3-D Seismic visualization

Henry C. Bland and Robert R. Stewart

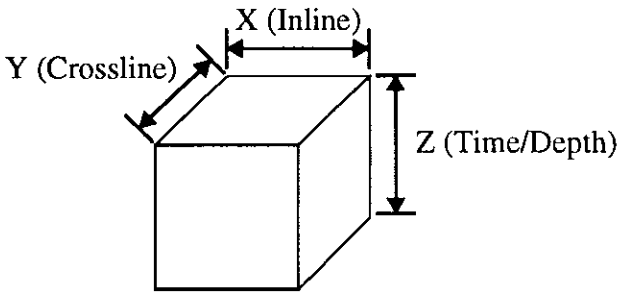
ABSTRACT

This paper discusses some of the current techniques for visualizing 3-D seismic data such as perspective views of data slices, cube displays, voxel based volume rendering, and surface rendering. An example of 3-D visualization of a complex seismic horizon is shown, along with the techniques used in creating the example. Lastly, new visualization techniques such as stereoscopic displays, 3-D pointing devices and virtual reality may be useful for seismic interpretation.

INTRODUCTION

Traditional interpretation involves viewing slices of the dataset: either as vertical sections or horizontal time slices. As 3-D surveys grow in size, complexity, and importance we need new ways to obtain more information from them using a more compelling presentation.

These seismic data come to the interpreter as a regularly grided 3-D block of sample points. This volume or block has spatial dimensions on two axis, and time or depth on the third axis. Traditionally, the two spatial dimensions fall along the inline and crossline directions of the 3-D survey grid. Interpreters attempt to infer structure, stratigraphy, and fracturing from this data volume. Optimal visualization of these features can be greatly benefit the understanding of the full dataset.



Slices

The traditional approach to looking inside a data volume is to slice the volume in the x-z or y-z planes to create inline and crossline sections. Time slices are created by slicing the volume horizontally. The individual slices can be interpreted using 2-D techniques but clearly we lose valuable information when we do this since we cannot see data coherency along the axis normal to the slice plane. To understand how slices relate to each other, it is useful to display them sequentially in an animated traversal through the volume. This is technique is simple and effective but still only allows us to see two of the three dimensions at one time. Since one dimension is displayed as

time, it is difficult to appreciate relative distances in all three dimensions simultaneously.

To provide a better appreciation of the data volume, we need to go beyond flat projections of slices. The key to improved visualization of slices is to use perspective. To do this, we build a 3-D model of the slices in space, orienting and positioning the slices to match the way they were removed from the data volume. If we now draw these slices using an orthographic projection we get a perspective view of the data, and we can understand how two or more slices interrelate in 3-D space. One popular way of orienting slices is to combine three slices (with orthogonal normals) to form the front, top and side of a cube. Since we view this cube from a corner-on point of view, we only see three faces, as the other three faces are obscured. With this configuration we get a very good 3-D view of the corner vertex, and we get a reasonably good understanding of the 3-D properties along the edges. We cannot however look inside the data volume: we still only see the outside surface.

Animation can be used to enhance this cube perspective. By changing the slice location in one, two, or even all three of the primary axis, we can obtain a traversal through the data volume. Even though perspective animation does not actually show much more data than the traditional animations of single-slices, we do get a better feeling for the 3-D nature of the data since the top and side faces of the cube give a cue as to the location of the front face's slice with respect to the whole volume. This depth-queuing helps us appreciate the fact that we are moving through the volume.

