Two-handed Interactive Stereoscopic Visualization

David S. Ebert
U. of Maryland Baltimore County [*]

Christopher D. Shaw
U. of Regina [†]

Amen Zwa
U. of Maryland Baltimore County [‡]

Cindy Starr
NASA GSFC [§]

Abstract

This paper describes a minimally immersive interactive system for visualization of multivariate volumetric data. The system, SFA, uses glyph-based volume rendering which does not suffer the initial costs of isosurface rendering or voxel-based volume rendering, while offering the capability of viewing the entire volume. Glyph rendering also allows the simultaneous display of multiple data values per volume location. Two-handed interaction using three-space magnetic trackers and stereoscopic viewing are combined to produce a minimally immersive volumetric visualization system that enhances the user's three-dimensional perception of the data. We describe the usefulness of this system for visualizing volumetric scalar and vector data. SFA allows the three-dimensional volumetric visualization, manipulation, navigation, and analysis of multivariate, time-varying volumetric data, increasing the quantity and clarity of the information conveyed from the visualization system.

1 INTRODUCTION

This paper describes a new system for the interactive display of volumetric data using two-handed interaction and stereoscopic viewing. The system provides a minimally immersive interactive visualization tool to increase the understanding of volumetric data while being affordable on desktop workstations. Glyph-based rendering [8] and hardware acceleration techniques are used to provide interactive rendering speeds for scalar and vector volumetric data. Minimally immersive interaction is achieved by combining three-space trackers in a two-handed interaction metaphor [10, 17] with stereoscopic display.

Glyph, or iconic, visualization is an attempt to encode more information in a comprehensible format and allows multiple values to be encoded in the parameters of the icons [15]. The shape, color, transparency, orientation, etc., of the glyph can be used to visualize data values. Glyph rendering [15, 16] is an extension to the use of glyphs and icons in numerous fields, including cartography, logic, semiotics, and pictorial information systems.

Several researchers have examined the use of virtual reality environments for visualization. Bryson and Levit built a flow visualization system called the Virtual Windtunnel [2], in which a DataGlove was used to specify the location of streamline and streakline sources. A BOOM head-tracked display was used to view the resulting streamline and streakline animations.

Work has continued for many years at UNC-Chapel Hill on the topics of exploratory molecular visualization [1] and real-time exploration of atomic scale phenomena [19], using a force-feedback mechanical arm as the main input device.

Hinckley et al. [10] designed a two-handed 3D neurosurgical visualization system where the user sits in front of a high-resolution monitor and manipulates the scene using two 3D magnetic trackers. The left hand holds the brain model represented by a tracker mounted inside a doll's head, and the right hand manipulates a cutting plane represented by a flat plate connected to a second tracker. The user moves the cutting plane to interactively generate cutaway views of the brain. New users immediately understand how the manipulation works and need essentially no training to use the system.

There have been several previous approaches to the interactive display of volumetric data. Many research systems [2, 3, 9, 13] and several commercial systems, including IBM DX, AVS, and FAST, allow interactive viewing of data using traditional 2D interfaces. Several systems also allow stereoscopic visualization [7, 14] while still utilizing two-dimensional interaction techniques.

Most previous techniques for interactive visualization of volume data use 2D interfaces or immersive head mounted displays. Our approach to interactive visualization combines glyph-based volume rendering with a new minimally-immersive interaction metaphor to provide interactive visualization, manipulation, and exploration of multivariate, volumetric data. Careful data value mapping to glyph rendering attributes allows better comprehension of the multivariate data than can be achieved using isosurfaces or direct volume rendering. The rendering techniques used in our system are described first, followed by the two-handed metaphor for interaction. The results of applying these techniques to the visualization of space physics data are presented in the following section. Finally, we describe future directions for research in this area.

2 RENDERING WITHIN SFA

The Stereoscopic Field Analyzer (SFA) allows the visualization of both regular and irregular grids of volumetric data. Because of the need to resample the space, standard voxel-based techniques for visualizing scalar fields are not appropriate when the sample points occur on an irregular grid as in many fields, such as astrophysics and computational fluid dynamics. Certain assumptions must be made about the underlying physical properties of the space being rendered that may not be true for the particular data set in question, potentially resulting in a meaningless or misleading visualization.

