Hypercube viewer

Robert G. Clapp, David M. Chen and Simon Luo

ABSTRACT

Efficient viewing and interacting with multi-dimensional data volumes is an essential part of many scientific fields. This interaction ranges from simple visualization to steering computationally demanding tasks. The mixing of computation and interpretation requires a library that allows user inputs and generated results to easily be transferred We wrote `Hyperview` in C++ using the QT library to facilitate this interaction. We describe the graphical user interface to the library and the basic design principles. We demonstrate the flexibility of the underlying libraries through a simple semblance picking application.

INTRODUCTION

Viewing and interacting with multi-dimensional volumes is necessary when working with 3-D data. SEP wrote its first movie program 28 years ago and has continually expanded on this initial idea (Claerbout, 1981; Sword, 1981; Ottolini, 1982, 1983, 1988, 1990). These movie programs have progressed from simply showing a series of frames to allowing greater and greater levels of interactivity.

Interactivity can take several forms. Numerous attempts have been made at SEP to take human input to geophysical algorithms. (Claerbout, 1987, 1991) built interactive filtering tools. Several authors van Trier (1988); Berlioux (1994); Clapp et al. (1994); Mora et al. (1995) have built interactive tools for velocity analysis. Cole and Nichols (1992, 1993) built a generic X11 based toolkit for interactivity.

More recent efforts have been focused on expanding Rick Ottolini's viewing program `Ricksep`. Clapp (2001) added the ability to view multiple datasets simultaneously and built hooks to allow processes like interactive NMO analysis. Chen and Clapp (2006) expanded on this work by adding the ability to overlay datasets and including the capability to display well logs and other data types. Two problems more further expanding `Ricksep` challenging. First, `Ricksep` was written in C rather than an object oriented language more appropriate to graphics. Second34, it uses the Motif graphics library, whose future is uncertain.

In this paper, we present a new slice viewing program `Hyperview`. `Hyperview` is written in C++ using the QT [1] library. It preserves almost all of the capabilities of `Ricksep` while adding numerous additional interactive features. This paper is

[1]http://www.trolltech.com/products/qt

broadly broken into three parts. The first part gives an overview of `Hyperview`. This is followed by a description the various menu and keyboard options available. The last portion of the paper is devoted to the design of the viewer and ideas for how it can be extended.

OVERVIEW

`Hyperview` can be more properly thought of as the front end to a library that specializes in viewing and interacting with hypercubes. `Hyperview` is a python script that reads in the files to display, guesses their type based on their suffixes and then calls the underlying program `Hyperview`. Appendix A describes the command line options to `Hyperview`. The python script calls `qt_cube`, a C++ program that takes as input one or more hypercubes of the same size. The library allows significant additional flexibility that is described in the design section of this paper. The remaining portion of this section will be limited to describing the default behavior, rather than the potential flexibility allowed by the library.

`Hyperview` displays one or more regularly sampled datasets that have between two and five dimensions. For `Hyperview's` default behavior, each dataset have the same number of dimensions and number of elements along each axis. The datasets are read into memory and stored as 8-bit unsigned char. Two different windows, status and display, are brought up when `Hyperview` is invoked.

Windows

The display window (Figure 1) contains one or more views of the dataset(s) that have been loaded into memory. The number of views is controlled by the `nviews` command line argument. Each `view` can choose to see any of the loaded datasets. Using the mouse, the user can navigate through up to three dimension of the hypercube. When working with four or five dimensional datasets the user has the ability to select which three dimensions to work with. All of the views are linked together, by moving to a new location in one view you will be taken to the same location in all other views.

The status window (Figure 2) is broken into two parts: information and history. The top portion of the window information about the datasets currently shown. You can view: axes information (origin, sampling, number of samples, and axis label), the names of the datasets that were read in, which dataset you are currently viewing in each view, the position in the hypercube, and the data value in each dataset at the current position. The history window records every mouse and menu action taken and is more fully described later.