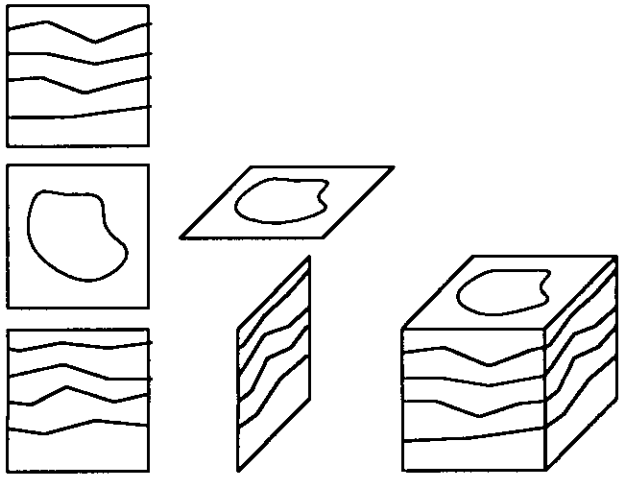


FIG. 2. (a) Inline, timeslice, and crossline slices of a data volume (simulated) (b) 45° skewing of top and side faces (c) cube-view of data volume

Implementing the cube-view is very simple and computationally inexpensive. The first step is to create three 2-D slices of the data volume along each of the three primary axes (Figure 2a). The front slice is shown flat in two dimensions. The other two slices are shown skewed. We skew the top face horizontally, and skew the side face vertically (Figure 2b). The skewing calculation can be simplified by using easy-to-compute orthographic projection angles such as 45°. For the 45° case, the top face is created by subsampling the top slice by 2, and shifting the image right one pixel per row. The left face can be similarly subsampled and skewed by in integer shift. The

simplicity of this technique lends itself well to interactive displays on systems that lack specialized graphics hardware.

Perspective view of surfaces

After the interpreter has picked seismic horizons, faults, or other items of interest the perspective view adds insight into the interpreted area. Traditional perspective views used wire-frame drawing techniques. Only recently have improved visualization techniques commonly been used. By breaking the surface into polygons, algorithmic shading of the polygons can result in a far more realistic rendition of the surface.

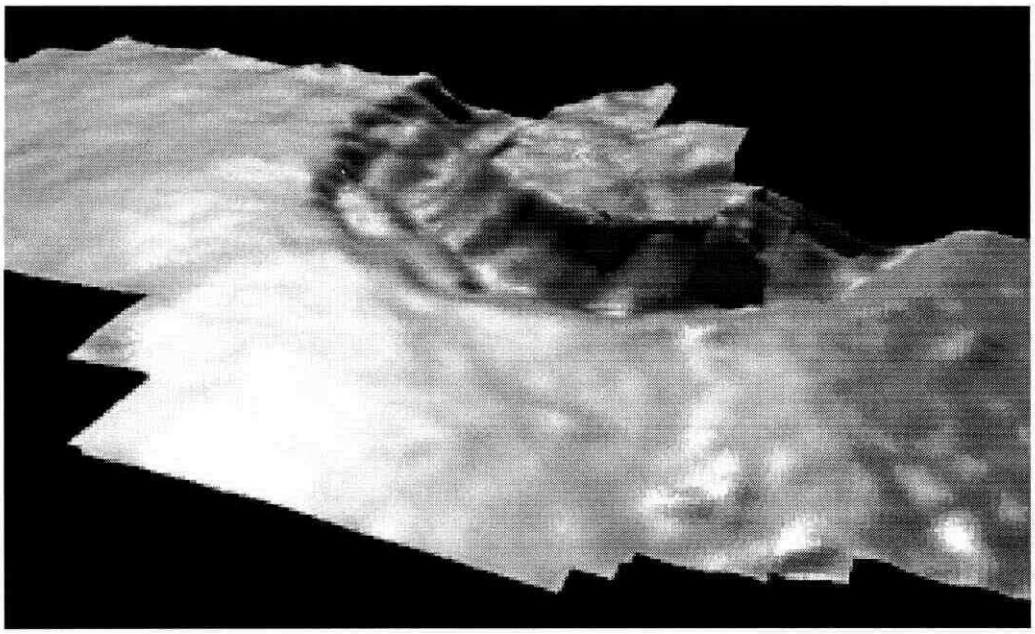


FIG. 3. Perspective view of a horizon using shaded polygon rendering

Volume Rendering

Although cube views provide more points of reference using perspective, the opacity of the cube sides prevent us from seeing the interior of the data volume. Volume rendering is a technique which represents the data volume with identical cube-shaped cells arranged in a fixed, regular grid. These equally sized, cube-shaped elements, called *voxels*, are each assigned reflective, and transmissive properties based on mapping function. Voxels can be considered 3-D versions of *pixels*, except pixels are generally assigned a solid colour, while voxels have varying transparency. Each data sample is represented by a voxel, just as each data sample in 2-D slice is represented by a pixel. When a collection of voxels is viewed from a distance, the square faces of opaque voxels merge together. We no longer see cubes, but a complex surface instead.

