Cutting Planes and Beyond

Michael Clifton and Alex Pang
Computer Information Sciences Board
University of California, Santa Cruz, CA 95064

1 Abstract

We present extensions to the traditional cutting plane that become practical with the availability of virtual reality devices. These extensions take advantage of the intuitive ease of use associated with the cutting metaphor. Using their hands as the cutting tool, users interact directly with the data to generate arbitrarily oriented planar surfaces, linear surfaces, and curved surfaces. We find that this type of interaction is particularly suited for exploratory scientific visualization where features need to be quickly identified and precise positioning is not required. We also document our investigation on other intuitive metaphors for use with scientific visualization.

Keywords: curves, cutting planes, exploratory visualization, scientific visualization, direct manipulation.

2 Introduction

As a visualization tool, virtual reality has shown promise in giving users a less inhibited method of observing and interacting with data. By placing users in the same environment as the data, a better sense of shape, motion, and spatial relationship can be conveyed. Instead of a static view, users are able to determine what they see by navigating the data environment just as they would in a real world environment. However, in the real world, we are not just passive observers. When examining and learning about a new object, we have the ability to manipulate both the object itself and our surroundings. In some cases, the ability to manipulate objects and surroundings may be more important than the ability to simply move around in a virtual world.

This paper describes a direct manipulation system that allows the user to intuitively slice, dice, and carve the data set under investigation. It is well suited for exploratory "spur of the moment" visualizations where the focus is on the data rather than the visualization system. The ease of use of our system comes from identifying intuitive metaphors that can be conveniently mapped to virtual reality devices. The devices used in this work include an 18 sensor Cyberglove to identify hand postures, two 6D trackers from Ascension for tracking hand position and orientation, and stereo glasses from StereoGraphics to aid navigation. We also considered immersive technologies such as head-mounted displays to provide a natural way of looking around the data set. One of the concerns of using head-mounted displays is the sensitivity of the human users to design limitations such as poor display resolution and lag time in head tracking. The latter is particularly troubling as it may cause simulator sickness. Aside from reducing the display resolution further, one can also attempt to reduce the rendering effort e.g. by reducing scene complexity, etc. With current technology, we feel that the overall sacrifice to image quality necessary to bring lag times in head-mounted displays to satisfactory levels is not acceptable for scientific visualization applications. Therefore, we take the approach that effective visualizations can be achieved without full immersion into the world of the data being examined. Instead, by allowing the user to directly manipulate the data, the user is no longer forced into the role of an observer and can play a more active role.

In order for the direct manipulation interaction to be as natural and intuitive as possible, we identify metaphors and tasks that people use when interacting with and visualizing their data. These metaphors can be identified by listening and observing user interactions during a visualization session. One might hear conversations about painting an isosurface a particular color, highlighting some features in the data, or cutting off some portions of the data to expose internal structures. It is not surprising that these map to common experiences and 3D interactions. This paper focuses on the cutting metaphor and how it is extended to support linear and curve surface cross-sections. The cutting metaphor is mapped to tools that behave like knives or blades in the physical world. Armed with this metaphor, the user already has an idea of how to manipulate these tools when they first start to use this system. When cutting a real world object, the user is exposing some interior properties of that object, most notably represented by the color of the internal structures. Similarly, when cutting with this visualization system, the user exposes the data values along the surface of the cut. To make these values meaningful, they are mapped to colors on the cutting surface.

While the metaphors and techniques that we report here can be extended to other forms of data gridding and organization, we first focus on simple three-dimensional regular grids of scalar values. This category encompasses a huge variety of data including 3D medical data from computer aided tomography scans and magnetic resonance imaging, and 3D physical data sets such as pressure, density, and temperature. These data also have the desirable characteristic of having a clear spatial layout. Note that the cutting tools simply specify a portion of the data that may be of possible interest. While it is mostly used for scalar data, it may also be used for vector data. In such case, the selected vector data can be visualized as hedgehog plots, streamlines, etc, on the cut surface.

Because direct manipulation systems are best suited for "spur of the moment" or exploratory type investigations, the user should consider a two pass approach if precise or repeatable visualizations are necessary – after a period of exploratory visualization to identify regions of interest, the user may then switch to a slower but more precise display. There are two main reasons. First, is the lack of precision and repeatability in a user's hand-eye coordination in specifying a cutting surface. While the direct manipulation cutting tools are easier to use, a numerically specified cut surface will always be more accurate. Second, is the necessity to tradeoff quality for speed in order to maintain reasonable interaction rates during direct manipulation exercises. With an interface meant to be transparent to the user, visualization delays are unacceptable, and the steps taken to eliminate them may lead to blurriness or blockiness in the result.

The rest of the paper is organized into five sections. The related work section gives a brief review of relevant human-computer interaction studies in support of direct manipulation, and other related visualization systems. We then present the system architecture. This is followed by details and results of the different cutting and visualization tools. Successes and shortcomings of the system are presented in the evaluation section. Finally, we point out areas for future work and summarize the results of this work.

3 Related Work

3.1 Cognitive Approaches to Human-Computer Interaction

In order to make visualization systems easy to use, the human-computer interface must be intuitive. For example, when manipulating objects in the real world, humans normally do not think in terms of vectors and plane equations, so the system should adapt to the user, not the other way around. Many researchers have asked how the user thinks and how to adapt systems to both take advantage of thought processes and to improve user understanding in complex tasks.

Sellen, Kurtenbach, and Buxton describe an experiment in which they try to help users avoid mode errors [22]. Mode errors are mistakes that occur when the user tries to give the computer a command when in fact the computer is in a mode that does not accept that command. An example would be trying to type a word in the UNIX text editor *vi* while the program is in navigation mode instead of text entry mode. Their experiments involved providing users with either a visual (screen color) or a kinesthetic (foot pedal position)

feedback to indicate the mode of the editor. Both forms of feedback resulted in users making fewer mode errors. However, the kinesthetic feedback setup resulted in significantly fewer errors than the visual feedback system. The conclusion of the researchers was that the visual feedback, since it was passively generated by the system, was too easily overlooked by the user. The kinesthetic feedback, on the other hand, since it was actively generated by foot pressure from the user, was not so easily overlooked.

