Fall 1991

AVS Network News

The Quarterly Magazine of the International AVS Center

Vol. 1 No. 1

Contents

Cityscape Visualization of Telecommunications Network Performance Data Using AVS 3

Module Submission Criteria 5

New Khoros Modules for AVS 6

A First Experience with AVS 11

Obtaining Modules 12

Using AVS at EPA's Scientific Visualization Center 13

AVS Tips and Tricks 14

What Is AVS? 16

Next-Generation User Interface 19

AVS '93 Announcement 20

In Touch with IAC

avs@ncsc.org
Email address for sending questions to the International AVS Center staff or articles for reprint in ***AVS Network News***.

avs93@ncsc.org
Email address for AVS '93 information.

avsemail@ncsc.org
Email address for an automated reply containing information on the International AVS User Group, current AVS module catalog, README file on the International AVS Center, and use of standard email to obtain modules.

avsorder@ncsc.org
Email address for ordering AVS module source code.

listserv@avsusers.ncsc.org
Email address for sending any message, with HELP as the message body, for an automated reply detailing the International AVS Center's various mailing lists and subscription procedures.

avs.ncsc.org
Anonymous ftp site address of the International AVS Center.

comp.graphics.avs
Newsgroup of AVS users' daily postings, also available by using **listserv**.

WHAT_IS_GOPHER
WHAT_IS_WAIS
Files on the International AVS Center's ftp site (in the **avs_readme** subdirectory) for information on other tools for accessing the ftp site.

AVS Network News

The International AVS Center serves as a catalyst for expanding the AVS user base and for increasing AVS functionality by fostering discipline-specific module development and new AVS uses. Located at the North Carolina Supercomputing Center, the worldwide clearinghouse collects, ports, and distributes user-contributed, public-domain modules and acts as liaison between users and vendors.

AVS Network News is the Center's quarterly magazine serving users and vendors with updates on AVS programs and their use, users' color images and editorial viewpoints, and product information postings from vendors and independent software developers. For an annual subscription, send a $12 check or money order ($22 if outside the United States), payable to the International AVS Center, to Post Office Box 12889, 3021 Cornwallis Road, Research Triangle Park, NC 27709-2889.

David Bennett, AVS Project Leader, **avs@ncsc.org**
Terry Myerson, AVS Specialist, **avs@ncsc.org**
Steve Thorpe, AVS Specialist, **avs@ncsc.org**
Dianne Reid, Administration Assistant, **dianne@ncsc.org**
Joel Page, Contributing Editor, **page@mcnc.org**
Fran Wise, Design and Layout Specialist, **wise@mcnc.org**

For more information, contact
International AVS Center
Post Office Box 12889
3021 Cornwallis Road
Research Triangle Park, NC 27709-2889
Telephone 919-248-1100
Fax 919-248-1101

Cityscape Visualization of Telecommunications Network Performance Data Using AVS

Ian Curington, Advanced Visual Systems Inc.
Graham Walker and Steven Whalley, BT Laboratories

Abstract
In this article, we discuss the potential benefits of data visualization in the planning and performance assessment of advanced telecommunications networks. A 3-D cityscape is implemented using AVS, and two example applications are presented — error rate monitoring in digital transmission systems and presentation of network quality-of-service data.

Introduction

The value of visualization techniques for the exploration, analysis, and presentation of large datasets is well established in many diverse disciplines ranging from computational fluid dynamics to molecular chemistry and archaeology. Although widely used at BT Laboratories to support research and development work in the physical sciences, few applications have been reported with direct relevance to telecommunications, despite a relative abundance of data in a business that primarily is concerned with the transfer of information. In this study, we use AVS to demonstrate the potential for visualization to contribute to improved management of future telecommunications networks.

The design and management of telecommunications networks is an area in which the quantity of data on which decisions must be based can be almost overwhelming. On the one hand, more detailed and sophisticated network models are being developed to assist with the planning of future networks, while, at the same time, more complex transmission and switching equipment provides greater freedom and flexibility in current network reconfiguration and an ever-increasing volume of raw data on the performance of individual links and nodes.

Optimizing overall network performance requires efficient use of data from all these sources, with a fluid interchange of information and ideas between traditionally distinct areas such as network planning and day-to-day management. It seems likely that visualization can contribute to the efficient extraction and exchange of information, and we suggest the cityscape as a potentially valuable technique [1].

The Cityscape Module

In the cityscape visualization, scalar values in a 2-D field are represented as blocks or buildings on a uniform grid (Figure 1). The height of each block is proportional to the value of the field element represented, forming a 3-D bar chart. In both of our examples, the color of the blocks is scaled to their height, but, depending on the application, it is suggested that color could be linked to a different property of the field. Two additional features of the cityscape demonstrate the ability of an effective visualization to reveal trends and patterns and to highlight the important features of a large dataset. First, the end-wall charts present the minimum, average, and maximum values projected through the cityscape. Second, the semitransparent threshold directs attention to the values that exceed a preset target.

Figure 1

The version of the **city_scape** module that is available from the International AVS Center includes controls to scale the height of the blocks and to set the separation between the cityscape and the end-wall projections, the width of the blocks, the level of the semitransparent threshold,

and the tick scale. Interaction is a critical aspect of any visualization, and an additional parameter that is often useful to adjust is the transparency of the threshold sheet. Increased transparency can reveal possible patterns in the complexity of the hidden data. In the following sections, we describe two potential telecommunications applications of the cityscape visualization technique.

Transmission System Performance Data

Modern digital transmission systems can provide megabytes of performance data in just a single day, and network managers must monitor this data for system degradations and failures. In a typical application, there might be a target error-rate performance for the system (for example, fewer than one error in every 10 bits transmitted), and remedial action must be taken if the system performance is worse than this threshold level.

The use of a cityscape visualization for performance monitoring of digital transmission systems was proposed by Hill and Hollan of Bellcore [1], and Figure 1 shows a possible implementation using AVS. The input data is a uniform 2-D field of error-rate data measured every day over a two-week period for a number of different transmission systems (A through L). Attention is drawn immediately to the threshold-breaking events, but potentially significant patterns or structures in the subthreshold activity are still visible.

Figure 2

Quality-of-Service Statistics

Figure 2 shows the cityscape visualization used to present network quality-of-service statistics, a critically important set of figures providing a high-level perspective of network performance. These statistics must be monitored closely against performance targets. The visualization in Figure 2 presents the statistics for call failure percentages, broken down into 10 geographical zones (A through J) for each month of the year. (As with the example of digital transmission system error rates, the data is not actual network data but rather is representative of a typical modern telecommunications network.) Also shown in this figure is the simplified interface that is presented to the user to hide the underlying complexity of the AVS Network Editor.

This second example includes several additions to the basic cityscape. These additions extend and enhance the visualization.

- The threshold sheet is no longer a simple plane but rather a staircase demanding improved network performance in successive quarters.
- Upstream data feedback allows the user to select an individual block in the cityscape and obtain exact quantitative information. A simple extension could present an entirely new dataset, allowing the user to explore a hierarchical data structure.
- The user can select data for national, local, or won exchange calls and also can select from the last six years of data.
- To assist with finding patterns or trends, an animation facility is provided to play back a sequence of cityscape frames.