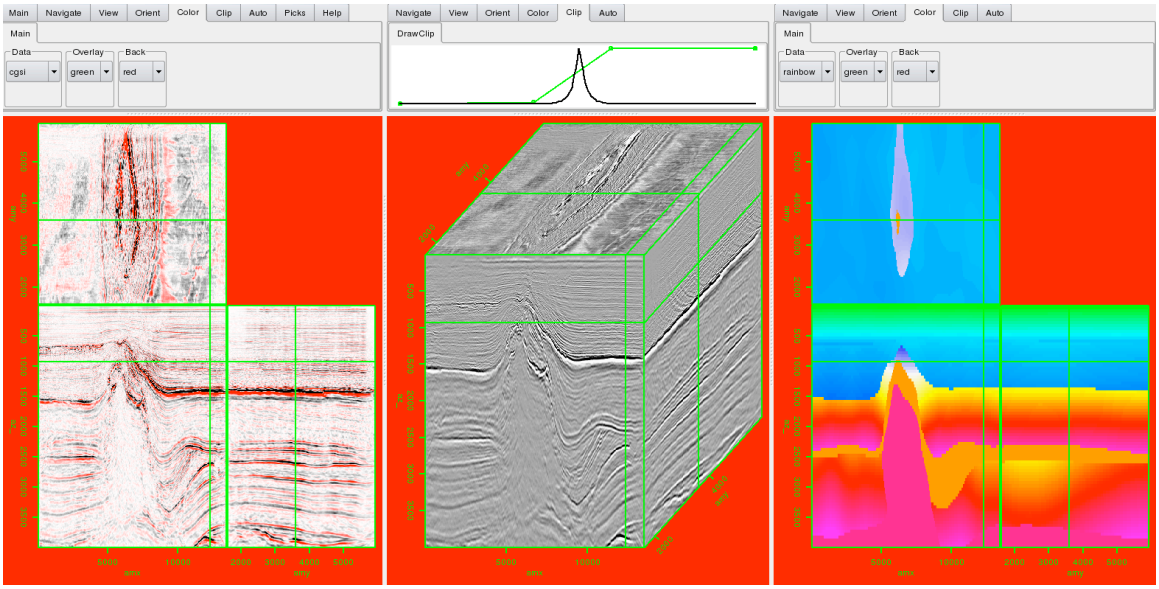

Figure 1: The display window with three different views. In this case three different datasets are being displayed. [**NR**]

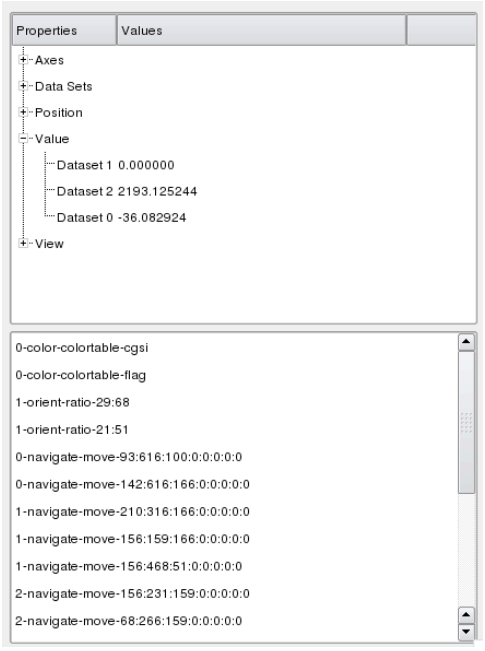

Figure 2: The control window. The top of half provides information about the dataset(s). The bottom shows the last ten actions performed by the user.[**NR**]

Data

`Hyperview` has significant flexibility in the type of data it can read. Its default behavior requires that all data volumes be the same size (e.g. same number of dimensions and same number of samples per axis). The object oriented nature does not make this a requirement; the velocity viewing example presented later demonstrates this flexibility.

Currently `Hyperview` supports five of the more common seismic data formats but can easily be expanded to read almost any other format. It reads SEPlib (and RSF) regular cubes in both byte and float format. It can read float formats of SEG-Y, SU, SeisSpace and the scaled integer format of SeisSpace. By default when reading float data `Hyperview` reads the first 5 MBs to find clip parameters and stores the entire cube in byte format based on the clip information. Adding `float_format=1` to the command line it will store the data as floats rather than bytes. This feature is useful for both clipping and for actions that require more precision than bytes.

When reading in float data, the program looks for a series of command clipping options. It first looks for `bpclip` and `epclip`, corresponding to a beginning and ending clip percentile. It next looks for `pclip` which corresponds to a percentile clip based on the absolute value of the data. It then looks for minimum and maximum clip values `bclip` and `eclip`. Finally, it looks for `clip` which corresponds to $bclip = -clip$ and $eclip = clip$. If none of these parameters are found, it defaults to $bpclip = .5$ and $epclip = 99.5$.

MODES

A standard mouse has three buttons and three potential actions (click, double-click, click-move-release). As a result, only 9 unique actions are possible. `Ricksep` gets around this limitation by using keyboard modifiers as the behavior of the mouse changes based on a key being simultaneously pressed on the keyboard. This approach is effective but somewhat cumbersome. In `Hyperview` we take an alternate approach which we call 'modes.' Currently two modes are defined, navigation (the default) and picking. To switch to the picking mode the user hits `Ctrl-p` on the keyboard and to switch back `Ctrl-n`.

