A Geoscience Perspective on Immersive 3D Gridded Data Visualization

Magali I. Billen [b,*] Oliver Kreylos [c] Bernd Hamann [d]
Margarete A. Jadamec [b] Louise H. Kellogg [b] Oliver Staadt [d]
Dawn Y. Sumner [b]

W. M. Keck Center for Active Visualization in the Earth Sciences (KeckCAVES)
[b] *Department of Geology, University of California, Davis*
[c] *Computer Science and Engineering, University of California, Davis*
[d] *Department of Computer Science, University of California, Davis*

Abstract

We describe visualization software, **Visualizer**, that was developed specifically for interactive, visual exploration in immersive virtual reality (VR) environments. Visualizer uses carefully optimized algorithms and data structures to support the high frame rates required for immersion and the real-time feedback required for interactivity. As an application developed for VR from the ground up, Visualizer realizes benefits that usually can not be achieved by software initially developed for the desktop and later ported to VR. However, Visualizer can also be used on desktop systems (unix/linux based operating systems including Mac OS X) with a similar level of real-time interactivity, bridging the "software gap" between desktop and VR that has been an obstacle for the adoption of VR methods in the Geosciences. While many of the capabilities of Visualizer are already available in other software packages used in a desktop environment, the features that distinguish Visualizer are: 1) Visualizer can be used in any VR environment including the desktop, GeoWall, or CAVE, 2) In non-desktop environments the user interacts with the data set directly using a wand or other input devices instead of working indirectly via dialog boxes or text input, 3) On the desktop, Visualizer provides real-time interaction with very large data sets that can not easily be viewed or manipulated in other software packages. Three case studies are presented that illustrate the direct scientific benefits realized by analyzing data or simulation results with Visualizer in a VR environment. We also address some of the main obstacles to widespread use of VR environments in scientific research with a user study that shows Visualizer is easy to learn and to use in a VR environment and can be as effective on desktop systems as native desktop applications.

Key words: 3D data visualization, virtual reality, immersive visualization, interactive exploration

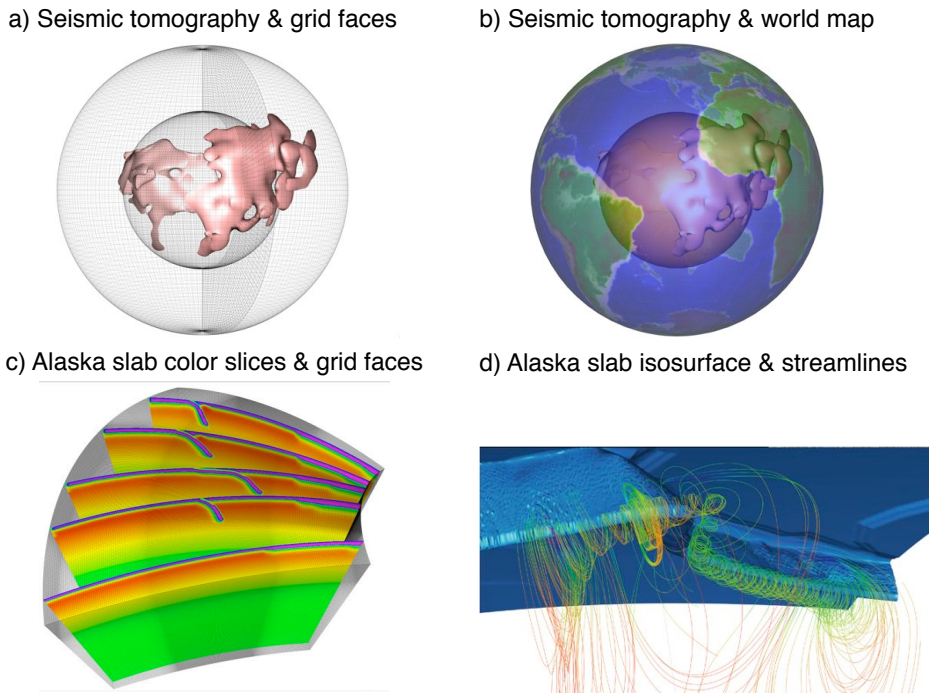

Fig. 1. Examples of geoscience gridded data sets illustrating the types of data that can be viewed with Visualizer. a) Seismic tomography model of mantle shear wave velocity with the spherical grid faces shown (Mégnin and Romanowicz, 2000). b) Seismic tomography model with world map superimposed for geographic reference. c) Color slices of the input viscosity structure for a subduction model of the Alaskan slab. d) Isosurface and streamlines for a numerical simulation of mantle flow near the Alaskan slab. See Movies 1–3 for demonstration of the Visualizer software.

1 Introduction

Geoscientists work with diverse data sets ranging in spatial scales from nanometers to thousands of kilometers and varying on time scales from femtoseconds to billions of years. Typically, these observations, or the results of numerical simulations, are 3D gridded data sets: seismic tomography images of the Earth's interior, finite element models of mantle convection, atmospheric mixing and oceanic currents, or fossils imaged in ancient rocks. Analyzing and interpreting these large volumetric data sets with information at multiple scales poses a significant challenge in geoscience research – one that can be addressed by the scientific visualization community, particularly through innovative use of interactive and immersive virtual reality (VR) environments (Lin and Loftin, 1998; Rhyne and MacEachren, 2004; Kreylos et al., 2006b). For example, analysis of complex geodynamic simulation results from one

* Corresponding Author
Email address: billen@geology.ucdavis.edu (Magali I. Billen).
URL: www.keckcaves.org (Magali I. Billen).

of the world's largest supercomputers, the Earth Simulator in Japan, has been found to benefit from visualization and direct manipulation in a CAVE VR system (Furumura and Chen, 2004; Ohno and Kageyama, 2007).

A substantial component of research in the geosciences involves identifying the most important processes in natural systems and developing computational models of key interactions. One efficient way to identify unknown processes is to look for correlations within and among data sets. Quantitative evaluation of correlations requires the scientist to have a detailed conceptual model of the underlying relationships; however, the fundamental processes are poorly constrained for many of the problems at the forefront of research. Thus, correlations can be extremely difficult to predict and extract mathematically. In addition, while statistical and graphical information derived from computer simulations or observational data are important characterizations, by their very nature, they limit the amount and type of information conveyed. In these cases, relationships among data are best identified by examining data visually in a flexible, interactive environment that takes full advantage of the human brain's ability to visually identify patterns, outliers and unexpected or abnormal features. Such visual examination of data can lead to the conceptual framework necessary to develop quantitative methods for further analysis.

Different software packages provide different methods with which to display and manipulate 3D gridded data sets, from clicking and dragging with a mouse to entering values via a keyboard. Throughout this article we use the term "interactive" to describe software that allows the user to directly manipulate visualization objects in real-time in a manner similar to how one would interact with a real object. To give two examples, 1) as opposed to the IVS 3D Fledermaus software (Fledermaus, 2007), where users change views by manipulating special rotation widgets with the mouse, Visualizer allows users to directly grab and rotate an object with the mouse or other input device, and 2) instead of entering the value for a desired isosurface numerically, Visualizer allows users to create isosurfaces passing through any point in the data's domain by selecting a point with the mouse or other input device, and to drag newly created isosurfaces in real time. This type of direct interactive visualization provides many advantages over other visualization approaches, not the least because it is intuitive and more natural than other approaches. However, it also places high demands on the software to respond in real time and to reproduce a real-world-like virtual space for data analysis. For example, in an immersive environment such as a CAVE, a user can create an isosurface by touching a point in the data's domain using a tracked input device such as a wand, leading to very natural interactions. On a desktop system, on the other hand, the software has to provide more involved interaction mechanisms to support intuitive and unambiguous 3D point selection using only the mouse, a 2D input device.

In 2003 a collaboration between computational scientists and geoscientists at U.C. Davis directed at developing scientific visualization methods and effective systems was formed in conjunction with the W. M. Keck Center for Active Visualization in the Earth Sciences, *KeckCAVES* (http://www.keckcaves.org). This collaborative effort is currently concerned with a range of scientific visualization problems including data quality and scientific analysis of LiDAR (Light Detection And Ranging) data sets of 10–350 GB in size (Kreylos et al., 2006a; Bawden, 2006), mapping on digital terrain data sets (Remote Interactive Mapping System (RIMS), Bernardin et al. (2006)), and model assessment and scientific analysis of 3D gridded data (Jadamec and Billen, 2006) (Figure 1). We discuss one of the software systems emerging from this collaboration, *Visualizer*, developed for analyzing 3D gridded data sets in an interactive,

immersive VR environment. Although the underlying VR Toolkit that Visualizer is built on (Vrui, (Kreylos et al., 2006a)) is capable of interfacing with real-time simulations to visualize dynamically evolving objects (Tipper, 1991), Visualizer has been developed to optimize the interactive experience with a single, unchanging, object. This software and instructions for installation and use are available for free download from the KeckCAVES website (http://www.keckcaves.org/software/index.html).

Visualization of 3D gridded data is not new, and is certainly not new to the Geosciences. For example, the results of 3D structures in seismic tomographic images and mantle convection models have commonly been presented as combinations of isosurfaces and color-mapped slices (Mégnin and Romanowicz, 2000; Tackley, 2000; Furumura and Chen, 2004; McNamara and Zhong, 2005; Wang et al., 2007), 3D seismic reflection-refraction data is presented as 3D rendered volumes or movies of color slices (Lin and Loftin, 1998; Gao, 2003; Ma and Rokne, 2004), and 3D reconstructions of fossils has been used to study biomechanics of extinct species (Kalvin et al., 1992; Sutton et al., 2001; Zollikofer and de Leon, 2005; Motani et al., 2005). However, visualization software has historically been divided between desktop and VR environments, forcing users to learn to use two or more applications in order to move between the realms. Additionally, the design of the software was focused more on producing a final image than facilitating exploration of the data set. In the following sections we discuss the motivation for developing a new software package for visualizing 3D gridded data in the Geosciences (sections 1.1–1.2), how this software is similar to or different from other existing software packages (section 2.0), how the software achieves real-time interactivity (sections 3.0–4.0), and three case studies and a user study illustrating how the software facilitates efficient and effective data exploration and analysis (section 5.0). Visualizer is a complex software package built upon a complex development toolkit for virtual reality applications (Kreylos, 2006). Therefore, we limit our discussion of its design and implementation to those aspects that users will most directly interact with, and that are essential for providing an intuitive environment for data exploration. As it is sometimes difficult to fully communicate the experience of using this software in a CAVE environment using words and still images, we strongly encourage the reader to view the accompanying movies showing the software being used in a CAVE (Movies 1 and 3) and on the desktop (Movie 2).