Although user interaction is essential to appreciate fully the benefits of the visualization, some of the features of the data are immediately apparent even from the static image in Figure 2. For example, the end-wall average across the zones shows a steady downward trend across the year, with a mean consistently below the target value. From the end-wall average across the year, it can be seen that performance is broadly similar in all zones, with the exception of zone J, where the figures are markedly worse but show an improvement over the year.

Conclusions
On the basis of these trial visualizations, we conclude that the cityscape is a potentially valuable technique for the presentation of telecommunications network performance statistics. Extensions such as the staircase threshold sheet and upstream data feedback significantly can enhance the value of the visualization for specific applications.

AVS proved to be an excellent platform for implementing the cityscape, although the initial writing of the **city_scape** module benefited from considerable AVS programming expertise. The ability subsequently to customize the module for specific applications is critically important, enabling the final visualization to exploit fully the characteristics of the data.

Finally, it should be noted that the visualizations described in this article are only examples and are intended to illustrate what might be possible. Investigation into the use of such techniques in a real job function is ongoing, and results from such trials will be essential in confirming the potential benefits of data visualization in the planning and management of telecommunications networks.

Reference
[1]W.C. Hill and J.D. Hollan, "Deixis and the Future of Visualization Excellence," *Proc. Visualization '91*, IEEE Computer Science Society Press, pp. 314-320, San Diego, 1991.

Module Submission Criteria

We encourage modules of all types. We are not looking for perfect productizable code but rather code that is used everyday and would be useful to users. Several users have indicated that their code was not good enough or would not be useful to the AVS user community. This is not true.

We have many different levels of users — from the total novice to the experienced professional — and code is needed for all levels. All we require is source code, a Makefile, and either a README file or man page. Data, networks, etc. are strictly optional.

Source file and Makefile — Source code and Makefile must accompany all modules submitted. We encourage extensive commenting and ask that port-specific lines be so commented.

Module description — A detailed description of the module must be included with its submission. This may be in the form of a man page or ASCII text format (as in a README file). Module description is one of the most important criteria for submission. Please note if multiple modules are in one source code file. The International AVS Center staff will rewrite the documentation as necessary.

Data — If your module uses a nonstandard data format, you must include an example (not necessarily real) of that data. Explanations of how to read the data should be placed in the required manual or README file.

Miscellaneous — You are requested to provide sample network and scripts when appropriate.

To submit modules, ftp to **avs.ncsc.org** and cd to the **SUBMIT** directory. You will be prompted in how to create a directory. The directory itself will be invisible so that others may not inadvertently copy over your files. When you are in your directory, use standard ftp protocol to submit your modules and associated files. Information on using ftp for basic submission and retrieval can be obtained by sending a request to **avs@ncsc.org**.

You also can submit modules and associated files by email to **avs@ncsc.org** and by tape (tape will be returned if requested) to the International AVS Center, Post Office Box 12889, 3021 Cornwallis Road, Research Triangle Park, NC 27709-2889.

New Khoros Modules for AVS

Terry Myerson and Dianne Reid, International AVS Center

*AVS network with **vrotate** and **vresize** modules converted from Khoros*

A sample output image from either the AVS or Khoros networks

The corresponding network in the Khoros environment

An exciting new development for AVS is currently under way at the International AVS Center (IAC). On October 1, 1992, a complete collection of 229 Khoros-derived AVS modules was placed on the IAC ftp site. As time goes by, this port will nicely complement an ever expanding visualization package that continues to create new opportunities.

Khoros, like AVS, is a dataflow-oriented application. Prior to this port, Khoros and AVS concentrated on two mutually exclusive areas. Khoros focused on 1-D digital signal processing (DSP) and 2-D image processing (IP), while AVS focused on 3-D volume visualization. In the near future, the Khoros group at the University of New Mexico will release Khoros Version 2.0, which will support volume visualization. With the AVS port of 229 Khoros DSP and IP routines, AVS now can support the 1-D and 2-D realms quite well.

For this AVS port, the code of Khoros Version 1.0 Patch 5 was used, so, in most respects, the algorithm code in the new AVS modules is well debugged. However, for the immediate short term, these modules still are considered alpha stage. This simply means that the quality of these modules is not yet guaranteed to be 100 percent, but, in order to refine the code, we actively are testing these modules at various sites around the country.

The IAC's goal is to have each and every module tested thoroughly by January 1, 1993. If a module you are using exhibits erratic behavior, a trivial coding error in the wrapper around the Khoros algorithm is likely. Please let the IAC know if a problem like this is encountered so that it can be rectified quickly.

The 229 Khoros-derived modules have been divided into 34 categories (See table.). Each of these categories compiles to form one binary executable that contains several AVS modules (i.e., the **arith_unary** executable actually contains 10 modules including **vabs**, **vclip**, **vconj**, etc.). On a technical coding note, a Khoros library routine exists for each module. The source code to the wrapper around this library routine is in the conglomerate AVS wrapper file.

For example, the module **vabs** has the library routine **lvabs**. The source code to the **vabs** wrapper is in the conglomerate AVS wrapper file, **arith_unary.c**. The source code to the Khoros algorithm, or library routine, would be in the file **lvabs.c.** The motivation for conglomerating multiple source codes into one file was to limit the disk space requirement for the complete Khoros module suite. Separately, the complete suite might have occupied more than 400 Mb for just the compiled binaries.

The Khoros algorithms make reference to functions in the Khoros libraries. Thus, before the AVS-Khoros modules can be compiled for a given platform, these libraries must be ported. Currently, we have only libraries for the DEC and Sun platforms. However, please feel free to port these libraries to your own platforms. The Khoros source code is available from **pprg.eece.unm.edu**. Likewise, the Khoros includes files that also need to be installed on a host system before the new AVS-Khoros modules will compile.

Some platform's compilers are not ANSI C compatible, as is the case on the Sun. In these instances, the gcc compiler must be used. If the gcc compiler is not installed on your platform, it is available through anonymous ftp to **prep.ai.mit.edu**.

This should be an exciting new development for AVS, as all those familiar with the capabilities of Khoros will know. The extensive image processing and digital signal processing implements available with Khoros should prove to be valuable tools for AVS. The new AVS-Khoros modules are available through anonymous ftp to the IAC in the **~ftp/avs_modules/khoros** directory. This port was accomplished through the efforts of IAC's Shawn Mehan, Terry Myerson, and Steve Thorpe and William Ivey of the North Carolina Supercomputing Center.

The current testing evaluation period for these modules is being coordinated by IAC's Dianne Reid and Mehan. The beta testers, to whom we are most grateful, are Ion Barosan of Eindhoven University of Technology, Dr. Loren Buhle Jr. of the University of Pennsylvania School of Medicine, Peter Highnam of Schlumberger Laboratory for Computer Science, Charles J. Williams III of the Naval Research Laboratory, and Wei Wu of the Biology Computational Center at the California Institute of Technology.

DIGITAL SIGNAL PROCESSING LIBRARY

1-D Frequency Domain Filters (**filter_1Dfreq**)

dfiltbp	Bandpass filter-design function.
dfiltbs	Bandstop filter-design program.
dfilthp	High-pass filter-design function.
dfiltlp	Low-pass IIR filter-design function.
dzresp	Calculates magnitude, power, or phase response of a discrete system.