The choice of a colour mapping and transparency mapping function is of great importance. If too many of the voxels are displayed opaquely, we will be unable to see the interior of the volume. If too few voxels are opaque, we might miss seeing valuable information within the data volume. The transparency mapping function is usually non-linear, so that only high amplitudes are opaque. Interactive manipulation

of the mapping function is required for this technique to be useful, since the mapping function often needs to be adjusted when there is a change in the viewing angle or region of interest.

Since volume rendering involves viewing thousands of data samples in one view, it can sometimes be difficult to distiguish between nead and far voxels. Movement, either of the observer or the model, creates motion parallax. Motion parallax is extremely helpful for depth-cueing however it requires that the scene can be re-drawn quickly from a different perspective. Hardware capable of doing this is currently available, but the cost of this equipment is still rather high. Most hardware is also limited by the amount of memory available for viewing large datasets.

EXAMPLE

The example we present comes from 3-D seismic survey which contains a possible meteorite impact crater. The dataset was acquired from the James River region of Alberta. After processing, the data were loaded into a Landmark Interpretation workstation. Interpretation of the data was done using SeisWorks 3/D software, using the ZAP! horizon autopicker. One of the horizons looked intriguing when displayed using the wire frame perspective display on the interpretation workstation. In order to get a better look at this possible impact crater we decided to create a high quality picture of the horizon using 3-D perspective.

Almost all manufacturers of workstations now produce graphics hardware that can display a small 3-D model realistically in perspective in real time on the workstation screen. Since 3-D graphics hardware was unavailable, a software approach to generating the 3-D picture had to be taken. Our approach was to use software to create a picture file, and then display this picture file on the screen using standard 2-D graphics hardware. Although this approach is much slower, it can produce pictures of with greater realism: more sophisticated rendering algorithms may be used, now that a real-time display is not a goal. The software we used to render our pictures is called "Rayshade". Rayshade takes a 3-D scene description file as input, and outputs a picture file. There are several freely available raytracing programs in addition to commercial programs. We chose Rayshade because it generates extremely accurate, high-quality output and supports features such as heightfield objects (for horizon surfaces) and image mapping (for seismic slices). Rayshade is written in C and thus runs on almost all UNIX systems. This made it a practical choice for our work, since we were able to run the software on our Sun Sparc and DEC Alpha[1] systems.

The first task in generating an output picture with Rayshade is to describe the scene to be rendered. This scene is described by specifying the (x, y, z) location of objects to be drawn. Rayshade supports geometric objects such as spheres, boxes, planes, cones, and heightfields. A heightfield is a collection of height values which define the height of a surface over a regular grid of positions. We make use of a heightfield object to represent the picked horizon. Most raytracing programs subdivide the heightfield surface into a number of triangles and render the surface as a collection of interconnected triangles. Since these triangles are extremely small the surface looks continuous and smooth when viewed from a distance.

[1] A DEC 4000 model 610 AXP computer and DEC 3000 model 500 AXP workstation were loaned to the University of Calgary and used in this study.

After the scene is defined, there are many other parameters which must be specified before the final picture can be generated. The observer's eye position must be defined as well as the location in the scene upon which the picture will be centred. The method of illumination is also controllable by specifying where light sources are located. By placing a point source of light well away from our heightfield object we can simulate illumination by sunlight. Additional ambient light is also added to simulate light from the sky. The final parameter we specify is the size of the output image in pixels.

