Stanford Exploration Project, Report 82, May 11, 2001, pages 1–104

AVS as a 3-D seismic data visualizing platform

Robert G. Clapp, Biondo Biondi, and Martin Karrenbach[1]

ABSTRACT

In an effort to increase SEP's ability to work with 3-D prestack data sets we have continued the development of our integrated 3-D seismic visualizer using Advanced Visual Systems (AVS) as our integrating platform. To the capacity of displaying SEP data sets, we have added the ability to simultaneously display non-seismic 3-D geophysical and geological data. The main thrust of our work has been the integration of GOCAD model building tool into AVS. To achieve this goal we have developed a few AVS modules to convert GOCAD surface into AVS geometry objects. In addition, we have begun integrating SEP batch programs into our AVS environment. Specifically, we have added an interactive wave modeler, which can be viewed simultaneously.

INTRODUCTION

As processing and interpretation become more tightly coupled, a need arises for visual interaction with all the relevant geophysical and geological data that describe hydrocarbon reservoirs. Thus, the challenge becomes to create a three dimensional, and possibly dynamic, view of the physical space where a multiplicity of data types can exist. To achieve this integration without developing our own specialized graphic software we use AVS, a software system developed by Advanced Visual Systems Inc. AVS has many attractive features as an integration platform. It supports flexible data types and, if needed, it allows the developer to define their own data types. It has powerful 3-D rendering modules that can be used for rendering 3-D geometries without the need of graphic programming. Its modularity enables quick prototyping of fairly complex applications by building networks of modules. Finally, converting existing batch programs into interactive AVS modules is fairly straightforward, thus increasing their effectiveness. As an example we converted a batch finite difference modeling program (**?**) program into an interactive module that enables the user to observe the propagating wave-field together with the velocity and geological models.

Previously, (**?**), discussed tools SEP developed for the visualization of 3-D pre-stack data sets. In (**?**), GOCAD, which is quickly becoming the industry standard for velocity model building, was used in conjunction with AVS in an iterative migration scheme. In this paper we report on our current work to expand on these initial building blocks. First, we add to

[1]**email:** bob@sep.stanford.edu,biondo@sep.stanford.edu,martin@sep.stanford.edu

our library of AVS modules by improving the integration of GOCAD into AVS. Specifically, we built a series of modules to convert GOCAD surfaces into AVS geometry objects. They can then be easily combined through preexisting AVS modules with velocity models and the data into an interactive display. Second, to facilitate understanding of 3-D acquisition, its relation to the data, and subsequent processing, we built a module that can visualizes acquisition geometry from a geometry database in conjunction with the geological structure and seismic data.

AVS INTEGRATION PLATFORM

One of the main strengths of AVS as an integration platform for geophysical visualization is the flexibility of the data types that can be used to represent the data that are exchanged among modules and visualized. The fundamental AVS data type is an AVS "field". An AVS field can be a scalar or a vector function, it can be regularly sampled or irregularly sampled, it can have an arbitrary number of dimensions, and it can be defined in a space of arbitrary dimensionality. Most geophysical data can be represented as AVS fields. For example, a one-component 3-D prestack data set is a five-dimensional scalar function irregularly sampled defined in a four-dimension physical space (time and three spatial dimensions). The capabilities of AVS of manipulating fields with higher dimensionality than three has a great relevance in seismic applications. First, it allows natural representation of 3-D prestack data. Second, it allows us to add dynamics to the visualization, i.e., displays can change as a function of time, or as a function of processing parameters such as velocity.

An AVS "geometry" is another fundamental AVS data type. An AVS geometry describes the spatial representation of an object and some of its properties useful for visualization (e.g. color). A geometry can be rendered in three-dimension using a rendering module called `geometry viewer`. The geometry viewer is a flexible rendering program that allows the user to manipulate the views of the objects as well some of their properties, such as transparency. In the standard AVS module library there are many modules (called mappers) that transform an AVS field into an AVS geometry which then can be rendered by the `geometry viewer`. The choice of the mapper depends on the nature of the field to be visualized. In addition to the supported library, there is a growing number of public domain modules available on the net by ftp.

AVS allows also the definition of User-Defined data types, when AVS fields are not adequate for parameterizing the data. An example of a useful application of User-Defined data type is the representation of GOCAD surfaces. In addition to the coordinates of its vertices a complete description of a GOCAD surface must include some topological information, i.e., the connectivity among vertices. A developer can exploit AVS's modularity by defining a "GOCAD surface" data type, which will allow for direct exchange of surface description information among AVS modules.

