Development of a Virtual Environment for Interactive Interrogation of Computational Mixing Data

Virtual reality (VR) has the potential to enable visualization of complex physical phenomena in their natural spatial scales and significantly improve the product development process. Virtual environments provide a tool for users to interact with three-dimensional models using intuitive methods in their natural spatial scales. This paper presents the implementation details of a virtual environment combining surround-screen virtual reality technology and particle tracing of massless particles to explore the benefits these tools bring to engineers seeking to understand the complex fluid behavior present in mixing processes. The implementation of interaction methods, including volume data selection using a convex hull approach, cutting plane generation, and the integration of a tablet PC as a user interface, will be presented. Design and technical details involved with the interaction methods will be discussed along with the resulting interactive virtual environment. [DOI: 10.1115/1.2409314]

Keywords: mixing analysis, virtual reality, immersive environments, computational fluid dynamics, tablet computer

Background

Mixing processes are essential in the chemical process industries, including food processors, consumer product corporations, and pharmaceutical manufacturers. Traditionally, the design of chemical process machinery has been carried out by empiricism, practical experience, and manual calculations [1]. However, the increased use of computational fluid dynamics (CFD) during the design and analysis of static and stirred mixers has provided increased insight into mixing processes. In the case of a stirred mixing vessel, it has been shown that the velocity, temperature, and pressure components are not sufficient to completely understand the mixing processes [2]. Far greater insight about the performance of the mixing vessel can be obtained by observing the spatial distribution of massless material elements within a stirred vessel. The material spatial distribution can be obtained by tracing a large number of massless fluid particles through the flow field as the mixing process proceeds with time.

Surround-screen virtual reality (VR) provides an ideal method for analyzing scientific data. Users can visualize the complex phenomena with the geometry scaled to actual size. Furthermore, the large display area of surround-screen VR systems provides a means to immerse users into the mixing data where they can collaboratively investigate the flow features. The work presented here integrates the particle tracing computation power of the HyperTrace™ commercial software application with new data interrogation techniques made possible by the use of virtual reality technology.

Mixing Analysis Using Computational Fluid Dynamics

Computational fluid dynamics has become a primary tool for analysis and design for chemical processes in industry. CFD is widely used to determine the performance characteristics of stirred mixing vessels, heat exchangers, fluidized beds, and other process equipment. It involves the solution of partial differential equations using a numerical algorithm on a computer to predict fluid flow, heat transfer, and mass transfer [3]. The velocities obtained from CFD solutions can provide great insight into the fluid mechanics behavior of the flow field. However, this solution provides little information on quantities of practical interest in mixing, such as material distribution, to a designer of a stirred mixing vessel [4]. To obtain this spatial material distribution, it is necessary to carry the solution one step further and obtain particle histories [2]. Although relatively computationally expensive, one method to obtain information about material motion within a mixing vessel is by numerically integrating

\[
\frac{d\bar{x}}{dt} = f(\bar{x})
\]

where \(\bar{x} = x, y, z\)

In Eq. (1), \(\bar{x}\) represents the particle position and \(f(\bar{x})\) represents the interpolated velocity from the numerically solved flow field. One method of solving Eq. (1) is to use a fourth-order Runge-Kutta scheme with trilinear velocity interpolation between the nodal points of the velocity field. This method enables scientists to obtain the particle history for a significant number of massless particles. The particle histories allow scientists to acquire detailed mixing information. While at Cray Research, Liu et al. [5] developed one of the only commercially available packages designed exclusively for mixing analysis, marketed as HyperTrace™.

HyperTrace was designed to effectively trace several thousand particle trajectories within a mixing vessel or other chemical processing equipment. HyperTrace contains two main components, a solver and a graphical viewer. The solver performs three main tasks. First, the solved CFD analysis datasets generated from a commercial CFD package are read into HyperTrace. Next, a number of massless particles are placed at various simulation positions within the flow field. Finally, the solver calculates the position of each particle at each time step over a user defined time interval. A fourth-order Runge-Kutta integration scheme with adaptive step-size control and trilinear interpolation is used to obtain the velocity values between the nodes for each time step to solve Eq. (1).
The system geometry and particle positions are written to a binary file on a time-step–by–time-step basis. The solver can be used in conjunction with the graphical interface that is used to aid in placing the particles or it can be run in batch mode, reading the initial particle positions from a text file.

