacity so that high ozone values are thick, far from the earth, and reddish while low ozone values are thin, close to the earth, and bluish. Replacing the map for annotation is a globe in the center of this ozone surface. The use of three redundant realization techniques results in textures for qualitatively identifying regions of spatial or temporal interest. The gauges on the left illustrate the daily total ozone statistics. The pseudo-hour hand position ranges from 100 to 650 Dobson Units, while the color corresponds to that of the ozone surface. From the top they show the mean, minimum, and maximum for each day. The value corresponding to geographic view for each frame is shown next. At the bottom is a bar chart indicating the standard deviation of the daily measurements. This approach to qualitative visualization is potentially applicable to a large variety of simulated or observed earth and planetary data on a large spatial scale, especially for two and three-dimensional scalar fields.

A browsing animation of the ozone would be one that illustrates the data on a daily basis as in figure 6 for the entire archive of available data (i.e., from late 1978 through early 1993). The geographic view of these data would change with each day to provide reasonable coverage of the entire globe over a complete year. The view would be chosen to concentrate on interesting regions such as the poles during appropriate seasons like Spring. Treinish (1992a) and Treinish (1993a) present examples of such browsing animations. Since these animations were produced, they have been useful for identifying periods of time or geographic regions that warrant further study.

These browsing animation sequences are derived from about 1 GB of data, a two-dimensional scalar field over a torn geographic mesh. For each of the more than 4700 frames of the sequence, several calculations are required to create a visualization image. Each image is actually composed of two images that have been blended. There is a background image, which is composed of frame-variant contents under a constant view: primarily opaque polygonal text, dials, and bars. This annotation changes to summarize daily statistics. There is a foreground image, which is also composed of frame-variant contents, but with a frame-variant view. Each foreground image contains a static globe with the surrounding translucent ozone surface. For each day, the ozone data are transformed (irregularized to remove regions of missing data and warped onto a sphere), realized (color and opacity mapped and surface deformed), and full-color rendered (about 45,000 to 50,000 translucent quads for the ozone; 259,200 full-color-mapped opaque quads on a sphere with normals for the globe). The foreground and background images are blended and combined to compose each final frame. Each frame at workstation resolution (about 1.2 million pixels) using DX on a 32-way (40 MHz) PVS required about 12 seconds of computing time. Hence, the entire animation took about 15 hours at that resolution.

Distribution

Current efforts on data distribution have focused on the application of compression techniques to three sample animation sequences on a 32-way (40 MHz) PVS equipped with a RAID-3 HiPPI disk array capable of 50 MB/sec sustained access speeds. The first example is a high-resolution sequence of 2040x1536 32-bit (8-bits of red, green, blue, alpha) images, 88 frames in length totalling about 1052 MB. An 8:1 block-truncation lossy com-
pression (i.e., each 32-bit pixel is represented by 4 bits) required about 41 seconds, resulting in a rate of approximately 26 MB/second or 2.15 Hz disk to disk. Lempel-Ziv lossless compression was applied to the entire sequence as a whole, not on a frame-by-frame basis. As expected the results took considerably longer, requiring about 2 minutes, 2 seconds, yielding a rate of approximately 8.7 MB/second disk-to-disk to achieve 45.4% compression. In both cases, the compression algorithms were parallelized on the PVS.

The second example is a 151-frame sequence of 640x480 32-bit images of about six months worth of animation similar to that illustrated in figure 6. Results with this considerably smaller collection (about 177 MB) are quite similar, pointing to the potential scalability of shared-memory, symmetric multiprocessor systems like a PVS to this problem. For the 8:1 lossy compression, about eight seconds were required, yielding a rate of approximately 23 MB/second or 18.9 Hz disk to disk. The lossless compression of the entire sequence required about 22 seconds to achieve 62.6% compression at 8.0 MB/second disk-to-disk.

The third example is 5853 frames of a digital video (D1) sequence (Treinish, 1993a). Most of the sequence consists of frames similar to Figure 6 -- one for each day from January 1, 1979 through December 31, 1991. Each D1 frame is composed of 10-bits each of YUV (a chrominance and intensity-based specification of color) at 720 x 487 resolution for playback at 30 Hz. Hence, this 3 minute, 15 second sequence is 10.6 GB in size, which is maintained as a single file on a PVS. A PVS-based (parallelized software) MPEG compression facility was used to create an approximately 225:1, lossy-compressed MPEG-1 sequence. About 12 minutes, 5 seconds were required for this operation, yielding a rate of 14.7 MB/second or 8 Hz, disk-to-disk.

Interactivity

Figure 7 is a snapshot of a DX Motif-based interface for a prototype interactive browsing system. It provides very simple modes of interaction: selection of space and time (i.e., geographic regions or seasons of interest) for browsing via the spherically warped presentation shown in figure 6. There are dial widgets for the specification of the geographic viewing centroid of the global "object", and slider widgets for selection of the year to examine. Other Motif widgets, most of which are not visible in figure 7, provide choices of other visualization techniques, use of cartographic presentations, selection of analysis techniques, etc. A VCR-like widget provides control over the choice of the portion of the year to browse by Julian day. Optionally, a zonal slice of the data being browsed at the specified longitude may be shown as a pseudo-colored line plot of the latitudinal distribution. The data illustrated in figure 7 are the aforementioned global column ozone data derived from the same 14-year archive as shown in figures 2, 5 and 6.

The prototype browsing system is built as a client-server, consistent with the architecture of DX, as shown in figure 8. A PVS functions as the browse server in this implementation. High-speed display of browsing visualizations is local to the PVS. Remote display is via standard XWindow services with update rates limited to what the network infrastructure can provide. This prototype also operates at lower performance (i.e., less interactive) with a Unix workstation as a browse server.

Conclusions

A study of qualitative methods of presenting data shows that visualization provides a mechanism for browsing independent of the source of data and is an effective alternative to traditional image-based browsing of image data. To be generally applicable, however, such visualization methods must be based upon an underlying data model with support for a broad class of data types and structures.

Interactive, near-real-time browsing for data sets of interesting size today requires a browse server of considerable power. A symmetric multi-processor with very high internal and external bandwidth demonstrates the
feasibility of this concept. Although this technology is likely to be available on the desktop within a few years, the increase in the size and complexity of archived data will continue to exceed the capacity of "workstation" systems. Hence, a higher class of performance, especially in bandwidth, will generally be required for on-demand browsing.

A few experiments with various digital compression techniques indicate that a MPEG-1 implementation within the context of a high-performance (i.e., parallelized) browse server is a practical method of converting a browse product to a form suitable for network or CD-ROM distribution.

Future Work

From this initial prototype implementation of an interactive data browser, there are several areas for future work. Since practical low-cost decompression of JPEG-compressed images is becoming available on the desktop, experimentation with JPEG is warranted (Pennebaker and Mitchell, 1993). As with JPEG, the MPEG-1 motion video compression technique is becoming available for multimedia applications of video sequences on the desktop, whether the animation is distributed via network or via CD-ROM. Additional testing with MPEG-1 and the higher quality, MPEG-2 as it becomes available is required within the browse server as well as on various desktop playback systems.

The second area of research would focus on fleshing out more of the interactive archive architecture schematically illustrated in Figures 1, 3, 4 and 8. Specifically, the prototype interface and visualization could migrate to a data-driven one conceptually similar to the primitive implementation discussed by Treinish (1989), which could be integrated with metadata and data servers to achieve a browsing archive system (i.e., with data management services at the aforementioned levels I, II and III). This would also imply the availability of integrated data and information services similar to those in the rudimentary system described by Treinish and Ray (1985). Such an approach would be further enhanced by the integration of the prototype browsing system with tools for data analysis, which are already available.

The strategies for qualitative visualization have focused on only a few methods for spherically-oriented data with large spatial extent. Clearly, investigation of alternative approaches of highlighting features in such data are required, for which there are a number of potential issues (Rogowitz and Treinish, 1993b). In addition, the extension of this browsing architecture to other classes of data is also warranted.

Acknowledgements

All of the data sets discussed above were provided courtesy of the National Space Science Data Center, NASA/Goddard Space Flight Center, Greenbelt, MD.

References


Figure Captions


3-6. Radially deformed pseudo-color and opacity-mapped spherically warped surface of global column ozone on September 30, 1992 with annotation.

Total Column Ozone (Dobson Units) — September 30, 1992
Total Column Ozone

Mean = 282 DU
Min. = 127 DU
Max. = 471 DU
View = 440 DU
S.D. = 59 DU

1992
Day 274

Dobson Units (DU)
Daily Stratospheric Ozone (Dobson Units): 1992, Day 274

Ozone at Viewing Centroid = 355 Dobson Units
4. A Rule-based Tool for Assisting Colormap Selection

Lawrence D. Bergman  Bernice F. Rogowitz,  Lloyd A. Treinish
bergman@watson.ibm.com  rogowitz@watson.ibm.com  lloydt@watson.ibm.com
IBM Thomas J. Watson Research Center
Yorktown Heights, NY

Introduction

Visualization is a process of mapping data onto visual dimensions to create a visual representation. A successful visualization provides a representation which allows the user to gain insight into the structure of the data, or to communicate aspects of this structure effectively (Bertin, 1967; Tufte, 1983; Cleveland, 1991). Even with modern visualization systems, which give the user considerable interactive control over the mapping process, it can be difficult to produce an effective visualization. One strategy for improving this situation is to guide the user in the selection of visualization parameters. In our previous work, we have described an interactive rule-based architecture for incorporating such guidance, and have described certain perceptual and cognitive rules which may be relevant (Rogowitz and Treinish, 1993a; Rogowitz and Treinish, 1993b, Rogowitz and Rabenhorst, 1994).

In this paper, we focus on improving the user’s selection of colormaps. To do so, we have built a library of colormaps, and a set of perceptual rules for selecting appropriate maps based on the structure of the data and the goal of the visualization. We have encapsulated this rule-based colormap selection process as a tool, PRAVDAColor, in the IBM Visualization Data Explorer software package, and demonstrate how this module can be incorporated into visualization applications involving the mapping of color onto two- and three-dimensional surfaces. This implementation demonstrates the viability of the technique, and provides a testbed for evaluating the rules.

Interactive Rule-Based Architecture

We have previously presented a rule-based architecture called PRAVDA (Perceptual Rule-Based Architecture for Visualizing Data Accurately) for assisting a user in making choices of visualization parameters (Rogowitz and Treinish, 1993a; 1993b). This architecture provides sets of appropriate choices for visualization based on a set of underlying rules which are used to constrain operations (e.g., selecting a colormap, selecting iso-contour line color). Rules incorporate information about the data, which we call metadata, such as minimum, maximum, or spatial frequency, and also information supplied by the user.

The architecture also provides for linkages between rules that control different visualization operations, with a choice of parameters for one operation constraining choices that are available for others. For example, if the user selects a colormap, that information is fed back to the operation for selecting contour lines, where rules constrain the parameters of the contour lines depending on which colormap has been selected. Hence, if the contour lines are superimposed over a dark region, as defined by the colormap, legibility rules would constrain the set of color choices to those offering sufficient luminance contrast to be detectable (Carswell and Wickens, 1990). This network of linked, intelligent operations help guide the user through the complex design space of visualization operations.

In our previous work, we have described the general principles for implementing such an assemblage of rule-based visualization operations. In this paper, we describe a full implementation of one of these operations, colormap selection. In PRAVDAColor, perceptual rules constrain the set of colormaps offered to the user based on system-provided metadata (data type, data range), metadata computed by algorithm (spatial frequency) and metadata provided by the user (the visualization task). This is in contrast to previous rule-based systems for
visualization which do not explicitly support user tasks nor color perception in the guidance they offer (e.g., Senay and Ignatius, 1994).

**Rule-Based Colormap Selection - Limitations of Current Technology**

Perhaps the most common operation in visualization mapping the values of a variable onto a color scale. Despite the importance of this operation, the creation and selection of colormaps is often not adequately supported in modern visualization systems, which typically offer the user a default colormap and a tool for creating custom colormaps. More importantly, however, these systems do not guide the user in selecting which colormaps will help the user understand the structure of the data, segment the data meaningfully, or highlight important characteristics of the data.

The most common default colormap, the "rainbow" colormap, is a hue-based scale from blue, through a rainbow of colors, to red. When this scale is mapped onto scalar data, the user is conceptually mapping a linear scale in hue onto a scalar variable. Perceptually, however, this scale does not appear linear. Equal steps in the scale do not correspond to equal steps in color, but look instead like fuzzy bands of color varying in hue, brightness and saturation. When mapped onto scalar data, this colormap readily gives the user the erroneous impression that the data are organized into discrete regions, each represented by one of the rainbow colors. This can lead the user to infer structure which is not present in the data and to miss details that lie completely within a single color region (Rogowitz, Ling, and Kellogg, 1992; Rogowitz and Treinish, 1993a).

Some visualization systems provide the user with tools for creating alternatives to the default colormap. These custom colormap facilities allow the user to construct almost any conceivable colormap, but tend to be difficult to use for creating complex colormaps. This inadequacy has led us to develop our own colormap generation tool, described below. Even when an adequate tool is provided, moreover, deciding what colormap to create is essentially an unbounded problem. Some systems, such as Interactive Data Language, address this problem by providing a library of colormap lookup tables (RSI, 1993). They do not, however, provide guidance about their application or alert the user to constructs that may introduce visual artifacts. In this paper, we improve on this approach by offering the user colormaps guided by principles of human perception. In this way, **PRAVDA-color** can offer the user a richer set of choices than a single default colormap, while ensuring that each choice is appropriate to the user's visualization task and the data being represented.

**A Taxonomy for Colormap Selection**

Table 4-1 shows our working taxonomy for the generation of colormaps. This table both incorporates recommendations currently in the literature, and provides new contributions, especially in the incorporation of spatial frequency.

Following other researchers (Della Ventura and Schettini, 1993; Carswell and Wickens, 1990; Ware, 1988; Robertson, 1988; Robertson 1994; Lefkowitz and Herman, 1992), we find the data types described in measurement theory to be very powerful in characterizing data representation strategies. The data type is shown in the left-most column. Currently, we support interval and ratio data, but plan to extend this approach to include ordinal and nominal data types. For each data type, we distinguish between low and high spatial frequency data, depicted in the second column.

The next three columns contain recommendations on creating colormaps for these eight data types which are appropriate for three different representation tasks. Following our earlier work (Rogowitz, Ling, and Kellogg, 1992; Rogowitz and Treinish, 1993a), an isomorphic task is one where the goal of the representation is to faithfully reflect the structure in the data. In a segmentation task, the goal is to divide the data into perceptually distinct categories. In a highlighting task, the goal is to call attention to particular features in the data.

The notes within each cell show our understanding so far of which types of colormap fulfill the requirements for each type of data for each task, broken down by spatial frequency. The following section describes the psychophysical rationale underlying these selections.
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Spatial Frequency</th>
<th>Representation Task</th>
<th>Highlighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio (true zero)</td>
<td>Low</td>
<td><strong>Luminance</strong>: uniform</td>
<td>Even number of segments</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: opponent or complementary pairs</td>
<td>Many segments OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: monotonically increasing from gray</td>
<td>Larger range for highlighted features</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td><strong>Luminance</strong>: monotonically increasing</td>
<td>Even number of segments</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: opponent or complementary pairs</td>
<td>Fewer segments</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: monotonically increasing from gray</td>
<td>Smaller range for highlighted features</td>
</tr>
<tr>
<td>Interval</td>
<td>Low</td>
<td><strong>Luminance</strong>: uniform</td>
<td>Many segments OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: opponent pairs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: monotonically increasing from gray</td>
<td>Larger range for highlighted features</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td><strong>Luminance</strong>: monotonically increasing</td>
<td>Fewer segments</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: uniform or small hue variation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: monotonically decreasing</td>
<td>Smaller range for highlighted features</td>
</tr>
<tr>
<td>Ordinal</td>
<td>Low</td>
<td><strong>Luminance</strong>: uniform</td>
<td>Fewer segments</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: variation around hue circle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: monotonically decreasing</td>
<td>Increase luminance of highlighted area</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td><strong>Luminance</strong>: monotonically increasing</td>
<td>More segments</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: variation around hue circle</td>
<td>Increase saturation of highlighted area</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: uniform</td>
<td></td>
</tr>
<tr>
<td>Nominal</td>
<td>Low</td>
<td><strong>Luminance</strong>: uniform</td>
<td>Fewer segments than 7 ± 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: variation around hue circle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: uniform</td>
<td>Increase luminance or saturation of highlighted area</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td><strong>Luminance</strong>: uniform</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Hue</strong>: variation around hue circle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Saturation</strong>: uniform</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-1. A Taxonomy of Colormaps based on Data Type, Representation Task, and Principles of Perception
Isomorphic Representation of Interval and Ratio Data

In order to accurately represent continuous data, the visual dimension chosen must appear continuous to the user. In an MRI image, for example, the degree of magnetic resonance, a continuous variable, is represented as a gray-level because continuous variations in gray-scale appear continuous to the user. That is, if the resonance increases monotonically over a spatial region, the brightness of the image will appear to increase monotonically over that spatial region. An inappropriate colormap, however, could create a visual representation which does not look monotonically increasing, for example, the default colormap described above.
Candidate colormaps which preserve the monotonic relationship between data values and perceived magnitude can be drawn from experiments done by Stevens (1951). Stevens identified a set of sensory dimensions (visual, auditory, and tactile) for which a monotonic increase in stimulus intensity produced a monotonic increase in perceived magnitude. In particular, he found the shape of this relationship to be a power law, with each sensory dimension characterized by the exponent of this power law. Perceived brightness obeys a power relationship with physical intensity (gray-scale) over a very large range of gray scales, making it a very good candidate for representing ratio or interval data. Another good candidate includes the color attribute saturation.

The Importance of Spatial-Frequency for Ratio and Interval Data

Ensuring that continuous variables are mapped onto perceptually continuous dimensions, however, will only give a faithful representation of the structure of the data if the spatial characteristics of the representation are taken into account. Human sensitivity to spatial variation is sketched in figure 4-1. The two curves show that our ability to resolve spatial variations differs for the hue and luminance mechanism in human vision. The luminance mechanism is tuned to higher spatial frequencies (that is, high resolution, finely detailed, or small-grained features). Colormaps which include a luminance component, therefore, are can adequately represent high spatial frequency information. The hue mechanism is tuned to lower spatial frequencies. Thus, saturation-based colormaps, which display variations in the magnitude of a hue, would be inadequate for conveying high spatial frequency information, but well-suited for representing larger-scale spatial variations.

4-1. Sensitivity to Spatial Variations of Hue and Luminance.

To ensure appropriate choices, the colormap rules in PRAVDAColor use metadata about spatial frequency to constrain the set of selectable colormaps. Metadata about spatial frequency are not generally included in data sets, so in PRAVDAColor it is computed. If the metadata indicate that the data, when realized on the display screen, would produce high spatial-frequency variations, then the rules offer the user colormaps which contain a monotonically-varying luminance component. These would include simple gray-scale colormaps and also col-
ormaps which also vary in hue (e.g., maps from dark green to light green, or maps from dark cyan to light lavender).

If the data contain predominantly low spatial frequencies, information about their spatial variation is more effectively carried by the hue channel, and the rules would offer the user colormaps with variations in saturation within a single hue or variations in saturation across two hues, such as the opponent pairs (Hurvich and Jameson, 1995) blue/yellow (saturated blue, through isoluminant gray, to saturated yellow) and green/red (saturated green, through isoluminant gray, to saturated red).

Data containing both fine spatial detail and gradual variations across space could be mapped to colormaps which combine these characteristics, such as dark saturated green to light saturated red.

PRAVDAColor incorporates these psychophysical relationships into its rule set. To create a faithful (isomorphic) representation of ratio and interval data, PRAVDAColor recommends colormaps which vary in luminance and saturation for high and low spatial frequency data respectively. A special distinction is made for ratio data where it is important for the representation to preserve the existence of a zero point. PRAVDAColor treats this as two separate scales, one above and one below the zero, and uses hue to distinguish them.

Colormaps for Segmentation Tasks

The rules for providing isomorphic colormaps for ratio and interval data are also effective in creating maps for segmented data. The luminance component conveys monotonicity in the data for high spatial frequency data, while the saturation component can be used to convey monotonicity in low spatial-frequency data. Since the steps are explicitly defined, however, luminance steps can also be effectively used for low spatial-frequency data. In creating a segmented colormap, it is important that the segments are each discriminably different from one another, which limits the number of steps which can be represented. We have found that more steps can be effectively discriminated for low spatial-frequency data than for high. For ratio data, where the zero is a semantically-important attribute, PRAVDAColor only offers segmented maps with an even number of steps (a transition occurs at the zero level).

Colormaps for Highlighting Tasks

Rules for selecting colormaps which highlight particular features in the data can be drawn from the literature on attention (e.g., Treisman and Gelade, 1980). Based on these rules, PRAVDAColor offers colormaps which allow the user to identify ranges of data to highlight perceptually. We currently provide colormaps which identify the mid-range data value, or set a threshold and highlight data values which exceed it.

In highlighting data, it is important that the dimension selected does interfere with a dimension already selected. For example, in our schema, nominal data are distinguished by hue. Increasing the luminance of the item to be highlighted draws attention to it without changing its hue, and therefore, without losing perceptual information about its semantic class.

The PRAVDAColor Interface

The PRAVDAColor user is presented with a set of colormaps appropriate to the data set and specified representation task. These colormaps are presented in a panel which pops up on the display when PRAVDAColor is executed. An example of this pop-up panel can be seen in the lower left corner of Figure 4-2. Each colormap is displayed as a colorbar and has an associated button, used for selecting that colormap. PRAVDAColor automatically selects one of the colormap set as an initial output, which is used to color the data. The user is then free to select a sequence of colormaps by pressing buttons, and can immediately see each selection applied to the data.

In addition to allowing interactive selection, PRAVDAColor allows the user some control over how the colormap is to be mapped onto the data set. Range sliders allow the minimum and maximum values in the data to be assigned to particular color values. Additionally, a mid-range slider specifies the position in the colormap
that is to be associated with the middle ((max-min)/2) value in the data. These sliders are particularly useful for changing the minimum and maximum levels assigned to colormaps which highlight extrema.

Implementation

**The PRAVDAColor Module**

PRAVDAColor has been implemented as a module for the IBM Visualization Data Explorer (Data Explorer) software package. Data Explorer is a general-purpose visualization system that supports visual programming through the construction of data-flow networks (Abram and Treinish, 1995). Data Explorer facilitated the development of PRAVDAColor's capabilities by providing a unified, extensible data model and polymorphic modules. Thus, rules were readily incorporated without the complication of accounting for multiple data types, and metadata (either calculated or provided by a user) can readily be added to data objects.

Figure 4-2 shows a sample Data Explorer data-flow network and the visualization that it produces. The PRAVDAColor module accepts as input the data set to be visualized (read by the import module in this example) and a user representation task index (created by the interactive Selector module).

The PRAVDAColor tool consists of two main portions. The first is a macro that selects the set of colormaps to be presented to the user based on characteristics of the data and a user-specified representation task. This macro, called ColorMapLookup, examines two characteristics of the data — spatial frequency, and presence or absence of a zero-crossing.

The spatial frequency analysis is performed in a simple fashion using existing Data Explorer modules. Input data is required to be topologically regular (a structured grid). The resolution of the grid is reduced (using a reduction factor derived from user input) and then interpolated back to the original resolution. This has the effect of low-pass filtering the data. The original data values are subtracted from the filtered values and then the standard deviation of the difference values computed and divided by the range of the original data values. This normalized standard deviation is used to determine whether the data contain predominately low-frequency or high frequency information. When the normalized standard deviation is less than or equal to 0.1, we consider the data to be low frequency, based on empirical results.

Using the computed values and a representation task, ColorMapLookup computes an index into a lookup table structured as shown in Table 4-1, which is used to determine a set of colormaps to be read into memory. If the data contain a zero crossing (i.e., the minimum and maximum values have different signs), the data are treated as ratio data. Otherwise, data are treated as interval data. The current implementation does not handle nominal or ordinal data.

The second major component of PRAVDAColor is the interactive colormap display/selector. This component is implemented as an outboard module in Data Explorer called ColorMapPicker, written in C using Motif/Xlib. Outboard modules are separately compiled and linked executables which are invoked by Data Explorer and communicate with the Data Explorer executive (main control component) via socket. As described above, ColorMapPicker allows the user to interactively select sequences of colormaps, as well as dynamically alter the assignment of colormap values to data values.

**Colormap construction**

To construct colormaps, we developed a tool similar to the interactive tool described by Rheingans (1990). Figure 4-3 shows the 3-dimensional colormap tool, an image that has been created using its output, and a view of its output in the Data Explorer Colormap Editor.

The colormap tool displays a hue-lightness-saturation (HLS) double-cone populated by color swatches. We chose to represent each swatch by a single square polygon rather than a solid to avoid the color artifacts created by a shading model.
The user can rotate and zoom the color space representation using the mouse. To ensure interactive updates, we draw only an outline of the space when the mouse is moving, filling in the color swatches when the mouse stops. By selecting color swatches with the mouse, a color path is constructed interactively. Figure 4-4 shows a path created with the tool.

The tool provides an 'undo' and 'move' function for editing the path. At any time, the user can send the current colormap to the rest of the Data Explorer network by clicking on a virtual button. The output colormap linearly interpolates between the selected swatches in HLS space. Similarly, the user may specify that a discrete colormap is to be produced containing equally sized segments for each selected swatch with no interpolation. This is useful in creating colormaps for segmented representation tasks.

The colormap tool was implemented in C using Motif/Xlib for interface elements and event handling, and OpenGL to create the three-dimensional graphics. The module converts all colors from HLS, used internally, to hue-saturation-value (HSV), required by Data Explorer.

Results -- Applications of PRAVDAColor

PRAVDAColor has been utilized internally as part of an informal testing effort as a prelude to making these tools available for evaluation by interested users of Data Explorer. PRAVDAColor has been integrated into various operational visualization activities.

Preserving the spatial structure of data

Consider figure 4-5, which is a screen dump of a typical Data Explorer program applied to two-dimensional data, where the default hue-based colormap from AutoColor is utilized to create an image of fluid density from a simulation of the noise produced by a jet aircraft engine.

The representation is dominated by the segmentation inherent in the default rainbow type of colormap. In order to study the spatial structure of these data as a continuum, AutoColor was replaced by PRAVDAColor to help in the selection of an isomorphic colormap. The results are shown in figure 4-2. It shows more of the fine spatial structure and turbulence inherent in these continuous data.

Different colormaps for different tasks

Figures 6 and 7 illustrate the importance of task specification for colormap selection. They show the result of a photochemical grid model of transport and deposition of airborne pollutants over the midwestern portion of the United States on June 26, 1987 at 18:00 local time. Ozone pollution concentration is shown in parts per billion by volume (ppbv). An isomorphic colormap was employed in figure 4-6. It effectively captures the inherent dynamics of the model by showing a snapshot of atmospheric motion (e.g., roughly circular filaments in yellow corresponding to higher ozone concentrations). On the other hand, the ability to indicate regions of moderate to high pollution is a different task. PRAVDAColor was used to select a segmented colormap in figure 4-7 for color-filled contours. In this case, higher pollution levels (e.g., above 160 ppbv) are clearly visible as yellow and red over Lake Michigan, to the east of Chicago. It should also be noted that this colormap allows the user to see some artifacts of the limited grid resolution of the model.

Color interaction

The analysis of concurrent color use by multiple objects is an additional criterion in colormap selection in order to avoid undesirable artifacts of color mixing. While the current implementation does not support dynamic interaction between visualization operations, PRAVDAColor does allow the user to address some of these issues explicitly. Several efforts to show multiple parameters using different colormaps were attempted. In these cases, the choices offered by PRAVDAColor were manually inspected for potential conflicts. The user can then choose colormaps that have minimal overlap in use of specific colors.

Figure 4-8, for example, is a display of global temperature and precipitation observed from weather stations. An isomorphic colormap is chosen for precipitation as a continuous pseudo-colored field, while contours of tem-

II-58
perature every five °C are colored using a segmented colormap composed of entirely different colors. This representation illustrates correlation between lack of precipitation with very high temperatures and high precipitation with moderately-high temperatures.

In figure 4-9, two distinct isomorphic colormaps are used for temperature and for wind speed from an empirical ocean/atmosphere circulation model. Sea level pressure is also shown as a deformed surface, which is pseudo-colored by surface temperature. The winds are shown as streamlines, pseudo-colored by wind speed. A region of ocean warming in the southeastern Pacific Ocean (pinkish-white) is associated with a decrease in sea level pressure and winds moving toward the South American coast.