In addition to rendering both regular and irregular grids, SFA provides the ability to render more than one set of volumetric data using the same grid. This is achieved by the use of glyph-based rendering [15, 16], a valuable tool for the volumetric display of multiple parameters. It has many of the advantages of volume rendering, while avoiding the limitations and difficulties of iso-surface rendering. An additional feature of SFA is the ability to interactively visualize and manipulate time-varying multi-variate volumetric data.

Currently, there are five types of glyphs available for scalar fields, and four types for vector fields. The scalar glyph types are single-pixel points, 4-pixel points, boxes, spheres, and random point clouds. The vector types are lines, cones, arrows, and cylin-

[*]Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 5401 Wilkens Ave., Baltimore, MD 21228, Phone: (410)455-3541, email: ebert@cs.umbc.edu

[†]Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada S4S 0A2, Phone: (306) 585-4632, email: cdshaw@cs.uregina.ca

[‡]Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 5401 Wilkens Ave., Baltimore, MD 21228, Phone: (410)455-3541, email: zwa@cs.umbc.edu

[§]Scientific Visualization Studio, NASA Goddard Space Flight Center, Greenbelt, MD, Phone: (301)286-4583, email: cindy@caspian.gsfc.nasa.gov

ders. All glyphs are also given a color based on either a different visualization data value or their position within the scalar range of all of the data points. That is, the data range is quantized into 256 steps, and a unique color is assigned to each step. When a glyph is drawn, its color data value is converted into a table index, and the corresponding color in this table is used.

2.1 User Control

Within this basic framework, the user may easily control the visualization in numerous ways, including selection of the glyph type and data set mapped to glyph color. To avoid the clutter induced by displaying insignificant data, the user may select the data ranges to be displayed, which may be a collection of disjoint intervals.

2.1.1 Color Control

The color range may be selected by choosing among a collection of standard pseudocolor tables. Three types of color tables are generally used, depending on the properties of the data file and the number of data files being displayed. For grids with a single data file that has only positive or only negative scalar values, any reasonable pseudocolor table will suffice. However, if both positive and negative values are to be displayed, the scalar value 0.0 is mapped to color 128 and the maximum magnitude is mapped to color 255, producing a symmetric color table about the 128th entry. The third consideration for color tables is the use of colors for displaying multiple data files at once. SFA allows for a different colormap for each data file, enabling the user to tell which glyph at each location corresponds to which data file. Users would typically choose a distinguishable hue for each data file, with the color ramping from low to high saturation.

2.1.2 Subsets

As a further aid in reducing visual clutter, the user may select maximum and minimum grid extents to display in each dimension. The user may choose to concentrate on a particular subset of the volume, culling the unwanted outlying glyphs. Similarly, the user may choose to display only every n^{th} grid point along each dimension. These two controls serve to reduce the number of glyphs drawn, thereby reducing the amount of time taken to draw the scene.

2.2 Rendering Style

To create a good 3D impression, SFA renders the scene using either opaque or translucent glyphs, and performing standard Z-buffer hidden-surface removal. At the user's option, the scene may be drawn in stereo using a pair of Liquid Crystal Shutter Glasses. We use the standard parallel camera model to render the 3D scene, which is identical in all respects except that it must be rendered twice, once for each eye. We have found that the stereopsis is extremely helpful in generating a 3D impression in most users.

3 INTERFACE

SFA was first developed using a standard 2D interface style, where interactions are performed using the keyboard and mouse. To control the view of the volume, a virtual trackball [5] or rotation sliders can be used, rotating the wireframe box in realtime. In addition, there are a number of control panels that contain one or more interaction techniques to control the display of the volume (see Figure 1).