After the HyperTrace solver generates a solution file, the file can be viewed by a graphical viewer, and further examination can take place. The HyperTrace graphical viewer was developed using OpenGL [6] and utilizes a GUI developed in Tcl/Tk [7] to control the simulation parameters. The graphical viewer is able to generate active stereo images or monoscopic images on a desktop monitor. Within the HyperTrace graphical viewer, users can choose to visualize the particles as points or lines. Particles placed in different areas of the simulation geometry can be assigned different, static colors or can be colored dynamically by their velocity. In the case of dynamically colored particles, the solver writes the velocity information into the binary file and it is read in along with the particle position by the graphical viewer. In this way, users could use HyperTrace to analyze the behavior in a stirred mixing vessel. In this research, we develop the existing version of HyperTrace into a VR application, hereafter named VirtualTrace.

VirtualTrace Application Design

This research is designed to take advantage to the C6 synthetic environment at Iowa State University. The C6 is an immersive surround-screen VR system with six 10 ft. × 10 ft. projection screens (four walls, a ceiling and a floor) and a wireless tracking system. One of the four walls of the C6 slides open to allow users to enter and exit the C6. The C6 is driven by a SGI® Onyx® computer system equipped with 24-R12000 processors running at 400 MHz, 12 GB of RAM, and six InfiniteReality²™ graphics engines. All surfaces of the C6, including the floor and ceiling, are back projected. The C6 utilizes synchronized Barco projectors to generate the six stereo images on the display walls. These images are rendered at a resolution of 1024×1024. Users within the C6 wear CrystalEyes® shutter glasses to enable active stereo viewing. In active stereo, an image for the left eye and an image for the right eye are drawn and projected in sequence. The shutter glasses are synchronized to the images at a frequency of 96 Hz such that when the left-eye image is displayed, the right-eye image is opaque. To give users the correct perspective view, one user’s head position is tracked by an Ascension Flock of Birds wireless magnetic tracking system. VirtualTrace utilizes the existing Hypertrace calculation software and binary file reader in conjunction with VR Juggler [8] to produce graphics in the C6 virtual environment. VR Juggler is an object-oriented, open-source virtual reality application development platform. An important characteristic of VR Juggler is that it enables the use of multiple VR devices, such as CAVE™ systems, HMDs, and Powerwalls. This allows any application programmed using the VR Juggler API to run seamlessly on a variety of VR systems, enabling tremendous flexibility. VR Juggler can also be used on a variety of operating systems, including Windows®, UNIX®, and Linux®. Graphics within VR Juggler are handled by OpenGL [5], a platform independent, open-standard graphics API. An important design feature obtained by using VR Juggler and OpenGL is the ability to migrate VirtualTrace to a cluster of commodity PCs using Cluster Juggler [9]. As hardware prices continue to decline and increased computer power becomes available on desktop PCs, it is becoming highly likely that the large mainframe computers required for immersive VR, such as the SGI® Onyx®, and Onyx2®, could be replaced by clusters of commodity PCs with high-performance graphics cards. VirtualTrace has been successfully demonstrated to the VR community on a cluster of commodity PCs using Cluster Juggler with minimal application modification.

The VirtualTrace visualization software acts independently of the solver that generates the particle traces. Therefore, VirtualTrace initially loads the binary data file of particle positions and fluid system geometry pre-calculated by HyperTrace, following the prepayment methodology established by Gerald-Yamasaki [10]. After loading the datafile, the visualization software acts independently and the user can perform analysis on the precomputed particle positions in the virtual environment.

Input Interface. Scientific visualization applications typically have complex menus that are hierarchical and contain a variety of specialized commands that do not map well to simple button presses. In addition, interaction with menus in a virtual environment is difficult and can prevent users from seeing the data that they actually wish to interact with in a virtual environment. This has led to attempts to integrate palmtop computers into virtual environments as interaction devices. Watsen et al. [11] developed a Palm Pilot™-based interaction device that could display quantities of interest, such as buttons, menus, and sliders, within a virtual environment. This type of interface could also be position-tracked for interactions such as selection and locomotion. The Palm Pilot interface allows users to interact with the inherently two-dimensional (2D) menus in a natural and familiar way. Users could make menu selections on the Palm Pilot, and the virtual environment would respond to their input.

