1-1-2002

Interactive interrogation of computational mixing data in a virtual environment

Thomas Jerome Duncan
Iowa State University

Interactive interrogation of computational mixing data in a virtual environment

by

Thomas Jerome Duncan

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Judy M. Vance, Major Professor
Francine Battaglia
Rodney Fox

Iowa State University

Ames, Iowa

2002

Graduate College
Iowa State University

This is to certify that the master's thesis of

Thomas Jerome Duncan

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the many people who helped me conduct my research and ultimately made the completion of this thesis possible. First and foremost, I would like to thank my parents who have made my education possible and who have taught me to work diligently in the direction of my dreams. Without their financial assistance, as well as emotional support, none of this would have been possible. Secondly, I would like to thank Dr. Judy Vance, who has served as a mentor for me throughout my graduate education. The time and effort that she put into my research and this thesis were invaluable. Her support and encouragement helped see this work to completion. In addition, I would like to thank Dr. Francine Battaglia and Dr. Rodney Fox for their assistance in ensuring the completeness and accuracy of this thesis. Third, I would like to thank the Procter & Gamble Company, Cincinnati, Ohio, for their financial and intellectual support of this work. Joseph Kitching and Krista Comstock provided invaluable assistance during the development on this thesis. Finally, I would like to thank my many colleagues at the Virtual Reality Applications Center for their many contributions to this thesis, particularly: Ross Fischer, for his assistance with the construction of this thesis; Denis Dorozhkin, for being a boundless resource on all things related to Virtual Reality; Andrew Schwantes and Justin Hare, for being C++ experts and passing on some of their knowledge to me; Ronald Sidharta, for helping me to develop Speech Recognition software; and finally, Dr. Carolina Cruz-Neira and the VRJuggler development team for their help and support of this research.

ABSTRACT

Mixing processes are essential in the chemical process industries, including food processors, consumer products corporations, and pharmaceutical manufacturers. 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. However, the velocities, temperatures, and pressures are insufficient to completely quantify a mixing process. A more complete understanding of mixing processes is given by the material spatial distribution of massless particles as they move through the flow field.

This research seeks to combine surround-screen virtual reality and particle tracing of massless particles into an interactive virtual environment to explore the benefits these tools bring to engineers seeking to understand the behavior of fluids in mixing processes. Surround-screen virtual reality (VR) provides a means to immerse users into the mixing data where they can collaboratively investigate the flow features as displayed on a large scale stereo-projection system. This work 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. Parallel processing to facilitate interactive placement of particles in the flow, volume data selection using a convex hull approach, cutting plane generation, and the integration of voice control and a tablet PC will be presented. Both a stirred mixing vessel and flow through a duct will be used as examples. Finally, the benefits of VR applied to mixing analysis are presented, along with some suggestions for future work in this area.

CHAPTER 1. INTRODUCTION AND MOTIVATION

The goal of this research is to develop an effective method to evaluate the performance of process mixing equipment, including stirred mixing vessels and static mixers. The food processing, consumer products, pharmaceutical, and petroleum industries all rely heavily on process mixing equipment to produce their respective products. Poor mixing results in low product yield, causing low efficiency and high consumer cost. The objective of this research is to develop a computer application that will allow users to view and interact with fluid dynamics analysis results in a virtual environment to allow rapid evaluation of the behavior and performance of mixing vessels. A key component of this research is the ability for a designer to be able to interact directly with the flow field as opposed to simply watching a pre-computed solution. Enabling users to directly interact with the flow field will promote greater understanding of mixing processes. A greater understanding of the fluid mechanics of chemical process mixing equipment will result in more efficient designs and higher product quality.

Traditionally, the design of chemical process machinery has been carried out by empiricism, practical experience, and manual calculations (Bakker et al., 2001). However, new computational tools have shown significant potential to increase the effectiveness of designs and reduce their cost. Computational fluid dynamics (CFD) is a method extensively used to analyze the behavior of fluid flow in a variety of applications. CFD attempts to solve the flow field for the x, y, and z velocity components, the temperature, and the pressure numerically by using the Navier-Stokes equations, or some simplified form of the equations such as the Euler equations. These computations are performed on a pre-generated grid over

the computational domain. The computation time in solving the governing equations is proportional to the complexity and size of the grid, as well as the boundary conditions, and the model equations used. For complex fluid systems, such as those found in industrial application, the time is often extremely long. Additionally, grid-generation for fluid systems involving complex geometries is a time-intensive and non-trivial process. Current numerical methods and computational power make it impossible to modify the geometry of the fluid system without completely resolving the governing equations throughout the flow field. This precludes a scientist from being able to modify the system geometry and view the results interactively.

In the case of a mixing vessel, it has been shown that the velocity, temperature and pressure components are not sufficient to completely understand the mixing processes (Lamberto et al., 2001). 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 proceeds with time. While a great number of applications have been developed that provide particle-tracing support, few provide the performance needed to trace tens of thousands of particles. In addition, accurate particle tracing is a time intensive process, which prevents interactive analysis.

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 allows for several scientists to analyze data at once allowing for collaboration and interaction. Finally, VR provides a natural interface to

interrogate scientific data by providing the ability to walk around the data and fully explore the 3D analysis results.

In this thesis, the development of a virtual environment for the interactive examination of fluid mixing problems is presented. This research utilizes an existing commercial mixing simulation package, HyperTrace™, as a starting point. The two components of HyperTrace™, the solver and graphical viewer, are expanded into a distributed interactive virtual reality application, named VirtualTrace. This is accomplished by improving the speed of the existing HyperTrace™ solver and replacing the existing visualization software with a virtual reality interface. The speed of the solver was improved by providing distributed memory parallelization. The performance of two different parallel programming paradigms, which are used to perform distributed computation of particle traces, is compared. In addition, the solver was modified to communicate particle positions through a network connection rather than a file. The visualization software was rewritten to utilize a VR interface. Analysis and interaction techniques specific to the problem of visualizing several thousand particles simultaneously are presented.

Chapter 2 of this thesis discusses the current state of VR for CFD and interactive design methods for finite element applications as they relate to the future of CFD analysis. A discussion of the design of virtual environments and parallel processing methods utilized is contained in Chapter 3. Chapter 4 presents an overview of interaction methods in virtual environments and elaborates on the interaction methods used in this research. Data interrogation methods utilized specifically for mixing analysis are detailed in Chapter 5. Chapter 6 gives an example of the application that was developed. The final chapter, Chapter 7, gives conclusions and recommendations.

CHAPTER 2. BACKGROUND

This chapter provides an overview of computational fluid dynamics (CFD), the use of CFD for fluid mixing simulation, and virtual reality (VR). Overviews of the VR applications developed for CFD examination are presented. In addition, an overview of the commercially available mixing analysis packages is presented. A discussion of interactive design methods that have been developed for finite element analysis (FEA) and CFD is also included. In-core and out-of-core visualization methods are also surveyed.

2.1 Computational Fluid Dynamics

Computational fluid dynamics (CFD) 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. CFD involves the solution of partial differential equations using a numerical algorithm on a computer to predict fluid flow, heat transfer, and mass transfer (Bakker et al., 2001). The general solution process for a CFD problem can essentially be broken into three steps which include pre-processing, numerical analysis, and post-processing (Tannehill et al., 1997). Pre-processing involves generating a grid for the fluid system and applying appropriate boundary conditions to this grid. Numerical analysis is the process of reducing the governing partial differential equations to an algebraic form and solving them using a numerical algorithm. These algebraic equations are solved at the nodal points of the grid with an appropriate numerical method. Velocities, temperatures, and other scalar quantities at each nodal point in the grid are calculated. Once a solution has been obtained from numerical analysis, users perform post-processing, which involves the examination of the calculated results at the nodal points displayed on the system geometry.

The governing set of partial differential equations for fluid flow problems includes the continuity equation, the conservation of momentum equation, and the conservation of energy equation. These five partial differential equations permit the complete description of the dynamics of a flow field if the flow contains no mass diffusion or chemical reactions. The continuity equation is the differential equation representing conservation of mass for a fluid flow. This is shown in equation 2.0.

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{2.0}$$

In the above equation and the equations that follow x, y, and z represent the Cartesian coordinate axis and u, v, and w represent the x, y, and z, velocities respectively. The conservation of momentum equations, analogously known as the Navier-Stokes equations, are non-linear, second order partial differential equations representing Newton's second law applied to a fluid element. This is shown in equation 2.1.

$$\begin{aligned}
\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)&=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)\\
\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)&=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)\\
\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)&=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)
\end{aligned} \tag{2.1}$$

In equation 2.1, ρ represents the density, p represents the pressure, and μ represents the fluid viscosity. The conservation of energy equation is the application of the first law of thermodynamics to fluid flow. This is shown in equation 2.2.

$$\rho c_v\left(\frac{dT}{dt}+u\frac{dT}{dx}+v\frac{dT}{dy}+w\frac{dT}{dz}\right)=k\left(\frac{d^2T}{dx^2}+\frac{d^2T}{dy^2}+\frac{d^2T}{dz^2}\right)+$$
$$2\mu\left[\left(\frac{du}{dx}\right)^2+\left(\frac{dv}{dy}\right)^2+\left(\frac{dw}{dz}\right)^2\right]+\mu\left[\left(\frac{du}{dy}+\frac{dv}{dx}\right)^2+\left(\frac{du}{dz}+\frac{dw}{dx}\right)^2+\left(\frac{dv}{dz}+\frac{dw}{dy}\right)^2\right] \quad (2.2)$$

In equation 2.2 T represents the temperature, c_v represents the specific heat, and k represents the thermal conductivity. During the numerical analysis process, the partial derivatives in the above equations are replaced with algebraic equations using finite difference or finite volume methods. In many cases assumptions can be made that enable a simplified version of these equations to be solved without sacrificing accuracy which significantly reduces computational time.

