SEPHTML5: HTML5 and interactive visualization

Robert G. Clapp

ABSTRACT

The growing move to a cloud-based/server-processing-dominant processing paradigm poses a problem for viewing and interacting with seismic data due to X11's high latency and large bandwidth requirements. HTML5's flexibility, support for strong client-server paradigms, and low latency design allows for a platform independent solution to visualizing seismic data. SEPHTML5 is written on top of HTML5 for interacting with seismic data. I present two different applications, a 3-D viewer and multi-slice comparison tool to highlight the potential of a HTML5 approach to viewing seismic data.

INTRODUCTION

Viewing and interacting with multi-dimensional volumes is necessary when working with 3-D data. SEP wrote its first movie program 30 years ago and has continually expanded on this initial idea (Claerbout, 1981; Sword, 1981; Ottolini, 1982, 1983, 1988, 1990; Arroyo and Clapp, 2002; Clapp et al., 2008; Clapp, 2010). These movie programs have progressed from simply showing a series of frames to allowing greater and greater levels of interactivity. In the last twenty years every viewing approach was written on X11 or a library, such as Motif or QT, relying on X11. As a result, they had both the strengths and weaknesses of X11. They had the advantage of being portable and effectively working in the 1990's local network environment where the user was accustomed to latency in most applications. As computers got faster, the many handshakes and bit image transferring approach of X11 led to ever more noticeable latency problems on even local networks, exacerbated when using remote connections. As a result, remote frame buffer approaches such as VNC have grown more and more popular. These approaches achieve speed by sending a compressed version of the portion of the screen that has changed. The downside of remote frame buffer approaches is they are completely reliant on the server and suffer when significant latency exists between the client and server.

Web development has had to deal with similar issues of how much work should be done on the client versus the server. The earliest web applications were completely server based. A web page would contain different requests to a web server which would respond with a new page, image, etc. The introduction of Java in 1995 enabled work to be done on the client in a platform independent method. While Java remains widely used, its popularity in web based applications has decreased from its peak due

to both competitors and security issues. Adobe introduced a competitor to Java with Macromedia/Adobe Flash. Flash has proven popular, but its use is again declining due to high compute requirements and its proprietary nature. The introduction of Javascript, and more recently its standardization/expansion with the introduction of HTML5, has allowed programmers another approach. With HTML5, the programmer can easily decide how much work should be done on the server versus the client.

In this paper I introduce SEPHTML5, a HTML5 based library for displaying and interacting with seismic datasets and two applications written on top of it. The first application allows a user to explore a 3-D cube by navigating, zooming, and panning. The second application allows several different two-dimensional slices to be compared using a variety of different approaches. I conclude by discussing different applications that could be written on top of the library.

METHODOLOGY

Writing a HTML5 application involves several significant choices. The first is how much work should be on the web server versus how much work should be done on the web viewer (client). The second decision is whether to rely on a third party library, andm if so, which library. Developing SEPHTML5 involved two other important decisions: how to pass data between the client and server and how to ensure data security.

Client-server model

Recent web applications have become more and more client-compute dominated. This is due to both the ever growing support for the HTML5 standard and the smaller load imposed on the server by such an approach, allowing an application to scale without significant additional server resources. One downside of the client dominated approach is that javascript is a text based language accessible to the user. Though code obfuscation is possible, copyright violation is easier than with compiled language approaches. The more important downside for SEPHTML5 is the size of seismic datasets. Transferring an entire dataset across the network would introduce significant startup delays and, even more problematically, is not possible given most web viewers' memory restrictions. As a result SEPHTML5 is a relatively balanced client-server application.

Upon an initialization request, the server loads a dataset(s) into memory.[1] The server describes the dataset(s) to the client. After this initial handshake, the server's sole responsibility is to return two-dimensional images, compressed using JPEG (Wallace, 1991), that the client requests. The client handles all interactivity. Zooming, panning, moving, and navigating are handled by the underlying SEPHTML5 library.

[1]This is easily extendable to a series of nodes loading the dataset.

The server can hold multiple datasets in memory simultaneously, dumping them after an extended time of non-use (e.g. 5 minutes).

Javascript programming

Javascript is an event based language. An event based language essentially runs a series of very short programs that have access to a single set of global variables. As a result, concepts of "sleep and then do an action" are replaced with "in some time period, create a new event and do an action" In addition to being a different way of thinking of programming, event based programming introduces challenges if sequencing is important. Programs must be robust to new events happening in an unexpected order. Probably the closest conventional programming comparision is pthread programming without locks.