1-D Time Domain Filters (**filter_1Dtime**)

dfilconv	Converts 1-D filter coefficient from transversal to lattice realization.
dfilter	Filters time-domain signals.
dllatt	Generates an error, a signal estimate, and a PARCOR sequence.
dlsynth	Synthesizes an output sequence.
dltapp	Generates an error, a signal estimate, and a weight sequence.
dmlatt	Generates an error, a signal estimate, and a PARCOR sequence.
dmtapp	Generates an error, a signal estimate, and a weight sequence.

File Information (**info**)

dpeakpick	Finds peaks or troughs in 1-D data vectors.
dstats	Computes 1-D statistics of an input field.
varviff	Takes a field as input and extracts a single data value.
vfileinfo	Prints out the header information that describes input field.
vprdata	Prints field to the screen or to a file in ASCII format.
vstats	Performs statistical calculations.

Create Signals (1-D) (**input_create1D**)

dgexpon	Creates 1-D exponential distributed noise dataset.
dggauss	Creates a 1-D Gaussian noise dataset.
dgpoiss	Creates a 1-D Poisson distributed noise dataset.
dgpwl	Creates a 1-D set of signals.
dgrleigh	Creates a 1-D Rayleigh distributed noise dataset.
dgsin	Creates a 1-D ensemble of signals.
dgsinc	Creates a 1-D ensemble of signals.
dgunif	Creates a 1-D uniform noise dataset.
dgwindow	Creates 1-D data sequence based on user-selected window function.

Linear Operations (**linearop1D**)

dacorr	Generates biased, unbiased, or FFT-based autocorrelation estimate.
dconvo	Generates time domain or FFT-based circular convolution estimate.
dxcorr	Generates biased, unbiased, or FFT-based crosscorrelation estimate.

Signal/Sequence Modification (**modify_seq1D**)

dcpad	Performs a constant value padding of a 1-D data sequence.
dextract	Extracts 1-D subsignals from the input data file.
dinsert	Inserts 1-D vectors from input file (img2) into vectors in img1.
dmirror	Generates a mirror data file.
dnormal	Normalizes the magnitude of the input data.

dscale Rescales data for processing, thus minimizing roundoff errors.
dshift Shifts or rotates the 1-D signals in the field.
dsubsamp Subsamples 1-D vector(s) from the field.
dwindow Multiplies user-selected 1-D data sequence by window function.

Spectral Estimation (**spectest1D**)
darpsd Determines the 1-D autoregressive PSD and phase spectral estimate.
dminorm Implements the minimum norm for spectral estimation.
dmusic Implements the music for spectral estimation.

Transforms (1-D) (**transform1D**)
dfft1d Performs 1-D Fourier Transform on a set of data sequences.
dfht Performs 1-D Fast Hadamard Transform on a signal.
dhart1d Performs 1-D Forward Hartley Transform of a sequence.
difft1d Performs 1-D Inverse Fourier Transform on set of data sequences.
dihart1d Performs 1-D Inverse Hartley Transform of a data sequence.
dmpp Determines magnitude, power, or phase of each 1-D data point.

REMOTE GIS

Surface Calculation and Modification (**surface**)
vcost Computes surface arc length from one pixel to every other pixel.
vdetilt Computes best-fit plane for image, then subtracts it from image.
vgettilt Computes least-squares best-fit plane for an image.
vslope Computes slope and aspect images from input elevation data.
vsurf Takes surface image and produces two output images.
vtilt Adds specified plane to an image to correct for a tilt-in luminance.

Multiband Conversion (**vector**)
vmapdlg Converts from actual geographical attributes to rgb of displayable color.

Warp Images (**warp**)
vbilinco Computes the coefficients for the two bilinear equations.
vexpwarp Alters the spatial properties of an image.
vpolyco Computes a set of bipolynomial coefficients.
vwarp Alters the spatial properties of an image.

FILE FORMATS

No File Formats (**no_format**)
asc2AVS Transforms ASCII data into a field.
raw2AVS Transforms raw data into a field.
AVS2raw Writes only the raw image data present onto **descriptor output_fd**.

Standard File Formats (**remote_gis**)
big2AVS Converts a large image file into a field.
dem2AVS Converts digital elevation map data (USGS format) to a field.
dlg2AVS Converts an optional format digital line graph file to a field.
elas2AVS Converts an ELAS image file format into a field.
elasinfo Reads and prints an ELAS image file format header.
AVS2big Converts a large image file into a field.
vll2utm Converts geographical to universal transverse mercator coordinate.

Standard File Formats (**standard**)
fits2AVS Converts a FITS image into a field.
mat2AVS Reads single matrix from file specified and places it in image.
pbm2AVS Converts a PBM image into a field.
rast2AVS Converts a SUN raster image into a field.
tga2AVS Converts a TGA image into a field.
tiff2AVS Converts a TIFF image into a field.
AVS2fits Converts a field into a FITS field format.
AVS2mat Outputs a matrix in desired format.
AVS2pbm Converts a field into a PBM image.
AVS2rast Converts a field into a SUN raster image.
AVS2tiff Converts a field into a TIFF image.
AVS2xbm Converts a field into an x11 bitmap format.
xbm2AVS Converts an x11 bitmap file into a field.

NUMERICAL ANALYSIS

Matrix Manipulation (**matrix_algebra**)
mcovar Computes maximum likelihood estimate of the covariance matrix.
meigen Computes eigenvectors and eigenvalues of a real or complex square matrix.
minfo Obtains information about a matrix.
minvert Computes the inverse of a real or complex matrix.
mlde Solves a system of constant coefficient linear ordinary differential equations.
mlse Solves a system of linear silseaneous equations.
mlud Computes the LU decomposition of a matrix.
mmult Multiplies two matrices.
msvd Computes the singular-value decomposition of a matrix.
mtrans Transposes the matrices contained in the input image.

IMAGE PROCESSING LIBRARY

Binary Arithmetic Operations (**arith_binary**)
vabsdiff Computes the absolute difference between two images.
vadd Adds two images, pixel by pixel.
vblend Forms composite image by adding X*(img1 data) to (1-X)*(img2 data).
vdiv Divides two images, pixel by pixel.
vmul Multiplies two images, pixel by pixel.
vsub Subtracts two images (img1 = img1 - img2), pixel by pixel.

Logical Operations (**arith_logical**)
vand Logically AND two images, pixel by pixel.
vlshift Shifts the pixel data in an image left s bit positions.
vor Performs a logical OR between two images.

vreplace Replaces pixels of first image with pixels of second image.
vrshift Shifts the pixel data in an image right s bit positions.
vxor Exclusively OR two images, pixel by pixel.

Unary Arithmetic Operations (**arith_unary**)
vabs Computes absolute value of an image.
vclip Limits range of values that gray levels may occupy.
vconj Computes the complex conjugate of the input image.
vexp Computes the base 10 antilogarithm of an image, pixel by pixel.
vfloor Clips the lower levels of the input image to the specified value.
vinvert Inverts the gray levels of a byte type image.
vlog Compliments image by subtracting pixel values from a constant level.
vnormal Normalizes an image.
vnot Compliments an image by subtracting pixel values from a constant level.
voffset Adds a specified offset to each pixel in the input image.
vscale Multiplies an image by a constant.
vsqrt Computes the square root of an image, pixel by pixel.
vsubstit Computes the erosion of a binary image by a structuring element.