It is common practice to use different software packages for preprocessing, numerical analysis, and post-processing. Grid generation is often a manual and iterative process for complex parts or fluid systems, and is often time consuming. A wide variety of grid generation packages exist, and some computer aided design (CAD) packages contain grid generation capabilities. Solving the governing equations is a computationally intensive process, often requiring hours or days to complete for complex geometries. Commercial solvers such as Star-CD™, Fluent™, or CFX™ are widely used to solve the governing equations. Commercial post-processors such as Ensight™or Fieldview™ are widely used in industry and research to examine the resulting vector and scalar values.

2.2 Mixing Analysis using CFD

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 (LaRoche, 1998). To obtain this spatial material distribution, it is necessary to carry the solution one step further and obtain particle histories (Lamberto et al., 2001). While relatively computationally expensive, one method to obtain information about material motion within a mixing vessel is by numerically integrating equation 2.3.

$$\frac{d\vec{x}}{dt} = f(\vec{x}) \tag{2.3}$$

where: $\vec{x} = x, y, z$

In equation 2.3, $\vec{x}$ represents the particle position and $f(\vec{x})$ represents the interpolated velocity from the numerically solved flow field. One method of solving equation 2.3 uses a fourth-order Runge-Kutta scheme with tri-linear 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.

2.2.1 HyperTrace™ Mixing Software

While at Cray Research, Liu et al. developed one of the only commercial packages designed exclusively for mixing analysis (Liu et al., 1998). HyperTrace™ was designed to effectively trace several thousand particle trajectories within a mixing vessel. HyperTrace™ contains two main components, a solver and a graphical viewer. The solver performs three main tasks. First, it reads the CFD analysis datasets obtained from a solution performed by a commercial CFD package. Next, it places a number of massless particles at various simulation positions within the flow field. Finally, the solver iterates the solution for the desired number of time steps. The solver can be used in conjunction with the graphical

viewer to aid in placing the particles, or it can be run in a batch mode, reading in a text file. The solver generates a specific number of time steps over a given time interval, which are input parameters given by the user. This software makes use of a fourth order Runge-Kutta integration scheme with adaptive stepsize control and tri-linear interpolation to obtain the velocity values between the nodes for each time step to solve equation 2.3. It then writes the particle positions to a binary file on a time step by time step basis.

After the HyperTrace™ solver generates a solution file, the file can be viewed by a graphical viewer, and further examination can take place. HyperTrace™ was developed using OpenGL (Woo et al., 1997) and utilizes a GUI developed in Tcl/Tk (Foster-Johnson, 1997) to control the simulation parameters. The graphical viewer is able to generate active stereo images or monoscopic images on a desktop monitor. In this way, users could use HyperTrace™ to analyze the behavior in a stirred mixing vessel in an iterative manner, first by generating a data file and then by viewing the data file.

HyperTrace™ also has the ability to generate statistics on the particle distribution. To generate statistics, HyperTrace™ initially generates a grid of equally sized square bins over the geometry of the fluid system. The nodal points of this grid are then checked against the underlying flow field to determine if they are inside or outside of the bounding geometry. As long as at least one nodal point from each bin is within the flow field, the bin is considered inside of the bounding geometry. Bins that are partially protruded from the boundary geometry are considered fully inside of the vessel. Any bins that are fully outside of the boundary geometry are discarded. Using these bins, two quantities of interest are generated, the volume coverage, which is defined as the total number of bins containing a

particle divided by the total number of bins, and the variance between the number of particles in the bins. The variance is calculated as shown in equation 2.4 (Liu et al., 1998).

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(n_i - \beta)^2 \qquad (2.4)$$

In equation 2.4, σ^2 is the variance, β is the average number of particles per bin, N is the total number of bins, and n_i is the number of particles in the i^{th} bin. These quantities are written to a text file during the generation of particle traces. One drawback of this method is that the statistics are not visible in the GUI and they must be computed prior to running the visualization program.

2.3 Virtual Reality

Virtual Reality (VR) technology allows users to become immersed within a three-dimensional, computer generated world. There are many definitions for VR. Stuart (Stuart, 2001) lists the essential elements of VR as immersion, presence, and engagement. Immersion results from the sensory cues that convey to a user that he or she is surrounded by a computer generated environment. Presence is the user's feeling that he or she is a part of the virtual environment. Engagement is a result of the user being deeply involved in the activities taking place within the virtual environment. An alternative definition given by Brooks (Brooks Jr., 1999) in his 1999 IEEE position paper on VR, states that a virtual reality experience is one in which the user becomes immersed in the virtual world and has dynamic control of the viewpoint. Utilizing this as our framework for VR, a virtual environment that makes a user feel as if he or she is an active participant in the world, rather than a passive observer must be created.

Scientific visualization, architectural walkthroughs, virtual prototyping and entertainment application have been among the most popular applications for VR. Virtual reality provides a natural interface to view large numerical datasets that result from scientific calculations. Through the use of spatial position trackers and stereo viewing, users can move around and through the computer generated scene. Furthermore, VR allows scientists to view these data sets in their real size, as opposed to viewing a scaled down version on a computer monitor. Finally, the majority of scientific problems currently being addressed are inherently three-dimensional in nature, which makes them attractive applications for VR.

VR allows scientists to view abstract quantities and concepts that would be impossible to visualize in the real world, giving them insight that would be otherwise impossible (Bryson, 1992). Van Dam et al. (Van Dam et al., 2000) advocate the use of VR for scientific visualization, citing the belief that VR will enable users to examine data more efficiently even though they believe the field is currently in an immature state. They maintain an effective scientific visualization application will enable users to naturally interact with the data being viewed. By interacting naturally with complex data in real life scales, it is believed that users can gain insight into the data not possible on the desktop, speeding innovation and creating superiorly designed systems.

In order to effectively create a virtual environment, specific hardware and software must be used. The hardware required for VR can be categorized into interaction devices, tracking devices, graphics engines, and display devices. VR interaction devices can be thought of as an analog to traditional keyboards and computer mice. The concept of VR interaction devices is an active research topic in the field of Human Computer Interaction (HCI) and will be discussed further in Chapter 4. Tracking devices provide the position and

orientation of objects within the virtual environment. The position and orientation of the user's head is necessary for accurate stereoscopic perspective viewing. Graphics engines produce the geometric primitives that are displayed within the virtual world. In this research, the graphics engine and the computer to which it is attached are considered one entity. Display devices span a wide spectrum, from single-user devices to multi-user devices. Single-user devices include Binocular Omni-Orientation Monitors (BOOM®), shown in Figure 2.1, and Head Mounted Displays (HMD), shown in Figure 2.2. Multi-user devices include surround-screen displays and PowerWalls. For this research a surround-screen active-stereo device, the C6, is utilized. This device is similar to the Cave Automatic Visualization Environment (CAVE™) (Cruz-Neira et al., 1993). The primary reason for choosing a surround-screen device as opposed to a HMD or a BOOM® is the ability for groups of participants to experience the virtual environment together. By utilizing a surround-screen system, several scientists can view the data at once enabling greater collaboration among users.

Figure 2.1 - Binocular Omni-Orientation Monitor (BOOM®). Image courtesy of Fakespace Labs.

Figure 2.2 - Head Mounted Display (HMD).

2.4 Interactive Design in Virtual Reality

Interactive design is the ability to change input parameters in a given system and visualize the subsequent changes immediately. VR is an ideal medium for interactive design because it allows designers to view things in their actual size and orientation. For many mechanisms and systems, this is critical to understanding the implications of a design change. For example, VR allows mechanics to determine whether or not they will be able to easily remove a part from a tractor before a physical prototype is made. Computational steering is the ability to control a simulation while it is progressing and affect the solution. Using VR in conjunction with computational steering technologies has the potential to reduce the number of iterations in the design cycle, saving manufacturers days or months in time, and reducing finished product cost considerably.

Yeh and Vance (Yeh et al., 1998) developed a VR application in which users could interactively change a part in the virtual world and immediately view the changed stresses on the part. The application started with an initial solution from a commercial FEA program. A linear Taylor series approximation was used to compute the stress changes quickly. While the FEA approximation method was not highly accurate, the application allowed designers to modify a part using natural human movement and get an intuitive feel for the areas of high stress affected by the shape change. Finally, the modified part is analyzed with a commercial FEA package to verify the design. Following this work, Chipperfield (Chipperfield, 2002) utilized parallel programming, the conjugate gradient method, and a meshless grid to recompute stresses on the part quickly. The method follows a two-step procedure. While the part shape is being changed in VR, the stresses are computed using a linear Taylor series approximation. Once an acceptable shape has been achieved, the actual stresses are computed using a conjugate gradient approach. The meshless method was used to avoid mesh distortion for large shape changes. This research added more accurate calculation of stresses for computational steering of interactive design, and gave designers a better feel as to how the part geometry changes affect stress distribution.

Ideally, the interactive computational steering research in FEA will guide the development of computational steering tools for fluid flow and heat transfer problems. For fluid mechanics and heat transfer, it would first be necessary to generate a grid, apply boundary conditions, and arrive at an initial solution. Next, an engineer could modify the system and see an approximation of the changes in the solution space. Finally, the fluid system would be resolved for the new parameters as a final step. Bryden, et al. (Bryden et al., 2000) have developed an application that interpolates flow characteristics between 58

solutions of a two-dimensional laminar flow over two cylinders. In this application, the user can move the cylinders independently within the virtual environment and observe the changes to the temperature field. However, it is unrealistic to solve a system 58 times for many complex three-dimensional flow situations. Faster and more accurate interpolation techniques need to be developed to utilize this for more advanced three-dimensional flows. Improved approximation methods could significantly cut the number of design iterations required for complex fluid systems. Utilization of genetic algorithms for this purpose is currently being researched.

