Stanford Exploration Project, Report 79, November 16, 1997, pages 1–15

Visualization of multi-dimensional seismic datasets with CM-AVS

Biondo Biondi and Jos van Trier[1]

keywords: *three-dimensional, visualization, velocity, gain, algorithm, prestack*

ABSTRACT

We have developed a visualizer for large 3-D prestack seismic surveys that takes advantage of the data handling (memory and I/O) and computational capabilities of SEP's CM-5. The visualization application is a collection of AVS and CM/AVS modules and networks. The CM/AVS modules read, window, gain, and slice large multi-dimensional (up to six dimensions) datasets on the CM-5 and then pass the images to a workstation for displaying or 3-D rendering with standard AVS modules. We demonstrate our application by visualizing a 3-D prestack marine survey and a 3-D migrated volume together with a 3-D velocity function.

INTRODUCTION

As SEP begins to address the problems of imaging 3-D prestack data and of estimating 3-D velocity models, it needs tools to analyze large 3-D prestack datasets, either computer generated or recorded in the field. For the visualization of 3-D datasets SEP has been relying on the Movie program developed by Rick Ottolini in (1984) and its more recent X-Window version. Over the years the Movie program has proven to be an invaluable tool for better understanding seismic datasets and for gaining an immediate intuition of the properties of seismic processing operators. However, the Movie program begins to show its age, and it has some important shortcomings for the task of displaying 3-D surveys. First, the program is limited to the display of three-dimensional datasets. Second, it is essentially a batch program in the sense that it displays data already converted to single-byte integers with predetermined clipping and gain factors. Finally, the effectiveness of the program in manipulating large datasets is hampered by the hardware characteristics of workstations (memory size, data handling capabilities, and computation speed). These limitations are important for visualizing 3-D prestack data, which are in the most general case five-dimensional, and may span several gigabytes, even for small "research-size" surveys.

[1]**email:** not available

To overcome Movie's limitations we developed a multi-dimensional visualization application running on SEP's CM-5 and based on the AVS (Application Visualization System) graphics system. We chose the CM-5 as the developing platform because of its data-handling capabilities. In particular, the CM-5 provides fast I/O for reading the data from files on disks that can span multiple gigabytes and it has up to one gigabyte of memory to store large subsets of the survey. In addition, the multiple data-paths to memory that are typical of a parallel computer allow fast access to data stored in memory according to a wide variety of data-access patterns. Finally, the processing capabilities of the CM-5 allow the user to perform some simple image processing (i.e. clipping and gaining) in an interactive fashion, improving the information content of the display.

We developed the visualizer using AVS and CM/AVS (an adaptation of AVS to the CM-5 developed by Thinking Machines) mainly for two reasons. First, AVS enabled us to quickly develop modular interactive applications. More importantly, AVS allows the visualization of many different types of geophysical data integrated in the same display. For example, our data-slicer can be integrated with a geological model (e.g. GOCAD (Mallet, 1993)) and velocity model visualizer. The final goal is to make available to SEP researchers a fully integrated tool for understanding 3-D seismic data and its relationship with geological and velocity models.

The visualization application that we have developed has some limitations. Some of them are intrinsic to the design, and thus it would be difficult to remove them without overhauling the whole application. The most important restriction is that the data to be visualized must be in SEP format, i.e. it must be regularly sampled along all the axes. This restriction does not necessarily limit the handling of marine surveys, but it is definitely a problem for land data. As for performance, presently the bottleneck is the in last step of the visualization process. Once images are extracted from the data and processed on the CM-5, they need be sent to an X-server. The efficiency of this communication step should improve with future developments of the CM-5 software and hardware, but it is presently unsatisfactory.

AVS AND CM/AVS

AVS was originally developed by Ardent, but is currently maintained and distributed by a separate company, AVS Inc. The main advantage of AVS is its modular design: a user can build an interactive application with little effort by putting together a network of predefined modules. Each module performs a simple task, such as displaying a raster image, rendering a 3-D geometry, reading or modifying a parameter, etc. Building a network is easy: AVS comes with an interactive tool that allows the user to grab modules out of a library and connect them on the screen by drawing lines. Each AVS module knows what kind of input data, output data, and parameters it can accept and produce, so there can be no ambiguity or design errors in building a network. Figure **??** shows an example of a network used in our visualization tool. The lines connecting the modules represent data-links between modules; data-links of different types (parameters, fields, colormaps) are indicated by different colors.

Over the years AVS has grown into an extensive toolkit that provides modules for most data visualization problems. As a result, the user is spared the work of coding graphics modules himself, and can concentrate on writing modules that are specific to his problem. Programming in AVS is straightforward with the module autowriter: the user interactively defines input, output, and parameters, the autowriter produces a skeleton program, and the user then writes the core of the module. The basic data structure in AVS is called a field, and most parameters associated with graphics applications (axes information, coordinate data, etc.) are part of the field definition. In Fortran, field parameters are accessed through subroutine and function calls. In C, a field is a predefined C structure that can be directly accessed and modified. AVS offers a wide range of interactive X-widgets for inputting and modifying parameters, which allows the programmer to create sophisticated user interfaces with little effort.

