Interactive Movie Machine
User's Documentation

Richard Ottolini

By attaching some "knobs and whistles", we have made our movie machine more enjoyable and rewarding to use. "As fun as a video game!", claims one anonymous observer. Now like a pilot in his cockpit, the SEP geophysicist can choreograph a display of light and motion as he probes the inner secrets of the earth.

The applications of interactive graphics to seismic data processing are potentially vast. We have only explored a small corner of it at SEP and only intend to explore the techniques which will be of the greatest use to us. First, we have concentrated on animation because we have a particularly able hardware set-up for experimenting with such. Second, we have directed our attention towards data processing as opposed to geologic interpretation. Movies help identify new phenomena in the data and provide a new way of looking at the old. The goal of seismic data processing is to enhance such phenomena when it reveals new geologic information, else suppress it when it obscures the geology.

This paper describes our current movie program for one who would actually sit down and use it. At the end of this document is a discussion of programming considerations which shaped the movie software and suggestions for future evolution.

Movie Machine Console

The movie machine user's basic hardware consists of the AED* color monitor, AED keyboard, and any one of the nearby ordinary CRT terminals. Other devices are attached to the AED for the purposes of recording movies, but are not essential for interactive use. The user logs in on the CRT terminal and executes the program *MOVIE*. The movie will soon appear on the AED color monitor. Movie parameters will be printed on the CRT terminal. Most

*Advanced Electronic Design model 512 color graphics terminal

interactive communication with the movie program will be through the CRT terminal keyboard. The other major interactive component is the joystick built into the AED and two black keys adjacent to the joystick. The rest of the AED terminal keyboard is not used.

Parameter Display

A parameter display appears on the CRT terminal when the *MOVIE* program is activated. A actual snapshot of the parameter display appears in figure 1. It is constantly updated as the state of the movie changes due both to the movie program and user interaction.

```
author= Rick          title= BARENT'S SEA SHOT PROFILES

    plane= +shot                    plane= +time          L  slice= Y-plane
    nz= 48                          thick= 4.1            O  dir= forward
    dz= 100                         p0= 0                 O  slow= 0
    z0= 0                           depth= 1.84           P  frame= 230
    /                             S /
D  /      horz= -geophone       R  /      horz= -geophone
A  ------ nx= 48                E  ------ width= 2.4e+03
T  :      dx= 50                E  :      h0= 273
A  :      x0= 273               N  :      hzoom= 10
   :                               :                      C
   vert= +time                    vert= +shot            O  bytes= 1179648
   ny= 512                        height= 4.8e+03        L  color= intensity+
   dy= 0.008                      v0= 0                  O  gain= 1
   y0= 0                          vzoom= 10              R  mask= 7654321_

    FILES: in= /data/Csaga        par= /data/Hsaga       overlay=
   INPUTS: joyx= 1                joyy= 0.996            key= S

JOYSTICK: p: pan, z: zoom, d: frame, s: slowness, g: gain,  SLICE: X:, Y:, Z:
LOOP: f: forward, r: reverse, b: both ways, j: step+, k: step-, a: new start,
e: new end,  AED: -: disconnect, +: reconnect,  SNAPSHOT: S:
COLOR: I:, i:, F:, T:, B:, W:, G:,   7,6,5,4,3,2,1,0: toggle color bit
```

FIG. 1. Parameter display from actual movie.

The main parts of the parameter display are:

(1) In the upper left is the coordinate system diagram of the input data, labeled DATA. The fast coordinate is x and slow coordinate z. For each of the three dimensions, the number of samples, initial value, sample spacing, and coordinate name is given. See the section on input data for a more thorough description. These parameters remain fixed during the duration of the movie program.

(2) On the right is a second coordinate system diagram labeled SCREEN. It gives the dimensions and orientation of the portion of the data cube which is displayed as a

movie. The movie cube is a sub-cube of the input data cube. The front surface of the cube represents the screen display, while the depth dimension is the movie coordinate. The name, origin, and scaled size of each of the three dimensions is given. These parameters are constantly updated by the movie program. The axes rotate as the direction of slicing through the movie cube changes.