Interactive design has the potential to significantly impact new product development. These techniques can significantly reduce the number of design iterations, resulting in a faster time to market, reduced product cost, and increased productivity.

2.5 Past and Present CFD work in Virtual Reality

In this section previous research in visualization of CFD results in VR will be surveyed. A number of applications have been developed for CFD visualization in VR, and their strengths and limitations will be outlined here.

2.5.1 The Virtual Wind Tunnel

The Virtual Wind Tunnel (VWT), developed by Bryson, et al. (Bryson et al., 1992), allowed scientists to interrogate aerodynamic bodies as if they were standing within a wind tunnel. This application uses a Fakespace BOOM® as a visualization device and a VPL Data Glove as an input device, which is position tracked using a Polhemus tracker. The VWT can display a variety of flow entities including streamlines, streaklines, pathlines, rakes,

isosurfaces, and cutting planes. Particle advection is also supported using a second order Runge-Kutta scheme. The second order scheme was chosen over the more accurate fourth order scheme because it requires only two trilinear interpolations per time step whereas the fourth order version requires four trilinear interpolations per time step. This design choice was necessitated by the computationally time consuming nature of trilinear interpolation.

Users interactively place entities, such as a rake, within the flow field using grab points located on the entity. These grab points are used in conjunction with the VPL Data Glove to manipulate and orient the entity. Once the flow entity is placed, the result is calculated and displayed in the virtual environment. In addition, the VWT contains a hierarchical virtual menu that could be displayed in the virtual environment to allow users to select the flow entities that they wished to place within the model. This menu contains sliders that are manipulated by the Data Glove so that users can adjust flow entities once they were placed. A user superimposed upon the virtual environment is shown in Figure 2.3.

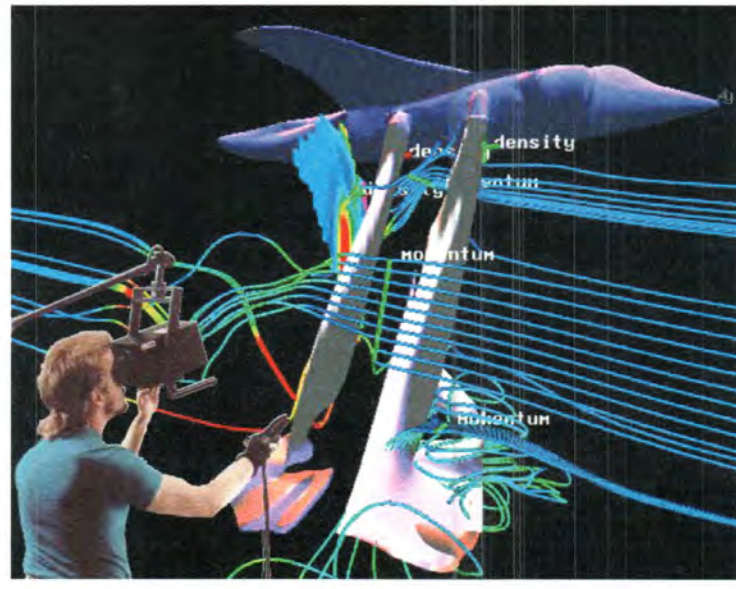


Figure 2.3 - The Virtual Wind Tunnel. Image courtesy of NASA Advanced Supercomputing Division.

When Bryson, et al. (Bryson et al., 1995) surveyed users, they noticed that the VWT seemed to be a natural interface for examining CFD datasets. Users responded well to the Data Glove interface, and were generally excited about using the new technology for CFD analysis.

Later, Bryson and Yamasaki (Bryson et al., 1992) developed and demonstrated a distributed version of the VWT. The distributed VWT utilized a four processor Convex vector supercomputer for computing particle paths and a Silicon Graphics (SGI®) 380GT VGX system containing eight 33 MHz processors for displaying graphics. This system was used to demonstrate the increases in performance available by performing computations on one dedicated computer and performing graphics computation on another dedicated computer. The distributed VWT also contained provisions for more than one visualization computer connecting to the supercomputer. In this way two users could simultaneously explore the data sets at a moderate performance penalty.

The VWT was able to handle the visualization of unsteady flow datasets as long as the data files from each time step were able to fit into the system memory. Another limitation was the fact that the data files had to be preprocessed to fit into one data file, as opposed to one file per time step. This provided an upper bound on the number of time steps that could be visualized in a unsteady flow. This was also ineffective for datasets that contained thousands of timesteps, as it was impossible to pre-process such datasets effectively (Lane, 1993). The unsteady flow analysis toolkit (UFAT) (Lane, 1994), developed by David Lane, mitigated some of these problems. UFAT was able to handle larger datasets by holding only the two most recent time steps in memory. However, UFAT

was not designed to be interactive in nature. Rather, UFAT would write a data file for users to view with the Flow Analysis Software Toolkit (FAST) to examine the resulting data (2002). FAST was later ported to VR for visualizing scientific data, but does not provide the interactive performance necessary for analysis of several thousand particle traces (Lane, 1995).

2.5.2 Commercial Packages

The following section reviews the currently available commercial packages that integrate VR and CFD analysis.

2.5.2.1 Ensight and Fieldview

Ensight™ and Fieldview™ are two popular commercial CFD post-processing applications. Both have strong user bases in the traditional CFD visualization domain, and are used in both academic and industrial research environments.

In early 2002, Computational Engineering International, Inc. introduced the Ensight Gold™ package. Ensight Gold™ contains support for surround-screen virtual environments and for 6 degree of freedom input devices, such as a wand. A wand is a remote control-like device that is position tracked and has buttons that can be used to give discrete input to an application. Participants can use the wand to place entities within the flow field and scale, rotate and translate the system geometry. However, no menuing support within the virtual environment is currently built into Ensight Gold™. Ensight Gold™ utilizes multiple processors on shared memory computers for the calculations of a variety of entities, including isosurfaces and multiple streamlines. All the features available in the desktop

version of Ensight Gold™ are currently implemented in the VR version of Ensight Gold™; however, interaction within the virtual environment is limited, and the majority of the program controls are still activated from the desktop.

In late 2001, Intelligent Light, Inc. unveiled a VR version of its popular post-processing software, Fieldview™. Fieldview™ is a general CFD postprocessor that provides a variety of features including streamline and isosurface calculation, as well as limited particle tracing capabilities. Fieldview™ VR was designed to be used in surround-screen virtual environments. It utilizes a wand for translation, rotation, and scaling, but all other controls are still based on a desktop menu. Interaction in the virtual world is limited because of this fact.

2.5.2.2 EXA

In 1999, BMW AG and EXA AG, maker of the lattice-based CFD code PowerFLOW™, developed a visualization application for virtual environments that performed particle tracing on multilevel cartesian grids (Schulz et al., 1999). Because of the multilevel nature of the grid, past particle tracing environments required extensive modification before being adapted for the application. This required that special search methods be developed for cell location and interpolation. This application utilized second and fourth order Runge Kutta methods with adaptive stepsize control for increased performance. The application that was developed was able to demonstrate interactive performance for approximately one hundred particles, with the tracing time varying from 1.67 seconds at the optimal case and up to 3.12 seconds in the worst case.

2.5.2.3 IVRESS

Advanced Science and Automation Corporation in conjunction with Old Dominion University developed IVRESS, or Integrated Virtual Reality for Synthesis and Simulation, for visualization of CFD results in surround-screen VR systems (Wasfy et al., 2001). This application supported the visualization of a variety of flow entities including stream lines, stream ribbons, surface arrows, isosurfaces, and vortex cores. This application was primarily designed for aeronautics applications and wind tunnel testing of prototype aircraft. The application is built in an object-oriented manner, and has the ability to read in a great variety of data file formats. Unlike many other commercial CFD applications that support visualization in VR, IVRESS contains numerous interaction capabilities. It application supports speech recognition (Wasfy et al., 2002) and utilizes a palmtop computer for menu interaction while in the virtual environment.

2.5.2.4 Virtual Vantage™

Nalco Fuel Tech, Inc. has developed a system to simulate particle dynamics in non-coupled two phase flows using an analytic method developed for linear tetrahedra (Diachin et al., 1996). They have demonstrated that this method is more accurate than a fourth-order Runge-Kutta method and is faster than the forward Euler method. This software, named the TrackPack toolkit, simulates massed, massless, and evaporating particles. The TrackPack toolkit development was primarily motivated by modeling of combustion in industrial boilers. This software is linked to a visualization application, named Virtual Vantage™, by a message passing library. This distributed software architecture allows for the visualization

and computation to be run on separate computers. Virtual Vantage™ then displays the results calculated from the TrackPack toolkit on surround-screen VR systems. Users can interactively place stream elements, such as streamlines, or place arrays of particles within the fluid domain and visualize results (Diachin et al., 1998). Virtual Vantage makes use of a virtual menuing system and the wand for interaction (Heath, 1998). It is reported that each streamline requires on the order of approximately 0.2 seconds for calculation and visualization (Diachin et al., 1998).

2.5.4 VR-CFD

Kutti and Vance (Kutti, 1998) at Iowa State University developed a VR application for displaying CFD results, named VR-CFD, which enabled users to place a variety of elements within the flow field, including stream lines, rakes, isosurfaces, vector fields, and particle animations. This application uses a static hierarchical menu on one of the walls for feature selection, and a position tracked wand for interaction within the virtual environment. Selecting menu buttons brings forth additional menus. A user interrogating a fluid system is shown in Figure 2.4.

