Computers & Geosciences ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Interactive 3D/2D visualization for geophysical data processing and interpretation ☆

Igor Morozov *, Glenn Chubak, Shannon Blyth

Department of Geological Sciences, University of Saskatchewan, 114 Science Place, Saskatoon, SK, Canada S7N 5E2

ARTICLE INFO

Article history:
Received 26 January 2007
Received in revised form
1 July 2008
Accepted 7 October 2008

Keywords:
Gravity
Magnetic
Object-oriented
OpenGL
Seismic
Qt

ABSTRACT

A new 3D/2D interactive display server was developed for the IGeoS geophysical data processing framework presented earlier. With introduction of this major component, the framework becomes conceptually complete and potentially bridges the gap between traditional processing and interpretation geophysical software.

The display server utilizes Qt toolkit and OpenGL graphics libraries while taking advantage of the object-oriented design of the core data processing system. It operates by creating image object trees that are automatically propagated to the server(s) residing on the same or remote hosts and producing complex, structured, and interactive data displays. The displays support custom interactive graphical user interfaces, which are constructed entirely by the user and do not require computer coding. With over 200 specialized processing tools available, this approach allows creating 3D visualizations and building custom interactive data analysis, interpretation, and modeling tools in many areas of application. In particular, we show examples of integration of seismic ray tracing, gravity, and receiver function modeling and inversion in deep crustal studies.

1. Introduction

Q2 In several previous publications (Morozov and Smithson, 1997; Morozov, 1998; Chubak and Morozov, 2006, 2007; Morozov et al., 2006), we described an open-source software package for geophysical data handling, analysis, and modeling, which we call IGeoS[1] (SIA). Started initially as a multicomponent seismic processing package, the approach proved to be quite unique in its broad scope capable of covering the full spectrum of seismic, potential-field, and other geophysical data analysis. The implementation extended to full object-oriented design based on C++, dynamic linking, integrated full-featured graphical user interface, parallel functionality, and web services.[2] Its abstract, logic-based back-propagation data handling model (Morozov and Smithson, 1997), the ease of implementing new tools, high code integration, and extensive documentation and development support allowed extending the system into a code development framework (Chubak and Morozov, 2006). Recently, an automatic software distribution and updating service[3] was added (Chubak and Morozov, 2007), which facilitated concurrent development and automatic maintenance of the package by programmers collaborating across the web.

In this paper, we describe a new component of the IGeoS framework—general-purpose, customizable, interactive 3D/2D visualization server program. As with other components of the system, in designing this server, we emphasized data abstraction, scalability, efficiency, and parallelism. The resulting code is nearly entirely content-agnostic and suitable for working with most types of geophysical data, in both passive (as data "viewer") and interactive ("editor") roles.

Traditionally, geophysical software packages developed, for example, in the reflection seismic industry have been differentiated into "processing" and "interpretation" systems. Processing systems emphasize flow-based design, with numerous operations applied to the data in complex processing sequences, and only limited interactive functionality offered by the individual tools. Special emphasis is made on handling large data volumes and on reproducibility of the results. This is generally accomplished through batch (unattended) execution by using "processing flows" often executed on multi-processor (up to several thousand nodes) computer networks. In such systems, processing flows are typically encoded in scripts or generated by interactive user interfaces and preserved as processing records. This guarantees reproducibility of computations and allows creating complex processing sequences.

☆ Code available from server at http://seisweb.usask.ca/igeos/doc/install.html

* Corresponding author. Tel.: +1 306 966 2761; fax: +1 306 966 8593.
E-mail address: Igor.morozov@usask.ca (I. Morozov).

[1] IGeoS project page: http://seisweb.usask.ca/igeos

[2] http://seisweb.usask.ca/igeos/ps.php

[3] http://seisweb.usask.ca/igeos/cs.php

0098-3004/$ - see front matter © 2009 Published by Elsevier Ltd.
doi:10.1016/j.cageo.2008.10.005

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

Several such processing systems were developed and are extensively used by the seismic industry. In the open-source community, most notable related projects are the Stanford Exploration Project software (SEPlib,[4] Seismic Un*x (SU; Stockwell, 1999), and FreeUSP,[5] and Madagascar (RSF[6]). However, these systems are geared towards specialized data types and narrow processing tasks (mostly reflection seismic processing) and still offer only basic user interfaces (essentially, UNIX and Perl shells). Because of their dependence on data types, these systems are restrictive in their data formats (typically SEGY-like formatted UNIX pipes or files) and limited in integration of the tools.

By contrast, interpretation systems are visualization-centred and based on data viewers. In commercial seismic interpretation, examples of such 3D visualization systems include GoCad,[7] GeoQuest, Petrel (now parts of Schlumberger Information Solutions), SeisWorks (Halliburton), OpendTect,[8] and several others. In such systems, data organization follows spatial patterns, and operation is mostly driven by data displays and user commands. Application of various "plug-in" tools is typically determined interactively by the user, and only a limited number of fast operations can be performed in real time.

In deep crustal geophysics, commercial processing, and particularly interpretation packages do not meet modern requirements. Systems natively capable of working in curved-Earth geometry, with sparse and heterogeneous datasets, and incorporating academic inversion methods and algorithms are needed. In addition, the costs of most 3D commercial systems are prohibitive for many academic users. Academic code also needs to be developed quickly and be maintainable by the community. These are our major motivations for developing an open, inclusive geophysical code framework.

In our visualization approach, we attempt to no longer follow the above differentiation between processing and interpretation workflows and perform them on a common software base and user interface. We give priority to the processing flow model as the one offering reproducibility, scalability, and the ability to handle large datasets. Construction of an image and interaction with it is thus performed by a processing flow, potentially concurrently with other operations with the data. As described below, images and user interfaces can be encoded in IGeoS data processing flows, which can be further rendered either in PostScript or by using the new interactive OpenGL-based display server. Because the content of the display is entirely determined by the underlying processing, the display server can in principle implement any functionality, such as showing seismic data, performing gravity modeling and seismic ray tracing in the same session. Through direct access to GMT databases, the server is also able to include 3D base maps in its displays. In addition, full seismic and other data processing capability is also available to the interpreter through the underlying batch flow capability.