Hundreds, if not thousands, of libraries have been written on top of javascript. These libraries simplify some group of potential applications. For SEPHTML5 I chose to write much of the interactive graphics on top of KineticJS (Kovalenko, 2013). KineticJS specializes in interacting with graphical objects using both mouse and touch controls. Javascript recognizes standard mouse actions such as down, up, move, in, out, and double click. In addition, it recognizes touch events such as start, move, and end, even for simultaneous touches. As a result, the programmer has great flexibility in designing applications that work on both mouse and touch based devices. Each mouse/touch event can initialize a new event the programmer can use.

Web sockets

Web sockets are a relatively new feature in HTML programming. Web sockets allow a persistent connection between a client and server. Initialization involves a handshake between the client and the server at which time another Unix socket is used on the server to transfer information to the client. The web socket approach has a couple of significant advantages. First, because the connection is persistent, each additional client request does not require a handshake, decreasing the latency of the communication, and allowing for a higher level of interactivity with the server. The other advantage is that anything can be sent over the connection, in ASCII or binary form. For SEPHTML5 this allows both compressed images and other information to be sent between the web viewer and the server.

Security

It is generally desirable to restrict access to the data that may be viewed by a given user. SEPHTML5 uses several different security measures. All pages are password protected. The server restricts data further by connecting different users with a

limited number of datasets that they have been given permission to view. Two further security procedures can be added: web sockets allow all connections to be done over a secure connection and standard web servers can restrict access to pages (and javascript code) based on domains.

THREE-DIMENSIONAL VIEWER

The first application written on top of SEPHTML5 is a three dimensional cube viewer similar to the approach of Ottolini (1982) and Clapp et al. (2008). Figure 1 shows a dataset in the viewer. Instructions on how to use viewer are available by selecting the "Controls" option using either the mouse or a touch device. Different input sources have different controls. The mouse approach uses a combination of the keyboard and mouse to allow the user to resize the viewer, navigate through the cube, start movies in various directions, change the color scheme, and zoom.

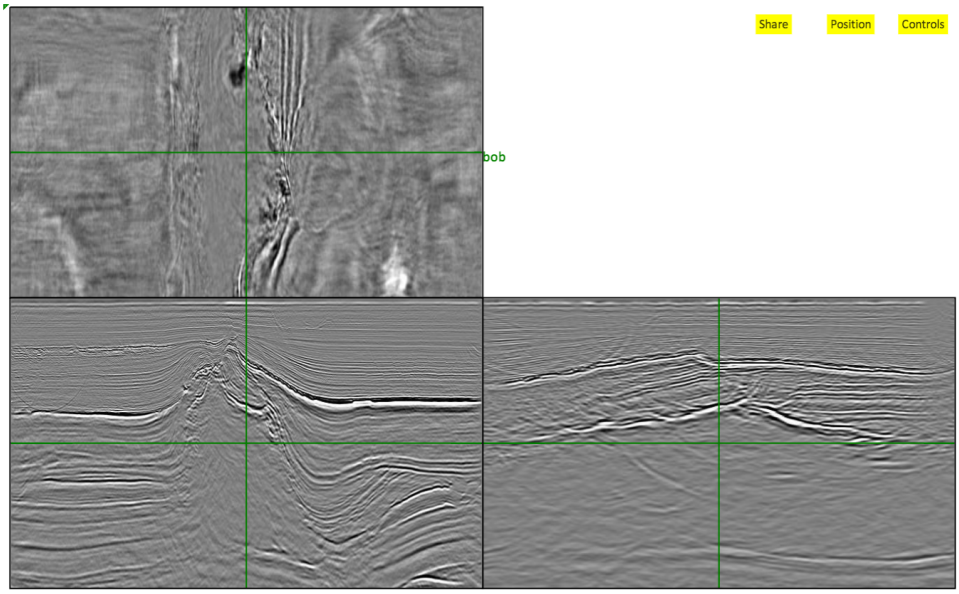


Figure 1: A screen capture of the cube viewer. Note the front, side, and top slice through the cube along with an option to view the current position and see the controls. [**NR**]

The same options are available using a touch device, the difference being that multiple fingers and swiping are used. Pinching will zoom in and zoom out. Flicking will start a movie. Three fingers will change the color table. Panning can be accomplished by moving a finger on a zoomed in region. The viewer can be used on iPads and iPhones using the Chrome browser (the browser with best current support of HTML5). Most users have found the touch interface to give a superior ability to interact with the dataset.