In both figures 8 and 9, the relationships between different parameters would be difficult to see without the use of complementary colormaps. Previous efforts along these lines required the custom construction of independent colormaps with the Data Explorer Colormap Editor and applying them in rendered images — a time-consuming process. PRAVDAColor reduced the number of iterations by only offering a small and manageable number of choices that had the additional benefit of being appropriate for the visualization task at hand.

Three-dimensional Data

The application of PRAVDAColor is not confined to two-dimensional data. Figure 4-10 shows one time step of a regional, three-dimensional weather model shown as an isosurface of wind speed at 20 m/sec. The surface is colored by the temperature values in the computed volume interpolated on the isosurface. PRAVDAColor is used to create an isomorphic colormap corresponding to the full temperature volume.

Integrating PRAVDA into real applications

PRAVDAColor can easily be incorporated into actual applications built with Data Explorer. Figure 4-11 is a screen dump of a generalized application that provides cartographic representations from a selection of available parameters stored in a user-defined file. The package was extended by replacing the standard Colormap Editor with PRAVDAColor. This enhanced application allows a user to select a colormapping task, for which PRAVDAColor then offers a set of choices. The image shows a segmented colormap applied to filled contours of total column ozone displayed as orthographic maps for the northern and southern hemispheres. The seasonal ozone depletion is visible as a black region over the south pole for these data taken on October 1, 1991. The available choices are illustrated in the ColorMapPicker panel.

Discussion

PRAVDAColor has been used effectively within IBM. For complex colormap selection activities, this tool has reduced the length of time required to develop satisfactory results. This has been sufficiently encouraging to warrant further enhancements, and an effort to make the tools available to current Data Explorer users interested in evaluating them and providing feedback for their improvement.

Importance of Spatial Frequency in Colormap Selection

Previous guidance for colormap selection has established the importance of various classes of colormaps appropriate for specific visualization tasks (e.g., Lefkowitz and Herman, 1992; Ware, 1988). However, there can be considerable variation in the collection of colormaps that effectively satisfy a given task from one data set to the next. The analysis of the spatial frequency coupled with a large library of colormaps offers a unique method for characterizing the appropriate representation of the data. The user is shielded from the intricacies of psychophysical research and color theory, and is automatically offered colormaps with significant luminance variations for representing fine spatial structure. Similarly, the user is offered colormaps with significant variation in saturation for representing coarse spatial structure.

Because of its obvious simplicity, the current algorithm for estimating spatial frequency might be criticized as being naive. However, it does have a number of virtues. The typical approach to such a problem would involve applying a (fast) Fourier Transform (FFT) to the data of interest. An estimate of the spatial frequency for use herein could be specified in the following manner for two-dimensional data:
\[ F(u,v) = |G(u,v) M_i(u,v)| \]

where: \( G(u,v) \) is the Fourier transform of the data represented by \( G(x,y) \).

\( M_i(u,v) \) is the frequency mask or filter for the \( i \)th class of spatial variation.

In practice, several \( M_i(u,v) \) functions would need to be defined, one for each class of spatial frequency for which a separate collection of colormaps would be made available. Generic versions of filters may be non-trivial to construct. Clearly, this approach could be extended to three-dimensional data. However, the development of three-dimensional filters would be even more difficult. Of course, a more complex representation of the spatial frequency than this might be required to be effective (e.g., introduction of weighting functions in the frequency domain). As with the approach of filtering in the spatial domain, metrics for evaluating the results to assign spatial frequency classes would need to be developed.

Unfortunately, an FFT-based method of frequency-domain filtering is generally limited to regularly gridded data (e.g., an image). Hence, this approach would not directly apply to curvilinear, partially regular or irregular data. In addition, it is not sensitive to disparate scaling that might apply to each dimension of a grid. Therefore, an alternative was required to support rule-based colormapping of more general classes of data. The spatial-domain filtering approach described earlier has the advantage of operating on any topologically regular gridded or structured data set and is affected by dimensional scaling. To date, this method has not been extended to topologically irregular or unstructured grids. Finally, the approach would appear to be computationally quicker than using FFTs, especially for three-dimensional data, which has enabled the application of PRAVDAColor to be interactive.

Integration of Intelligence into Existing Systems

In our earlier work, we established the importance of interactive rule-based tools to extend the utility of modern visualization software systems. The approach capitalizes on the breadth of their capabilities while minimizing the number of iterations required to create more appropriate and better visualizations. As a first step in this effort, the implementation of PRAVDAColor has demonstrated that this is feasible and practical. Although Data Explorer required some enhancements to improve the interaction between the custom tools that were developed, the basic implementation was relatively straightforward. These enhancements will ease the creation of new PRAVDA tools, and they also have advantage for other applications that may need to interface cooperatively with Data Explorer. We expect other visualization systems to be extensible to incorporate PRAVDAColor-like tools through the addition of custom code as well as existing visual programming modules (e.g., Rasure and Wallace, 1991; SGI, 1991; Upson et al., 1989).

Extensions

In this paper, we have presented a Data Explorer module for guiding the user in selecting colormaps. The next step would be to add additional rule-based operations and provide feedback between the operations.

The PRAVDA architecture can be readily extended to allow the user to interactively modify the metadata and the rules. For example, the user could choose to modify the algorithm for computing spatial frequency, or decide to see what mapping choices were available if the goal of the visualization were changed. Likewise, the rules could be changed to reflect new insights or information. For example, if the user realized that the structure being sought was confined to a certain range of values in the data set, the rule could be modified so that data in this range would be highlighted by mapping it to an appropriate visual dimension. Our first focus has been on implementing rules based on perceptual principles, but could be easily extended to include other types of rules, for example, rules about mappings or conventions particular to certain domains (Regowitz and Treirish, 1994).

Conclusions

A framework for rule-based guidance can improve the effectiveness of systems by assisting the user in making two types of appropriate representation choices. One concerns domain-independent factors, such as ensuring
that data content is reflected in images and that perceptual artifacts are not erroneously interpreted as data features. A second type concerns task-dependent factors. For example, different advice on representation is required depending on whether the goal of visualization is exploration or presentation.

The PRAVDA architecture explicitly incorporates guidance based on principles of human perception, cognition and color theory (Rogowitz and Treinish, 1995b). These principles are incorporated in rules which the user can select during the visualization process. Depending on the higher-level characteristics of the data, the rule constrains the way in which the data are mapped onto visual dimensions. This architecture has been utilized to build a tool which eases the burden of creating colormaps for many visualization applications.

The PRAVDAColor module has been implemented as an initial proof-of-concept of the PRAVDA rule-based architecture for advising a visualization developer. We have developed a set of rules that simplify the task of colormap selection. The user is presented a set of choices that are appropriate, based on characteristics of the data being visualized and the desired form of the representation.

Using this rule-based colormap advisor, we have constructed a set of visualizations that demonstrate the advantages of colormaps that conform to perceptual principles over an uninformed choice. We anticipate users finding the current implementation helpful in constructing visualizations, and the utility increasing as we add additional rules and visualization operations to the system.

Acknowledgments

We thank the Data Explorer development team for their cooperation in this effort and valuable discussion that led to the successful implementation of PRAVDAColor.

Jet engine noise data are available courtesy of Combustion Research and Flow Technology.

Surface station temperature and precipitation data, oceanographic temperature, pressure and wind data, magnetic field data and total column ozone data are available courtesy of NASA/Goddard Space Flight Center.

Three-dimensional weather model data are available courtesy of Glenn Wightwick, IBM Australia.

Photochemical model data are available courtesy of the United States Environmental Protection Agency, Research Triangle Park, North Carolina.

References


Color Figure Captions

4-2. Data Explorer Visual Program Incorporating PRAVDAColor.

4-3. Construction of a Data Explorer Colormap with a Three-Dimensional Colormap Tool.

4-4. Specification of a Colormap in HLS Space.

4-5. Data Explorer Visual Program Using the Default Colormap Tool (AutoColor).

4-6. Photochemical Pollution Model with an Isomorphic Colormap.

4-7. Photochemical Pollution Model with a Segmented Colormap.

4-8. Global Temperature and Precipitation with Complementary Colormaps.

4-9. Ocean/Atmospheric Conditions Shown with Two Isomorphic Colormaps.

4-10. Data Explorer Visual Program Using PRAVDAColor with Three-Dimensional Data.

4-11. Data Explorer Application Incorporating PRAVDAColor.
US EPA Regional Oxidant Model -- Midwest Ozone (ppbv): June 26, 1987, 18:00
US EPA Regional Oxidant Model -- Midwest Ozone (ppbv): June 26, 1987, 18:00
Yearly Average of Global Weather Station Data for 1960

Precipitation (mm)

Temperature (Degrees C.)
Collaborative Computing and Integrated Decision Support Tools for Scientific Visualization

Theresa Marie Rhyne
Lockheed Martin
U.S. EPA Scientific Visualization Center
U.S. EPA Environmental Research Center
86 T. W. Alexander Drive
Research Triangle Park, North Carolina
trhyne@vislab.epa.gov

Introduction

This section of the ACM SIGGRAPH 96 course notes on Visualizing Scientific Data and Information explores the concepts of renaissance teams, collaborative computing and integrated decision support tools. There are four sections: 1) The Three Classes of Visualization Tasks; 2) Customizing Software for Analysis & Decision Making (A First Step in Developing Collaborative Computing Tools); 3) Multi-Variant Physical & Natural Sciences Visualization; 4) Collaborative Computing and the Three Stages of Metacomputing and 5) Looking on the Horizon - Integrated Decision Support Tools.

III-I The Three Classes of Visualization Tasks

In dealing with large scientific data sets, there are three classes of visualization tasks which are independent of data or technique. They include: a) analysis and exploration; b) decision support; and c) presentation. Each of these tasks usually involves collaborative efforts between research scientists, policy analysts, artists, programmers, and other expert staff.

III-I. a) Analysis and Exploration
The analysis and exploration tasks focus on examining physical and natural sciences data sets. These data sets can include remotely sensed or monitoring site observations as well as large scale computational output from supercomputers. For air quality and water quality modeling efforts, some data visualization tasks include the comparisons of emission inventories which are data inputs to a model with the resulting data output from executing the model. In exploring subsurface contamination, data associated with in-situ observations is frequently combined with the generation of a three-dimensional isosurface of the contaminated zone. Thus, visualization is used as an exploratory tool for examining data integrity and data validity issues.

In the physical and natural sciences, visualization has also been used as a technique for calibrating the computational algorithms which are components of large computer models. Here, interactive visualization tools are helpful for gaining insight into the impacts of modifying algorithms. Many of these issues have already been highlighted in the first and second sections of this course.
III-I. b) Decision Support

Visualization techniques also assist with the physical and natural sciences decision making process. At the U.S. Environmental Protection Agency (U.S. EPA), visualization is used by the Office of Air Quality Planning and Standards as a visual display tool in the process associated with developing air quality standards, policies and procedures. The U.S. EPA Great Lakes National Program Office has used visualization as an inquiry and decision support tool for water quality and ecosystem analyses.

These activities have required the customization of visualization software to support policy decision making efforts. This has included the creation of color legends and titling tools which are linked into the visual display. These tools are interactive and usable by policy analysts. Customized point and click user interfaces to visualization tools have also been built.

Figure # III - 2: Example of a Decision Support visualization of the sediment concentrations in Lake Erie resulting from a large storm.
Here the various components of the computational model output (wind velocity, sediment concentrations, erosion & deposition, and depth) are depicted as individual layers.
III-I. c) Presentation

There is also a need to develop visualizations and animation sequences which educate the general public and inform high level decision makers about physical and natural sciences concerns. These presentation visualizations often require the use of high end animation tools. The final product is often a polished production with voice over narration and background music sound tracks. Issues associated with producing these types of animations are highlighted in the fourth section of this course.

Figure # III - 3: Example of a Presentation visualization of the sediment concentrations in Lake Erie resulting from a large storm. Here arrows depict the wind direction. This image appears in the Federal High Performance & Communications: Toward a National Information Infrastructure (1994) publication. (Customized tube code for wind vector display written by Mark Bolstad.)

III-I. d) The Role of Renaissance Teams

The development and usage of tools which support these three classes of visualization usually involves collaborative efforts among
scientists, policy analysts, artists, programmers and other expert staff. This is often defined as a Renaissance Team.


At the U.S. EPA, the Renaissance Team approach has been applied to visualization toolkit development for collaborative computing. The composition of the team includes: a) environmental and computational scientists in an EPA research Lab; b) policy analysts and computational scientists in an EPA program office; c) computational model builders; and d) visualization specialists. The goal has been to build tools which scientists and policy analysts can use for the daily examination and visual display of physical and natural data. Current efforts include the transfer of this technology beyond the Federal government to State environmental protection agencies.

The wide usage of visualization tools will also allow for collaborative teams which support multi-disciplinary research activities in the physical and natural sciences. The next section of these course notes will highlight efforts to customize visualization software for exploring multi-variant data sets.
III-II. Customizing Software for Analysis & Decision Making 
(A First Step in Developing Collaborative Computing Tools)

Although standard visualization software can be effective in developing initial displays of physical and natural sciences data, some customization is usually required to support both the analysis and decision making process. Customizing software encompasses the development of user interfaces which can support collaborative computing and easy access to integrated decision support tools. Some of these issues are highlighted below.


III - II. a) Spatial Context

There are several factors which influence the visual representation of physical and natural sciences data. These include: type of data, relationships among different components of a data set, placement of data in a spatial and temporal context and interpretation of the data.

Frequently, earth sciences data is geographically registered. As a result, a map of the geographic domain is a helpful visual aid to provide spatial context for the data. Advanced principles of cartography can also be applied to develop more sophisticated projections for mapping coordinate systems. At the U.S. EPA, we are currently exploring methods to integrate our Geographic Information Systems set of tools with Scientific Visualization software in order to create a comprehensive software environment for the visual display of geographically registered environmental data sets.

Spatial context is also important in examining other types of physical and natural data sets. In the realm of computational chemistry, merging a molecular visualization with a traditional line drawing diagram of the molecule's structure establishes a base line for
decision making. In examining air flow in and around buildings, developing a three-dimensional display of the building is helpful. The level of detail depicted in the three-dimensional characteristics of the building depends on the granularity of the computational model of air flow. If the computational model is attempting to examine general air flow patterns around a building, a simple cubic representation may only be required. However, if the computational model is examining particle tracing associated with air flow inside the building, a very detailed architectural rendering of the interior of the building might be desired. The challenge for the detailed architectural rendering approach might involve merging Computer Aided Design (CAD) systems with Scientific Visualization tools.

III - II. b) Simple Visual Cues

In air quality and water quality visualizations where concentration levels of pollutants and times of exposure are critical, visual cues which describe these changing activities are important. Color bars and legends are helpful for these purposes. At the U.S. EPA, we have often customized visualization software to support environmental researchers' and policy analysts' needs to depict several emission scenarios for developing air and water quality guidelines. Here, the ability to support multiple color maps and discrete color mapping functions in a single visualization/animation sequence becomes important.

Complex air quality and water quality computational models often examine multiple pollutants for a given scenario. Thus, the data sets from these kinds of computational runs include multiple chemical species examined across multiple atmospheric or water layers for episodes lasting over 100 or more time steps for a given geographic domain. Visualization tools which support labeling and titling functions are helpful here. Time clocks and counters are also effective visual cues for these animation sequences.

To support the analysis and decision making process, we have often used discrete color maps (originally developed for printouts from
computer plotting devices). Cool hues and colors (e.g. dark purple and blue) indicate low concentrations while warmer tones (e.g. orange and red) denote higher values. In some circumstances, yellow has been used to indicate a midrange value in the data where air quality or water quality standards could be exceeded.

III.II. c) User Interface Design - Distributed Networks

In developing visualization tools for scientists and policy analysts, it cannot be expected that all or the majority of decision makers will learn how to visual program. As a result, customized and pre-established visual programming modules and networks need to be created which support the visual display of output from physical and natural science models and data sets. As mentioned in the previous section, data output from air quality and water quality models can consist of multiple chemical species examined across multiple air or water layers for a given episode having 100 or more time steps. Designing effective user interfaces that allow decision makers to visually examine these type of data sets is one of the challenges we are presently dealing with at the U.S. EPA. We have used the widget and button tools of visual programming environments (often encompassed in visualization toolkit packages) to build user interfaces which are linked to pre-established visualization networks.
Figure # III - 4: User Interface to the U.S. EPA High Performance Computing & Communications Program's photochemical modeling system. Here two pollutant scenarios and the resulting difference are visualized simultaneously. (Also see Case Study #1 of these notes)

EPA researchers frequently execute their computational models on the Cray C90 remotely located at the National Environmental Supercomputing Center (NESC) in Bay City, Michigan. The data frequently requires mass storage for retrieval at a later point in time after execution of the computational models. As a result, customized visualization modules and networks need to address distributed network processing and remote module execution. We have built visualization networks which combine modules from a heterogeneous group of compute engines, storage systems, and workstations. The environmental decision maker operates this distributed visualization network from a user interface located on a Unix workstation which supports the graphical display of the data.

III.II. d) The Need for Collaborative Computing
Once visualization networks are built and user interfaces designed, there is a need to provide physical and natural scientists with the capabilities to share visual information in real time. Often researchers are located in sites geographically remote from one another. Then, the real-time sharing of visual information requires the usage of high-speed networking. Thus begins the journey of collaborative computing.
III - III. Multi-Variant Physical & Natural Sciences Visualization

An important initiative of the U.S. High Performance Computing and Communications Program involves Grand Challenge research efforts that attempt to examine the multi-variant concerns of physical and natural sciences problems. For the environmental and earth sciences, this encompasses the merger of air, water and subsurface data sets into single visualization presentations. These multi-variant projects involve collaborative efforts among physical and natural scientists located at research sites across the United States and abroad. (See Figure III - 2 as an example multi-variant visualization)

Some of the system design issues associated with collaborative computing which supports the examination of multi-variant data types include: data format standards; data management; graphics-client software; and tracking & steering functions for collaborative efforts. Historically, air quality, water quality and subsurface computational models have been developed independently of each other. As a result, the data output format structures differ. Determining a common data output format is a part of the collaborative process. This includes determining the appropriate time step value for animation sequences. Some data sets might animate according to hourly time steps while others might change over daily or monthly time periods. Data storage, access and retrieval are important data management issues for effective exploration of multi-variant physical and natural sciences data. These item have been highlighted in earlier sections of this course.

There are many situations where simulation codes have already been executed but there continues to remain a need for collaborative analyses of the computational output. This analysis function provides for two or more scientists at remote locations to simultaneously view the computational output and pass control of the interactive analysis to each other to allow for question-answers, mutual clarification, or expert to novice advice on interpretation of the data in question. Data storage and retrieval mechanisms become important for these situations.
The Tecate Visualization System, developed at the San Diego Supercomputer Center, is a software environment that supports exploratory visualization of data collected from networked data systems. A simple World Wide Web interface accesses and stores earth sciences data into a database management system. This visualization management system is intended to extend beyond the typical database management environment by storing information on how to visualize the data with the data itself. For collaborative computing, this approach will allow scientists to return to their data sets with a "record" of previous visualization efforts. This record is helpful when exploring multi-variant data sets which have not previously been combined.

III - IV. Collaborative Computing and the Three Stages of Metacomputing

Collaborative computing involves facilitating information discovery and scientific visualization activities between researchers located at various remote sites. It includes the use of visualization and information retrieval in a high speed networked environment. Computing resources become transparently available to researchers via the networked environment and this results in a metacomputer. The metacomputer is a network of heterogeneous, computational resources linked by software in such a way that these linked resources can be used as easily as a personal computer. For any one research project, a scientist might use a desktop workstation, remote supercomputer, a mainframe supporting mass storage archive of data, and a specialized high performance graphics workstation.

The metacomputer concept is still evolving and software tools to support these activities are beginning to emerge. Three stages of metacomputing are outlined in the following discussion.

III - IV. a) Metacomputing Stage 1 - Building Access Tools

The first stage of effective collaborative computing is primarily a software and hardware integration effort. It involves connecting all of the metacomputing resources with high-speed networks, implementing a distributed file system, coordinating researchers' access across the various computational elements, and creating a seamless software access to the computing technology. Examples of software which support information discovery and its visual display in a metacomputing environment are NCSA's Mosaic and Netscape from Netscape Communications Corporation. These tools allow for browsing the World Wide Web (WWW). These browsing tools are hypertext windowing systems and are available for the X window system, Apple Macintosh and Microsoft Windows environments. Using point and click methods, researchers are "linked" to information resources across the Internet. With appropriate graphics hardware and memory, it is possible to access, display and run
animation files. With WWW browser tools, researchers in the physical and natural sciences are able to access and share information across heterogeneous computing platforms.

Figure III - 5: Example of a World Wide Web (WWW) page for the U.S. EPA Gulf of Mexico Program Office (http://www.epa.gov/docs/gumpo). This WWW application was developed for the U.S. EPA with support from Lockheed Martin.

Another important component of these collaborations involves usage of the Multicast Backbone (MBone) on the Internet. The MBone provides scientists access to video-conferencing type capabilities from their appropriately configured desktop workstations. Public domain tools that support live audio, video and whiteboard activities are currently available for these efforts. These multi-media tools allow scientists located at geographically remote sites to interact in real time and share visual information.

III - IV. b) Metacomputing Stage 2 - Computing in Concert

The second step of collaborative computing focuses on spreading components of a single research application across several computers. This permits a center's heterogeneous collection of computers to work in concert on a single problem. Software that supports collaborative visualization by researchers at remote sites is just now emerging. One example of a prototype interactive scientific data analysis and visualization system has been built at NASA's Jet Propulsion Laboratory (JPL). The Linked Windows Interactive Data System (LinkWinds) is designed to support the two and three dimensional visualization of multi-variable earth and space sciences data sets. LinkWinds supports networked collaborative data analysis. The graphical user interface (GUI) is X-Windows based.
while the computer graphics rendering functions rely on the Silicon Graphics, Inc. (SGI) OpenGL specification. LinkWinds is designed to handle direct access to a variety of data formats which allows for the merger and visual display of data sets from multiple computational sources and scientific disciplines. The networking functions of LinkWinds do not rely upon the X Window networking facilities. Instead, the implementation (based on MUSE) transmits only individual control values and button or menu selections. This reduces the sizable steam of commands which sometimes result under X Window networking facilities.

![Figure III - 7: Example session from the Linked Windows Interactive Data System (LinkWinds) developed at the NASA Jet Propulsion Laboratory. Image courtesy of Bud Jacobson. LinkWinds WWW site: (http://linkwinds.jpl.nasa.gov/)](http://linkwinds.jpl.nasa.gov/)

III - IV. c) Metacomputing Stage 3 - Surfing on the Infrastructure

The third stage of this process will be a transparent national network that will dramatically increase computational and information resources available to explore an physical and natural sciences research application. This stage is closely tied to the activities of the National Information Infrastructure.

The High Performance Computing and Communications Program (HPCC) in the United States is supporting research and development (R&D) in gigabit speed networks. This technology is designed to support researchers' requirements to continuously display on local workstations the output from model simulations running on remote high performance systems. These R&D efforts are examining satellite, broadcast, optical, and affordable local area networking designs. These networking technologies are intended to support the rapid movement of huge files of data, images and videos in a shared, collaborative computing environment which spans thousands of networks and encompasses millions of users.

Reference: High Performance Computing and Communications: Foundation for America's Information Future, (Supplement to the President's Fiscal Year 1996 Budget), A report by the Committee on Information and Communications, National Science and Technology Council, 1996,[http://www.hpcc.gov/blue96/index.html](http://www.hpcc.gov/blue96/index.html)

There are positive and negative technical and social impacts associated with surfing on this telecommunications infrastructure. Positive aspects associated with these high speed networked collaborations center on real time visualization and information discovery among geographically remote research or Renaissance Teams. There are also negative impacts or roadblocks associated with metacomputing. Network transmission difficulties and differences in desktop workstation architectures can cloud the actual visualization two collaborating researchers are simultaneously viewing and steering. Setting up and learning to use the metacomputing infrastructure can be all consuming and thus refocus
the basic education or scientific discovery process. These remain unresolved issues as we move into the realm of multimedia.

III. V. Looking on the Horizon -- Integrated Decision Support Tools

There are many unresolved computing challenges associated with scientific visualization. Here, we present two such issues for thought and consideration: a) the notion of GIS - VIS integration and b) the use of the WWW and intelligent agents for assisting scientific visualization efforts.

III.V. a) GIS - VIS Integration

Geographic Information System (GIS) and Visualization methodologies and techniques are used to examine some types of physical and natural sciences data. Interestingly, both of these disciplines developed and have often been implemented in parallel to each other. In many situations, Visualization is primarily associated with computational simulations and modeling efforts as well as data obtained from satellite and remote sensing systems. GIS environments have supported the collection, analysis, and display of large volumes of spatially referenced data pertaining to geographic, cultural, political, and statistical arenas.

Often the hardware configurations optimized to support GIS are not compatible with Visualization methodologies. Frequently, Visualization software is customized to encompass standard cartographic and spatial display capabilities of GIS environments. Here, we propose the integration of these two disciplines to enhance decision making. This will allow scientists and policy makers the capability to move between multiple information and visual displays of their data. The GIS capabilities will allow for researchers to query data points and obtain precise locations and attributes. The Visualization functions will support the creation of 3-dimensional surfaces and animations of multiple data sets.
Three levels of GIS and Visualization integration can be defined: rudimentary, operational and functional. The rudimentary approach uses the minimum amount of data sharing and exchange between these two technologies. The operational level attempts to provide consistency of the data while removing redundancies between the two technologies. The functional form attempts to provide transparent communication between these respective software environments. At this level, the user only needs to request information and the integrated system retrieves or generates the information depending upon the request.

Software links that attempt to integrate GIS and Visualization toolkits are currently being established. IBM Data Explorer (Visualization Toolkit) modules which link to Environmental Systems Research Institutes (ESRI)'s Arc-Info (GIS) software are under development and testing. Systems designers at Advanced Visual Systems have also built Application Visualization System (AVS - Visualization Toolkit) modules which support Arc-Info data types and formats. The US Army Construction Engineering Research Laboratory has expanded its public domain GIS system (GRASS) to support visualization capabilities, (http://www.cecer.army.mil/grass/viz/VIZ.html).

This type of integration will provide scientists and policy analysts with an improved set of information and visual display tools for data exploration and decision making as well as increasing the capabilities for collaborative computing.
Figure III - 6: Integrated AVS - Arc Info system. Research conducted at the US EPA Scientific Visualization Center, Thomas Fowler and Theresa-Marie Rhyne / Lockheed Martin Technical Services, 1996.


III.V.b) WWW & Intelligent Agent Assistance for Visualization

A significant limitation of existing hypermedia Internet tools, like Mosaic and Netscape, is the inability to rapidly find and quickly recall information resources of interest on the WWW. Infrequent (and general) users of distributed hypermedia systems can easily become overwhelmed by the large number of links to information resources and disoriented while navigating between the various remote file servers. One potential solution to this dilemma is the incorporation of intelligent or remote agent capabilities into browser programs.


An agent is an automated program that examines the Internet on its operator's behalf searching for specified information. There are currently a few agents already existing which are called "web crawlers" or "search engines". Using keyword-based searches, web crawlers automatically search through the Internet and index the information they find. Efforts are also underway to develop "metacrawlers" which perform multiple searches, in parallel, across the Internet. Trainable WWW agents based on neural-network software have also been developed.

References:
"AutoNomy" - Neural Autonomous Information Agents [http://www.camneuro.stjohns.co.uk/multimed/autonomy.html](http://www.camneuro.stjohns.co.uk/multimed/autonomy.html)

In the digital libraries and data base management domains, research efforts are underway to expand visual information retrieval (VIR) technology. VIR supports searching through image databases using the visual information contained in the image, such as color, texture,
composition, and structure rather than key words. This concept of content extraction provides a user the capability to retrieve visual information by asking a query like "Give me all pictures that look like this". The VIR system satisfies the query by comparing the content of the query picture with that of all target pictures in the database. This is call "Query by Pictorial Example", OBPE.