(3) In the upper left hand corner are parameters giving the state of the film loop such as slice direction, loop direction, speed, and frame count.

(4) On the middle right is color information.

(5) Above and below the two coordinate systems are general label and file information. Also the state of the joystick and keyboard inputs are given.

(6) In the lower part of the screen is a list of the interactive keyboard and joystick functions of which more will be said in the following sections.

(7) At the bottom is a space for program messages.

Should the parameter display be injured at anytime during the course of the movie program, typing CONTROL L will recover it.

Typing the key *P* will save an image of the parameter display in an uniquely named file. The file is named *movXXX.XX.XX* where the X's refer to day of the year, hour, and minute the file was made.

Types of Interactive Input

(1) *Keyboard interrupt.* Typing a single key on the CRT keyboard will change the state of the movie program and/or the function of the joystick. A list of key codes is printed at the bottom of the parameter display (see figure 1). This is the most common method of program interaction.

(2) *Joystick.* The joystick may be programmed to do about a half dozen different functions by keyboard interrupt. The joystick is used to search through some range of parameter space. Some functions use the joystick in a 2-D mode, while others use it in a 1-D mode (horizontal). The two black buttons to th left of the joystick change its operation slightly. When the *vernier* is on, the joystick returns smaller increments. However, the tradeoff is that the joystick only has about ten percent of the range before the vernier was turned on. The *rate* button causes the joystick to continuously change its returned value even when it is not being moved. The rate and direction of this change depends upon the position of the joystick.

(3) *Typed in parameters.* Some keyboard interrupts allow the user to type in a parameter following a prompt message. This method is rarely used.

(4) *Programs sharing the AED.* Occasionally the AED may need some special servicing which the *MOVIE* program cannot provide. Two keyboard interrupts - and + disconnect and reconnect the AED from the *MOVIE* program. The state of the current movie is not lost then while the AED is being serviced.

Input Data Description

The input dataset is a three dimensional array in AED format. AED format is a set of one byte long color codes arranged in scan line order. The program *taplot* converts converts a 3-D array of floating numbers into color codes and transposes them into scan order.

The format of the input dataset is specified by parameters typed in after the command *MOVIE*, or kept in a parameter text file. Wherever the parameters may be specified, they are of the form *name=value* (example: nx=48). Typing the command *MOVIE* with nothing following it returns a list of parameters which may be specified.

In= is the name of the input file which defaults to standard input. *Par=* is the name of the optional parameter file. *Overlay=* is the name of an optional file containing text pen commands which will create a stationary line drawing overlaying the movie images. The movie is identified by the optional *author=* and *title=* parameters.

The integer dimensions of the dataset array are specified by $nx=$ (the fast, horizontal, scan direction), $ny=$ (the vertical direction), and $nz=$ (the slow, frame direction). $Nx=$ and $ny=$ are required while $nz=$ defaults to the entire dataset. For scaling purposes in the parameter display, the sample interval and origin of each dimension are given by $dx=$, $dy=$, $dz=$, $x0=$, $y0=$, and $z0=$. The sample intervals default to unity and origins default to zero. *Horz=*, *vert=*, and *plane=* take label names for the three coordinate axes. Multiple word names are put between double quotes. Axis names default to X, Y, and Z. The synonyms h for x, v and t for y, and p for z are recognized in order to maintain compatibility with other array manipulating programs.

There are no bounds on the dataset dimensions. However, the AED screen only permits viewing of about 500 samples on a side. Furthermore, the maximum movie frame window is one quarter of this area. Movies less than 4 million samples altogether run completely in core and are relatively fast. Movies between 4 and 10 million samples run in slower virtual memory. Movies greater than 10 million samples run only from disk and are about three or four times slower than fully in-core movies.

Movie Cube Reorientation

The direction along which *MOVIE* slices through the data cube may be changed on the fly by typing the keyboard interrupts X, Y, or Z. The default is going through the Z direction. For example, if a 3-D dataset is organized as seismic lines, the options X, Y, and Z generate cross-line, depth slice, and in-line movies. The state- frame, speed, etc.- of each movie orientation is remembered upon re-entering the same orientation. Options X and Y require in-core transposes and are somewhat slower than the Z option. These options are not available when the movie is so large (>10 million samples) that it must be run from disk.