Below, we describe the new IGeoS 3D/2D display server. In this short publication, we only focus on its data abstraction and the processing/interpretation model while leaving aside the details of implementation. As most other components of the system (Fig. 1), the display server currently operates on several types of Linux computers, ranging from Red Hat and Mandriva notebooks to 32- and 64-bit Fedora and SUSE workstations and multi-processor Red Hat Enterprise servers and clusters (Chubak and Morozov, 2006). Due to the use of cross-platform GNU compilers, Qt, and OpenGL libraries, the system should be readily portable to IBM, Sun Solaris, and SGI IRIX. In fact, IGeoS was initially developed and extensively used on these operating systems in the past (Morozov and Smithson, 1997). Porting to Mac OS would also appear straightforward and important because of its growing popularity in the seismological community (Jim Fowler and Rick Aster, personal communications, 2007).

In the following, after an overview of the general IGeoS architecture, we describe the underlying parallel object data communication protocol and summarize the key features of the new visualization component. Further, we show how complex images are formed and interactivity programmed into the processing flows, and present application examples from crustal seismic and gravity studies. In conclusion, we briefly discuss the significance and potential extensions of this approach.

2. IGeoS architecture

IGeoS object code design is generally similar to that used in commercial seismic processing systems (such as ProMAX,[9] Omega,[10] and Echos, formerly Disco/Focus[11]) and differs from other academic projects such as SEP or SU. Precompiled code is arranged in several shared and over 200 plug-in tool libraries (Fig. 1, top). The tools are selected by the user by using either script job descriptions or the graphical user interface (GUI; Fig. 1, left) and linked dynamically during run time to form a custom executable for each task. Multiple data processing/modeling tasks can be created and executed concurrently across the computer network (Fig. 1). These flows operate in independent UNIX processes communicating via the parallel virtual machine (PVM; solid arrows and grey shading in Fig. 1). The GUI monitors statuses of the processes and allows two-way communication with them (Chubak and Morozov, 2006). The PVM interface automatically converts data formats between different computer architectures across the network.

The contents of processing, used file formats, models, etc., are completely determined by the selection of tools made by the user. When needed, processing flows can communicate with the GUI or with the display program (Fig. 1, right). Each component of the package has access to the shared dynamic libraries of all tools, and so methods offered by the tools (e.g., data handling or graphics) are readily available to all components.

Processing-flow scripts simply contain the selections of tools and their parameters in XML or in a format resembling the older Disco jobs (an example is given in Appendix A). The spreadsheet-like format is simple, intuitive, and generally familiar to seismic processors. As in Disco or ProMAX, data flow branching, loops, and multi-threading is achieved by invoking the corresponding tools and not through a complex scripting language. Text substitutions and macro-commands can be used for coherent parameterizations. The GUI can be used to bypass all scripting and to access online help and run-time monitoring of the flows (Chubak and Morozov, 2006).

Although processing flows are represented by sequences of tools, their functionalities in IGeoS are much broader than in commercial batch-driven based systems or in SU. Processing is logic-driven, and along with seismic records, the tools can handle other types of data and produce results that are more general than, for example, altering the contents of data files (Morozov and Smithson, 1997). Tools are allowed to create complex, named

[4] http://sepwww.stanford.edu/software/seplib/
[5] http://www.freeusp.org
[6] http://rsf.sourceforge.net/wiki/index.php/Main_Page
[7] http://www.earthdecision.com/
[8] http://www2.opendtect.org/
[9] http://www.halliburton.com/ps/default.aspx?pageid=862&navid=221&prodid=MSE::1055450737429153
[10] http://www.westerngeco.com/content/services/dp/omega/tools/omega2.asp
[11] http://www.pdgm.com/echos/products.aspx

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

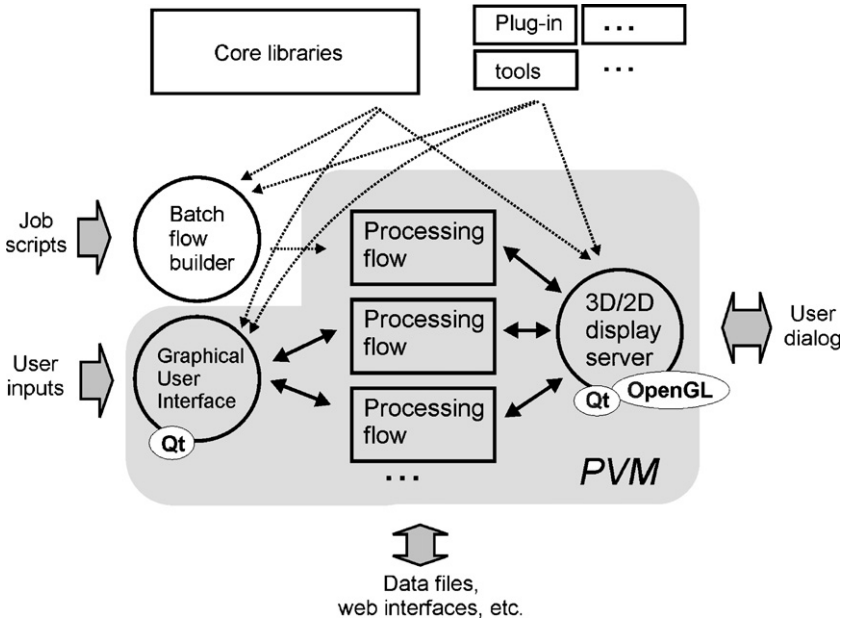

Fig. 1. Simplified IGeoS code structure and process communication. PVM supports two-way communication between programs and graphic objects (solid black arrows) which may run on different compute hosts. Dotted arrows represent dynamic linking of codes from object libraries.

C++ objects and communicate with each other. By using a particular group of such objects, many tools achieve uniform 3D/2D display functionality.

Instead of directly using X11, Motif, or other graphics libraries, as it is done in ProMAX, SU, and similar pro[illegible] we chose to develop an abstract, high-level display C++ object protocol. Although requiring a major initial effort, this allowed us to reduce the rendering efforts at the plug-in tool level to a minimum and to build a single display server for all (at least foreseeable at present) future applications. In addition, by simply replacing the rendering driver, the same protocol can be used in publication-quality PostScript plotting (using GMT programs) and for cloning multiple synchronous displays that can be useful in teaching environments.