In the scientific visualization arena, a number of research groups have begun to explore building intelligence into visualization software. This concept allows a researcher or policy analyst to prescribe a particular analysis task such as compare ozone concentrations with power plant emissions for a given air pollution computational model scenario. The software system then automatically creates an appropriate visualization. The users of these task-directed or rule based visualization software systems will specify their area of interest, describe the data parameters, and determine an analysis objective. The intelligent software tool will then suggest and describe visual representations of the data which might include contour plots, isosurfaces, volume renderings, and animated vector representations.


Here, we propose the notion of integrating these three research efforts. We suggest building an intelligent WWW-based application that educates and assists novice and advanced users in the application of scientific visualization techniques. The future development of intelligent agents and VIR databases that are incorporated into Internet browsing tools and scientific visualization software will aid in the building of comprehensive decision support systems. These task directed decision support systems will allow researchers and policy analysts to specify analysis requirements. The system will then automatically construct appropriate visualizations that are linked to information databases.

III. V Acknowledgments

These notes and a number of the images are the result of many conversations and insights from my colleagues at the U.S. EPA Scientific Visualization Center .... so a warm thank you to Mark Bolstad, Tom Boomgaard, Al Bourgeois, Todd Plessel, Dan Santistevan, Mike Uhl, Jeff Wang, Zhang Yanching and Dudley Bromley (Manager of Visualization) in the Lockheed Martin Services Group.

I am also appreciative to EPA's Work Assignment Manager for Visualization - Lynne Petterson and my collaborator on GIS - Visualization integration research, Thomas Fowler (Lockheed Martin - GIS expert). Our work lives would not be as they are if it were not for the many scientists within and outside of EPA who have shared their data and sense of wonder with us. Special thanks to Donna Cox, Peter Kochevar, Don Brutzman, Bud Jacobson, Ron Vetter, and Polly Baker for their concepts and ideas cited in the references of these course notes. Warm regards are also extended to Bill Brown, Loey Knapp, William Ivey, and Tom Mace who shared their insights on the GIS - VIS Integration topic.
This page intentionally left blank.
Topic #4:

Maintaining Interactivity
in Visualizing Large Data Sets

Mike Botts and Ron Phillips

Earth System Science Laboratory
Global Hydrology and Climate Center
The University of Alabama in Huntsville
Huntsville, AL 35899
botts@stromboli.atmos.uah.edu

INTRODUCTION

The exponential increase of CPU power allows more sophisticated processing to be performed on increasingly larger data sets on the end-users workstation. Furthermore, scientific projects are capable of collecting much larger volumes of data than have been possible in the past. The Mission to Planet Earth EOSDIS system is expected to obtain and generate about 1 TB of data every day for a decade or more. The pre-EOS Pathfinder data sets that exist today are distributed as 80 to 600 MB files, which represent a day’s worth of sensor data or derived products.

Through need or desire, the scientist will be working with ever larger data sets and greater volumes of data than are practical with existing hardware and software. While Random-Access Memory (RAM) capacities per dollars have been growing somewhat, that increase has not kept pace with the growth of relatively inexpensive CPU power and disk storage capacities. Scientist are thus often frustrated by the inadequacies of most visualization tools to handle data sets that exceed available RAM capacities.

One factor that makes this issue particularly troublesome for the visualization tool developer is the need to maintain some sense of interactivity while visually exploring the data. This has generally forced visualization tool developers to either (1) require that all desired data be
retained in Random-Access Memory (RAM) for rapid access, or (2) forego true interactive visualization within the tool.

Interactivity is vital to the successful exploration of scientific data through visualization tools. Tools must be developed that provide and take advantage of capabilities for retaining interactivity while dealing with very large data volumes.

**IMPORTANCE OF INTERACTIVITY**

Interactivity is defined here as the perception of the user being able to rapidly change the data or the view of the data, in lieu of the user’s perception of only changing the view of a single image of the data. The last five years have seen an important growth from primarily “presentation visualization” toward “exploration visualization”. In presentation visualization, whether its the generation of a single image for hardcopy or a sequence of images that will be recorded onto video, high speed of rendering is desired but is not a requirement. For effective exploration visualization, interactivity, and thus rapid access to the data and high-speed rendering, are essential.

Interactivity, coupled with an intuitive interface, allows the scientist to explore the data in ways that only he or she is capable of doing. The tool developer should provide choices for the scientist and allow the scientist to rapidly alter the visual representation of the data, to freely move around through different space and time representations, to bring in new data or remove or hide other data, to test hypothesis and insight using easily accessible analytical capabilities, and as will be discussed below, to control the degree of accuracy relative to the desired degree of interactivity. In contrast to presentation visualization, in which the scientist more or less knows what he or she wishes to see, exploration visualization provides the scientist with the very important capability to investigate various “what-if” scenarios.
UTILIZING THE MEMORY HIERARCHY

Most existing visualization tools rely totally on reading and storing all required data in RAM. A few developers have recognized the importance of allowing access to large data sets and either (1) have relied primarily on accessing large data files on disk, thereby foregoing any true interactivity, or (2) have provided capabilities for filtering or subsetting large data files while reading from disk to RAM, thereby focusing primarily on interactivity at the expense of accuracy. Few, if any, development efforts to date have effectively utilized the total hierarchy of memory available in workstations in order to interactively and adaptively provide a balance between accuracy and interactivity within visualization tools.

The memory available to an application consists of a hierarchy of storage media ranging from:

<table>
<thead>
<tr>
<th></th>
<th>speed</th>
<th>capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video RAM</td>
<td>extremely high</td>
<td>very low</td>
</tr>
<tr>
<td>RAM cache</td>
<td>very high</td>
<td>very low</td>
</tr>
<tr>
<td>hard disk cache</td>
<td>very high</td>
<td>very low</td>
</tr>
<tr>
<td>RAM</td>
<td>high</td>
<td>medium</td>
</tr>
<tr>
<td>hard disk</td>
<td>medium</td>
<td>high</td>
</tr>
<tr>
<td>mass store</td>
<td>slow</td>
<td>very high</td>
</tr>
</tbody>
</table>

One strategy for maintaining interactivity in visualizing large data sets is to fully utilize all levels of memory available and to take advantage of the strengths of each level with regard to speed and capacity. Relying on swapping provided by the operating system is an inefficient means of utilizing this hierarchy.

To improve the use of the memory hierarchy, an application should have a general model of the workstation it was running on that would provide information regarding various cache and memory sizes, memory latencies, cache associativity, bus bandwidths, swap devices, co-processors, etc.. With this information, decisions could be made either under algorithmic control or under user-control about how to best spread application data across the various memory hierarchies. This workstation model would
obtain its values either from simple run-time tests performed during application initialization or, more likely, would read the various parameters from a common database supplied either locally or remotely. Providing such a general cross-platform workstation model as well as the user-interface front-end for algorithmically or directly allocating data should be an active area of research.

Provided with this information, a wiser and more application-specific approach would be to utilize "informed swapping", that is, swapping based on the data's natural structure and on the requirements of the application and the user. Some implementations of informed or intelligent swapping could be implemented regardless of the application's knowledge of the data or the scientist's intent, while others would require intimate knowledge of the data and the desires of the scientist.

Though this data information can be provided by the user, forcing them to provide detailed descriptions of data from unfamiliar sources can prevent a scientist from using the widest range of available data. In the ideal world, the data itself would provide all the needed information for the application to best utilize the available hardware. In practice, this data information tends to be split between the data and the application by using a common data format and a data-management library. This technique is widely used and solves many of the near-term data description problems. Unfortunately, as the data-management library's capabilities are updated over the years, the application may need to be updated, even though the underlying data format may not have changed. New workstation or networking hardware or new algorithms may also require significant changes to the data-management library or the common data format.

An even more application-specific strategy is to re-format the data to take best advantage of a particular visualization or analysis technique. Options would include creating multiple resolutions, subsetting areas of interest and/or tiling, resampling to lower numeric accuracy, removing unneeded data, pre-filtering to a specific range of values or grid, or any other technique that reduces or indexes the data. Any variant of this strategy is based on the assumption that the time spent re-formatting the data is much less than the time spent visualizing and exploring the data. This strategy also assumes that the scientist knows ahead of time which data sets to re-
format, thus the problem of having many large, unfamiliar datasets to explore and analyze is not generally helped by this strategy.

**BALANCING ACCURACY AND INTERACTIVITY**

As a general rule, increased accuracy requires greater computation time and can thus decrease interactivity. Said another way, one can take advantage of lower accuracy to increase interactivity. In general, the word inaccuracy strikes fear in the heart of most scientists, yet any rendering of data within the limits of our display media inherits a certain degree of inaccuracy. Furthermore, there are times when accuracy is less important than the ability to rapidly explore the data in order to gain insight using different perspectives or through testing multiple “what-if” scenarios. At other times, during final rendering for presentation or final analysis of the data, the highest degree of accuracy is desired no matter how long it takes. The secret is to provide adaptive and customizable capabilities for selecting the proper balance between interactivity and accuracy within both display and analysis processes.

Image warping and texture-mapping provide one example of using a simple means to control the tradeoff between accuracy and interactivity. As seen in Figure 1, consider an interactive tool that displays a three-dimensional globe with a texture-mapped global cloud-cover image wrapped around it. On the surface of the globe is a geolocated image from a satellite view, also texture-mapped into place. One technique for generating this scene would be to use time-consuming algorithms to accurately place each pixel from the source images into the final image of the globe.

A more flexible technique takes advantage of the fact that the human eye can't readily distinguish when a pixel may be a little off in a complex scene, particularly if the scene is animated. By defining one grid around the globe and another grid within the satellite image's view on the globe as seen in Figure 2, the technique only uses the time-consuming pixel placement algorithms at grid intersection points. The space between the grid points, which represents the majority of the pixels to be drawn, is filled-in using a simple image warp using the corners of each four-sided grid space as a guide. By allowing user control over the number of grid points within
either the globe or the satellite view, a simple means for trading accuracy for interactivity is achieved. With fewer grid points, the globe and the satellite view become increasingly “faceted” in appearance and more inaccurate within each grid space. The scene can be calculated rapidly however.

With more grid points, the resulting image becomes smoother and more accurate within each grid space at the expense of taking longer to calculate. In the limit, when the number of grid points equals or exceeds the number of pixels to be drawn, the resulting image is at least as accurate and time-consuming as the original technique mentioned. The user is free to interactively choose the representation that best suits their varying needs and workstation graphics capabilities. This technique works as effectively with hardware-assisted texture-mapping graphic libraries as it does with custom software-based texture-mapping libraries.

Though there can never be an optimal strategy for all data and visualization types, there can be “islands of expertise” in which software components are created that focus on domain-specific scientific areas and provide a range of interactivity versus accuracy, a range of visualization techniques, and a range of data “smarts”. In the near-term future, distributed data managed by distributed object servers will allow scientists to more easily use these interacting domain-specific software components to visualize and explore data, as opposed to waiting for the next great “everything-to-everyone” visualization package. Unfortunately, today’s domain-specific scientific and visualization knowledge is still buried in applications that cannot share their knowledge with other applications. This knowledge also cannot generally be scaled, allowing a scientist to choose to receive a quick but less accurate result. Too many scientific packages stress extreme accuracy over “back-of-the-envelope” approximations.

Regardless of the whether the software is implemented with the latest object-oriented language or classic FORTRAN, the ability to trade-off accuracy for interactivity cannot be easily made unless a change is made in the scientific data production and usage mentality. In most cases, a scientific dataset is mostly observed or processed data with little or no computationally useful information about the system or algorithms that produced the data. The dataset is generally produced (navigated, binned,
gridded, and projected) by a data production center and then distributed to the scientists who must live with any assumptions made at the production site. Without a computationally useful “data context”, it is difficult to reduce accuracy and gain interactivity without also reducing the data to a more meaningless form. Likewise, a domain-specific software component strategy cannot provide detailed expertise without access to a useful data context. Currently, time-consuming user intervention can supply data context, but it is generally not transferable between applications or datasets and it is an inefficient use of the scientist’s time. This shortage of useable data context is holding scientific visualization and analysis back by having too many data options frozen at a production site and by requiring the scientist to always work with ever-larger datasets and never a suitable proxy of the dataset.

As an example of the power of data context, consider a satellite dataset containing un navigated and unprojected data products, but rather minimally-processed satellite swath data plus the mathematical description of the satellite’s orbital and scan characteristics. Using a Space-Time Toolkit developed by the authors and an Observation Dynamics Model (ODM) such as SPICE, provided by NASA/JPL, the raw swath dataset can easily be navigated and projected by the end-user almost as quickly as reading a pre-projected swath from storage. The ODM provides the data context to describe any satellite or aircraft-based sensor, allowing the end-user to interactively change the accuracy and extent of the navigation, whether for rough, interactive data exploration or precise, non-interactive data generation. The Space-Time Toolkit provides a sensor and projection independent means to store and manipulate data in its native four-dimensional space, postponing the projection until final display. Corrections to the satellite's navigation, its attitude or time-offset, or the underlying map projection can be easily made, providing the scientist with a simple interactive means to explore “what-if” scenarios that are difficult to provide without the ODM's data context and the Space-Time Toolkit's native data storage.

Furthermore, algorithmic search techniques can take advantage of the mathematical satellite description to automatically find areas of sensor coincidence subject to specific constraints such as sun-angle, look-angle, or regional coverage. These geometry-based search techniques have the added
benefit of not requiring the raw swath data to be present. This allows a scientist to browse data in a hierarchical manner, starting with the most abstract description of the data and progressing towards the actual data in a series of steps that allow useful interactive data exploration along the way. In other words, data visualization and analysis doesn't begin when the scientist receives the dataset, it begins when the scientist poses their first data query.

CONCLUSIONS

The key to effectively using ever-larger scientific datasets is not to force all datasets to some common, production-site specific format useable by large, general tools but to provide encapsulated knowledge about the datasets that can be utilized by interacting domain-specific software components. The implementation challenge is to continually push software to be more flexible and to work more closely with scientists to produce what they need, rather than what software developer's want them to need. Directions for future research include developing smart models of the user's workstation and network memory hierarchy, developing additional data context schemes and scaleable domain-specific software components, researching useful distributed object interaction techniques, and providing scaleable techniques for interactive visualization on a wide variety of graphics hardware. Of these directions, developing techniques for data context generation and use is the most difficult to implement, but it holds great promise for providing a highly flexible framework within which the scientist decides how best to interact with their data.

The visualization developer should become more aware of the desire to explore larger files and greater data volumes within end-user tools. In parallel, the need for interactivity within visualization and analysis tools is placing greater requirements on rapid access to data in memory and rapid processing of the data within the tool. Two general philosophies for maintaining interactivity while visualizing large data sets have been presented. These include efficient and intelligent utilization of the entire memory hierarchy within the workstation and the ability to adaptively balance between interactivity and accuracy.
The development of general libraries for customizable memory utilization would be a boon to developers of future visualization and analysis tools. Likewise, rendering libraries and visualization application software should provide capabilities for the developer or end-user to customize and utilize the trade-offs between interactivity and accuracy.

Figure 1. Texture mapped images of GOES global satellite image and a background image from the Optical Transient Detector
Figure 2
Grids for globe and satellite view
Case Study #1: Visualizing Photochemical Model Data

Examining an Air Pollution Model

Theresa Marie Rhyne
Lockheed Martin
U.S. EPA Scientific Visualization Center
Research Triangle Park, North Carolina
trhyne@vislab.epa.gov

Photochemical models are used by atmospheric researchers to examine air pollution concerns. The setting of air quality standards has been an activity of the United States Environmental Protection Agency (EPA). Within EPA, a number of air quality models have been used for specific purposes. These include the Regional Acid Deposition Model (RADM, for exploring acid rain concerns), the Regional Oxidant Model (ROM for examining regional ozone concentrations associated with Clean Air Acts) and the Urban Airshed Model (UAM for investigating state and local ozone concentrations associated with the Clean Air Act). Each of these models has its own data format structure. As a result, customized Application Visualization System (AVS) data input modules were built to visualize these respective model outputs.


As part of its participation in the Federal High Performance Computing and Communications Program (HPCC), the EPA has evolved a "Grand Challenge" air quality management project entitled Models 3. In collaboration with the North Carolina Supercomputing Center (NCSC) and several universities, EPA is building a user friendly, multipollutant, multiscale air quality modeling system which will encompass the models listed above. This umbrella system will support collaborative computing and visualization. Issues associated with the umbrella system will be highlighted in the Future Horizons section of this case study.
For now, we will examine further the Regional Oxidant Model.

Overview of the Regional Oxidant Model:

The Regional Oxidant Model simulates most of the significant chemical and physical processes responsible for the photochemical production of ozone over 1000 kilometer or more domain for episodes typically lasting 10 - 15 days. These processes involve (a) horizontal transport; (b) atmospheric chemistry and subgrid-scale chemical processes; (c) nighttime wind shear and turbulence associated with the low-level nocturnal jet; (d) the effects of cumulus clouds on vertical mass transport and photochemical reaction rates; (e) mesoscale vertical motions included by terrain and the large scale flow; (f) terrain effects on advection, diffusion, and deposition; (g) emissions of natural and anthropogenic ozone precursors; and (h) dry deposition. Three separate atmospheric layers (above the ground surface) are described by every ROM model run and the output provides the geographic distribution of 37 chemical species including ozone, NOX, and ROG. (Note: NOX = NO + NO2 and ROG = carbon-weighted concentration sum of organic categories.) Typical geographic domains of the model include the northeast corridor, southeastern section and midwestern region of the United States.


ROM Visualization Efforts at the U.S. EPA:

Visualizations which use discrete colormaps to depict ranges of ozone concentrations have been developed for specific geographic domains. A map of the specific geographic domain is overlaid over the visualization to provide spatial context for the Regional Oxidant Model Data sets. Color legends and text labeling provide visual cues about the specific data sets. Time series animation sequences are displayed for the episode period using customized visualization software on Unix workstations. These animations are also frequently transferred to videotape for internal agency briefings and presentations at technical meetings.

Reference: Rhyne, Theresa, Mark Bolstad, Penny Rheingans, Lynne Petterson and Walter Shackelford, Visualizing Environmental Data at the
Figure #1: Visualization of Regional Oxidant Model for Northeastern Domain. NOX concentrations are depicted with the yellow-red color bar which Ozone concentration are shown with the blue-green color bar.

Working with atmospheric researchers and computational scientists, visualization techniques which examine wind inputs to the Regional Oxidant Model have also been developed. Wind vector representations are animated over time. These types of visualizations improve EPA researchers' understanding of inputs to photochemical models and can result in modifications to respective model algorithms.

In addition, volume visualization techniques have been applied to Regional Oxidant Model outputs. Using a customized splatter technique, we generate a simultaneous visualization of the three atmospheric layers of the model. This graphical representation displays cloud-like structures for each of the chemical species, animated over time. This provides a new way to view photochemical data.

Figure #2: Volume rendering (using the splatter technique) of three atmospheric layers of Ozone Concentrations for the Regional Oxidant Model - Northeast Domain.

The development and implementation of these Regional Oxidant Model visualizations has involved a "Renaissance Team" approach to visualization toolkit development. The composition of the team includes: a) atmospheric and computational scientists in the U.S. EPA's National Exposure Research Laboratory; b) policy analysts and computational scientists in the U.S. EPA's Office of Air Quality Planning and Standards; c) computational modelers of the Environmental Sciences Group of the North Carolina Supercomputing Center; and d) visualization specialists affiliated with the North Carolina Supercomputing Center and the Lockheed Martin Services Group (at the U.S. EPA Scientific Visualization Centers). Our goal has been to build tools which atmospheric scientists and policy analysts can use on Unix workstations for the daily examination and visual display of air pollution data sets.
Figure #3: Prototype AVS-based User interface for the Models 3 system. Here wind vectors are being visualized for two scenarios. (User Interface Designers: Todd Plessel and Kathy Pearson)

The U.S. EPA's Models 3 visualization efforts have recently advanced beyond the AVS-based prototypes and incorporated other visualization tools. VIS-5D is being used by scientists to visualize data sets from air pollution computational models. New development work with IBM Data Explorer is also underway. See Case Study #3 of these SIGGRAPH 96 course notes for more information on IBM Data Explorer. Case Study #4 provides a discussion of the VIS-5D tool.

Future Horizons: Collaborative Modeling:

The transfer of high performance computing and visualization technologies to State environmental protection agencies is an important concept included in the U.S. EPA's HPCC Program. One aspect of this process involves collaborative modeling for air quality efforts associated with the 1990 Clean Air Act. Support of this activity involves recognizing that each State will have its own criteria for computer hardware acquisition. As a result, environmental modeling and visualization tool development must address heterogeneous computer architectures.
Collaborative Modeling & Visualization: A Definition

Collaborative modeling involves facilitating the sharing of computational code and visualization tools among researchers and analysts located at various remote sites. For air quality efforts, collaborative modeling supports the sharing of photochemical computing resources among the State agencies and U.S. EPA offices. A fundamental step in this process involves utilizing the existing and proposed nation-wide networking infrastructure (NII, Internet) for sharing data and electronic communication. Another step includes the development of visualization tools and user friendly interfaces for collaborative exploration of environmental data sets.

Steps for Technology Transfer:

Critical issues associated with this technology transfer include:

a) Fostering the development of collaborative modeling by encouraging a common operating system environment (Unix) among the State agencies executing photochemical models.

b) Utilizing the existing and proposed nation-wide networking infrastructure (NII, Internet) for data transfer, electronic communications, and distributed computing.

c) Recommending State environmental protection agencies' access to high performance computing machines for running photochemical models. This includes evaluating the potential of providing State agencies with access to EPA's Cray C90 at the National Environmental Supercomputing Center.

d) Supporting the design of visualization tools and user interfaces which address the specific needs connected with photochemical modeling and associated environmental policy analyses and regulatory concerns. These tools need to operate across multiple computer architectures.

e) Examination of issues associated with modernizing and optimizing photochemical model code (specifically the regulatory versions). This includes the use of visualization as a diagnostic tool for the evaluation processes.
In order to facilitate the sharing of these high performance and visualization technologies, it is hoped that many of the tools developed will be in the public domain and located at an anonymous ftp site on the Internet.


**Example of the Need for Collaborative Modeling**

One example of the need for collaborative modeling and visualization among state environmental agencies is featured below. This and other items were highlighted at a May 1993 Urban Airshed Modeling Workshop sponsored by the U.S. EPA's Office of Air Quality Planning and Standards. Representatives from each of the state environmental agencies and respective EPA Program Offices attended this workshop.

**Going Beyond State Boundaries**

One example of collaborative modeling efforts involves determining the size of a given geographic domain before executing a photochemical air pollution model. A simple approach would be for each individual State to model its own geographic domain. Unfortunately, mother nature and natural events do not observe political State boundaries. This forces the designers and executors of photochemical models some discomfort. Trying to characterize incoming air pollution emissions and ozone from large upwind sources, outside of State boundaries or a prescribed geographic domain, poses challenges. Collaborations among adjacent States and sometimes the expansion of photochemical modeling domains to regional
zones rather than single State zones have been recognized and achieved. Visualization provides a diagnostic tool to depict the resulting impacts of establishing boundary conditions for modeling scenarios involving air pollution control strategies for the Clean Air Act.

This concern is complicated further when examining the timing of air pollution control strategy implementation. The guidelines of the 1990 Clean Air Act can create a situation where an upwind geographic domain (i.e. an individual State) has an air pollution control attainment date which is later than an adjacent downwind area (another adjacent State). This results in the potential for upwind control measures to be implemented on a time schedule incompatible with the downwind area's earlier attainment strategy requirements. Collaborative modeling and visualization tools can provide for mutual examination and evaluation of this photochemical modeling concern. Multi-domain regions can be examined which include all air pollution source and receptor areas. Time series animations provide for the visual display of pollution transport between geographic domains.

Concluding Remarks:

The U.S. EPA Visualization Center has worked with visualization software and user interface design on various Unix platforms. These include SUN, IBM RS 6000, DecStation 5000, Cray C90, Data General Aviion and Silicon Graphics workstations. Using the Application Visualization System (AVS) software environment, we have been able to transfer visualization implementations between these platforms. Technical issues associated with compiling software on multiple computer architectures do exist. The EPA Visualization team is continuing efforts to understand and clearly define these issues. Work is also underway to develop effective user interfaces to visualization tools on Unix workstations. Future horizons also include developing computing environments which support collaborative photochemical modeling. It is hoped that these tools will provide atmospheric researchers with the opportunity to explore and share the visual display of their data. Work is also underway to provide World Wide Web implementations which will provide researchers, policy analysts, and the general public with Internet access to air pollution information and visualizations.

Reference: Rhyne, Theresa Marie, Collaborative modeling and visualization: An EPA HPCC initiative, (under Hot Topics: Ronald D. Williams, editor), Computer (special issue on visualization - Arie E.


Acknowledgments:

Ellen Baldridge, Norm Possiel, Ken Schere, Tom Link and many other scientists and policy analysts at the U.S. EPA have been fundamental in helping to define and apply these ideas. Thanks are extended to Joan Novak, Robin Dennis and Walter Shackelford (U.S. EPA HPCC Program Officers) as well as Ken Galluppi (MCNC Environmental Sciences Group Manager) for providing additional encouragement. My colleagues at the U.S. EPA Scientific Visualization Center are: Mark Bolstad, Tom Boomgaard, Al Bourgeois, Todd Plessel, Dan Santistevan, Mike Uhl, Jeff Wang, Zhang Yanching and Dudley Bromley (Manager of Visualization) in the Lockheed Martin Services Group. Lynne Petterson is EPA's Work Assignment Manager for Visualization. Special thanks to Kathy Pearson and Lee Westover for their concepts and ideas cited in the references of this case study.
Case Study #2: The Interuse Experiment

Interactive Tools for Geolocation and Visual Comparison of Disparate Data Sets

Mike Botts
Earth System Science Laboratory
Global Hydrology and Climate Center
The University of Alabama in Huntsville
Huntsville, AL 35899
mike.botts@atmos.uah.edu

INTRODUCTION

Often the earth scientist must deal with a wide variety of data which is disparate in space and time. These data might exist for example as satellite or aircraft swath data, as binned and gridded data within different resolutions and map projections, as irregularly-spaced station data, as vectors, polygons, or paths, or as episodic event data. This disparity of data with regard to space and time presents major challenges for the tool developer who generally lacks expertise in geolocating satellite and aircraft swath data and in properly transforming between projection and image space. For this reason, few existing visualization tools provide adequate capabilities for visual fusion and comparative analysis of disparate data, without requiring prior processing and transformation by the user.

A tool’s ignorance of a data set’s space-time domain can sometimes be acceptable when visualizing a single data set. However, the use, or “interuse”, of multiple disparate data sets within a visualization tool requires that the tool not only have knowledge of the space-time domains of the data sets, but it must also be capable of transforming between these various space-time coordinate systems.

As part of the NASA/NOAA EOS Pathfinder Program, the Interuse Experiment team at the University of Alabama in Huntsville (UAH), in conjunction with two Interuse teams at NASA JPL, has been working to
refine and develop tools for providing scientific software developers with
the expertise required to meet space-time requirements for disparate earth
science data sets. These efforts have focused primarily on four activities:

(1) providing a standard Observation Dynamics Model for describing
the complete geometry and dynamics of all satellite and aircraft
sensor systems within end-user tools

(2) providing object-oriented libraries for transforming between map
projected and geographic space

(3) developing an improved general space-time toolkit for
incorporation into visualization and analysis tools

(4) providing improved capabilities for query and subsetting of
disparate data based on coincidence in space and time

Sample applications of these capabilities will be demonstrated in an
interactive computer-driven presentation.

OBSERVATION DYNAMICS MODELS

Dynamic sensor data, such as imagers and scanners aboard satellites and
aircraft, have traditionally not been well supported by visualization tools.
Most of these sensor systems sample a narrow swath of information along
the Earth’s surface during an instant in time. The navigation, geolocation,
and mapping of these data have traditionally been the realm of data
production centers and not end-user tools. Software for geolocating and
processing swath data have generally been written for a specific sensor
system and for application with the data center. The data production center
then creates higher-level data products by mapping the data to “standard”
2D map projections and by applying algorithms to generate meaningful
geophysical values. These products are generally produced with the
assumption that the end-user’s visualization and analysis tools lack data
sophistication and are confined to handling basic images or simple 3D grids.

However, there is a greater desire and need for the end-user to utilize the
low-level (i.e. unbinned, unmapped) data products within visualization and
analysis tools. These requirements are being driven primarily by high-resolution regional studies, by those scientists wishing to develop new or improved algorithms, and by strong desires to intercompare multiple disparate data sets. Unfortunately, software developed for and used by the data production centers for locating sensor pixels on the Earth's surface and for describing the dynamics of the sensor system is generally inadequate or too cumbersome for use by the developer of end-user tools. This is particularly true of the tool developer wishing to support a multitude of satellite and aircraft sensor systems, all with software (some proprietary) developed at various sites for completely different purposes.