The results of this experiment suggest that a direct manipulation system may not only lead to fewer errors by the user, but also a greater understanding of what operations are currently being performed. Because the user's hand is physically active during tool manipulation, the user is more likely to maintain an active, correct notion of what tool he is using, where it is, and what state it is in.

Donald Norman describes artifacts as "tools or devices that enhance human ability and performance" [4]. In the physical world, these artifacts are actual tools such as pulleys to make people stronger or cars to make them faster. Using the pulley as an example, the person is empowered to lift heavier objects. However, from the point of view of that person, the task has been changed to pulling a rope instead of lifting the object.

Another type of artifact Norman discusses is the cognitive artifact. Cognitive artifacts are meant to display or represent information in such a way as to improve human cognitive performance. Scientific visualization techniques in general are cognitive artifacts since they take data and give it a representation that hopefully leads to better understanding than the raw data by itself. In our system, the cutting tools are further artifacts. Instead of presenting the user with a pre-generated visualization to study, the nature of the task has been changed to allow the user to generate the visualization interactively in areas of interest, in many cases leading to a more complete understanding of the data itself.

3.2 Related Visualization Systems

There are numerous visualization systems today. Some of these are application specific, while others are more general. Most of them use traditional keyboard and 2D mouse input, while others use more exotic hardware. Our system borrows some of the best ideas and techniques from these systems and improve upon them. In particular, we find that cutting planes are almost universally available in visualization systems. However, their ease of use and applicability have been limited by the interface imposed upon the users. Included here is a non-exhaustive but representative review of systems that provide cutting plane techniques, extension to curved cutting planes, and the use of virtual reality devices for visualization applications.

General visualization systems such as AVS and Data Explorer, as well as more application domain specific systems such as VIS5D [9] and OVIRT [12] all provide cutting planes with options for pseudo-coloring or contouring the cross-sections. Some of them limit the cutting planes to the three orthogonal directions, while others allow arbitrary planes. These planes are usually specified by entering numeric coordinate values or by adjusting graphical sliders. An alternative way of specifying arbitrary planes is through a 3D spray can widget [15]. Here, the plane is normal to a line originating from the nozzle of the spray can widget. As the spray can is moved and rotated, a new cut plane is generated.

Rhodes et al. [18] and Udupa and Odhner [24] describe visualization systems targeted for visualization of volumetric medical data. Both of these allow specifications of curved cross-sections. In the former, the user specifies points with a trackball along the desired curve from a sagittal (side) cross-sectional view. The points are then fitted with a polynomial curve, and a coronal (front) view through this curve is generated. In the latter case, the user first specifies a two-dimensional curve in a fixed 2D view of the data, and then specifies the depth of this curve, resulting in a curved cut. While these may be the only way to specify curve cross-sections with 2D input devices, the availability of 3D input devices can make this task easier and more natural.

A paper by Ney and Fishman details the features of another visualization system designed for use on medical data [13]. Instead of using cutting planes, this system uses cutting volumes to identify pieces of a larger data set. The user specifies the cut volume by drawing out shapes such as boxes, ellipsoids, and extruded polygons with a mouse. The extracted region is then volume rendered. A similar approach is reported by Cignoni, Montani, and Scopigno [5] with their MagicSpheres. Here, a spherical volume of

interest is specified. Then, one or more filters associated with different visualization techniques are applied to the volume of interest. Another variation is to use the flashlight metaphor [16] to shine on the volume of interest. Different shapes can be associated with the flashlight to produce conical, pyramidal, etc. volumes of interest.

Another system of related interest to this thesis is NASA's Virtual Wind Tunnel, developed by Bryson and Levit [2]. This system is a flow field visualization tool that makes use of some specialized input and output hardware to immerse the user into the system. First of all, the Virtual Wind Tunnel uses an immersive display so that the user gets the impression of being inside the wind tunnel. By simply walking around and turning his head, the user can alter the view of the data set. Of more interest, is how the Virtual Wind Tunnel uses a Data Glove to manipulate tools for generating visualization. The main tool is a *rake* where seed points along it emit particles that are then advected into space thereby showing the features of the flow field. With the Data Glove the user can reach out to grab and manipulate the rake. This ability is similar to our system. However, there is an important distinction. In the Virtual Wind Tunnel, the user moves tools around with his hand. In our system, the hand of the user is the tool itself. Instead of grabbing and rotating or translating, the user's hand is more intimately linked to the tool.

Our work also differs from other work that applies virtual reality interfaces to visualization, notably the use of 3D widgets such as a floating rectangle to represent a cutting plane [11]. Here the users specify the cut by grabbing anchor points (e.g. vertices, edges, plane center) and moving the rectangle around. Our work take this a step further and insist that the users' hand be the tool. That is, different postures and gestures of the users' hand would activate different tools. It is important that these postures correspond to the physical objects or metaphors in use.

4 System Architecture and Components

The system architecture is designed to be extensible to other virtual reality devices other than those used in this work. It is also designed to support other metaphors beyond those reported here. An overview of the system architecture is depicted in figure 1. It is similar to the application independent device interface described in [8]. The important feature is that system components such as the device interfaces, and visualization operations are abstracted from the application. There are four major components in our system. They are the Glove and Tracker libraries, the Posture Recognition component, and the Visualization component.