The scene is described using Rayshade's scene description language. This language is very simple. For example to tell Rayshade to create a picture that consists of a heightfield we create a file with the following Rayshade commands:

```
eyep 10 10 10
lookp 20 20 20
heightfield cambrian.hf 0 0 0
```

The *eyep* command tells Rayshade where the observer's eye is positioned. The three numbers following eyep describe the location in (x, y, z) space. The *lookp* command tells Rayshade where the observer is looking. The *heightfield* command specifies that a heightfield should be drawn at the given location using data from the provided file name. Since a heightfield may contain thousands of data values, Rayshade reads the height data for a heightfield object from a separate binary file.

To create this heightfield file from the Landmark horizon database we used Landmark's SeisWorks/3D software. While running SeisWorks/3D one can export the horizon to an ASCII file from the Map View window. This operation outputs a single line for each point in the horizon. A small program was written to reformat the text file into a binary file in Rayshade heightfield format.

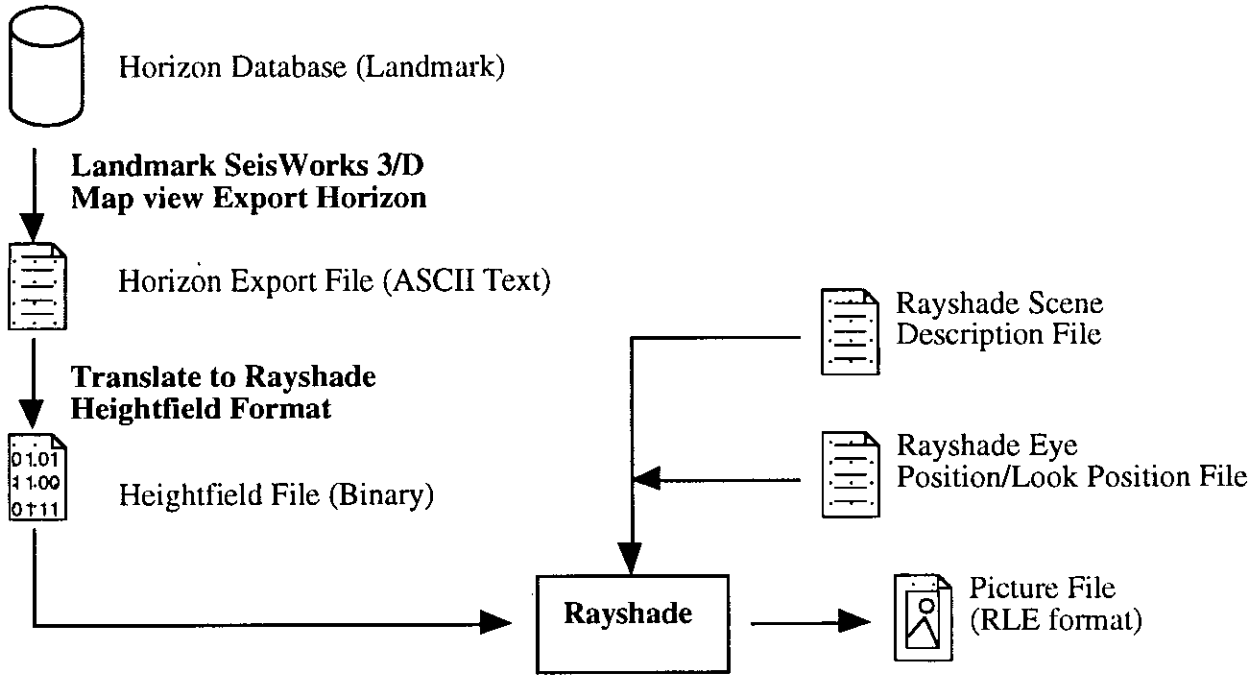


FIG. 7. Processing Flow

Creating a movie

Having succeeded in creating a single pictures of our horizon we decided to create an animated fly-through of the horizon. An animated movie allows different areas of the horizon to be observed in detail. The movement also creates parallax so that image depth can be better appreciated. We use the same Rayshade software to create hundreds of frames of the same scene, changing the *eye* position and *look* position as our imaginary plane flies through over the crater below. Creating a movie is significantly more complex since we must select not one, but hundreds of *eye* positions and *look* positions.