Classification/Clustering Algorithms (**classify**)
lrfclass Classifies an image using the Localized Receptive Field (LRF).
lrftrain Trains on an image for the weights used with the LRF.
viso2 Converts an image into vectors and performs iso2 clustering.
vkmeans Converts an image into vectors and performs **kmeans** clustering.
vlabel Performs a labeling in a multiband image or a cluster image.
vmindis A simple minimum-distance classifier.
vqerr Computes RMS quantization error with two images.
vquant Performs N-dimensional vector quantization.
vwmdd A weighted minimum distance detector.

Color Transformations (**convert_color**)
vcltrans Specifies the linear transformation matrices.
vcluv Converts to cieucs Luv color spaces.
vcmtrans Performs linear color space conversions with a 3x3 matrix.
vcrgbhls Converts from RGB to HLS and vice versa.
vcrgbhsv Converts from RGB to HSV and vice versa.
vgamut Takes a three-band image and generates a pseudocolor image.

Data Format Conversion (**convert_data**)
band2vect Reformats data stored in a band type image format.
bandcomb Combines all bands from m images into one image.
bandspt1 Takes band from multiband image and creates image with only that band.
bandspt3 Extracts three data bands from a multiband image.
vctor Performs simple mappings of complex data to real form.
vect2band Rearranges the data stored in vector format to band format.
vect2row Reformats data so that each vector is represented as a row.
vmos2band Converts mosaic image into image where subimage makes up a band.
vrect2pol Converts data file from rectangular to polar coordinate and vice versa.
vrtoc Converts real data to complex data.

Feature Extraction (**feature**)
vfractal Performs a fractal feature extraction of an image.
vpml Estimates the fractal dimension of an image.
vpolygon Creates a vector image resulting from a linear approximation.
vrmatch Analyzes two input images that are supposed to be similar.
vshape Performs a shape analysis on two different types of images.
vspatial Performs a spatial feature extraction on an image.
vstereo Analyzes image produced by **vrmatch** and produces image.
vtexture Extracts texture features from an image.

2-D Frequency Domain Filters (**filter_2Dfreq**)
vblse Performs band-limited surface extrapolation or apodization.
vbpf Performs a band-pass filtering operation.
vbrf Performs a band-reject filtering operation.
vhpf Performs a high-pass filtering operation on an image.
vinverse Restores images in the spatial domain by inverse filtering.
vlpf Performs a low-pass filtering operation on an image.
vsvpsf Performs restoration of space-varying blurs.
vwiener Restores images in the spatial domain by **wiener** filtering.

2-D Spatial Domain Filters (**filter_2Dspatl**)
vconvolve Computes 2-D spatial convolution of two images.
vdiff Performs image differentiation by running 2 NxN weighted operation.
vdrf Optimal filter for edge detection.
vgef First derivative operator for symmetric exponential filter.
vgrad Differentiates an image to perform image sharpening.
vhmed Computes a 2-D median filter.
vqmed Computes a 2-D median filter.
vsdef Second derivative operator for symmetric exponential filter.
vspeckle Reduces the speckle index of an image.

Morphological Operations (**filter_morph**)
vm180 Computes a 180-degree rotation of a morphological kernel.
vmcdilate Computes the dilation of a binary image by a structuring element.
vmclose Computes the closing of an image by a structuring element B.
vmcustom Creates a new morphological kernel via keyboard entry.
vmdilate Computes the dilation of a binary image by a structuring element.
vmerode Computes the erosion of a binary image by a structuring element.
vmopen Computes the opening of an image by a structuring element B.

vmoredge Computes an edge extraction in an xvimage structure.
vmskel Dilates a binary image region by a single pixel.

Geometric Manipulations (**geomanip**)
vexpand Expands the size of the image pointed.
vflip Performs reflection or "flip" operations on an image.
vresize Magnifies or reduces an image by using bilinear interpolation.
vrotate Rotates an image by a specified angle from a specified point.
vshrink Decreases the size of an image.
vtranslat Computes the translation of an xvimage structure by vector (i,j).
vtranspos Computes the transpose of the rows and columns in an image.

Histogram (**histogram**)
enhpatch Equalizes or stretches image contrast.
venhance Creates a new image g(x,y) from input image f(x,y).
vheq Performs histogram equalization.
vhisto Computes the histogram of a gray-level image.
vhsee Outputs a histogram in a VIFF format in displayable form.
vhstr Performs a histogram stretch of the gray levels.
vhxray Enhances an image by using histogram stretch or equalization.

File Information (**info**)
dpeakpick Finds peaks or troughs in 1-D data vectors.
dstats Computes 1-D statistics of an input field.
varviff Takes a field as input and extracts a single data value.
vfileinfo Prints out the header information that describes input field.
vprdata Prints a field to the screen or to a file in ASCII format.
vstats Performs statistical calculations.

Create Images (2-D) (**input_create2D**)
vcustom Creates a custom kernel file in the VIFF format.
vgauss Generates an image that contains one or more Gaussian distributions.
vgbox Fills buffer with gray-level representation of a parallelogram on a CCD sensor.
vgcirc Creates an image of a filled circle.
vgconst Creates an image of dimension rows*columns*bands.
vgfractal Creates a fractal image with a desired fractal dimension.
vggauss Creates a 2-D multiband Gaussian (noise) image.
vgpwl Creates a 2-D piecewise linear image.
vgshot Corrupts an image uniformly with spikes.
vgsin Creates a 2-D image of dimension rows*columns.
vimpulse Creates an image containing one or more unit impulses.
vmarr Creates a convolution filter kernel.

Dither (**output_dither**)
vdith45 Performs 8x8 ordered block dither with major axes on 45-degree angles.
vdither Performs 5x5 ordered dithers.
verrdif Performs dithering operation using four-direction error diffusion.

Print Image Hardcopy (**output_print**)
vlpr Formats a byte image for output on a standard line printer.
vpostscr Formats a byte image for output on a Postscript laser printer.

Segmentation (**segment**)
v1bgamut Generates a one-band image with map of displayable color levels.
vclose Tries to close the boundaries.
vdistance Computes the distance from the nearest boundary point.
vdyth Performs dynamic thresholding on an image.
vgamth Generates a binary image (value or 0).
vgrow Computes the median axis inverse transform.
vmediaxis Computes the medial axis transform.
vthresh Separates binary image (value or 0) by thresholding input image.

Subimage/Subsignal Operations (**subregion**)
vextcent Extracts rectangular image from center of larger image.
vextract Removes rectangular piece from an image to produce a subimage.
vinsert Inserts subimage into the input image to form output image.
vpad Pads a data sequence with a constant.
vsamp Samples image along line, given image and a line through it.

Transforms (2-D) (**transform2D**)
vfft Computes the 2-D Fast Fourier Transform of an image.
vfht Performs a Fast Hadamard Transform on the image.
vmpp Computes various representations of complex image data.

A First Experience with AVS

Ken Flurchick, NCSC Research Institute

Representation of scientific data is difficult. Printing the results is lengthy and not always useful. Scientific visualization is a method to display all the data from a simulation at one time. This allows relations among the data to be discovered — the power of scientific visualization. New insight into the problem and leads for further investigations become apparent, and new science is the result.