Hill and Cruz-Neira [12] extended this system when they developed JAIVE, or Java™ Interface to the Virtual Environment. Although the software was written to support a variety of palmtop computing devices, the application was illustrated on a Windows-based tablet computer. Hill’s implementation used a Java-based menu program to enable users to develop menus for their VR application. The IRIX®-based VR application interfaced with the Windows-based PC by using a TCP/IP socket. Connections between the tablet computer and the VR application are made during the initialization of the application. Changes to menu states on the tablet computer are immediately communicated to the VR application. JAIVE makes use of WaveLAN IEEE 802.11 Wireless Ethernet technology to handle the communication between the tablet computer and the visualization application.

The approaches Hill and Cruz-Neira [12] and Watsen et al. [11] provided an intuitive interface for complex menus within virtual environments. These systems are easy to use and allowed applications to move seamlessly from the desktop to VR while maintaining the same user interface. Additionally, a tablet computer allows input and display of numerical or text data in an intuitive manner. Finally, a tablet computer can act as both an input device and an output device by displaying data of interest.

Because the desktop visualization software developed for Hypertrace contained a GUI developed in Tcl/Tk that users of the software were familiar with, it made sense to maintain the functionality of this menu system within the virtual environment. By keeping the existing menu system intact, users could migrate from the desktop version to the VR version without retraining, enabling a more seamless transition. In addition, a menu system is valuable to enable complex interaction techniques, such as those described later in this paper. Therefore, an approach similar to JAIVE [12] was chosen. The Intermece Tablet Computer was selected as the input device for VirtualTrace, shown in Fig. 1. This computer utilizes an Intel Celeron® processor running at 266 MHz, containing 32 MB of RAM, and a 4.3 GB hard disk.

Since the existing menu system was built in Tcl/Tk, a TCP/IP networking class was developed to handle communication between the VR application and the Windows tablet computer. This class has nearly the same design as the JAIVE software, but instead of interfacing with Java, it interfaces with Tcl/Tk. The existing menus were ported to the Windows tablet computer shown in Fig. 1. A picture of a user utilizing the tablet computer to control the VirtualTrace simulation is shown in Fig. 2.

An object-oriented class was developed using TCP/IP [13] sockets to communicate between the tablet computer and the visualization program. After instantiating a connection between the tablet computer and the visualization program, Pthreads [14] are used to create separate threads for the communication tasks within the VR application. This essentially means that a separate process
is started during runtime that only handles the communication functions between the tablet computer and the visualization program. Once this thread is spawned, it can communicate back to the original application. These separate threads are used to poll for input on TCP/IP sockets so that any data received on a socket is immediately sent back to the application for processing.

On starting the VirtualTrace application, a network connection between the VR application and the GUI is instantiated. Next, the data file is loaded. This data file is written in binary and contains the coordinates for the geometry of the fluid system and the particle positions for each frame. Based on the input data, the necessary data structures are dynamically populated and the geometry is loaded into a display list. After all the frames have been loaded into memory, the user can run the simulation and view the results. The user can also use the interactive analysis tools to investigate different aspects of the data.

VirtualTrace Interaction Methods

VirtualTrace provides two methods to interact with the data in the virtual environment. This same approach was utilized with success by the IVRESS [15] CFD application. Simple interaction capabilities are provided with the wand. Examples of these types of interactions include placing particles within the tank to be mixed, defining a selection volume, and placing the cutting planes within the environment. For more detailed tasks, or for tasks for which the wand is not appropriate, the menu interface on the tablet computer is used. This provides support for several additional functions, including display of numerical data.

A central problem when displaying between $10^4$ and $10^5$ particles involves user perception of data. When viewing the chaotic motion of a mixing process in VR, users can immerse themselves inside a mixing tank, but they are essentially surrounded by a storm of particles. It is impossible for users to completely comprehend the data being displayed to them. Although users can gain valuable insights about the macroscopic properties of the flow, such as particle distribution, information on particle motion in specific areas is occluded. Because of this, important features of the flow may be missed, including areas of low particle velocity or long particle residence time. Knowledge of the location of these features can be useful for the design and analysis of mixing processes.

In this research, two methods have been developed and implemented to better view the chaotic motion present in mixing processes. These methods enable users to be able to understand the features of a specific flow.