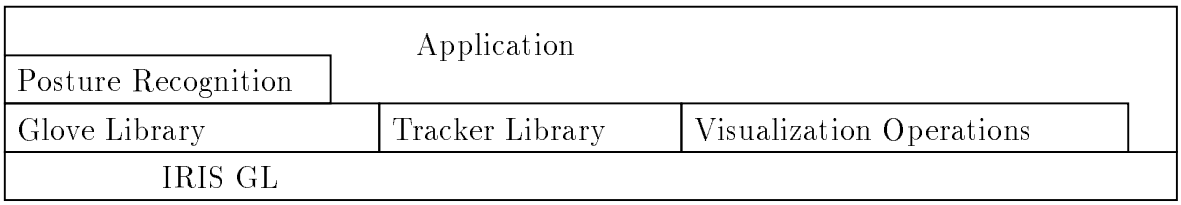

Figure 1: System Architecture

4.1 Tracker Library

The trackers and the glove devices communicate with the host computer through separate serial ports. Byte instructions are defined by the manufacturers and are specific to their own devices. As such, the immediate layer of software is device dependent but shields software layers above it from device specific codes. Other than this immediate software layer, the tracker library provides necessary functionality such as: InitTracker, CloseTracker, and ReadTracker, as well as convenience functions such as: SetTrackerSpeed,

MovePointWithTracker, RotPointWithTracker, and ResetTracker. These functions provide a high level interface for an application program to communicate with the 6D trackers.

4.2 Glove Library

The glove used to measure the user's hand posture is the Cyberglove. It is a thin, spandex glove with flex sensors to measure joint angles. The one we used is a left-handed glove with 18 sensors – two for the first two joints of each finger, four for the spread between fingers, two on the wrist, and two for palm flexing. Although the end joints of the fingers do not have sensors in the glove, the glove library simulates these joints by setting their angles equal to those of the previous joint on each finger. This approximation seem reasonable for relaxed hand postures. The glove also has a small switch attached to the wrist that can be tested for a simple on/off state.

The glove library has a function, InitGlove, similar to the tracker library, that checks for the presence of a Cyberglove unit attached to a serial port. The application may continue without a glove, but then should not depend on any glove functionality.

The main routine provided by the glove library, ReadGlove, reads the joint angles of the glove. This routine accomplishes the conversion of analog sensor signals to actual joint angles by multiplying the values by a scaling factor and adding an offset. To be accurate, specific offset and scale values must be assigned to each of the 18 sensors for each person using the device. This becomes awkward and time-consuming, especially as the sensors might report different values depending on conditions such as whether the unit is warmed up, and if the glove was pulled on tighter than last time, resulting in the need for a lengthy calibration process. Instead, we designed a greatly simplified calibration process into the glove library. This routine, CalibrateGlove, automatically generates the offset and scale values for each sensor. To calibrate the glove, the user places his hand, palm down, flat on the table, with the wrist and fingers straight and not spread, as in figure 2. This represents the configuration in which all joint angles should be zero. While in this position, the user presses a 'C' on the keyboard to invoke the CalibrateGlove routine.

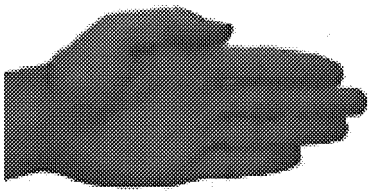

Figure 2: Calibration Posture

As the Cyberglove warms up and the glove stretches and shifts, the calibration typically loses accuracy, but this simple calibration procedure is easily repeated whenever necessary, typically after every one or two minutes of use.

Most applications using the glove will want to display the user's virtual hand on the screen, so a routine, DrawGlove, is provided for drawing a simple hand using IRIS GL graphics routines. The hand model drawn is a relatively simple, boxy representation, but all of the fingers and joint motions are clearly observable. The user generally has a good sense of the joint angles of their fingers, but the display of the hand is useful for identifying the location of the virtual hand and for reassuring the user that the virtual hand indeed corresponds to the his actual hand movements.

Because of their common use, the glove library also provides two other functions: IndexTip and PalmCenter. IndexTip returns the position of the tip of the index finger based on the wrist and index finger sensors. PalmCenter returns the position of the center of the palm based on the wrist sensors. Any point of the hand can be calculated, but these two are widely used to warrant inclusion in the glove library.

4.3 Posture Recognition

The posture recognition system interprets what the hand is doing based on the joint angles returned by the glove library. At first thought, a posture recognition scheme would want to compare all 18 joint angles in the glove to a set of known postures and find a match. However, the joint angles for a given user will never exactly match a predefined posture. The joint angles of a single user will not even be the same for the same posture at different times. As another complication, calibration differences make it impossible to get exact matches. Thus, any posture recognition scheme must account for the fact that the same posture will never appear the same.

One possibility for handling these differences might be a neural network. With proper training, a neural network might successfully learn how to consistently differentiate between postures despite the variations. However, this was not the approach taken for this project. Instead, a simpler mechanism using thresholds was employed with the idea of possibly moving to a neural network or some other sophisticated approach later on. However, the simple approach proved to work very well.

The feature vector to uniquely identify a particular posture does not require all 18 glove sensors. For example, the wrist sensors does not play a role in most postures and can be safely ignored. In addition to the wrists, the thumb poses some ambiguities when matching postures. For example, consider the two pointing postures illustrated in figure 3. The thumb position is different. If asked to point, two different people might use these different postures.