Choosing a flight plan and directing the camera angles is simplified if done interactively. Since the images could not be created in real time, we designed our flight plan using a map view of the horizon. This map view was exported from Landmark SeisWorks 3/D, translated into a standard XPM format image file, and placed into drawing program. Inside the drawing program the flight path was drawn using an continuous set of line segments. Dashed arrows were used to indicate the camera target. The head of the arrow points to the location where the camera should look while the tail points to a flight path control point. This ties the camera target to a particular moment/position in the flythrough. The plane's altitude is indicated with solid arrows. The altitude is determined by the length of these arrows and the arrow direction is ignored. Lastly, the flight speed is indicated with numbers placed at various points along side the flight path. All these control symbols were drawn using a free X-Window drawing program called *TGIF*. TGIF was chosen because it runs on our workstations and produces output files that are easily processed by other programs.

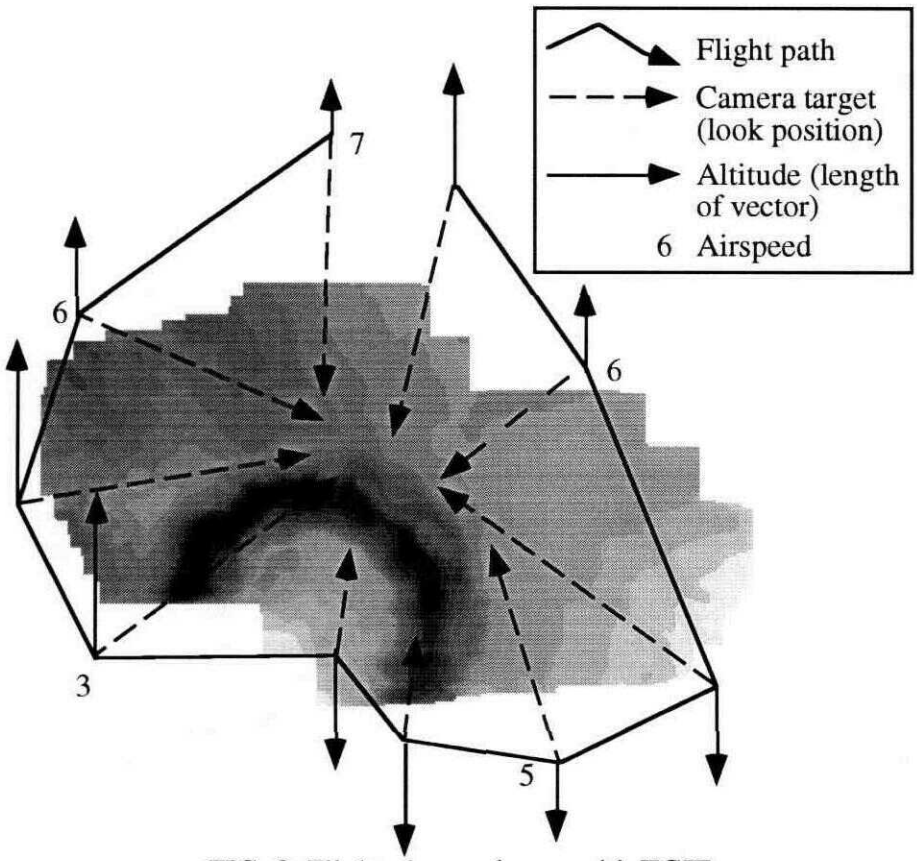


FIG. 8. Flight plan as drawn with TGIF

A program was written to read a TGIF file (depicting all the flight plan information) and translate it into a series of Rayshade commands for each frame in the movie. To make the flight path and camera target points flow smoothly, a bicubic spline was used to interpolate a smooth path between control points. The program generates an output file of the form:

```
#ifdef FRAME1
      eyep 10 9 8
      lookp 0.2 0.3 0.4
#endif
#ifdef FRAME2
      eyep 9.9 9 8
      lookp 0.2 0.3 0.35
#endif
...continued...
```