Depending on how modules are linked, AVS applications will execute and communicate in two different ways. If modules are linked into separate executables, each module runs as a separate UNIX process, communicating with other modules through UNIX sockets. If all modules are linked together, the application runs as one executable, calling individual modules as subroutines, which pass data and parameters through pointers in memory. These two execution modes can have considerable differences in performance and memory usage, especially on the CM-5 (see below).

CM/AVS consists of a set of AVS modules that operate on data residing in CM-5 memory. The modules can be manipulated within AVS in the usual fashion, and they can be connected to standard AVS modules. During execution, a network that contains both AVS and CM/AVS modules will generate processes that run both on the AVS-server (an HP 735 at SEP) and on the CM-5; communication between these processes takes place through UNIX sockets. The goal in designing a CM/AVS application is to execute the data and computationally intensive parts of the application on the CM-5, and to display the results on the workstation. This is an important advantage because the size of datasets on the CM-5 can easily be in the order of several hundred megabytes, whereas the size of a typical graphics image is generally less than one megabyte. Programming in CM/AVS is similar to programming in AVS, with the main difference that programs are written in Connection Machine Fortran (CMF), and that fields are CMF arrays.

There are several limitations to AVS that the programmer needs to take into account. First, AVS is designed such that a change in a module parameter only affects modules "downstream" in the network. Although this can significantly speed up the application (fields in "upstream" modules do not have to be recomputed), it can also seriously reduce memory available to the user, because each module must make a copy of the data. To lessen this problem the latest release of AVS allows the user to overwrite input fields, but only if input and output field have the same dimensionality. A second limitation arises from the fact that the CM-5 operating system does not currently implement parallel UNIX sockets; if CM/AVS modules communicate through sockets all data transfer takes place through the CM-5 partition manager. Obviously, this limitation would defeat the purpose of writing CM/AVS applications. The current work-around for this problem is to link all modules into one executable as described above. In the next release of CM/AVS the parallel sockets will exploit the high-bandwidth CM-5 data network to exchange data efficiently between modules.

MULTI-DIMENSIONAL SEISMIC DATA VISUALIZER

Our multidimensional data visualizer is actually a collection of CM/AVS networks built from many AVS and CM/AVS modules. We developed the following CM/AVS modules: Browser, Slicer, Gain, and Pan_Zoom. The visualizer is started by invoking a C-shell (CM_Slicer) and a particular network is selected by specifying parameters on the command line. Different networks may include different modules, and the modules may be in different order. For example, one of the command-line parameters is `gain`. If the user types `gain=no` on the command line, a CM/AVS network without gain modules is fired up. On the contrary, if `gain=all` the Gain module will proceed the Slicer modules, causing the whole dataset to be gained at once. Finally, if `gain=every` the Gain modules will follow the Slicer modules and the different slices will be gained independently.

Figure 1 shows a example of CM/AVS network that can be started by invoking CM_Slicer. The network begins with a standard AVS module called file browser that is used for selecting the input file name, that must be a SEP header file. The file name is passed on to the Browser that reads from disk a subset of the dataset, as specified by the user. The data so loaded in memory are passed to three instances of the Slicer module that extracts two-dimensional slices out of the multi-dimensional dataset. Each of the three slices is then passed to a Gain module for interactive gain, then to a Pan_Zoom module, and finally the three images are merged together and displayed in one window by the standard AVS module image viewer. The image viewer module displays the slices in a flat view, but the same slices can be rendered in three dimensions by the geometry viewer AVS module, as we will show in the next section.

As an example to illustrate the application we visualize a marine 3-D prestack dataset with no cable feather; that is, a four-dimensional dataset. The data has 3 parallel lines with 501 shot gathers each. The gathers have 120 traces of 704 time samples each. The total size of the dataset is only about 500 Mbytes; therefore the whole dataset can be read and stored in the CM-5 memory without sub-sampling. Figure 2 shows the window displaying a common-offset section, a shot gather, and a time slice extracted from the prestack dataset. In the remainder of this section we will describe in more detail the CM/AVS modules that we developed, and we will illustrate their user interfaces.