1.1 Why Use a 3D Immersive Environment?

Research on human visual perception of the world attests to the complexity and power of human visual ability. We perceive a 3D world primarily through a combination of binocular vision and the use of motion parallax (Harris, 2004). However, the brain obtains many clues as to the depth and shape of objects from the surrounding environment (ground-plane, shadows, relative motion). Immersive environments allow scientists to take advantage of the way the brain already interprets visual information (Hubona and Shirah, 2005) and provides key advantages for evaluating and analyzing Earth science data and simulations, including use of peripheral vision to provide global context, body-centric judgements about 3D spatial relations and enhanced 3D perception from stereo and motion parallax (head tracking) (van Dam et al., 2000). Together these spatial indicators create a more natural environment and thus promote more efficient exploration of 3D data. On a regular computer screen, the strongest available depth cue is motion parallax. As a result, to understand 3D data from images on a screen, users need to constantly rotate the view or rock it back and forth to perceive depth. This motion

interferes with detailed examination or measurement of the displayed data, and is not required in stereoscopic environments where other depth cues are available. Furthermore, immersive environments – offering head tracking in addition to stereoscopy – enable motion parallax without the user having to consciously move the 3D data: motion is simply created by moving one's head or walking around the data; an intuitive response that does not interfere with analysis tasks.

While the use of proprietary software in 3D immersive environments in industrial applications has spread in recent years, and despite the possible advantages for scientific discovery, the use of immersive technology in scientific research has not grown as quickly (van Dam et al., 2000). Several concerns arise in evaluating the role of 3D immersive VR for scientific applications. Is anything new learned from viewing data in a 3D immersive VR environment? How do we quantify what is seen? Does the 3D viewing provide a real advantage over viewing multiple 2D slices or 2D images on a computer screen? Does using a 3D immersive environment save time? Are 3D immersive environments sufficiently accessible/available? The answers to each of these questions vary depending on the data type, size of data, and type of 3D environment considered. For example, only a few academic research groups (less than 10) in the U. S. have regular access to a CAVE environment, but more than 500 have access to a GeoWall (Johnson et al., 2006). With the continued decrease in cost of stereo-capable projection systems based on polarization ("passive stereo"), frame interlacing ("active stereo"), or other methods, GeoWall-type 3D immersive environments could become commonplace. However, this will only happen if software is already available that is easy to use and provides all the capabilities scientists require.

As we look to the future of larger and more complex data sets and model results, we would like to prepare for the time when the current ways of carrying out scientific analyses are no longer effective or time-efficient. Is it possible to create software for 3D immersive VR environments that is easy-to-use and provides an effective and efficient tool for scientific analysis? The range in types of data and applications found in the Earth Sciences provides an ideal testing ground for developing such software. We present one software system designed with this long-term perspective in mind. This software does not provide everything one would need or want, but it demonstrates one way of bridging the visualization methods we are accustomed to on the desktop to a new way of analyzing data in 3D immersive VR environments.

1.2 Visualizer: 3D Volume Visualization Software.

It has long been known that graphical representations of complicated data sets on 2D displays provide efficient and insightful ways of interpreting quantitative data (Tufte, 1983), but similar analysis of 3D data sets using 3D visualization has lagged behind. One reason may be that the design of 3D volume visualization software has traditionally focused on providing an environment for users to create a final image of a data set that is effective at communicating ideas and results (TecPlot, 2006; Vis5D, 2006; CAVE5D, 2006). This type of visualization software usually facilitates enhancing the appearance of structures of interest, or synthesizing various data types, e. g., isosurface of temperature, color-mapped slices of viscosity or stream-tubes of fluid flow in a numerical fluid dynamics model. However, this design objective is focused on visualizing known structures, such as when the value of the best isosurface to display is already known. However, this approach does not support exploring a data

set in which the features of interest are yet to be discovered. Exploring a data set visually in order to identify features or processes of scientific interest requires an interactive environment and real-time visualizations that can be modified on-the-fly. In addition to making the software adaptable to different geoscience applications, it must be straight-forward to load data sets of various formats into the software, and easy to learn and use.

The 3D volume visualization software presented here, called Visualizer, has been specifically designed for highly interactive 3D VR environments and therefore follows different design principles than software that was originally developed for use in a 2D desktop environment (Bowman et al., 2004). For example: 1) navigating (picking up, rotating and translating a slice or isosurface) in a VR environment is simply done by moving a tracked input device, such as a wand, while pushing a button, 2) intuitive direct manipulation, such as creating isosurfaces by touching a seed point using a tracked input device, is favored over indirect methods such as entering numbers, and 3) creating isosurfaces or slices occurs in real time as an input device is moved through a virtual data set.

2 Related Work

To put Visualizer into context, and compare it to other 3D visualization software for desktop and immersive environments, we first discuss several desktop applications, and then several applications developed for VR (Table 1).

TecPlot (TecPlot, 2006) is a commercial visualization software for 3D volume data in a desktop environment, and is widely used in the Earth sciences. Its main goal is the production of publication-quality graphs and figures, but it also contains several features that make it applicable to visual data analysis. Its main volume *visualization elements* are color-mapped slices and isosurfaces, and the program allows one to change the set of elements interactively. However, interaction is limited to entering desired isovalues into dialog boxes or dragging axis-aligned slices through a data set using sliders in a dialog box. The program's response to changes is not in real time: especially changing an isovalue causes delays of tens of seconds before the display is updated. We found that directly observing the changing shape of an isosurface under varying isovalues is a very powerful analysis tool (see Movie 3); the fact that TecPlot does not support this style of exploration is a major limitation. Additionally, TecPlot's navigation methods are limited when a user wants to explore a small feature in a larger data set in detail. Although TecPlot provides the usual virtual trackball interface, it can only rotate the data around its centroid, not around arbitrary 3D points. This, and the fact that TecPlot reduces the resolution of surfaces during navigation, severely limit detail analysis. TecPlot does contain a measurement tool to query the spatial position of and data values at arbitrary locations, but it is not intuitively clear how TecPlot translates a 2D mouse position into 3D space for measurement.

Vis5D (Vis5D, 2006) is an open-source visualization software aimed at time-varying, multivariate 3D volume data. It is often used in the Earth sciences, especially in atmospheric science. Its main goal is the production of figures and animations. Vis5D's main visualization elements are color-mapped slices and isosurfaces, but it also supports direct volume rendering. The level of interactivity of Vis5D is similar to TecPlot's, with the same limitations for visual data analysis, but Vis5D contains some

Examples of Commonly-Used Software

Software	Positive Features	Negative Features
Desktop Applications		
TecPlot	- easy to learn - easy to input data - multiple input data types - measurement tools - virtual trackball rotation (VTR)	- enter slice location/desired isovalues into dialog box - response to use changes is not real-time (>10-60 s) - image resolution reduced during rotation/translation - difficult to determine location of measurement in 3D - VTR limited to rotation around data centroid
Vis5D	- easy to learn - measurement tools - direct volume rendering - slices can be dragged through data - VTR around screen-center - animations	- enter slice location/desired isovalues in dialog box - difficult to determine location of measurement in 3D - long rendering times for isosurfaces
VR Environment Applications		
CAVE5D	- direct port of Vis5D - slices can be dragged through data	- slices must be dragged by box corner - change isosurface value in virtual dialog box - model rotates around its centroid or user's head - non-intuitive navigation is difficult to learn
NASA's Virtual Wind Tunnel	- developed for VR environment - direct 3D manipulation of data - real-time rendering - seeded slices and isosurfaces - grab space navigation - grab space data manipulation	- only portable to limited range of VR environments - only one grid format for input

Table 1
Examples of visualization software commonly-used in the geoscience community summarizing some of the positive and negative features of the software evaluated for interactive 3D visualization.

improvements: slices can be dragged by direct manipulation with the mouse, and the virtual trackball for navigation always rotates around the screen center, improving the user's ability to examine small features in detail. Vis5D's volume rendering feature uses a simple slice-based algorithm, and is due to its long rendering times is not applicable to interactive exploration.

CAVE5D (CAVE5D, 2006) is a direct port of Vis5D to immersive environments based on the CAVE (Computer Assisted Virtual Environment) library (Cruz-Neira et al., 1993). It runs in VR environments compatible with CAVELib, and uses a CAVE wand to control the visualization. Even though CAVE5D was introduced in late 1995, it is still used for Earth science visualization in immersive environments, especially CAVEs. One reason might be that it allows scientists to visualize data on the desktop first using Vis5D, and then to import the visualizations into a CAVE (e.g., a four-walled immersive

visualization environment).

The development of CAVE5D from Vis5D is a good example of the challenges posed by porting desktop software to immersive environments. The main benefits of VR, intuitive navigation and direct manipulation of 3D objects, are not realized because the original desktop program does not contain functionality to support them. For example, Vis5D allows a user to drag a slice by manipulating it with the mouse, and CAVE5D uses the same mechanism, but based on a 6-DOF input device. Instead of just moving the device to a position of interest and pressing a button to create a slice at that position (or drag an existing slice), the user has to aim the device at an "interaction box" at the corner of the slice to drag the slice along its axis. This makes it quite difficult to change a slice while zoomed-in to examine a feature, and is not the most appropriate way of using a 6-DOF input device to manipulate a 3D object. Interestingly, isosurfaces are still changed by numerically entering a desired isovalue; however, since VR environments have no keyboards, users have to use the wand to enter numbers via a virtual 3D keypad. This style of interaction is actually less effective in a VR environment than on a desktop. Navigation also does not take full advantage of interaction using a 6-DOF input device: rotating the wand causes the data set to rotate, but not around the current position of the wand. Instead, the model rotates either around its centroid or the user's head position, depending on the navigation mode. Both navigation modes are hard to learn, and even experienced users sometimes have problems moving a model in the desired way. As a result, CAVE5D is mostly used to present previously created visualizations in a more impressive environment, and not to create or analyze visualizations by interactive exploration.