VR-CFD reads in Plot-3D multiblock grids and uses VTK (Martin et al., 1998) to extract features from the flow field. VTK was chosen because of the wide variety of flow visualization functions that it provides. However, it was noted that VTK did not provide speed necessary for real time interaction and display of results. In related research, Shahnawaz et al. (Shahnawaz, 2000) examined methods to approximate results of a geometry change between two solved CFD datasets. This research resulted in being able to modify the geometry of the system and view changed results interpolated within the virtual environment.

Two different interpolation methods were investigated, simple linear interpolation and a B-spline fitting scheme. However, each interpolation scheme required multiple solutions of the underlying dataset and the accuracy of the interpolation was closely related to the complexity of the flow field.

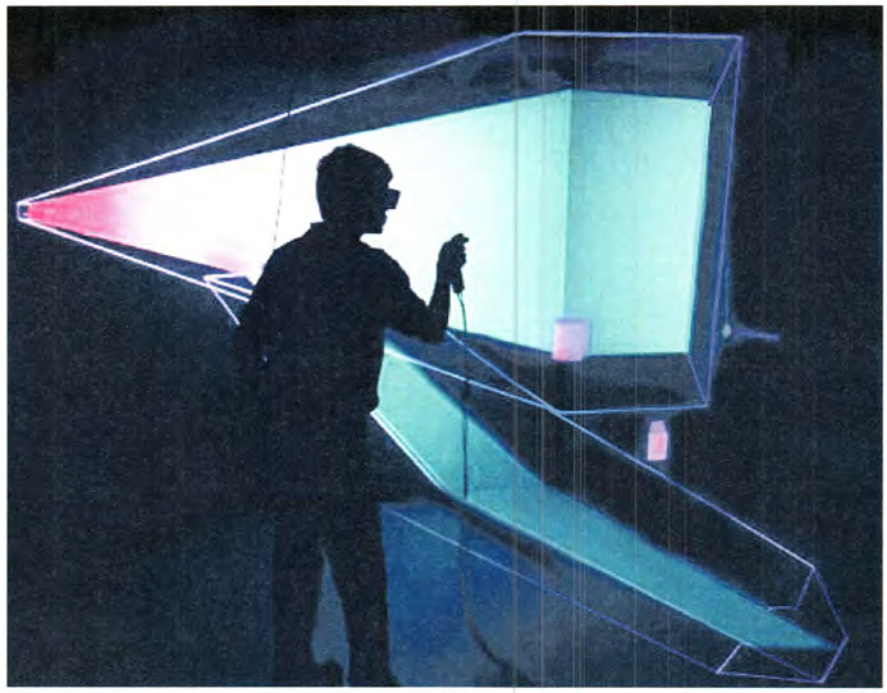


Figure 2.4 - VR-CFD Application. Image courtesy of Virtual Reality Applications Center.

2.5.5 VR-XPR

Recently, researchers at Iowa State University developed a rapid prototyping environment called VR-XPR (Virtual Reality Explorer) (Chu, 2001). VR-XPR was built using VTK and supports the visualization of stream lines, stream ribbons, isosurfaces, contour surfaces, and vector fields. The application contains a static 3D user interface similar to that used by VR-CFD. VR-XPR provides support to read in CFD data from Star-CD™, Fluent™, and Plot3D™ data formats. One of the primary advances VR-XPR made

over previous CFD visualization environments was the use of shared memory parallelization techniques to provide faster calculation of visualization results.

2.6 In Core versus Out-of-Core Visualization Methods

The particle tracing methods included in the surveyed VR applications are exclusively in-core methods. In core methods generally follow a three step process. First, the underlying CFD data set is read into memory. Next, the initial particle position is given. Finally, the velocity of the particle is interpolated and the new position is integrated through the use of an appropriate numerical method. This approach tends to be computationally intensive in terms of processor time and memory needed, especially for time varying flow fields where a new data file must be loaded for each respective time step. Few applications currently have shown that this can be done for a significant number of particles while maintaining an interactive frame rate, generally defined as between 30 and 60 frames per second.

Another approach to particle tracing is to use an out-of-core method (Bruckschen et al., 2001). In out-of-core particle tracing methods, a uniform emitting volume is chosen by the user and a dense grid of particle traces is pre-calculated from that volume. These results are stored in a set of data files that can be quickly searched during later data exploration. Within the virtual environment, the user interactively selects the area from within the pre-computed volume of emitters where he would like to see the particles emitted from. Next, the out-of-core algorithm loads the necessary data files reads in the needed particle positions. The particles are continuously released from these seed positions, and their positions are continuously read in from the data files. Bruckshen et al. (Bruckschen et al., 2001) have

shown that interactive frame rates can be maintained for a large number of particles using this method. They report that for 64 emitter positions, with continuous particle emission, they achieve a maximum number of particles on the order of 60,000 while maintaining an interactive frame rate of sixty frames per second. In their work, they have shown that tracing on the order of 400,000 initial particle locations gives them this capability. However, this work requires a substantial amount of disk space. In the work of Bruckshen, et al. (Bruckschen et al., 2001) it required a 120 GB RAID (Redundant Array of Inexpensive Disks) system.

While out-of-core methods may be advantageous for future real-time mixing simulations, for this work the disk space requirements are currently unfeasible. In addition, since the number of initial particle positions to be traced in mixing simulations is frequently 25,000 – 100,000 particles, a much denser initial grid would need to be traced, making storage requirements even more prohibitive.

2.7 Conclusions

This chapter has summarized work done for visualization of CFD simulations in VR. A brief summary of the pertinent concepts from CFD and the use of CFD for mixing evaluation has also been presented. An introduction to the HyperTrace™ mixing software has been given, which will be described in more depth in the following chapters.

The survey of published literature provides several conclusions that can help guide this research. First and foremost, interaction methods are of paramount importance within the virtual environment. The user needs to be able to interact easily with the data in the virtual environment, and this can be achieved primarily through astute choices when

designing the user interface. Secondly, nearly all applications have looked towards a distributed hardware environment to achieve interactive performance. Finally, while out-of-core methods look promising, they are not yet suitable for this research.

Since this application focuses solely on particle tracing, the implementation of the VR application can focus on this one operation. This makes HyperTrace™ a good choice as a starting point for this research. HyperTrace™ also has the advantage of specifically focusing on particle tracing, whereas all other applications surveyed focused on a variety of data examination schemes.

The software and hardware architecture of the VR implementation of HyperTrace™, hereafter known as VirtualTrace, is discussed in chapter 3. Interaction methods are discusses in chapter 4 and 5, and an example application is shown in chapter 6.

CHAPTER 3. VIRTUALTRACE DESIGN

This chapter contains an overview of the issues pertinent to the design of an interactive virtual environment. This chapter begins with a discussion of the distributed hardware environment chosen for this research, and then discusses the programming paradigms used to interface with this hardware. This is followed by a summary of parallel programming methodologies and their relation to this research. This chapter concludes with a thorough discussion of the performance characteristics of the application that was developed.

As discussed in chapter 2, few applications provide the capability to trace the thousands of particles necessary to gain a thorough insight and understanding of a mixing process. HyperTrace™ is the only currently available software providing the necessary performance for near interactive particle tracing. However, even HyperTrace™ requires extensive modifications to enable interactive computation of particle traces within virtual reality. Specifically, the visualization program must be completely rewritten to use VR technology and the particle tracing program must be modified to run on distributed hardware. In addition, it is necessary for the particle tracer to communicate particle positions to the visualization program over a network rather than through a file. These software modifications are discussed in this chapter and the interaction modifications are discussed in chapters 4 and 5.

While a variety of design issues must be considered in the design of a virtual environment, Bryson (Bryson, 1996) outlined three critical areas of interest to designers of virtual analysis of CFD datasets. These are:

- Data Management
- Computation
- Graphics

Data management refers to the processes primarily associated with the loading and handling of the CFD data sets. Because of the massive size of many CFD datasets, memory management is critically important if interactive performance is to be maintained. Gerald-Yamasaki (Gerald-Yamasaki, 1997) has extensively examined the two primary methods: prepayment or pay-as-you-go. In his study, prepayment is defined as loading all necessary data files at start-up and taking all data access costs at the start of a visualization program. On the other hand, pay-as-you-go methods load data on an as needed basis. The choice of data management method is highly problem dependant. Because of the size of the data sets used for this research, prepayment methods showed superior performance and were used exclusively.

Computation refers to the calculation of analysis results such as streamlines, isosurfaces, or particle traces. Efficient algorithms for computation are essential to maintain interactive frame rates in a virtual environment. To ensure high performance computation, a distributed hardware environment has been chosen, which is an approach commonly used for the visualization of CFD datasets.

Finally, Graphics refers to the use of fast routines to display graphics to the VR display devices. To ensure fast graphics display, a graphics API that supports hardware rendering to speed graphics computation is chosen. Fast graphics computation is critical to ensure that the frame rate stays at interactive levels, which is generally accepted to be greater

than 30 frames per second. If the frame rate slips too far below 30 frames per second, users can experience cyber-sickness, a sensation similar to motion sickness induced by computer generated images, and other undesirable effects, such as image artifacts.

The utilization of appropriate data management, computation, and graphics programming methods are primarily software design choices. To better understand our software design, a through understanding of our hardware is necessary. Section 3.1 discusses the characteristics of the hardware utilized for this research. After summarizing the hardware, discussion of the software design parameters is included in section 3.2. In section 3.3 an overview the operations of the application developed is presented. Finally, in section 3.4 performance results for the application developed are presented.

3.1 VirtualTrace Hardware Framework

It is readily apparent from the survey of literature in chapter 2 that a primary design feature of CFD visualization environments is the use of distributed hardware along with parallel programming methods. Separating the graphics processes from the computation processes enables the use of specific hardware for each process. This hardware can be chosen based on the different characteristics necessary for fast display of graphics and fast numerical computation. The graphics computation component of this research project utilized the C4 and the C6 VR systems present at Iowa State University's Virtual Reality Applications Center. Although the primary operating environment of this research is the C6, a discussion of the C4 virtual environment will be included because of its similarity. The particle tracing computations will take place on a Linux® cluster of multiprocessors (CLUMP).