The navigation mode duplicates much of `Ricksep`'s functionality. Pressing the left mouse button and moving the cursor selects a region to zoom in on. Single-clicking the middle mouse button allows you to navigate through the cube. Double-clicking the left button unzooms. When viewing three faces of the cube (view modes: cube, cut, and three-face) you can change the relative sizes of the three faces by selecting with the left mouse button the shared corner point, moving it, and releasing the mouse button. Zooming in one axis in one display will cause they same zoom to occur in every other view where the axis is displayed.

When in picking mode the left mouse button actions are still enabled, but the

center and right buttons' functions change. Selecting the right button you can add a pick to the cube, and the center button delete the closest point.

MENUS

The menu is arranged in two levels of tabbed sub-menus. There are nine primary menus: `Main, Navigate, View, Orient, Color, Clip, Auto, Picks,` and `Help`. When using multiple views, the first view will contain all nine menus while subsequent views will have six of the menus, as `Main, Picks,` and `Help` contain global rather than view-specific options. In the following section I will describe the various options in these menus.

Main

There are three functions in the main view. The first is only relevant when running with multiple view windows. The user can choose to 'lock', or synchronize, the various views. When views are locked you guarantee that each view will have the same color table, perspective, orientation, etc. The main view also allows you to quit the application and to save the history of all the actions performed during the session.

Navigate

The navigation menu has two sub-menus, `Movie` and `Direction`. The `Movie` sub-menu relates to displaying a series of slices along one axis of the hypercube. The `Movie` sub-menu allows you to start and stop the movie (`Go` and `No`) and advance the move one frame. You also can control the speed of the movie with a slider.

The `Direction` sub-menu allows you to control both which axis to loop over and in which direction. You are limited to selecting one of the three axes currently displayed in the view. To describe the effect of the various `Direction` options imagine a 3-D cube with the axes depth, X position, and Y Position. You can change the depth slice (`+Z` and `-Z`), different Y-Z slices (`+X` and `-X`) or X-Z slices (`+Y` and `-Y`) being displayed. Changing the movie direction you also automatically advances one frame.

Views

The `View` menu contains two to four sub-menus depending on whether you are working with a single dataset or multiple datasets. You always are presented with a `Main` and `Save` sub-menus and will have an additional `Data` and `Overlay` sub-menu when multiple datasets are loaded.

The `Main` sub-menu allows you to set the perspective, the font size, and whether or not to draw a colorbar. You have six perspective options, three that view a single slice of the data, and three that view multiple slices of the data. The three single slice options are `FRONT, SIDE,` and `TOP`; these correspond to viewing slices containing the 1-2, 1-3 and 2-3 axes, respectively. In addition, you can see a plan view `THREE`, a cube-view `CUBE`, and cut `CUT` into the data. The font menu allows you to control the font used to draw the axes, and the colorbar allows you to view a colorbar with a superimposed histogram.

The `Save` sub-menu allows you to save the current view as a ppm file or a large or small postscript file. The small postscript option is appropriate for inclusion in presentations while the large option's 1200 DPI is appropriate for papers. The large option is quite time consuming because it redraws the current view at approximately 64 times the resolution of a standard screen.

The `Data` and `Overlay` sub-menus are only available when using multiple datasets. The `Data` sub-menu allows you to choose which dataset, or combination of datasets you wish to see in the current view. You have the option of selecting any of the current datasets or overlaying two datasets. The overlay is done by modifying the opacity channel of the overlying dataset. Figure 3 shows an example of overlaying a velocity model and a migrated image. The figure itself is generated through `Save` menu. You can control the level of opacity through the `Opacity` sub-menu. In addition, you can change the colormap of the overlying dataset in this menu. With the `Cycle` button in the `View` sub-menu you can cycle through all of the datasets.

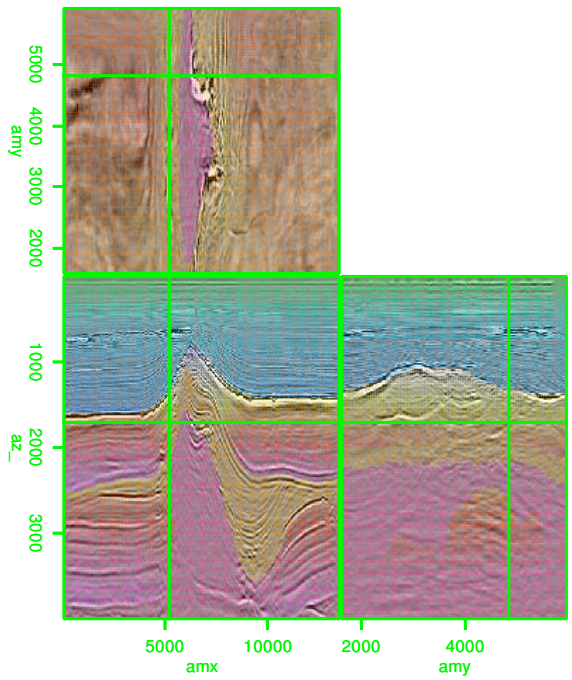

Figure 3: An example of overlaying a velocity model on top of a migrated image. [**NR**]

Orient

Generally, every dataset that `Hyperview` displays must be of the same size. Each view has its own axis order and direction. By default each views' first axis corresponds to the first axis in the dataset, the second the second, etc. The `Orient` menu allows you to manipulate the axis order and direction through its two sub-menus, `Main` and `Transpose`.