Split Screen Movies

The program *SMOVIE* runs portions of two movies on the same screen. The AED screen is split into right and left halves. When one changes the direction of how the movie slices through the data cube, the new new movie is shown on the other half of the screen. The last frame shown from the previous movie remains on the old half of the screen. The movie orientation can be changed as often as desired with the new movie switching to the other side of the screen. A yellow line (which moves on the stationary frame, but is still on the movie side) indicates where the two movies intersect. In just about every other aspect *SMOVIE* operates just like *MOVIE*.

Film Loop Control

The basic direction of a film loop is reset by typing f for forward, r for reverse, or b for both ways. The default initialization is forward for each of the X, Y, and Z movies.

The loop is put under discrete control by typing d for joystick, j for step forward, and k for step backwards options. The joystick scales to the length of the film loop. The vernier key may be needed resolve a specific frame if the film loop is long. The rate key allows the joystick to change both the speed and direction of the movie.

The speed of a film loop is altered by typing s. Then the joystick adds a delay from 0.0 to 2.0 seconds between each frame. The default speed of the film loop depends on (1) how busy the computer is, (2) the number of data samples in a frame, and (3) whether the movie cube is being shown in a transposed direction.

Typing a or e causes the beginning or end of the film loop to be set to the current frame number.

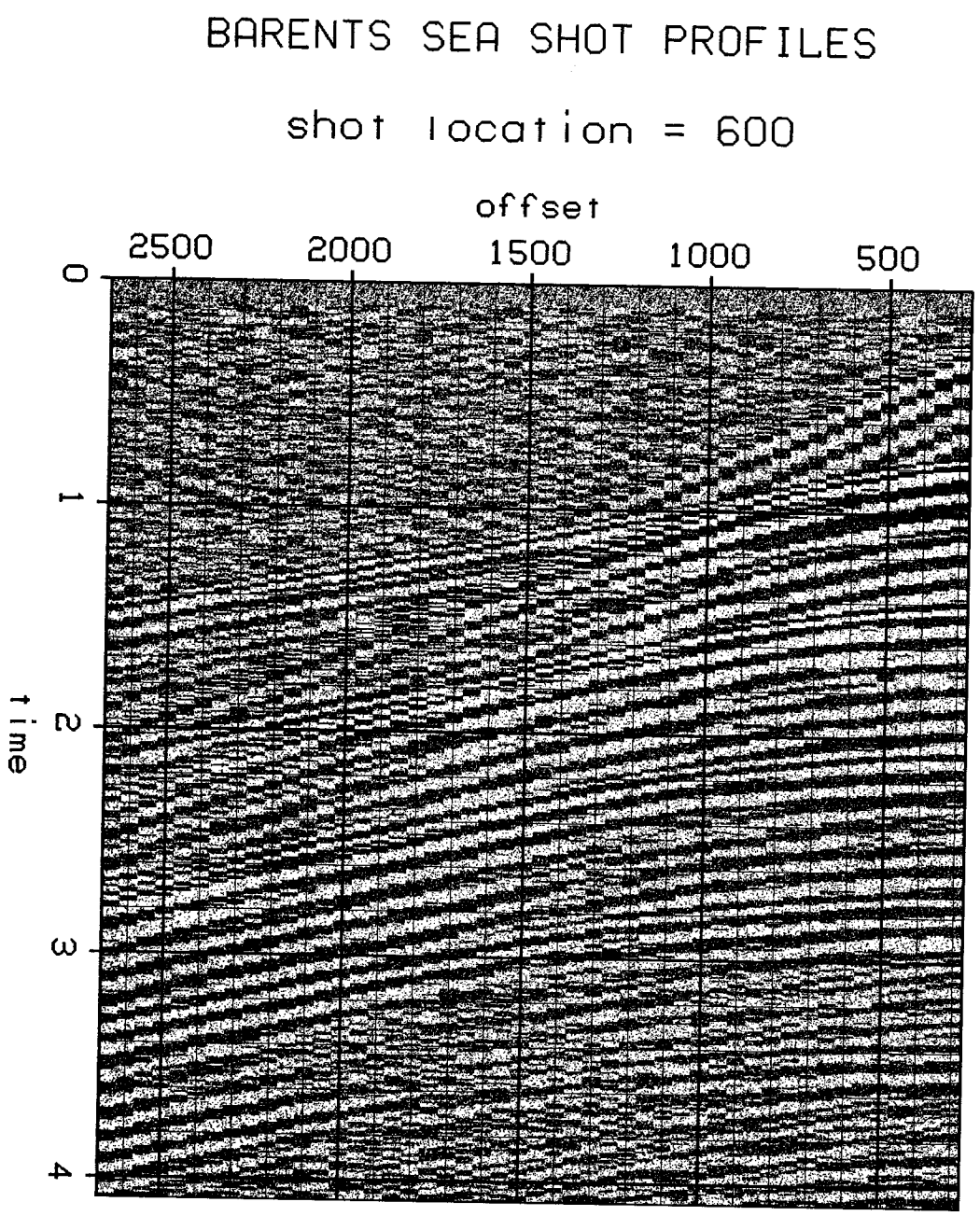


FIG. 2. Hardcopy of shot profile frame from dataset described in figure 1. (Key option *Z*.)