3. Object image protocol

As required by their functionalities, processing flows can spawn sub-processes, and in particular, they can start a new or connect to a running display server (Fig. 1, right). This is achieved by including a tool called "gui" in the processing flow, as illustrated in Appendix A. For a single user and display host, the same server handles requests from all flows (Fig. 1). Therefore, if needed, the resulting images can contain objects mapped from different processing flows distributed across the computer network.

Similarly to the GUI, the display server maintains two-way, asynchronous PVM communication with the processing flows (Fig. 1). Whenever a graphic object is created by any flow, its image is automatically transferred to the display server and rendered. Conversely, wherever the user makes changes to an object in the display server screen, the object sends PVM messages to its counterparts in the host flow. This enables background interaction with the running flows. The character of this interaction is determined by the particular graphic objects. In addition, some objects may possess complex interactive functionality on the server side as well—for example, seismic ray tracing and 2D gravity modeling in the examples below are implemented in such a way.

Like any modern interactive graphical display program, the display server is devoid of practically any of its own functionality but simply responds to requests by the client processing flows. It uses PVM messages to maintain hierarchical trees of data objects representing the images being displayed, without interfering with the processing sequences. Identical image trees are stored on both the client (flow) and server sides, and data exchange is carried out automatically whenever either of these si[illegible]s updated.

Schematically, the structure of an ima[illegible]ee is illustrated in Fig. 2. Each node of the tree represents a C++ object that is able to move across the PVM interface (Fig. 1). When an update to such an object is received by ether the flow or the display server, it performs the requested action. For example, the "canvas" object initiates a new display window or updates it, "layout" subdivides the window into Qt layout grid, and other objects place their respective images into the grid. Object "image" carries the coordinate mapping information, and numerous components of the "graphics" database are not displayed themselves but provide image colours, line and fill styles, palettes, lighting, and other viewing parameters (Fig. 2). A group of objects (axes, buttons, sliders, etc.) provides user controls that can be placed on the image for interactive functionality.

Object image trees are included in neither the display server nor processing flow codes. The images are constructed entirely by the user by placing the appropriate IGeoS tools into the flows (an example is given in Appendix A). In particular, tools "image" provide most of the general-purpose objects (2D and 3D lines, surfaces, grids) derived from IGeoS database tables and seismic/potential field records. Specialized tools provide their own objects, which may be quite complex, such as 2D velocity and gravity models (e.g., tools "rayinvr" and "tracer"), or trace sections (tools "plot" and "plotrt"). In principle, whenever useful, any tool can be equipped with a graphical representation and parameters editable from the display server. While containing the graphics objects, the tools take no part in actually displaying the images, which maintain themselves automatically, as described below. This makes the application code simple and robust.

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

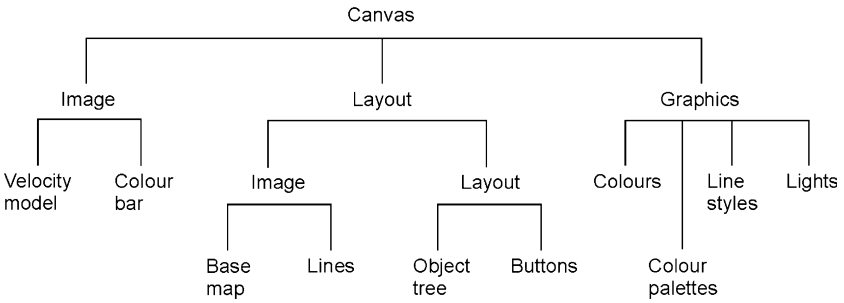

Fig. 2. Sample image tree of a hypothetical Graphical User Interface designed by user. Each node corresponds to an object sent through the PVM communication pipe (black arrows in Fig. 1). Two copies of such a tree are automatically maintained at both client (flow) and display server sides (Fig. 1) and continuously communicate to each other.

4. IGEoS 3D/2D display server

The display server is implemented by using Qt C++ graphics libraries (on which, for example, the popular Linux KDE graphical environment is based), with complex 3D/2D graphics using OpenGL. This ensures that the system should work on a wide variety of platforms and takes advantage of the hardware acceleration on graphics cards and processors. It also allows native 3D rendering on stereoscopic displays, such as GeoWall.[12] The use of C++ ensures high performance and code abstraction, integration and reuse. The look and feel of the program is similar to that of typical modern graphical user interfaces, with drag-and-drop functionality, status lines, tool tips, and elegant window themes available. Generally, code design follows the traditional style of Qt (slot-connection) and OpenGL C-library programming.

The only items in the display server window preconfigured by its code design are the main window menus along its top and the status bar in the bottom (Fig. 3). The remaining main part of the window is subdivided by using docking windows and nested Qt layouts. Layouts are named, described in the processing job, and placed into the object trees together with other objects, which allows construction of both simple and complex displays (Fig. 3). Note that this system of nested, named image frames is similar to the browser frame sets in HTML. In addition to layouts, docking windows are used to hold service objects, such as the object directory tree, property editors, and custom control panels designed by the user. Docking windows appear only when requested by the processing job, and they can be moved to any position on the screen and collapsed into toolbars during the interactive session. All this low-level graphical functionality is performed by Qt objects.

The display server program continuously watches its image object tree (Fig. 2) for updates and rebuilds the display whenever a modification is detected. These modifications can occur as a result of user actions or come from the associated processing flow(s) through the PVM connection. The character of displays is also determined by the objects themselves, and therefore new custom graphical objects can be introduced simply by adding classes into the plug-in library.

Viewing directions and zoom levels are controlled by the mouse as it is done in all 3D interpretation programs. Several mouse command modes and cursor shapes (for rotation, editing, selection, and image information) are provided. Optionally, the image aspect ratio can be set to remain constant during screen resizing by modifying the underlying OpenGL transformation matrices. For large objects containing hundreds of thousands of elements, efficiency may become a serous issue. We addressed this issue by automatic resampling the images during rendering, depending on the current viewable area and screen resolution.