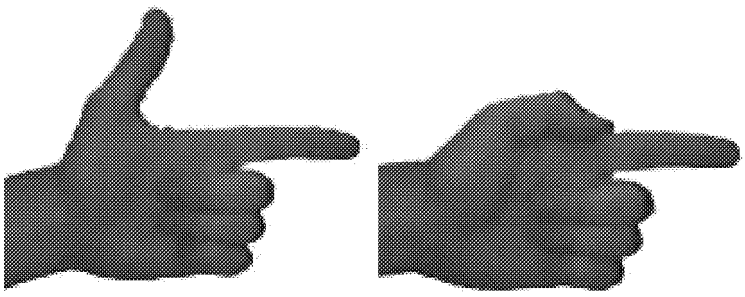

Figure 3: Two of Many Possible Pointing Postures

In such a case, the thumb would be different enough to declare one of these poses incorrect. In many postures, it turns out that the thumb is not playing an important role. Thus, when a program specifies a table of meaningful postures, it can tell the posture recognizer to either use or ignore the thumb for each of the postures. In figure 3, the thumb could be ignored, thereby resulting in the same posture. The thumb angles are still available, though, so they can be used independently of the posture to control some action. For example, if the user manipulated some virtual tool such as a virtual spray can or flashlight, the thumb could be used to toggle the tool on and off. Currently, only the thumb can be ignored in a posture, but it is conceivable that certain applications might need to ignore other fingers as well.

Each posture is represented as 16 angles and a flag signifying whether the thumb is to be used. With a call to, WhichPosture, the recognizer compares the current hand configuration to the table of known postures. This comparison is the average of the difference between each of the 15 (or 16 with thumb) angles. The posture from the table that is closest to the current hand posture is reported as the current posture. If the user's hand is currently too different from any of the meaningful postures (as defined by a threshold value), the system reports no current posture.

A service provided by the posture recognizer is the interactive definition of new postures. A user defines new postures by simply putting his hand in the desired posture and clicking a mouse button. Several postures can be defined this way and then tested to see how well they work together. In the testing phase, the program indicates what it thinks the posture of the hand is in to the user. This testing is important when creating a

set of multiple postures to ensure that none of them are too similar to cause the posture recognition system to evaluate them incorrectly. When the user is satisfied with the results, the posture utility program creates the table of known postures. In this fashion, it is easy to add new postures or customize an existing posture set.

Note that the hand may move and still be in the same posture. Thus, a static finger pointing posture, and the same finger pointing posture while the hand traces out a circle, are recognized as the same posture. If an application program needs to treat these two differently, then a separate gesture (as opposed to posture) recognition system is required. In our application, gestures are not used. But postures that may move over time are used. Each valid posture represents one of the visualization tools described in section 5. These tools are then able to generate a graphical representation of the data according to the tools' method of display.

4.4 Visualization Operations

The visualization system provides two basic methods of visualizing the data. These are isosurfaces and pseudo-coloring of cut surfaces.

An isosurface is created by tiling points in the data set with the same constant value. Here, we use the marching cubes algorithm [10] with a modified lookup table to remove ambiguities. Isosurfaces are rendered transparently so that tools manipulated within the data are always visible to the user. A family of transparent isosurfaces can also serve as a reference and give the user a good overview of the main features of the data. However, the user should realize that the shape of the isosurface is only the shape of the data set at the specified constant value. Inside or outside of the isosurface, the data may have a radically different shape, so further investigation is generally necessary in order to understand the true nature of the data set. Some means of exploring is also available by selecting different constant values through the graphical user interface or through the Threshold tool (see section 5). Varying the threshold value helps the user identify small features and general trends in the data.

When constructing an isosurface, data points are classified as being inside or outside of the isosurface depending on whether their value is above or below the constant value. Since this in/out classification is very similar to clipping operations, the isosurfaces can also be used as clip surfaces. With this realization, an option was implemented to clip a cut surface to an isosurface. That is, only the portion of a cut surface that lies within an isosurface is actually displayed. This option makes the cut surface look like a cross-section of an object.

Pseudo-coloring simply maps data values to color values. However, the color mapping process can be very subtle, or it can make a dramatic difference in what the user learns from the data. In general, the color mapping function should be designed for the specific data set being visualized and not just a generic color map. Thus, we allow the users to specify arbitrary mapping from data value to color by choosing a custom color palette file. That is, color maps are created by specifying any number of data values and the corresponding color to be mapped to a particular value. During visualization, if one of these data values is found, then the matching color is displayed. If the data point lies between two values specified in the color map, the color displayed is a linear interpolation of the two surrounding colors in RGB color space. Color maps are stored as external files and can be chosen by the user when viewing a data set. The user may even try out two or more different color maps on the same data set to see if one works better or reveals some feature hidden by the other one.

4.5 User Interface and Navigational Cues

Although the visualization tools are controlled directly by hand, there are still various details that would be cumbersome to specify with hand postures, such as specifying data set to examine, toggling isosurfaces on and off, specifying the sensitivity of tools, and adjusting viewing parameters. These details are controlled by various sliders and buttons. Fortunately, these are used relatively infrequently and the user may concentrate on his hands and how to move them through data.

During the design phase, we had the option of using a second tracker as a virtual camera with the eyeball-in-hand metaphor [16]. By moving this tracker around, the user would effectively change his viewing position. Holding the tracker away from his body and pointing it back at himself would result in a view from behind the data. Unfortunately, when viewing the scene from some arbitrary angle, the hand with the virtual tool would not move as expected. Such a third person point of view would require the user to consciously map his hand motions relative to the virtual camera, resulting in a less natural and direct interface. While this virtual eye may likely be useful in some applications, it was left out of this system. Instead, the second tracker is used to move and rotate the data volume. Together with the direct manipulation tools, arbitrary cuts can be generated with this two hand approach.

There are several visual cues to assist the user in navigating around the workspace while manipulating the tools and the data (see color plate 1). Since there is a possibility that these navigational aids may in fact interfere with the exploratory visualization process, they are all optional and may be disabled individually.