Volume Selection in Virtual Reality. In an effort to understand the features of a given flow, it is desirable to be able to select volumes of particles from certain areas of the flow. This enables users to select subsets of particles and view the motion of only those particles as they progress through the flow, as opposed to viewing all of the particles. Although a great deal of effort has been spent on how to most efficiently compute and view particle traces in scientific literature, little effort has been placed on how to interact with them once they have been placed within a flow. There have been a number of papers published on selection methods, where the user picks an object to manipulate. However, there has been virtually no research on how to select groups of objects within a user-defined area, such as particles. Volume selection is the process of defining a bounding volume and selecting all objects within this volume. To the best of the author’s knowledge, there has been no published discussion about volume selection within virtual environments.

A great variety of selection methods have been used in VR. Research by Bowman and Hodges [16] provides an overview of three widely used techniques and their respective benefits and drawbacks. The methods presented include a ray-casting technique, an arm-extension technique, and a world-in-miniature technique. The ray-casting technique essentially uses a laser pointer drawn from the user’s hand or wand that is extended into space. Users gesture with their hand until this ray intersects an object of interest. This method is utilized for menu selections in many static menuing systems. The computer detects the intersection, and the user can subsequently move the object. The world in miniature technique involves drawing a small projection of the world near the user’s head with which the user can interact. In this technique, the user uses his or her hand to pick and place objects within the miniature world. These objects are mapped to the larger world, and changes in the miniature world are subsequently made in the larger world. Finally, arm-extension techniques are methods where the user’s arm either grows or contracts at a constant rate based on user’s input. The user selects objects when the virtual arm intersects an object in the world. The user can then pick and place the objects. Although these selection techniques allow users to select individual objects, they do not scale well to the selection of thousands of objects, such as particles in a stirred reactor.

Arns et al. [17] utilized a paintbrush technique in their VR-Gobi application to retrieve information on statistical data sets viewed in VR. The users could “brush” across the data points with the brush that was drawn from the wand. When the brush intersected a point, information about that point was displayed. This proved to be a good method to gather information about a fairly small section of points but was not extended to larger volumes of points.

![Fig. 1 Intermeck Pen*Key 6642 tablet computer](http://mechanicaldesign.asmedigitalcollection.asme.org)

![Fig. 2 User using tablet computer to control mixing simulation in virtual environment](http://mechanicaldesign.asmedigitalcollection.asme.org)
Providing a volume selection method for VR applications is a challenging undertaking. Users must be able to interactively modify the volume by either placing new points for the boundaries or modifying existing boundaries, possibly by dragging them to new positions. For this application, the former approach was selected and convex hulls were chosen as a method to define the volume. Convex hulls provide an intuitive interface to define volumes of interest by simply placing boundary points around the volume of interest. The hull is easily modifiable and can readily be expanded to include more particles. In addition, convex hulls provide an easy test to determine if a point is inside or outside of the selected volume. Once a volume is defined, the user can select the particles within that volume and hide all other particles.

Convex Hull Algorithm. A convex hull can be defined as the set of points that define the smallest convex set containing all of the points in a given set [18]. To illustrate this concept using a 2D example would be to define the set of points that a rubber band touches if you were to stretch a rubber band around all of the points in a plane as a convex hull. Expanding the example to three dimensions is equivalent to wrapping a volume of points with shrink-wrap. The points that are touched by shrink-wrap on the surface form the convex hull.

There are several algorithms that have been presented for the calculation of convex hulls. For this application, an implementation of the incremental algorithm presented by O’Rourke [19] is modified to run in VR. The theory of the incremental algorithm follows closely from the works of O’Rourke [19] and de Berg et al. [18].

The basic premise of the incremental algorithm is straightforward. The plan is to construct an initial convex hull from four points and then incrementally add each successive point. Each face of the convex hull will be a triangle. As points are added, the algorithm determines if they are inside or outside of the existing convex hull. If the selected volume is to be the convex hull, then each successive point p falls into one of two categories. Either the point p resides within the convex hull H or the point p resides outside of the convex hull H. If the point p falls inside the existing convex hull H, then the point can safely be discarded. However, if the point p does not reside in the convex hull H, then the point is added and the hull must be redefined.