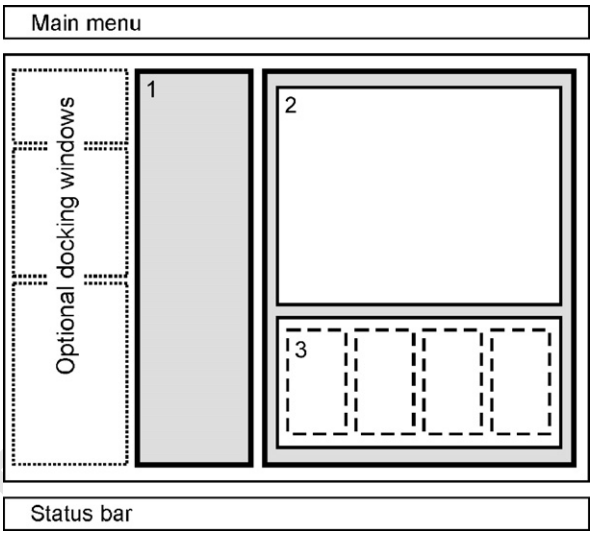

Fig. 3. Nested layouts for generation of complex co-ordinated displays. Here, top-level layout 1 is indicated by grey, lower-level layouts 2 and 3—by white colour and dashed contours, respectively.

An important enhancement of the viewer compared to the traditional 3D displays (such as GoCad and OpendTect) is the introduction of custom, user-configured "views," such as plan or map views, fence diagrams, frontal cross-sections, projections of 2D images into 3D, or preset zooms. These views are created by using tool "view3d" and are also represented by named data objects on the image tree (Fig. 2). In the image property editor, active views can be selected via drop-down menus, which allow quick transitions between them. Fig. 5 shows an example, and Appendix A illustrates the use of this method for creating 3D displays.

Any object on the image tree is allowed to implement an "autoplay" method which is called periodically by a separate thread on the viewer to perform animations. In particular, autoplay of the image object (Fig. 2) modifies the OpenGL transformation matrix causing continuous rotation, movement, or zooming the entire image in and out. Such animated displays may sometimes be helpful during data interpretation or presentations of the results. As with other options, these playback operations are configured by the user and included in job parameterization for tool "image" (see Appendix A).

From a programming standpoint, processing objects (tools) and most graphical objects are introduced into the IGeoS system by redefining functionality of the base `SIA_MODULE` class. For example, Fig. 5 shows a 2D visualization using our re-implementation of the popular ray-tracing program *rayinvr* (Zelt and Ellis,

[12] http://geowall.org. See also: http://geowall.geo.lsa.umich.edu/

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

1988). The corresponding tool in the IGeoS package is also called *rayinvr*, with its base class named, by our convention, `RAYINVR`:

```
class RAYINVR : public
   SIA_MODULE
{
  CHARSTR module_name();      ///< name appearing in the GUI
  int edit();                 ///< Edit phase (parameter input)
  boolean process();          ///< Process phase (data processing)
  UI_X *X();                  ///< Accessor for the graphical object
  boolean call(...);          ///< custom operation performed by
                              this tool
  ...
}
```

Here, method `edit()` defines the parameter input from the job (see examples below and in Appendix A) and `process()` describes the tool functionality during data (i.e., seismic trace) processing. Note the method `X()` returning a pointer to the object performing graphical representation of the model (Fig. 5), which is also derived from the base graphical base class named `UI_X`. When a "rayinvr" image is requested in the job, this object gets attached to the image tree (Fig. 2) and automatically propagated to the display server (Fig. 1).

On the display server, an identical graphical object is created, also placed on the image tree, and initiates two-way PVM communication with its counterpart in the flow. The object also provides all the necessary information for the rendering system. It builds its images by combining the available OpenGL plotting primitives (such as sequences and strips of lines, triangles, quadrilaterals, texts, markers, and bitmaps) which are further converted by the display server to OpenGL call lists, optimized, and rendered on the available hardware. In addition to serving the graphics, the object sometimes performs data analysis (such as seismic ray tracing and gravity modeling in this case).

Note the method `call(...)` in the example above, which is available in many IGeoS tools and object-tree objects. These methods perform custom data operations requested, for example, by buttons pressed on the user displays. In the case of "rayinvr", these calls initiate ray tracing, saving, or exporting the model into files (Fig. 5). In the following section, we show examples of how such interactive interfaces can be designed.

Finally, the display server supports stereoscopic displays by rendering OpenGL images in two frame buffers using slightly different viewing angles. This requires no changes to the client flows. We have tested this approach on our dual polarized-light GeoWall projector system. In addition, multiple display servers can be run in parallel while communicating with the same client, thereby producing multiple cloned displays.

5. Custom displays and user interfaces

An important consequence of using the abstract graphics protocol is the ability to construct 3D/2D, graphical, interactive tools by using regular processing flows. This is achieved virtually without computer programming and requires no pre-processing or creating any files other than flow scripts. The resulting tools can be invoked as standalone applications from the command line or, similarly to any other flow—from the IGeoS GUI. Below, we describe the general approach to building such displays by using examples from different subject areas. Samples of the corresponding job scripts can be found in Appendix A.

Generally, all displays are created by combining the following tools in the jobs:

(1) "Graphic"—selects or creates layouts, colours, line and fill styles, colour palettes, buttons, and other objects used in rendering. All these items are given names (identifiers) by which they can be accessed by other tools.
(2) "Image"—creates a sequence of objects from the data content, such as data grids and lines, and attaches them to the image object tree.
(3) "Gui"—sends the image trees to the display server.

Several instances of "graphic" and "image" can be used to create complex displays. Further details about these tools can be obtained from the tool index.[13]

In the first example, consider interactive seismic wide-angle ray-tracing and travel-time modeling. This is perhaps the single most important inversion approach commonly employed in wide-angle crustal seismic studies. This procedure requires painstaking analysis of multiple seismic phases using high-quality interactive graphics, which is limited even in the most popular modeling programs, such as *rayinvr* by Zelt and Ellis (1988). Several authors offered enhancements to this program, and Song and ten Brink (2004) developed a graphical interface for it.[14] Also, Zelt created a powerful editor for *rayinvr* velocity models by using Fortran graphics libraries PGPLOT,[15] and Gorman (2002) modified this program for working in spherical coordinates and made some modifications of its graphics. However, the above programs still do not support interactive 3D plotting or provide integration with other data displays. For efficient and comprehensive interpretation, it is desirable to be able to analyse seismic sections, refraction and reflection travel times, receiver functions (which describe crustal responses to teleseismic waves), and perform gravity modeling and tomographic inversion concurrently with modifying the model. The existing programs also offer graphics quality that is often below the modern capabilities of Qt and OpenGL. These and potentially numerous future improvements become readily available in our modular approach.