The first navigational aid is a wireframe box surrounding the data. Since the user has the ability to move the data volume anywhere, this is an essential navigational aid to quickly orient the user to the current position and orientation of the data volume. The second navigational aid is the hand model that represents the user's hand position within the data volume. At the coarsest level, it gives the user the knowledge of where the tools should be manipulated to visualize the data. The third navigational aid is placing the data volume within a virtual room. Instead of just letting the data float in empty space, the floor and two walls of the virtual room provide a sense of space to the user. It also helps somewhat in depth perception, but more importantly, it provides a context for casting shadows. Shadows can be cast by the data set and the virtual hand onto the walls and floor of the virtual room. Shadows are particularly useful for providing the user with the relative depth information of the virtual hand and the data. Because their primary purpose is as navigational aids, they do not need to portray all the details. Instead, we simply project the hand and the data onto the floor and walls. Hence, the shadows produced in this manner are very fast to compute and do not degrade the interactivity of the tools.

The final navigational aid is stereo vision. This is achieved with a pair of StereoGraphics glasses that rapidly presents alternate left and right views to the user. The stereo vision provides users with a better sense of depth and allows them to manipulate the tools and understand the data better.

4.6 Subdivision

The cutting tools mentioned in section 5 are responsible for describing the cut surface. Points on the cut surfaces are then used to sample the data volume which are then mapped to color. If every point on the cut surface were used to sample the data, the computation would be too expensive to maintain interactivity. To combat this problem, data values are only computed at triangle vertices. Gouraud shading is then used to render the triangles. At the expense of slower interactions, better approximations can be achieved by subdividing the triangles. We provide two subdivision schemes: regular and adaptive.

Regular subdivison is accomplished by connecting the midpoints of each triangle edge. The result is that the original triangle is transformed into four smaller triangles. Instead of just three, six data points are calculated after the subdivision. Figure 4 shows a hypothetical triangle that makes up a cutting surface and the same triangle after subdividing it into four smaller triangles.

Since the recursive level of subdivision will affect the fluidity of the user interaction with the tools, the level of subdivision is a parameter adjustable by the user so that he can decide on an acceptable tradeoff between speed and quality for a given situation.

An alternative subdivision scheme is to adaptively subdivide only in regions where the data is changing rapidly. Here, a triangle is split up only if the data values at its vertices are sufficiently different to warrant another level of subdivision. The meaning of sufficiently different can be modified by the user, but it is basically a test of whether the difference between the greatest data value and the least value of the three vertices is above a certain threshold. The threshold is expressed as a percentage of the total range from minimum to maximum in the entire data set. Adaptive subdivision allows the system to spend more time

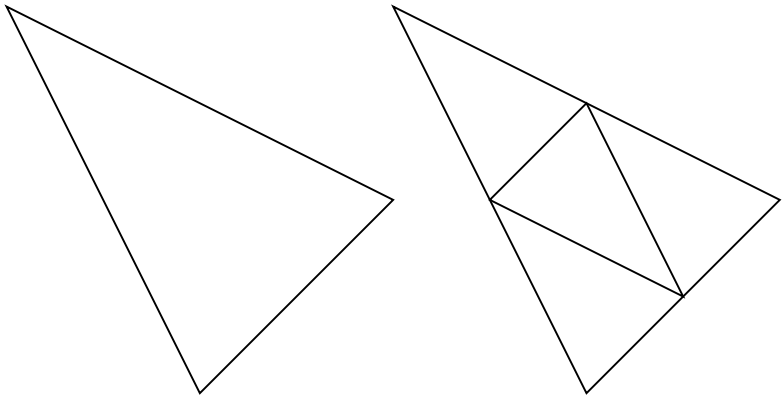

Figure 4: Original and Subdivided Triangle

in areas of finer details, without wasting time in relatively empty spaces. Figure 5 illustrates the results of adaptive subdivision on the hypothetical triangle, while figure 6 shows the subdivision through data with varying levels of detail. In areas where the data is changing rapidly, the plane is subdivided much more finely, whereas areas with constant (or near constant) data values, the plane is not subdivided very much at all.

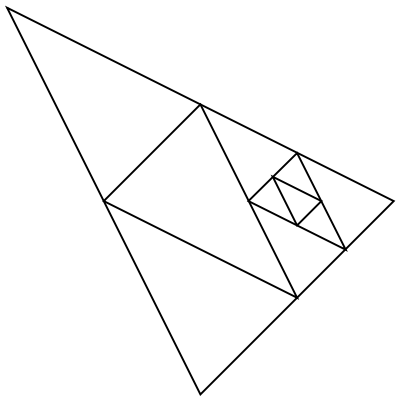

Figure 5: Adaptively Subdivided Triangle

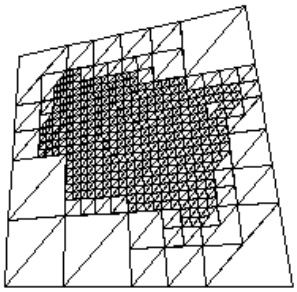

Figure 6: Adaptive Subdivision

Regular subdivision can be used together with adaptive subdivision. By default, the system is set up to perform two levels of regular subdivision followed by two levels of adaptive subdivision. During the exploration phase, the user can change the subdivision levels as desired.

5 Direct Manipulation Tools

Several direct manipulation tools are available to aid the user in exploratory scientific visualization. Each tool is activated by forming the appropriate posture for the tool. While we do not support gestures, the hand postures can be moved through space to define tool parameters. The sensitivity of each tool can be adjusted so that tools may be swept through large portions of the data with relatively small hand movements, or they can be moved through smaller regions with the same range of hand movements for close up manipulations.