There are some drawbacks to the HTML approach. ARM based tablets, when handling large (1000x1000) images, can handle only about 5-10 frames per second, while conventional CPUs can handle 10-20 frames per second. By sending movies rather than static images, it should be possible to get the frame rate up on both devices.

FRAME VIEWER

The second use of the library is to compare two-dimensional slices. This use is designed for a seminar environment which involves comparing before and after pictures and multiple 2-D slices. The user specifies a series of slices to load. Figure 2 shows an initial grid of slices. The user can move and resize these figures. Each figure's color table can be changed and soon clipping will be added as an option. In addition, the user can place one figure on top of another. This will cause the two images start to blink back and forth. Additional images are just added to to the movie.

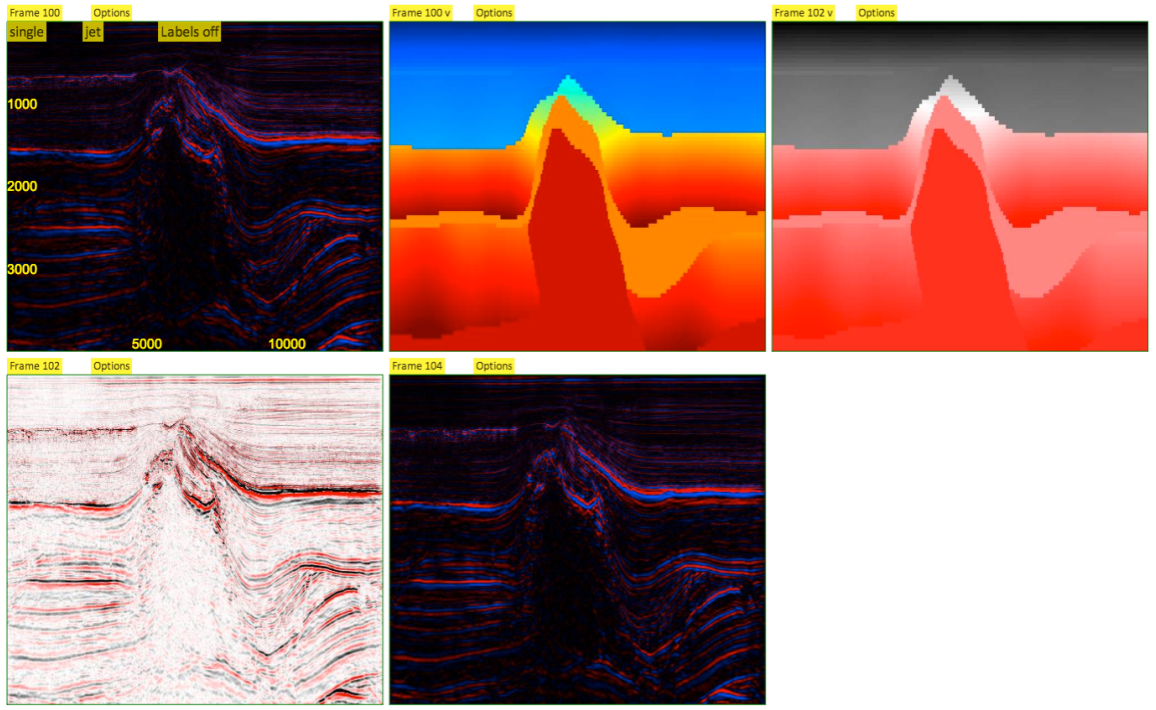


Figure 2: A view of five different slices from two different datasets. Note the different color tables used in the various images.**NR**

When only two images are placed on top of each other, additional comparison tools are enabled. One option is to overlay the two images with adjustable transparencies. Another option is to flip between the two images using a slider. Finally, the user can choose to peer through one image to the other in a flashlight view. Figure 3(a) demonstrates the opacity feature, Figure 3(b) shows the flipping effect, and

Figure 3(c) shows the flashlight effect.

When two images are on top of each other, clipping and color tables can be changed simultaneously allowing for easy comparisons. Zooming will soon be added as an additional option.

FUTURE WORK

Several additional features need to be added to the library. The multi-dimensional viewer needs to be expanded to view cubes larger than 3-D and to handle multiple volumes simultaneously. Picking and other common interactive tools need to be added. The frame viewer, beyond the before mentioned clipping and zooming, should have the ability to view datasets larger than two dimensions.