Because of its broad use in crustal seismic studies, we also maintain the "rayinvr" model format in our system while extending its functionality. The model now includes *P*- and *S*-wave velocity, attenuation, and density parameters, with additional superimposed density anomaly parameterization as in Geomodel.[16] Along with several enhancements (accurate correction for crooked-line geometry, detailed and non-surface consistent near-surface structures, and mixed-mode *P*- and *S*-wave ray tracing), the model now allows fully interactive data analysis.

By combining a single call to "rayinvr" with multiple instances of "image" and other tools, we obtain several displays shown in Fig. 4. Colour shading can be selected interactively for viewing the *P*- and *S*-wave velocities, velocity ratios, or attenuation parameters. Over 40 preset plus user-defined colour palettes are available, and buttons can be used to perform model saving and printing (Fig. 4). Wireframe displays of the model structure can be selected for model editing (Fig. 4b–d). The images have coordinated axes (i.e., horizontal zooming or translation of one image modifies the two others) and allow numerous display options. Ray tracing, gravity, and receiver function modeling are performed synchronously with model editing, and the corresponding elements are repainted as the user drags a model node or scrolls the mouse wheel to change the velocity or density. For the first time, complete crustal interpretation can be carried out in combined, interactive displays including all available data. Also note that in Figs. 4c and d, seismic depth conversion and receiver-

[13] http://seisweb.usask.ca/igeos/index/

[14] RayGUI 2.0—A graphical user interface for interactive forward and inversion ray-tracing, U.S. Geological Survey Open-file Report 2004-1426, http://pubs.usgs.gov/of/2004/1426/

[15] http://www.soest.hawaii.edu/users/bzelt/vmed/vmed.html

[16] Free 2D gravity and magnetic modeling tool by RockWare: http://www.rockware.com/

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

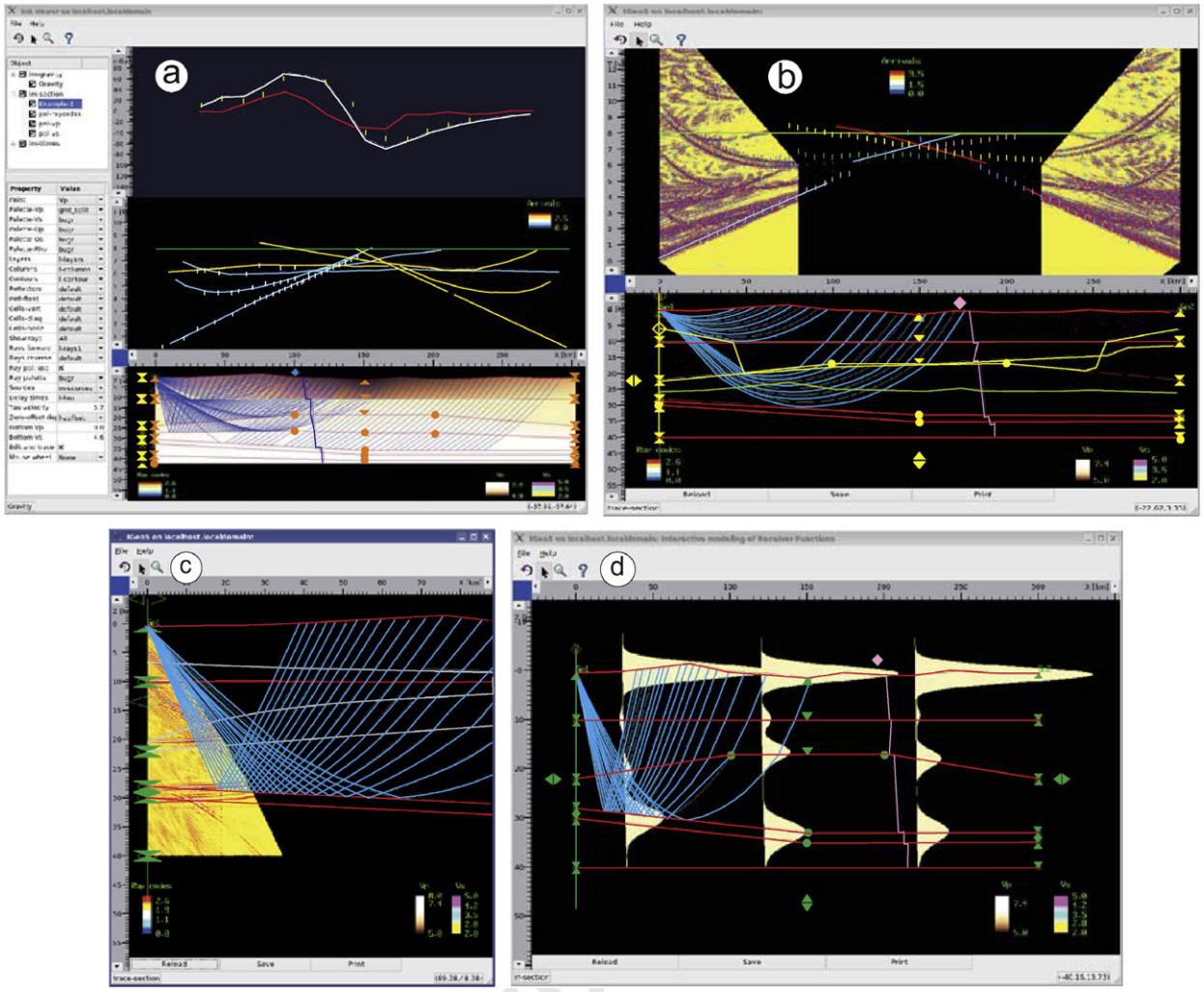

Fig. 4. 2D wide-angle ray tracing integrated with travel-time analysis of seismic phases and gravity modeling. Colour shading (grey in print) represents property selected by user (seismic velocity, Poisson ratio, attenuation parameters, or density). Depending on underlying processing flow, displays show (bottom to top): (a) model cross-section, travel times with ray amplitudes, and gravity; (b) same model (editing mode) with velocity contours and seismic rays, wide-angle and vertical-incidence travel times and seismic sections; (c) same model with seismic records converted to depth and seismic rays; (d) same model with teleseismic receiver functions modelled in it.

function modeling are performed by the same processing flows as conducting the display, without preparatory steps and proliferation of data files.