In recognition of the traditional difficulties in geolocating swath data, many data centers are releasing low-level data with precalculated latitude-longitude values for each pixel, as well as other values of potential interest, such as solar incidence angles, look-angles, etc. While this has provided some benefits to the end-user, there are significant disadvantages in relying solely on explicit values for latitude and longitude: (1) data volumes of already large data sets is often increased by 80-300%, (2) inevitable corrections or refinements to the navigation/geolocation are difficult to implement without redistribution of all the data to all who have previously received the data, (3) geometric and dynamic relationships that are not precalculated are difficult or impossible for the end-user to derive, (4) the ability to geolocate data relative to a different planet model (e.g. the troposphere, the geoid, a digital terrain model, etc.) is lost, and (5) query and subsetting of swath data based on time-space parameters is greatly encumbered.

An alternative and significant beneficial approach is to provide capabilities for the end-user to generate on-the-fly all latitude-longitude and other relational information based on mathematical models of the sensor system. In essence, this approach brings the capabilities normally available at a data production center to the end-user. With such a system, values for latitude-longitude or any other relationship do not need to be precalculated, stored, and distributed, and could be easily refined at a later time. Such a system, referred to here as an Observation Dynamics model (ODM), would in essence completely describe the geometry and dynamics of the any supported sensor system. In order to be useful to the end-user or tool developer, an ODM must be developed in such a way as to be portable, to
easily handle a wide variety of dynamic sensor data (e.g. on satellites, aircraft, ships, etc.), be accurate, and to be efficient with regard to timely calculation of a large number of values.

In order to begin to meet the developer's, and thus the end-user's, needs for adequately handling low-level Earth-directed sensor data, the Interuse Experiment teams at UAH and NASA/JPL have been involved in adapting and testing the SPICE ODM system, developed originally by NASA JPL/NAIF for planetary and solar system exploration.

The SPICE system was developed to meet multi-mission needs and to be portable and usable by the end-user and software developer. The SPICE concept abstracts all sensor-system specific knowledge into standard data files which are separable from the more general software library which is used to calculate any required values on the fly. This is in contrast to traditional navigation software in which the sensor and spacecraft specifics are intimately interwoven into the software.

The system-specific data files, referred to in SPICE as "kernels", are broken down into the separable components of the sensor system, namely the Spacecraft position with time (S), the Planetary position, orientation, and shape (P), the Instrument geometry and dynamics (I), the platform rotation with time (C), and a time-tagged listing of system Events (E). The current SPICE software libraries exist as FORTRAN (SPICElib) and C++ (OoSPICElib) versions, with the FORTRAN version currently being the most robust and supported.

From the developer's view, an ODM such as SPICE, requires the developer to interface to only one software library for all sensor systems that are currently supported under that ODM. An ODM with a properly-developed high-level API also relieves the developer of the need to establish an expertise in satellite and aircraft navigation and geolocation.

The Interuse teams have developed prototypes which illustrate the importance of incorporating ODM capabilities within end-user visualization and analysis tools. Figure 1 illustrates the ability to interactively correct geolocation errors in satellite imagery data by manipulating the spacecraft attitude parameters (pitch, roll, and yaw) and spacecraft clock timing. The
Figure 1. The use of the SPICE ODM for interactive correction of geolocation errors within the Optical Transient Detector satellite data.

<table>
<thead>
<tr>
<th>Time</th>
<th>01:42:35.30002 102-1995</th>
<th>Engaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll</td>
<td>-5.600</td>
<td></td>
</tr>
<tr>
<td>Pitch</td>
<td>-1.400</td>
<td>Attitude Engaged</td>
</tr>
<tr>
<td>Yaw</td>
<td>5.300</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.700</td>
</tr>
</tbody>
</table>
satellite image is interactively geolocated using the new navigation parameter values, allowing the user to visually compare the result to an "absolute" geolocation criteria such as coastlines. In Figure 2, the use of satellite dynamics modeling allows the user to interactively determine times of coincidence between two satellite sensors or between satellite imagery and a particular geophysical phenomena (e.g. a hurricane). This provides the end user with the ability to query and request a subset of the large satellite archives, without requiring that the data be locally available. These and other applications will be demonstrated in the session. In addition to being used in in-house prototype tools, these capabilities are currently being integrated into existing visAnalysis tools, such as LinkWinds and IDL.

IMPROVED SPACE-TIME TOOLKIT

Most visualization and analysis tools require prior massaging of disparate data into a common, regular spatial grid before being able to be visually fused or analytically compared. These tools tend to function primarily in image space rather than a spatial and temporal domain which is more appropriate for the data. For instance, figure 3 illustrates the difference between working strictly in an image space, in which the bounding boxes in the upper left of the data sets are considered to be the same area, and working in geodetic space in which a tool understands the mapping of each data set to geodetic space, regardless of whether the data is a swath of satellite data or data gridded into various map projections. Similarly, these tools assume that data space and display space will be one and the same.

For tools that deal only within image space, the end user must resample all data into a common grid space before those data can be integrated or compared within the tool (figure 4). In addition to the significant overhead often required to transform and process data before visualization, this requirement forces the scientist to disturb the original data through binning and interpolation schemes, thereby reducing the amount and quality of information within the data and introducing highly undesired artifacts into the data. Likewise, within the temporal domain, the scientist must collapse and interpolate disparate data into common, discrete time slices before
Figure 2. The use of the SPICE ODM and Space-Time Toolkit for coincident search between two polar orbiting satellite data sets and a storm system, as viewed by the GOES weather satellite.

<table>
<thead>
<tr>
<th>Name</th>
<th>Visible</th>
<th>Time Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>OID</td>
<td></td>
<td>0.000 sec</td>
</tr>
<tr>
<td>SSMI</td>
<td></td>
<td>1755.000 sec</td>
</tr>
<tr>
<td>CONUS Lightning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mapping in Image Space versus Mapping in Geodetic Space

Figure 3. Tools that map only within image space would mistakenly consider the data in the bounding box at the upper left to be related. Tools that understand the transformation from data space to geodetic space understand the spatial relationship between data sets, regardless as to whether those data exist in a variety of spatial domains.
Figure 4. The typical space-time model used in many visualization tools requires that data existing within different spatial and temporal domains be resampled into a common grid before being integrated or compared within the tool.

DATA SPACE and DISPLAY SPACE are the same

Tool Domain

TYPICAL SPACE–TIME MODEL
being able to visually or analytically compare time-varying data within existing tools.

Time has traditionally been treated within tools and data sets as simply an index into images or files, and not as a true coordinate within the space-time continuum. As a result, few visualization, analysis, or GIS tools handle disparate data within the space-time coordinate frame in a manner that is adequate for meeting the scientist’s needs for comparative visualization and analysis.

Under the Interuse Experiment activities, an improved space-time toolkit is being developed in order to provide any application tool developer with improved capabilities for visualizing and analyzing multiple data sets that are disparate in space and time. The improved space-time toolkit is based on two premises: (1) the retention of data in its native form, and (2) the delaying of spatial and temporal transforms until the last possible moment, preferably at final display or analysis. As illustrated in figure 5, the Space-Time Toolkit assumes that the native spatial and temporal domains of the data may be different from the display space and that the tool must be able to transform the data on-the-fly to the any user-selected display space.

The improved space-time toolkit is capable of receiving and properly handling a wide-variety of data in which the values for latitude, longitude, altitude, and time are supplied. These data can consist of swath data, images, gridded products, static vectors, dynamic paths, irregularly-spaced station data, or irregularly occurring event data. In addition, the SPICE ODM and the interuse map projection library, provide capabilities for mathematical transforms from sensor space and map projection space to the latitude-longitude-altitude-time domain. The flexible design of the improved space-time toolkit allows continual addition of user-defined transformation capabilities as required.

The improved space-time toolkit incorporates an ability to adjust the accuracy of the mapping transformations in order to adaptively balance the requirements for accuracy and interactivity. Much of this capability relies on the ability to mathematically describe the spatial and temporal domain of the data through transforms. In addition, the space-time toolkit is designed
**Figure 5.** The improved model for space-time utilized within the Space-Time Toolkit allows data to be ingested and retained within its native space-time domain and provides “on-the-fly” transforms between data space and the user-selected display space.
to extend GIS concepts into 4D space, while providing more flexible spatial domains and smarter transforms than available in traditional GIS.

The data intelligence provided by these capabilities provides much greater flexibility in dealing with large multiple data sets. The tool illustrated in figure 6 is running on a system with 64 MB RAM, yet allows highly interactive exploration of multiple data sets with a data volume of about 150 MB. The tool utilizes adaptive tradeoffs between accuracy and interactivity, as well as adaptive decimation of the data based on display characteristics (e.g. zoom factors, etc.).

SOFTWARE

Libraries.

A major focus of the Interuse activity is to provide improved capabilities for any existing or future visualization and analysis tool. Thus, most of the core functionality being advanced through these activities will be incorporated into development libraries with both high-level and low-level APIs. With the exception of the JPL/NAIF provided SPICElib, these libraries are being developed in C++, but will be provided with APIs in FORTRAN, C, and C++.

The existing or anticipated libraries include the SPICE ODM, the space-time toolkit, a map projections library, and a binning/gridding library. Others for assisting in the comparative analysis of disparate data may be added in the future.

Applications.

Most of the application development under the Interuse Experiment activities have focused on either prototype applications to illustrate the practicality, use, and importance of the improved interuse capabilities or have been directed toward meeting specific needs of alpha test science teams. Efforts are currently underway to move these capabilities into the NASA/JPL LinkWinds visualization tool and to begin using IDL as a prototyping environment whenever possible.
Figure 6. A prototype interactive application running on a 64 MB workstation utilizing the SPICE ODM and the Space-Time Toolkit for visual integration of multiple data sets with a total volume of 150 MB.

Master Time: 19:23:45.48000 277-1995
Master Time Step: 491.410 sec
The applications to be demonstrated in this case study include the dynamic mapping of coincident data from multiple satellites and the query and subsetting of data archives based on time and space, or based on coincidence with other data sets and events.

ACKNOWLEDGMENTS

This activity has been funded through the NASA/NOAA Pathfinder program, under NASA Headquarters grant NAGW-3674, referred to as the Interuse Experiment. The author wishes to thank Martha Maiden for her insight into these issues and for her support, and wishes to acknowledge and thank the Interuse team members for their significant efforts and input: Chuck Acton and Bill Taber (NASA/JPL NAIF), Meemong Lee and Richard Weidner (NASA/JPL IAS Group), and Bob Creasey and Ron Phillips (UAH/ESSL).
Correlative Visualization Techniques for Disparate Data

A Case Study and Demonstration for

ACM SIGGRAPH '96 Course #16: Visualizing Scientific Data and Information: Focusing on the Physical and Natural Sciences

Lloyd A. Treinish
IBM Thomas J. Watson Research Center
Post Office Box 704
Yorktown Heights, NY 10598 USA
914-784-5038 (voice)
914-784-5130 (facsimile)
lloydt@watson.ibm.com

August 1996
Correlative Visualization Techniques for Disparate Data

Table of Contents

1. Introduction to Correlative Visualization Techniques for Disparate Data..................3-1

2. Visualization of Stratospheric Ozone Depletion and the Polar Vortex.......................3-3

3. Cartographic Projections.......................................................................................................3-25

4. Visualization of Scattered Meteorological Data: Study of Severe Rainfall Events in Northwestern Peru................................................3-33

5. An Extended Data-Flow Architecture for Data Analysis and Visualization.............3-51

6. Applications of Data Explorer..............................................................................................3-69

7. Author's Biography...............................................................................................................3-83
1. Introduction to Correlative Visualization Techniques for Disparate Data

Traditional visualization techniques have been quite limited in their application to earth, space and environmental sciences data. To illustrate the complexity of this challenge and viable solutions, consider some example problems using real data and real science. The first example concerns the analysis of spacecraft observations of global ozone, which are useful in understanding ozone depletion. An approach called correlative visualization, which is a set of methods to examine disparate data simultaneously, is applied to these observations as well as atmospheric dynamics data. For earth, space and environmental sciences applications, cartography must be introduced, which are methods of creating maps of the Earth. A second example is also offered, which illustrates techniques for working with scattered or point meteorological data. A commercial software package has been used to implement these techniques and address these problems via generic, data-independent methods, which have potential application to a much larger variety of data than is discussed herein. Abstracts for these topics follow.

2. Visualization of Stratospheric Ozone Depletion and the Polar Vortex

Observations made by the Total Ozone Mapping Spectrometer (TOMS) aboard NASA’s Nimbus-7 spacecraft have been critical to the study of stratospheric ozone. Direct analysis of the TOMS data yields important information on the morphology of the annual austral depletion region. However, when these data are visually correlated with other relevant atmospheric data (e.g., objective analyses of temperature, geopotential heights, winds), information about the underlying diurnal atmospheric dynamics of the stratospheric polar vortex and potential contributions from the upper troposphere can be gleaned. This includes the formation and breakup of the depletion region each Antarctic spring. These data require care in their presentation so that artifacts due to the visualization process are not introduced and erroneously interpreted as features in the data. The provided form of these data is ill-suited for the study of such phenomena that occur continuously over a nominally spherical surface (i.e., it tears the data). In addition, they are not uniformly available for the entire earth or at least spatial regimes being examined. Each of the data sets being examined are generally not geographically co-registered and are defined on differing geometric structures. These characteristics require non-traditional techniques for visual correlation based upon registration of multiple data sets of disparate structure with cartographic warping of regular and irregular geometries. Such an approach does not introduce interpolation and its artifacts into the registration or realization process. It is also independent of the choice of realization technique, and hence, provides a framework for experimenting with different visualization strategies. As a result, the fidelity of the original data sets is preserved in a coordinate system suitable for three-dimensional, dynamic presentation and examination of upper atmospheric phenomena.

3. Cartographic Projections

Cartography is an ancient art and science of methods to project -- mathematically transform all or part of the surface of a sphere (e.g., the earth) onto a two-dimensional, flat surface or plane. The process of map projection introduces distortions of the data and/or its geometry. The choice of a specific projection method in visualization is very important for the proper communications of information. It is very much dependent on the visualization task (e.g., exploration, analysis, presentation, decision support, etc.). Too often, a very popular projection, such as Mercator, or a simple rectilinear projection is employed without knowledge of the resultant distortion of the visualized data. A brief survey of common projection methods is provided.

4. Visualization of Scattered Meteorological Data: Study of Severe Rainfall Events in Northwestern Peru

The climate of coastal Peru ordinarily is very arid. Every few years this condition is disturbed by a phenomenon called El Niño, characterized by an ocean warming which appears off the coastlines of northwestern Peru and southwestern Ecuador. Excessive and severe rainstorms are the most disastrous consequences of El Niño, and such storms can cause great damage to human life, property, crops, and animal life. An examination of
daily data from 66 rainfall stations in the Chiura-Piura region of northwestern Peru from late 1982 through mid-1983 (before, during and after a major El Niño episode) yields information on the mesoscale structure of these storms. The data from the rainfall stations are typical of observational data that are scattered at irregular locations in two or three dimensions (i.e., data with no notion of connectivity or topology). Clearly, discrete realizations such as scatter plots are straightforward. However, the application of continuous realization techniques for qualitative visualization (e.g., surface deformation or contouring for such two-dimensional data) requires an intermediate step -- gridding, interpolation, kriging, etc. to define a topological relationship between the locations of data to form some type of mesh structure. From such a mesh, then one could apply various conventional continuous realization techniques. Several common methods are considered, and the results of their application to the study of the rainfall events are analyzed.

5. An Extended Data-Flow Architecture for Data Analysis and Visualization

IBM Visualization Data Explorer (Data Explorer or DX) is a general-purpose software package for scientific data analysis and visualization. It employs a data-flow-driven client-server execution model and is portable. It is built on a foundation of a generalized data model for unified data handling. It offers a large variety of polymorphic processing, realization and rendering capabilities. These architectural choices have significant impact on the scaling of operations to large, complex data sets, especially in the earth, environmental and space sciences.

6. Applications of Data Explorer

A few simple examples of how Data Explorer can be used for the class of applications discussed herein are discussed.
2. Visualization of Stratospheric Ozone Depletion and the Polar Vortex

Introduction

In the earth and space sciences, it is very common to organize geographically located data as a rectilinear grid with horizontal extent over the entire surface of the earth (i.e., latitude and longitude). In two dimensions that implies a topological primitive that is a rectangle of various sizes. In three dimensions the cell is a parallelepiped, with the height corresponding to altitude or atmospheric pressure, for example. These rectilinear mesh structures are ill-suited for the study of phenomena that occur continuously over a nominally spherical surface (i.e., they tear the data). In addition, these grids may not be fully populated due to missing data (i.e., when observations could not be made). In such cases, the grids could be viewed as being irregular.

Independent of grid type, cartographic techniques are often introduced to suitably deform the data to compensate for the problems inherent in the original structure (i.e., the use of a rectilinear representation for a spherical surface). Traditionally, such a transformation is accomplished by defining a new cartesian grid in the cartographic projection coordinate system, and then interpolating from the original rectilinear grid to the new one prior to any other operation. Given the curvilinear nature of such transformations, non-linear interpolation techniques are typically required to make the transformation of acceptable quality. Figure 2-1 illustrates this operation schematically in which the data values at three points in the original rectilinear, regular grid are combined to define the value at a single point in transformed space (e.g., the values are weighted averaged, where the weight is a function of distance from their original points to that in transformed space). In addition to being computationally expensive, such interpolation makes it impossible to preserve the fidelity of the data prior to rendering, especially if regions of no data or other discontinuities are present. Such discontinuities would be smoothed out.

Alternatively, by warping the underlying mesh structure, the geometry itself is transformed without affecting the data. Figure 2-2 illustrates this operation schematically in which there is a one-to-one mapping of each point in the original and the transformed grid. At each node in the deformed grid there is a data value that corresponds to a specific node in the regular grid.

Thus, any realization (the application of one or more visualization strategies that generate renderable geometry from a collection of data) is independent of the choice of a specific cartographic coordinate system, the data or mesh themselves, or how the data are specified with respect to the underlying mesh (e.g., assigned at each node or to an entire cell at its center). In addition, interpolation is not required as the initial operation to be applied to the data to be visually correlated. Instead, interpolation can be isolated to be the last step in the visualization process, namely rendering (e.g., Gouraud-shaded surfaces).

The use of appropriately warped curvilinear grids can preserve the fidelity of the data prior to rendering. This does require an environment that supports direct realization and rendering of data on curvilinear grids as well as regular ones. This must be coupled with the ability to independently manipulate data and its underly-
ing mesh structure or base geometry. In addition, the ability to simultaneously render disparate geometry (e.g., points, lines, surfaces and volumes of varying color and opacity) is very helpful in viewing the realization of multiple data sets.

Correlative visualization implies two tenets. First is the capability to look at multiple sets of data in exactly the same fashion (i.e., visual comparison within a common framework). That is, displaying different sets of data of disparate structure in an arbitrary geographic coordinate system independent of the data sets. Second is the capability to utilize a variety of visualization strategies within the chosen coordinate system, for examining a single set of parameters from one source or many parameters from multiple sources. Specific representation techniques illustrate different aspects of data. Hence, no single method is always suitable and not all techniques are useful for all data sets.

Therefore, methods that support the registration of multiple data sets in geographic coordinates, using similar cartographic warping of the respective data locations for the data sets in question shows promise as an alternative to meshing, interpolating or resampling the original grids of each data set to a common rectilinear grid in projected space for correlative visualization in the earth and space sciences. The former is used via the IBM Visualization Data Explorer developed by Visualization Systems, IBM Thomas J. Watson Research Center [Lucas et al, 1992]. This is in contrast to the latter approach used by the author via the NSSDC Graphics System, developed by NASA/Goddard Space Flight Center [Treinish, 1989] [Treinish and Goettsche, 1991].

**Stratospheric Ozone Depletion and the Polar Vortex**

The aforementioned approach for the analysis of multiple data sets via correlative visualization can be illustrated by an example scientific problem. There is a phenomenon that occurs in the earth’s upper atmosphere (primarily the stratosphere) above Antarctica during the winter and early spring of every year known as the polar vortex [Schoeberl and Hartmann, 1991]. This effect is characterized by a cyclonic circulation pattern around the south pole. Many researchers believe that ozone-destroying chemicals are trapped in this vortex during the cold and dark of Antarctic winter. Once spring begins and the polar region emerges from the long night, it is theorized that these substances react photochemically with ozone to break the molecule apart and thus, aid in the creation of the so-called Antarctic ozone hole. Hence, in late winter, regions of ozone depletion around the pole begin to form. Within a few weeks the ozone hole is completely established. By late spring the vortex weakens, causing the ozone depletion region to fragment and eventually dissipate. The question of interest is, what are the characteristics of the south polar vortex that can be derived from diurnal observations of atmospheric dynamics and how do they relate to independent measurements of ozone? The study of the appropriate data sets for the southern hemisphere winter and spring (June through December) are relevant. The examination of a single year, 1987, is made because that year showed the greatest amount of ozone depletion until recent years [Krueger et al, 1992].

**Total Column Ozone Data**

Perhaps the most critical effort to study stratospheric ozone has been via observations made by the Total Ozone Mapping Spectrometer (TOMS) aboard NASA’s Nimbus-7 spacecraft. Nimbus-7 is in a (polar) sun-synchronous orbit, which means that it can roughly provide global coverage of the earth for its suite of instruments once per day. Each portion of the earth was observed nominally under the same illumination conditions from day to day. Measurements made by Nimbus-7 TOMS show the daily global distribution of stratospheric ozone from late 1978 until early May 1993. It measured the total column density of stratospheric ozone by observing backscattered solar ultraviolet radiation in seven spectral bands. Approximately 200,000 such measurements were made each day, which covered the entire globe [Fleig et al, 1986].

TOMS required sunlight to operate. Hence, there are periods of missing data due to local polar winters (i.e., it is dark) in addition to the usual data dropout problems associated with spacecraft observations. These regions are visible as gaps in various realizations of the data. They are NOT the ozone hole. The data have been gridded in a regular lattice of 180 (1.0° in latitude) x 288 (1.25° in longitude) from the raw observations for daily global coverage with cells without data being flagged. The locations of missing cells are considered in all realizations. The total ozone measurements are in Dobson Units (DU), corresponding to a column density of 2.69 x 10^{16} molecules of ozone cm^{-2}. 

3-4
Figures 2-3a and 2-3b show traditional two-dimensional visualizations of the ozone data on October 1, 1987, which is during the ozone depletion season. The rectangular presentation of the data is consistent with the provided mesh in that it is torn at the poles and at a nominal International Date Line. This cartographic representation of the earth is known as a cylindrical equidistant or plate carré projection. The ozone data are overlaid with a map of world coastlines and national boundaries. In Figure 2-3a the data are realized as iso-contour lines at 50 DU intervals, which indicate the spatial distribution of discrete thresholds within continuous data. In Figure 2-3b a pseudo-color spectrum is used, which is linearly mapped over a range (110 to 650 DU) valid for the year of study (i.e., to provide consistent comparisons between single days or for animation), and should provide a continuous representation of continuous data. Figure 2-3b also has a fiducial overlay (lines of latitude or parallels and longitude or meridians at 30° spacing) in white, which have been registered in this same rectilinear coordinate system. The grid cells where there are no data are visible as gaps in the pseudo-color realization. The area of low ozone is visible as a bluish band stretched across the bottom of the pseudo-colored rectangle.

**Total Column Ozone (Dobson Units) -- October 1, 1987**

![Map of total column ozone on October 1, 1987.](image)

Figure 2-3a. Contour–mapped global column ozone on October 1, 1987.

Figure 2-4 shows the same representation as figure 2-3b, but transformed by the Mollweide cartographic projection. In addition, iso-contours lines at every 20 DU have been overlaid to help in the interpretation of the pseudo-color image. The ozone density value for each line has been used to assign the same color to the line as the surrounding image, but at a lower level of brightness. The Mollweide and similar projections are used relatively often by earth scientists as a way of preserving area in a display of the entire globe compared to the cylindrical equidistant projection. Other projections may preserve shape or linear distance, for example, on selected portions of the globe [Pearson, 1990]. For the Mollweide projection, all meridians, which converge at the poles, are ellipses except for the central meridian, which is a straight line and considered (a) true (representation of a line on the earth's surface). For example, the 90° meridians are circular arcs. The parallels are straight lines perpendicular to the central meridian. The equator is considered true. The Mollweide projection can be characterized by the following:

\[
x = R \sin(\phi) \\
y = \theta \cos(\phi)
\]  

(1)
where \( (\varphi, \theta) \) represents the location of each node on the earth's surface in the original mesh as [latitude, longitude],
\[ R \] is a scaling radius (e.g., 90°) and
\[ (x, y) \] represents the location of each node in the deformed, curvilinear (Mollweide) coordinate system with an assumed pole point of 90° north latitude and 0° east longitude.

To examine a continuous phenomenon with a central focus far from the equator and the Prime Meridian, such as the ozone hole, either a different pole point for the Mollweide projection must be used or in this case it would be appropriate to use a different map projection. Figure 2-5 illustrates the same data as in figures 2-3 and 2-4 except a polar orthographic projection for both the southern and northern hemispheres is employed, which are shown in the left and right side of the figure, respectively. For the orthographic projection, all meridians are straight lines radiating from the central pole. The parallels are concentric circles, which become compressed toward the equator. The orthographic projection can be characterized by the following:

\[
\begin{align*}
  x &= R \cos(\varphi) \cos(\theta) \\
  y &= R \cos(\varphi) \sin(\theta)
\end{align*}
\]

where \( (\varphi, \theta) \) represents the location of each node on the earth's surface in the original mesh as [latitude, longitude] and
\[ R \] is a scaling radius (e.g., 90°) and
\[ (x, y) \] represents the location of each node in the deformed, curvilinear (orthographic) coordinate system with an assumed pole point of 90° north latitude and 0° east longitude and an assumed pole point of 90° south latitude and 0° east longitude for the northern and southern hemispheres, respectively.

In addition to the pseudo-color spectrum, this two-dimensional cartographic projection of the data is extended by redundant realization as a deformed surface (i.e., both height and color correspond to ozone density). The bluish contiguous area over Antarctica clearly depicts the depletion region, illustrating the advantage of choosing an appropriate cartographic coordinate system. The height mapping clearly dramatizes the concept of a depression in the ozone layer while the color enhances this perception as color enhances a topographic map. It provides a continuous representation consistent with the spatial structure of the data and in animation presents the dynamics in an easily discernable fashion.

It should be noted that the use of hue-based pseudo-color mapping for realization and rendering of data as images can create problems in interpretation due to how the human visual system responds to color [Rogowitz et al, 1992]. For example, discontinuities appear in the pseudo-color representation (e.g., figure 2-3b) that are not in the data but are due to how humans perceive hue (i.e., discretely). However, such pseudo-color maps are virtually standard in many disciplines. The acceptance of alternate, perceptually correct pseudo-color maps (e.g., luminance-based, [Lefkowitz and Herman, 1992]) would be limited in these disciplines because of their unfamiliarity. Therefore, the introduction of redundant realization techniques such as surface deformation retains the familiar pseudo-color scale but helps to lessen its negative perceptual impact.

Below each translucent surface is a hemispherical map that has been registered in the same orthographic coordinate system as the ozone data. This map consists of a monochromatic topographic surface, which is deformed based upon height above or below sea level. The monochromatic scale is chosen to impart the appearance of a topographic map (e.g., the oceans are dark gray) for each hemisphere. This surface is created from a topographic data base on a rectilinear grid at 0.5° resolution. The grid is warped by the same orthographic projection that was applied to the ozone data, although the data were originally in a different coordinate system at different resolution. Both the ozone and topographic surfaces are Gouraud-shaded. In addition, the topographic map is overlaid with the same coastline map in magenta with political boundaries corresponding to each hemisphere that was used in figures 2-3a and 2-3b. The map geometry was transformed in a manner similar to that of the ozone and topographic data.
There is a seam in each of the orthographic surfaces corresponding to where east longitude is either \(-180^\circ\) or\(+180^\circ\), nominally the International Date Line, which is an artifact of the warping of the original rectilinear data onto a continuous surface, welding the discontinuity in the provided form of the data. The use of coordinate warping does preserve this inherent discontinuity in the data, which would not be the case if traditional interpolation techniques were chosen. In general this seam will not smoothly connect the surface due to how TOMS gathers data. This scanning instrument examined each portion of the earth at a different time of day, but still covering the entire globe once per day. Hence, observations on each side of that line were taken approximately 24 hours apart and usually are not the same.