The *Attribute* panel controls the type of glyph to use for display, allowing the user to cycle through the available glyph types for either scalar or vector fields. This panel also allows the user to select among the various data files to display. For the current data file, a special double-ended slider called the *twinslider* allows the user to select both the minimum and the maximum scalar value to display.

The *Subset* panel contains three simple up-down counter controls to adjust the value of n for every n^{th} grid point to display along the X, Y and Z dimensions. In addition, three twinsliders are used to control the minimum and maximum displayed extents along each dimension.

The *Color* panel allows the user to view the current color map. The user may also toggle the display of all glyphs that map to a particular color entry by adjusting that entry's *alpha* (transparency) value, allowing the user to select which values to display and their opacities.

3.1 Discussion

The original 2D SFA interface presented a number of problems to users. First, it was not conceived with real-time rendering in mind, so the interaction style was similar to a compile-edit loop: users adjusted the various controls, triggered a redraw, and examined the static picture that resulted.

Second, non-realtime rendering forced the user to interact with the dataset without seeing it dynamically update, imposing another cognitive burden. In particular, selecting a volume subset could be somewhat difficult because the user could not see the resulting volume until it was redrawn. Third, all of these eminently 3D spatial operations used 1D interaction techniques, with a corresponding increase in the user's cognitive load.

With its 2D interface, SFA embodies a common conception of the scientific visualization task as being the specification of a static scene followed by the examination of that scene. The difficulty with this approach is that people do not examine real objects in this way unless there is some compelling reason to do so, such as a requirement to leave the object undisturbed, or if the object is too big or too small to examine directly. Therefore, requiring a careful statement of viewing parameters forces the user to expend unnecessary cognitive effort. We have, therefore, developed a direct manipulation approach, where scene specification and examination occur as one fluid process, allowing the user to concentrate on understanding the object at hand.

4 TWO-HANDED MINIMAL IMMERSION

Our examination of these difficulties led us to extend SFA with the introduction of a pair of 3D magnetic trackers, transforming it into a minimally-immersive desktop visualization system. One of the main justifications for this approach is that complex 3D objects can be more easily manipulated by 3D tracking devices than with a mouse [20], because the user does not have to mentally break down the 3D task into a sequence of 2D operations. The use of a 3D device allows the user to directly manipulate the objects of interest without intermediate steps. Two 3D devices give the user access to double the spatial bandwidth, since both hands can be employed in parallel to quickly achieve the desired operation.

The user sits in front of a graphics console that has a screen, keyboard, mouse, and the two 3D sensors (see Figure 2). Each 3D sensor, or *Bat*, has three buttons glued onto the surface, as shown in Figure 3. The user interacts with the system by manipulating the two trackers, and pressing the tracker buttons to invoke operations. Each 3D sensor has a distinct role, with the dominant hand being responsible for picking and manipulation, and the less-dominant hand being responsible for context setting of various kinds. For the sake of rhetorical convenience, we will refer to the dominant hand as the right hand and the less-dominant hand as the left, but the system is ambidextrous because the Polhemus trackers are symmetric and can be handled with equal ease by either hand.

We chose the minimally immersive style because it allows 2D functionality to remain if 3D trackers are not available at the user's workstation. Thus, SFA has a certain amount of interface redundancy to allow for different hardware configurations. We chose not to adopt the immersive style, in which a Head Mounted Display (HMD) is used to display the scene to the user. The nominal benefit of this style is that the HMD gives a large panoramic field of view, but it comes at a cost of cutting the user off from traditional I/O devices like the screen, keyboard, and mouse.

We use two sensors because their simultaneous use takes advantage of people's innate proprioceptive knowledge of where their two hands are in space. Guiard [11] gives psychophysical evidence for the idea that the left and right hands quite often act as elements in a kinematic chain. For right-handed people, the left hand acts as the base link of the chain. The right hand's motions are based on this link, and the right hand finds its spatial references in the results of motion of the left hand. Also, the right and left hands are involved in asymmetric temporal-spatial scales of motion (right hand for high frequency, left hand for low frequency). [1] The Two-Handed interface to SFA uses this natural division of manual labour by assigning the (low-frequency) setting of spatial context to the left hand, and the (high-frequency) selection and picking operations to the right.