The `Main` sub-menu allows you to flip (reverse) any of the three currently viewed axes. This amounts to changing the direction in the view's map. With the `Main` sub-menu you can move to the center of the currently viewed cube or to an edge of the cube. In addition, you can reset the cube to its default orientation. This option will undo any flips, transposes, and/or movements you have done. The final option in the `Main` sub-menu is the ability to view the cube in true proportions. This option will make sure that the length of each axis (the number of samples times the sampling) is proportional to the amount of space it takes on the computer screen.

The `Transpose` sub-menu allows you manipulate the view's axis order. For example, transposing the first and second axis, flips the front and top panel's positions. Most of `Hyperview's` actions work on only the view's first three axes. With the transpose options you can view and manipulate the fourth or fifth axis of a dataset.

Color

The `Color` menu controls color map of a dataset. Currently there are five colormaps: gray scale, flag, CBI, CGSI, and rainbow. Flag maps positive value to red and negative value to blue, zero is white. CGSI behaves similarly except negative values are mapped to black. CBI maps maps positive to blue, zeros to black, and negatives to red. The rainbow maps negative values to green transitioning to blue, white for zero, yellow, red, and finally purple for large positive values. The `Color` menu also allows you to change the color used for the text and the background color.

Picking

The picking capabilities of `Hyperview` are substantially improved over `Ricksep`'s. There are nine different pick colors available. Each color represents a different pickset. Picks are stored as integers corresponding to the sample selected. When the slice you are viewing corresponds to a pick's coordinate, you see the pick in its primary color. You can also see picks when you are close to their position with the color slightly changed.

There are four sub-menus under picking: `File, Size/Distance, Color`, and `Draw`. The `File` menu allows you load and save all picks. The picks are saved in ASCII format with the pick location and color stored in the file. The `Size/Distance`

sub-menu allows you to change the range in which you can see a pick and the size of the square corresponding to a picked location.

The `Color` sub-menu allows you to change the active pickset. It also allows you to view a specific pickset or all the picksets. The `Draw` sub-menu gives you the option to display the picks as either lines or points. In the simple logic of the program, in order to draw a line it must have an axis that is single valued. For example, in velocity analysis you will not select more than one velocity per time sample. In the `Draw` sub-menu you select the multi-valued axis. This indicates that any slice that contains the multi-valued axis also contains a single valued axis. A line can then be drawn connecting points along the single valued axis. By default, the multi-valued axis is set to the first axis (depth). Figure 4 shows an example of picking. Note the three different colors used and how lines appear in the front and side views while points are displayed in the top section.

Clipping

Using the clip menu you can change the dynamic range of your data. The clip menu displays two lines. The green line is a histogram of your dataset. Specifically the data is mapped into 256 different regions. In byte mode this corresponds to the 256 different values that a byte can take on. In float mode the regions range from the minimum to maximum clip value.

The second black line shows the mapping from these 256 different regions to the 256 different colors in the colormap. By default the first region maps to the first color map value, the second to the second, etc. The horizontal axis is the regions and the vertical axis is the color map index. The black line is constructed by linearly interpolating between control points, (note the squares at the far ends of the display). The right mouse button creates additional control points can be added (the center button deletes the points). If you wish to increase the dynamic range, you can introduce additional control points. Figure 5 shows an example. The left panel is the original data; the right panel is after adding control points that map outlying data values to the same color map index. Note how the right panel of Figure 5 shows much more dynamic range.

Auto picking

Currently there is a single auto-picking mechanism in `Hyperview`. The user selects a series of control points. These points are honored and then the algorithm attempts to fill in between the selected points. The user then has the ability to add in additional control points and re-pick, or delete the auto-pick points entirely.