Figure 2-6 carries this cartographic theme to a three-dimensional continuous surface by performing a cartesian to spherical coordinates transformation. For the spherical projection, all meridians are great circles converging at the poles. The parallels are also great circles, which become compressed toward the poles. The spherical projection can be characterized by the following:

\[
\begin{align*}
x & = (h + r)\cos(\phi)\sin(\theta) \\
y & = -(h + r)\cos(\phi)\cos(\theta) \\
z & = (h + r)\sin(\phi)
\end{align*}
\]

where \([\phi, \theta, r]\) represents the location of each node on the earth's surface as

- \([\text{latitude}, \text{longitude}]\) at its radial distance, \([r]\), from the earth's center in the original mesh
- \(h\) is the height above the (radial) surface of the earth, and
- \([x, y, z]\) represents the location of each node in the deformed, curvilinear (spherical) coordinate system

The ozone is now triply redundantly mapped to height (now radial), color and opacity so that high ozone values are thick, far from the earth and reddish while low ozone values are thin, close to the earth and bluish. Replacing the map for annotation is a globe in the center of this ozone surface, which is created from the same topographic data used in figure 2-5. The topography is warped onto a smooth, Gouraud-shaded opaque sphere (i.e., 259,200 polygons) and pseudo-colored to give the appearance of a globe by having all values around sea level and below appear light blue. The topography and ozone data are registered in a common spherical, earth-centered coordinate system. As with figure 2-5, southern and northern hemispheric views are shown, which are shown in the left and right side of the figure, respectively. Each spherical object gives the appearance of looking at a continuous phenomenon from two vantage points. The use of three redundant realization techniques results in patterns or textures, which are particularly effective in animation of time sequences for qualitatively identifying regions of spatial or temporal interest in the data. Such an approach shows promise for data browsing. On the other hand, the orthographic projection technique does not yield such an impression, although it does impart more of a quantitative “feel” to the visualization. Hence, this projection will be revisited in the subsequent discussions.

**Dynamics Data: Atmospheric Temperature and Winds**

Global atmospheric dynamics data (e.g., temperature and wind velocity) are often derived from spacecraft, balloon and aircraft observations, which have been modelled and gridded on a 2.5\(^\circ\) grid, 144 x 73 cells (longitude x latitude) at different levels in the atmosphere, based upon their pressure. Hence, a two-dimensional slice of these data at a specific pressure level is organized in a torn mesh similar to that of the total column ozone, but at lower resolution and in a different geographic coordinate system. These data may also have gaps in coverage, including only a partial value for some wind cells (i.e., one or two of the three vector components are missing). For the data being examined, there are seven pressure levels (1000, 850, 700, 500, 300, 200 and 100 millibar [mb]).

If one considers the Mollweide cartographic projection discussed in equation (1), figure 2-7 might be the result for October 1, 1987. Each of the seven pressure surfaces or slices of temperatures are independently used to define a cartographic warping, where the temperature is pseudo-colored according to a constant scale from 185 K
to 315 K with isothermal contour lines every 5 K and overlaid with a coastline and national boundary map in the same manner as the ozone data were shown in figure 2-4. Each of the seven Mollweide ellipses are stacked vertically according to a linear scale in pressure from 1000 mb to 100 mb, which can be seen in the axis. In addition, the opacity of each of the two-dimensional pseudo-color slices corresponds to the pressure height such that the 100 mb slice is almost transparent while the 1000 mb slice at the bottom is opaque. Since a pseudo-colored cartographic map is commonly used by climatologists to display two-dimensional data, figure 2-7 could be viewed as an attempt to extend that traditional method to three-dimensional (i.e., volumetric) data. Unfortunately, it is perhaps only effective for viewing a small number of slices simultaneously.

Alternatively, if one considers the spherical projection discussed in equation (3), figure 2-8 might be the result for October 1, 1987. In this case, the temperature data are treated as a true volume by warping the parallelepiped mesh representation of the atmosphere into a collection of concentric spherical shells that compose a volume of 73,584 nodes. The temperature data are pseudo-color and opacity-mapped, but now direct volume rendered. At the center of the volume is a globe, which is created from the topographic data as in figure 2-6. The radius of the globe is chosen to be the same as the inner radius of the spherical shell corresponding to the 1000 mb level. Although this picture may be interesting, little quantitative information can be derived. Hence, surface extraction techniques are appropriate for a more quantitative visualization.

Figure 2-9 illustrates these ideas for both the temperature and wind data on October 1, 1987. For the temperature data, surface extraction techniques are employed because direct volume rendering shows little quantitative information. The temperature data are realized as pseudo-color and opacity-mapped isothermal surfaces. The isosurfaces are at 194 and 294 K with the higher value (i.e., closer to the earth’s surface) being more opaque. The pseudo-color spectrum on the left corresponds to that of the temperature isosurfaces. The wind data are realized via streamlines generated by numerically integrating the paths taken by injecting massless particles in the wind field at 150 points uniformly distributed within the volume. The lines are pseudo-color-mapped by horizontal speed ranging from 0 to 80 m/sec, which correspond to the pseudo-color spectrum on the right. The vertical component of the wind ranges from about -33 to about +45 mb/sec.

Further study of the data using these techniques shows that there is a region of cold air (i.e., around 200 K) over the Antarctic during this period that is concentrated near 100 mb in pressure height, that is surrounded by high-speed winds. The shape of this cold air mass and its diurnal variation at first glance appear to be similar to the depletion region in the total column ozone data. The shape of this air mass can be seen clearly in figure 2-10. Two spherical, wire-frame meshes surround a monochromatic globe. The outer green mesh of quads is at the 100 mb pressure level. The inner blue mesh of triangles is at the 200 mb pressure level. An isosurface at 194K is shown for October 1, 1987, which has two components, south polar and equatorial. The isosurface is pseudo-colored on a linear scale according to the pressure height at which the 194K value is determined. The pseudo-color scale ranges from blue (100 mb) to green (200 mb). Regions below 200 mb are colored pink. The south polar component of the isosurface is mostly at the 100 mb level. Hence, further study focuses on 100 mb data.

Correlating Ozone with Temperature and Winds

Two different approaches to visually correlating the ozone and dynamics data for the 100 mb level are taken utilizing the concept of coordinate warping to achieve geographic registration. The 100 mb data are the same as used in figures 2-8, 2-9 and 2-10, and hence, are on a different grid than that of the ozone data. Since the aim of this study is to examine a phenomenon that is focused on a polar region and is nearly hemispheric in geographic extent, the orthographic cartographic projection discussed in equation (1) is utilized. They are illustrated using data from October 1, 1987 in an attempt to show the formation of the polar vortex and the ozone hole itself in figures 2-11 and 2-12.

Figure 2-11 shows four separate and different data-driven representations of the atmosphere over the southern hemisphere in the same geographic coordinate system utilizing the same three redundant realization techniques: pseudo-color mapping, surface deformation and iso-contouring. This is the same approach as used for the ozone data in figure 2-5, except for the addition of contour lines. In the upper left is the ozone column density with contours every 50 DU from 100 to 650 DU. The upper right shows the 100 mb temperature with contours every 5 K from 180 to 245 K. The lower left shows 100 mb horizontal wind speed with contours every 10
m/sec from 0 to 80 m/sec. Cells where one or more components of the wind velocity are missing are shown as gaps in this surface.

In an attempt to show the correlation among these observable quantities in data space, a simple linear model is constructed such that

$$M = C_O \tilde{O} + C_T \tilde{T} + C_v |\tilde{v}|$$

(4)

where $\tilde{O}$ is the normalized total column ozone ranging from 0 to 1 (scaled for the dynamic range of 110 to 650 DU),

$\tilde{T}$ is the normalized 100 mb temperature ranging from 0 to 1 (scaled for the dynamic range of 180 to 245 K),

$|\tilde{v}|$ is the normalized 100 mb horizontal wind speed ranging from 0 to 1 (scaled for the dynamic range of 0 to 80 m/sec),

$M$ is a scalar field representing a unitless linear combination of the three parameters ranging from 0 to 3, and

$C_O$, $C_T$ and $C_v$ are normalized weighting factors, which are set to 1.

The ozone data are bilinearly interpolated to the grid on which the 100 mb data are available prior to the normalization. Independent gaps in both the ozone and wind measurements are properly maintained in the computation of $M$, which is shown in the lower right with contours every 0.5 from 0.0 to 3.0. Each of the surfaces shows a similar structure -- a depression of comparable shape and areal extent over Antarctica for low ozone, temperature and wind speed, respectively, each with a boundary corresponding to that of the polar vortex. The relative contribution of each of these parameters to the depression structure for $M$ can be examined by interactively adjusting the weighting factors, $C_O$, $C_T$ and $C_v$ in the computation of $M$.

Figure 2-12 combines each of the three different parameters into one visual object. As with figure 2-5, both hemispheres are shown in the left and right sides of the figure, respectively. The data are stacked vertically and shown with topographic, coastline and national boundary maps. The difference between figures 2-12 and 2-5 are representations for the 100 mb horizontal wind velocity and temperature stacked between that of the ozone and the maps. Below the ozone surfaces are plates of vector arrows representing horizontal winds, whose direction correspond to the direction of the wind, and size and color correspond to wind speed, ranging from 0 to 80 m/sec. Below the winds and above the maps are flat, translucent planes corresponding to the 100 mb temperature realized as pseudo-color-mapped filled isothermal contours every 5K over the range of 180 K to 245 K.

A daily animation sequence from June through September shows the availability of polar ozone data as well as the formation of the depletion region in both presentations. A similar signature is visible as a blue region in the temperature data, before the ozone depletion occurs, as a cold air mass forms over Antarctica in the winter and persists into spring. A analogous but less well-defined shape forms in the wind realization. With the onset of spring, ozone observations become available and form a similar depression pattern to that of the temperature and wind. During this period, daily animation shows a clear rotation of the depletion region surrounded by the boundary of the polar vortex.

This rotation, which has a period of several days, is synchronous between the 100 mb and ozone data. The arrangement of the wind velocity arrows in figure 8 evoke a cyclonic pattern corresponding to the polar vortex, that appears almost steady-state in the winter and early spring. By early November, the warming of the upper atmosphere over Antarctica is obvious with direct correspondence to the dissipation of the polar vortex in the wind data and the breakup of the ozone depletion region.

Although the four time-varying surfaces do show the correlation among the ozone and dynamics data, they are difficult to observe, especially in animation. The eye tends to focus on one or two of the surfaces only. Therefore, at the cost of obscuring some of the data, the stacked approach shows the synchronous circulation in each data set for the southern hemisphere during this period at single glance.
Implementation

The techniques described herein have been developed via the IBM Visualization Data Explorer (DX), a general-purpose software package for scientific data visualization and analysis. It employs a data-flow-driven client-server execution model and is currently available on Unix workstations (e.g., Sun, Silicon Graphics, Hewlett-Packard, IBM, DEC and Data General) as well as a medium-grain, coherent shared memory parallel supercomputer (IBM POWER Visualization System) [Lucas et al, 1992]. DX simplified the implementation of cartographic warping and its simultaneous application to disparate data sets for correlative visual display by providing an extensible tool kit of polymorphic operations that are interoperable and appear typeless to the user. This polymorphism is a consequence of DX being built on a foundation of an unified data model, which describes and provides consistent access services for any data that is to be studied independent of shape, rank, type, mesh structure or dependency or aggregation. As a result, regular and irregular, structured and unstructured data are uniformly supported as well as regions where data are missing.

The relevance of DX to correlative visualization problems is illustrated with a simple example. Although DX has several interfaces, the techniques discussed herein utilized visual programming, in which each computational task is assigned an icon and the flow of control and data are defined by connecting the icons as a direct acyclic graph. Figure 2-13 shows a visual program that generates a visualization of total column ozone as a radially deformed, pseudo-color- and opacity-mapped spherical surface surrounding a globe similar to that in figure 2-6. Figure 2-13 also shows the pseudo-color and opacity map for the ozone surface via a colormap editor and the resultant image of the ozone data registered with a globe. Similar visual programs would show the atmospheric temperature around a globe, although the data are different in structure. The visual program has the following key operations:

1. Import               read data from disk
2. Color                assign a pseudo-color and opacity map
3. Include              subset data and mesh by value and indicate regions with no data
4. RubberSheet          deform mesh by value (for two-dimensional data, a deformed surface is created)
5. Sphere               warp mesh onto a sphere as in figure 2-2 according to equation (3)
6. Normals              compute normals for Gouraud shading
7. Globe                generate a globe representation of the earth from two-dimensional topography data
8. Collect              aggregate multiple sets of data into a single group
9. Image                generate an image and provide direct interaction with it

Almost the same visual program could be used to generate the atmospheric temperature around a globe as in figure 2-8, although the data are quite different in structure. Since the data are volume rendered instead of being shown as shaded surface, one would eliminate the RubberSheet operation from the program. For extraction of surfaces from volumes, the RubberSheet module would be replaced with Isosurface. The polymorphic nature of these operations permits their application to a variety of data. For a simple pseudo-color image display similar to figure 2-4, for any of these data, one would only have steps 1., 2., and 3., followed by the Mollweide operation, and then Image (step 9.). This notion could be applied to any number of cartographic projections.

The Sphere and Globe are implemented as macros -- their operation is specified in terms of other visual programming operations -- without custom coding. The Globe operation actually utilizes steps 1., 2., 5., and 6. (with an option for 4.). Hence, this program could be further simplified by encapsulating these steps into a macro. However, the lower-level operations are shown to tie the implementation with the aforementioned techniques.
Summation and Future Work

With easy-to-use tools to access, reorganize, realize and render yet preserve the salient characteristics of multiple data sets, a scientist can readily and appropriately scrutinize such data at many different levels through disparate techniques. The support of a plethora of visualization strategies properly coupled with powerful manipulation functions promotes the (visual) exploration and correlation of diverse data sets and thus, enables a scientist to extract knowledge from complex data. Specifically, the application of cartographic warping to the correlation of global atmospheric data sets yields visualizations that illustrate a simple notion about the possible relationship between temperature and winds, and their contribution below the tropopause to the formation of the polar vortex and ozone depletion.

The use of interpolation prior to realization appears to be unnecessary for the visualization of gridded data in a system with a sufficiently robust infrastructure of data structure and geometric support. However, the art of good interpolation is still required for the realization of scattered or point data as well as data on grids with strange or variable topology via continuous visualization techniques. In the latter cases, however, decomposition into unstructured meshes of the same simple primitives (e.g., triangles for planes and surfaces or tetrahedra for volumes), may appear to be an acceptable compromise. Such decomposition and refinement of interpolation methods are topics for additional research.

The ideas introduced with the analysis of observational data related to ozone and tropospheric dynamics can be extended by considering the correlation between these same ozone data and objective analyses that include the entire troposphere and stratosphere as a continuum. Visual correlation of these data is also useful for the examination of potential depletion regions in the northern hemisphere or the dynamics conditions for their possible formation. Data reflecting the typically disturbed conditions in northern polar regions in mid-winter through spring relative to the southern hemisphere yield more complex realizations than is the case for comparable southern polar regions. Although these data require more care in their presentation so that artifacts due to the visualization process are not introduced and erroneously interpreted as features in the data, an approach similar to that used for the southern hemisphere can still apply. In addition, comparison of these data with spacecraft observations of clouds may yield additional insight into the dynamics of ozone depletion since polar stratospheric clouds are believed to provide sites for conversion of ozone-destroying chemicals from inactive forms to highly reactive ones during the darkness of polar winter [Hamill and Toon, 1991].

Acknowledgements

All of the data sets discussed above were provided courtesy of the National Space Science Data Center, NASA/Goddard Space Flight Center, Greenbelt, MD.

References


Color Figure Captions


2-7. Mollweide warped pseudo-color-mapped and contoured global atmospheric temperatures stacked by pressure and opacity-mapped for 1000 mb to 100 mb on October 1, 1987.


2-10. Isosurface of global atmospheric temperatures at 194 K isolated mostly at the 100 mb level on October 1, 1987, showing the signature of the polar vortex.

2-11. Southern hemisphere orthographic warped pseudo-color-mapped deformed surfaces with iso-contours of global atmospheric data on October 1, 1987: column ozone with contours every 50 DU from 100 to 650 DU (upper left), 100 mb temperature with contours every 5 K from 180 to 245 K (upper right), 100 mb horizontal wind speed with contours every 10 m/sec from 0 to 85 m/sec (lower left), normalized linear combination of column ozone, 100 mb wind speed and 100 mb temperature with contours every 0.5 from 0 to 3 (lower right).

2-12. Southern and northern hemisphere orthographic warped global column ozone as pseudo-color-mapped deformed surfaces, 100 mb horizontal wind velocity as vector arrows pseudo-colored by speed and 100 mb temperature as pseudo-color-mapped disks with contours every 5K on October 1, 1987.

2-13. Example user interface from the IBM Visualization Data Explorer to generate a radially deformed pseudo-color and opacity-mapped spherically warped surface of global column ozone on October 1, 1987.
Figure 2-3b
Total Column Ozone (Dobson Units) -- October 1, 1987
Figure 2-4

Total Column Ozone (Dobson Units) -- October 1, 1987
Total Column Ozone (Dobson Units) -- October 1, 1987
Figure 2-7
Global Temperature for October 1, 1987: (1000 mb to 100 mb)
185K (blue) to 315K (red) with isotherms every 5K
Global Atmosphere
(1000 – 100mb)
October 1, 1987

Figure 2-8
Global Atmosphere (1000 - 100 mb) on October 1, 1987
Figure 2-10

Global Atmosphere (1000 - 100 mb) on October 1, 1987
194K Isothermal Surface Pseudo-Colored by Pressure Height
Green Wire Frame is 100 mb Level
Blue Wire Frame is 200 mb Level
Southern Hemisphere Upper Atmosphere
Figure 2-11

October 1, 1987

Total Column Ozone (DU)

100mb Temperature (K)

100mb Wind Speed (m/sec)

N(Speed + No(Ozone) + Mt.Temp.)
Figure 2-12

Global Stratospheric Column Ozone (110-650 DU)
100mb Wind (0-80 m/sec) and Temperature (180-245 K): October 1, 1987
Figure 2-13
3. Cartographic Projections

Introduction

Cartography is an ancient art and science of methods to *project* -- mathematically transform all or part of the surface of a sphere (e.g., the earth) onto a two-dimensional, flat surface or plane. The process of map projection introduces distortions of the data and/or its geometry. The choice of a specific projection method in visualization is very important for the proper communication of information. It is very much dependent on the visualization task (e.g., exploration, analysis, presentation, decision support, etc.). Too often, a very popular projection, such as Mercator, or a simple rectilinear projection is employed without knowledge of the resultant distortion of the visualized data. As a very brief introduction to the subject, consider four families of map projections: cylindrical, conic, azimuthal and singular, some of which have been applied to a data set of coastlines and national boundaries consisting of 109,928 points on 1247 lines. Each of these lines consists of anywhere from two to 3949 points.

Cylindrical Projections

Cylindrical projections are based upon the projection of the earth onto a cylinder that can be then unrolled into a plane. The familiar Mercator projection is the best-known of these projections. The principal characteristics of cylindrical projections are that (if the usual aspect is seen) all meridians and parallels are straight lines, with the meridians parallel and equally spaced, and the parallels also parallel and perpendicular to the meridians. The spacing of the parallels is what distinguishes one projection from another in this family.

Conic or Conical Projections

Conical projections are based upon the projection of the earth onto a cone, which, like the cylinder, can be unrolled. They are rarely, if ever, used for the whole world and are best suited to maps covering small ranges of latitude. However, for such ranges they often give the most "natural" looking maps. The principal characteristics of conical projections are that (if the usual aspect is seen) all meridians are straight lines converging at the pole, and all parallels are concentric arcs of circles. As in the cylindrical family, the spacing of the parallels is the only distinction between one conic projection and any other.

Azimuthal or Zenithal Projections

Azimuthal projections are in fact the limiting case of conic projections as the cone becomes a plane. Unlike the conics, they are frequently used for large areas of the world. Although they resemble the conic projections in having circular parallels and straight-line meridians in the normal (polar) aspect, they are distinguished by the fact that the parallels in conic projections are arcs of less than 360 degrees, while those in azimuthal projections are 360 degree circles. However, a more important factor is that the azimuthal projections unlike the conic projections, are frequently used in other than normal aspect, and in those cases the meridians and parallels are less of a clue as to the type of projection.

Singular Projections

Singular projections include all projections not included in the three families described above. A few of these projections are the Bonne, Sanson-Flamsteed sinusoidal and Mollweide projections. (In fact, the sinusoidal is a special case of the Bonne, but it is mathematically different because of the effect of the substitution of straight lines for circles as the curvature goes to zero). The Werner Cardiform projection is a special case of the Bonne, and is obtained by letting the standard latitude be 90 degrees.
Characteristics of Projections

A summary of twenty-five common projections and their useful characteristics and limitations is provided.

*Albers Equal-Area*

This is a conic projection chosen to preserve equality of areas. It is one with very little distortion in the areas between the standard parallels, although there is little but taste to recommend it over the Lambert Equal-Area Conic in this respect.

*Azimuthal Equidistant*

The principal feature of this projection is that all distances measured from the pole of the projection are true. Like all azimuthal projections, this one also preserves direction from the pole of the projection. This projection is frequently used with its pole chosen to be other than at the earth’s pole, when true distances from one specific point are desired.

![Figure 3-1. Azimuthal Equidistant Projection of Coastlines and National Boundaries](image)

*Bonne*

This is an equal-area projection which is particularly good for small areas, although it can be used for the whole world if desired. Parallels are circles, equally spaced, and distances measured along the parallels are true. Distances along one central meridian are also true. Shapes are relatively little distorted except far from the central meridian and standard parallel. Two special cases of this projection are the Werner (standard latitude 90 degrees) and sinusoidal (standard latitude 0 degrees); however, the sinusoidal is best considered as a separate case in its own right.

*Cylindrical (General) Perspective*

This is a rarely-used general perspective cylindrical projection.
**Cylindrical Central-Perspective**

This is a rarely-used projection which resembles the Mercator in its appearance.

**Cylindrical Equidistant (Plate Carré)**

This is the simplest cylindrical projection, and has the property that equal spaces on both meridians and parallels are to be found. Scale in the meridional direction is true; in other directions it varies with latitude.

![Cylindrical Equidistant Projection](image)

**Figure 3-2. Cylindrical Equidistant or Plate Carré Projection of Coastlines and National Boundaries**

**De L’Isle Conic**

This is one of a number of conic projections with equally spaced parallels, which thus guarantees a precise scale on the meridians.

**Gnomonic**

This is an azimuthal projection with one major advantage: all straight lines on the map represent great circles on the globe. It cannot, however, represent even one-half of the globe, as size distortion increases rapidly as one gets further from the pole of the projection.

**Lambert Azimuthal Equal-Area**

This is one of a large number of projections bearing Lambert’s name, and is the most common of them. As the name suggests, it is an equal-area projection and is also azimuthal; it and the Azimuthal Equidistant are the most commonly represented azimuthal projections. Shapes are greatly distorted in areas far from the pole of the projection, but it is one of only two azimuthal projections (the Azimuthal Equidistant is the other) that can picture the whole world.

**Lambert Conformal Conic**

The name of this projection is sufficient to describe it; it is conformal (preserves angles) except at the pole of the projection, and it is a conic with all the general characteristics of conic projections.
Lambert Cylindrical Equal-Area

As in all equal-area projections, this projection preserves equality of areas by compensating horizontal-scale increases with vertical scale decreases. In a cylindrical projection, this causes large shape distortions away from the equator of the projection. A transverse aspect of this projection is known as Cassini’s projection and is frequently used for areas near a specific meridian which is taken for the equator of the projection.

Lambert Equal-Area Conic

This is actually closely related to the Lambert Azimuthal Equal-Area, but adapted to a conic projection.

Mendeleev-Conic

This is an alternate name for the Simple Conic (see Simple Conic).

Mercator

This cylindrical projection is the most common one in use. Its main advantage is conformality, which means that shapes and sizes are true. Sizes, whether figured as distances or as areas, increase without limit as the position gets further from the equator. However, to preserve conformality, the scale at any point does not depend on the direction.

Figure 3-3. Mercator Projection of Coastlines and National Boundaries

Miller

This modification to the Mercator reduces the increase of size as distance from the equator increases by sacrificing absolute conformality. It is close enough to conformality to be used instead of the Mercator, but allows the entire world to be mapped, unlike Mercator which cannot include the poles. There are two versions of the Miller projection differing only in the degree of compression. The first type is nearer to the Mercator, the second is more extensively compressed.
**Mollweide**

The principal feature of this projection is that all the meridians are ellipses. The central meridian is a straight line and the 90 degree meridians are ellipses of eccentricity 0, or circular arcs. The equator and parallels are straight lines perpendicular to the central meridian. The central meridian and the equator are true length.

Figure 3-4. Mollweide Projection of Coastlines and National Boundaries.

**Murdoch Minimum-Error Conic**

This conic projection was designed to compromise the distortions in size and shape of various conics. It is not exact in size or shape, but close in all respects.
Orthographic

This azimuthal projection is often used when it is desired to give a visual appearance, since it represents the actual appearance of a globe as viewed from far away. It cannot show more than half the world at a time.

Figure 3-5. South and North Pole Orthographic Projections of Coastlines and National Boundaries

Plate Carré

This is an alternate name for the Cylindrical Equidistant Projection (see Cylindrical Equidistant Projection).

Ptolemy Equidistant Conic

This conic projection is one of the oldest, as its name implies, from Ptolemy’s atlases. It is, like the Simple Conic and several others, a projection with equally-spaced parallels, but is chosen to have a true scale along a standard parallel.

Sanson-Flamsteed

Also called the Sinusoidal Projection (see Sinusoidal Projection).

Simple Conic (Mendeleev Conic)

This, as its name implies, is the simplest of the conic projections. All parallels are equally spaced, and distances measured from the pole are true.

Sinusoidal (Sanson-Flamsteed)

This is an equal-area projection related to the Bonne. However, unlike the Bonne projection, the parallels are straight lines. Because the Sinusoidal Projection is most distorted far from the central meridian, it is frequently used in an “interrupted” mode in which the globe is divided into a number of sections (known as gores) centered on different meridians. Scales are true on all parallels and the central meridian (or central meridians, if interrupted) and so this projection, like the Bonne, combines equal area with equidistance along many lines.
**Stereographic**

This projection is the conformal member of the azimuthal family, that is to say, it preserves shapes and angles but not sizes. The distortion in size becomes significant at any distance from pole of the projection, but not as great as in the Gnomonic. It can represent all but one point on the world, but is frequently used for a circle representing only a relatively short latitude departure from the pole.

**Werner**

This projection is a special case of the Bonne (see Bonne). It can be used for the whole world, and when this is done, a heart-shaped map is obtained, hence the name *Werner Cardiform* projection. All the comments regarding the Bonne Projection apply to the Werner.

**References**

4. Visualization of Scattered Meteorological Data: Study of Severe Rainfall Events in Northwestern Peru

Introduction

The climate of coastal Peru and southwestern Ecuador is mainly controlled by the Humboldt current, a cold ocean current which travels northward along the coastline of Chile and Peru before dispersing near the equator. The current helps cause dry conditions to persist continuously along the Peruvian littoral, making the land strip between the Andes and the Pacific Ocean one of the most arid deserts in the world. Every three to seven years this condition is disturbed by a phenomenon called El Niño, characterized by an ocean warming which appears off the coastlines of northwestern Peru and southwestern Ecuador. This warming modifies the Humboldt current destroying the persistent high pressure zone normally induced by the Humboldt on the west side of the Andes, which in turn generates major changes in the local meteorology and climate [Rasmussen, 1985]. Excessive and severe rainstorms are the most disastrous consequences of El Niño, and such storms can cause great damage to human life, property, crops, and animal life. The rainfall from such episodes causes the flooding of existing rivers, huaycos (mud slides), and the sudden creation of new rivers and lakes.