SEP's 3-D seismic AVS modules

The foundations for our 3-D environment were established last year. Biondi et al. (**?**) describes a set of AVS modules to visualize multi-dimensional seismic data stored in SEP format. These modules run on SEP's CM-5 and take advantage of its data handling and computational capabilities. Specifically, four modules were developed: `Browser cm`, `Slicer cm`, `Gain cm`, and `Pan_Zoom cm`. The `Browser cm` module reads in either all or a subset of a multi- dimensional SEP data set and transforms it into and AVS field. The `Slicer cm` extracts a two-dimensional slice from a multi-dimensional AVS field. While the `Gain cm` and the `Pan_Zoom cm` modules allow the user to gain, pan, and zoom the image interactively. To complete these seismic data visualization tools we have added a new module called `Trace Geometry`, which visualizes the acquisition geometry of a survey, starting from the trace geometry information. We plan to use this module for analyzing irregularly sampled 3-D prestack data.

The `Trace Geometry` module is a tool to analyze the properties of acquisition geometries according to different criteria. It can simply display vectors at source and receiver positions, or, similarly, display offset vectors at midpoint locations. It also allows the user to define a regular binning grid and then to visualize its properties, such as fold. Figure 1 shows one of these binning displays. The arrows start at the bin locations and represent the offsets of the seismic trace belonging to the respective bins. To avoid overcrowding of the display the displayed binning grid can be undersampled with respect to the actual binning grid. The displayed geometry is the actual geometry of a 3-D land survey that we have at SEP. The data was acquired by rolling over parallel swaths of geophones. The display illustrates the narrow azimuthal coverage at large offsets that is typical of these acquisition geometries.

The next major step in our development is the capability of displaying and manipulating GOCAD surfaces with AVS. Because GOCAD surfaces can not be effectively described as AVS field, we developed modules that converts GOCAD's representation of surfaces into AVS geometries. The next section will describe this work in more detail.

CONVERTING GOCAD SURFACES TO AVS OBJECTS

Before discussing how we convert GOCAD surfaces to AVS geometry objects it seems appropriate to explain our reasoning for choosing AVS over GOCAD as our integrating platform. GOCAD was designed as primarily as a 'model' building tool, and has been slow to integrate the capability to display other data forms. It is not until the latest release that it incorporated the ability to display seismic data, and is still limited to displaying no more than a 3-D volume. Further, converting existing batch programs into the GOCAD environment requires a fairly extensively knowledge of C and GOCAD subroutines.

AVS, on the other hand, does not have such limitations. AVS routines can be written in either Fortran or C. In addition AVS provides a module-writing routine that further simplify the process. Further, the graphical nature of the AVS environment makes combining different data sources a simple matter. Module inputs/outputs are color coded based upon their type. Sending information from one module to another simply requires connecting from one output

../bob/./Figs/acqgeom.ps

Figure 1: Bin locations and offsets superimposed on the surface model.

port to an input port of the same color. GOCAD has several different types of objects (**?**) including, ‘p-lines’,‘t-surfs’, and ‘v-sets’. For seismic velocity model building we are mainly concerned with the t-surf object class. The t-surfs, in their most elemental form, can be defined through vertices and triangles, which define the connectivity between the vertices. In GOCAD a t-surf can take two forms, ASCII or binary. The ASCII form is the simplest of the two forms and consists of a series of lines beginning with either VRTX or TRGL. The lines beginning with VRTX, specify a vertex number and the x,y,z coordinates of a point in the modeling space. Lines beginning with TRGL define the connection between the vertices and consists of three vertexes specified through there vertex number. In ASCII form each surface is defined by one and only one file. In binary form several surfaces are combined into a “project”. The project is in fact a directory containing the surfaces in binary form and a series of files that GOCAD uses to interpret the binary form. The binary form of the surfaces is interpreted by GOCAD as a C structure, with the x-coordinate specified by $trgl- > p\_atom[0]- > p\_vrtx- > pos.x$. We chose to convert from the binary form rather than the ASCII form because:

- It is the intermediate form that GOCAD uses for manipulation.
- Many routines, such as our gridding program, require a project to be defined.
- Storage and I/O efficiency.