Control Within a Frame

The zoom changes the size and aspect ratio of the display. Typing *z* enables the 2-D joystick to change the size of a movie image horizontally and vertically. The center-of-screen coordinate remains constant as data expands off of the AED screen. If a display dimension is less than screen size, the image is centered within the screen. The movie speeds up as the number of data samples displayed decreases.

Typing *p* for pan enables the joystick to change the location of a window within the plane of data. The joystick is scaled so that its extremes match the boundaries of the data plane. If the dimension of the window is less than the AED screen size, then no panning takes place in that dimension.

Color

The color schemes are loaded by typing a keyboard interrupt:

Key	*Name*	*Positive*	*Zero*	*Negative*
I	intensity+	white	gray	black
i	intensity-	black	gray	white
F	flag	red	white	blue
B	bi-color	red	black	blue
W	surf	white	blue	black
G	GSI	red	white	black

The default color is *I* unless *color=no* is typed in the command line. Typing *g* puts the gain of the seismic data under joystick control.

Typing the digits 0 through 7 turn the respective bit in the color code on or off. The effects of such masking depend upon the loaded color spectrum. Turning off the low order bits makes the data appear more clipped. If more than one dataset have been combined into the same color code (for example, a "before" in the high 4 bits and "after" in the low four bits) then the mask choice controls which dataset may be displayed.

Hard Copy

Typing *S* stores a copy of the parameter display in a text file. The file is named *movXXX.XX.XX* where the X's refer to the day of the year, hour, and minute when the file was created. Figure 1 is a sample output.