The heating of the ocean off the Peruvian coast during El Niño periods is part of a larger scale warming of the eastern equatorial Pacific Ocean by several °C that creates large anomalies in oceanic and atmospheric circulation. These have, for example, led to the loss of much marine life. The El Niño of 1972 virtually destroyed the Peruvian anchovy fishing industry, which at that time represented a significant percentage of the world’s protein supply with a catch of about 12 million tons per year [Quinn et al, 1978]. The 1982-1983 El Niño has received wide attention for its severity [Philander, 1983]. In Peru alone, it was responsible for much loss of life, damage affecting over 80% of the highway system, railroad washouts, and material loss estimated in the billions of dollars. Such destruction emphasizes the need to better understand the meteorological forces unleashed by this powerful ocean-air interaction.

Goldberg et al [1987] have investigated the mesoscale structure of severe rainfall events during the 1982-1983 period by examining daily data from 66 rainfall stations in the Chira-Piura region of northwestern Peru. Figure 4-1 shows the location of this region, which was selected because it was most severely affected by the 1982-1983 El Niño and because the data were highly reliable and complete.

![Figure 4-1. Location of Peruvian Rainfall Stations](image)

These data support the study of rainfall characteristics over this localized region during El Niño and non-El Niño periods, as a function of elevation, geographic location, and time of year. The stations are enumerated in the Appendix.

Treatment of Scattered Data

Proper care in the treatment of these rain gauge data is required for effective analysis. Such care must be extended to the process of visualization, which creates pictorial representations of data that are a critical aid to interpreting these measurements. The data from the rainfall stations are typical of observational data that are scattered at irregular locations in two or three dimensions (i.e., data with no notion of connectivity or topology). Figure 4-2 is representative of a straightforward discrete realization of such data as a scatter plot to show the spatial distribution. Figure 4-3 illustrates the temporal distribution for a single station.

or the phenomena of which they represent discrete samples. Thus, the application of continuous realization techniques (e.g., surface deformation or contour-
An intermediate step of defining a topological relationship between the locations of data to form a mesh structure is required. Conventional continuous realization techniques can then be applied to such a mesh.

There is a long history of mathematical methods to create meshes from scattered data points. Each method does change the data and their artifacts must be understood because they will carry through to the actual visualization. This discussion is only meant as a very brief introduction to the topic. Nielson [1993] summarizes many of the methods in use today and their relative advantages and disadvantages.

The simplest and quickest approach is to create a regular grid from the point data by nearest neighbor meshing — find the nearest point to each cell in the resultant grid and assign that cell the point’s value as illustrated in figure 4-4. Such a technique is valuable because it preserves the original data values and distribution of a grid after a coordinate transformation may have taken place on a collection of points. Although computationally inexpensive, the results may not be very suitable for qualitative display because of the preservation of the discrete spatial structure.
An alternate approach that preserves the original data values involves imposing an unstructured grid dependent on the distribution of the scattered points. In two dimensions, this would be a method for triangulating a set of scattered points in a plane [Agishtein and Migdal, 1991]. This technique first requires the Voronoi tessellation of the plane with a polygonal tile surrounding each of the scattered points. These tiles are such that the locus of all points within a particular tile are closer to the scattered point associated with that tile than they are to any other points in the set. A triangulation can then be constructed which is the dual of the Voronoi tessellation (i.e., connecting a line between every pair of points whose tiles share edges). This is known as Delauney triangulation and is illustrated in Figure 4-5 as applied to the rainfall stations.

Figure 4-5. Delauney Triangulation of Rainfall Stations.

For a relatively random distribution of a small number of points such as these rainfall data, the application of continuous realization techniques to the triangulated mesh does not yield useful qualitative results. Consider figure 4-6, in which the mesh from figure 4-5 is pseudo-colored by amount of rainfall. Over the pseudo-colored grid are two magenta lines, which are the coastline of the Pacific Ocean and the Peru-Ecuador border, respectively, registered with the rainfall and altitude data. The rendering process applies bilinear interpolation to the value at each node to determine the color of each pixel in the image. Although the original data are preserved, the sparseness of the points results in a pseudo-colored distribution that is difficult to interpret. Artifacts of interpolation in color space from rendering dominate the appearance of the image instead of the spatial distribution of the data.

A potentially more appropriate method, and certainly one that is more accurate than nearest neighbor meshing, uses weighted averaging as illustrated in figure 4-7. For any given cell in a grid, the weighted average of the n nearest values in the original data distribution spatially nearest to that cell is chosen. A weighting factor, \( w_i = f(d_i) \), where \( d_i \) is the distance between the cell and the ith \((i = 1, \ldots, m)\) point in the original distribution, is applied to each of the n values. Figure 4-7 illustrates the case where \( n = 3 \). A common weight is \( w = d^{-2} \). These are variants of Shepard’s method [Shepard, 1968]. For example, Renka [1988] modified this approach with local adaptive surface fitting. Collectively, these methods are typically \( O[n\log(n)] \) in cost. Intermediate in quality and computational expense would be using linear instead of weighted averaging.

Figure 4-7. Weighted Average Gridding

All methods in the aforementioned class do introduce aliasing or smoothing of the data to achieve a gridded structure. The form of the interpolation may also impart artifacts on the results depending on the relative spatial variability of the original data vs. how close the interpolation function may be able to model that structure. Given a goal of qualitative visualization, such artifacts may be acceptable. Figure 4-8 shows a regular mesh with spacing of 0.04 degree (of latitude and longitude), onto which the rainfall data for January 26, 1983 has been gridded using \( d^{-2} \) weighting for \( n = 5 \). A quadratic was chosen to have local extrema at data sites. The number of neighbors was chosen empirically by experimentation with the application described in the Implementation section below. Five nearest neighbors provides a reasonably smooth distribution that is not significantly enhanced by the use of additional points.

The mesh and the data locations have been similarly pseudo-colored. There is good, but NOT perfect correspondence between the original data values and that of the interpolated grid, which is sufficient for qualitative realization. From this grid, isocontour lines of constant rainfall every 25 mm and a pseudo-colored image are created as shown in figures 9 and 10, respectively. Unlike in figure 4-6, the pseudo-color image in figure 4-10 is dominated by the distribution of the data, rather than rendering artifacts.
The detailed results of the analysis of the rainfall data are presented by Goldberg et al [1985]. The data have shown that enhanced rainfall during El Niño can be divided into two categories; first, a general increase in background levels over non-El Niño periods and second, sporadic bursts of intense rainfall superimposed over the enhanced background levels (cf., figure 4-3). The two categories of enhanced rainfall in comparison to the normally dry regions led to the choice of the hue-based pseudo-color map (e.g., figure 4-10) in which variations in each are illustrated. Hence, this increased background level is shown in yellow and green instead of brown (dry regions) while the bursts are shown with increasingly darker shades of blue. It is these sporadic bursts which were most responsible for the great damage during the 1982-1983 El Niño episode.

Effective gridding of the observations is critical for analysis. The weighted average method described earlier is used to create a pseudo-colored mesh independently for each day over the eight months of daily data being examined (November 1, 1982 through June 30, 1983). As a further aid to the study of spatial and temporal variations, the mesh is deformed by the altitude at each node, which is determined from the same gridding process applied to the altitude of each station as shown in the Appendix. The result is a simple elevation model, which gives a reasonable approximation of the topography in northwestern Peru, especially given the paucity of high-resolution elevation data for this region. This surface with pseudo-colored rainfall is used in figures 12 and 13.

The data further show that the severe storms (or bursts) often originate near the Andean foothills and may be induced by the interaction of rainbands moving inland from the coast with mountain downslope winds. This is best shown with a time sequence during such an event with local topography pseudo-colored by rainfall amount. Figures 12 and 13 illustrate the distribution of rainfall for January 24-27, 1983, and May 19-22, 1983, respectively, when a severe series of storms took place. Deep blue regions corresponding to these storms can be seen starting near the coast and diverting to the northwest after reaching the foothills of the Andes.

Applications to Other Data

To illustrate the generality of the distance weighted average approach to gridding of scattered data, consider its application to other data sets. Figure 4-13 shows a visualization of a collection of yearly averages of weather data for 1960 from 1702 stations scattered around the earth. Temperature and precipitation data are independently gridded to an irregular mesh of quads, which is a digitized representation of the earth’s land masses at one-degree resolution in latitude and longitude. The precipitation data are shown as a pseudo-color map in a Mollweide projection while the temperature data are shown as a pseudo-color contour overlay every five °C. The gridded representation illustrates correlation between lack of precipitation with very high temperatures and high precipitation with moderately-high temperatures, for example. These relationships would be difficult to see with scattered realization methods.

Figure 4-14 illustrates observations made by a nadir-looking instrument on board a polar-orbiting spacecraft, which examined backscattered ultraviolet radiation in the earth’s upper atmosphere. These measurements are used to determine the vertical distribution of ozone at discrete geographic locations along its flight path. The set of such ozone profiles measured on October 1, 1987 are shown on the left as pseudo-colored tubes, whose color corresponds to the ozone density and the diameter is the size of the instrument’s viewing footprint at the earth’s surface. The radial extent of these tubes on the scale of the earth’s diameter is exaggerated to reveal the profile structure. The locations of the observations are warped onto samplings on spherical shells in an earth-centered, three-dimensional coordinate system and registered with a gray-scale globe, which is derived from topography data. At the right is a direct volume rendering of the same data in the same coordinate system. The scattered data are gridded at each level to a mesh defined by the output of a complementary in-
instrument on the same spacecraft. The data are only interpolated within the valid geographic coverage of the instrument, then stacked vertically, spherically warped, pseudo-colored and opacity mapped. The spatial structure of ozone can be seen including a heavy concentration in the upper troposphere and lower stratosphere (i.e., the red shell) and a signature of the so-called ozone hole at intermediate and upper levels in the southern hemisphere in cyan. Application of familiar volumetric techniques to reveal this structure would not be possible without introducing the gridding process.

**Implementation**

The techniques described herein have been developed with IBM Visualization Data Explorer, a general-purpose software package for scientific data visualization and analysis. It employs a client-server architecture with an extended data-flow execution model and is available on Unix workstations (e.g., Sun, Silicon Graphics, Hewlett-Packard, IBM, DEC and Data General) [Abram and Treinish, 1995]. Data Explorer provides tools for operating on both scattered and gridded data. The Data Explorer **Connect** module performs the Delauney triangulation used in figures 5 and 6 while the **Regrid** module performs the weighted average interpolation used in figures 8, 9, 10, 11, 12, 13 and 14. The **Regrid** module provides independent control of the exponent of the weighting factor, the size of n and the radius of influence from each node of the grid within which to consider data points. In this case a radius of 0.36 degree (of latitude and longitude) was used. It should be noted that for each day of data, not all stations have rainfall measurements. This is NOT the same as a station reporting no rain. Data Explorer supports a notion of data invalidity. Hence, for any given day, only those stations having a measurement are considered by both the **Connect** and **Regrid** modules in creating gridded versions of the data. The choice of modules that support continuous realization is independent of the use of **Connect** or **Regrid** even though they result in different mesh structures because these Data Explorer operations are polymorphic and appear typeless to the user. This polymorphism is a consequence of Data Explorer being built on a foundation of an unified data model, which describes and provides consistent access services for any data that is to be studied independent of shape, rank, type, mesh structure or dependency or aggregation.

To illustrate how these tools may be applied to scattered data, consider figure 4-15, which contains a Data Explorer visual program. There are nine modules in this program:

- **Import** - reads rainfall data from disk.
- **Sequencer** - provides a frame counter for controlling daily animation.
- **Select** - chooses which day from the time series to process.
- **Include** - flags which stations failed to report a measurement.
- **Construct** - defines the aforementioned 0.04-degree-resolution grid (36 x 46) for interpolation.
- **Regrid** - interpolates the point data to a grid, which is highlighted. The configuration for this module is also shown, where the number of nearest neighbors, radius of influence and weighting factor are defined.
- **Colormap** - provides interactive construction of a custom color map, which is shown.
- **Color** - applies the color map to the gridded rainfall data.
- **Image** - renders an image and provides interactive tools for its manipulation.

This program will generate an animation sequence of the daily rainfall as (gridded) pseudo-color images, one of which is shown. A similar visual program was used to create figures 8, 9, 10 and 13. If Delauney triangulation were preferred over weighted average interpolation, then the use of the **Construct** and **Regrid** modules would be replaced by the **Connect** module (e.g., for figures 5 and 6).

Such a visual program can be expanded into a complete application that supports the study of the rainfall data. A snapshot of such an application in operation is shown in figure 4-16. The data for one day are shown as a pseudo-colored deformed surface, using the aforementioned technique. A time history of the rainfall for a single station is shown as a plot. Continuous and discrete realization techniques are provided. For the continuous techniques (e.g., contouring, imaging), the data are gridded as discussed earlier. The discrete techniques (e.g., scatter plots) may be overlaid with the continuous ones to evaluate the quality of the interpolation. Optionally, the topographic surface may be shown. The daily distribution of the rainfall at an individual station may be plotted. In addition, the gridding process may be adjusted interactively. The number of stations used to compute a value at each grid point and the radius of influence from each data point may be independently selected. These features are selected through three control panels containing Motif widgets. A VCR-like widget may be used to create time-sequenced animations of the daily data. Of course, the geographic view of the data may be varied interactively.
Conclusions and Future Work

The characteristic topography near regions such as Chulucanas (roughly in the center, cf., figures 3, 11, 12 and 16 and the Appendix), where such storms were observed to occur on a frequent basis, is ideal for the aforementioned interaction between the rainbands and the Andean foothills. The origin of the east-west rainbands near the north Peruvian coast is less clear but may be caused by low altitude wind surges, which are driven northward along the coast of Peru by a large and quasi-permanent high in the southeastern Pacific.

The potential influence of this phenomena may be determined by looking for changes in ocean conditions due to El Niño prior to the onset of rainbands. Figure 4-17 illustrates the ocean surface from an empirical model on January 24, 1983 before the major storm that is shown in figure 4-11. Sea level pressure is shown as a deformed surface, pseudo-colored by surface temperature. Streamlines of surface winds are also shown, pseudo-colored by wind speed. A region of ocean warming in the southeastern Pacific Ocean (pinkish-white) is associated with a decrease in sea level pressure and winds moving toward the South American coast.

Careful methods of gridding scattered (measured) data are critical for effective visualization, especially when used for continuous realization to yield qualitative information. The techniques described herein appear to be suitable for earth sciences applications other than meteorology (e.g., hydrological samplings, petroleum or mining well logs) as well as independent disciplines as diverse as medicine (e.g., measurements distributed on a patient's skin) or aerospace engineering (e.g., pressure along an airfoil or temperature inside a jet engine). Enhancement of the current study or extensions to other domains will require investigation of the applicability of other methods of gridding scattered data (e.g., [Nielson, 1993], [Gmelig-Meyling and Pfluger, 1990], [Smith and Wessel, 1990], [Yue-sheng and Lü-tai, 1990]).

Acknowledgments

The data are available courtesy of NASA/Goddard Space Flight Center, Greenbelt, Maryland.

References

### Appendix. Alphabetical Listing of Rainfall Stations and Coordinates.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number</th>
<th>Latitude (°S)</th>
<th>Longitude (°W)</th>
<th>Altitude (m)</th>
<th>Name</th>
<th>Number</th>
<th>Latitude (°S)</th>
<th>Longitude (°W)</th>
<th>Altitude (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTAMIZA</td>
<td>29</td>
<td>5.07</td>
<td>79.73</td>
<td>2600</td>
<td>LOS ALISOS</td>
<td>21</td>
<td>4.97</td>
<td>79.53</td>
<td>2150</td>
</tr>
<tr>
<td>ANIA</td>
<td>17</td>
<td>4.85</td>
<td>79.48</td>
<td>2450</td>
<td>MALLARES</td>
<td>6</td>
<td>4.85</td>
<td>80.73</td>
<td>45</td>
</tr>
<tr>
<td>ARANZA</td>
<td>16</td>
<td>4.85</td>
<td>79.58</td>
<td>1300</td>
<td>MIRAFLORES</td>
<td>11</td>
<td>5.17</td>
<td>80.62</td>
<td>30</td>
</tr>
<tr>
<td>ARDILLA</td>
<td>40</td>
<td>4.52</td>
<td>80.43</td>
<td>150</td>
<td>MONTEGRANDE</td>
<td>13</td>
<td>5.35</td>
<td>80.72</td>
<td>27</td>
</tr>
<tr>
<td>ARENALES</td>
<td>8</td>
<td>4.92</td>
<td>79.85</td>
<td>3010</td>
<td>MONTERO</td>
<td>48</td>
<td>4.63</td>
<td>79.83</td>
<td>1070</td>
</tr>
<tr>
<td>ARRENDAMIENTOS</td>
<td>57</td>
<td>4.83</td>
<td>79.90</td>
<td>3010</td>
<td>MORROPO</td>
<td>10</td>
<td>5.18</td>
<td>79.98</td>
<td>140</td>
</tr>
<tr>
<td>AUL</td>
<td>42</td>
<td>4.55</td>
<td>79.70</td>
<td>640</td>
<td>NANGAY DE MATALACAS</td>
<td>18</td>
<td>4.87</td>
<td>79.77</td>
<td>2100</td>
</tr>
<tr>
<td>AYABACA</td>
<td>2</td>
<td>4.63</td>
<td>79.72</td>
<td>2700</td>
<td>OLLEROS</td>
<td>53</td>
<td>4.70</td>
<td>79.65</td>
<td>1360</td>
</tr>
<tr>
<td>BARRIOS</td>
<td>33</td>
<td>5.28</td>
<td>79.70</td>
<td>310</td>
<td>PACAYPAMPA</td>
<td>23</td>
<td>5.00</td>
<td>79.67</td>
<td>1960</td>
</tr>
<tr>
<td>BERNAL</td>
<td>36</td>
<td>5.47</td>
<td>80.73</td>
<td>32</td>
<td>PAITA</td>
<td>67</td>
<td>5.08</td>
<td>81.13</td>
<td>6</td>
</tr>
<tr>
<td>BIGOTE</td>
<td>34</td>
<td>5.33</td>
<td>79.78</td>
<td>200</td>
<td>PALOBLANCO</td>
<td>28</td>
<td>5.05</td>
<td>79.63</td>
<td>2800</td>
</tr>
<tr>
<td>CANCHAQUE</td>
<td>35</td>
<td>5.37</td>
<td>79.60</td>
<td>1200</td>
<td>PALTASHACO</td>
<td>30</td>
<td>5.12</td>
<td>79.87</td>
<td>900</td>
</tr>
<tr>
<td>CHALACO</td>
<td>25</td>
<td>5.03</td>
<td>79.80</td>
<td>2250</td>
<td>PANANGA</td>
<td>43</td>
<td>4.55</td>
<td>80.88</td>
<td>450</td>
</tr>
<tr>
<td>CHIGNIA</td>
<td>60</td>
<td>5.60</td>
<td>79.70</td>
<td>360</td>
<td>PARAJE GRANDE</td>
<td>65</td>
<td>4.63</td>
<td>79.92</td>
<td>1500</td>
</tr>
<tr>
<td>CHILACO</td>
<td>3</td>
<td>4.70</td>
<td>80.50</td>
<td>90</td>
<td>PASAPAMPA</td>
<td>31</td>
<td>5.12</td>
<td>79.60</td>
<td>2410</td>
</tr>
<tr>
<td>CHULUCANAS</td>
<td>9</td>
<td>5.10</td>
<td>80.17</td>
<td>95</td>
<td>PICO DEL ORO</td>
<td>41</td>
<td>4.53</td>
<td>79.87</td>
<td>1325</td>
</tr>
<tr>
<td>CHUSIS</td>
<td>37</td>
<td>5.52</td>
<td>80.82</td>
<td>12</td>
<td>PIRGA</td>
<td>61</td>
<td>5.67</td>
<td>79.62</td>
<td>1510</td>
</tr>
<tr>
<td>CIRUELO</td>
<td>64</td>
<td>4.30</td>
<td>80.15</td>
<td>202</td>
<td>PUENTE INTERNACIONAL</td>
<td>63</td>
<td>4.38</td>
<td>79.95</td>
<td>408</td>
</tr>
<tr>
<td>CORPAC</td>
<td>15</td>
<td>5.20</td>
<td>80.62</td>
<td>49</td>
<td>SAN JOAQUIN</td>
<td>32</td>
<td>5.13</td>
<td>80.35</td>
<td>100</td>
</tr>
<tr>
<td>ESPINDOLA</td>
<td>49</td>
<td>4.63</td>
<td>79.50</td>
<td>2300</td>
<td>SAN MIGUEL</td>
<td>12</td>
<td>5.23</td>
<td>80.68</td>
<td>29</td>
</tr>
<tr>
<td>FRIAS</td>
<td>20</td>
<td>4.93</td>
<td>79.93</td>
<td>1700</td>
<td>SAN PEDRO</td>
<td>27</td>
<td>5.08</td>
<td>80.03</td>
<td>254</td>
</tr>
<tr>
<td>HUANCABAMBA</td>
<td>68</td>
<td>5.23</td>
<td>79.43</td>
<td>1052</td>
<td>SANTO DOMINGO</td>
<td>24</td>
<td>5.03</td>
<td>79.87</td>
<td>1475</td>
</tr>
<tr>
<td>HUAR HUAR</td>
<td>62</td>
<td>5.08</td>
<td>79.47</td>
<td>3150</td>
<td>SAPILLICA</td>
<td>56</td>
<td>4.78</td>
<td>79.98</td>
<td>1466</td>
</tr>
<tr>
<td>HUARA DE VERAS</td>
<td>47</td>
<td>4.58</td>
<td>79.57</td>
<td>1680</td>
<td>SAUSAL DE CULUCAN</td>
<td>4</td>
<td>4.75</td>
<td>79.77</td>
<td>980</td>
</tr>
<tr>
<td>HUARMACA</td>
<td>14</td>
<td>5.57</td>
<td>79.52</td>
<td>2100</td>
<td>SICCHEZ</td>
<td>44</td>
<td>4.57</td>
<td>79.77</td>
<td>1435</td>
</tr>
<tr>
<td>JILILI</td>
<td>46</td>
<td>4.58</td>
<td>79.80</td>
<td>1330</td>
<td>SUYO</td>
<td>39</td>
<td>4.50</td>
<td>80.00</td>
<td>250</td>
</tr>
<tr>
<td>LA ESPERANZA</td>
<td>7</td>
<td>4.92</td>
<td>81.07</td>
<td>12</td>
<td>TACALPO</td>
<td>50</td>
<td>4.65</td>
<td>79.60</td>
<td>2010</td>
</tr>
<tr>
<td>LA TINA</td>
<td>1</td>
<td>4.40</td>
<td>79.95</td>
<td>427</td>
<td>TALANEO</td>
<td>26</td>
<td>5.05</td>
<td>79.55</td>
<td>3430</td>
</tr>
<tr>
<td>LAGARTERA</td>
<td>54</td>
<td>4.73</td>
<td>80.07</td>
<td>307</td>
<td>TAPAL</td>
<td>55</td>
<td>4.77</td>
<td>79.55</td>
<td>1890</td>
</tr>
<tr>
<td>LAGUNA RAMON</td>
<td>59</td>
<td>5.55</td>
<td>80.67</td>
<td>9</td>
<td>TEJEDORES</td>
<td>5</td>
<td>4.75</td>
<td>80.25</td>
<td>230</td>
</tr>
<tr>
<td>LAGUNA SECA</td>
<td>19</td>
<td>4.88</td>
<td>79.48</td>
<td>2450</td>
<td>TIPULCO</td>
<td>52</td>
<td>4.70</td>
<td>79.57</td>
<td>2600</td>
</tr>
<tr>
<td>LANCONES</td>
<td>45</td>
<td>4.57</td>
<td>80.47</td>
<td>110</td>
<td>VADO GRANDE</td>
<td>38</td>
<td>4.45</td>
<td>79.60</td>
<td>900</td>
</tr>
<tr>
<td>LAS LOMAS</td>
<td>66</td>
<td>4.65</td>
<td>80.25</td>
<td>265</td>
<td>VIRREY</td>
<td>58</td>
<td>5.53</td>
<td>79.98</td>
<td>230</td>
</tr>
</tbody>
</table>
Color Figure Captions

4-6. Pseudo-Colored Rainfall Distribution from Delauney Triangulation of Stations.

4-8. Pseudo-Colored Mesh from Weighted Averaging and Rainfall Stations.

4-10. Pseudo-Colored Rainfall Distribution from Weighted Average Gridding of Stations.


4-13. Pseudo-Colored Precipitation and Pseudo-Color Temperature Contours from Gridded Global Station Data.

4-14. Weighted Average Gridding Applied to Atmospheric Profile Data.

4-15. Data Explorer Visual Program Illustrating Weighted Average Gridding of Stations.

4-16. User Interface of a Data Explorer-based Application for Studying Peruvian Rainfall Data.

4-17. Conditions in the Pacific Ocean on January 24, 1983.
Figure 4-6
Rainfall (mm) in Northwestern Peru (from El Nino): January 26, 1983
Figure 4-8
Rainfall (mm) in Northwestern Peru (from El Nino): January 26, 1983
Figure 4-10
Rainfall (mm) in Northwestern Peru (from El Nino): January 26, 1983

Longitude (Degrees East)

Latitude (Degrees North)
Figure 4-11
Figure 4-13
Yearly Average of Global Weather Station Data for 1960
Figure 4-14
Nimbus-7 SBUV Ozone Layer Profiles on October 1, 1987

Original Atmospheric Profiles

Profiles Gridded Separately at Each of 12 Pressure Levels (1000 to 0.24 matm-cm)

Ozone Column Density (Dobson Units)
Figure 4-17

Ocean Conditions on January 24, 1983

[3D diagram showing ocean conditions with temperature, sea level pressure, and wind speed indicators]
5. An Extended Data-Flow Architecture for Data Analysis and Visualization

Greg Abram and Lloyd Treinish
IBM Thomas J. Watson Research Center

Introduction

Over the last several years a number of software systems that provide "visual programming", which embodied a notion of data flow, have been implemented (e.g., Haeberrl [1988], Upson et al. [1989], Abram and Whitted [1990], Dyer [1990], Rasure and Wallace [1991], SGI [1991], Kass [1992], Lucas et al. [1992]). They were created under the premise that this paradigm was simple enough for users that are not experienced programmers to build applications. It was further assumed that this approach would greatly simplify the implementation and prototyping of computer graphics, data analysis and visualization systems that are composed of varied and often complex tasks. However, as the visualization community matured and the users of these tools grew in their sophistication, efforts to apply these systems to problems of realistic size and complexity illustrated a number of deficiencies within the typical implementations (Ribarsky et al. [1992], Burnett et al. [1995]). The challenge from the perspective of developing tools for data analysis and visualization based upon the data-flow paradigm is to preserve the virtues of such an approach while trying to minimize the inherent limitations embodied by the use of a naive "data-flow" execution model for the visual programs. An outline of the key advantages and disadvantages of the data-flow architecture will establish these points. This will serve as background for a discussion of an implementation that extends the idea of data flow to include capabilities necessary to support realistic problems, while continuing to support its traditional advantages.

Visual Programming and Visualization

Modular visualization environments, such as IBM Visualization Data Explorer (Lucas et al. [1992]), incorporate visual programming tools that allow complex systems to be constructed as networks of atomic tasks. For users with an idea of their goals and a basic understanding of the set of provided functions, the construction of sophisticated applications is made simpler and more intuitive -- programming by plumbing. The building of an application using these tools closely matches the user's mental model of the computation. In effect, the visual program is simply a graphical representation of the process to be executed.

Visualization applications seem particularly well tailored to the use of a visual programming paradigm. Generally speaking, the atomic operations of the visualization are well-defined and high-level so that sophisticated visualizations can be created by relatively simple networks of tools chosen from a predefined set. Figure 5-1 illustrates this idea for a program that imports data, computes both an isosurface and a planar mapping, and renders the results in a single image. Visualizations may be parameterized by the incorporation of inputs that relate interactor widgets to network inputs. These inputs may be represented with the visual program simply as modules with no inputs. Instead, their inputs come asynchronously from an associated "input device" or interactor (e.g., a graphics user interface widget such as a slider).

![Figure 5-1. A visual program incorporating two visualization techniques.](image)
Data-flow Execution

In a true data-flow implementation, all modules are pure functions (i.e., their outputs are fully defined by their inputs). Hence, processes are stateless with no side-effects (Arvind and Brobst [1993]). An examination of figure 5-1 leads to such an execution model. Imagine a set of available processes waiting for their inputs from the processes upstream in the network. In figure 5-1, the Collect module waits for inputs from the Isosurface and MapToPlane modules. When their inputs are received, they run, and when finished they distribute their results to the modules waiting downstream. In figure 5-1, Import would send its results to the waiting Isosurface and MapToPlane modules. In effect, this execution model is entirely data-driven and top-down: the execution of modules is dependent solely on the passage of data through the system.