The method is a modified version of the Viterbi algorithm described in Clapp (2008). This is a purely 2-D picking scheme. `Hyperview` loops through all of the

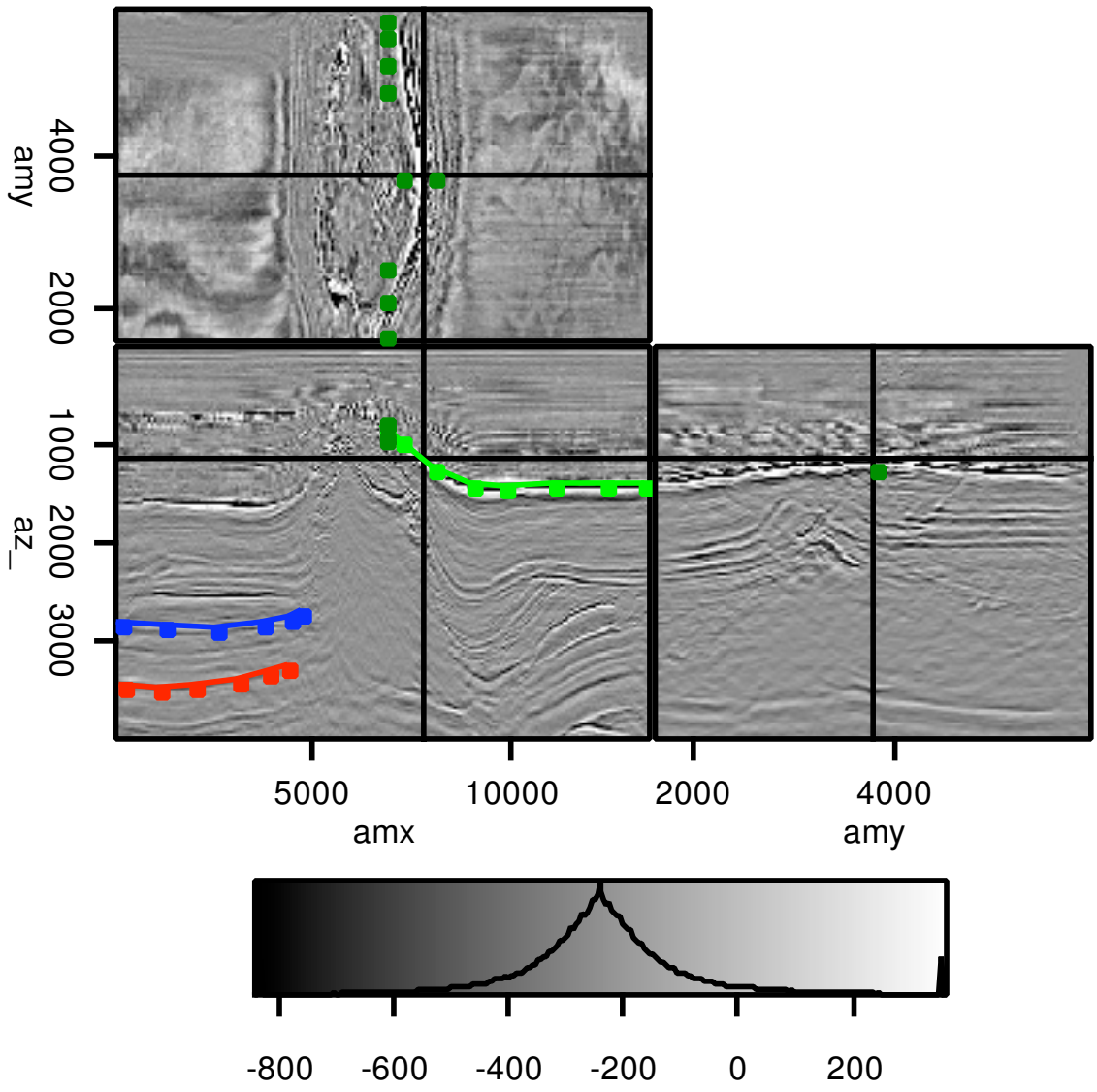

Figure 4: The result of picking three reflectors. Note the use of a colorbar and how picks show up as line in the front and side panel but as points in the top panel. [**NR**]