All except one of the tools perform cutting operations. Following a knife metaphor, the tools turn the

hand of the user into a virtual blade that cuts through the data set. That is, they create cross-sectional surfaces that pass through the volume in which the data is defined and on which the data is displayed. The most straightforward case is a cross-section planar cut through the data volume. We also provide two extensions to the standard cutting planes. One that produces linear surfaces – defined by sweeping a straight edge through space; and one that produces curve surfaces – defined by sweeping a varying curved segment through space. All of the cutting tools make use of a data structure called a *cutPlane*. It is essentially a polygon which is positioned within the data space and is later mapped with color according to data values that it intersects. A simple cross section may use only one polygon while a tool that creates curved surface cuts may need to create many small *cutPlane*s to approximate the curved surface. Each tool is responsible for generating cuts by following the motion of the hand according to the method of operation of that specific tool. However, once *cutPlane*s are generated they are treated the same way by the visualization system.

As soon as a tool posture is recognized, the tool becomes active and continuously generates cut surfaces. In situations when the user wishes to view two cuts at once, or maybe a complex cut generated by multiple tools, a tool locking mechanism is used. Once a tool is locked, the visualizations from that tool are frozen in place, and a new tool or another instance of the old tool may be activated by the user to create new cuts. The switch that comes with the Cyberglove is used to lock the current tool without having the user shift his concentration away from the data. If there is no current tool when the switch is pressed, the most recently locked tool is deleted.

5.1 Flat Cut Tool

The first of the cutting tools in the system is the flat cut tool. The concept for the user when this tool is invoked is a chopping-like action. This is reinforced by the posture required to use this tool, a flat, open palm as shown in figure 7.

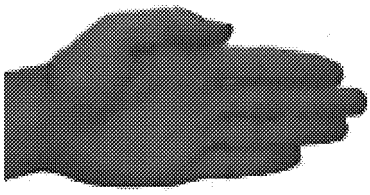

Figure 7: Flat Cut Posture

The flat cut tool generates a plane that is oriented with the user's palm and can be moved around intuitively as if it were an extension of the hand (see color plates 2 and 3). To convert the plane into a visualization object, the flat cut tool generates two triangle-shaped *cutPlane*s, properly oriented, and passes them to the visualization system for subdivision and color mapping. A good use of the flat cut tool is to drag it through the data, and watch the selected planes for important features in the data. Because it generates a flat, wide surface, it is a good tool to use on a new, unexplored data set to help locate features of interest and learn the overall layout of the data.

5.2 Axis-Locked Cut Tool

The axis-locked cut tool is very similar to the flat cut tool, except that any cut generated by this tool is constrained to be perpendicular to either the x, y, or z axis of the data set. To invoke the axis-locked cut tool, the user places his hand in a fist posture to denote a constraint or restriction as shown in figure 8.

When manipulating this tool, it moves freely with the hand of the user, but its orientation is snapped to the closest of the three axes. Since the data set can be rotated independently with the second tracker,

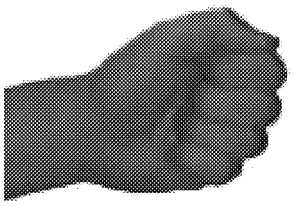

Figure 8: Axis-Locked Cut Posture

the axis-locked cut tool always aligns itself to one of the axes of the data set and not the world coordinate system (see color plate 4). The axis-locked cut tool is useful when a thorough examination of a data set is to be made. The constraints inherent in this tool makes it easy to maintain parallel cuts as the tool is dragged through a data set.

5.3 Linear Cut Tool

This cutting tool is more flexible than the previous two and is used to generate curved cuts. The behavior it tries to mimic is that of a knife blade cutting out a (possibly) curved sweeping path in 3-space. The hand posture for this tool is a common imitation of a cutting process as shown in figure 9.

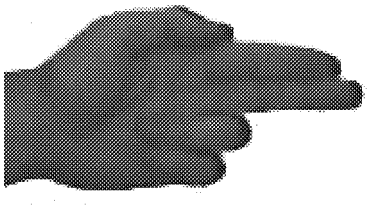

Figure 9: Curved Cut Posture

Once this tool has been activated by the user making the appropriate posture, it begins tracking the hand position. As the user moves his hand through space, the first two fingers trace out a 3D ruled surface (see color plate 5). This surface is defined to be the cut that the tool is making through the data and is reconstructed by placing many narrow rectangles side by side to approximate the curvature of the surface.

The linear cut tool is generally used when the user knows something about the layout of the data set. If this is the case, the user can employ the linear cut tool to sweep out a continuous, and possibly complex, surface through the region of interest. In keeping with the knife metaphor, this tool is also equipped with an adjustable blade length. The length is adjusted through a slider. A short blade is useful for constructing detailed cuts, or even for simple cuts where the user wishes to restrict it to a local region in the data. The blade may also be used in a long configuration, suitable for making a curved cut that extends through the entire data set.

5.4 Curve Cut Tool

The curve cut tool generates curved 3D surfaces. The hand posture mimics a knife with a curved blade that is swept through space. The hand posture in figure 10 illustrates this point. Note that the posture is similar to the curved cut tool posture, except that the first two fingers are bent to indicate that the knife blade is curved.

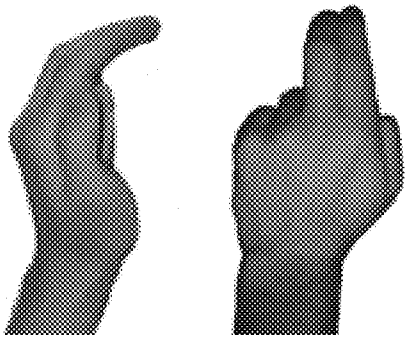

Figure 10: Doubly-Curved Cut Posture (Two Views)

With the curve cut tool, the user is able to generate a surface that is as much a scoop as a cut. The curvature along the knife blade is created by the three connected finger segments. As the user manipulates this tool, he may change the amount of bending in his first two fingers (see color plate 6). Changing this bending results in a change of the curvature of the blade of the tool. Thus, not only is the blade curved and able to be curved to different amounts, but the curvature is adjustable while the tool is in motion, varying the concavity of the blade as it cuts out a surface. As with the linear cut tool, it also shares the variable blade length feature.