In the above output example, we can see that different eye locations and look locations are specified for each frame number. A second program called *animfly* was written to execute the Rayshade program once for each output frame. Each time Rayshade is run, it is instructed to create a different frame number and output the frame to a different output file:

```
rayshade -D FRAME1 -O frame1.out
rayshade -D FRAME2 -O frame2.out
rayshade -D FRAME3 -O frame3.out
```

The animfly program can be instructed to run the Rayshade program on a number of networked systems. This way, each system dedicates itself to generating one frame at a time. The overall processing speed is greatly enhanced through this simple form of parallel processing. The generation of our movie involved 11 systems. The aggregate flow rate was 98% of the ideal parallel speed.

The final task in making the movie was to transfer the output images to video tape. This requires some expensive, specialized video equipment. The University of Calgary Computer Science Department has an computer system with a NTSC video output card connected to a single frame video tape recorder. The computer operates the video tape recorder through a controller interface. This equipment allows each frame to be copied to video tape in succession. The movie quality is very high and flicker free, since it is recorded at 30 frames per second. Very few systems, even ones with specialized 3-D graphics hardware, are able to generate displays at this speed. For this reason, video tape is currently the best way to produce presentation-quality animations.

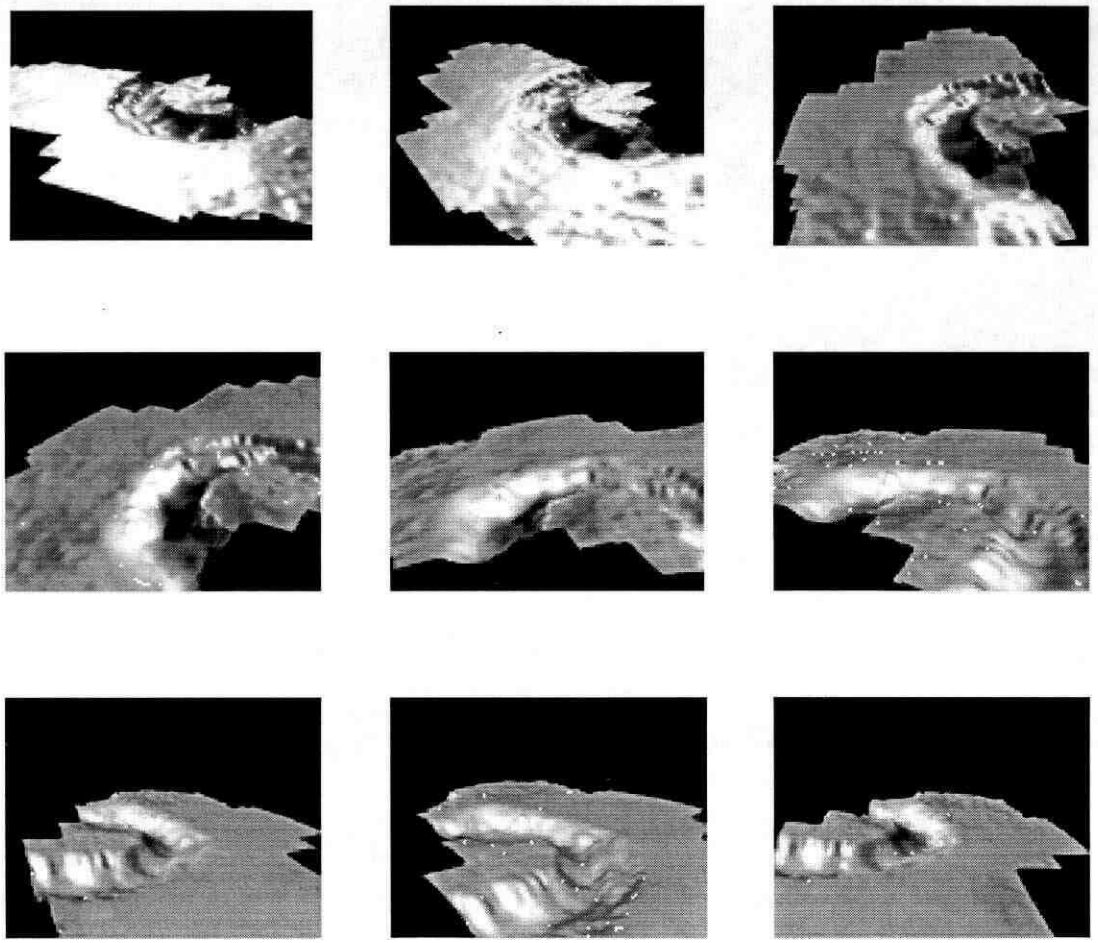


FIG. 9. A sequence of still images from the movie. The actual movie contains 100 frames for each still shown above.

FUTURE DIRECTIONS OF VISUALIZATION

Stereopsis

One effective way to get a better feel for image depth is to make use of stereopsis. Stereopsis can be achieved for a 2-D display by computing an image for each eye and sending each image to the respective eye. Currently, the best technology for stereo vision is a system which employs shutter glasses. These glasses are worn like sunglasses and contain an electronically controlled shutter-lens. This lens polarizes light in one direction, and then the other as directed by an electrical signal. Each eye has a second, differently polarized lens in front of it. When the polarity of the light is enabled in one direction, the left eye's view is clear and the right eye's view is obstructed. When the polarity is reversed, the clear and obstructed views reverse respectively. A video screen in front of the user alternates between showing left eye and right eye images synchronously to the shutter glasses. Since the switching takes place very quickly, the user does not notice the flickering image. The result is a very realistic stereo view of a 3-D scene.

Computing images for stereopsis involves drawing the image from two slightly separated points of view. For static display computing two images is not a problem, but significant compute power is necessary to render a complex scene in stereo. Video hardware must also be specialized, since it must be able to alternate between two images many times per second.