3.1.1 Graphics Hardware Overview

The C6 is a six-walled surround-screen system with six 10 ft. by 10 ft. walls and a wireless tracking system. One of the six walls on the C6 slides open to allow users to enter and exit the C6. The C6 is driven by a SGI® Onyx 2® computer system. The Onyx2® system has 24 R12000 processors running at 400 MHz, 12 gigabytes of RAM, and six InfiniteReality2™ graphics engines. All walls of the C6, including the floor and ceiling are back projected. The C6 also utilizes synchronized Barco projectors to generate the six stereo images on the display walls. A picture of the C6 is shown in Figure 3.1. This image shows the cubic structure of the device and the placement of the six projectors used to project images. The mirrors used to reflect the floor and ceiling images are also visible.

The C4 is a four-walled surround-screen system with a wireless tracking system driven by a 12 processor SGI® Onyx® computer. The Onyx® system contains contain 12 195 MHz R10000 processors, four InfiniteReality™ graphics engines, and one gigabyte of RAM. It has three 12 ft. by 9 ft. projection screens and a 12 ft. by 12 ft. floor. The C4 is an MD Flex system constructed by Mechdyne, Inc. The MD Flex system can be reconfigured by rotating the two sidewalls to produce one flat 36 ft long display wall. Graphics are back projected onto the three sidewalls and front projected onto the floor. Synchronized Barco 909 analog projectors display stereo images to the four walls. An image of the C4 is shown in Figure 3.2.

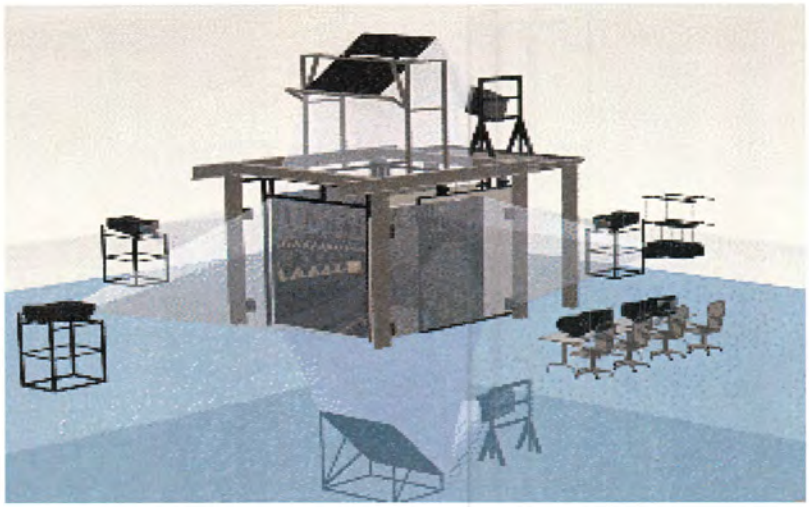


Figure 3.1 - C6 Virtual Environment. Image courtesy of Virtual Reality Applications Center.

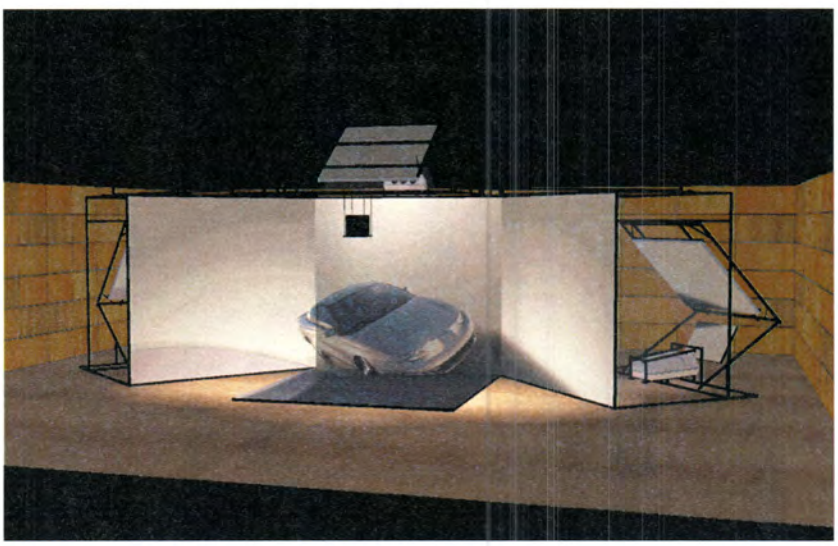


Figure 3.2 - C4 Virtual Environment. Image courtesy of Mechdyne, Inc.

Users within both systems 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. A

pair of shutter glasses is shown below in Figure 3.3. The emitter that synchronizes the glasses is to the left of the glasses in the image.

Figure 3.3 - CrystalEyes® shutter glasses. Image courtesy of StereoGraphics, Inc.

These images are rendered at a resolution of 1024 x 1024. To give users the correct perspective view, one user's head position is tracked by a wireless magnetic tracking system. This system can give the position and orientation of any tracked object within the virtual environment. An Ascension Flock of Birds wireless tracking system is shown in Figure 3.4. All images within the virtual environment are drawn based on the position and orientation of the tracked user's head.

Figure 3.4 - Ascension Flock of Birds Motion Tracking System. Image courtesy of Ascension Technology, Inc.

3.1.2 Distributed Architecture Overview

Traditionally high performance computing applications have been executed on large mainframe computers. While these computers have shown excellent performance, they are

prohibitively expensive. The current trend in parallel computing favors using clusters of multiprocessors (CLUMPS). In 1997, of the five hundred fastest computers, over sixty percent were scalable parallel processors (Dowd et al., 1998). Another factor in the increased usage of cluster is that the majority of off the shelf PC architectures are dual processor capable, making them inexpensive candidates for nodes in a distributed memory cluster. CLUMPs present an interesting programming challenge and an intriguing opportunity for large computational problems. Each of the nodes is distributed, requiring the use of a local area network to connect the nodes, but within each node multiple processors share memory on a system bus. This provides the opportunity for the use of hybrid programming methods, combining methods from shared memory programming paradigms along with methods from distributed memory paradigms. Each node in the cluster is a symmetric multi processor (SMP), a computer where each processor has access a single main memory.

In this research, a Linux® Cluster containing 8-nodes will be used for the calculation of particle traces. Each node contains two AMD Athalon™ processors running at 1.2 GHz and contains 896 megabytes of system memory. A gigabit Ethernet network connects the nodes. This yields a 16-processor cluster. The system topology is shown in Figure 3.5.

Because of the architecture of CLUMPs contains both distributed and shared memory, two programming paradigms may be used. Each of the nodes is distributed, requiring the use of a message-passing library, such as MPI (Snir et al., 2001). However, between the processors on each node the memory is shared on a system bus.

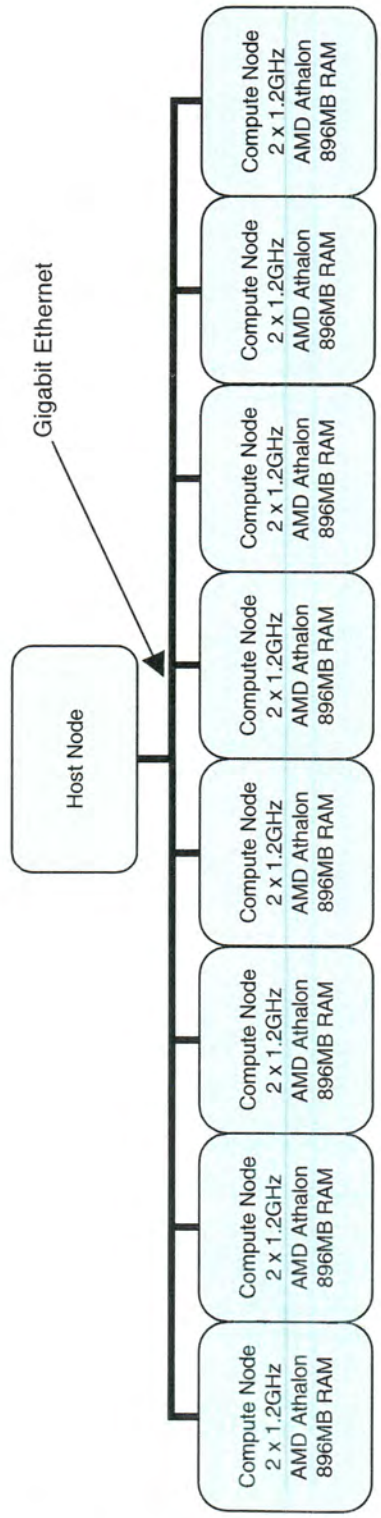


Figure 3.5 - Linux® cluster architecture

This allows for the use of two programming methods, either utilizing this shared memory within the nodes by means of OpenMP (Dagum et al., 1998) or some other shared memory language, or alternatively by running two independent MPI processes within each node. These two programming methods are shown graphically in Figure 3.6. This figure depicts two compute nodes. On the compute node on the left, two MPI processes are run, one on each processor. This is the MPI-only parallel programming paradigm. However, on the compute node on the right, one MPI process is run and one OpenMP thread is run on each processor. This is the hybrid programming paradigm.