There are several directions to take SEPHTML5. Following on the work of Clapp (2012) it could be extended to an interactive processing platform where the data exists on a secure server and the user sets processing parameters from the web viewer. Processing could include conventional processing flows and more highly compute-intensive interactive processing such as regularized Hessian-based inversions.

Another avenue is to expanded its 'seminar' ability. Specifically the ability for all of the audience to not only simultaneously interact with same dataset(s) but to share their interactions with everyone viewing the same dataset. Such an approach could lead to a more dynamic/collaborative research environment.

CONCLUSIONS

I present a library and two applications to view seismic data over a web connection. The first application is a 3-D cube seismic cube viewer that allows the user to slice through a dataset. The second applications allows the user to compare multiple images using a variety of different approaches. In the future, the library will be extended to allow multiple people to share their interaction with a dataset.

REFERENCES

Arroyo, E. R. and R. G. Clapp, 2002, Displaying seismic data with vtk: SEP-Report, **111**, 395–403.

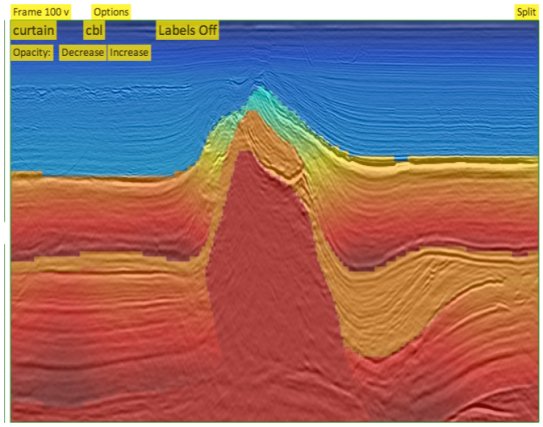
Claerbout, J. F., 1981, On-line movies: SEP-Report, **28**, 1–4.

Clapp, R. G., 2010, Hypercube viewer update: SEP-Report, **140**, 229–232.

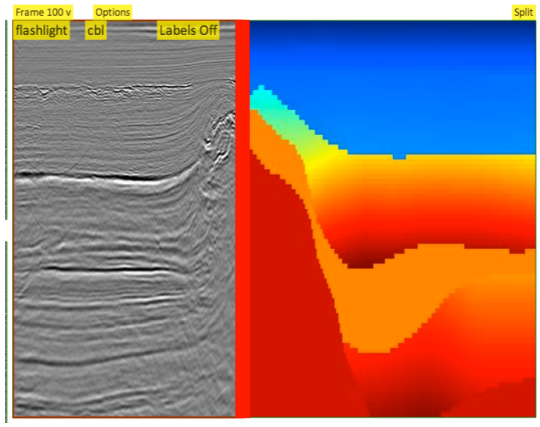
———, 2012, Interactive processing: Geometry manipulation: SEP-Report, **148**, 101–108.

Clapp, R. G., D. M. Chen, and S. Luo, 2008, Hypercube viewer: SEP-Report, **134**, 179–192.

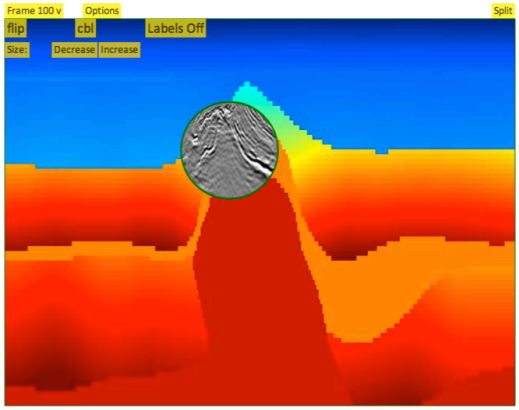
Kovalenko, A., 2013, Instant kineticjs starter: Packt Publishing Ltd.



(a)



(b)



(c)

Figure 3: Panel (a) shows the result of combining two images. In this case velocity and a seismic image. Panel (b) show two different datasets which can be quickly switched between by adjusting the center bar. Panel (c) shows the ability to look through one dataset into another. [**NR**]

Ottolini, R., 1982, Interactive movie machine user's documentation: SEP-Report, **32**, 183–195.

———, 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.

Wallace, G. K., 1991, The jpeg still picture compression standard: Communications of the ACM, **34**, 30–44.