To test whether point p is inside H the algorithm uses the fact that a point p is inside H if and only if p is to the positive side of the plane determined from every face of H. The three points from the triangular face, a, b, and c, can be used to define this plane. A point is defined as being inside of the plane if the determinant of the volume formed by the three points from face of the triangle, a, b, and c, and the point of interest, d, is positive. The point is outside of the plane if the determinant is negative. This determinant is calculated from Eq. (2), where points a, b, and c are the points from the triangular face of the hull, and d is the point being examined.

\[
det \begin{bmatrix}
    a_x & a_y & a_z & 1 \\
    b_x & b_y & b_z & 1 \\
    c_x & c_y & c_z & 1 \\
    d_x & d_y & d_z & 1
\end{bmatrix}
\]

(2)

Based on the sign of the result of this computation the algorithm either discards the point because it resides inside the hull, or adds the point to the convex hull. To determine that a point is inside of the convex hull, the algorithm must iterate this calculation over all of the faces of the hull. This is shown schematically in Fig. 3.

If the point lies outside of the existing convex hull H, the point must be added. Adding a point to a convex is a complex operation. First, the tangent planes between the hull, H, and the point, p, must be found. The planes form a cone with point p at the apex. This cone forms the new faces that will be added to the hull. This operation is a two-step procedure. First, the algorithm determines which of the faces from the existing convex hull are visible to the point. This is done using Eq. (2). When determining whether or not the point is inside the convex hull, the algorithm checks for a positive sign from the determinant. A point is inside of the hull by definition if the determinant of the point of interest and all faces, calculated by Eq. (2), is positive. However, when determining which faces of the convex hull are visible to the point, the algorithm searches for negative signs from the determinant. Negative signs indicate that the point is outside of the plane formed by the triangular face. This indicates that the point is visible to the face. Figure 4 shows a convex hull with a point being added. The point being added is shown in blue. The triangular faces of the hull shown in yellow are determined to be visible to the point. Alternatively, the faces of the hull shown in green are invisible to the point.

After determining which faces are visible to the point, a cone of new faces is added to the hull. Since the edges of the visible triangular faces are also visible to the point, the algorithm

Fig. 3 Flowchart of determination process for whether or not a point is inside of a convex hull

Fig. 4 Convex hull. The point in the center of the image is being added. Yellow faces are visible to the point; green faces are invisible to the point (courtesy of Tim Lambert, University of New South Wales).
searches for the edges that define the boundary of the visible region. Since each edge touches two faces, the edge is defined to be a boundary edge if it has one adjacent face visible to the point and one adjacent face not visible to the point. Edges meeting this criterion can be defined as a boundary edges. The boundary edges are shown in Fig. 4 as the edges that border both a yellow and green triangle. After the algorithm determines which edges form the visible boundary with the point \( p \), new triangular faces are built with the point \( p \) and the boundary edges. Finally, all edges and faces that were previously visible to the point are deleted, leaving a new convex hull. A flowchart of this algorithm is shown in Fig. 5.

**VirtualTrace Convex Hull Implementation.** This method was implemented into an extensible C++ class, where the points are input using a linked list developed through user selection in the VR application. For the implementation of this algorithm, the first four points given form the initial hull. It is crucial that the four points be noncoplanar, since a convex hull cannot be formed from coplanar points. Therefore, the initial convex hull will always be a pyramid. Once the convex hull has been built, the algorithm can test whether or not a point resides inside or outside of the hull as described above. The points that reside outside of the convex hull are turned off, and only the points within the hull are visualized.

By selecting points in this manner, users can determine the motion of the flow in specific regions. A user building a convex hull in the virtual environment is shown in Fig. 6. A view of the convex hull is shown in Figure 7. The user utilizes the wand to interactively place points within the virtual environment to create the hull. Once four points have been placed within the environment, the hull is drawn. The user can continue adding points until he or she feels that an adequate volume has been defined. After defining the convex hull, the user can select the points that are within the volume and hide all points that are outside of the volume. The user can then visualize the particle motion. A user is shown visualizing the selected particles selected from the hull in Fig. 8. The datasets visualized in Figs. 6–8 are courtesy of Minye Liu, formerly of Cray Research.

The activity of volume selection is a natural activity in VR, allowing users to select specific areas of particle activity and visualize the results. Volume selection is cumbersome, if not impossible, on a desktop display, and there is no natural analog to volume selection with a wand on a desktop. In VR, users can move about a full-scale mixing vessel with between \( 10^4 \) and \( 10^6 \) particles and select particle from areas of interest and observe their fluid motion independent of the macroscopic properties of the flow.