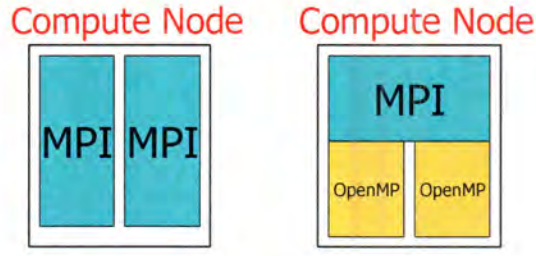


Figure 3.6 - Schematic view of MPI only programming methodology and hybrid programming methodology for a compute node

Current research in high performance computing has failed to conclusively answer the question of which of these methods is better suited for application development on a CLUMP. Hybrid code refers to using MPI for communication between nodes and OpenMP for communication between processors on a node. The advantages of a hybrid code as opposed to an MPI-only code on a CLUMP has been summarized by Boku et al. (Boku et al., 2001) as having the following characteristics:

- Low Communication Costs: Communication within a node via OpenMP is extremely low cost. In fact, communication within a node can be achieved without the use of an explicit function call.
- Dynamic Load Balancing: Since OpenMP threads are dynamically spawned during the tracing of each time step, they are load balanced automatically for each time step, as opposed to being statically set.
- Coarse Grain Communication: Since the number of communicating nodes is reduced, the number of communications is also reduced, resulting in less communication overhead.

Capello, et al. have shown that the choice of memory model is strongly application dependant by comparing hybrid and distributed memory models of the NAS parallel benchmarks (Cappello et al., 1999; Cappello et al., 2000). In their work some of the parallel benchmarks were faster with hybrid methods and some were faster with MPI-only methods.

Because of the nature of particle tracing, it can be parallelized quite obviously. The particle tracing application can be divided into three distinct categories including, (1) reading in of the geometry and solved velocity fields at the nodal points, (2) parallel tracing of particles, and (3) writing the data file for use by a visualization package or by sending the results directly to an application over a TCP/IP socket. Parts one and three are inherently serial, while part 2 is inherently parallel because the particles do not interact with each other.

Since the only information any particle needs is information of its previous location, parallelization based on particle traces is an obvious choice. In this method, the only communication required is the gathering of the particle positions from each distributed node to one node for storage at the end of each time step. However, since the particle position

cannot be predicted in advance, parallelization by this method has the unfortunate side effect of requiring every processor to be able to access the entire solved CFD grid. For large unsteady datasets, this can be problematic. Another option, explored by Malevsky, et al. (Malevsky et al., 1992) is to distribute the grid evenly between the processors and pass the particles between the processors as the particles move from one cell to the next. However, the performance of this method suffers if the particles are not evenly distributed over the grid. Since the initial position of particles will rarely meet this condition, the early steps of the simulation would exhibit poor load balancing. Additionally, this method would require extensive communication, and could potentially saturate the gigabit Ethernet between the nodes. For this reason, the code has been parallelized by gathering the particle positions at the end of each time-step.

3.2 VirtualTrace Software Framework

This section will discuss the software tools and design considerations used to create VirtualTrace. The rest of this chapter will focus primarily on the visualization and computation aspects of VirtualTrace, whereas chapters 4 and 5 will focus on the interaction methods.

While many VR CFD visualization applications surveyed in chapter 2 provide several visualization tools such as isosurfaces, streamlines, and vector fields, VirtualTrace will focus solely on the visualization of particle traces. The computation of particle traces should be quick enough to allow interactive design and to provide tools to interact with large numbers of particles.

In order to provide interactive performance, the two previously uncoupled parts of HyperTrace™, the particle path computation and the graphics display, must be coupled. This means that the particle tracing engine must send the particle positions to the VR application immediately after they have been computed, as opposed to the current output method of writing them to a data file for later viewing. Furthermore, the graphical viewer must be completely rewritten in order to take advantage of the immersive VR hardware. Both of these tasks were accomplished as part of this research. An application level schematic of the two programs making up VirtualTrace is shown in Figure 3.7.

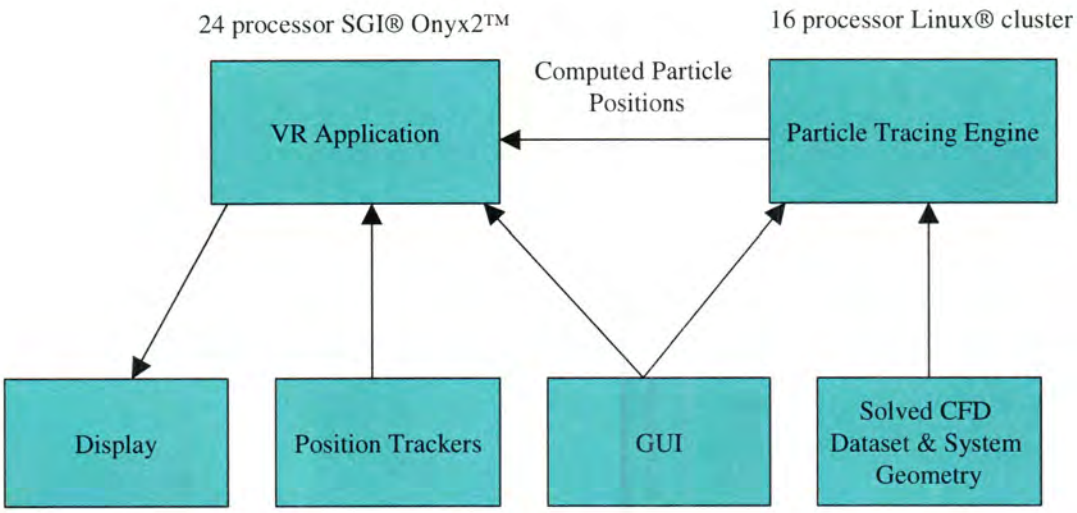


Figure 3.7 - VirtualTrace Application Schematic

The two applications shown in the Figure 3.6 represent the main computational components of VirtualTrace. The VR application is responsible for visualization, interacting with the user's input, and performing the various data interrogation tasks. On the other hand, the particle tracing engine is responsible for reading in the solved CFD dataset and the corresponding geometry and setting up all necessary data structures for the particle tracing.

Both applications interact with a graphical user interface (GUI), which is described in Chapter 4, and is used to control the application functions.

3.2.1 Design Considerations

The distributed nature of VirtualTrace and the differing hardware architectures of the machines require that the particle-tracing engine and the VR application have significantly different software design requirements. Because of this, their design factors are discussed separately. However, since they are executed at the same time, their usage is summarized together. Care was exercised during the software development to utilize open-source or open-standard pieces of software. This ensures that the software will be portable between different hardware architectures.

3.2.1.1 VR Interface Design

The original visualization software for HyperTrace™ had to be completely rewritten to take advantage of VR. For the VR application, the following software tools were chosen:

- C++

  C++ is a widely used and extremely powerful programming language. It is completely object-oriented, making software development and debugging quick and robust.

- VR Juggler

  VR Juggler (Bierbaum, 2000) 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,

Powerwalls, and BOOM® systems. This allows any application programmed using the VR Juggler API to run seamlessly on a variety of VR systems, enabling tremendous flexibility. A recent addition to VR Juggler, Cluster Juggler (Olson, 2002), makes it possible to utilize clusters of commodity PC's to run VR Juggler applications within a CAVE™ environment. As hardware prices continue to decline, and increased computer power becomes available on desktop PC's, 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 PC's with high performance graphics cards. This would provide a substantial cost savings and enable VR to reach a far wider user community. VR Juggler enables this application to be moved to PC clusters for inexpensive visualization. Finally, there are versions of VR Juggler for a variety of operating systems including Windows®, UNIX®, and Linux®.

- OpenGL

  OpenGL (Woo et al., 1997) is a platform independent, open-standard graphics API. While OpenGL is a very low level graphics API and does not directly support advanced features like scene-graphs or model loading, its features can be utilized to achieve very fast graphics. In addition, OpenGL hardware acceleration is a common feature in a variety of computer architectures, making OpenGL a solid choice for graphics programming.

- Pthreads

  Pthreads (Nichols et al., 1996) are used to create separate threads for specific tasks within the VR application. This essentially means that a separate process is started

during runtime. Once this thread is spawned, it can communicate back to the original application. In particular, separate threads are used to poll for input on TCP/IP sockets, which are network connections between separate programs (Stevens, 1998) so that any data received on a socket is immediately sent back to the application for processing.

Based on these programming tools, new visualization software was written. The new VR application can be executed in either of two modes: interactive mode or playback mode. In the playback mode, the particle traces have been pre-computed and stored in a binary file for later viewing. In interactive mode, the particles are computed interactively on external hardware. In general, both modes follow the same series of steps at startup. These steps are covered in more depth in section 3.3.

3.2.1.2 Parallel Numerical Method Design

The particle tracing application presents a different challenge from the VR application. Since the primary design concern is high performance for numerical computations, software tools were chosen to enable high speed. To achieve the highest speed possible, the following software tools were chosen:

- C

  While C is not object-oriented, it enables code to be developed modularly, which means that code can be developed and tested in small blocks. C also contains significantly less overhead than C++, which makes C run substantially faster than C++ for scientific applications.

- MPI

  The Message Passing Interface (MPI) (Snir et al., 2001) provides an API for parallel programming on distributed memory computers. MPI is an open-standard, and free versions of MPI are available. In addition, many vendors provide optimized versions of MPI for their specific hardware. MPI provides an extensible API that includes tools to handle efficient communication of data between the distributed nodes of the computer. The version of MPI that this program uses is MPICH, (Gropp et al., 1996; Gropp et al., 1996). MPICH is a free implementation of MPI designed for clusters of homogenous PC's. MPI is used to handle communication between the nodes of the cluster.

- OpenMP

  OpenMP (Chandra et al., 2000; Dagum et al., 1998) is an open-standard for shared memory parallelization. OpenMP is used on machines where more than one processor has access to the memory by a system bus or an interconnect. OpenMP is used here for communication within each node of the cluster.

These tools were used to parallelize the existing particle tracing code for distributed memory systems.