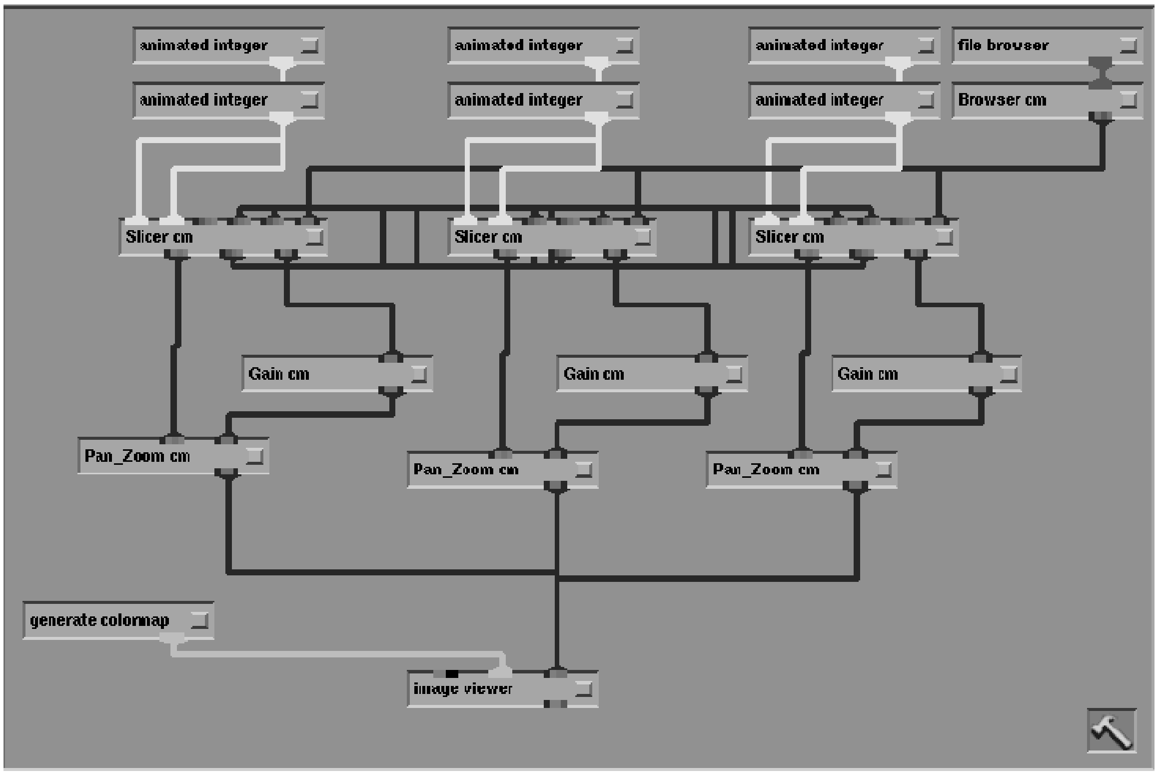

Figure 1: CM/AVS network for reading, windowing, slicing, and gaining a multi-dimensional seismic dataset. The module with cm at the end of the name run on the CM-5, the others are standard AVS modules running on a workstation. biondo1-3d-sl-net [NR]

Browser

The Browser module reads and decimates the data. The data file can be either on the Scalable Disk Array (SDA) of the CM-5 or on any file system in the network mounted on the CM partition manager. In the first case the read operation will be fast (about 10 Mbytes/s), whereas in the second case it will run at network speed. The disk file can span several gigabytes (our SDA capacity is about 17 Gbytes), but, at any given time, only about 600 Mbytes can be held in memory and visualized. To enable the analysis of large datasets the Browser module provides the capability of reading from disk only a subset of the data. The subset to be read into CM memory is selected by using sliders, as shown in Figure 3. Following conventions adopted in the SEP Window program, the values of `n3` and `n4` will determine the lengths of the third and fourth axis of the subcube to be read in memory. The parameter `f3` and `f4` determine the starting points along the third and fourth axis, and `j3` and `j4` the sub-sampling rates. Once the sliders are set on desired values, the button `Browser-Enter` is clicked, and the read operation is initiated.

Slicer

The Slicer module extracts a two-dimensional orthogonal slice out of a multi-dimensional dataset. There are two options for extracting the slices. The default option is to extract the slices from the original dataset on the fly; the extraction of each slice then requires inter-processor communications. The alternative option is to pre-transpose the data such that the sliced axes are mapped locally into the processors' memory. Consequently, the slices can be extracted from the transposed data by a fast local memory reference, allowing quick data slicing. The drawback of this option is that the transposed copy of the data takes as much memory space as the original dataset, thus limiting the size of the subset of the survey that can be kept in memory and interactively sliced through.

Figure 4 shows the window to control the three Slicer modules connected to the network shown in Figure 1. The two options to perform the slice-extraction operation (pre-transpose and on-the-fly) are selected and monitored by the toggle buttons on the right hand side of the controls. The axes to be displayed are selected by typing their number near the labels `HAxis` and `VAxis`. After this selection the program automatically pops up at the bottom of the window the sliders for positioning the slices. If N is the dimensionality of the dataset to be displayed (in our example $N = 4$) the program creates $N - 2$ sliders. A movie of the data can be created by continuously stepping along the sliced axes. Our network includes many instances of the AVS module animated integer for animating the sliders in the Slicer module. Figure 5 shows the control panels that are available to the user for controlling the looping in the movie.