This interface is similar to one designed by Shaw for free-form surface modeling [17], and some of the interaction techniques are derived from it. Hinckley et al's system [10] offers a similar assignment of hand roles, but there is intentionally only a very limited set of interaction techniques. Other work at the University of Virginia [18] has further explored the idea of two-handed interfaces using an object, or *prop*, attached to each sensor.

4.1 Left Hand Operations

The left hand has two tasks to perform:

- Manipulate the position and orientation of the entire scene.
- Select the drawing context from a 3D tracker-based hierarchical menu.

Scene orientation is a toggle controlled by left bat button 3. Pressing button 3 attaches the volume to the left cursor, and pressing it again leaves the volume in place. This clutching mechanism allows the user to spend a large amount of time moving the workpiece around without the burden of continuous button pressure.

Bat button 2 pops up a hierarchical Sundial menu [17], as shown in Figure 4, which is a menu controlled by the orientation of the left bat. The menu choices are arrayed on the circular plate of the Sundial, each on its own pie-shaped sector. The desired item is picked by pivoting the *shadow stick* about its base so that the stick's endpoint lies visually in front of the sector. The base of the shadow stick is located at the center of the plate and, when the menu first pops up, the stick is aligned with the line of sight, pointing directly at the user. Around the base of the stick is an inner circle which can be used to indicate *no selection*. The position of the Sundial is fixed rigidly to the left bat, and the dial lies parallel to the projection plane.

This is a 3D version of the pie menu [4] which encodes the position of the 2D cursor by projecting the 3D position of the endpoint of the shadow stick onto the 2D Sundial plate. To implement this, we trace a ray from the eyepoint through the endpoint of the stick and intersect it with the Sundial plate. The realignment of the shadow stick along the line of sight at popup time is needed to orient the stick to a neutral location. Realignment also has the advantage of allowing the user to have the bat at any orientation when the menu pops up.

[1] For left-handed people, the roles of right and left are reversed.

This main menu in SFA contains a submenu to select the glyph type, a submenu for each of X, Y, and Z step interval selection, and cursor control menu that scales the scene up or down and reorigins the bats to the center of the workspace. The interval step and glyph type submenus are each radio buttons which allow only one of the parameter values to be selected at a time.

The Sundial menu is provided to allow ease in selection while both hands are occupied manipulating the bats. If only mouse-based menus were provided, then the user would have to continually pick up and put down the right bat in order to interact with the scene. Also, while stereo mode is active, the mouse-based menu system must account for stereo drawing, or present the menus in only one eye's view. Because the Sundial menu is a 3D object, incorporating it in the stereo scene requires only the trivial step that it be drawn twice.

4.2 Right Hand Operations

The right hand has two tasks to perform:

- Select the 3D volume subset.
- Pick a glyph to print out its value.

To select a volume subset, the user presses right button 2 to place one corner of the volume to be displayed, drags the right bat to the opposite corner, then releases right button 2, automatically culling the glyphs outside of this box. The user may restate the volume at any time by sweeping out a new subset box. To select a glyph, the user orients a probe into the volume, and the closest glyph has its value printed and optionally passed through a socket connection to software to allow closer examination of this data point. A probe, represented by a narrow cylindrical shaft, is attached to the right cursor, and the user controls the position and orientation of the probe with the right bat. The distance from this probe is computed for each control point using a specialized distance metric called the *probe metric*, and the grid point generating the smallest probe metric value is the one picked.

Because the user is selecting among many objects, the probe axis is drawn to some arbitrary length, and a translucent cone which has a radial spread angle of 10 degrees is drawn. The probe is the cone axis, and a spotlight is attached to the cursor that has identical visual geometry to the translucent cone. As the user sweeps the probe about the scene, the objects that fall within the cone intersect visually with the cone wall and are highlighted by the spotlight. This is purely a visual cue to help the user locate the cone in 3D visual space.