Using the aforementioned programming tools and the parallel programming methodology outlined in section 3.1.2, the existing particle tracing engine was rewritten. First, the code was parallelized. MPI versions and hybrid MPI and OpenMP versions were developed in order to compare the performance of these codes. This is discussed in section

3.4. Next, the code was modified to send the newly computed particle traces over a TCP/IP network connection rather than write them to a binary file. Finally, the code was modified to allow the user to place the particles by using the wand within the virtual environment, which will be discussed in more detail in chapter 4. Figure 3.7 shows how the software components of VirtualTrace form one cohesive application and which applications run on each separate computer. The left hand side of Figure 3.8 shows the improved HyperTrace™ solver that was rewritten using MPI and OpenMP as part of this research and is run on the Linux® cluster. The right hand side shows the VR interface developed for VirtualTrace that is run on the SGI® Onyx2® computer system. Both programs interface with a GUI that will be discussed in Chapter 4.

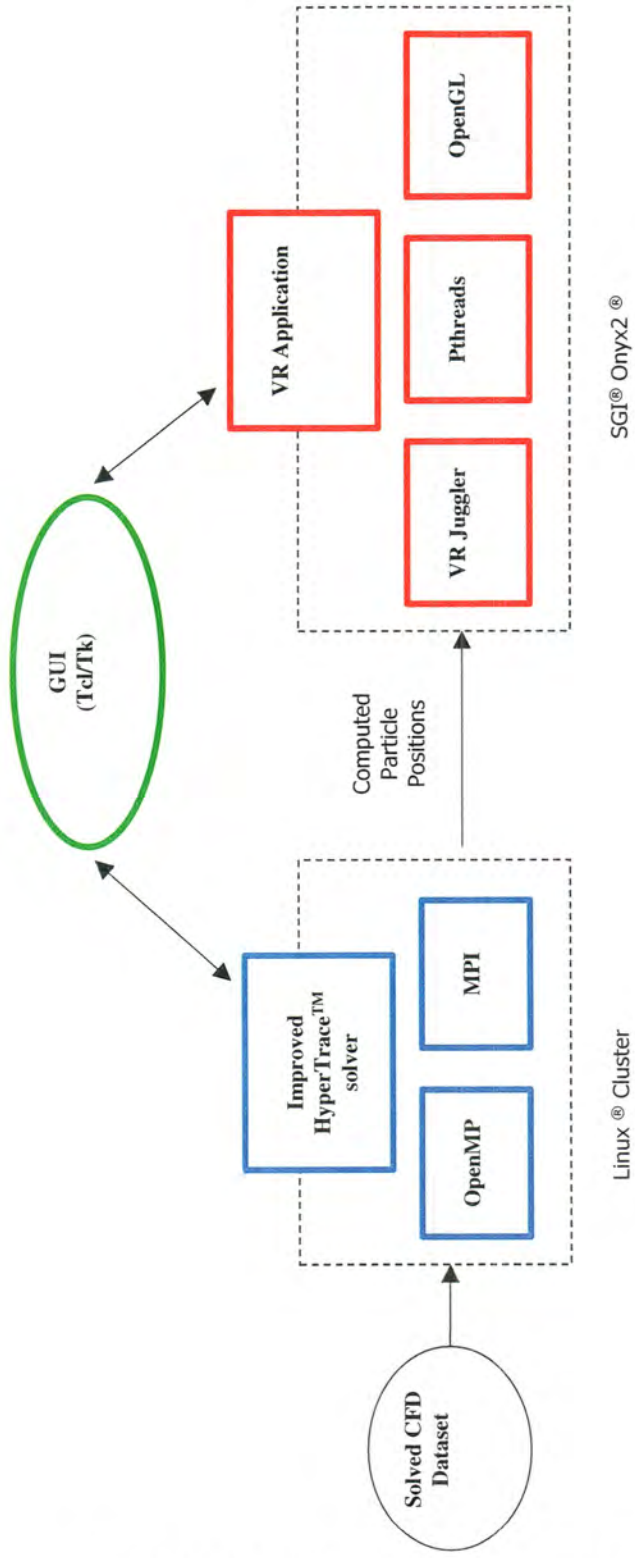


Figure 3.8 - VirtualTrace application overview

3.3 Modes of Operation

The visualization software can operate in either of two different modes: a playback mode or a interactive mode. The playback mode is where the particle traces have already been computed and are being read from a data file, whereas the interactive mode is where the particle positions are being computed on the fly on a separate cluster computer connected to the VR application over a network. Both modes of operation utilize a graphical user interface (GUI) and a speech recognition program for user interaction. These interaction methods are discussed in more depth in chapter 4.

3.3.1 Playback Mode

In playback mode, the visualization software acts independently, and the particle traces have been pre-computed and written to a binary file. Therefore, the visualization application first reads in a data file that contains both the geometry of the fluid system and the particle positions.

The first thing that happens in playback mode is the creation of a network connection between the VR application and the GUI. Next, another network connection is created to interface with the speech recognition software, if the speech recognition capability is requested. Finally, 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 data read in, the necessary data structures are populated and the geometry is loaded into a display list. After all of 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. These are described in chapter 5. The startup processes for the VR application are listed in Figure 3.9.

VR Application

1. Connect client socket to GUI
2. Connect client socket to speech recognition program
3. Open geometry file
    1. Create display list of system geometry
    2. Load in all particle positions
4. Display particles at users request

Figure 3.9 - Program operations during playback mode

3.3.2 Interactive Mode

Figure 3.10 lists the processes involved in interactive mode. In interactive mode, users first start the VR application. Then, the VR application waits for a network connection from the particle tracing software. Next, the user starts the particle tracing software on the distributed hardware. The particle tracing software then connects to the VR application over the local network. Next, the VR application opens a network connection to the GUI. Meanwhile, the particle tracing software loads in the solved nodal grid, the velocities, and the fluid system geometry. After the particle tracer has read in the solved data set, it writes out a binary file containing the system geometry. This binary file contains a complete description of the boundary geometry. A file is written, rather than the use of the socket, because of the quantity of information required to be transmitted between the two programs. After the binary file is written, the VR application loads the boundary geometry from the file that has

been written. This file is stored on the local file server, which is mounted to both the SGI® Onyx2™ and the Linux® Cluster via the network file system (NFS). The polygons from the binary file are read in and loaded into a display list for hardware-accelerated drawing. Both applications then go into a wait state. The user interactively places particles into the fluid system, using the wand, and controls the simulation parameters with the GUI. When the user starts the simulation, the particle tracing application begins to trace particles and notifies the VR application that it should expect particle positions. The VR application now waits for the particle-tracing engine to send it a packed buffer of particle positions. After the particle paths have been computed for a given time step, the particle tracer packs them into a buffer and sends this buffer to the VR application over the previously established network connection. The packed buffer contains an integer description of the particle positions. This packed buffer is then unpacked into the corresponding particle positions by the VR application. These particle positions are subsequently displayed in the virtual environment. This final step is repeated until all of the time steps have been received from the particle tracer. Once all of the particle traces have been received, the VR application functions in the same manner as if it were in playback mode, allowing the user many options to explore the data. However, when the VR application is receiving particle positions from the particle tracer, the analysis tools are not available.

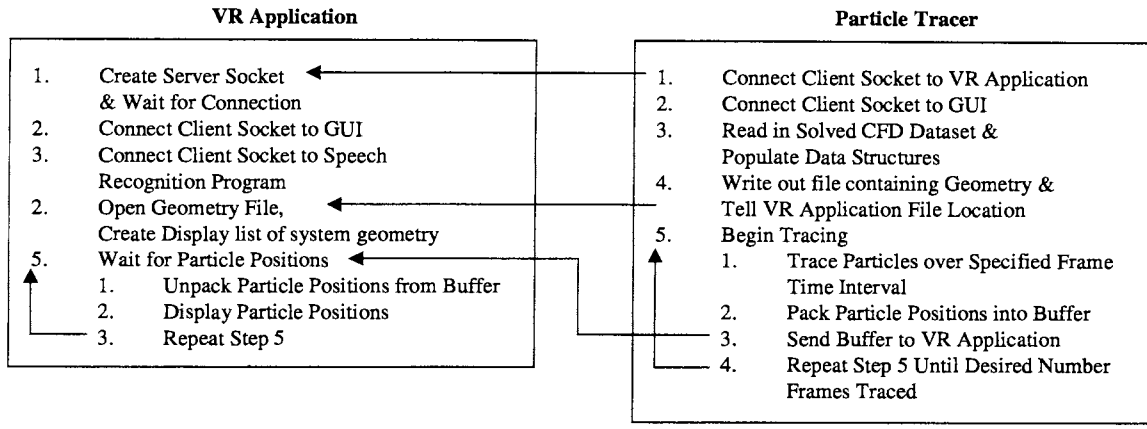


Figure 3.10 - Program operations during interactive mode

3.4 Performance Results

Performance testing of VirtualTrace took place on an 8-node Linux® cluster described in section 3.1.2. Four different solved CFD data sets of mixing vessels were used for performance evaluation. Data sets were provided by The Procter & Gamble Company as data sets that are indicative of typical industrial processes. The data set characteristics are shown below in table 3.1. Dataset four is significantly larger than the other sets because it is an unsteady flow and therefore has to load a separate velocity file for each of the 40 time steps simulated.

Table 3.1 – Data files used for performance testing

	Number of cells	Number of time steps	Size
Dataset 1	2,500,000	1 (steady)	130 MB
Dataset 2	1,500,000	1 (steady)	75 MB
Dataset 3	785,000	1 (steady)	60 MB
Dataset 4	875,000	40 (unsteady)	1.8 GB

These solved data sets were generated using Fluent™ and are used to compare the performance of the hybrid (MPI and OpenMP) particle tracing code and the MPI only particle tracing code. Each data set was run serially, and then with the MPI only code and the hybrid code. This resulted in a total of 260 trials. Each data set was tested by tracing particles for 100 time steps for 5000, 10000, 25000, and 50000 particles, respectively. Particles were given the same initial positions for each trial.