When more than one orthogonal slice is extracted from a dataset, and the slices are displayed in a flat view, it may be useful to show the relative positioning of all

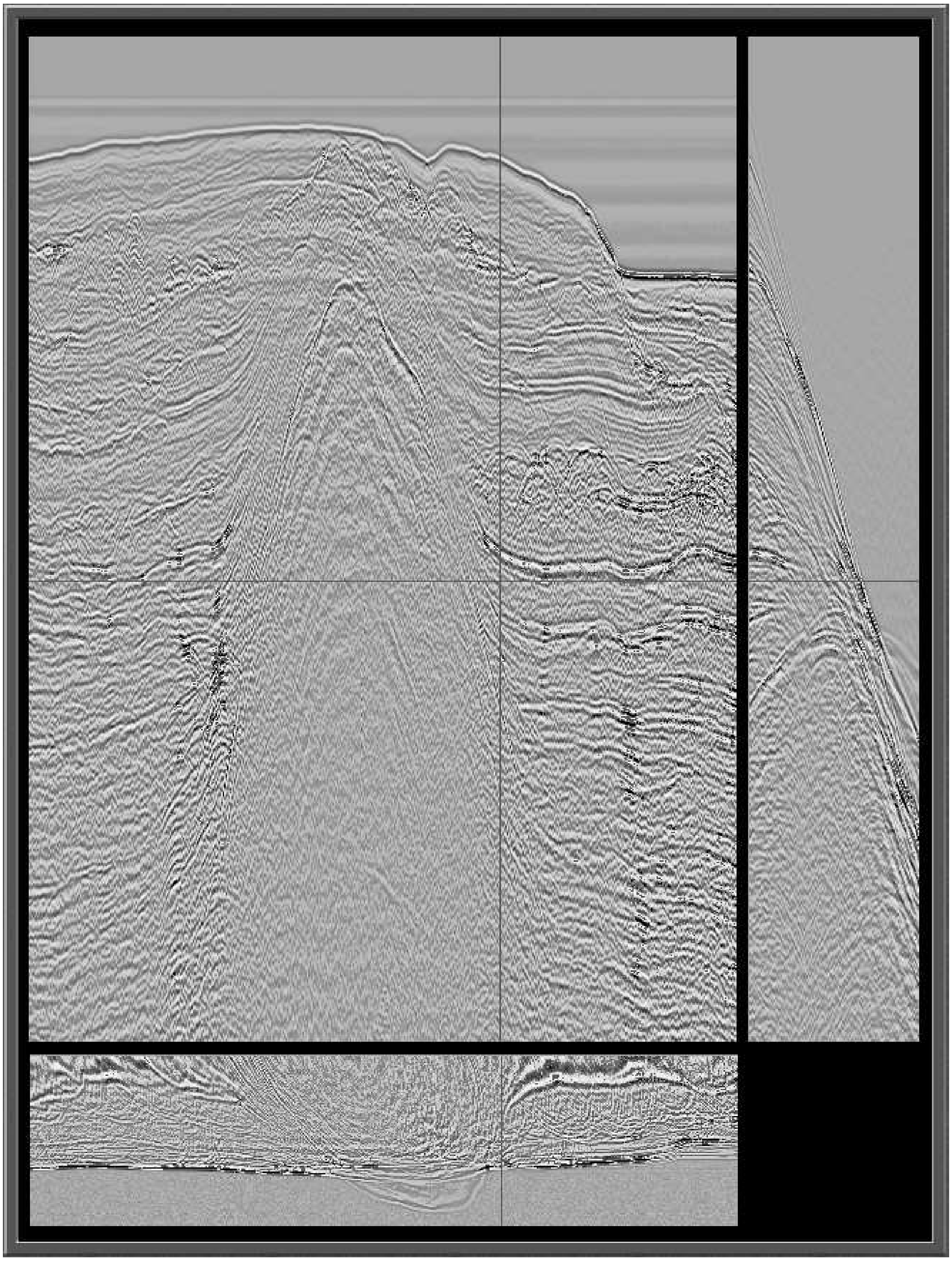

Figure 2: Display of a 3-D prestack marine survey created using the network shown in Figure 1. The three slices show a common-offset section, a shot gather, and a time slice. biondo1-Amoco-salt-disp [NR]