The non-Euclidean probe metric is designed to closely mimic the shape and behavior of the probe axis. Given a point, the smaller the angle between the probe axis and a line from the cone apex to the point, the smaller the metric. If two points have equal angular distances from the probe axis, the point that is closest in Euclidean distance to the cone apex yields the smaller metric. To compute the probe metric from a cursor located at point C, with the probe axis directed along the normalized vector R, we need to generate V, the vector from the cursor to grid point P:

$$V = P - C$$

and

$$ProbeMetric = |V|(1 + \frac{180 \cos^{-1}(R \cdot V)}{\pi}) = |V|(1 + Angle)$$

Multiplying the angle by $|V|$ scales the angular metric by the Euclidean metric. Adding 1 ensures that very small angles are properly scaled.

The probe metric also allows the selection of objects that are small and/or far away. Simply tracing a mathematical ray along the probe axis and picking the first intersecting object is not effective, because a high degree of cursor accuracy is required to pick small objects.

5 REALTIME RENDERING

On current, low-end SGI workstations, SFA can render volumes of less than 20,000 grid points at realtime rates (10 frames per second) if points are used. However, with larger volumes or more complex glyphs, rendering falls below the realtime threshold and 3D interaction becomes impossible.

One solution is simply to acquire a graphics machine that can draw the desired datasets quickly enough, but this merely changes the maximum number of glyphs renderable in realtime, and can be costly. A more general solution to this problem is to reduce the amount of computation that must be performed.

5.1 Data Access Optimization

A simple analysis of the drawing loop of SFA shows that the algorithm is $O(n)$, where n is the number of glyphs. There is also a fixed amount of time spent per frame clearing the frame buffer and, on machines with rudimentary graphics hardware, this can take 30 or more milliseconds. Given the linear nature of the drawing algorithm, the only opportunities for speedup are to reduce n or the amount of time spent drawing each glyph. The remainder of this section outlines the approaches we found valuable to improve the drawing performance.

Although SFA is written in C, it was initially constructed in an object-oriented manner. For example, adding a new glyph was simply a matter of writing the glyph drawing routine and adding the new routine's function pointer to a table. The color lookup function was similarly general and none of the results were cached in any way. Drawing a glyph, therefore, required four function calls. Replacing the procedure calls with inline code reduced data access from 5.5 μs per grid point on a 150 MHz Indigo to 3.2 μs per grid point, or a factor of 1.74 speedup[2].

A further optimization that trades memory for speed is to cache the drawing parameters for each glyph instead of recalculating them every update. For scalars, data to be cached is the (x, y, z) location, color and magnitude. Storing all this information together increases the locality of reference for a single glyph. Also, the computations required for quantizing the color data into a color table index need only be done once and saved. Using this cache cuts the total drawing cost to 1.0 μs per grid point, or a factor of 5.5 speedup over the object-oriented approach. This cuts the total access time to 50.7 ms per update for 50,000 grid points, and is about the best one can possibly expect, since the absolute minimum of information is being accessed. The SGI GL drawing cache (object) was used for each glyph, but not for the drawing cache, because GL objects cannot be quickly edited. Moreover, the Performer manual [12] notes, and our tests confirmed, that the prime source of speedup with GL objects is quad-word alignment.

5.2 Drawing Optimization

To achieve fluid interaction, the system must update the scene at least at 10 frames per second and, given the above analysis, the only way to do this is to reduce the number of glyphs that are drawn. One could draw only the wireframe box while the user is moving the volume, but this forces the user to re-assimilate the scene once the volume stops moving. A better approach is to approximate the volume while the user is moving it and to draw the full volume when the user stops moving.

[2] Both times are for code compiled and linked with the *-O2* option.

Our method draws the largest glyphs first and keeps drawing smaller and smaller glyphs until the 100 ms time limit is reached. Thus, when the volume is in motion, the most visible objects within the scene are still drawn, and the smaller objects disappear. The user still has an extremely clear idea of all the “landmarks” in the volume because they are still visible while the volume is in motion.