How does this actually occur? From a scientist's point of view, a flexible tool is required to manipulate the graphical objects in a visualization, as well as an interface with a wide range of input data formats. Also, the tool needs to provide many different forms of analysis and data manipulation to allow for experimentation to find the right representation.

Figure 1

Along comes AVS. This tool allows for the experimentation of data representation, even by a scientist. AVS has numerous data input forms and a great number of data analysis components. Looks like the right tool. However, this tool is not without its own set of challenges.

Figure 2

The interface is expressed in visualization terms and is formidable. The titles of columns of modules are not all obvious. Input and output are straightforward in meaning; however, mappers and filters are not so straightforward. ("Of course, this module is a mapper!"). For most scientists, this can be a hurdle.

Then there is the sheer number of modules. What do all these modules do? I have found no easy way to move through all this information, except just to go through it. However, there are ways around these hurdles. Working with AVS experts and the International AVS Center provided explanations of the visualization concepts and some of the large number of modules. After a while, all this stuff started to fall into place. ("Yes, this really is a mapper.")

Figure 3

Now I'm ready, but of course none of the modules read my data. The approach, I am told, is to write an AVS module. Module writing for data input was a horrifying prospect. ("Not more programming!") AVS, however, helps with the module generator. This gives the framework and hooks into the AVS kernel and saves the scientist's sanity. Module writing is fun — well, not quite. The potential power of AVS and the range of displays made the effort to understand the visualization concepts and leap the programming hurdles worth the attempt.

After some slow starts and dead ends, I worked with a group of high school students to make an AVS visualization that was not just another teapot spinning around. We computed

the positions of molecules as a function of time. Then we worked with Visualization Specialist Dave Bock of the North Carolina Supercomputing Center (NCSC) to develop an AVS module to read the data from our simulation. Using this module and the versatility of AVS, we could see the results of our simulation.

This transformed the problem from a column of numbers into a visualization that we could manipulate as well as use to investigate variations of the model. An example of this simulation is shown in Figure 1. The frame is from the molecular dynamics simulation using the **load_md** module available from the AVS anonymous ftp site, **avs.ncsc.org**. Following this, I worked with the NC SuperQuest teams to develop other modules with the help of Terry Myerson of the International AVS Center (IAC).

In addition to using AVS for education, I began applying AVS to the science at NCSC. One problem was to show contaminant flow in a cylindrical pipe. In this model, whose data was generated by Dr. Tom Goldring of the NCSC Research Institute, the cylindrical pipe section is filled with water vapor that is scavenged by flowing pure gas through the pipe. The concentration of water vapor is computed, and the data ranges from 0 to 100 parts per billion.

Myerson and I developed a cylindrical pipe geometry module as a separate module to represent the pipe. The cylindrical symmetry of the problem required a different module to read the data as an (r,z) pair and generate a 3-D irregular field for AVS to display. The display used three orthogonal slicers, one isosurface, and two additional modules developed at NCSC and is shown in Figure 2.

An additional feature of AVS is the contributions from the Consortium members. ConvexAVS has modules for manipulating electron densities from electronic structure calculations. The problem was to compute the electronic surface of a cluster of carbon atoms fixed in a plane, for which UNC-Charlotte's Dr. Art Edwards generated data. The calculation was done using MOPAC 6.0 on the CONVEX, which has additional routines to generate data for ConvexAVS.

ConvexAVS has modules to take data from these calculations and create displays that provide quick analysis of these calculations. An example of this is shown in Figure 3. The physical system is a planer cluster, C_54 H_18, and the figure shows the electron density at the highest occupied molecular orbital. The positive and negative charges are depicted in this visualization.

AVS has transformed the process of scientific visualization from a postprocessing procedure involving visualization experts and a lot of time to a near real-time analysis tool for complex problems. The range of possible data manipulations and distributed access makes AVS an important component of computational science. This tool, while not initially easy, is powerful and worth the effort.

Obtaining Modules

Modules may be obtained by three basic methods. The first is standard ftp protocol. When you ftp to **avs.ncsc.org** (128.109.178.23), text will be displayed that will guide you in using the ftp site. Please read the AVS_LICENSE file located in the top-level directory before downloading modules from the site.

Among the directories on the site is **avs_modules**, which has several directories within it:

catalogs/	**mappers/**
data_output/	**khoros/**
data_input/	**filters/**

The catalogs directory contains ASCII and PostScript versions of the latest modules, whereas the other directories contain the modules themselves. Each module subdirectory has many Makefiles, one for each port of the module and its associated files.

The second method for obtaining modules is through standard email. When you send an email message to **avsemail@ncsc.org**, you receive automated replies detailing this procedure. These replies also include a copy of the module catalog, the latest README file about the International AVS Center, and International AVS User Group information.

Finally, if you don't have network access, you may request a tape via mail. Please send a tape and a $5 handling fee ($15 if not in the United States) with your request to the International AVS Center, Post Office Box 12889, 3021 Cornwallis Road, Research Triangle Park, NC 27709-2889. State the hardware configuration you will be using, and we will send you a copy of all of the modules on the ftp site.

Using AVS at EPA's Scientific Visualization Center

EPA Scientific Visualization Center Technical Leader Theresa Rhyne (Unisys Corporation)

The Scientific Visualization Center at the U.S. Environmental Protection Agency (EPA) has used AVS for a number of specific projects. The Scientific Visualization Center is located in Research Triangle Park, NC, and supports the visualization of environment research data on an agency-wide basis. Below is a summary of some of these efforts.

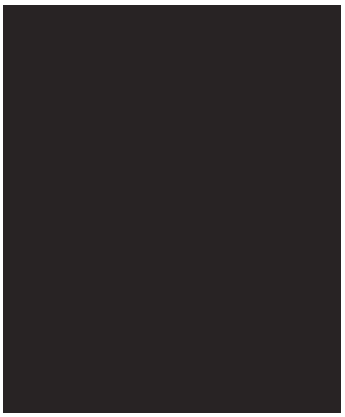
Fluid Flow Around Buildings
Researchers in the Atmospheric Research Exposure and Assessment Laboratory (AREAL) in RTP have been using AVS to examine the behavior of fluid flow around buildings. Using a customized AVS package called AVS-Flow, these researchers have explored 4-D datasets. The datasets contain X, Y, and Z coordinates and change over time.

Fluid Flow in the Lung
An AVS network that assists with the visualization of data modeling fluid flow in the lung has been built. A group of scientists in the Health Effects Research Laboratory at RTP currently are developing this numerical model. Visualization is being used as an aid in troubleshooting numerical simulation difficulties.

Subsurface Visualization
Using data provided by the Great Lakes National Program Office, work is also under way to build an AVS network that allows for the depiction of contaminated sediment zones. This visualization effort is a collaboration between the EPA visualization team and visualizers at Advanced Visual Systems Inc. The project has involved the development of AVS modules that support volumetrics and the creation of isosurfaces from unstructured cell data.

Buffalo River contaminated sediment

Water Flow Studies
In another visualization project for the Great Lakes National Program Office, stream flow data from researchers at the University of California at Santa Barbara were visualized. This required the EPA visualization team to develop an AVS data input module that read data from a noncurvilinear grid. This grid was designed to map the twisting and curving path of a river bed.