Problems with Data-Flow Execution

While this simple data-flow execution model seems a natural mechanism for the execution of visual programs, a closer examination reveals that real-world problems are more complex. In order to function efficiently, it is vital that the system avoid unnecessary work. In general, there are two reasons why modules present in a visual program or (directed acyclic) graph may not need to be executed when their turn comes: 1) their results are not actually required by a result of the network and 2) their inputs are unchanged from the last time the module was executed (i.e., the result will be the same).

Identifying Required Results

In reality, the outputs of a visualization network occur in modules that have side-effects. They produce results outside of the network itself such as the display of images on a workstation or the creation of output files. Unless the result of a module ultimately affects an input to a module that produces a side-effect, that module does not have to be executed (e.g., conditional execution -- see below).

Eliminating modules that are not ancestors of modules with side-effects can be done by pre-processing the network before the actual data-flow network evaluation commences. This is done by traversing the graph bottom-up, beginning at each module known to have side-effects and flagging each module as it is encountered. Once this is complete, modules that have not been flagged do not have to be executed.

Conditional Execution

A much more difficult problem arises when conditionals are incorporated in networks. Conditionals may be used to offer the user a selection among several methods of visualizing a data set. In figure 5-2, a Switch module allows either an isosurface or a mapping plane to be displayed, based on user preference.

In a simple data-flow execution model, this Switch module will be executed when its inputs are available (i.e., the results of the Isosurface and MapToPlane modules, and the value of the selection input). On execution, the Switch module chooses whether to pass the Isosurface or MapToPlane result to the output based on the selection input. In the case of a pure data-flow model both the Isosurface and MapToPlane modules execute before the decision as to which will actually be used is made. Since both operations can be computationally expensive, the superfluous execution of both of them is very inefficient.

Figure 5-2. A data-flow network with a user choosing between two visualization techniques.

Again, this problem can be handled within the simple data-flow execution module by an examination of the graph prior to execution. In this case, the selection value, which comes from an external source (e.g., an interactor presented to the user) is essentially static, and known a priori. Hence, the selection may be performed by a simple transformation of the graph: excising the Switch module altogether, and substituting arcs from the selected source (either Isosurface or MapToPlane) to each of the modules that, in the original network, received
the result of the Switch module. This leaves the un-selected module dangling. It and any of its ancestors that are therefore made unnecessary will not be executed.

It should be noted that this approach fails when the selection value is not static (e.g., it is determined elsewhere in the network). Figure 5-3 illustrates this problem, when the network selects either an isosurface or a set of vector glyphs depending on whether the data are scalar or vector. In this case, the selection value for the Switch module cannot be determined before the execution of the graph. Instead, the graph must be evaluated in stages: 1) determine the selection value, 2) determine the necessary input to the Switch module and 3) evaluate the remainder of the graph. Since dynamic inputs may themselves be descended from other non-static inputs (e.g., computed in the network), this process may have to be performed repeatedly.

![Diagram](image)

Figure 5-3. A data-flow network with computed input to a conditional.

**Iterative Re-Execution**

Unlike the simple example in figure 5-1, most real visualization involve some form of iteration. This may either be direct interaction, where the user is adjusting parameters of the visualization and observing their effect on the resulting images, or animation, in which one or more inputs to the network may vary from frame to frame.

In iterative applications, there are often major parts of the network that are unaffected when input parameters are modified. In figure 5-2, if the iso-value input to the Isosurface module is changed, only the affected modules and their descendents need to be executed. The output of Import is not affected by the change. Hence, it can be re-used, which avoids a superfluous access of data on disk. The MapToPlane module also does not need to be executed, since its inputs did not change either.

One way to implement this capability is via a caching mechanism for partial results. Instead of immediately re-executing when its inputs arrive, a module may first determine whether its inputs have changed. If they have not changed, it can simply retrieve its results from the cache. Otherwise, the module re-executes, placing its new result into the cache.

**Data Explorer Approach**

The IBM Visualization Data Explorer (or simply Data Explorer) is a general-purpose software package for data analysis and visualization. It has a client-server architecture and a data-flow-driven execution model (Lucas et al [1992]). Data Explorer has been implemented for Unix workstations from Sun, Silicon Graphics, IBM, Hewlett-Packard, DEC and Data General, and personal computers using OS/2.

**Client-Server Architecture**

The client process in this package is the graphics user interface. It utilizes X Window and the Motif window manager and is implemented in C++. The server process operates as a computational "engine" and is implemented in C. It may reside on the same or different systems than the client. The server is controlled via a data-flow executive, which determines what tasks need to be executed based upon user requests and schedules their execution. The server accepts a well-defined protocol (a scripting language), which is generated by the user interface. The executive can be operated independently of the user interface via that scripting/programming language.

**Uniform Data Model**

One of the design criteria for Data Explorer was adaptability to new applications and data, and the utilization of multiple types of data simultaneously. Another was efficiency for access among the functions that a user might employ. Both of these requirements have been addressed by building the module set on a foundation of an integrated,
discipline-independent data model (Haber et al [1991]). The implementation of this data model describes and provides uniform access services for any data brought into, generated by, or exported from the software for a number of interesting classes of scientific, engineering, and graphics data, which can be described by shape (size and number of dimensions), rank (e.g., scalar, vector, tensor), type (float, integer, byte, etc. or real, complex, quaternion), where the data are located in space (positions), how the locations are (topologically) related to each other (connections), mesh dependency of data (i.e., node or cell center), nodes or cells that may be invalid, user-defined metadata or aggregation (e.g., hierarchies, series, composites, multizone grids). It also supports those entities required for graphics and imaging operations within the context of Data Explorer (e.g., viewing camera, normals for shading, etc.).

All operations on data within Data Explorer, independent of a role in generating pictures, work with shared data structures in memory via an uniform interface that is presented by the data model. This permits the same consistent access to data independent of its underlying grid, type or hierarchical structure(s). To minimize copying and reduce memory utilization, data communication among subsequent operations is accomplished by passing pointers. In addition, sharing of these structures among such operations is supported.

One result of this approach is easy integration of disparate, multiple data sets, a requirement for many visualization problems (e.g., results of observation and simulation, remote sensing from space and ground truth, differing medical imaging modalities, structural analysis, fluid flow and design data). This integration can take place without forcing various conversion or interpolation operations that would corrupt the data. An example of this idea is shown in figure 5-4, where a number of distinct atmospheric cloud parameters are shown simultaneously in a three-dimensional, earth-centered coordinate system.

Visual Programming and Polymorphism

An important consequence of the unified method of data handling is that operations in Data Explorer (modules) are polymorphic, interoperable and appear typeless to the user. This is in contrast to other available implementations, in which each class of supported data is handled more or less independently and is utilized with a separate set of modules.

Consider figures 5 and 6, which show a very simple example of visual programming with Data Explorer. They each contain a screen dump of the Visual Program Editor with a 4-node network. The available modules are shown in various categories on the left. A number of options associated with creating and manipulating visual programs as well as interaction with other aspects of the Data Explorer executive and user interface are available through pull-down menus from the top of the Visual Program Editor. The Import module reads specified data from a file or pipe. The Isosurface module computes surfaces of constant value. In this case, the third input has been specified with the number 4, which means that four surfaces at four equally spaced values over the range of the data will be computed. The AutoColor module computes a linear hue-based color map (blue to red) over the full range of the data. The Image module renders an image from the input it receives and provides tools to interact with the rendered image.

The image in figure 5-5 shows four isosurfaces computed from a three-dimensional stack of CAT scans of a human spine, comprising a regular, rectilinear volume of cubes. The image in figure 5-6 shows four contour lines computed from ultraviolet intensities observed from a spacecraft in a curvilinear, irregular grid of quadrilaterals. The polymorphic nature of the modules allows the same set of tools, and hence, the same visual program to be applied to disparate data sets without intervention by the user. For example, Isosurface computes surfaces from three-dimensional data, lines from two-dimensional data, and points from one-dimensional data, independently of the type of mesh structure or space within which the data are embedded.

Module Toolkit

The aforementioned collection of polymorphic modules in Data Explorer provides various computational tools for the user. They support a number of realization techniques for generating renderable geometry from data. These include color and opacity mapping (e.g., for surface and volume rendering), isosurfaces, histograms, two-dimensional and three-dimensional plotting, surface deformation, etc. for scalar data. For vector data, arrow plots, streamlines, streaklines, etc. are provided. Realizations may be annotated with ribbons, tubes, axes, glyphs, text and display of data locations, meshes and boundaries. Data probing, object picking, arbitrary surface and volume sampling, and arbitrary cutting/mapping planes are supported. Data Explorer supports a number of non-graphical functions such as point-wise mathematical expressions (e.g., arithmetic, transcendental,
boolean, type conversion, etc.), univariate statistics and image processing (e.g., transformation, filter, warp, edge detection, convolution, equalization, blending, FFT, morphological operations, etc.). Field/ vector operations such as divergence, gradient and curl, dot and cross products, etc. are provided. Non-gridded or scattered data may be interpolated to an arbitrary grid or triangulated, depending on the analysis requirements. The length, area or volume of various geometries may also be computed. Tools for data manipulation such as removal of data points, subsetting by position, sub/supersampling, grid construction, mapping, interpolation, re-gridding, transposition, sorting, etc. are available. The data structures that support the data model may also be manipulated at the module level.

Data-Flow Execution in Data Explorer

A number of problems associated with the data-flow execution of graphs produced by visual programming and their potential solutions have been discussed. The execution model of Data Explorer incorporates the two aforementioned approaches (data cache and graph analysis) as well as several others.

Iteration: The Data Cache

As described earlier, efficient execution of visualization programs in an iterative context demands the retention of results of a module. Hence, if the inputs to the module have not changed on a subsequent execution of the graph, the result can be re-used without re-computation. Data Explorer extends this notion by incorporating a cache for all partial results. This cache retains results not only the previous execution of the network, but from all prior executions. This is subject to memory limitations and a least-recently-used cache flushing strategy. Caching may also be explicitly set by user for each output of each module to optimize memory utilization. For example, in figures 5-5 and 5-6, if one only wanted to keep the colored isosurface, then caching could be turned off explicitly for the upstream modules: Isosurface and Import. Using information stored in the cache, one can not only retrieve the results of parts of the network that have not changed from the previous execution, but can also return to previous states efficiently. If a module has executed with a given set of inputs at some time in the past, and one returns to those input settings, re-execution of the module may be avoided by finding the result in the cache.

This capability is particularly useful in conjunction with the Data Explorer Sequencer module. The Sequencer provides Data Explorer with a very simple and flexible animation capability -- an automatic method of managing a frame counter in a graph, which is updated with each new execution based upon a VCR-like interactor. The Sequencer includes buttons for stop, pause, run forward and run backward. It also includes buttons to place it into single-step mode, to cause it to continually loop and to do so in "palindrome" mode. The settings window is used to specify the limits of the sequence of numbers generated.

The first time the Sequencer is "played", it will cause the network to be executed with new values for the Sequencer output. Each execution, which may be time consuming, will result in a new image being generated. These images, which alter all, are simply the result of a rendering module, will be retained in the cache. When the Sequencer is "re-played", the inputs to the network are the same as they were for the first execution. Thus, the result of the execution (the images) will be immediately available from the cache. Hence, Data Explorer provides an automatic mechanism to create real-time animations even when the computation of each frame is slower than real-time. These features are illustrated in figures 5-7 and 5-8.

The value produced by the Sequencer can be used in a number of ways. Figure 5-7 shows how the Sequencer may be used to iterate through a time-dependent data set, causing the visualization to operate on each time step in turn, resulting in an animation showing how the data vary with time. The user may look at a daily sequence of one or more isosurfaces of atmospheric temperature over the earth's southern hemisphere as well as to interactively change the value(s) specified to the Isosurface module. The program also illustrates the support of a subroutine hierarchy. Two of the tools, Projections and WorldMapProjections, are not atomic operations but macros, visual programs that include references to a number of modules and other macros. Alternatively, figure 5-8 shows how the Sequencer can be used to drive the isovalue input to the Isosurface module. Note how the Statistics and Compute modules are used to scale the integer Sequencer values to vary through the actual data range of the data. Polymorphism in the modules enables such a network to operate on any scalar data for any data type primitive.

The idea of storing results in cache can be extended to include inputs to interactors, which correspond to user interface widgets. Interactors used in this fashion are considered data-driven and thus, have state. Both figures
5-7 and 5-8 illustrate data-driven interactors. In figure 5-7, the data range is used to automatically determine default settings for the ScalarList interactor (e.g., minimum and maximum values). In this case, the output of the ScalarList interactor is used to set the values at which isosurfaces should be computed. In figure 5-8, information derived from statistics computed from the data is used to define the maximum number of frames that can be generated by the Sequencer.

Conditional Execution

Data Explorer incorporates two mechanisms to control execution flow through the network: the Switch and Route modules. Switch essentially implements a case construct, which has \( n+1 \) inputs: a selection value (from 1 to \( n \)) and \( n \) inputs of potential selections. Route is the inverse of Switch, with \( n \) outputs and two inputs: a selector value (which may be a list of integers) and an object to be passed to the selected \( n \) outputs.

As described earlier, implementing conditionals in data-flow systems efficiently requires that unnecessary paths through the network be skipped. In the case of Switch, only the required input (if one is selected) is evaluated. Route "kills off" the sub-networks depending on unselected outputs until a Collect module joins a result depending on an unselected output of the Route module with valid results. Figure 5-9 shows how Route can be used to allow the user to select a subset of visualization techniques from a set implemented in the network.

Data Explorer also permits a network to control the execution of aspects of the user interface such as control panels (e.g., open/close), execution mode, image windows (e.g., display mode), etc. internally, that ordinarily would require user action. Coupled with tools for conditional execution, portions of the interface can be made available or hidden based upon user input or computation.

External Asynchronous Data Sources

Many applications of visualization tools call for a direct interface with external data sources, especially ones that generate data to be studied (e.g., a computational simulation). The execution model of Data Explorer provides the framework for real-time visualization of data generated asynchronously by such a process. An external data source is linked into a Data Explorer network by incorporating a communications module, which receives data from the external source, often across a socket, and passes the resulting data object to the module's output. This module (and its descendants) will only run when the external data source has indicated that new data are available.

Data Explorer also provides a mechanism for direct manipulation of the executive (e.g., mode, passing data, error handling, etc.) and the user interface (e.g., window visibility and mode, tracking mouse events, etc.) from an external application. This allows control of Data Explorer from other software and peer-to-peer communications.

Parallelism

The aggregation of all Data Explorer tasks representing a collection of computational "modules" are mapped to a single process with intratask parallelism. Under user control (i.e., within the user interface client), the server may be distributed such that arbitrary portions of a Data Explorer program may be specified to execute within a slave server(s) process operating on another networked system(s). Each of the server process(es) may contain any number of tasks as with the (original) master process. In addition, user-defined modules may be utilized via separate executables from the server process(es), or data may be accepted via a pipe from another process.

In principle, the use of parallelism is an effective way to improve performance. To achieve maximum benefit the system must provide near-linear speed-up as one adds processors. If the software only supports intermodule parallelism, which can be consistent with a distributed execution module, it may be very difficult to achieve efficient parallel execution even on a modest coarse-grain machine for more than a handful of processors. Intramodule parallelism is better suited to exploit such coarse-grain parallelism. Visualization is a complex operation for which the benefits of parallel execution may vary from problem to problem. It is therefore important that a visualization system provides both intermodule and intramodule parallel execution modes, as does Data Explorer. Intermodule parallelism is best suited for problems where two or more computationally intensive operations can execute independently on multiple processors. Linear speed up may not occur in this case due to the speed of communication between nodes for passing data. It is however, the simplest mode of parallelism.
to implement, especially on clusters of workstations and distributed memory multiprocessor systems. Intramodule parallelism is best suited for the exploitation of shared-memory multiprocessor systems applied to problems which have a sequence of one or more computationally intensive operations. By enabling modules to execute in an intramodule manner, computation can be accelerated on multiprocessor machines. This method also obviates the need to pass large volumes of data across a network between multiple processors. On symmetric multiprocessor systems, intramodule parallelism is supported through a simple fork-join shared-memory paradigm.

The Data Explorer executive process uses data domain decomposition and task scheduling. The data domain is partitioned by use of facilities in the data model whereby a single field can be split into a group of smaller spatially local self-contained regions (i.e., composite field). The boundaries of the sub-fields are "grown" to avoid boundary effects in subsequent realization operations. Each parallelized module generates sub-tasks to operate on each partition of the data. This approach also avoids the explicit use of locks, thereby reducing the possibility of a deadlock.

Preserving Explicit State

Some visualization applications require the retention of state from one execution to the next, which cannot be supported within the context of pure data-flow. Consider, for example, the creation of a plot of data values at a point while sequencing through a time series. In effect, the state of the plot is retrieved from the prior execution, appended with the new time-step results, the updated plot is produced and the results are preserved by re-saving the state for the next execution. This capability is provided via two modules: Set, which places an object into the cache, and Get, which retrieves objects from the cache. While Get and Set (cache) provides a simple mechanism for storing such state, they do give rise to the same difficulties outlined above for external data sources - the execution of Get depends on more than its inputs, it also depends on whether the saved object has been changed. Fortunately, the same solution can be utilized. Whenever Set executes, it flags its paired Get to be run on the next execution.

An application of Get and Set is illustrated in figure 5-10, which shows a visual program that access a year's worth of observations of global ozone one day at a time. The Sequencer is used to specify the days to be examined. One Get and Set pair is used, which is highlighted. For each time step, a contour is determined at a specified level, which is colored (blue to red) and labelled by the day of the year. The Get and Set pair holds the accumulated time series of annotated contours, which is appended for each day in the series. The results up to the current day are displayed as a geographic map, which shows the evolution of the contour.

This approach can also be used to support true looping inside a program, which is illustrated in figure 5-11. Atmospheric temperature data are read and sliced by latitude. A loop is then initiated via the ForEachMember module, which computes the mean value for each slice and accumulates it into a series. As with the previous example, Get and Set are used to store the series for each iteration, but inside the loop. When the loop is finished, which is signaled to two Route modules, two images are produced, one showing a pseudo-color image of the temperature data with a map overlay, and the other a plot of the mean temperature in each latitude zone.

Extending Data Flow vs. Alternatives

Song and Golin [1993] and Pang and Alper [1994] have discussed the idea of using a fine-grain decomposition of computation specified through data flow, instead of the more typical coarse-grain implementations. They show promise as a more efficient way to use memory and computational resources for operations that may be done serially on data subsets (e.g., isosurface, rendering). Unfortunately, this approach does not appear to be practical for specification of applications requiring dozens or hundreds of individuals tasks under conditional execution, where saving state may be desirable, parallelized implementation of specific operations, or operations that are not easily decomposed in a fine-grain manner (e.g., streaklines - flow integration across multiple time steps of an unsteady vector field). Conventional data flow with the aforementioned extensions appears to be more effective at addressing such problems. In contrast, this approach may be very useful for supporting simpler visualization and analysis tasks on small machines (i.e., personal computers).

Hibbard et al [1992] in VIS-AD offers the aforementioned virtues of an uniform and extensible data model within a programming environment that provides for sufficient control of operations to build realistic applications. However, it operates at a lower-level than Data Explorer and other data-flow tool kits with only basic
graphics, data structure and computational primitives. Although VIS-AD does provide the facilities available in the extended data-flow architecture of Data Explorer with greater flexibility, it is at a cost of a larger learning curve and greater effort to build complex visualization and analysis operations. But it does provide an easier mechanism than a traditional programming language to develop new algorithms because of its inherent data model and interactive graphics primitives.

Conclusions

Traditional implementations of a data-flow execution model are quite limited when applied to problems of realistic complexity. Fortunately, a number of extensions to such a model are practical ways of resolving these difficulties while still preserving the virtues of "pure data flow". Extensions such as graph evaluation, conditional execution and caching have been embodied in the IBM Visualization Data Explorer software package. Efforts are continuing to enhance the implementation of the execution model in Data Explorer in response to user requirements for data analysis and visualization.

References


Color Figure Captions

5-4. Several Atmospheric Parameters Shown Simultaneously

5-5. Simple Data Explorer Visual Program Applied to a Regular Three-Dimensional Data Set

5-6. Simple Data Explorer Visual Program Applied to an Irregular Two-Dimensional Data Set

5-7. Adjustment of an Input Parameter and Sequencing through a Time Series in a Data Explorer Program

5-8. Use of the Data Explorer Sequencer for Iteration

5-9. Flow Control in a Data Explorer Program Using the Switch and Route Modules

5-10. Preserving State in a Data Explorer Program Using the Get and Set Modules

5-11. Looping in a Data Explorer Program
Figure 5-4
ISCCP C2 Cloud Top Shell
Pseudo-Color = Temperature (180-330K)
Surface Deformation = Pressure (30-1000mb)

Opacity = Optical Depth, Contours = Surface Temperature (180-330K x 5K)
6. Applications of Data Explorer

The previous chapters discussed specific science problems and visualization techniques, which were implemented via Data Explorer, but did not focus on how Data Explorer was used. Therefore, a few simple of two types examples are offered to illustrate the approach that is taken. All of the examples are screen images captured while Data Explorer was executing. The first category is of applications built with Data Explorer that only expose Control Panels and image windows. The second category shows how such applications can be built using the Visual Program Editor and a set of tools that support cartographic warping and other techniques described earlier. These images and the example applications are available via anonymous ftp at ftp.tc.cornell.edu.

**ozone**

Figures 6-1 and 6-2 show the interface for an application built to support the visualization and analysis of total column ozone data from NASA’s Nimbus-7 Total Ozone Mapping Spectrometer. The former shows the southern hemisphere data as a deformed, pseudo-color-mapped surface in an orthographic map projection with a zonal profile plot at 0° longitude. The latter shows the data as a pseudo-colored spherically warped surface with a zonal plot at 60° east longitude after being histogram equalized.

This program provides analysis of daily total column global ozone for the period September 1, 1987 through November 30, 1987. It supports conventional two-dimensional geographic mapping, for which the data are realized as a pseudo-color image optionally overlaid with similarly pseudo-color mapped contours. The user has the ability to adjust the boundaries of a pseudo-color map, which defaults to a non-linear RGB color map and contouring range as well as the increment of the contour lines.

The program supports five different cartographic map projections, which are implemented without interpolation via coordinate warping. The projections are Cylindrical Equidistant, Mollweide, Mercator, northern and southern hemisphere orthographic and spherical. The data are overlaid with conventional maps of world coastlines and political boundaries as well as fiducial lines. The user has the ability to geographically pan and zoom the data being rendered.

From the choice of two-dimensional projection each day of data may be realized as a deformed surface, which is redundantly represented by both the pseudo-color spectrum and height. Below each translucent surface is a world coastline map in magenta with political boundaries corresponding to each hemisphere. The map is from a data base of lines, which has been transformed in a manner similar to that of the ozone data.

This program permits geographic/spherical browsing of daily total column global ozone by carrying the cartographic theme to a three-dimensional continuous surface by performing a cartesian to spherical coordinates transformation. By default the ozone is triply redundantly mapped to height (now radial), color and opacity so that high ozone values are thick, far from the earth and reddish while low ozone values are thin, close to the earth and bluish. However, the capability to select any of these mappings independently is available. Replacing the map is a globe in the center of the ozone “asteroid”, which is created by an appropriately color-mapped topographic data base on a rectilinear grid, which is similarly warped to a smooth or deformed sphere.

Independent of the specific realization techniques, the user can examine the daily data, the difference from one day to the next or a running difference from a base day. On any of these choices of data, the user has the option of applying histogram equalization, various signal processing techniques (e.g., edge detection) or select a longitude value for viewing a zonal profile plot of the ozone.
Geographic Viewing Control Panel: The user may precisely select the latitude and longitude of the geographic viewing centroid. The actual ozone value at that location is reported in the main image and a marker is indicated for that location on the ozone surface.

Realization Control Panel: The user may select from 2, 2-1/2 or 3d realization techniques for these 3d data. The user may control the range of values to be realized. Optionally, a zonal profile plot may be shown. The zonal slice is taken at the longitude of the viewing centroid. The user may choose among several color maps including a linear RGB, non-linear RGB and gray-scale.

Asteroid Control Panel: If 3d (spherical) realization is chosen, the user had the option of showing a smooth or radially deformed spherical shell and whether the color of the shell is constant or pseudo-color. In either case, the opacity is mapped to the ozone value. A viewing width or altitude may be selected as well as whether the globe is smooth or radially deformed.

2 and 2-1/2-Dimensional Control Panel: For flat representations, the user may select from pseudo-color images, contours, image and contours and filled contours. An increment in DU may be chosen for contours. Line contours may be black or pseudo-colored. The actual values may also be indicated. The contour increment option controls the frequency of the labelling. If flat or surface realization techniques are chosen, then the user may select the geographic map projection, the width of the map and whether fiducial lines should be shown. For surfaces, the inclination may be chosen.

Analysis Control Panel: The data to be visualized may be the original data. Optionally, histogram equalization, a Sobel filter or a Gaussian filter may be applied. For "flat" surfaces, the data may be viewed in a smoothed or cell-based fashion. The data are derived from the daily grids. Optionally, diurnal differences or running differences from a specific day may be selected. A threshold value may be entered, below which the corresponding area of the earth’s surface is calculated. This can be used to estimate the areal extent of the ozone depletion region. The result is reported in the main image window.

Rendering Options Control Panel: The user may choose the size of the image, whether to use hardware or software rendering, and if annotation should be shown on the main image window. If hardware rendering is chosen, the user may select the approximation to be used.

**troposphere**

Figure 6-3 shows the interface for an application built to support the visualization of atmospheric dynamics data. Pseudo-colored and opacity-mapped temperature isosurfaces at 194K and 285K are shown with streamlines of wind velocity, pseudo-colored by horizontal wind speed on registered with a gray-scale globe in spherical/geographic coordinates.

This program allows a user to examine gridded, volumetric atmospheric temperature and wind data. The data are at a spatial resolution of 2.5 degrees over the whole earth from 1000 mb to 10 mb. The data are derived from the spacecraft, balloon and aircraft observations, which have been modelled and gridded. They are on a 2.5 degree grid, originally 144 x 73 cells (longitude x latitude). The data are transformed from the original collection of seven stacked two-dimensional grids to a collection of spherical shells. These daily data are for October 1, 1987 through October 10, 1987 at 00:00 GMT. The user may examine either the daily temperature and/or the wind data through a number of different realization techniques. The temperature data may be realized via direct volume rendering, surface extraction, pressure surface or cutting plane with contours, all of which is pseudo-color mapped. One or more values may be chosen for the isosurface. The specific pressure level may be selected. The cutting/mapping plane in this spherical, earth-centered coordinate system is an annulus. A probe is used to select the normal of the annulus, which is marked with an arrow. The wind data may be realized as glyphs, stream-ribbons or streak-ribbons, which are pseudo-color mapped by horizontal wind speed. The ribbons may be optionally twisted by the curl of the velocity field, which is proportional to the wind vorticity. The user may enter specific seed points for the lines. For glyphs, their size is redundantly mapped to horizontal wind speed. Optionally, the user may choose to realize the three-dimensional horizontal wind speed as a volume, pressure surface or isosurfaces.
vortex

Figure 5-4 shows the interface for an application built to support the visual correlation of the aforementioned atmospheric dynamics and ozone data. Southern hemisphere ozone is shown as a pseudo-colored deformed surface registered with vector arrows of 100 mb wind velocity, pseudo-colored by wind speed, and pseudo-colored and contoured 100 mb temperature in an orthographic projection. A simple linear combination of these parameters are shown as a single pseudo-colored deformed surface in the same coordinate system.

This program provides a view of the south polar vortex by comparing global total column ozone with 100 mb temperatures and horizontal winds for September 1, 1987 through November 30, 1987 on a daily basis. Atmospheric temperatures at the level in the atmosphere in which temperature is 100 mb, which is below the tropopause may be realized using the same techniques as the ozone (pseudo-color image, contours or deformed surface). The data range from 180 to 235 K. Above the orthographic plate of temperature is a similar plate of vector glyphs which correspond to the horizontal winds at the 100 mb level. The size of the vector glyphs and the color correspond to wind speed, which ranges from 0 to 80 m/sec. In addition, the scalar techniques used with the ozone and temperature may be optionally applied to the wind speed. The resultant geometries are stacked vertically to permit correlative visual comparisons.