The NASA virtual wind tunnel (Bryson and Levit, 1991; Meyer and Globus, 1993; Bryson, 1996) is an even older application than CAVE5D, but it was directly developed for immersive environments and takes into account the particular benefits and constraints of VR. Its main purpose, as the name implies, is the analysis of computational fluid dynamics data, but it could be used for other 3D volume data as well. Its main visualization elements are streamlines/streaklines, particle traces, color-mapped slices, and isosurfaces. As opposed to CAVE5D, all visualization algorithms are optimized for direct 3D manipulation and real-time feedback. For example, streamlines are created by directly selecting their starting point in 3D space, and isosurfaces are created by growing them from a selected seed point in space, instead of specifying their isovalue. Isosurfaces are based on time-outs, i. e., the result of creating a surface will be visible in the display in less than 0.1 s, enabling direct observation of an isosurface's change as the seed point is dragged (see section 4.2 for description of seeded algorithms). Navigation is also intuitive: users can "grab space" using a 6-DOF input device, and then reposition the data set by moving/rotating the input device. Overall, the virtual wind tunnel is an effective visual analysis application. Its main limitations are that it only supports a single grid format and that it is only portable to a very limited range of VR environments.

3 Interactive Exploration of 3D Volume Data

In the development of the Visualizer software, we followed many of the design principles first exhibited by the NASA virtual wind tunnel (Bryson and Levit, 1991) and later described in more detail in (Sherman and Craig, 2002). These design principles are summarized in Table 2 and described in more detail

Key Design Features of the Visualizer Software

Element	Description
Virtual Reality	High Frame rates (>30 Hz) for head-tracked stereo-viewing – Real-time response to user interaction (< 1/10 s)
Interaction	Display can not "freeze" during rendering – intermediate results presented in < 1/10 s – generated triangle set rendered in < 1/30 s Direct manipulation for navigation, element creation and manipulation – grab space ability – point-and-click selection of visualization elements
Portability	Use same software in multiple environments – built on VRUI development tool-kit – Visualizer is completely independent of underlying VR environment – Optimal matching of interaction patterns to capabilities of underlying environment
Multiple Grid Formats	Supports wide range of input data types – regular (Cartesian), hexahedral (curvilinear), simplical (tetrahedral) grids, and heterogeneous FEM meshes – optimized visualization algorithms implemented for different data types

Table 2
Key design features in the Visualizer software required for interaction in VR environments and use in geoscience applications.

here. Three movies also illustrate how these design principles create an efficient and effective tool for exploration of 3D volume data on the desktop or in a CAVE (see online supplementary material).

The two main constraints of VR are the high frame rates upwards of 30 Hz required for head-tracked stereo viewing, and real-time response to any user interaction within 1/10 s (Bryson, 1996; Kreylos et al., 2001). These constraints influence the implementation of all visualization algorithms, whose "standard implementations" typically do not observe them. A standard Marching Cubes (Lorensen and Cline, 1987) implementation, for example, might require several seconds or minutes to generate millions of triangles for larger data sets. An interactive and immersive implementation of this algorithm must ensure that the display does not "freeze" during that time, that at least intermediate results are presented to the user after at most 1/10 s, and that the generated triangle set can later be rendered at frame rates upwards of 30 Hz (Bryson and Levit, 1991). These goals require multithreaded programming (Lewis and Berg, 1998), special algorithms such as seeded isosurfaces (Meyer and Globus, 1993; Kreylos et al., 2001), advanced rendering using compressed geometry or multiresolution methods (Engel et al., 1999), and careful optimization of the extraction and rendering algorithms.

The main benefits of VR are the direct 3D perception enabled by head-tracked stereoscopic displays and the ease with which users can select positions in 3D space using 6-DOF input devices. To fully exploit these benefits, visualization software has to follow a direct manipulation approach (Shneiderman, 1983) across the entire range of functionality, from navigation over creation of visualization elements to quantitative analysis. If users see a feature of interest in a data set, they must be able to quickly

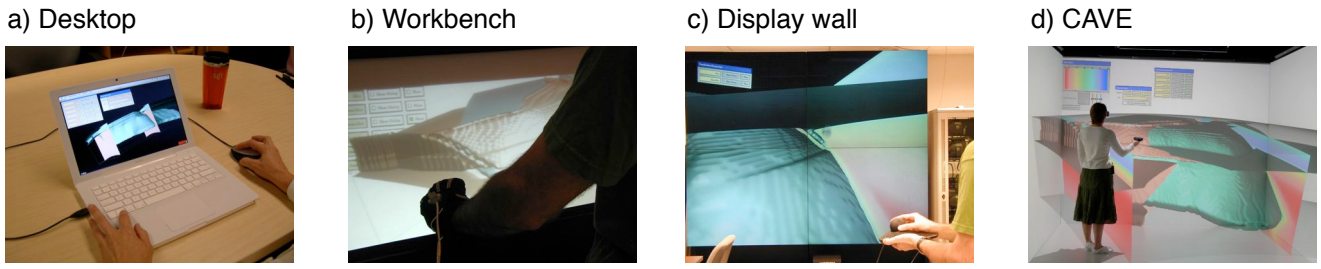

Fig. 2. The Visualizer application in four different VR environments: a) desktop, b) responsive workbench, c) tiled display wall, and b) CAVE. Visualizer achieves this cross-system portability by being built on the VRUI development toolkit (Kreylos, 2006), being completely independent of the underlying VR environment and optimally matching interaction patterns to the capabilities of the underlying environment.

create appropriate visualization elements to explore the feature in more detail; it is not appropriate to first have to measure the position of or the data value at the feature, and then enter those numbers into text fields to create slices or isosurfaces. Instead, VR software should allow users to create elements by "point-and-click." For example (Movie 3), when looking for connections between fossil structures in rocks, dragging the wand through the data to create isosurfaces allows the user to determine if the feature of interest is isolated or at what iso-value it is connected to nearby structures.

An additional important design goal is the ability to run the visualization application effectively on a wide range of VR environments with different sets of input devices, including desktop environments with only a mouse and keyboard (Figure 2). Our experience has shown that scientists are reluctant to use VR software because it forces them to use a (shared) VR environment for all their analytical work, or become proficient with two or more different visualization applications. Being able to use the same VR software on the desktop or low-cost GeoWall-like environment first for preview generation and initial quality assessment, and then only perform important and detailed explorations in VR alleviates these concerns and should lead to a wider acceptance of VR methods in scientific domains. To achieve maximum portability, we developed Visualizer on top of the Vrui VR development toolkit (Kreylos, 2006). Vrui supports highly interactive and high-performance VR applications that are completely independent of the underlying VR environment, including the type and number of available input devices. For example, Visualizer's direct manipulation paradigm relies on a user's ability to select arbitrary points in 3D space. In an immersive environment, this is achieved by directly touching points with a 6-DOF input device. On the desktop, on the other hand, users are limited to 2D interactions using mouse and keyboard. Here, Visualizer (through Vrui) implements 3D point selection as an extension of the virtual trackball-based navigation mechanism. Users can rotate the view around the current screen center, can translate (pan) the view parallel to the screen plane, and can translate (dolly) the view orthogonally to the screen plane. This means it is possible, and efficient after some practice, to bring any arbitrary 3D point into the screen plane, and then select it by clicking it with the mouse. As a result, Visualizer's approach to data exploration works well even on desktop systems, and our user study, discussed in Section 5, shows that while Visualizer is very effective in a CAVE, it is also effective in a desktop environment – sometimes as effective, or even more so, than other native desktop visualization applications.

Finally, we designed Visualizer such that it can be applied to a wide range of data formats. The differ-

ences between data formats such as regular (Cartesian), hexahedral (curvilinear), simplical (tetrahedral) grids, and heterogeneous finite-element meshes are so fundamental that each normally requires its own implementation of all visualization algorithms (Shen and Johnson, 1995; Lorensen and Cline, 1987). As VR software must be carefully optimized to satisfy VR's real-time constraints, a software architecture should allow easy experimentation with different algorithms and data structures. For example, developers might have to change the representation of isosurfaces from individual triangles to indexed triangles or triangle strips to evaluate which performs best on a given system (Engel et al., 1999). If there are separate implementations of the underlying algorithms for different data formats, applying such changes while keeping all versions working together is a major software engineering challenge. As a result, most visualization programs, especially VR visualization programs, support only a single data format. Visualizer is based on a separation between data formats and algorithms that allows one to develop visualization algorithms only once, and apply them to all supported data formats. In fact, Visualizer only contains a single piece of code implementing its isosurface extraction algorithm, and this code is applied to regular, hexahedral, and tetrahedral grids (for more on isosurface extraction methods, see Shen and Johnson (1995); Lorensen and Cline (1987)). This data format and algorithm abstraction uses the C++ template mechanism (Alexandrescu, 2001; Stroustrup, 1997) to create highly efficient code that, according to our experiments, performs on par with, and sometimes out-performs, other code developed directly for a specific data format.

4 System Architecture

The Visualizer software was designed with maximum modularity in mind, as a toolbox of generic interacting components, each encapsulating a particular functionality. While it is beyond the scope of the paper to describe each component of the software package, we provide references which describe these elsewhere and focus the discussion on those aspects that the user interacts with and are directly related to achieving the real-time, direct manipulation of data. The lower-level components of this architecture have been described previously (Kreylos et al., 2001), but have since been completely redesigned for better performance, and the higher-level components and the module system have been added. The overall architecture of the new component toolbox is shown in Figure 3.

While many of these individual components are not new to the visualization community, our contribution lies in integrating these pieces into a software package that is geared towards data exploration and is portable to a wide range of VR or desktop environments. Components fall into three basic categories: *Data representation*, *visualization algorithms*, and *visualization elements*. Visualizer uses the C++ template mechanism to combine concrete instantiations of all components required to visualize a particular data format, and a *data file reader* required to load individual data set formats, into a *visualization module*. These visualization modules are packaged as external plug-ins, and loaded into the overall Visualizer application at run-time when a user requests loading a data file. A module advertises to the overall application the scalar and vector variables contained in the current data set, and the visualization algorithms that can be performed on it. Visualizer then creates a graphical user interface to allow users to select variables and algorithms and assign them to input device buttons. The result of executing a visualization algorithm on a data set is a visualization element, e. g., a volume, a color-mapped slice or an isosurface. Elements are stored in a scene graph inside the Visualizer

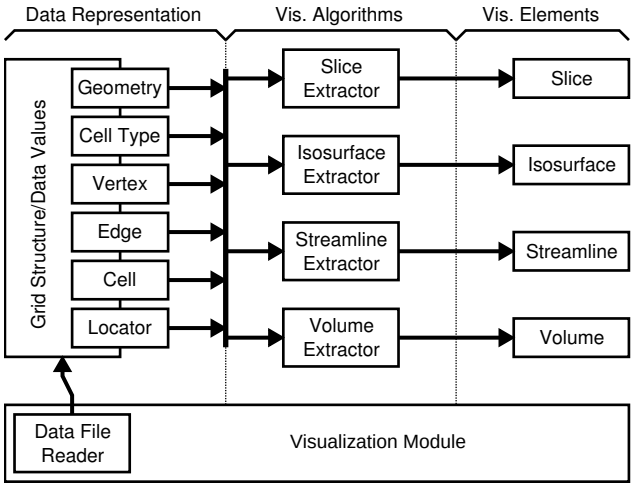

Fig. 3. Visualizer's system architecture. Modules are plug-ins encapsulating the data structures and algorithms to visualize a particular data format, and are created by linking concrete instantiations of the underlying generic components.

application and can be toggled on/off and deleted individually.