The 2D models similar to shown in Fig. 4 can be projected onto a cross-section surfaces in 3D to form fence diagrams showing the true geometry of the survey (Fig. 5; Appendix A). Such displays could be very important for most seismic surveys on land. A unique feature of the new display server is the availability of coastline data derived from databases distributed with the generic mapping tools (GMT) package (Wessel and Smith, 1995). Coastline contours (including rivers, channels, state and marine boundaries) are rendered directly in 3D. Only a specification of the target map region is required, and the resulting image can be combined with other objects and viewed interactively in 3D (Fig. 5). The levels of resolution are interactively selectable according to the available GMT databases, and line and fill colours are editable from the object property menu (Fig. 5). Note that the databases are accessed by the display server directly, and the PVM link is free from transferring large data volumes, resulting in efficient and fast displays. The model can also be combined with other objects, such as distributions of reflection points, crustal thickness maps (Fig. 5), wells, or other cross-sections. All modeling capabilities remain in this 3D display.

In a purely seismic example, Fig. 6a shows a synthetic reflection shot gather. Standard seismic display modes (one- and two-sided variable-area and variable-intensity displays) are supported. Trace gain, bias, and clipping can be set in the underlying flows and adjusted interactively in the property editor (lower-left part of Fig. 6a). Note that the three-component reflection synthetics shown here were also computed within the IGeoS package.

The generalized logical data processing model (Morozov and Smithson, 1997) and the new interactive capability allows IGeoS to operate as a real-time data collection system (compare to Antelope).[17] Fig. 6b shows a real-time flow displaying real-time data from our remote Internet seismograph. In the underlying processing flow, the data are continuously loaded from a network socket, processed on the fly (saved to disk, resampled, recut, filtered, events picked and saved in an external SQL database), and displayed in the form of continuously moving waveforms. Note that all this is done only by including the corresponding tools in the processing flow. In the near future, we intend to enhance the real-time monitoring system functionality and reproduce its

[17] http://www.brtt.com/software.html

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

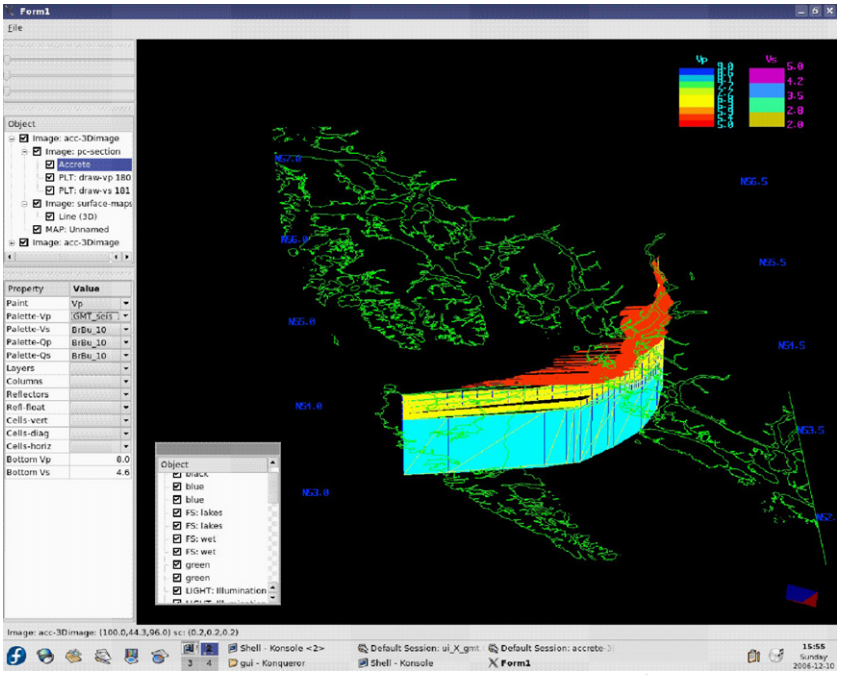

Fig. 5. Wide-angle crustal ray-tracing model from ACCRETE wide-angle seismic experiment (Morozov et al., 2001). Property editor allows switching between displays shading using *P*- and *S*-wave velocities, velocity ratios, or wireframe views. Red dots on surface indicate source-receiver midpoints. Coastline map is derived directly from GMT databases. Note editing controls in Properties menu. Rotation sliders (upper left) can be used for precise rotation around vertical and horizontal axes. Floating window (inset) summarizes graphical elements (colours, lines, palettes) that can also be edited in Property editor. See Appendix A for job scripts used in this display.