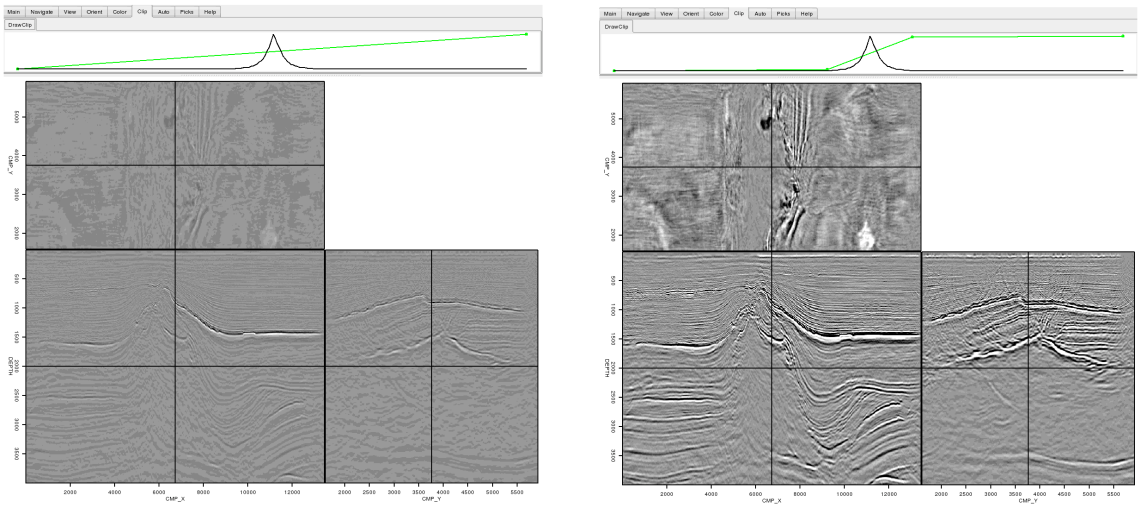

Figure 5: The effect of adjusting the clip. The left panel shows the original data and clip. In the right panel the clip has been adjusted by adding control points closer the range of most of the data's value. Note how the image on the right shows significantly more dynamic range. [**NR**]

currently displayed views. For each view it finds all of the points in the current active pick color. These picks are used as control points, and the Viterbi algorithm attempts to find a smooth path between these points. By looking for large amplitudes between the picks (semblance) or finding the path with peaks in local correlation (reflectors).

An additional auto-picking sub-menu is `Lloyd`. The Lloyd's algorithm approach, described in Clapp (2008), is an attempt to make QCing easier. It takes a set of picked points and tries to find a reduced set of points which contains virtually the same information. The Lloyd's approach is currently a 2-D operation. It attempts to throw away points which are nearly linear combinations of neighboring points.

KEYBOARD SHORTCUTS

In addition to switching between modes described earlier, there are additional keyboard shortcuts.

Key	Action
c	Cycle forward one colormap.
$>, <$	Cycle forward or back one dataset.
v	Cycle through different perspectives.
x,X	Cycle forward or back along the x (second axis).
z,Z	Cycle forward or back along the z (first axis).
y,Y	Cycle forward or back along the y (third axis).
r	Start a movie running.
s	Stop a movie.
q	Quit current `Hyperview` session.

HISTORY

`Hyperview` borrows from Gocad (among other software packages) the ability to save and script actions. Every action through the menus, keyboard, or by the mouse is recorded by the program. Actually, what is recorded is both the action and how to counter the action. You can see the recorded actions in the history window. By clicking on an action you can undo that action all subsequent actions. In addition you can save all the actions you do during a run. This functionality is useful in two ways. First, it make it possible to consistently find a view that best illustrates a given point. Second, you can edit and rerun your file for a presentation. You can also add `sleep X` commands that will pause a presentation for a given number of seconds. The history functionality enables figure reproducibility. By saving the history rerunning it in batch mode the figure can be recreated from the command line.

DESIGN

`Hyperview` was designed to be easily modified for a variety of purposes. In this section we will cover the core objects, point out important public functions, and discuss some possible extensions that would add additional capabilities.

Window

Each view is a different `DrawWindow` object. It is a container class for all of the different objects associated with building, displaying, and interacting with a given dataset. It stores the current dataset being viewed, the colormap used to view the data, and how to display the data. Many of the menu and keyboard options modify the objects stored in this class.

Colormap

The `image_factor` object and its children describe how to take a 2-D field of unsigned chars and create an image of a given size. Currently this is limited to making a raster plot of the data based on a given colormap. One potential extension would be to add the ability to draw wiggle plots, graphs, contour plots.

Slice

The `slice` object's job is to display a 2-D plane. It requests from a data object a 2-D plane, potentially draws its axes, and the current location, and then calls the `image_factory` object to display the slice's contents.

View

The `view` object's children contain information about how a given `DrawWindow` object is going to display its dataset. Currently the view object has two children: `single`, which displays a single 2-D plane from a dataset, and `multiple` which display multiple views of a given dataset. There are several potential additions to the view object. For example `Ricksep` allows an array of planes from a given dataset and a fences view of a given dataset.

Picks

The `pick_groups` object contains a series of `pickset` objects. The `picksets` are associated with a given color. Each `pickset` contains as series of`pick`'s. The `pick`'s contain its location and a integer type flag. Currently this flag is used to differentiate between picks made by the user and autopicked locations. The `pick_groups` object also contains how to draw the picks on a given slice.