The C++ template mechanism is very powerful for creating component architectures. As opposed to run-time polymorphism, where descendants of the same base class can only differ in the implementation of virtual functions, templates allow one to additionally use different data types, supporting more powerful abstractions (Alexandrescu, 2001). Furthermore, templates are instantiated and linked at compile-time, allowing the compiler to perform full optimization on the generated code. As a result, generic code often performs as well as specialized code, and sometimes better than specialized C code due to the compiler's ability to optimize across function calls (GCC Compiler, 2007; Alexandrescu, 2001). To combine the benefits of high performance and run-time polymorphism, our module architecture links sets of closely interacting components at compile-time into larger polymorphic modules that only loosely interact with the overall application.

4.1 Data Representation

The core component of all visualization modules is the representation of the visualized data set. Visualizer currently supports regular (Cartesian), curvilinear (hexahedral) and unstructured (tetrahedral) grid structures, commonly found in finite element simulations (Hughes, 2000). The interface between data representations and visualization algorithms is implemented as a set of utility classes giving access to the underlying geometry of a data set, i. e., the dimension and scalar type of its domain space, the type of its cells, currently simplices or n-dimensional cuboids, and the vertices, edges, and cells defining the data values and grid structure. A final accessor class, *Locator*, is a spatial iterator: it makes it possible to query data values and local grid structure at arbitrary positions inside the data set's domain. Depending on the grid structure, data representation components contain acceleration structures to query the neighborhood relationships between cells, and to support the interface to the Locator.

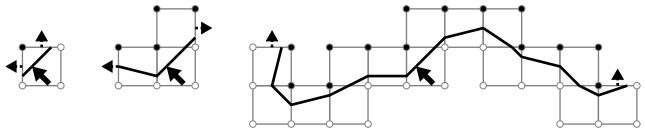

Fig. 4. Creation of a seeded isosurface. Extraction starts with the cell containing the Locator. After each cell is processed, the algorithm determines into which neighboring cells the isosurface extends, and adds those to the queue of pending cells. Processing the queue in order causes the isosurface to grow outwards from the cell containing the Locator. The black and white dots denote grid vertices whose value is above and below the isovalue (the interpolated data value at the Locator's position), respectively.

4.2 Visualization Algorithms

Visualization algorithms create visualization elements such as slices and isosurfaces based on the grid structure and data values of a data set, and the position/orientation of a Locator. Although Visualizer contains "global" algorithms such as isosurfaces specified by isovalue and slices specified by position and orientation, its user interface focuses on direct manipulation, i. e., the creation of elements based only on a point/orientation of interest: this is a key feature that distinguishes Visualizer from other VR software (Table 1). Visualizer currently supports color-mapped slices, isosurfaces, streamlines, and volume rendering (Hansen and Johnson, 2004). In accordance with the direct manipulation approach, and to provide more meaningful immediate feedback to users, most algorithms are *seeded* implementations (Meyer and Globus, 1993; Kreylos et al., 2001). As illustrated in Figure 4, a seeded algorithm does not create visualization elements by processing an entire data set cell-by-cell, but instead starts element creation from the cell containing the point of interest, and traverses all other cells containing the same element radially outwards. This means that any intermediate results created by seeded algorithms provide local information in an area around the point of interest, and allow a user to explore a region of a data set by moving the point of interest while observing the change of the element's shape as it is dragged along (see movies). Once the user stops dragging, the partial element is created to its full extent, or to the maximum number of graphics primitives the display system can render in its alloted frame time (see movies).

The main benefit of a generic component architecture is that algorithms can be expressed independently of grid format. For example, Visualizer contains only a single implementation of isosurfaces, which is applied to all grid formats. By using the interfaces described in the previous section and listed in Table 2, the isosurface algorithm only contains the logic of how to create isosurface fragments inside a single cell based on vertex values, how to traverse all cells containing the isosurface, and how to use timers to satisfy real-time constraints. Any additional required information, such as triangulation case tables, cell neighborhood information, and the formulas used to interpolate vertex positions/data values along cell edges, are provided by the data representation interface classes.

Another benefit of a generic component architecture is the ability to provide specialized implementations of components. For example, there are many different ways to volume-render 3D data, and some of the highest-performance ones only work on particular data types (Hansen and Johnson, 2004). In these cases it is possible to provide special-case components, and the C++ compiler will use them automatically when possible. For example, Visualizer's generic volume rendering algorithm is based on blending color-mapped slices (Reed et al., 1996); the specialization for regular (Cartesian) grids uses

hardware-assisted volume rendering based on 3D textures (Cabral et al., 1994) to achieve frame rates high enough for immersive visualization.

4.3 Visualization Elements

Visualization elements, such as color-mapped slices or isosurfaces, are produced by visualization algorithms, and stored in a scene graph (Wernecke, 1994) managed by the overall Visualizer application. Element components share lower-level implementations, such as triangulated surfaces optimized for high-performance rendering, and provide an interface for algorithms to create those in a streaming fashion. This additional separation of algorithms and their resulting data allows developers to optimize them separately. For example, changing the initial implementation of isosurfaces from unordered triangle sets to indexed triangle sets increased rendering performance substantially, and only required changing a single type definition inside the Isosurface component, and no changes in the Isosurface Extractor component. Once the superior performance of indexed triangle sets was established, we changed the Isosurface Extractor to generate indexed triangles internally, increasing extraction speed by a large factor. We expect that implementing even higher-performance surface representations will be not much more difficult. While such features are of no immediate interest to users, they allow developers to more quickly and easily experiment with new algorithms or data structures, both to improve performance or to create new analysis tools. It is this flexibility that has enabled us to rapidly expand Visualizer for new scientific applications such as the ones described in Section 5.

4.4 Visualization Modules

A visualization module ties together all components required to visualize a particular data format, and a file reader to load concrete data sets from external storage. The actual code of a visualization module is usually very short. It only contains "glue code," i. e., type definitions to describe the internal structure of the data and the required components, and the code to read grid structures and data values from an input file and store them in the data representation.

The module concept is the incorporation of an approach to data handling that differs from many other visualization applications. Many applications, including Tecplot and Vis5D, define a "native" data format, and users have to convert their data to this data format in a pre-processing step. Although these conversions are generally simple, having to keep several versions of the same data in several formats wastes storage space, and conversions can lose precision, especially since most interchange formats are plain ASCII tables. More importantly, conversion means that it becomes impossible or inefficient to directly stream intermediate results from a running simulation code into a visualization application to monitor the process of an ongoing simulation, and potentially even manipulate simulation state on-the-fly (Kreylos et al., 2002). Our ultimate goal for Visualizer is to have it used in such a context; hence we decided not to enforce native data formats for each basic grid structure, but to give programmers the ability to make the simulation's data format native to Visualizer by coding a visualization module. From a user's point of view, Visualizer does not have one native data format, but as many as there are visualization modules. Our approach is also different from providing file reader plug-ins, which convert

data file formats into an internal format. Instead, all data representation, visualization algorithm, and visualization element components related to a particular data format are compiled specifically for the data format, giving the compiler the option to optimize across component boundaries. We believe that our approach yields higher extraction and rendering performance.

4.5 Overall System Architecture

The Visualizer application itself is responsible for managing all loaded visualization module plug-ins, all loaded data sets, the graphical user interface that lets users select variables and algorithms, and the scene graph of created visualization elements. Another important component is the *DataLocator* module responsible for translating input device interactions to extracting visualization elements, as explained in Section 3, and for displaying intermediate extraction results for real-time feedback. Additionally, Visualizer contains modules to measure the position of and data value at arbitrary locations, to edit the color maps applied to all data variables individually, and to add interactive clipping planes (i.e., planes that remove all 3D content behind a 3D plane defined by the user's input, to reveal the internal structure of visualization elements, or previously occluded elements) to the visualization.

4.6 Tying It All Together

Judging by the descriptions provided in this section, it might seem that Visualizer is more of a programming toolkit for visualization software than an actual application aimed at end users. And this impression is partially true; our experience shows that VR visualization requires fine-tuned algorithms and data structures that sometimes depend on the particular data set to be visualized, and sometimes a particular scientific question requires custom analysis tools that need to be implemented at a low level to perform in real time. Visualizer's component toolkit enables programmers to quickly add such custom algorithms, and experiment until their optimal implementation is found. Finally, reading a particular data set requires writing a data file reader, and the "glue" code that binds all required components into a visualization module.

However, from a user's point of view, Visualizer is a complete application for visual exploration. If users happen to use a data format that is already supported by a visualization module, they can read it directly without the need for any scripting or programming; if there is no module, they can initially convert their data to a supported format in the same way they previously did for other visualization software, or find a programmer to write a custom module for their data. In our experience, this has not been a barrier for a wide range of datasets.

5 Evaluation

5.1 Visualizer Performance for Interactive Virtual Exploration

As Visualizer is designed for immersive VR environments, it is important to ask whether its implementation satisfies the VR real-time constraints. To evaluate this, we measured its performance on the data set described in case 1 and used in the user study (see below and Figure 5), on a desktop PC with a 2.4 GHz AMD Athlon 64 X2 CPU, 1 GB of RAM, and an Nvidia Geforce 7800GS graphics card. The test data set is defined on a curvilinear hexahedral grid of $271 \times 81 \times 201$ vertices, with two variables (temperature and viscosity) given for each vertex. The data set is stored in the native ASCII format written by the simulation code, and occupies 353.5 MB of disk space.