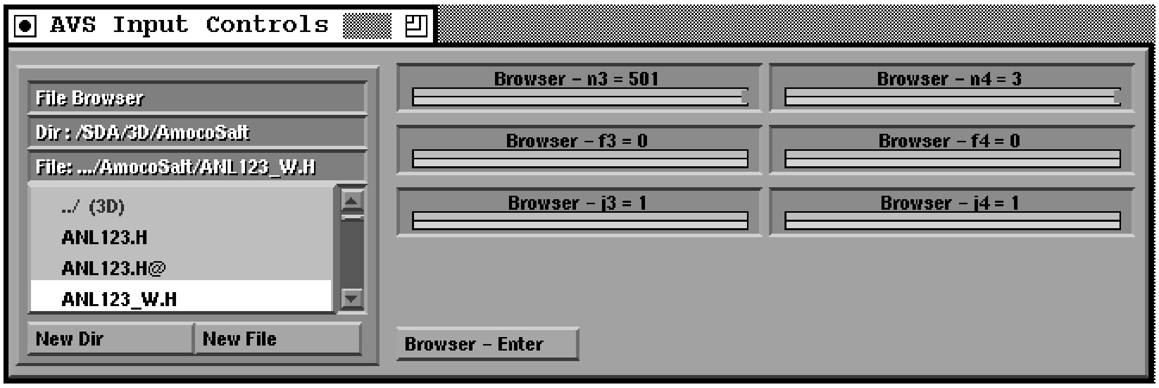

Figure 3: Control panel for selecting the input data to be read from disk. The file browser on the left is used for selecting the data file. The sliders on the right are used for selecting a subset of the dataset to be read into the CM-5 memory. $\boxed{\text{biondo1-Input-cont}}$ [NR]

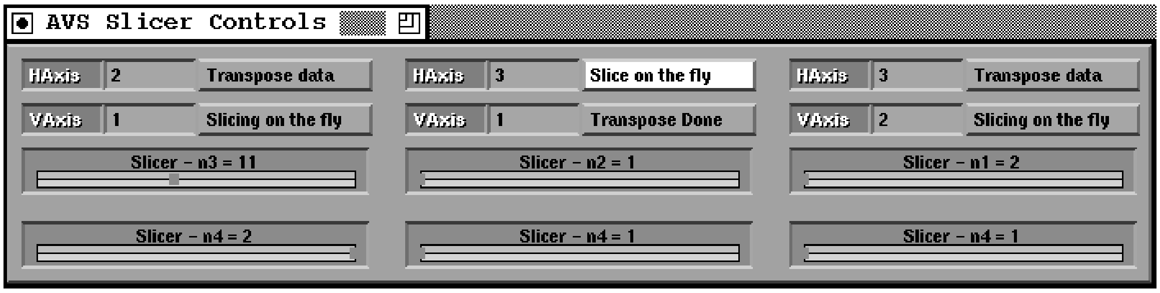

Figure 4: Control panels for slicing through the multi-dimensional dataset stored into CM-5 memory. The user can choose the axes to be displayed, the mode with which to slice the data, and finally the position of the slices to be displayed. $\boxed{\text{biondo1-Slicer-cont}}$ [NR]

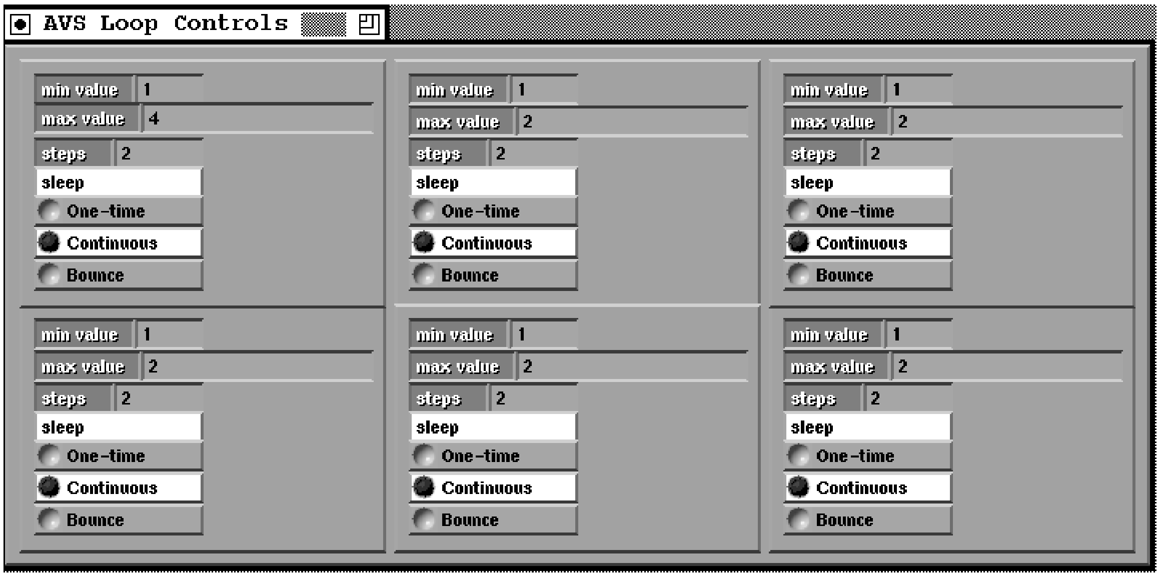

Figure 5: Panels for controlling the dynamic slicing through the dataset. There is one control for each of the sliders shown in Figure 4. biondo1-Loop-cont [NR]

the slices in the display. This task is accomplished by superimposing lines onto the images where the slices intersect each other. To properly position the lines on the images, each instance of the Slicer module needs to communicate with all the other instances of Slicer in the network. These communications are performed through the horizontal data links shown in the network of Figure 1. Using the information gathered from the other Slicers, each Slicer can determine the positions of the lines on its own image. This information is passed to the Pan_Zoom module that actually superimposes the lines onto the images. To further facilitate the interpretation of the display, the superimposed lines are color coded; i.e., lines indicating the position of the same slice are shown in the same color.