In addition to the vertex (atoms in GOCAD nomenclature) and triangle information GOCAD also stores:

- normals of the atoms,
- stationarity, used to define movability in smoothing routines,
- colors of the surfaces.

AVS also has several different types of geometry structures including: mesh objects, for objects which can be described by a regular grid; polytriangle, for object that can be specified by adjacent triangles; sphere, for spherical objects; and polyhedrom, for objects which can be defined by a series of polygons.

We chose to use the polyhedrom object class as the basis for constructing the GOCAD surfaces because it allows for surfaces which may not be contiguous, a common feature in GOCAD surface. The polyhedrom object class, at its most basic level can be specified by two arrays. The first array is a two dimensional array of vertices. The second array is a one-dimensional connectivity array, which specifies to AVS how the various vertices connect. The connectivity array is specified by a number (N) representing the number of sides of the polygon followed by N values indicating where within the vertex array the various apices exist (see Figure 2). In addition AVS attaches normals and colors to the object.

Figure 2: Arrays specifying AVS geometry class polyhedrom. The connectivity array, left, consists of **N** polygons. Each polygon is described by a number **m** corresponding to the number of vertices in the polygon, followed by **m** numbers indicating where in the vertice array, right, the corresponding apex can be found. The vertice array consists of three columns which describe the location of the vertices in the modeling space.

Conversion routine

In order to allow easy manipulation of the surfaces we chose to create an intermediate form for the surfaces. From this form we can currently read a GOCAD binary form, manipulate the surfaces, and write them to an AVS geometry. In the future we will add the ability to re-save the manipulated surfaces into a GOCAD binary form. The conversion routine itself is rather simple:

```
GOCAD project file is opened and the specified surface loaded
All of the triangles are looped over
        {
        Each vertex of the triangle is examined
                {
                If not previously encountered written to a vertex array
                If written before the associated vertex number is loaded and stored in a
                connectivity array
                }
        The three associated vertexes are written to a second array
        }
Normals of the surfaces calculated
Polyhedrom geometry written
```

AVS Interface

The GOCAD to AVS conversion was accomplished with two modules. The first module, `readsurface`, works as a modified file browser. It contains two AVS *chooser* sockets. The first provides the user with the ability to peruse through the directory tree until one finds the project (indicated by the .PJ at the end of the directory name) containing the surface(s) he wishes to load. Then the user simply has to click on any file in the project directory. The module then uses GOCAD functions to search the designated project for the list of surfaces contained within. This list is then piped to the second selection window. The user can then search through the available surfaces selecting which one to load. Once the surface has been selected, the surface and project names are piped to the second module, `readgocad`.

The `readgocad` modules performs the conversion from GOCAD to AVS and will be the main routine that all future GOCAD to AVS modules will interact with. Currently the `read-gocad` modules have two possible operating environments, loading and editing. In the future a save option will be incorporated. If the user selects the load environment two *chooser* and one *typein* widgets appear. The first *chooser* widget enables the user specify a color for the surface which is to be loaded, as a default the six colors are automatically cycled throusix gh. The second *chooser* widget and the *typein* widget take advantage of the AVS concept of parent and child objects. The parent/child object allows an object (the parent) to be defined by a group of objects (the children). Taken to the extreme (see Figure 3) the parent/child concept would allow each polygon (*polygon 1...n*) of a surface to be a separate object, the surface its parent (*surface1*), all of the surfaces that represent faults might be grouped into another object (*faults*), and finally all parts of the geologic model another object(*model*). The advantage of the parent/child concept is that a parent can be assigned a property, such as color, and all of the children can automatically inherit the property.

The second option, editing, is still being developed. Currently, when editing is chosen (see Figure 4), six *dial* widgets pop up. The first three widgets allow the user to move the surface in the x_1 direction of the modeling space from $-.5*$(extent of model in the x_1 direction) to $.5*$ (extent of model in the x_1 direction). The second set of widgets allow the the size of the surface to be expanded/contracted by a factor of 10. In the future we plan to expand the editing feature to incorporate the GOCAD DSI smoothing routine and to allow a single vertex within the surface to be moved.

../bob/./Figs/structure.ps

Figure 3: AVS parent-child structure.