On loading the data set, Visualizer needed 12.4 s to parse the input file, convert all vertex positions from spherical to Cartesian coordinates, calculate the decadic logarithm of the viscosity values (viscosities are best visualized logarithmically), and create a kd-tree, a part of the interal representation of curvilinear grids containing all cell centers (Preparata and Shamos, 1993), that is later used to quickly find the cell containing a given position. Most of this time is spent parsing the ASCII input file (creating the kd-tree of 4.4 million vertices takes about 3 s); storing input data as binary files reduces load times substantially. Afterwards, we measured how long it takes to extract the isosurface shown in Figure 5, b), using smooth shading with vertex normal vectors computed as data value gradients. Creating a seeded isosurface from the center point of the feature shown in Figure 5, c), took 304 ms and generated 339,722 triangles. For comparison, creating a global isosurface of the same isovalue took 744 ms and generated 339,854 triangles (the isosurface has a small disconnected component not extracted by the seeded algorithm). After extraction, Visualizer was able to render either isosurface at a frame rate of 146.8 frames/second, or 49.9 million triangles per second. During dragging, the extraction algorithm was able to create about 100,000 triangles before it had to stop and display the intermediate result due to the 0.1 s time-out; in other words, it was able to visualize a large region of the isosurface around the point of interest in real time.

Performance is similar in immersive environments. Startup in our CAVE takes a few seconds longer since Visualizer itself and the input data set have to be replicated to all six cluster nodes, but isosurface extraction times are about the same. The rendering performance in the CAVE is about a factor of two lower, since all surfaces have to be rendered twice in each frame (once for the left eye and once for the right eye).

5.2 Scientific Applications Using Visualizer: Three Case Studies

One of the most difficult questions to address regarding 3D visualization is whether analyzing observational or experimental data or computer simulation results in a 3D immersive environment actually leads to scientific understanding or results that would not have been found using more traditional methods on a desktop computer. While it is not possible to provide a definitive answer to this question for all scientific data sets, we present three examples of how using Visualizer in both immersive VR

and desktop environments has impacted our own research and led to scientific understanding that was missed or not possible using other software or methods.

5.2.1 Assessing 3D Simulation Input for Subduction Dynamics Models

The interaction between tectonic plates at the Earth's surface and convection in the mantle often manifests in deformation such as mountain building or rifting over long periods of time (e.g., millions of years). On these time scales the deformation of tectonic plates and the mantle can be modeled as viscous flow to gain insight into how coupling between plates, the viscosity and density structure of the mantle, and crustal structure contributes to observed deformation (Billen and Gurnis, 2001; Billen et al., 2003). Numerical simulations of deformation related to plate tectonics often incorporate the geometry of structures such as plate boundaries, crustal terranes or subducted lithosphere (slabs) for a particular study region. While 2D cross sections of regional geometries can be relatively simple, i.e., gently sloping faults, and smoothly varying slab dip, in 3D space these same structures can vary rapidly in directions that are not orthogonal to the model mesh, making them challenging to represent on meshes of limited resolution required for simulations. In particular, the success of iterative solution methods used in finite element models are strongly dependent on the smoothness of the viscosity and temperature field defining the tectonic plates and plate boundaries. In addition, the model simulations take anywhere from three days to two weeks to run on a 64-processor beowulf cluster. Therefore, the ability to troubleshoot problems with the input model before running the simulation can save an enormous amount of time and money.

Coauthors Billen and Jadamec are using Visualizer to assess the appropriateness of input models for finite element simulations of crust and lithosphere deformation in southern Alaska, and to analyze the results of these simulations. Figure 5a–d shows the viscosity structure for a 3D model of the subducting Pacific plate beneath southern Alaska. The realistic 3D geometry of the subducted plate and plate boundary weak zone incorporated in this model is based on the seismicity within the subducting plate (Ratchkovski and Hansen, 2002; Page et al., 1989; Gudmundsson and Sambridge, 1998) and is unusual for tectonic simulations, which usually use simplified geometries. The viscosity structure depends on both the constructed thermal structure of the subducting Pacific plate and overriding North American plate, and the specification of the geometry, width, and weakening within the weak zone of the plate boundary. Before Visualizer was available, the thermal structure and weak zone structure were constructed and examined with multiple mesh-orthogonal slices using MATLAB. Because the large size of the data set (20 million mesh points), it was not possible to view isosurfaces of the individual input structures, or to overlay the structures to examine their alignment. In addition, due to the large amount of time required to generate and view all the slices, only a subset of mesh slices were viewed to assess the smoothness and appropriate superposition of the two input structures. However, from the examination of the individual data sets, it was concluded that the structures were correctly positioned relative to one-another and were appropriately smooth for input to the numerical simulation.

Following failure of the numerical simulation to converge to a solution, the output viscosity structure was viewed in a CAVE environment using the Visualizer software. Isosurfaces created for multiple isovalues obtained by dragging the wand through the data set immediately revealed grid-aliasing problems (stair-stepped isosurfaces from the thermal structure) and unintended holes in and protrusions

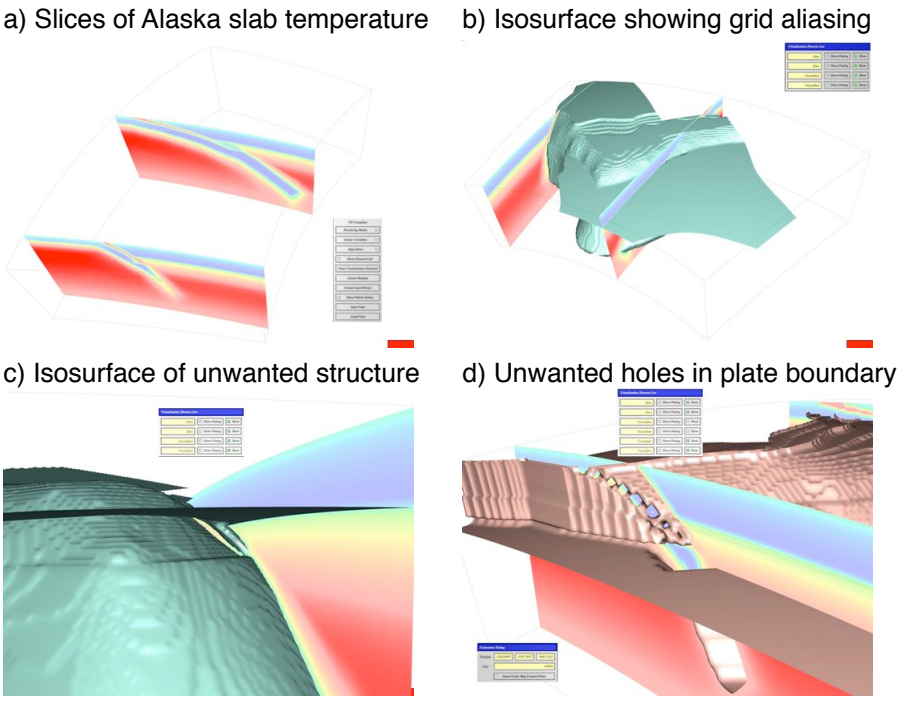

Fig. 5. Case study 1: Snapshots from input data set for finite element model of subduction (also used in user study). a) Color-mapped slices showing cross section of a subducted slab is used to locate initial problem features in a finite element model. b) Isosurface providing a 3D view of the subducting tectonic plate structure with stair-step aliasing on grid. c) Close-up view of a "problem feature" protruding from slab surface. d) Close-up view of a second problem feature, holes in the isosurface of the slab edge. See also Movies 1 and 2.

from the tectonic plate (Figure 5b–d). Dragging slices through the data set also illustrated the holes and protrusions in these regions. These unintended features were located in regions where the radius of curvature of the subducting plate was small and so changed rapidly with respect to the underlying mesh. These features indicated that the failure of the model simulation to converge was a result of the challenges in constructing the realistic 3D plate boundary zone geometry, rather than a limitation of the iterative solver or finite element scheme. Based on analysis of the location and type of unintended structures, a new method of smoothly defining the thermal structure and weak zone geometry was implemented. Visualizer was used to quality-check the input data set before running the simulation. The ability to drag isosurfaces that could be viewed in real time from multiple perspectives and to drag slices through the data set for any orientation provided an efficient method of visually inspecting the entire model input data set and make improvements before running the model. The newly constructed model input was then used for the simulation, which converged on a solution for the flow and deformation associated with the subducting Pacific plate beneath Alaska (Figure 1c, d).

In this application, the interactivity built into Visualizer is the key advantage over other available software. The ability to recognize the features of interest were similar on a desktop computer and in the CAVE. However, both environments were used for different tasks. The immersive VR environment was used when discussion of the causes of features and decisions on how to modify the input model were made by more than one person. The desktop version was used to quickly check the result of incremental

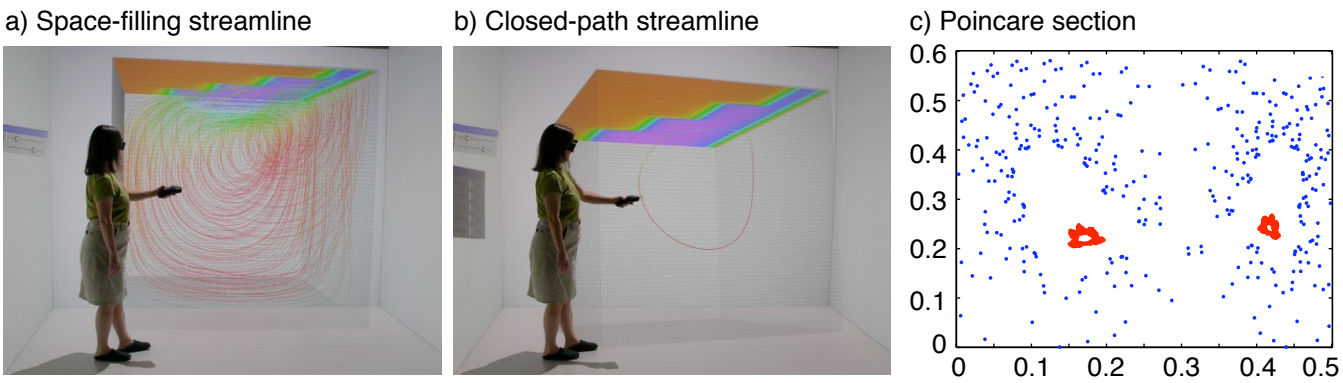

Fig. 6. Case study 2: Using the streamline visualization tool to investigate a 3D steady velocity field. (a) A particle path that appears space-filling. The plane shows the horizontal velocity near the top edge of the calculation; the spreading center and transform faults are visible. (b) Use of the streamline tool to explore the field reveals a contained region. This particle path has the same number of timesteps as the path in (a). (c) The Poincaré section shows crossing points for particle paths in the two regions shown in (a) and (b). The small area covered by the contained particle path (red) makes it difficult to find this feature without the use of Visualizer.

changes made to the procedure and methods used to make the input model. The flexibility of being able to use Visualizer in either environment, while preserving the interactivity developed for use in an immersive environment in the desktop environment, proved to be another key advantage over other software.