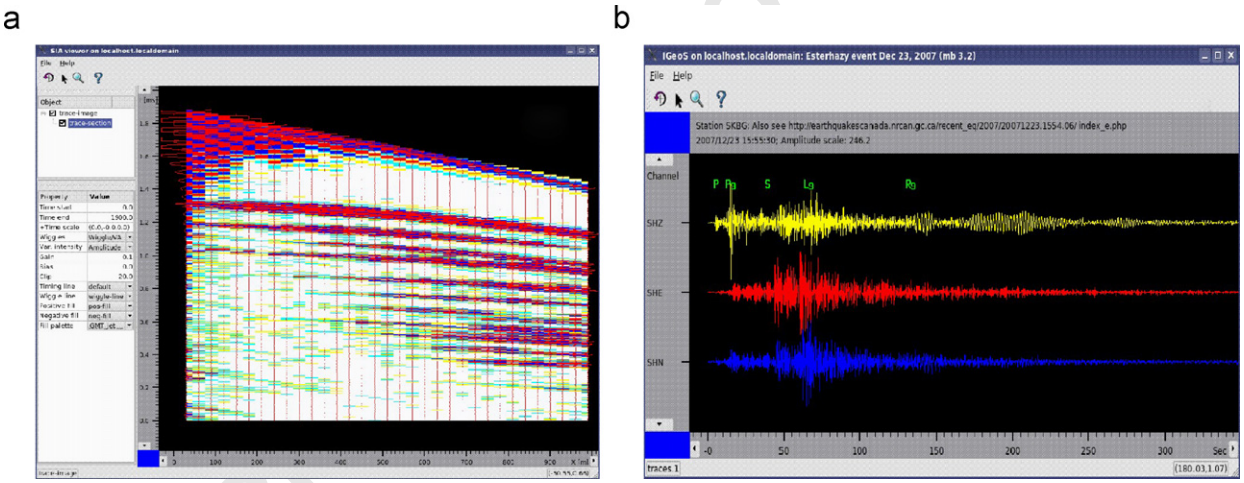

Fig. 6. Examples of seismic data displays: (a) synthetic reflection seismic record. Variable-area wiggle overlaid over variable-intensity amplitude display is used. Note interactive display options in Property editor (lower left). GMT "jet" colour palette (originally re-implemented from Matlab) used for trace background. (b) Three-component real-time seismic records from an Internet seismograph.

traditional look and feel closer to the popular Antelope[18] and Datascope[19] packages.

6. Discussion and further development

A novel and potentially the most useful result of this development could be in the enhancement of geophysical data analysis by integration of its many hitherto disjoint components. For example, with the new 3D viewing capability, ray-tracing models (Fig. 5) from multiple crossing lines can be inverted concurrently in common displays and performed together with gravity modeling and analysis of other data.

Apart from populating the displays with additional graphics objects, an important line of potential development could consist in expanding the interpretation-style functionality described in the Introduction. Tools and processing flows can also be sent through the PVM connection and placed on the image tree (Fig. 2), and therefore they can be associated with the various items in the same display. In such a way, the display could become a data integration hub, with programmable processing flows feeding various types of data into it.

Finally, the development of a user-customizable visualization server advances us to the ultimate goals of the project, which can be summarized, by using analogies from popular geophysical software packages, as follows:

[18] Documentation for Antelope Environmental Monitoring Software, Release 4.1: http://www.brtt.com/docs/arts.pdf

[19] Datascope tutorial: http://www.brtt.com/docs/datascope.pdf

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

(1) Open-source, modular seismic processing pipe similar to that of Seismic Un*x but with a significantly broader data model and processing logic;
(2) Modern graphical user environment and high-performance, common address space processing similar to ProMAX, Echos, or Omega;
(3) Parallel and distributed processing capability in excess of the above;
(4) Interactive general 3D visualization similar to GoCad or OpendTect;
(5) 2D/3D potential-field data analysis and inversion capability;
(6) Geophysical "toolbox" processing versatility and high-level coding style, ultimately approaching that of Matlab or Octave;
(7) Built-in access to "academic" GIS data and PostScript plotting, similar to GMT;
(8) Remote (Internet) data acquisition, real-time displays, database capability, similar to Datascope or Antelope, and access to open-source SQL databases;
(9) Web-service operation (we are aware of no analogs to date);
(10) Automatic software distribution and updating from source code and collaborative development.

Addition of graphical capabilities to the tools should improve the user experience and benefit most of the areas above.

7. Conclusion

A new 3D/2D interactive display server was developed for the IGeoS geophysical data processing framework (Chubak and Morozov, 2006). The server utilizes Qt and OpenGL graphics libraries and takes advantage of the object-oriented and content-agnostic design of the core IGeoS processing system. It operates by creating image object trees that are automatically propagated to the one or multiple server(s) residing on remote hosts, producing customizable, complex, structured and interactive displays. We show applications of this approach to several areas of geophysics.

With introduction of this component, the IGeoS system becomes conceptually complete and helps bridging the gap between the traditional processing and interpretation geophysical software. Its unusually broad scope includes: (1) high-performance, object-oriented data processing; (2) applications to many types of seismic and non-seismic geophysics; (3) parallel operation on multiprocessor computer networks; (4) processing web services; (5) support for collaboration and automatic software updating; and now (6) parallel, interactive, and animated 3D visualization.

Appendix A. Job example: 2D ray tracing model and GMT base map in 3D (Fig. 5)

As an example, we present fragments of the IGeoS job used to produce the 3D display shown in Fig. 5. We use the traditional spreadsheet job format used by older disco processing system. Note that this description simply represents a list of tools with parameters rather than a scripting language. Tool descriptions begin with tokens '*call'. The parameters are identified by the positions of the corresponding tab-delimited input fields. Note that tool parameterizations can be extensive and are structured by using "parameter lists" identified by predefined keywords. Some of these lists are indicated by commentaries in the example. Lists themselves may also span multiple lines and contain complex parameterization of the display.

```
#############################################################
# ACCRETE model in 3D
#############################################################


###### Job set-up. In particular, use ellipsoidal geometry
###### with kilometers for distance and depth units
#####################################################################


*setup
noapp
units           km              km


###### load the profile track line and midpoints from ASCII files
#####################################################################


*call           readtab         table           x-profile
x               real            1
lat             float
lon             Float


file    all       pc-coords.txt


*call           readtab         table           midpoints
lon             float
lat             float           1
z               float


file    all         ../MAP/midpoints-rev.table
                z               0.              # draw midpoints at z = 0.0
```


Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

```
### load the interpolated Moho depth grid
#########################################################

*call          readtab          map              moho
lon            float            1
lat            float            2
z              float

argrang        lon              −131.3           0.01            111
argrang        lat              54.6             0.01            61

file           all              moho_depth.xyz

###### create graphic elements
#########################################################

*call          graphic
backgr         black
foregr         black

# line and fill styles:

line           solid            1                gray                                    line-shore
line           solid            1                white                                   line-borders
line           solid            1                blue                                    line-rivers
line           dash             1                melon                                   line-grid

fill           none             blue                             fill-wet
fill           none             blue                             fill-lakes

# palette for Moho depth:

backgr         -same-
foregr         -same-           # foreground of the same color as top of the palette

#palette       pal-moho         gmtrainbow                       25.0            31.0
#palette       pal-moho         gmtjet                                           25.0            31.0
palette        pal-moho         buor                                             25.0            31.0

# buttons for interactive operation:

button         button_trace                      Trace rays
pc-section     exec             trace

button         button_save                       Save model
pc-section     exec             save

button         button_summary                    Print summary
pc-section     exec             print.summary

button         button_display                    Display data
pc-section     print            This option only prints in this demo!

###### Define 3D projectors
#########################################################

### this will project 2D images onto profile cross-section:
*call          view3d           fence3D
fence          x                lon              lat             x-profile
geom           sphere                                                            km

### this will project 2D images onto the surface:

*call          view3d           surface3D
geom           sphere                                                            km
```


Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

```
##### Create surface image
#####################################################

*call          image                 surface-maps
range-x        -131                  -130                    # longitudes
range-y        55                    56                      # latitudes

# map of the Moho

line           none

surface        lon                   lat             z                                       moho
pal-moho                             z                       # fill colors based on z (depth)

# scatter plot of midpoints:

line           points                3               red                                     line-midpoints

3dtabs         lon                   lat             z
midpoints

###### Create 2D Accrete Portland Canal section (included file)
#####################################################

*incl          pc-image.inc

###### Assemble the 3D image
#####################################################

*call          image                 acc-3Dimage
rangec3        400                                           # 200-km viewing box

# list selectable views, the first is the default:

view           From above                            -131.5  55                      0.              -20
               10.                   1.0
view           Globe                                         0                       0

# place 2-D cross-section in ''fence'' projection:

object         pc-section                                    fence3D

# place 2-D sirface maps in ''surface'' projection:

object         surface-maps                                                          surface3D

# draw in a GMT basemap at ''high'' resolution:

coast          -134                  -129            52      58                      high
surf           land                                          fill-dry
surf           sea                                           fill-wet
surf           lakes                                         fill-lakes
bndry          shore                                         line-shore
bndry          natnl                                         line-borders
bndry          state                                         line-borders
bndry          marine                                        line-borders
rivers         r                                             line-rivers
grid           line                                          line-grid               1.0             1.0

# Moho depth color bar:

cscale (km)    vert                  0.1             0.8     0.05                    0.2             Moho depth
pal-moho

# image lighting:
```


Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

```
light           0               0               1000            0.9                             Illumination
diff                            white
amb                             white
spec                            cyan

# buttons for interactive operation:

object          button_trace
object          button_save
object          button_summary
object          button_display

# animation (autoplay):

a-rot           0.1             60              30              1.0

##### start the display server
#####################################################

*call           gui
acc-3Dimage

# request docking windows:

dockers
otree           canvas                                          # object tree
otree           graph                                           # graphics DB tree
prop                                                            # property editor
```

Note that the job above uses an included file `pc-image.inc` containing a description of the 2D velocity cross-section along the Portland Canal line. This 2D image is projected onto a "fence diagram' along the line of cross-section and combined with a 2D distribution of a source-receiver midpoints and a coastline plot from GMT. Various editable items, such as coordinates, palettes, colours, and selections of displayed items do not need to be described in the job and are available automatically from the corresponding image objects. Finally, animation (rotation) about a tilted axis in 3D is also illustrated.

Script `pc-image.inc`used in this example, and which can also be utilized for a purely 2D cross-section display, is shown below.

```
### graphics elements

*call           graphic
backgr          black
foregr          black

palette         draw-vp1
-0.3            red                             0               blue
0               white                           0.3             red

# palettes for velocities

palette         draw-vp                         gmtseis                                         5.5
                8.4
palhue          draw-vs                         color           cont            2               5.
                4               0.05            .95

line            solid           1               green                           layers
line            solid           1               blue                            cells-vert
line            solid           1               yellow                          cells-diag
line            solid           1               red                             rays

fill            solid           green                           velocity

### RAYINVR 2D ray-tracing tool, loading model from file accrete.v.in1
*call           rayinvr         edit            ps              Accrete
vin             vp                                              accrete.v.in1
summary
```


Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005

```
### Form the 2D velocity/interface cross-section image

*call           image          pc-section                  ##### plot the model and rays
range-x         -150           200
range-y         -50            100

axis-x          100                        500          annot                                  border-pen
X (km)

axis-y          10                         100          annot                                  border-pen
Depth (km)

velmod          Accrete
layers                                                                                         layers
cells                                      vert                                                cells-vert
cells                                      diag                                                cells-diag
vs                                                                                             draw-vs
vp                                                                                             draw-vp

# colour bars

cscale          vert           0.8         0.8          0.05           0.2           Vs
draw-vs

cscale          vert           0.6         0.8          0.05           0.2           Vp
draw-vp
```

References

Chubak, G., Morozov, I., 2006. Integrated software framework for processing of geophysical data. Computers & Geosciences 32, 767–775.

Chubak, G., Morozov, I., 2007. Automated maintenance of geophysical software from distributed web repositories. Computers & Geosciences 33, 835–837.

Gorman, A.R., 2002. Ray-theoretical seismic traveltime inversion-modifications for a two-dimensional radially parameterized Earth. Geophysical Journal International 151, 511–516.

Morozov, I.B., 1998. 3D seismic processing monitor. Computers & Geosciences 24, 285–288.

Morozov, I., Reilkoff, B., Chubak, G., 2006. A generalized web service model for geophysical data processing and modeling. Computers & Geosciences 32, 1403–1410.

Morozov, I.B., Smithson, S.B., 1997. A new system for multicomponent seismic processing. Computers & Geosciences 23, 689–696.

Morozov, I.B., Smithson, S.B., Chen, J., Hollister, L.S., 2001. Generation of new continental crust and terrane accretion in Southeastern Alaska and Western British Columbia from P- and S-wave wide-angle Seismic Data (ACCRETE). Tectonophysics 341 (1), 49–67.

Stockwell Jr., J.W., 1999. The CWP/SU: Seismic Un*x package. Computers & Geosciences 25, 415–419.

Wessel, P., Smith, W.H.F., 1995. New version of the generic mapping tools released. EOS, Transactions, American Geophysical Union 76 (33), 329pp.

Zelt, C.A., Ellis, R.M., 1988. Practical and efficient ray tracing in two-dimensional media for rapid traveltime and amplitude forward modeling. Canadian Journal of Exploration Geophysics 24, 16–31.

Please cite this article as: Morozov, I., et al., Interactive 3D/2D visualization for geophysical data processing and interpretation. Computers and Geosciences (2009), doi:10.1016/j.cageo.2008.10.005