Updater

One of the most important objects is the `update` object. This object is called by all menu, mouse, and keyboard actions. It stores a given action in its history along with how to undo the action. It then calls the appropriate object functions to accomplish its tasks. Any additional functionality must include an addition to the `update` object.

Data

The most important object is the `h_data` object. The data object influences the functionality of many other portions of `Hyperview`. The `h_data` object how to read the data its linked to in memory. Normally, this takes the form of reading one of the five data types described above. The `h_data` object knows to return a 2-D slice at a given location. Conventionally this involves extracting a portion of the buffer that was read in, but it could be expanded on to read from disk instead.

The data object also allows you turn off some default behavior if it is inappropriate for a given dataset. For example, each `h_data` says whether or not you can navigate within a view using the object, whether you can pick on the object, and what action to perform given a pick on the dataset. The contents of the dataset can also change. The `update_h_data` function is called before viewing in `DrawWindow` object. The `h_data` object has the option (based on the current position, picks that have been made, etc.) to change the data associated with a given view. Finally, the data object has the concept of fake axes. Generally `Hyperview` is limited to five dimensions. In fact an eight dimensional position is stored. These other three axes can be used to display datasets that conform to a subset of the first five dimensions. In the next section I will demonstrate how to use a number of these features.

SEMBLANCE PICKING

One of the most basic interactive processing operations is picking move-out. Writing a move-out application involved creating two new classes inherited from the `h_data` object, `nmo_semblance` and `nmoed`.

The `nmo_semblance` object is initialized with the data object associated with the pre-stack gathers. The `nmo_semblance` object takes advantage of the fake axis concept mentioned above. Instead of having an offset axis, a sixth axis, velocity, was added to the dataset. The data is generated from the CMP gather so the `read_h_data` does not perform any action. Finally, the `get_h_data` is modified. When initialized, every time the position changes the `nmo_semblance` object grabs the current CMP gather. It performs semblance on the gather and returns the resulting field. The center panel of Figure 6 shows the semblance calculated at the given CMP location.

The `nmoed` object is initialized with the both the pre-stack data and the `nmo_semblance` objects. Like the `nmo_semblance` object, it does not use the `read_h_data` function. The `get_h_data` function is also modified. The current CMP gather is grabbed from the data object, and the RMS picks closest to the current CMP are taken from the `nmo_semblance` object. The picks are then used to form a RMS velocity function and the CMP gather is NMOed with this function. The result is shown in the right panel of Figure 6. Picking is only allowed on the `nmo_semblance` object, and navigation is only allowed on the data object.

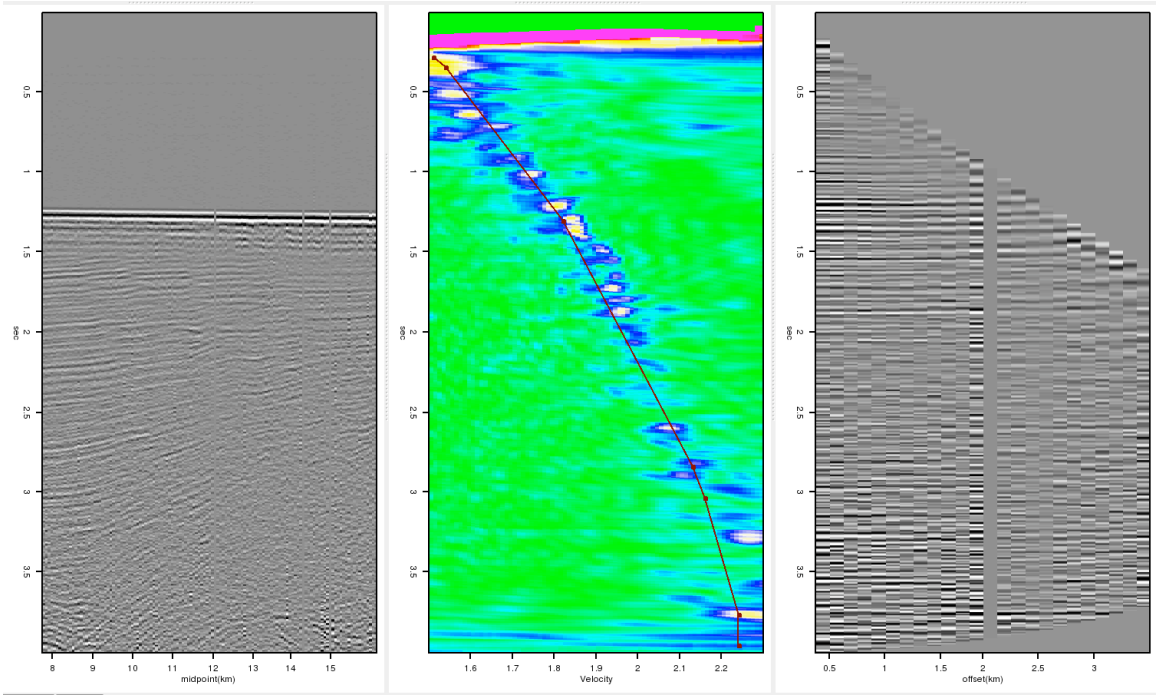

Figure 6: A typical semblance analysis display. The left panel show a depth-midpoint section. The center panel show the semblance at the given CMP overlain by the user selected v_{rms} function. The right panel is the result of NMO using the selected RMS function.[**NR**]