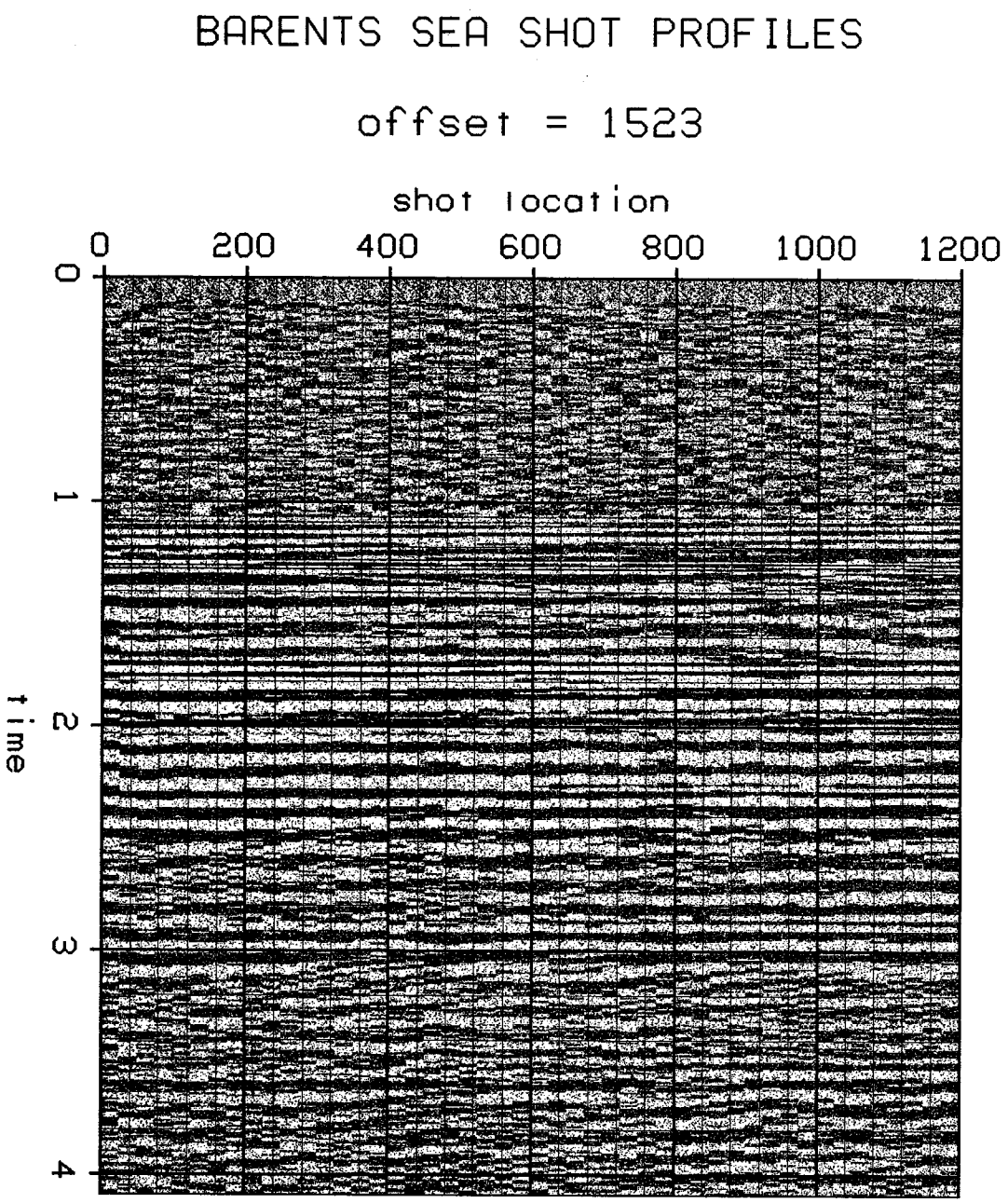


FIG. 4. Hardcopy of constant offset section from dataset described in figure 1. (Key option *X*.)

Typing P prints an image of the AED screen on an electrostatic plotter. The image is according to the I intensity scheme described above. Another paper by Ottolini in this SEP volume describes how these images are made. A labeled set of axes also comes out. Figures 2 to 4 are sample outputs.

Auxiliary Programs

The follow programs are used either before *MOVIE* or when the AED is disconnected during the *MOVIE* program.

(1) *Taplot* converts floating point seismic data into AED format.

(2) *Color* allows the user to directly specifying the red-green-blue color intensities for coloring seismic data.

(3) *Apen* executes *vplot* format vector commands to draw lines on the AED. It can be used to make overlays.

(4) *Aedplot* displays labeled, full resolution movies suitable for filming. It does not run in real time, however.

(5) *Atoe* puts a 2-D AED image on one of the electrostatic plotters. The input dataset must be in AED format (described earlier in this paper.)

(6) *Azoom* runs AED language commands from a CRT keyboard. It can be used to repair bugs during the movie program.

(7) *Aedreset* trys to recover the AED if it locks up because of software failure.

Programming Considerations: Convenience

An interactive function is of greatest use when it is most convenient. Convenience means fast and easy. For example, the transposed movies were almost never made when the dataset had to be transposed with a separate program. Within a couple of days of installing the transpose option in the *MOVIE* program, a new way of identifying velocity anomalies in field data was discovered (See Rocca and Ottolini, this SEP report).