**Cutting Planes.** Another feature of use in the analysis of many fluid-mixing simulations is a cross-sectional plot. Cross-sectional plots provide a means to examine flow through static mixers and determine whether or not a mixer is providing a uniform particle distribution at its exit. These plots also provide a means to show areas of regular particle motion and areas of random, or chaotic, particle motion. Finally, these plots allow the user to show areas of high and low velocity within the tank. A cross-sectional plot...
shows the location of particle intersection and the velocity of the particle when it crossed the cutting plane of interest. The equation for a plane is as follows:

$$Ax + By + Cz + D = 0$$  \hspace{1cm} (3)

where

$$x = x_1, x_2, x_3$$

$$y = y_1, y_2, y_3$$

$$z = z_1, z_2, z_3$$

The position of the plane is arbitrarily constrained to one of the three principal axes of the model geometry, as placed by the user. After selecting the constrained axis on the tablet computer, the user selects the plane placement position with the wand. Once this selection position is set, three points on the plane are known. Utilizing these three points, it is possible to solve for the values of $A$, $B$, $C$, and $D$ in Eq. (3). The solution to these values is given by the determinants in the following:

$$A = \begin{bmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{bmatrix}$$  \hspace{1cm} (4)

$$B = \begin{bmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{bmatrix}$$  \hspace{1cm} (5)

$$C = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$  \hspace{1cm} (6)

$$D = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$  \hspace{1cm} (7)

After placing the plane in the virtual environment, the user begins the simulation. At each frame, every particle is tested to determine if it has intersected the cutting plane. The intersection of the massless particles and the user placed plane within the virtual environment is calculated between the current particle position $p^t$ and the particle position in the previous frame $p^{t-1}$. Since point’s $p^t$ and $p^{t-1}$ defines a ray, particle intersection can be tested using this ray. The parametric equation of a line can be written in the following form:

$$\vec{P} = \vec{P}_1 + u(\vec{P}_2 - \vec{P}_1)$$  \hspace{1cm} (8)

In Eq. (8), we substitute $p^t$ for $P_1$ and $p^{t-1}$ for $P_2$. Using these values, we can solve for the parametric parameter $u$ in Eq. (8), using

$$u = \frac{Ap^t_x + Bp^t_y + C p^t_z + D}{A(p^t_z - p_1^{z-1}) + B(p^t_z - p_2^{z-1}) + C(p^t_z - p_3^{z-1}) + D}$$  \hspace{1cm} (9)

Once $u$ is found, we check to see if it is between 0 and 1, indicated that the intersection of the line and the plane takes place between points $p^t$ and $p^{t-1}$. If $u$ is determined to be between 0 and 1, then the intersection point is found through linear interpolation and the intersection point is displayed on the cutting plane. The color of the point on the cutting plane is determined by linearly interpolating the velocity values between the cutting two points, $p^t$ and $p^{t-1}$.

Virtual reality provides a natural interface for the placing and interaction of cutting planes in a virtual environment, and the wand is utilized to allow us to place the cutting plane at a location of interest within the virtual environment. A user placing a cutting plane in the virtual environment is shown in Fig. 9. The flow field shown in Fig. 9 is Poiseuille flow, solved numerically using FLUENT® on a 5000 cell grid. The resulting cross-sectional plot is shown in Fig. 10.

**Results and Conclusions**

VirtualTrace has been presented to several groups from both industrial and academic backgrounds. Users from Dow Chemical, Procter & Gamble, and Fluent have viewed mixing data using VirtualTrace. Scientists and engineers have been generally enthusiastic about using VirtualTrace. The application proved to be an excellent communication tool that provided a visual focus to stimulate discussions that resulted in better understanding of the analysis results. Because of the tablet interface that closely matched the traditional HyperTrace® interface, users who had
Acknowledgment

Financial support of this research was provided by a grant from Procter & Gamble, Cincinnati, Ohio. The authors would also like to thank Joseph Kitching and Krista Comstock, from Procter & Gamble, and Dr. Minye Liu, formerly of Cray Research, for their technical assistance.

References


previous experience using HyperTrace felt immediately comfortable with the user interface and there was only a short training period needed before they could fully investigate the dataset. Perhaps one of the most beneficial aspects to the development of this VR interface was the ability to investigate the data using full-scale immersion. Position tracking and largescreen stereo projection allowed the users to interact with the data in ways they had not interacted before, by moving around inside the data easily with normal human motions instead of mouse and keyboard commands. New insights were achieved when the data were displayed in a fully immersive virtual environment instead of scaled to fit the computer monitor. Users felt that the immersion added to their perception of the subtle relationships occurring in the scientific data they were viewing.