The curve cut tool is best suited to cases where the user wants to exclude some data feature from the visualization or to view the shell area surrounding a feature.

5.5 Probe Tool

The probe tool is different in behavior and operation from the other tools in the system. Although it still is directly manipulated by the user through hand postures and motions, it is not used to create pseudo-colored cuts through the data. Instead, the user invokes the probe tool in order to query the data value at a specific point in space. The posture used to create a probe tool is the most common technique to indicate a location, pointing with the index finger as shown in figure 11. Note that the thumb posture is ignored for the this tool.

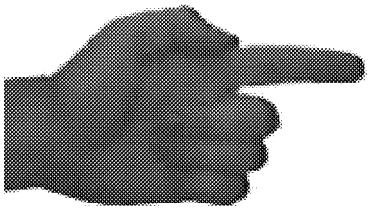

Figure 11: Probe Posture

When the probe tool is invoked, it replaces the display of the virtual hand with a hand that has the index finger replaced with a thin wire. The tip of this wire is the point being evaluated in the data set. As long as the tool is in use, the user may move his hand around to probe different parts of the data set. Next to the tip of the probe, the numerical value of the data at that point is displayed for the user.

5.6 Threshold Tool

Like the cutting tools, the threshold tool also generates surfaces. However, unlike the cutting tools, the surfaces are isosurfaces in the vicinity of where the hand is swept. To activate this tool, the user places his hand in a posture similar to a wiping action, as shown in figure 12.

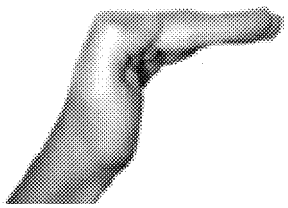

Figure 12: Threshold Posture

In addition to placing his hand in this position, the user specifies a threshold value for using a slider. When the threshold tool is moved through the data, local isosurfaces are generated through the swept area (see color plate 7). The user only needs to sweep out general areas of interest, and the threshold tool will seek out nearby regions of data at the threshold value. In the current implementation, nearby is defined to be in the range of the two nearest data points in each direction.

This tool is very useful for extracting features when the user knows the threshold value of interest, but may not know exactly where the feature lies. The computer could easily generate complete isosurfaces, but this may lead to a complicated geometry. By letting the user's hand directly guide the isosurface construction, local isosurfaces may be constructed which are easier to understand, both because they do not take up the entire display and because they are personally placed by the user.

5.7 Multiple Tools and Instances

Plate 8 shows a more complex example. This example uses the same data set as in plate 7. Several cuts have been applied by the user in different areas of the data set and the clipping and trimming features have been generously applied to keep the uninteresting parts of cutting planes from overlapping and obscuring the view.

6 Evaluation

Overall, the system has been found to be easy to use, and especially well-suited to a casual, exploratory style of visualization. There are some limitations to the system, but these are not fundamental flaws. Improvements could be made to correct many of the limitations, making it a more usable and robust system.

6.1 Successes

When evaluating this system, one should remember that it is driven by direct human interaction. This leads to both positive and negative results. On the positive side, the system is very easy to use. The user never needs to worry about angles or coordinates when placing cutting planes. Just a few simple hand postures need to be remembered, and these are easy to keep in mind because they are very similar to gestures used in the real world when performing various cutting operations. Direct manipulation has been found to be very natural and this can be attributed to the fact that in this system of interaction, the computer is being adapted to the user rather than the user to the computer (at least more so than using standard numerical or mouse-driven techniques).

Besides being easier to use, direct manipulation seems to promote a different style of interaction. Because it is so easy to manipulate a visualization tool in this system, the user will likely tend to change tools or change the position of a single tool at a moment's notice. This ability leads to a more casual type of interaction in which the user is more comfortable exploring the data without feeling restricted in what they can do. An exploratory approach is very useful when dealing with a new data set since the ability to rapidly alter cutting planes lets the user scan the overall data set quickly while looking for regions that might be of interest for later detailed analysis.

6.2 Shortcomings

The limitations of this system can be categorized in four ways. First, there are limitations due to the nature of human interaction. Second, there are limitations due to the current sensing technologies used in the glove and tracker devices. Third, there are limitations of compute power. And fourth, the visualization portion of this system has some drawbacks that should be addressed.

Unfortunately, the direct user interaction model also leads to some drawbacks. First of all, because of the nature of human interaction itself, it is difficult to be precise when manipulating visualization tools. If the user has a particular cutting plane in mind it would be very difficult, if not impossible to recreate that cutting plane using direct hand-controlled tools.

In addition, the magnetic tracking system always has a bit of noise, resulting in small vibrations of the cutting tools. When a tool is frozen by the user it no longer vibrates, but it could freeze anywhere in this vibration range. For ordinary work, the vibration is so small that its effect is irrelevant. However, it would be difficult to achieve exact positioning.

Because the tools are tied directly to the hand of the user, drawing speed is a major concern. If something on the screen does not keep up with a mouse movement, the user can usually deal with it. But if a cutting tool cannot keep up with hand motions, it becomes difficult to work with. In order to keep up with the hand position, it is necessary to avoid finely subdividing cutting planes. This in turn leads to imprecise pseudo-coloring on cuts. Using the system on faster computers will fix this situation, but it would be useful if the user could work on any machine with the required input hardware.

Another drawback of the system is that it currently only supports data that is sampled on a regular grid. This is a limitation of the visualization portion of the system, and not the user interaction model. If the user wants to visualize non-rectilinear or scattered data, there is currently no choice but to first resample the data onto a regular grid.

7 Summary

We presented direct manipulation tools for the express purpose of exploratory visualization. These tools operate satisfactorily for such use and within the constraints imposed by the current technology. However, they fail if precise positioning of tools is required. In those situations, traditional numeric entry would be more appropriate. During the course of our investigations, we have also identified a few areas for further enhancement and investigation.