5.2.2 Locating Closed Particle Paths in Chaotic 3D Flow Beneath Spreading Ridges

Convection in the mantle is responsible for mixing material that is recycled at subduction zones; to interpret the different geochemical signatures of mantle-derived basalts one must understand the extent and timescales of mantle mixing (Kellogg and Turcotte, 1990; Kellogg, 1992). Mixing in 2D has been studied extensively, for example (Gurnis, 1986; Christensen, 1989; Kellogg, 1992), but studies of mixing in 3D models of mantle flow have been limited by the few available methods for visualizing and analyzing the flow in general and the resulting mixing in particular. Mixing in 2D incompressible flows may be quite different from 3D mixing. For example, in steady-state flows in 2D space, the flow is characterized by closed particle paths (Ottino, 1989); efficient mixing requires time-varying flow (McKenzie, 1979; Olson et al., 1984) and chaotic particle paths can only develop in time-dependent flows (Christensen, 1989; Kellogg and Stewart, 1991; Kellogg, 1992). In contrast, in steady-state 3D flows with both a toroidal and poloidal component, chaotic particle paths have been found (Ferrachat and Ricard, 1998; van Keken and Zhong, 1999). Moreover, turbulent mixing can coexist with regions of laminar flow in which mixing is inefficient (Ferrachat and Ricard, 1998).

One method for finding nonchaotic regions embedded in a chaotic flow is based on constructing Poincaré sections, in which the crossing points of a particle path are plotted on a plane bisecting the flow. The nonchaotic paths will form a regular pattern of points restricted to a limited region of the plane. This method requires a certain amount of serendipity in that the Poincaré plane must cross the non-chaotic paths, and the particle paths selected must include the nonchaotic paths. In exploring a simplified

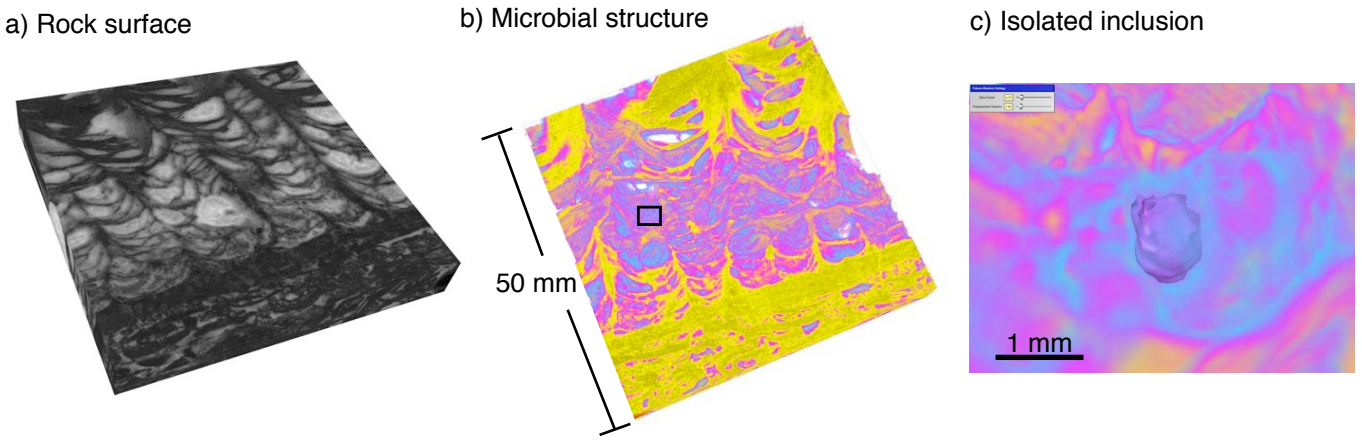

Fig. 7. Structure of fossil mats and biofilms in rocks. a) Reconstructed 3D volume of scanned thin sections of a rock. b) Color image of microbial structure in rock. c) Isolated inclusion of microbial material. See also Movie 3.

model of mantle flow beneath a spreading center with transform faults, co-author Kellogg found no non-chaotic particle paths using the Poincaré method. However, after a few minutes of exploration of the velocity data field using a streamline tool that interactively updates particle path stream lines emanating from the wand, closed loop paths were discovered (Figure 6a,b). Starting at the point specified by the user using the wand, Visualizer numerically integrates the velocity field to obtain a particle path. (The user specifies the time-step used and number of steps in the integration.) These paths occupy a small volume of space and are not single closed loop paths, which makes them difficult to locate using traditional methods. Once the appropriate particle paths were identified, we were able to use Poincaré sections to compare the structure to results from previous work (Figure 6c).

In this application, using Visualizer in an immersive environment provides both the interactivity needed to thoroughly explore the 3D data set and the 3D viewing experience that allows the multiple streamline paths to be viewed without the clutter of overlapping paths seen in the figure.

5.2.3 Exploration of Intricate Fossil Microbialites

The morphology of some microbial communities provides information about the organization and behavior of the microbes forming the mats and biofilms. These structures are sometimes preserved in ancient rocks and therefore are a record of early life on Earth. To use morphology to constrain the early evolution of life, the structures in ancient rocks much be quantitatively compared to those created by specific microbial processes and a precise understanding of the 3D geometry of intricate structures is needed. This requires the reconstruction of the morphology of ancient structures and the measurement of relevant features. Co-author Sumner is using Visualizer to characterize Archean fenestrate microbialites (Sumner, 1997, 2000). The reconstruction of microbialite samples is done by successively polishing 20 μm off the surface of a sample and scanning each new surface. Scanned images are aligned and assembled to define a reconstructed 3D volume (Figure 7a, Movie 3). Visualization consists of rendering the volume and using a number of tools to manipulate views of the data. For this application, the structure of the ancient microbial community is most effectively visualized by

rendering mineral components (white to light gray) fully transparent and microbial components (black to dark gray) in color with variably transparency (Figure 7b). Visualization of the volume in a non-stereo system provides significantly improved understanding of the structures over observing successive opaque images. Visualization in a 3D immersive environment provides even more substantial increases in understanding over non-stereo volume visualization, in addition to enabling precise measurement of features.

One significant advantage of stereo volume visualization is the ability to investigate data and see it in 3D space while it is not moving. With non-stereo 3D visualization, one must continually manipulate the volume and use parallax to comprehend the full geometry of structures. This motion obscures details in the data; the user must remember the spatial relationships when viewing fine details in a static image, which is a significant challenge. For example, it is very difficult to evaluate whether a small microbial feature within a predominantly mineral-only areas is attached to a larger sheet of microbial structures or is fully isolated. In the absence of stereo rendering, one must move the image significantly to obtain parallax 3D vision and, if the feature is fairly isolated, detailed spatial relationships are only observable through slicing the data and removing geometrical information. Then the user must remember these relationships and observe the data at a high zoom to look for very small features such as isolated clumps of organic inclusions (Figure 7c, Movie 3). Rotations of the data must be found where multiple clumps occlude each other to obtain 3D spatial information. The need for parallax significantly limits the user's ability to understand the geometry. In contrast, in an immersive stereo environment, the user only needs to observe the volume at a high zoom, seeing the relationships among minute structures at any orientation and simultaneously seeing their relationships to larger structures with peripheral vision. The increased spatial understanding is obtained almost immediately and with much higher accuracy than is possible in a non-stereo system.

The insights from viewing the data can be extended to quantitative characterization of geometry within Visualizer. For our isolated clump example, the separation of structures could be due to either the primary growth geometry of the microbial community or the decay of organic matter after burial. Measurements such as the spacing between patches of preserved organic inclusions and the orientation of preserved patches are two very important constraints which cannot be obtained from 2D data and are inaccurate and extremely difficult to make without stereo rendering of the 3D data. In Visualizer, these measurements are easily made in the volume, both along and oblique to the original scanned images (Movie 3). With the precise locating of the pointer implemented in Visualizer, measurements can be made with an accuracy of the full resolution of the data set more quickly than the user can decide where to measure. The limits to obtaining quantitative information shift from the time spent manipulating data to obtain the measurements desired to the intellectual challenge of deciding which measurements best characterize the data.

5.3 Assessing The User Experience

Two of the major drawbacks of some software used in immersive environments are, first, that the software is difficult to learn and use, and second, that most software tools only work well on the desktop or in an immersive environment, but not in both (Table 1). Throughout the development of

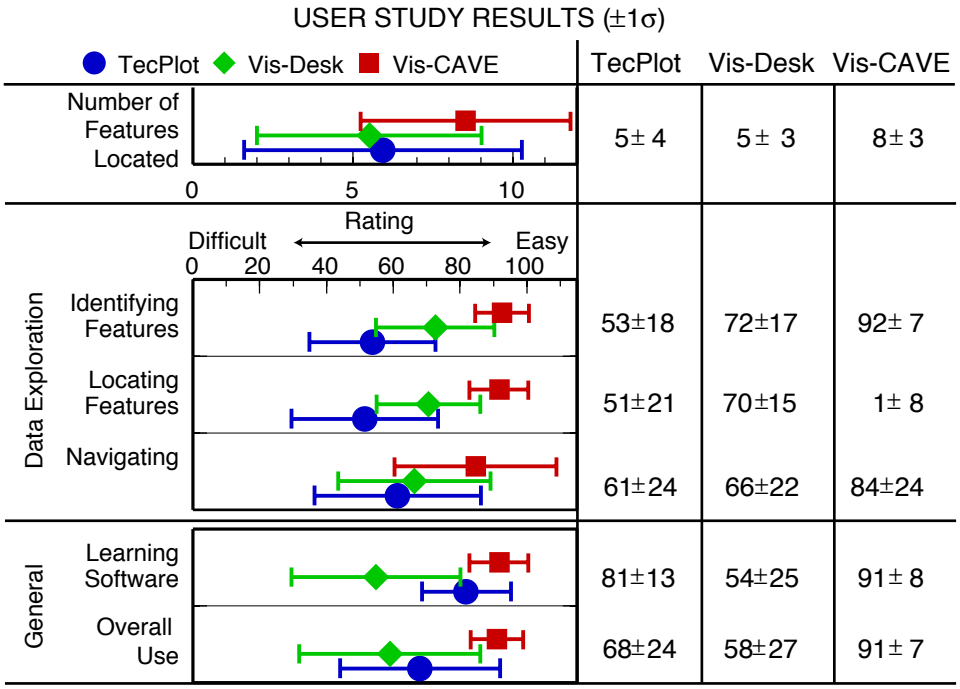

Fig. 8. User study results. Rating of data exploration (navigating, identifying and locating features) and general use (learning software and overall usability) displayed as the mean and standard deviation. Results show that Visualizer used in the CAVE made data exploration very easy and the software was easy to learn and use overall. Visualizer used on a desktop was also better for data exploration than Tecplot, but was more difficult to learn since the tool assignment process is somewhat cumbersome on the desktop.