Gain

The Gain modules can be used for interactively gaining and clipping the data for improving the quality of the display. Typical operation are clipping of high amplitudes (`clip`), changing the dynamic range of the data (`gpow`), or gaining the data as a function of traveltime to compensate for geometrical spreading (`xpow`). Figure 6 shows the dials that are available to the user for interactively changing these parameters.

Pan_Zoom

Presently the Pan_Zoom module accomplishes the simple task of superimposing onto the images the lines that show the positions of the other slices in the display. The in-

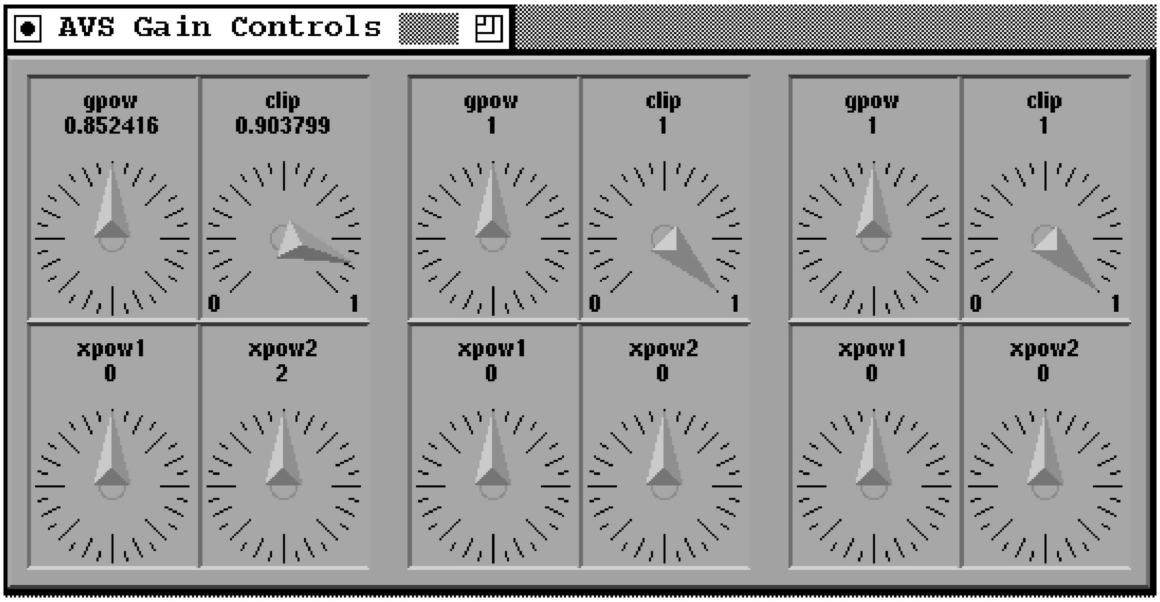

Figure 6: Control panels for interactively gaining and clipping the images. biondo1-Gain-cont [NR]

formation necessary for locating and color coding these lines is received by Pan_Zoom from a Slicer module. In the future, the main task of the Pan_Zoom module will be to window, translate, and zoom the slices before sending them to a display module. For the moment these capabilities are provided by the AVS module image viewer; image viewer is a flexible tool for displaying images, but it is also the data-flow bottleneck of the visualization. Before being displayed by an image viewer module, the images must go through many communication steps: from the processing node of the CM-5 to its partition manger memory, from the partition manager to the AVS-server, and finally to the X-server. Future developments in CM-5 hardware and software should enable to send the data directly from the CM-5 nodes to an X-server, without passing through the partition manager, thus speeding up the whole visualization. However, when these fast communications are possible, Pan_Zoom will need to have the pan and zoom capabilities now provided by image viewer.

INTEGRATION OF SEISMIC WITH OTHER GEOPHYSICAL DATA

An important reason for having chosen AVS as the software platform to develop our seismic data visualizer is that within AVS seismic data can be easily displayed together with other geophysical data. For example a 3-D dataset can be rendered together with a surface representation of the geological structures by merging the display of our visualizer with the display of a geological modeling program (e.g. GOCAD). Another useful application is to display a velocity model together with a 3-D seismic dataset. The AVS rendering module geometry viewer is the basic tool that can be used to integrate the display of different geophysical data. It can render in the same view multiple 3-D geometries, each of which can describe seismic data, velocity models, or geological models.