Buffalo River concentrations

EPA's Scientific Visualization Center has worked with AVS on various UNIX platforms. These include SUN, IBM RS 6000, DECstation 5000, and Silicon Graphics workstations. We have found that it is possible to transfer AVS networks among these platforms. Technical issues associated with compiling AVS modules on multiple architectures do exist. The EPA visualization team is continuing efforts to understand and clearly define these issues.

The aforementioned work was developed by members of the EPA scientific visualization team in collaboration with environmental researchers throughout the agency. Penny Rheingans (Unisys Corporation) and Mark Bolstad (Sterling Software) built a number of the customized modules. The efforts of the Great Lakes National Program Office involved collaboration with the EPA Scientific Visualization Center and Advanced Visual Systems Inc. Special thanks to Advanced Visual Systems Inc. for its assistance on the subsurface visualization efforts.

AVS Tips and Tricks

Larry Gelberg, Advanced Visual Systems Inc.
Terry Myerson and Steve Thorpe, International AVS Center

This article reviews several AVS technical topics that may prove useful in your scientific visualization endeavors. We thank the many AVS users who have posted to the **comp.graphics.avs** newsgroup and contacted the International AVS Center (IAC) at **avs@ncsc.org** for help. Many of the following questions and answers have been extracted from these archives.

1. *How can I check on the progress of the **tracer** module?*

If you are using the **tracer** module in conjunction with **display_tracker** and it's taking a while to compute, you can find out how far along the tracing computations are. Simply cause the **display_tracker** window to get an expose event (iconify it, uniconify it, or hide it and reshow it). Because the output from **tracer** is in shared memory, all of the currently rendered portion is overwriting the existing buffer. You can use this technique for tracking the progress of the module.

This trick does not work if (a) it's the first time through, (b) you've just changed the output image size from **tracer**, (c) the modules are on different machines, or (d) you use **display_image** or the **image_viewer**.

2. *What is a recommended method to make shaded, ray-traced images of very large datasets?*

Try the **compute_shade** module, contributed to the IAC's module repository by Robert Mazaika and Larry Gelberg of Advanced Visual Systems Inc. This module combines the functions of the **colorizer**, **compute_gradient**, and **gradient_shade** modules into a single memory-efficient module. These modules are used primarily to make shaded ray-traced images. The problem is that they are highly inefficient in terms of memory allocation. The **colorizer** module takes in 1 byte per voxel and outputs 4 bytes per voxel. The **compute_gradient** module takes in 1 byte per voxel and outputs 12 (3 floats), and **gradient_shade** outputs 4 bytes per voxel.

These three modules produce 20 bytes for every input dataset byte. For this reason, some people have experienced problems trying to render ray-castings of large datasets. The tracing code is fairly computationally efficient, but most of the system resources are spent swapping data rather than computing the image. The **compute_shade** module does gradient computation, colorizing, and shading on a per-slice basis. All in all, it takes less time than running the original three modules in sequence. However, it does take longer than running only **gradient_shade** alone. Therefore, it is most useful for extremely large datasets (>100x100x100 voxels) that normally would choke a system's memory.

3. *How do I animate the deformation of UCD data over time?*

The **ucd_offset** module deforms a UCD structure that has a vector (delta-x, delta-y, delta-z) component. If you have only a single deformation computed, you can animate several frames from the nondeformed state to the deformed state by plugging the **animated_float** module into the "offset factor" port of the **ucd_offset** module. If you have several time-steps computed, you need to be able to read in each file individually. This can be done either with a cli script or by using the **animate_file** module contributed to the IAC's module repository by Terry Myerson.

4. *How can I create an isosurface with fewer polygons than the AVS **isosurface** module extracts?*

The **isosurface** module has an Optimize Surface button that reduces the total polygon count by about half. The reason that this is an option is that the **isosurface** module produces polytriangle strips for more efficient rendering. It is quicker just to run through the data, creating polygons and stringing them together into polytriangle strips using "cobwebs" (zero-area triangles), but this increases the total polygon count (and memory). Turning on Optimize Surface creates a graph of the polygons and searches for efficient

polytriangle strips. It takes a little longer, but the polygon count is usually cut almost in half. It's a user-selectable tradeoff between surface computation time and rendering time.

5. *Is it possible to start AVS up with no display in order to generate images without actually displaying to the console?*

Yes, boot up AVS using the command, "avs -nodisplay -cli." Then you can run scripts and other items that produce images and use the **write_image** module (for example) to dump the images to disk. With AVS4, you can save images directly out of the **geometry_viewer** for archiving. With the **animator**, you can dump an image sequence in an encoded format.

6. *How can I write code that is portable across a large number of hardware platforms?*

If you minimize the assumptions made in your code about a platform's hardware and operating system, it will be easier to port AVS modules to different systems. For example, it is wise not to assume you know the size of a particular data type. So when allocating space for an array of 256 floats, allocate 256*sizeof(float) bytes rather than 256*4 bytes to ensure portability. Another helpful hint is to use calls supplied by AVS for memory allocation and deallocation rather than calling a particular operating system's library. Using **alloc_local** and **free_local** is more portable than using **malloc** and **free**. Finally, create your Makefile for your AVS modules using the example found in **/usr/avs/examples/Makefile** as a base from which to work. This file includes **/usr/avs/include/Makeinclude**, where system-dependent definitions are made. Please see the *AVS Developer's Guide* for further guidelines on writing portable code.

7. *What's the easiest way to obtain copies of all of the IAC's public-domain modules?*

When you connect by anonymous ftp to **avs.ncsc.org** (128.109.178.23), you will see a directory called **avs_archives**. Changing to this directory with cd will show the following self-explanatory message.

```
The files in this subdirectory are compressed tar format
archives created from files on this anonymous ftp site at the
IAC. You easily can create a local copy of almost all of the ftp
site with these files, which consist of the following.

ALL_OF_FILTERS.tar.Z     /* AVS filter modules */
ALL_OF_INPUT.tar.Z     /* AVS data input modules */
ALL_OF_KHOROS.tar.Z    /* AVS Khoros modules */
ALL_OF_MAPPERS.tar.Z    /* AVS mapper modules */
ALL_OF_MISC.tar.Z    /* miscellaneous files */
ALL_OF_OUTPUT.tar.Z    /* AVS data output modules */

For example, if I wanted to get all of the IAC's filter
modules, I would take the following steps.

1. Use binary ftp to download ALL_OF_FILTERS.tar.Z
2. uncompress ALL_OF_FILTERS.tar.Z
3. tar -xvf ALL_OF_FILTERS.tar
4. rm ALL_OF_FILTERS.tar
```

8. *How can I use AVS to import and export image formats other than the standard AVS image format?*

Check out Terry Myerson's **read_any_image** and **write_any_image** modules on the IAC's ftp site. These modules are based entirely upon the San Diego Supercomputing Center's Image Tool library. The following image file formats are supported.

eps	encapsulated postscript file
gif	Compuserve graphics image file
hdf	hierarchical data file
icon	Sun icon and cursor file
iff	Sun TAAC image file format
mpnt	Apple Macintosh MacPaint file
pbm	portable bitmap file
pcx	ZSoft IBM PC Paintbrush file
pgm	portable gray map file
pic	PIXAR picture file
pict	Apple Macintosh QuickDraw/PICT file
pix	alias image file
pnm	portable any map file
ppm	portable pixel map file
ps	postscript file
ras	Sun rasterfile
rgb	SGI RGB image file
rla	Wavefront raster image file
rle	Utah run length encoded image file
rpbm	raw portable bitmap file
rpgm	raw portable gray map file
rpnm	raw portable any map file
rppm	raw portable pixel map file
synu	synu image file
tiff	tagged image file
x	AVS X image file
xbm	X11 bitmap file
xwd	X-window system window dump image file

9. *I am trying to visualize molecules and can't get good resolution on my images. The spheres look more like multiple square-sided surfaces than spheres. How do I correct it?*

Check out the Subdivisions widget at the bottom of the **geometry_viewer** control panel. The higher this number, the better looking your spheres will be.