FUTURE DIRECTIONS

Throughout the paper we discussed potential extensions to the current viewer. The most important feature to improve is the ease with which a programmer can retrieve information from the GUI to send back an improved image. The current mechanism is effective but requires too much knowledge of the underlying libraries.

CONCLUSION

Interacting with multi-dimensional hypercubes is an essential tool in seismic exploration. The program, `Hyperview`, is a powerful viewing and interacting platform.

REFERENCES

Berlioux, A., 1994, Building models with GOCAD: SEP-Report, **80**, 617–634.

Chen, D. M. and R. G. Clapp, 2006, Data-fusion of volumes, visualization of paths, and revision of viewing sequences in ricksep: SEP-Report, **125**.

Claerbout, J. F., 1981, On-line movies: SEP-Report, **28**, 1–4.

———, 1987, Interactive filter design in the z-plane: SEP-Report, **56**, 263–272.

———, 1991, Interactive one dimensional seismology program ed1D: SEP-Report, **71**, 293–294.

Clapp, R., 2001, Ricksep: Interactive display of multi-dimensional data: SEP-Report, **110**, 163–172.

Clapp, R. G., 2008, Lloyd and viterbi for qc and auto-picking: SEP-Report, **134**, 2000–2001.

Clapp, R. G., B. Biondi, and M. Karrenbach, 1994, AVS as a 3-D seismic data visualizing platform: SEP-Report, **82**, 97–106.

Cole, S. and D. Nichols, 1992, Xtpanel: An interactive panel builder: SEP-Report, **75**, 497–520.

———, 1993, Xtpanel update: Interactivity from within existing batch programs: SEP-Report, **77**, 409–416.

Mora, C. B., R. G. Clapp, and B. Biondi, 1995, Velocity model building in AVS: SEP-Report, **89**, 133–144.

Ottolini, R., 1982, Interactive movie machine user's documentation: SEP-Report, **32**, 183–196.

———, 1983, Movie cubes: SEP-Report, **35**, 235–240.

———, 1988, Movies on the Macintosh II: SEP-Report, **59**, 255–268.

———, 1990, Seismic movies on the XView graphics system: SEP-Report, **65**, 315.

Sword, C. H., 1981, SEP goes to the movies: SEP-Report, **28**, 11–20.

van Trier, J., 1988, An interactive interface for velocity optimization using geological constraints: SEP-Report, **59**, 241–254.

APPENDIX A

The script `Hyperview` has numerous command line arguments. The general form is

```
pname file1 [file file3 pars]
```

where `file1...3` are data files and `pars` are series command line arguments of the form `param=value`. Generally, the options break down into three categories, commands that are data specific, view specific, and everything else.

Data specific options - By default `Hyperview` checks the suffix of all input files. It then tries to guess the data format. It makes the following suffix assumptions:

Suffixes	**Data type**
H,h,T,t,HH	SEPlib
rsf, RSF	RSF
su, SU	SU
segy,SEGY,sgy, SGY	SEGY
sp,SP	SeisPak

If the suffix of the file does not match one of these defaults or the default type is incorrect the user can add `typeX=FORMAT` where X is the order of the dataset (starting with 0) on the command line and `FORMAT` is either `SEGY, SEP` (RSF is equivalent to SEP), `SU,` or `SEISPAK`.

View specific options - View specific options take the form `paramX=Y` where X is the view number.

Option	**Default**	**Description**
viewX	`THREE`	Style of view, must be `FRONT, SIDE, TOP, CUT, CUBE,` or `THREE`.
orderX	1,2,3,4,5,6,7,8	Axis order for the given view.
backgroundX	red	Background color, must be `red, green, grey, black`, or `white`.
overlayX	green	Overlay(text) color, must be `red, green, black,` or `white`.
colortableX	gray	Colortable to use view the given view, must be `flag, cbl, cgsi`, or `rainbow`

Other options - Two additional command line arguments are available: `nviews` and `position`. The `nviews` option tells how many different views to bring up in the display window;, by default a single view is created. The `position` argument is the initial location to view in the cube. By default the center cube is displayed.