Figure 7 shows an example of a combined display of 3-D seismic data with a 3-D velocity function. The data and the velocity model are independently sliced, the slices are then rendered in a three-dimensional space and displayed in the same view. To facilitate the interpretation of the display, a contour plot of the velocity model is superimposed to the velocity slice, and two different colormaps are used for color-coding the velocity model and the seismic data. Figure 8 shows the CM/AVS network we used to generate the display in Figure 7. The seismic data are read, gained, and sliced on the CM-5, and then passed to the AVS module field to mesh that generates 3-D geometries that are finally rendered and displayed by the geometry viewer module. Similarly, the velocity model is read, sliced, and gained on the CM-5, before being rendered together with the data by geometry viewer. Although the rendering is limited to 3-D geometries, the capability of our CM/AVS modules to manage datasets with higher dimensionality than three can be exploited by stepping in time along the other axes. For examples, the data shown in Figure **??** are the results of many 3-D poststack migrations merged together along the fourth axes of the dataset. The migrations were obtained using different filters for the Hale-McClellan algorithm (Biondi and Palacharla, 1993; Palacharla and Biondi, 1993). The quality of the results can be easily compared by switching between different migrations by slicing the 4-D dataset along the fourth axis.

CONCLUSIONS

The visualization of large 3-D poststack and prestack datasets is crucial for the development of innovative methods and algorithms for processing 3-D surveys. The size and dimensionality of 3-D surveys require visualization tools that can handle datasets with more than three dimensions and spanning many gigabytes of data. By exploiting the data-handling capabilities of SEP's CM-5 and the modular design of CM/AVS we have developed a visualization tool for visualizing large multi-dimensional datasets. The same tools used for the visualization of seismic data can be integrated with tools that visualize other geophysical datasets, such as velocity models and geological models.

ACKNOWLEDGMENTS

We would like to thank Amoco and Halliburton for kindly providing to SEP the datasets that we used for demonstrating our data visualizer.