10. *How can I learn about other AVS tips and tricks?*

Consider ordering a videotape of Larry Gelberg's AVS '92 lecture, *Tips and Tricks with AVS*. This can be obtained for a nominal fee from the IAC at the address listed below. Send email to **avs@ncsc.org** for further ordering information. Also, check the frequent postings on the **comp.graphics.avs** newsgroup. If you don't have access to a news feed, postings can be routed to your electronic mailbox from the IAC's **listserv** system. Use anonymous ftp to check the file **avs.ncsc.org** (128.109.178.23): **avs_user_group/avs_user_reg** for instructions on how to subscribe to this convenient service.

11. *What's the easiest way to learn about the IAC?*

Send any electronic mail message to **avsemail@ncsc.org**. You will receive several automated replies, including a README file about the IAC, the latest AVS module catalog, and information about AVS user groups. If you don't have access to electronic mail or prefer a personal reply, contact

International AVS Center
Post Office Box 12889
3021 Cornwallis Road
Research Triangle Park, NC 27709-2889
Telephone 919-248-1100
Fax 919-248-1101
Email **avs@ncsc.org**

What Is AVS?

Complex Information in a New Light
Visual computing — a new visual approach to numerically intensive computing — can help you work more efficiently and insightfully while exploiting the full potential of your graphics hardware. Visual computing offers insight into complex problems and, for the first time ever, gives you interactive control over the exploration and analysis of large datasets in a wide variety of scientific, commercial, and engineering applications. At the heart of this is the Application Visualization System (AVS), the industry standard in platform-independent visual computing environments for application development.

A Broad Spectrum of User Experience and Requirements Met

- Application developers creating interactive visual applications for users in their organizations or for commercial distribution.
- Researchers needing to customize their visualization application to both their own and their coworkers' needs.
- Users simply wanting an easy-to-use turnkey application to visualize their data.

AVS has the depth and flexibility to meet every type of user need. It has been in active use in a wide variety of disciplines, including computational chemistry, computational fluid dynamics, education and research, environmental sciences, medical imaging, oil and gas exploration, GIS and remote sensing, and structural analysis.

The Most Extensive Suite of Data Visualization Techniques Available
AVS includes a comprehensive suite of data visualization and analysis capabilities that make it the most complete visual computing environment available. AVS incorporates traditional visualization tools, such as 2-D plots and graphs and image processing, as well as advanced tools, such as 3-D interactive rendering and volume visualization. And you don't have to be an application developer to take advantage of the sophisticated capabilities of AVS.

If you need a point-and-click visual application to view your data, AVS provides an intuitive, easy-to-learn interface to the most frequently used visualization techniques. These interfaces — or viewers — can help you explore your data immediately — even *before* you begin to tailor AVS to your specific needs.

A Standard Environment for Your Multivendor Network

The system environment for visual computing is typically a combination of workstations, X terminals, and computational servers for a single work group. In order to share results among colleagues, a common base for visual applications is required. Only AVS provides the common base for developing and sharing these critical applications. AVS currently operates on all of the major UNIX workstations and supports the full range of graphics hardware available on these platforms. In addition, AVS support for remote X-window displays means that you can make use of low-cost, 8-bit X terminals and workstations.

Easy-To-Use Visual Application Builder Components

Visual Programming Environment

- Network editor for application construction
- Module generator for code development
- Layout editor for user interface design

Advanced Application Development Support

- Comprehensive visual computing toolset
- Modular and extensible architecture
- Open access to AVS through a command-line interface
- Support for multiple levels of user interface development

Multiplatform Support

- Full support of high-end graphics adapters
- Software renderer for advanced rendering functionality
- Heterogeneous, platform-independent operation
- Distributed computing on networks and supercomputers
- Open, based on standard interfaces (e.g., UNIX, X, PEX, and GL)

Visual Programming — An Intuitive Approach to Building Custom Applications

At the heart of AVS is the network editor, a unique visual programming environment. Visual programming is an intuitive process wherein the network editor helps you graphically connect AVS modules together to build a visualization network. This network becomes an application that can be saved, reused, and modified. You have the freedom to experiment with your data by building custom networks until you are satisfied with the results. The AVS visual programming environment allows you to be productive almost immediately. A combination of complete documentation, online help, example networks, and ready-to-use customizable modules will help you expand your use of AVS at your own pace.

Multiple Platforms Supported from a Single Development Environment

For many developers, the difficulty of multiplatform 3-D graphics application development has limited the use of visual computing. Now, you can use AVS to provide sophisticated interactive graphics applications for your colleagues or clients. Since AVS acts as your interface to high-performance graphics, you don't have to worry about graphics programming. Once your application is up and running, you will find that it takes very little effort to port to other hardware platforms. AVS handles platform-dependent technical issues such as 3-D rendering, user interfaces, file I/O, data transport, and communications. This, in turn, will lower your development and porting costs significantly. Now you can support more platforms more quickly and with less effort.

Shortened Time from Prototype to Application Delivery

For the developer who builds custom applications, AVS provides an unparalleled application development environment. The visual programming environment is designed to accelerate progress from a prototype application to a tuned, deliverable application. In constructing an application, you choose the level of effort at which you create and develop your application, as follows.

- Build networks and design user interfaces without any programming, perhaps as a fast prototyping stage.

- Add your own AVS modules to include algorithms and operations to meet specific application requirements.
- Build an application with the look and feel of your own user interface, but let AVS handle your data processing and graphics display.

Visualization Power Added to Your Current Applications

Although AVS comes with a comprehensive collection of modules, you can write new modules. You can include existing C, C++, or FORTRAN routines in order to incorporate your own algorithms or link AVS to external applications. Built-in software development tools, such as the AVS Module Generator, provide you with a highly productive way to write software. The AVS Module Generator is a complete environment for rapidly creating and maintaining AVS module code in an object-oriented fashion. You spend more time on your research and less time on AVS-specific programming.

The Power of Visual and Distributed Computing Combined

For solutions to large-scale problems, AVS is designed to run in a distributed environment. AVS modules can execute on supercomputers and massively parallel machines, helping to bring visual computing to where your data and simulations reside. When transferring data between machines with incompatible data formats, AVS handles all data format conversion for you.