Often the user will have to explore some space of parameters when tuning the movie display. In this case, convenience means choosing the appropriate degree of flexibility in the search strategy. In more concrete terms, this means some functions are better implemented by a keyboard interrupt and others by a joystick. Rarely have we found it useful to directly type in a number. Other technologies such as voice command recognition and graphical tablets may be useful for certain types of graphical interaction.

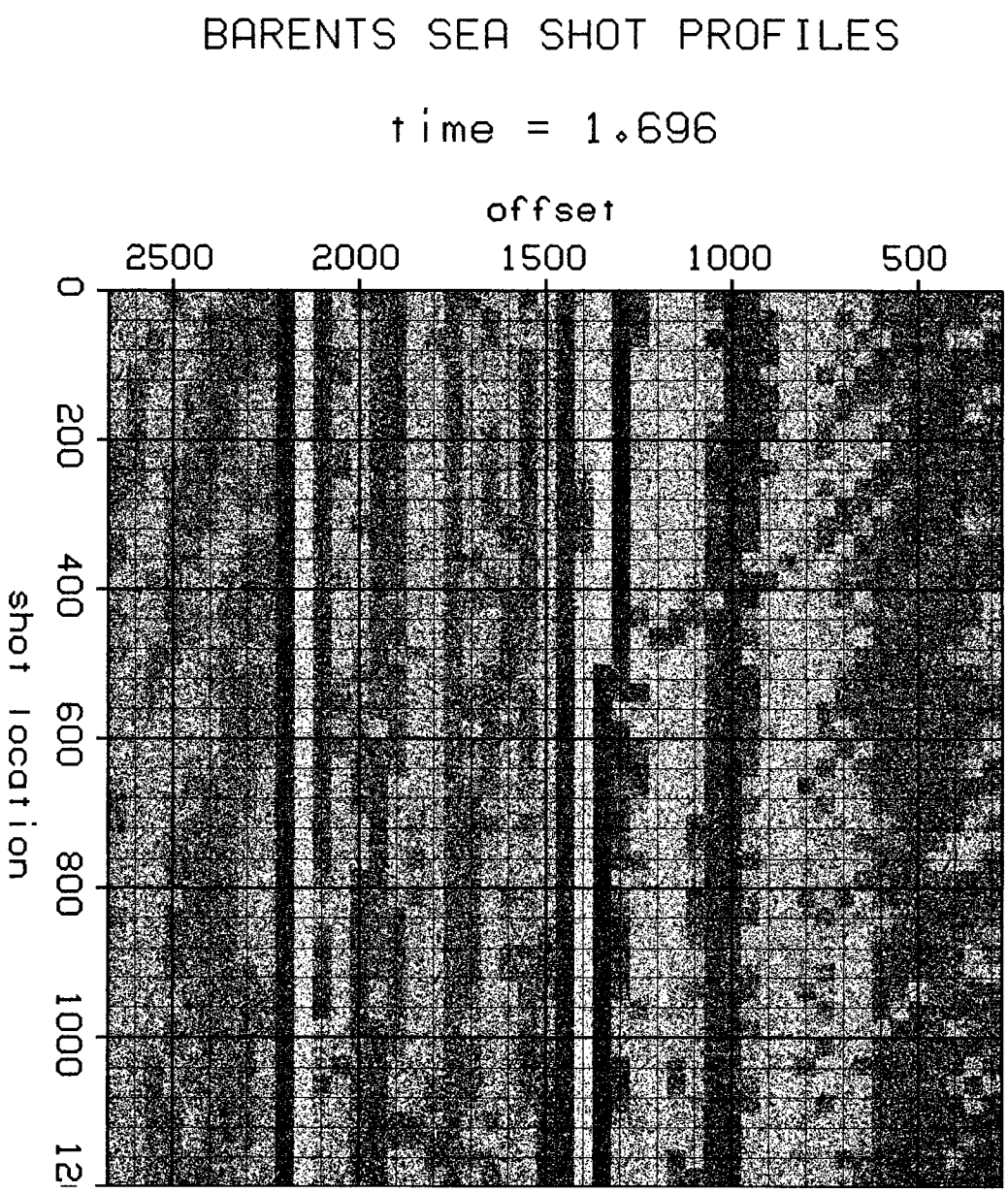


FIG. 3. Hardcopy of isotime cross section from dataset described in figure 1. (Key option *Y*.)