Visualizer the goal has been to create a flexible, easy-to-use and intuitive tool for interactive, scientific analysis of 3D volume data sets that can be used in multiple environments. However, the verification of whether the software achieves this goal must be done by the scientists using the software. To this end, we have conducted a user study aimed at assessing how well Visualizer aids researchers in evaluating unknown data sets, by comparing it to Tecplot (TecPlot, 2006), described in Section 2, a software package commonly used by geoscientists. Tecplot was chosen as a comparison because it has a menu/button-driven interface that is similar to many other visualization programs. In the first part of the study both visualization systems were compared in their native environments (Visualizer in a CAVE and Tecplot on a desktop computer). In the second part of the study, Visualizer was also used on a desktop computer and the results were compared to the those from the CAVE; Tecplot does not run in VR environments.

5.3.1 User Study Format

In each software/environment setting, the study participants were asked to explore a data set and to identify features that did not meet specified criteria: 1) features must be smooth and continuous, without steps or faceted surfaces, and 2) features must be continuous, without holes or protruding structures. The data used in all parts of the study is the model input of a subducting tectonic plate beneath southern Alaska described in the case study on 3D numerical simulation (Figure 5). The tasks

performed by the study participants were based on the same steps used by geoscientists in analyzing the appropriateness of a model input data set for a FE model calculation.

Before beginning the prescribed tasks, the study participants were taught how to use the software in each environment using an idealized input data structure. Afterwards, the participants were then given 30 minutes to explore the target data set and instructed to write down the coordinates and data value for features they found that did not meet the specified criteria. After completing the analysis, the participants were asked about the tools within the software that they found most and least useful for navigating through the data set and identifying features. They were also asked to note any tasks that they found to be particularly easy or difficult to complete. Finally, after completing the tasks in all three software/environment settings, the participants rated the software on a scale of easy to difficult for the tasks of navigating, identifying features, locating features, learning the software, and overall use.

The study included 19 participants – eleven graduate students, five undergraduate students, and three faculty – engaged in geoscience study and/or research. Of the 19 participants, two had minimal previous experience using a CAVE and one had previously used Tecplot. Although none of the participants were experts on the scientific objectives related to this particular data set, all the participants were familiar with the general geologic problem.

5.3.2 User Study Results

The results of the user study, shown in Figure 8, demonstrate that the high level of interactivity provided by Visualizer in the CAVE makes analysis of 3D volume data both easy and effective. Participant responses indicate that not only is data exploration (navigating, identifying and locating features) easier to do in the CAVE than using Tecplot, Visualizer is also easier to learn how to use and easier to use overall. In addition, on average more features were located using Visualizer in the CAVE than using Tecplot or Visualizer on the desktop, although this part of the results is not conclusive and demands further investigation. The participants also found that data exploration is easier on the desktop using Visualizer than using Tecplot, but they also found it more difficult to learn how to use, which probably affected both their overall impression of the software and ability to locate features.

Participant feed-back on the individual software provided additional information on why Visualizer is easy to learn and use in the CAVE and more difficult to learn on the desktop. First, nine participants stated that they did not find any of the tasks to be difficult in the CAVE and found it particularly easy to identify and locate features. As one user reported, "the Visualizer-CAVE was by far the easiest to work with. All the features stood out very clearly and it was incredibly easy to navigate." In contrast, 13 participants stated that identifying and locating features in Tecplot was the most difficult task. We interpret this response to reflect both advantages of viewing the data set in a fully immersive environment and the intuitive design of the software.

Second, 13 users stated that assigning tools in Visualizer on the desktop was the most difficult task. In any environment, tool assignment in Visualizer requires first choosing what one would like the program to do (e. g., create isosurfaces) from the main menu and then assigning this action to to an input device button, e.g., a mouse button, by selecting a locator tool from the tool selection menu using the desired

button. On the desktop, however, two things make this process more cumbersome. First, buttons on the mouse are used to navigate (e. g., left button for rotating and right button for translating the data) and they are also used to create color-mapped slices or isosurfaces by using a modifier key on the keyboard (this caused confusion for new users who had to remember the button-key assignments). Second, the tool selection process on the desktop includes an extra step of creating a "virtual input device" to map 2D mouse positions into 3D model coordinates. User responses indicate that remembering the button-key combinations made tool assignment and use more difficult; however, they also stated that, once this was overcome, Visualizer was better for identifying and locating features. As one user stated, "the greatest difficulty is probably the initial complexity of click and key combination, but even in 30 minutes I became pretty efficient with what I learned and I can see getting used to it very quickly." Based on the responses and suggestions of the study participants, we plan to add an optional, fixed "tool bar" for tool assignment that can be used to introduce new users to Visualizer, but can be turned off once button-key combinations, which are faster to use, are learned.

The user study ratings and responses show that the interactivity provided by using Visualizer in a CAVE is intuitive and allows users to focus their attention on exploring the data set. In addition, they indicate that while interacting with a 3D data set in a 2D environment can be made effective in an interactive visualization system, it requires practice, while an immersive 3D VR environment provides immediate benefits by allowing the user to interact with data in a natural and intuitive manner. These results, combined with our experiences illustrated in the case studies above, suggest that immersive environments will become popular within scientific applications if software design meets the goals we addressed in Visualizer.

6 Conclusions and Future Work

We have presented new software for visualization of 3D gridded data sets, which has been designed for highly interactive, immersive 3D VR environments, but also achieves cross-system portability making it a powerful tool for visualization in the Geosciences. As an application aimed at immersive VR environments, Visualizer satisfies two real-time constraints: 1) it maintains a high frame rate upwards of 30 Hz to create a realistic immersive experience, and 2) it provides real-time feedback (display of visualization elements) to any user interaction within about 1/10 s to enable interactivity. Interaction is further enhanced by following a direct manipulation approach across the entire range of functionality, from navigation to creation of visualization elements to quantitative analysis. At the software development level these constraints are met by using multithreaded programming, special algorithms such as seeded isosurfaces, advanced rendering using compressed geometry or multiresolution methods, and careful optimization of extraction and rendering algorithms. Meeting the real-time constraints for VR environments translates into an efficient and highly-interactive desktop application for very large data sets, which are often too large for real-time interaction using other software. Visualizer achieves this cross-system portability by being built on the Vrui development toolkit, being completely independent of the underlying VR environment and optimally matching interaction patterns to the capabilities of the underlying environment. While many of the optimization methods and visualization techniques used in Visualizer have been available in other desktop visualization software, our contribution lies in integrating these pieces into a software package that is designed for data exploration and analysis and

is portable to a wide range of VR and desktop environments.

The three case studies outlining how Visualizer has been used for Geoscience applications provide direct evidence of how Visualizer can enable discoveries or insight related to complex 3D gridded data sets. These case studies illustrate 1) how the interactivity speeds up analysis of the data leading to discovery of new behaviors (case study 2), 2) how the portability allows users to choose which environment best suits a particular task of data exploration and evaluation (case study 2), and 3) how viewing data with complex 3D morphology in a VR environment allows the user to recognize previously unseen features (case study 3). These examples also demonstrate how immersive stereo environments facilitate more detailed analysis of complex structures by eliminating the need to constantly move data to create parallax motion, and by providing zoomed-in views of data, while maintaining the perspective of larger spatial relationships with peripheral vision. Finally, the user study also provided valuable feedback on how further to improve Visualizer. We learned that effective exploration requires "transparent" user interfaces, such as intuitive navigation and "point and click" creation of visualization elements, and we will continue developing Visualizer, and the Vrui toolkit itself, to provide a more efficient, and easier to learn, user interface especially on the desktop.

Acknowledgements

This research has been support in part by a grant from the W. M. Keck Foundation. We thank Barbara Romanowicz for the shear wave seismic tomography model, N.S. Conjeepuram for spreading ridge model results and Poincaré section, and Patrick Senge for serial sectioning the microbialite sample. We thank John Tipper and Eric de Kemp for their thoughtful reviews.

References

Alexandrescu, A., 2001. Modern C++ Design: Generic Programming and Design Patterns Applied. Addison-Wesley Proffesional.

Bawden, G., 2006. Imaging postseismic transient nearfield deformation following the 2004 parkfield earthquake in central california with ground-based lidar. Eos Transactions AGU Fall Meeting Supplement Abstracts 87 (52), S23C–0172.

Bernardin, T., Cowgill, E., Gold, R., Hamann, B., Kreylos, O., Schmitt, A., 2006. Interactive Mapping on 3D Terrain Models. Geochemistry, Geophysics and Geosystems in press.

Billen, M. I., Gurnis, M., 2001. A low viscosity wedge in subduction zones. Earth and Planet. Sci. Lett. 193, 227–236.

Billen, M. I., Gurnis, M., Simons, M., 2003. Multiscale dynamic models of the Tonga-Kermadec subduction zone. Geophys. J. Int. 153, 359–388.

Bowman, D. A., Kruijff, E., LaViola, J. J., Poupyrev, I., 2004. 3D User Interfaces: Theory and Practice. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA.

Bryson, S., 1996. Virtual reality in scientific visualization. Communications of the ACM 39 (5), 62–71.

Bryson, S., Levit, C., 1991. The Virtual Windtunnel: An environment for the exploration of three-dimensional unsteady flows. In: Proc. of Visualization '91. IEEE Computer Society Press, Los Alamitos, CA, pp. 17–24.

Cabral, B., Cam, N., Foran, J., 1994. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In: VVS '94: Proceedings of the 1994 symposium on Volume visualization. ACM Press, New York, NY, USA, pp. 91–98.

CAVE5D, 2006. visualization software. http://www-unix.mcs.anl.gov/~mickelso/CAVE2.0.html.

Christensen, U., 1989. Mixing by time-dependent convection. Earth and Planetary Science Letters 95, 382–394.

Cruz-Neira, C., Sandin, D., DeFanti, T., 1993. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In: Proc. of SIGGRAPH '93. ACM Press, Anaheim, CA, pp. 135–142.