This was implemented with a bucket sort using magnitude as the sort key, and inserting each data point into the drawing cache at a location determined by the sorting algorithm. To avoid bucket chaining, a 256 entry histogram of magnitudes is collected for each data file as it is read in.

Once the cache is filled, the drawing process passes through the cache in order from largest to smallest. When the user is moving the volume, the drawing process continues until 100 ms have passed. This allows the user to see the largest glyphs being updated in realtime, as can be seen in Figure 5. When the user stops moving, the entire volume is drawn. If a volume subset has been selected, the cache is refilled with the selected subset of data, and the cumulative and offset tables are used by the drawing loop to select the correct glyphs. This optimization allows volumes with an arbitrary number of data points to be drawn.

5.3 Glyph Approximation

Since box and sphere glyphs typically take 5 to 20 times as long to draw as single-pixel points, they are approximated with points for small scalar values. We determine when to start drawing points by calculating the pixel area covered by a complex glyph when it is within 10 centimeters of the near clipping plane. If the complex glyph would occupy less than 4 pixels, this and all smaller sizes are drawn as points. The effect of this optimization strongly depends on the data set, but speedups of a factor of 10 have been observed in real datasets.

6 RESULTS AND CONCLUSIONS

SFA performs realtime, interactive visualization of scalar and vector time-varying data. On an SGI 150 MHz Indigo XS24, SFA can display about 20,000 1-pixel points at 10 frames per second without dropping small glyphs from the drawing cache. If boxes are used, 97% of the glyphs must be dropped in order to keep the same frame rate. Typically, the user will move the volume around until a desired view is reached, then will stop moving and wait for the volume to fill in. On an SGI 250MHz Indigo2 High Impact, SFA can display about 70,000 points at 10 frames per second. For boxes, about 80% of the glyphs must be dropped.

In terms of rendering, the glyph approach does not suffer the initial costs of isosurface rendering or voxel-based volume rendering and, at the same time, offers the capability of viewing the entire space. Thus, SFA allows the user to immediately examine and explore the data.

SFA has been used to visualize several sets of scientific data, including gas plasma vortex data and other magnetofluid simulations. Figure 6 shows a visualization of a magnetohydrodynamics simulation of the solar wind in the distant heliosphere (20 times the distance of the Earth to the Sun). The simulation data is a $33 \times 33 \times 33$ grid containing the vector vorticity for the simulation.

SFA's users (scientists at NASA) have commented on the ease of use of the two-handed system, and have been enthusiastic about its use for exploring complex fluid dynamics simulations. One of the main benefits of SFA is the ability to quickly understand the contents of the 3D space. The two handed interface is especially useful in giving a proprioceptive sense of the 3D space, which is impossible in other approaches. In combination with stereoscopic display, a powerful 3D impression can be given to the user. SFA, therefore, allows volumetric visualization, manipulation, navigation, and analysis of multivariate, time-varying volumetric data, increasing the quantity and clarity of the information conveyed from the visualization system.

7 FUTURE EXTENSIONS

Several extensions can be made to the current system, including further optimization of the rendering algorithms to increase the performance of the system, the design of more complex glyphs that compactly represent many data items per grid point, and the use of texture memory to provide annotations to glyphs and help distinguish glyph types. We are also exploring the application of these techniques to visualizing large information spaces [6].

8 ACKNOWLEDGMENTS

We wish to thank the Dr. D. Aaron Roberts for his support of this research, David Brookshire for assistance with generating the images, and Susan Wrights for assistance in preparation of this manuscript. This project was supported by the National Science Foundation under grant CCR-9409243 and by NASA under grant NAG 5-2893.

References

[1] Lawrence D. Bergman, Jane S. Richardson, David C. Richardson, and Frederick P. Brooks, Jr. VIEW – an exploratory molecular visualization system with user-definable interaction sequences. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 117–126, August 1993.

[2] Steve Bryson and Creon Levit. The virtual wind tunnel. *IEEE Computer Graphics and Applications*, 12(4):25–34, July 1992.