Migration of Hardware into Software

Our original conception as to what a movie machine could do was largely constrained by the functions built into the hardware. However, with experience our imaginations have grown and needs have superceded the hardware functions. One method of meeting these needs is to transfer functions to software. For example, the pan function built into the AED uses the joystick to explore the internal AED memory. However, if the dataset is smaller than the AED memory one can pan to blank areas. Conversely, if the dataset is larger than memory, then the hardware pan can't view certain portions of the dataset. Therefore, we replaced this function with a software pan scaled to the size of the data.

Eventually, we run into hardware barriers too difficult to surmount with software, or the software implementations become too complex. We feel that the latter is the more common case at SEP. Some hardware features we will seek in our next graphics terminal include:

(1) *Larger screen size.* We would like more than one cross section of the input data cube at one time. However, we probably wouldn't animate more than one screen area at a time. Our current 512 x 512 sample screen size is too small to display multiple cross-sections in a satisfactory way, though it is adequate for a single cross-section.

(2) *Faster data transfer.* We animate seismic data by continuously sending images to the graphics terminal. Currently our computer limits us to an animation screen size of 250 by 250 at 6 frames per second.

(3) *Improved interactive input devices.* Our joystick has two degrees of freedom and sometimes we would like three (for example, red-green-blue color control). If the space of parameters is relatively large (>100), then our joystick can either sample it coarsely (not every possible gradation), or the search range must be limited. Other mechanical devices such as 3-D joysticks or trackballs may be better for certain functions.

(4) *Faster computer.* The logical extension of realtime interactive displays are realtime interactive processing and interactive interpretation. Unfortunately, some of these applications require a more powerful host computer than we have.

We are satisfied with our present hardware flexibility to manipulate color and the AED graphics command language.

Virtual Devices

One particularly guiding concept in the design of the *MOVIE* program has been that of the virtual device. A *virtual device* is a model of some I/O device which the program uses. The virtual device is constructing with data structures and subroutines. For example, our electrostatic plotters are modelled as vector drawing plotters by special software.

A virtual device simplifies programming by isolating all of the device dependent code in a separate subroutine. Then the programmer need not have to see the nitty-gritty of machine dependent commands, optimizations, etc. This feature makes it easier to modify the *MOVIE* programming which is coordinating a half dozen input-output channels at any time. Also, if one buys a new graphics terminal, then only the subroutine need be changed.

A virtual device modifies the apparent functionality of an I/O device. Complicated functions can be simplified. New functions can be simulated in software. The *pan* function described earlier is an example. It amplifies the hardware pan. Because the software pan is scaled to the size of the dataset, it never moves to a blank area of the screen if the dataset is smaller than the screen size and it windows through datasets larger than the screen size.

Virtual devices in the *MOVIE* program include:

(1) *Parameter display.* The parameter display looks exactly like a 24 lines of the program. If one wants to reformat the display design, one just edits this piece of program. The parameter display fits on any of the CRT terminals we have, even though screen sizes and other characteristics are slightly different.