WAVE MODELING

We perform interactive wave equation modeling by wrapping an existing Finite Difference modeling code (**?**) into an AVS module. The modeling is carried out remotely on the massively parallel computer. The input to the modeling module is provided by the `Browser cm` module and consists of a slice taken out of the 3-D model volume. The Finite Difference code is second order in time and 16th order in space and allows the user to model wave propagation acoustically or elastically in the 3-D model. The modeling code has numerous options for types of medium, boundary conditions, and recording types to mention a few. However, only a small subset of its capabilities is currently available in the AVS module For speed reasons we restricted ourselves to modeling in 2-D since we wanted to perform interactive tests by changing the modeling parameters and see the effect on the wave field. The output of the modeling module are wave field snapshots and seismograms. The snapshots can be rendered by AVS and displayed simultaneously with the original gridded model and the geological interfaces into one 3-D viewing window. Figure 5 shows the propagating wavefield through the velocity with the superimposed surfaces. The snapshots are displayed in real time as they are calculated and it is easy to follow the wave field effects and relate them to the structure of the model. Slices can be cut through the model at arbitrary orthogonal positions and each of those slices can be displayed in the **Geometry Viewer** simultaneously. It is therefore possible to approximate the 3-D behavior of the wavefield by 2-D wave field slices. The original application of the interactive wave modeling was the rapid analysis of modeling parameter changes. For examples changes in source frequency content could be directly related to the occurrence of numerical dispersion in the wave field. Switching between an acoustic and elastic model clearly showed which effects are suppressed by simplifying the earth model. Thus the interactive wave modeling as an integrated AVS flow allows the user to quickly assess model parameter changes and its effects on the seismic data. Ultimately we would like to be

../bob/./Figs/readgocad2.ps

Figure 4: A control panel of `readgocad` module in editing mode. Panel illustrates several of the available AVS widgets including: *choice* which gives the user one of three options (edit,load, or save), each of which will produce a different set of widgets; *choice browser* which allows the user to select from a predetermined list of options, in this case color; and *radio buttons* which allow the user to interactively set float parameters.

../bob/./Figs/snap2.ps

Figure 5: Geologic model with velocity model (background) and propagated wavefield (foreground) superimposed.

able to interactively change the geological model and regrid the velocity model all within AVS, and follow up with an immediate modeling of wave propagation through the newly gridded model.

CONVERTING SEP PROGRAMS TO AVS MODULES

In the process of converting the Finite Difference modeling program into an AVS coroutine module we found that AVS and SEP follow a very similar philosophy. The translation from one to the other follows a simple scheme and we have tried to write a Perl-preprocessor for this task. The processor was able to convert about 90% of the modeling code, while the remaining 10% had to be modified manually. We are optimistic that in the future we will have a preprocessing script that will automatically convert SEPLIB-saw programs into AVS coroutine modules. The SEPLIB way of reading parameters from the command line and defaulting is reciprocated by AVS by reading parameters from dials or other widgets; default settings and bounds are specified within the program. Allocation of data arrays in SEPLIB corresponds to allocation of AVS fields, where the fields contain all the information that is hidden in the SEPLIB header file. Instead of reading and writing data with `sreed` and `srite` in SEPLIB, we access an AVS field by calling `corinput` and `coroutput`. Using UNIX pipes or sockets to connect various SEPLIB programs has its analog in AVS where we draw connecting pipes between modules. Even the SEPLIB data topology translates directly into a uniformly sampled scalar AVS field topology. An AVS field contains the SEP data structure as a subset. This clearly shows the power of using AVS as an integration tool for totally different applications such as SEPLIB and GOCAD.

FUTURE WORK

Though our initial efforts have been quite successful we are far from our final goals. Presently we plan to expand along three main fronts. First, we hope to build AVS “wrappers” on many of the existing programs in the SEP library, allowing a great amount of the work on 3-D data sets to be done in the AVS environment. Our first target for an AVS wrapper is the GOCAD to SEP gridding program, (**?**) which will provide the framework for allowing the entire velocity model building process to be included in AVS. In the future we also plan to integrate a migration algorithm into AVS and improve the preprocessor script to make the conversion from SEPLIB-saw programs simpler. The second thrust will be to more fully integrate some of the GOCAD functions into AVS. In addition to the ability to save altered surfaces we plan to integrate the DSI function into AVS. Finally, we plan to build new modules for AVS that take advantage of its ability to display and move multiple data types. Currently we are considering adding the ability to display well-log data and interactive model building by picking surfaces on orthogonal slices of the data.

REFERENCES