As one sequences through September, coming out of Antarctic winter, the availability of polar ozone data is apparent as well as the formation of the ozone hole. Precursor and correlative signatures are visible in the temperature data and the wind patterns evoke a cyclonic pattern corresponding to the polar vortex. The program supports four different cartographic map projections, which are implemented without interpolation via coordinate warping. The projections are Cylindrical Equidistant, Mollweide, Mercator and northern and southern hemisphere orthographic. If the south pole orthographic projection is selected this correlation corresponding to the vortex is quite clear in each data set.

Viewing Control Panel: The user has the ability to select which of five data sets to display, total column ozone, 100 mb temperature, 100 mb horizontal winds, topography and a map of coastlines and national boundaries. The user can choose the geographic map projection to use. The ozone, temperature and wind may be stacked and registered with the maps or may be viewed separately in separate windows with the coastlines map.

Realization control panel: There are a few options for the realization of each data set. The ozone and the 100 mb temperature can be shown as a flat (i.e., image) or deformed surface, which is pseudo-colored. If flat, then the option of showing contours every 20 DU for ozone and every 5K for temperature is available. The contours may be line or filled. The 100 mb wind speed may be shown in the same way as the ozone and temperature. Alternatively, the wind velocity may be shown as speed-colored vector glyphs.

Model Control Panel: The user may compare the ozone, temperature and wind (speed) data by forming a simple model relating the parameters. This model is a linear combination of the three parameters (i.e., \( A^\text{ozone} + B^\text{temperature} + C^\text{wind} \)). Each of the three parameters are normalized (i.e., the values range between 0 and 1) and the weights for each are between 0 and 1. The weights may be adjusted through separate interactors. The resultant combination may be viewed as a pseudo-colored image, contours or deformed surface.

Building Applications

Figure 6-5 shows a simple visual program that reads one day of TOMS data (import), identifies where "missing" data are (include), applies a linear hue-based pseudo-color map (AutoColor) and renders an image (Image). Figure 6-7 shows the extensions to that program to create an animation (Sequencer) of daily data as a pseudo-colored deformed surface (RubberSheet), which has been warped to south and north pole orthographic projections (Projections), and registered with a coastline and national boundary map (WorldMapProjections).

The cartographic tools that support map projections via coordinate warping are implemented via Data Explorer macros. Figure 6-6 is typical of these tools, Mollweide, which shows the calculation for a Mollweide projection (Compute). Data Explorer uses a convention of defining vectors as \([a.x, a.y, a.z] \) where \( a \) is a vector consisting of three components, \( x, y, \) and \( z \). Ordinarily, the Compute module operates on data. The Mark mod-
ule is used to indicate that subsequent operations are to be applied to another component of a field (i.e., the positions — where data are located in space, which are typically 2-vectors are 3-vectors).

Several of these techniques are combined to build a very simple application in Figure 5-7 (page 3-63), which allows a user to look at a daily animation sequence of one or more isosurfaces of atmospheric temperature in a south pole orthographic map projection. A single day of these data may also be easily shown via direct volume rendering in spherical coordinates registered with a gray-scale globe as illustrated in Figure 6-8 or as isosurfaces in Figure 6-9.
Color Figure Captions

6-1. User interface for a Data Explorer application supporting analysis of Nimbus-7 TOMS data.

6-2. User interface for a Data Explorer application supporting analysis of Nimbus-7 TOMS data showing histogram equalization applied to irregular data.

6-3. User interface for a Data Explorer application supporting visualization of atmospheric dynamics data.

6-4. User interface for a Data Explorer application to study the correlation between Nimbus-7 TOMS and 100 mb temperature and wind data.

6-5. Visual programming applied to Nimbus-7 TOMS data.

6-6. Visual program for constructing deformed polar orthographic surfaces from Nimbus-7 TOMS data.

6-7. Data Explorer cartographic macro for coordinate warping by the Mollweide projection.

6-8. Visual program for constructing direct volume rendering of atmospheric temperature in spherical coordinates registered with a topographic globe.

6-9. Visual program for constructing isosurfaces of atmospheric temperature in spherical coordinates registered with a topographic globe.
Global Stratospheric Ozone for 1987, Day 266
Area below 200 DU = 14130345 sq. km. (2.6% of the Earth).

Total Ozone at Viewing Centroid = 300 DU
Ozone Zonal Profile Plot

Global Total Column Ozone (Dobson Units)

Global Stratospheric Ozone for 1987, Day 266:
Area below 200 DU = 14130345 sq. km. (2.8% of the Earth's surface).

Total Ozone at Viewing Centroid = 587 DU

Analysis Options
- Data Source: Daily
- Data Operations: Histogram Equalization, Smooth Data
- Threshold for Measuring Area (DU): 200

Sequencing Control
- Surface Inclination (Degrees): -45
- Viewing Width (Degrees Longitude): 270.0
- Map Projection: South Pole Orthographic
- Fiducial Lines: No
- Ozone Minimum (DU): 100.0
- Ozone Maximum (DU): 650.0
- Dimensionality: Sphere (2D)
- Pseudo-Color Map: Non-linear RGB
Global Atmosphere (1000 - 100 mb) on October 2, 1987
### Include

**Notation:** Include

<table>
<thead>
<tr>
<th>Inputs:</th>
<th>HIDE</th>
<th>Type</th>
<th>Source</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td></td>
<td>data field</td>
<td>Import</td>
<td>NULL</td>
</tr>
<tr>
<td>min</td>
<td></td>
<td>scalar, vector</td>
<td></td>
<td>-0.1</td>
</tr>
<tr>
<td>max</td>
<td></td>
<td>scalar, vector</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>exclude</td>
<td></td>
<td>flag</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>cull</td>
<td></td>
<td>flag</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outputs:</th>
<th>Type</th>
<th>Destination</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>field</td>
<td>AutoColor</td>
<td>All Results</td>
</tr>
</tbody>
</table>

---

**Rendering Tools:**

- AmbientLight
- Arrange
- AutoCamera
- Camera
- ClipBox
- ClipPlane
- Display
- FaceNormals
- GeoCamera
- GeoLight
- Image
- Light
- Normals
- Overlay
- Render
- Rotate
- Scale
- Shade
- Transform
- Translate
This page intentionally left blank.
Case Study # 4: Examining Earth Sciences Data in Real Time

(Vis5D and VisAD demonstrations for interactively visualizing and steering Earth and space science computations)

Bill Hibbard and Brian Paul
Space Science and Engineering Center
University of Wisconsin- Madison
whibbard@macc.wisc.edu

For more information see the SSEC Visualization Project home page at:

http://www.ssec.wisc.edu/~billh/vis.html
SCIENTISTS SIMULATE AND OBSERVE NATURE

Simulations are formulated as mathematical models, but are implemented as computer algorithms to simulate complex events.

Observations are analyzed in terms of mathematical models, but the volumes of observations dictate that analyses be automated.

Computers have become essential tools to scientists.

PROBLEM: Automated computations are invisible, and the number of operations makes the relation between input and output non-intuitive.

SOLUTION: Make computations visible, and enable scientists to experiment with their algorithms.

Interactive visual exploration of large data sets: decide what to see next based on what you are looking at now.

Physical processes imperfectly understood, and Algorithms are imperfect finite approximations to mathematics. Models and algorithms are improved through experiment.

INTERACTIVITY IS THE KEY

a fast cycle of: question → experiment → feedback

Scientists spend much of their time developing algorithms for simulating physical systems, and for analyzing observations of those systems.

The most important role for scientific visualization is to help scientists understand and improve their algorithms.
SIMULATIONS OF THE EARTH'S ATMOSPHERE AND OCEANS

Store traces of computations in: HISTORY FILES

5-D rectangles of data:
- time history - a sequence of discrete time steps
- multiple fields (e.g. temperature, pressure, moisture, wind)
- 3-D volume - a regular 3-D grid

For example: 80 latitudes by 80 longitudes by 30 vertical levels by 100 time steps by 10 model fields ≈ 200,000,000 values

Simulations 10 or 100 times larger exist

History file does not contain all time steps - changes are too small

THE Vis5D (Visualization of 5-D data sets) SYSTEM

Creates a virtual Earth environment from model history files [1]
- non-immersive virtual reality - the user is outside looking in

Reach into the environment with the mouse to:
- rotate and zoom the 3-D view
- drag 2-D slices through the volume
- place seed points for trajectories

Start, stop and single step animation
Select combinations of fields to depict
Control how fields are depicted:
- iso-level surfaces - control iso-levels
- iso-level lines on planes - drag planes through volume
- colored planes - move planes and control color map
- volume rendering - adjust color and alpha maps

All controls concurrent with animation
Vis5D transforms simulations of the Earth's atmosphere and oceans into an interactive graphical environment.
Vis5D VOLUME RENDERING

Render grid as a series of planes with variable alpha

Chose planes through grid most nearly perpendicular to view direction

Algorithm described in paper "Interactivity is the Key" in 1989 Chapel Hill Workshop on Volume Visualization [2]

Uses tmesh rendering, so co-exists with other graphics primitives

Artifacts:
1. Can see layers near grid edge and near iso-surfaces
2. Aliasing of thin features
3. Abrupt picture changes when planes change (during rotation)

Color Figure 2 was produced by Vis5D's volume rendering mode
it shows vorticity in a simulation of two interacting downbursts
(simulation by John Anderson and Leigh Orf)
The Vis5D volume rendering algorithm slices volumes into parallel planes of polygons, choosing the planes are most nearly perpendicular to the view direction.
USING Vis5D TO COMPARE SATELLITE IMAGES WITH MODEL DATA

Vis5D can texture map satellite images onto the Earth topography. These can be used to visually evaluate the accuracy of model predictions.

Vis5D can accept the following as input:

1. A single image (in SGI rgb format) that remains fixed during animation. For example, this might be an image of land cover.

2. A time sequence that animates with model data. For example, this could be a sequence of GOES satellite images showing cloud cover. These images can be McIDAS areas or can be flat files containing rectangles of pixels.

Color Figure 3 shows a GOES image of clouds being compared to clouds predicted by Bob Aune's CRAS model.

This is a view looking down from the top without perspective, so that the predicted clouds (volume rendered in yellow) and the observed clouds (textured mapped in white) are geographically registered.
MODELS TAKE HOURS, DAYS OR WEEKS OF COMPUTER TIME  
the drive for resolution implies that this will continue  

SO, scientists do not sit and watch models run  

Rather, they visualize history files as a post-process to simulation,  
OR, visualize partial history file to periodically monitor a simulation  

No meaningful way for user to tweak parameters as simulation runs  
simulations are unlike computations that converge on an answer  
tweaks are non-algorithmic  

An experiment is not continuous interactive steering  

Numerical experiment cycle is:  
run model - find and diagnose problem - fix problem - re-run model  

IMPORTING DATA INTO Vis5D  

Skeleton data conversion programs (C and Fortran)  
includes logic to write Vis5D files  
commented spot: read your data here, for time step number it  
and field number ip, into the G array  

user adjusts:  
# of time steps, and their dates and times  
# of fields, and their names  
size and spatial extents of 3-D grid  

Most modelers invent their own file formats  

Use of standards is beginning:  GRIB, HDF, netCDF  
Vis5D import programs for these exist/in the works  

Vis5D also includes skeleton import programs for  
Earth topography and map outlines
Vis5D MAXIMIZES DATA SET SIZE BY COMPACTING DATA

Grid values scaled linearly to 1-byte integers
(users have the option of using 2-byte integers or 4-byte floats)

Values sampled to $A + B \cdot n$ for integers $N \leq n \leq N + 254$

A and B vary between different fields and time steps
to optimize use of 255 scaled values
i.e., B chosen so that all 255 levels are used for a field

N varies between different vertical levels, BUT
A and B do not vary between levels

Otherwise the same value would sample to different values
at different levels, causing artificial contours

256th scaled value indicates missing data
important for sensed data, like volume radar data

DATA SET SIZE VERSUS WORKSTATION MEMORY

Vis5D manages its own memory allocation
paging delays can be very confusing to some users

Vis5D configuration sets limit on memory use
determines whether to use virtual memory
but user can override limit

Without virtual memory,
Vis5D needs about 2.5 bytes per grid value

With virtual memory,
about 1 byte per grid value gives pretty smooth animation

Graphics primitives are also compacted:
vertex coordinates to 2-byte integers
normal components to 1-byte integers
DISTRIBUTED Vis5D FOR DATA SETS TOO LARGE FOR WORKSTATIONS
  a graphics client running in a workstation
  a data server running in a supercomputer
  a high speed network connection [3, 7]

Grid data are stored on fast supercomputer disks
Graphics primitives calculated on-the-fly
Graphics sent to the workstation for rendering

Very expensive, if you have to pay for the supercomputer and network resources

Response times are very bursty (e.g., 15 seconds of dead time)
  if the supercomputer is time sharing

But for very large data sets, it may be the only way
Distributed Vis5D is organized as a queuing network in order to maximize resource use through parallelism.

The multiple "compute and send graphical primitive" boxes exploit parallelism in the server (e.g., Cray YMP) and the multiple "build graphical object" boxes exploit parallelism on the client (e.g., SGI ONYX).

As far as possible, computation, rendering, disk access, and communications are all asynchronous and parallel. The user interface reflects this, providing faster response to controls that can be performed locally in the client (e.g., rotation) than control that must use the server (e.g., change the iso-level of an iso-surface).
Demonstration Using Vis5D at the IEEE Supercomputing 95 Conference [7]:
1. A very large coupled ocean-atmosphere global coupled climate simulation over a 100 year time span
2. Data set resided on an IBM SP-2 server at Aronne National Lab in Illinois
3. Vis5D ran in CAVE virtual reality client system at the conference in San Diego
4. Client and Server connected by OC-3 I-WAY network
5. Data set partitioned into one-year sections that were moved from server to client as needed for visualization - time delay between 25 and 30 seconds
6. Most user interactions do not require network data transfer
7. User interactions asynchronous to network data transfers

Network transactions usually have significant delays because of shared access to network and server.

Large grained network transactions reduce the number of user interactions that require network transactions - a few large delays are preferable to many moderate delays.

Vis5D AND VIRTUAL REALITY

Vis5D has been adapted to the CAVE virtual reality system for demonstrations in the Siggraph 94 VROOM and the Supercomputing 95 GII Testbed

Vis5D has been adapted to a boom system at NASA / GSFC
Vis5D AS A WORLD WIDE WEB MEDIUM

To serve Vis5D files via the Web:

You can serve Vis5D files from ftp or Web servers, and embed links to these files in your HTML formatted Web pages. Here is an example of an embedded link to a Vis5D file on an ftp server:

<a href="ftp://iris.ssec.wisc.edu/pub/vis5d/LAMPS.v5d.Z">LAMPS</a> (4 MB)

and here is an example of an embedded link to a Vis5D file on a Web server:

<a href="http://iris.ssec.wisc.edu/SCHL.v5d">SCHL</a> (3.5 MB)

These filenames must have the extension .v5d and may have the additional extension .Z if they are compressed. Note that we have added the sizes of the files in parentheses after the links, so that people will know what size of data transfer they are requesting. In order to serve Vis5D files from your Web server, you need to add the following line to your resource configuration file (often named srm.conf):

AddType application/vis5d v5d

To view Vis5D files via the Web:

In order to use Vis5D to view these embedded links, you need to install Vis5D on your workstation and you need to tell your Web browser to invoke Vis5D whenever it encounters a link to file with the filename extension .v5d. You can obtain binary executable versions of Vis5D suitable for use as Web viewers through our Web page at:

http://www.ssec.wisc.edu/~billh/view5d.html

Follow the instructions on this page for downloading and uncompressing the viewer appropriate for your workstation. Then add the following line to the .mime.types file in your home directory:

application/vis5d v5d

and add the following line to the .mailcap file in your home directory:

application/vis5d: /DIR/WHERE/Vis5D/IS/vis5d %s -path /DIR/WHERE/Vis5D/IS

If there is no .mime.types or .mailcap file in your home directory, just create them using a text editor. Note that "/DIR/WHERE/Vis5D/IS" stands for the name of the directory where you installed Vis5D.

For more information see the Web page at:

http://www.ssec.wisc.edu/~billh/view5d.html
Vis5D ENABLES USERS TO CALCULATE DERIVED DIAGNOSTIC FIELDS

For example: to calculate vorticity or divergence from wind fields

Dynamic linking to user-written C or Fortran via TCP sockets
can modify function, re-compile, re-link and re-run dynamically

Calculate 3-D grids for a new field, from 3-D grids for existing fields
if available, from floating point grid values - for accuracy
otherwise from compacted grid values

Use real programming languages (C and Fortran)
generality to compute any function (unlike analysis GUI)
link to users' existing code
speed of compiled languages for interactivity

Vis5D can provide a very simple user interface for visualization
since GUI does not address analysis
Vis5D AS A VISUALIZATION SUBSYSTEM OF OTHER SYSTEMS

Starting with Version 4.2, there is an application programming interface (API) between Vis5D and its user interface. This makes it possible for system developers to use Vis5D as a visualization subsystem of their systems.

The API currently supports:

1. Vis5D’s standard graphical user interface.
2. A scripting language that enables Vis5D to be used as an automated animation production system.
3. The use of Vis5D for 3-D graphics by NASA / GSFC's Interactive Image Spread Sheet (IISS).

Developers of several other systems are planning to use Vis5D as a subsystem via its API. These include interactive data visualization and analysis systems, and interactive environmental simulation systems.
ANALYSES OF ENVIRONMENTAL OBSERVATIONS

PROBLEM: adapt to the variety of data organizations used in data analysis algorithms

Data model issues:
1. types of primitive values (float, int, char, etc.)
2. how values are aggregated
   - in C these are array, structure, pointer
   - in Viz systems, may be image, field, geometry, color map
3. metadata - how data relate to nature
   - when, where, how, how accurate, missing?

Extensive data model: list of particular aggregates
   application must adapt to data model

Intensive data model: rules for building aggregates
   data model can adapt to application

THE VisAD (Visualization for Algorithm Development) SYSTEM

Defines an intensive data model

Three data structuring rules:
1. Scalar type
   - application-specific name (time, temperature)
   - primitive type (int, real, real2d, real3d, string)

2. Tuple of values of other types
   - like a C structure

3. Array of values of another type,
   - indexed by values of a third type - index must be a scalar

Can adapt to application needs,
   building complex data types as hierarchies of arrays and tuples
   these are the data types of the VisAD programming language
Examples of data types that users can define in the VisAD data model, a diagram of VisAD's voxel-based display model, and examples of the scalar mappings that VisAD users define to control how data are depicted in the display model.
COLOR IMAGES OF THE GOES_SEQUENCE DATA TYPE

Color Figure 4 shows a data object of type goes_sequence (defined in the preceding diagram) displayed in four different frames of reference.

The top right window is defined by the mappings:

- map lat_lon to xz_plane;
- map ir to y_axis; /* ir radiance defines a terrain surface */
- map vis to color; /* vis radiance colors the terrain */
- map region to selector; /* all sixteen regions are selected for display */
- map time to animation; /* these terrains can be animated */

The top left window:

- map ir to x; /* the mappings of ir, vis and variance create a 3-D scatter diagram - note lat_lon is not mapped */
- map vis to y;
- map variance z;
- map texture to color; /* it is a colored 3-D scatter diagram */
- map region to selector; /* only one region is selected for display */

The bottom right window:

- map lat_lon to xz_plane; /* only pixels whose ir radiances fall in the selected range are visible */
- map ir to selector;
- map vis to color;
- map time to y_axis; /* time sequence of 4 images stacked up along the y axis */

The bottom left window:

- map lat_lon to xz_plane;
- map ir to color; /* ir radiances mapped to red shades */
- map vis to color; /* vis radiances mapped to blue-green shades */

Interactively Visualizing and Steering Computations - Hibbard and Paul - page 4-18
VisAD PROGRAMMING LANGUAGE
   an interpreted language similar to C

Expresses high-level science algorithms
   can play a role similar to data flow diagrams or spread sheets

But can support low-level operations and complex control flow

Dynamic linking via TCP sockets to external functions
   written in C or Fortran, for efficiency

No need for graphics logic in this language
   any data object can be displayed with point and click

User defines display frame of reference
   set of mappings from scalar types into display model
   displays of complex types derived from mappings of their
      scalar components
BBE BUBBLE SORT ALGORITHM IN VisAD PROGRAMMING LANGUAGE

a simple algorithm for an example

type temperature = real;
type time = real;
type sortdata = array [time] of temperature;

bubble_sort(sortdata a; time n;) {
    time i, j;
    temperature b;
    for (i=n; i>1; i=i-1) {
        for (j=1; j<i; j=j+1)) {
            if (a[j-1] > a[j]) {
                b = a[j];
                a[j] = a[j-1];
                a[j-1] = b;
            }
        }
    }

Display frame of reference:
    map time to x_axis;
    map temperature to y_axis;

Color Figure 5 is a snapshot of VisAD applied to this bubble sort algorithm
the text window on the left contains the program
the window on the right contains the displays of algorithm data objects
the small text window on top is used to edit display mappings
VISUAL EXPERIMENTS WITH ALGORITHMS

edit program

start and stop execution, single step
set breakpoints (point and click in program text)

display any combination of data objects
edit display mappings
    can control projection, scaling and sampling of mapping
display objects in multiple sets of mappings
interactive display controls:
    rotate and zoom scalars mapped to space
    animate scalars mapped to animation
    adjust color map for scalars mapped to color
    select ranges for scalars mapped to selector
    adjust iso-levels and intervals for scalars mapped to contour

COMPUTATIONAL STEERING

intrinsic slider function:
    value = slider(name, low, high, default)

for example:
    bin_size = slider("histogram_bin_size", 1.0, 100.0, 10.0);
    this creates a slider icon
    every time this statement is executed, function returns value user sets on slider icon
    enables interactive selection of histogram bin size
Diagram of VisAD organization
METADATA

1. names for values
   every primitive value has a scalar type name

2. sampling information
   every value is taken from a finite sampling
   these finite samplings are documented in objects
   e.g., a satellite image is a finite sampling of a continuous radiance field

3. missing data indicators
   any value or sub-object may take the *missing* value

sampling information and missing data indicators are built into
VisAD's internal representation of data objects

sampling (resolution) information is used to optimize object storage

IMAGE PROCESSING EXAMPLE OF THE ROLE OF METADATA IN
PROGRAMMING LANGUAGE SEMANTICS:

```plaintext
   type ir_radiance = real;
   type latitude_longitude = real2d;
   type image = array [latitude_longitude] of ir_radiance;

   image goes_diff, goes_east, goes_west;
   latitude_longitude loc;

   sample(goes_diff) = goes_east;
   foreach (loc in goes_east) {
       goes_diff[loc] = goes_east[loc] - goes_west[loc];
   }

OR, AS A VECTOR OPERATION:
   goes_diff = goes_east - goes_west;

If loc is outside goes_west region, goes_west[loc] = missing

Arithmetic operations are missing if any operand is missing
```
CLOUD CENSUS ALGORITHM EXAMPLE
percentage of cumulus cloud cover as a function of time-of-day and location
(cloud census algorithm by Bob Rabin)

Data types:
- type latitude_longitude = real2d;
- type cumulus_percent = real;
- type topography = real;
- type outline = real;
- type time = real;

- type cumulus_census =
  array [time] of
  array [latitude_longitude] of
  structure {
    .cc_percent = cumulus_percent;
    .cc_topo = topography;
    .cc_outline = outline;
  }

Color Figure 6 shows the correlation of topography with cumulus cloud formation:

- map time to animation;
- map latitude_longitude to xz_plane;
- map topography to y_axis;
- map outline to color;
- map cumulus_percent to color;

Animate this to see the daily cycle of cumulus cloud formation

Color Figure 7 shows the time series as a stack of images:

- map time to y_axis;
- map latitude_longitude to xz_plane;
- map topography to selector;
- map cumulus_percent to color;
ANALYSIS OF SSMI DATA FROM POLAR ORBITING SATELLITE
(data analysis by Mike Botts)

Data types:

```plaintext
type peel = real2d;
type lat = real;
type lon = real;
type time = real;
type radiance = real;
type x = real;
type y = real;
type z = real;
type image = array [peel] of
  structure {
    .im_lat = lat;
    .im_lon = lon;
    .im_time = time;
    .im_radiance = radiance;
    .im_x = x;
    .im_y = y;
    .im_z = z;
  }
```

Color Figure 8 shows many orbits of the satellite as narrow strips over a Mercator map
(the strips wind upward because time is mapped to the y axis):

```plaintext
map lat to x_axis;
map lon to z_axis;
map time to y_axis;
map radiance to color;
```

Color Figure 9 shows this same data wrapped onto the globe:

```plaintext
map x to x_axis;  /* x, y and z and computed from lat and lon */
map y to y_axis;
map z to z_axis;
```
ALGORITHM FOR INTEGRATING THE LORENZ EQUATIONS

the Lorenz equations simulate a very simple (i.e., the phase space has three dimensions) 2-D cell of atmosphere with chaotic dynamics (i.e., turbulence) (developed for Roland Stull's class on Atmospheric Turbulence)

Data types:

```plaintext
type loc = real2d;
type temperature = real;
type stream_func = real;
type phase_x = real;
type phase_y = real;
type phase_z = real;
type time = real;

type atmos_temp = array [loc] of temperature;
type atmos_stream = array [loc] of stream_func;
type phase_point =
    structure {
        .point_x = phase_x;
        .point_y = phase_y;
        .point_z = phase_z;
    }
type phase_history = array [time] of phase_point;
```
Color Figure 10 shows the integration of the Lorenz equations displayed in three different frames of reference

the right window shows temperature (red is warm and blue is cool in a data object of type atmos_temp) and wind streamlines (contour lines of the stream function in a data object of type atmos_stream) in a 2-D atmosphere cell:

```
map loc to xy_plane;
map temperature to color;
map stream_func to contour;
```

the lower left window shows the strange attractor in 3-D phase space as a "red" data object of type phase_history (overlaid by a single "blue" point that is the display of a data object of type phase_point - it shows the phase space state corresponding to the cell of atmosphere shown in the right window):

```
map phase_x to x_axis;
map phase_y to y_axis;
map phase_z to z_axis;
```

the upper left window shows apparently random (i.e., chaotic) alternations in time between the two lobes of the phase-space attractor (the same data object of type phase_history shown in the lower left window):

```
map phase_x to x_axis;
map phase_y to y_axis;
map time to z_axis;
```
How to get Vis5D and VisAD by anonymous ftp

Vis5D is our system for interactive visualization of simulations of the Earth's atmosphere and oceans. Vis5D is free and available by anonymous ftp over the Internet. To get it, do the following:

% ftp iris.ssec.wisc.edu
   (or % ftp 144.92.108.63)
login: anonymous
password: myname@mylocation
ftp> cd pub/vis5d
ftp> ascii
ftp> get README
ftp> bye

See Section 2 of the README file for complete installation instructions.

VisAD is our system for interactively visualizing and steering scientific computations. VisAD is free and available by anonymous ftp over the Internet. To get it, do the following:

% ftp iris.ssec.wisc.edu
   (or % ftp 144.92.108.63)
login: anonymous
password: myname@mylocation
ftp> cd pub/visad
ftp> ascii
ftp> get README
ftp> bye

See Section 2 of the README file for complete installation instructions.

For more information via the World Wide Web, see our home page at:

http://www.ssec.wisc.edu/~billh/vis.html
REFERENCES


VIDEOS OF LIVE DEMONSTRATIONS (these are all unedited recordings of live workstation sessions):

Vis5D: Siggraph Video Review numbers 43, 61, 74, 82, 105 and 114

VisAD: Siggraph Video Review number 108, IEEE Visualization '92 Video Proceedings
Color Figure 2 / Hibbard & Paul
/* sort.v */

type big = real, sample = linear_set (0.0, 1.0);
type temperature = real, sample = linear_set (0.0, 1.0);
type time = real, sample = linear_set (0.0, 1.0);
type sortdata = array [time] of temperature;

sort(sortdata @ time n) {
    for (i = 1: n) {
        temperature i;
        for (j = 1: i) {
            if (temperature j > temperature i) {
                big i = j;
                temperature j = i;
            }
        }
    }
}

Color Figure 5
Hibbard & Paul