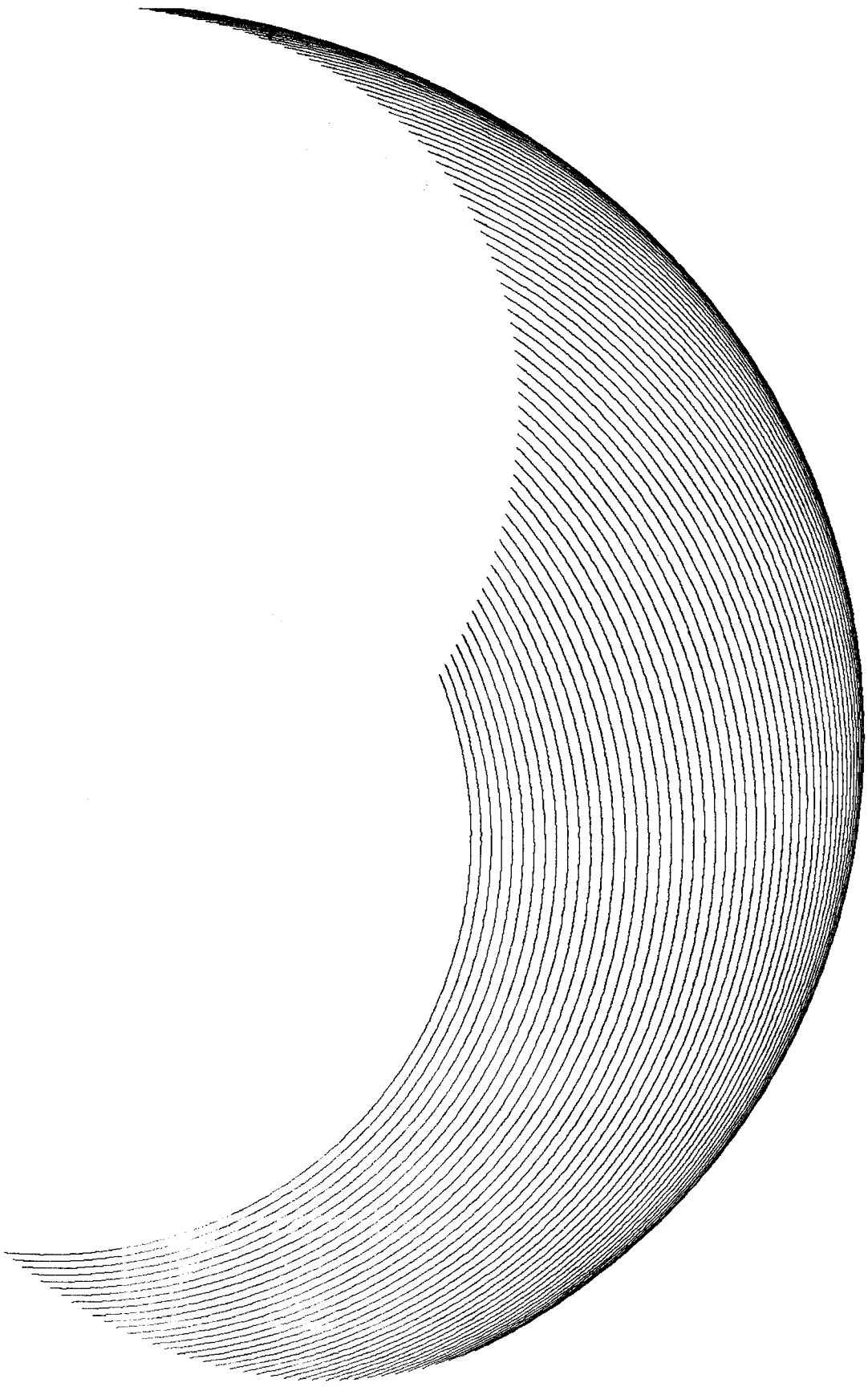
(2) *AED screen.* The AED screen appears the same size as a slice of the dataset, irrespective of which one is larger.

(3) *Movies. MOVIE* can slice through the data cube in any of the three cartesian dimensions. When it returns to a previous direction it remembers where it left off. This is done by simulating three movie machines simultaneously. Two are suspended when the third is running. This makes it easy to implement multiple screen formats in the future.

(4) *Input.* It does matter what kind of device sends interactive input in the program as long as it can send interrupts (currently the keyboard) and a range of parameters (currently the joystick). Tablets, track-balls, voice recognizers, etc. could be substituted with no major program changes.

The Future

Several future directions for development have been mentioned: multiple sub-movies, interactive color, interactive processing. We will continue to focus upon the the use of animation for studying processing algorithms. We also have to weigh the returns against increased programming effort. The potential of movie machine seems limited only by our imagination.

Acknowledgements

Many people have contributed to the evolution of the *MOVIE* program since last described in the SEP reports. Bert Jacobs and Chuck Sword figured out how to use the joystick in an effective way. Ron Ullmann is our interactive color magician. Jon Claerbout and Fabio Rocca made valuable suggestions as to interactive parameters and parameter displays.