A Complete Visual Computing Toolset

Data Viewer
- Easy-to-use data visualization
- Point-and-click menu interface
- Comprehensive suite of visualization techniques
- Extensible application with prepackaged networks

Geometry Viewer
- Interactive 3-D geometric display
- Choice of render modes: wireframe, surface
- Surface transparency and reflectance
- Multiple light sources
- 2-D and 3-D texture mapping
- Multiple viewports

Image Viewer
- Real-time image pan and zoom
- Region-of-interest operations
- Flip book animation
- 8-bit and 24-bit support
- Look-up table operations
- Data-resizing operations

Graph Viewer
- Line, bar, scatter, and area plots
- Contour plots
- Variable line type and plot color
- Variable axis range and scale
- Titles, axis labels, and legends

Advanced Visual Techniques
- Streamlines and particle advection
- Image processing
- Volume rendering
- Isosurfaces and slice planes
- Comprehensive finite element data visualizations

Presentation
- Image labeling and annotation
- Postscript and encapsulated postscript output
- AVS Animator for high-quality animations
- Wide range of video and print output options

Volunteers Needed

The International AVS Center (IAC) needs volunteers to assist with preparations for AVS ’94. Individuals are needed to chair lecture tracks, assist with workshop tracks, schedule speakers, and prepare and coordinate panel discussions. A volunteer from the West Coast is needed to work with IAC in preparing for AVS ‘94, and a volunteer from Europe is needed to assist with preparations for AVS ’95. For more information, contact

International AVS Center
Post Office Box 12889
3021 Cornwallis Road
Research Triangle Park, NC 27709-2889
Telephone 919-248-1100
Fax 919-248-1101
Email **avs.ncsc.org**

Next-Generation User Interface

David Bennett, International AVS Center

We recently were contacted by Richard M. Friedhoff, author of *The Second Computer Revolution*, *Visualization*, about creating a group to define the protocols for next-generation user interface. We have completed our initial meeting and are working on funding for the group at this time. The group will include individuals from the field of perception and vision physiology, such as Dr. Richard Gregory and Margaret Livingston, as well as members of the North Carolina Supercomputing Center visualization staff and university and industrial collaborators from the social sciences and other fields.

Livingston is the author of a wide variety of distinguished papers and is one of the most outstanding experts in the submodalities of visual perception — breaking down visual perception into components of motion, color, depth, and form. She was a 1984 Presidential Young Investigator and is now the principle co-investigator of Harvard Professor David Hubel, who was awarded the 1982 Nobel Prize in Medicine for his work in visual physiology. Their Harvard laboratory is recognized widely as one of the most distinguished in the world.

Gregory is the author of *Eye and Brain*, *Oxford Companion to the Mind*, and at least nine other widely admired books on the subject of perception. He is the founder of the academic journal *Perception* and has been on the editorial board of many other academic journals. He is author of more than 75 papers on the subject of perception and cognition and has an equal number of theoretical papers on instrumentation such as microscopy, astronomy, and laboratory instrumentation. Gregory is the founder and director of the Brain and Perception Laboratory at the University of Bristol and currently is developing a cognitive neurobiology program at Oxford University.

Open positions for this committee on next-generation user interface are available. If you would like further information for joining the effort or additional documentation on the project, telephone Project Leader Dave Bock at 919-248-1100. Funding for this first stage is limited and will be used primarily for travel.

Next-generation user interface will be conceived from the standpoint of the perceiver rather than the programmer. Until now, our industry has provided hardware and software to make it possible for the engineer and scientist to visualize data. However, little has been done to show users how to make meaningful images from these remarkable tools. Indeed, we now find that this is the rate-limiting step in many of the areas in which visualization has the greatest potential.

Our present goal is to lower the difficulty threshold for using software to the absolute minimum. At least two basic and related innovations will be incorporated into the new interface: perceptual mapping and question-mediated interfaces. For the first few months, we will work on developing a visualization protocol. Friedhoff and Livingston have suggested that we start with an ordering of the dimensions of perception to the dimensions of data, which will need considerable testing with real-world data.

Gregory and others will work the problem from the broadest terms while others will implement, experiment, and confirm these scientifically based predictions with actual comparisons of visualizations. This is the perceptual mapping stage. We then will work with software engineers to determine the best ways to incorporate ideas into this “intelligent” interface that will guide and nudge users in the appropriate directions. We will keep you updated on our findings. Suggestions or ideas are welcome.

Articles Needed

If you have articles for reprint in ***AVS Network News***, send them to **avs@ncsc.org** or to the International AVS Center, Post Office Box 12889, 3021 Cornwallis Road, Research Triangle Park, NC 27709-2889.

AVS '93 presents

The Magic of Science

2nd Annual International AVS User Group Conference
WALT DISNEY WORLD DOLPHIN
Lake Buena Vista, Florida

May 24-26, 1993

Visualization. Imaging. Computer graphics. Whatever term you use, the Application Visualization System (AVS) is now the worldwide standard for that spectacular industry. And now, AVS '93 puts *The Magic of Science* in your hands with spellbinding visualization power.

AVS '93 blends a masterful mix of workshops, tutorials, and lectures; mesmerizing panels and video reviews; plus entrancing exhibits. It's a wondrous world of visualization capabilities that make the magic of science work for you.

Join us for a spell. You'll see.

TOPICS
Computational chemistry • Computational fluid dynamics • Earth sciences • Education and research • Geographic information systems and remote sensing • Image analysis • Medical imaging • Molecular modeling • Oil and gas exploration • Scientific visualization • Structural analysis

REGISTRATION
To obtain an AVS '93 registration form and detailed program, send your name; company name; complete mailing address, including city, state, country, and zip code; email address; and telephone number to International AVS Center, AVS '93 Conference Management, Post Office Box 12889, 3021 Cornwallis Road, Research Triangle Park, NC 27709-2889, USA, Telephone 919-248-1110, Fax 919-248-1101, Email **avs93@ncsc.org**, Program Chair David Bennett, Telephone 919-248-1182.

SPONSORSHIP
AVS '93 is presented by the International AVS Center and the North Carolina Supercomputing Center, a division of MCNC, in cooperation with the following corporations.

Advanced Visual Systems Inc. • CONVEX Computer Corporation • Digital Equipment Corporation
Hewlett-Packard Company • IBM • Kubota Pacific Company Inc. • SUN Microsystems Inc. • Wavetracer Inc.

Established in 1980 as a private, nonprofit research consortium, MCNC fosters technology research and education to attract and generate technology-based industry for North Carolina. MCNC matches the strengths of major research universities with those of established and emerging industry for economic growth and development in the state. The organization provides advanced capabilities and expertise in microelectronics, communications, and high-performance computing to support research, education, and industry.

The North Carolina Supercomputing Center (NCSC), the supercomputing division of MCNC, makes supercomputing time and related resources available to North Carolina's academic researchers and encourages nonacademic institutions to explore collaborative ventures. NCSC also facilitates research in the areas of computer science, computational science, and scientific visualization.

The International AVS Center, housed at NCSC, is the worldwide clearinghouse for collecting, porting, and distributing public-domain visualization modules. These modules enable dramatic advances in the scope and speed of research in a variety of disciplines. The International AVS Center serves as a catalyst for expanding the AVS user base and for increasing AVS functionality by fostering discipline-specific module development and new AVS uses.