Engel, K., Westermann, R., Ertl, T., 1999. Isosurface extraction techniques for web-based volume visualization. In: VIS '99: Proceedings of the conference on Visualization '99. IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 139–146.

Ferrachat, S., Ricard, Y., 1998. Regular vs chaotic mixing. Earth and Planetary Science Letters 155, 75–86.

Fledermaus, 2007. IVS 3D Fledermaus Software.
URL `http://www.ivs3d.com/products/fledermaus/`

Furumura, T., Chen, L., 2004. Large scale parallel simulation and visualization of 3d seismic wavefield using the earth simulator. Computer Modeling and Engineering Sciences 6, 153–165.

Gao, D., 2003. Volume texture extraction for 3D seismic visualization and interpretation. Geophysics 68 (4), 1294–1302.

GCC Compiler, 2007. The GNU GCC compiler online manual.
URL `http://gcc.gnu.org/`

Gudmundsson, O., Sambridge, M., 1998. A regionalized upper mantle (RUM) seismic model. J. of Geophys. Res. 103 (B4), 7121–7136.

Gurnis, M., 1986. Stirring and mixing in the mantle by plate-scale flow: large persistent blobs and long tendrils coexist. Geophysical Research Letters 13, 1474–1477.

Hansen, C. D., Johnson, C. R., 2004. The Visualization Handbook. Adacemic Press.

Harris, J. M., 2004. Binocular vision: moving closer to reality. Phil. Trans. R. Soc. Lond. A. 362, 2721–2739.

Hubona, G. S., Shirah, G. W., 2005. Spatial Cues in 3D Visualization. In: Cai, Y. (Ed.), Ambient Intelligence for Scientific Discovery, LNAI 3345. Springer-Verlag, Heidelberg, pp. 104–128.

Hughes, T. J. R., 2000. The Finite Element Method: Linear Static and Dynamics Finite Element Analysis. Dover, Mineola, New York.

Jadamec, M. A., Billen, M. I., 2006. Building a geodynamic model of Alaska. Geological Society of America, Cordilleran Section, 102nd annual meeting 38 (5496), 97.

Johnson, A., Leigh, J., Morin, P., Keken, P. V., 2006. GeoWall: stereoscopic visualization for geoscience research and education. IEEE Computer Graphics and Applications, 10–14.

Kalvin, A. D., Dean, D., Hublin, J.-J., Braun, M., 1992. Visualization in anthropology: reconstruction of human fossils from multiple pieces. In: Proceedings of the IEEE Conference on Visualization.

Kellogg, L. H., 1992. Mixing in the mantle. Annual Review of Earth and Planetary Sciences 20, 365–388.

Kellogg, L. H., Stewart, C. A., 1991. Mixing by chaotic convection in an infinite prandtl number fluid and implications for mantle convection. Physics of Fluids A 3, 1374–1378.

Kellogg, L. H., Turcotte, D. L., 1990. Mixing and the distribution of heterogeneities in a chaotically convecting mantle. Journal of Geophysical Research 95, 421–432.

Kreylos, O., 2006. Environment-independent VR development. Tech. rep., U. C. Davis, KeckCaves.

Kreylos, O., Bawden, G. W., Bernardin, T., Billen, M. I., Cowgill, E. S., Gold, R. D., Hamann, B., Jadamec, M., Kellogg, L. H., Staadt, O. G., Sumner, D. Y., 2006a. Enabling scientific workflows in virtual reality. In: Wong, K. H., Baciu, G., Bao, H. (Eds.), Proceedings of ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications 2006 (VRCIA 2006). ACM Press, New York, pp. 155–162.

Kreylos, O., Bernardin, T., Billen, M. I., Cowgill, E. S., Gold, R. D., Hamann, B., Jadamec, M., Kellogg, L., Staadt, O. G., Sumner, D. Y., 2006b. Enabling scientic worksflows in virtual reality. In: Procòf the ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications (VRCIA) 2006. ACM SIGGRAPH, ACM SIGGRAPH.

Kreylos, O., Bethel, E. W., Ligocki, T. J., Hamann, B., 2001. Virtual-reality based interactive exploration of multiresolution data. In: Farin, G., Hagen, H., Hamann, B. (Eds.), Hierarchical Approximation and Geometrical Methods for Scientific Visualization. Springer-Verlag, Heidelberg, Germany, pp. 205–224.

Kreylos, O., Tesdall, A. M., Hamann, B., Hunter, J. K., Joy, K. I., 2002. Interactive visualization and steering of CFD simulations. In: Ebert, D., Brunet, P., Navazo, I. (Eds.), Data Visualization 2002 (Proceedings of VisSym 02). Association for Computing Machinery, New York, NY, pp. 25–34.

Lewis, B., Berg, D. J., 1998. Multithreaded Programming with Pthreads. Sun Microsystems Press.

Lin, C.-R., Loftin, R. B., 1998. Application of virtual reality in the interpretation of geoscience data. In: VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology. ACM Press, New York, NY, pp. 187–194.

Lorensen, W. E., Cline, H. E., 1987. Marching Cubes: A high resolution 3D surface construction algorithm. In: Proc. of SIGGRAPH '87. ACM, pp. 163–169.

Ma, C., Rokne, J., 2004. 3D seismic volume visualization. In: Integrated image and graphics technologies. Kluwer Academic Publishers, Norwell, MA, USA, pp. 241–262.

McKenzie, D., 1979. Finite deformation during fluid flow. Geophysical Journal of the Royal Astronomy Society 58, 689–715.

McNamara, A. K., Zhong, S., 2005. Thermochemical structures beneath Africa and the Pacific Ocean. Nature 437, 1136–1139.

Mégnin, C., Romanowicz, B., 2000. The shear velocity of the mantle from the inversion og body, surface and higher mode waveforms. Geophysical Journal International 143, 709–728.

Meyer, T., Globus, A., 1993. Direct manipulation of isosurfaces and cutting planes in virtual environments. Tech. Rep. CS–93–54, Brown University, Providence, RI.

Motani, R., Amenta, N., Wiley, D. F., 2005. Possibilities and limitations of three dimensional retrodeformation of a trilobite and plesiosaur vertebrae. PaleoBios, 88.

Ohno, N., Kageyama, A., 2007. Scientific visualization of geophysical simulation data by the cave vr system with volume rendering. Physics of the Earth and Planetary Interiors 163, 305–311.

Olson, P., Yuen, D. A., Balsiger, D., 1984. Mixing of passive heterogeneities by mantle convection. Journal of Geophysical Research 89, 425–436.

Ottino, J. M., 1989. The kinematics of mixing: stretching, chaos, and transport. Cambridge University Press.

Page, R. A., Stephens, C. D., Lahr, J. C., 1989. Seismicity of the Wrangell and Aleutian Wadati-Benioff zones and the North American plate along the Trans-Alaska Crustal Transect, Chugach Mountains and Copper River basin, southern Alaska. J. of Geophys. Res. 94, 16059–16082.

Preparata, F. P., Shamos, M. I., 1993. Computational Geometry: an Introduction, 5th Edition. Mono-

graphs in Computer Science. Springer Verlag.
Ratchkovski, N. A., Hansen, R. A., 2002. New evidence for segmentation of the Alaska Subduction Zone. Bulletin of the Seismological Society of America 92, 1754–1765.
Reed, D. M., Yagel, R., Law, A., Shin, P.-W., Shareef, N., 1996. Hardware assisted volume rendering of unstructured grids by incremental slicing. In: VVS '96: Proceedings of the 1996 symposium on Volume visualization. IEEE Press, Piscataway, NJ, USA, pp. 55–ff.
Rhyne, T. M., MacEachren, A., 2004. Visualizing geospatial data. In: SIGGRAPH '04: Proceedings of the conference on SIGGRAPH 2004 course notes. ACM Press, New York, NY, p. 31.
Shen, H.-W., Johnson, C. R., 1995. Sweeping simplices: A fast iso-surface extraction algorithm for unstructured grids. In: VIS '95: Proceedings of the 6th conference on Visualization '95. IEEE Computer Society, Washington, DC, USA, p. 143.
Sherman, W. R., Craig, A. B., 2002. Understanding Virtual Reality: Interface, Application, and Design. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
Shneiderman, B., 1983. Direct manipulation: a step beyond programming languages. IEEE Computer 16 (8), 57–69.
Stroustrup, B., 1997. The C++ Programming Language, 3rd Edition. Addison-Wesley Professional.
Sumner, D. Y., 1997. Late archean calcite-microbe interactions: Two morphologically distinct microbial communities that affected calcite nucleation differently. Palaios 12, 300–316.
Sumner, D. Y., 2000. Microbial versus environmental influences on the morphology of late archean fenestrate microbialites in microbial sediments. In: Riding, R., Awramik, S. (Eds.), Microbial Sediments. Springer, Berlin, pp. 307–314.
Sutton, M. D., Briggs, D. E. G., Siveter, D. J., Siveter, D. J., 2001. Methodologies for the visualization and reconstruction of three-dimensional fossils from the silurian herefordshire lagerstätte. Palaeontologia Electronica 4 (1).
Tackley, P. J., June 2000. Mantle convection and plate tectonics: Toward an integrated physical and chemical theory. Science 288, 2002–2007.
TecPlot, 2006. visualization software. http://www.tecplot.com.
Tipper, J. C., 1991. A prototype general-purpose dynamic visualization system. GeoByte 6 (3), 11–15.
Tufte, E., 1983. The Visual Display of Quantitative Information. Graphics Press.
van Dam, A., Forsberg, A. S., Laidlaw, D. H., Jr, J. J. L., Simpson, R. M., 2000. Immersive VR for scientific visualization: A progress report. IEEE Computer Graphics and Applications 20.
van Keken, P., Zhong, S., 1999. Mixing in a 3d spherical model of present-day mantle convection. Earth and Planet. Sci. Lett. 171, 533–547.
Vis5D, 2006. visualization software. http://www.ssec.wisc.edu/~billh/vis5d.html.
Wang, S., Zhang, S., Yuen, D. A., 2007. Visualization of downwelling in 3-d spherical mantle convection. Physics of the Earth and Planetary Interiors 163, 299–304.
Wernecke, J., 1994. The Inventor Mentor: Programming Object-Oriented Graphics with Open Inventor, Release 2. Addison-Wesley Professional.
Zollikofer, C., de Leon, M. P., 2005. Virtual Reconstruction: A Primer in COmputer-Assisted Paleontoogy and Biomedicine. Wiley.