Improved posture recognition is an area deserving of further study. The current system can distinguish among approximately six different postures. An improved posture recognition scheme might incorporate some form of learning algorithm to be able to distinguish among more postures. More postures would allow for more tools for the user. An improved system might also recognize gestures, or postures that change over time. Gestures provide a way to allow more complex forms of interaction.

Another important enhancement would be to support other data organizations (scattered data samples, curvilinear grids, etc.) and data types (vector data, multiple scalar values at a single point, etc.). In addition, time-varying data would be a useful addition. This could include time-varying scalar data such as temperatures in a volume, or vector data that is visualized using time-varying techniques.

To complement new data organizations, data types, and visualization requirements, other visualization metaphors should be investigated. We have identified and used other metaphors aside from the cutting tools, for example, spray cans and flashlights [15]. However, we have not adapted these to direct manipulation interfaces.

Besides making interaction easy for a single user, a multiple-user setup could assist in collaboration and instruction. If both users were equipped with gloves, one user might be the principle user, examining the data, but the second user could occasionally reach out and create a cut instead of giving vague verbal instructions telling the principal user how to work.

8 Authors' Biographical Sketches

MICHAEL CLIFTON. RealSpace, Inc. B.S., Electrical Engineering, UC Davis M.S., Computer Science, UCSC. Current research interests are in image-based rendering systems, multimedia, and real-time photorealistic rendering effects.

ALEX PANG. Assistant Professor of Computer Sciences at University of California, Santa Cruz. Current research interests are in computer graphics, scientific visualization, collaboration software, multimedia, and virtual reality interfaces.

References

[1] M.P. Baker. Human factors in virtual environments for the visual analysis of scientific data. Tech. Report (draft), National Center for Supercomputing Applications, September 1995.

[2] S. Bryson and C. Levit. The virtual wind tunnel. *IEEE CG&A*, 12(4):25–34, July 1992.

[3] B. Buxton. Smoke and mirrors. *Byte*, 15(7):205–210, July 1990.

[4] J. Carroll, editor. *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press, 1991.

[5] P. Cignoni, C. Montani, and R. Scopigno. MagicSphere: an insight tool for 3D data visualization. *Computer Graphics Forum*, 13(3):C317–328, September 1994.

[6] J.H. Clark. Designing surfaces in 3D. *Communications of the ACM*, 19(8):454–460, August 1976.

[7] J.D. Foley. Interfaces for advanced computing. *Scientific American*, 257(4):127–135, October 1987.

[8] T. He and A. Kaufman. Virtual input devices for 3D systems. *Proceedings Visualization '93*, pages 142–148, October 1993.

[9] William Hibbard and David Santek. The Vis-5D system for easy interactive visualization. In *Visualization '90*, pages 28–35, 1990.

[10] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (ACM SIGGRAPH Proceedings)*, 21(4):163–169, July 1987.

[11] Tom Meyer and Al Globus. Direct manipulation of isosurfaces and cutting planes in virtual environments. Technical report, Dept. of Computer Science, 1993. Technical Report 93-54.

[12] R. Moorhead, C. Everitt, B. Hamann, S. Jones, J. McAllister, and J. Barlow. Oceanographic visualization interactive research tool (OVIRT). *Proceedings of the SPIE*, 2178:24–30, 1994.

[13] D. Ney and E. Fishman. Editing tools for 3D medical imaging. *IEEE CG&A*, 11(6):63–71, November 1991.

[14] J. Pajon and V. Tran. Direct mouse-driven manipulation for interactive visualization of structured data. *Third Eurographics Workshop on Visualization in Scientific Computing*, pages 175–192, April 1992.

[15] Alex Pang. Spray rendering. *IEEE Computer Graphics and Applications*, 14(5):57 – 63, 1994.

[16] Alex Pang and Michael Clifton. Metaphors for visualization. In *Sixth Eurographics Workshop on Visualization in Scientific Computing*, page to appear, 1995.

[17] F.H. Rabb. Magnetic position and orientation tracking system. *Aerospace and Electronic Systems*, AE5-15(5):709–717, September 1979.

[18] M. Rhodes, Y. Azzawi, E. Tivattanasuk, A. Pang, K. Ly, H. Panicker, and R. Amador. Curved-surface digital image reformations in computed tomography. *Proceedings of the SPIE*, 593:89–95, December 1985.

[19] L.G. Roberts. The Lincoln wand. Tech. Report, MIT Lincoln Laboratories, 1966.

[20] E. Sachs, A. Roberts, and D. Stoops. 3Draw: A tool for designing 3D shapes. *IEEE CG&A*, 11(6):18–26, November 1991.

[21] B. Schneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, August 1993.

[22] A. Sellen, G. Kurtenbach, and W. Buxton. The prevention of mode errors through sensory feedback. *Human-Computer Interaction*, 7(2):141–164, 1992.

[23] D. Sturman. Whole-hand input. Ph.D. Thesis, Massachusetts Institute of Technology, December 1991.

[24] J. Udupa and D. Odhner. Fast visualization, manipulation, and analysis of binary volumetric objects. *IEEE CG&A*, 11(6):53–62, November 1991.

[25] C. Ware and D.R. Jessome. Using the Bat: A six-dimensional mouse for object placement. *IEEE CG&A*, 8(6):65–70, November 1988.

[26] T. Yoshimura, Y. Nakamura, and M. Sugiura. Direct manipulation interface: Development of the Zashiki-Warashi system. *Computers and Graphics*, 18(2):201–207, March/April 1994.

[4] [10] [20] [13] [26] [14] [3] [22] [21] [12] [2] [24] [17] [25] [23] [1] [6] [19] [7]