[3] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In Arie Kaufman and Wolfgang Krueger, editors, *1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, October 1994. ISBN 0-89791-741-3.

[4] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems*, pages 95–100. ACM SIGCHI, Washington, DC, 1988.

[5] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 121–129, August 1988.

[6] David S. Ebert, Christopher D. Shaw, Amen Zwa, and Ethan L. Miller. Minimally-immersive interactive volumetric information visualization. In *Proceedings Information Visualization '96*, October 1996.

[7] Elliot K. Fishman, Brian S. Kuszyk, David G. Heath, Luomin Gao, and Brian Cabral. Surgical planning for liver resection. *IEEE Computer*, 29:64–72, 1995.

[8] J. D. Foley and C. F. McMath. Dynamic process visualization. *IEEE Computer Graphics and Applications*, 6(3):16–25, March 1986.

[9] Allen Van Gelder and Jane Wilhelms. Interactive animated visualization of flow fields. *1992 Workshop on Volume Visualization*, pages 47–54, 1992.

[10] John C. Goble, Ken Hinckley, Randy Pausch, John W. Snell, and Neal F. Kassel. Two-Handed Spatial Interface Tools for Neurosurgical Planning. *Computer*, 28(7):20–26, 1995.

[11] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *The Journal of Motor Behavior*, 19(4):486–517, 1987.

[12] Jed Hartman and Patricia Creek. *IRIS Performer Programming Guide*. Silicon Graphics, Inc., 1994.

[13] Nelson Max, Roger Crawfis, and Barry Becker. New techniques in 3D scalar and vector field visualization. In *First Pacific Conference on Computer Graphics and Applications*. Korean Information Science Society, Korean Computer Graphics Society, August 1993.

[14] T. V. Papathomas, J. A. Schiavone, and B. Julesz. Applications of computer graphics to the visualization of meteorological data. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 327–334, August 1988.

[15] Frank J. Post, Theo van Walsum, Frits H. Post, and Deborah Silver. Iconic techniques for feature visualization. In *Proceedings Visualization '95*, pages 288–295, October 1995.

[16] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherja. Glyphmaker: creating customized visualizations of complex data. *IEEE Computer*, 27(7):57–64, July 1994.

[17] Chris Shaw and Mark Green. THRED: A Two-Handed Design System. *Multimedia Systems Journal*, 4:to appear, 1996.

[18] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 265–272, Denver, Colorado, May 7-11 1995.

[19] Russell M. Taylor, II, Warren Robinett, Vernon L. Chi, Frederick P. Brooks, Jr., William V. Wright, R. Stanley Williams, and Eric J. Snyder. The Nanomanipulator: A virtual reality interface for a scanning tunnelling microscope. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 127–134, August 1993.

[20] Colin Ware and Danny R. Jessome. Using the bat: A six-dimensional mouse for object placement. *IEEE Computer Graphics and Applications*, 8(6):65–70, November 1988.

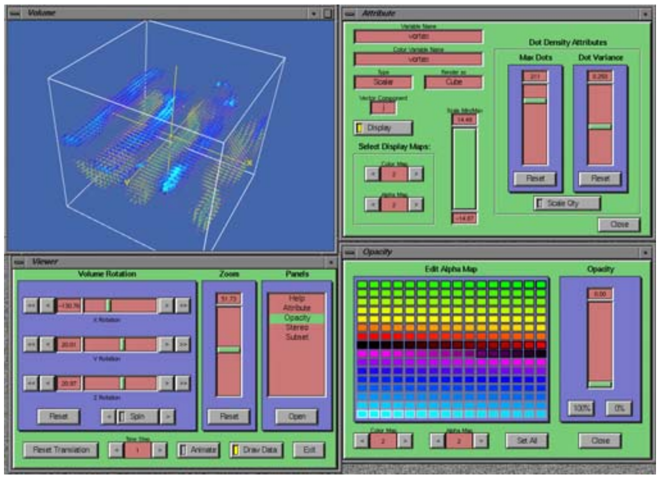

Figure 1: 2D interface and control panels for SFA.