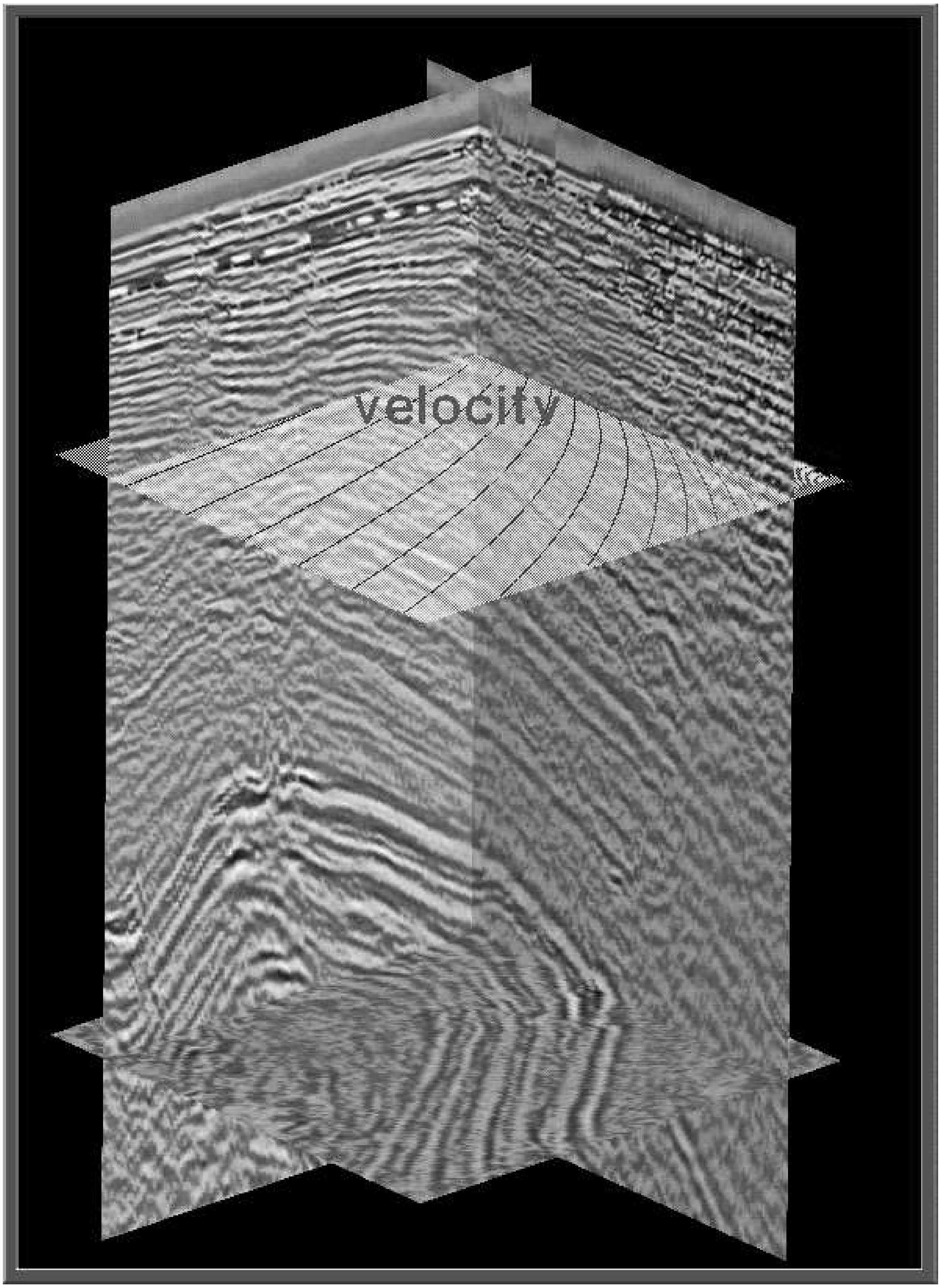

Figure 7: Integrated display of a 3-D seismic dataset together with a 3-D velocity function. A contour plot of the velocity slice is superimposed to the rendering of the slice itself. biondo1-3d-geo-disp-vel [NR]

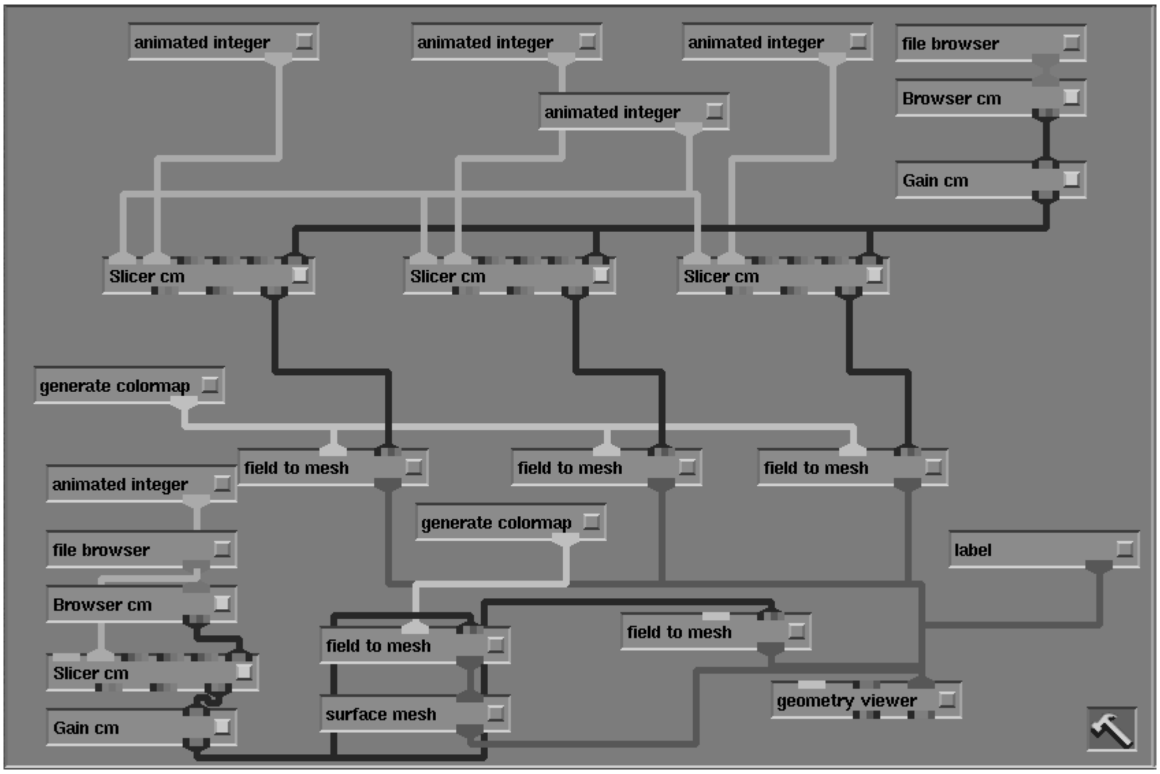

Figure 8: CM/AVS network for displaying a 3-D dataset together with a velocity model. biondo1-3d-geo-net-vel [NR]

REFERENCES

Biondi, B., and Palacharla, G., 1993, 3-D wavefield depth extrapolation by rotated McClellan filters: SEP–**77**, 27–36.

Mallet, J. L., 1993, GOCAD Report, version 8.1, Confidential: Association Scientifique pour la Géologie et ses Applications.

Ottolini, R., Sword, C., and Claerbout, J. F., 1984, On-line movies of reflection-seismic data with description of a movie machine (short note): Geophysics, **49**, no. 2, 195–200.

Palacharla, G., and Biondi, B., 1993, 3-D Hale-McClellan migration: a salt intrusion example: SEP–**79**, 93–102.