Virtual Reality

The term *virtual reality* is associated with highly realistic, real-time visualization combined with input devices that have intuitive human interfaces. When visualization responds quickly to human stimuli, the feedback to the user makes the visualization seem almost real. The technology of virtual reality will allow us to explore a 3-D dataset in a natural and intuitive way that exploits our highly developed skills in visual-pattern recognition. Using real-time interaction with the dataset, along with head-mounted, direction sensing, stereoscopic viewing gear, the geophysical interpreter will be able to look at the dataset in a much more interactive fashion. Head-mounted viewing gear allows the viewed image to change as the observer changes viewing target. Slight motions of the head contribute to *head-motion parallax* which greatly improves depth cueing.

In addition to improvements in 3-D scene viewing, enhanced control devices improve the interaction. Some promising control devices include pointing devices with 6 degrees of freedom, such "floating mice" or "space balls". These devices allow the user to translate the observed object in three dimensions, as well as rotate it to any orientation. Another type of pointing device is a data glove. A glove allows for the same translation and rotation control, as well as the ability of the computer to understand the user's hand gestures. Hand gestures could be used to indicate the direction of traversal through the data set. They could also be used to speed up typical seismic processing operations such as 3-D picking. Horizons could be picked with sweeping hand motions (for course picking) and small finger motions to finely adjust picks.

CONCLUSIONS

The search for improved methods of visualizing 3-D data must follow the path of improving realism. The more we can create an illusion of being actually underground, the better we will understand the data at hand. Techniques such as perspective views, volume rendering, motion parallax, stereopsis and improved user control are all steps toward duplicating the sensory experience in the real world. We have shown that partial solutions to the visualization problem can be obtained using today's technology. With the high speed of technical advancement, the future for 3-D seismic visualization looks very promising.

ACKNOWLEDGEMENTS

We would like to thank following: Ms. Helen Isaac for providing us with the interpreted horizon of the James River possible meteorite impact crater, Husky Oil for supplying the James River dataset, the many authors of Rayshade and the Utah raster toolkit for supplying their software for free, and Dr. Brian Wyvill and Mr. Mark

James for the use of their video transfer equipment and their valuable technical assistance in making the movie.

REFERENCES

Foley, J et al., 1990, Computer Graphics: Principals and Practice: Addison-Wesley Publishing Company, pp1035,pp616