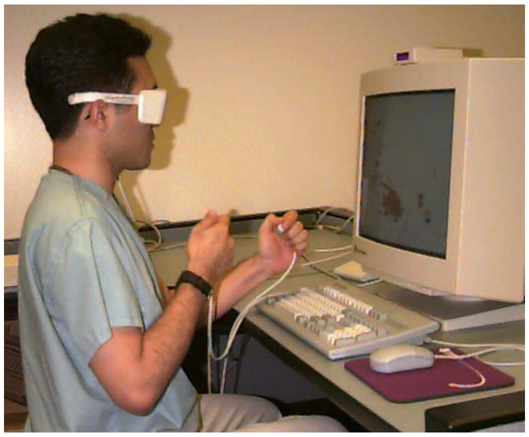

Figure 2: A user using the two−handed stereo interface to SFA.

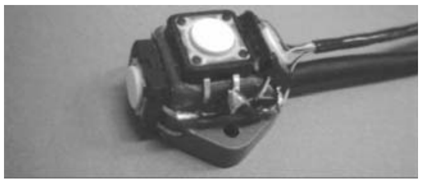

Figure 3: Polhemus sensor with attached buttons.

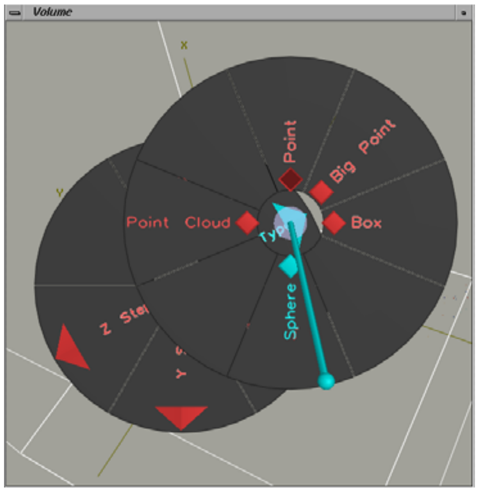

Figure 4: 3D hierarchical Sundial menu.

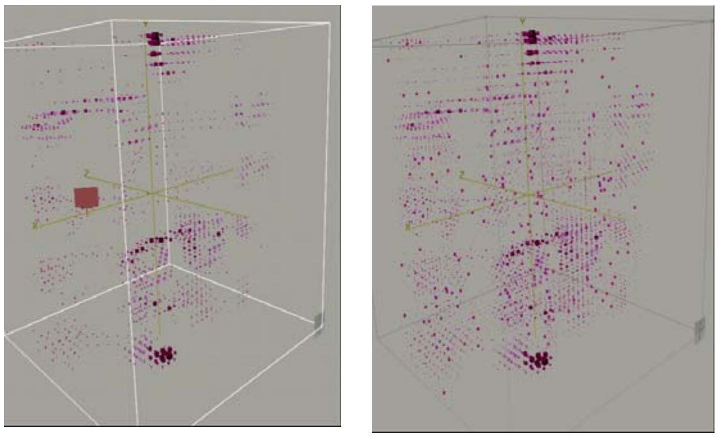

Figure 5 : (a) Data caching during quick interaction with the magnetohydrodynamics simulation. (b) Rendering of the full data.

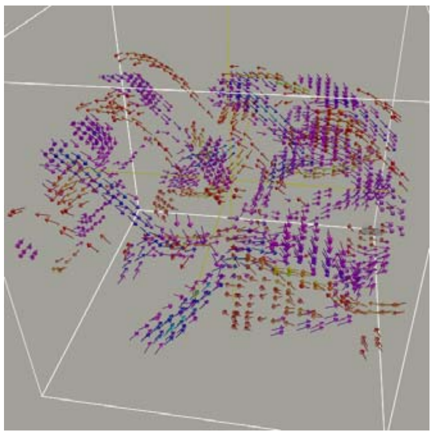

Figure 6 : Visualization of a magnetohydro−dynamics simulation of the solar wind in the distant heliosphere.