Previous work analyzing the performance of a conceptually similar particle tracing code in hybrid and MPI only versions was performed by Boku et al. (Boku et al., 2001), Smith (Smith, 1999), and Majumdar (Majumdar, 2000). In the work of Boku et al., the hybrid code did not perform as well as the MPI only code. The reason for the performance gap was CPU data cache misses due to irregular data access patterns in OpenMP. Smith's hybrid application also failed to perform as well as the MPI only code. Her work came to the similar conclusions to the work of Boku et al. In addition, Majumdar's work also found the MPI only code to be superior to the hybrid code. Majumdar attributes the performance difference between the codes to compiler inefficiencies when compiling OpenMP code. Despite these results, hybrid programming methods are an actively researched topic because of their potential for increased performance, as was discussed in section 3.1.2.

The primary measure used for performance analysis of a parallel program is speedup. Speedup is the measure of the performance increase by running a job with N processors as opposed to the performance on a single processor. Speedup is calculated by dividing the runtime of a code which uses a single processor by the runtime of that code on N processors. If the value of the speedup is equivalent to the number of processors used, then the program

is said to have linear speedup or, analogously, ideal speedup. Linear speedup is rarely achieved, however, due to communication overhead, data access time, and other factors.

Results are shown in Figures 3.11, 3.12 and the Appendix for each of the time trials. These graphs show several trends. First and foremost, for the problem sizes studied and the number of processors available, the MPI only code has superior performance in nearly all cases. This corresponds well with other published research. However, in several cases the MPI only code shows a significant performance drop when more than 10 processors are used, while the hybrid code shows no significant performance drop in any cases. In fact, in 10 of the 16 trials, if the current trends were extrapolated to additional processors, the hybrid code would show superior performance. Figure 3.11 depicts this trend. However, Figure 3.12 depicts the trend for the other 6 trials which shows no presence of performance drop in the MPI only code and shows a significant performance gap between the MPI only code and the hybrid code. It is believed that the 6 cases not currently showing this trend would begin to exhibit it with additional processors. This suggests that the hybrid code might achieve higher speedup than the MPI only code if it were scaled to more than 16 processors. This correlates to the number of MPI processes at which communication between the nodes becomes a bottleneck. The second conclusion that can be drawn is that if the entire problem domain cannot be fit into main memory, then the hybrid version of the application is superior. This is visible in Figures A.7 and A.8. Since the total problem size of the unsteady data set is over a gigabyte, and therefore greater than the main memory of each node on the cluster, significant performance degradation is seen when more than one copy of the data set is placed into memory, for example when the number of MPI processes on any node is greater than eight or the total number of processes is greater than 8. Notice that there is a clear

performance penalty in all cases, and in the cases of 10000, 25000, and 50000 the performance of the hybrid code is superior above 12 processors.

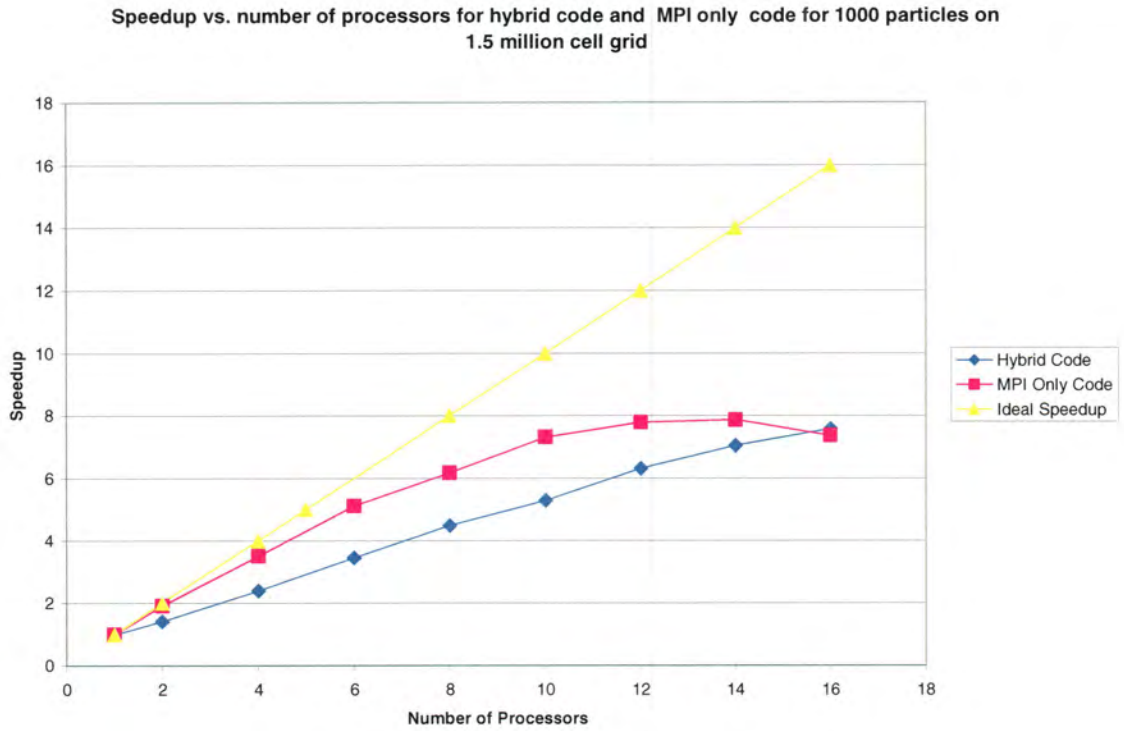


Figure 3.11 - Speedup trend witnessed by 10 of the 16 trials.

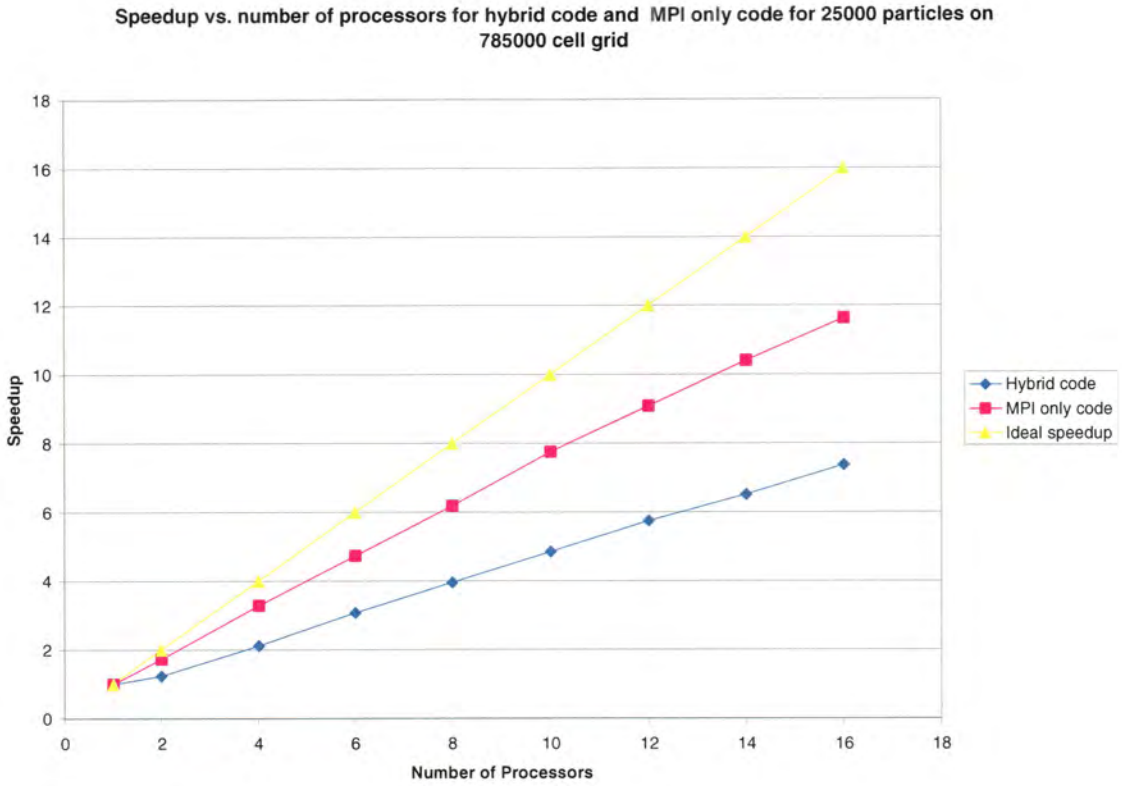


Figure 3.12 - Speedup plot depicting the MPI-only code trend for 6 of 16 trials

Overall, the performance increase from the distributed memory parallelization of the particle tracing code is substantial and the timesavings is significant. The timesavings achieved allow for particle tracing to be performed interactively within the virtual environment and for the virtual environment to achieve interactive frame rates.

CHAPTER 4. VIRTUALTRACE INTERACTION METHODS

Interaction methods are of particular importance to the design of a virtual environment. To achieve immersion the user must feel that he or she is an active participant in the interrogation of data. For a user to be an active participant within a virtual environment, the application must provide natural, intuitive ways to interact with the data, in this case, the flow field.

This chapter contains a discussion of issues pertinent to the choice of interaction methods in the design of a virtual environment. The first section of this chapter gives an overview of some current interaction techniques in virtual environments. This is followed by a summary of the interaction methods implemented for Virtualtrace.

4.1 Overview of Virtual Reality Interaction Methods

This section contains an introduction to several popular interaction methods used in virtual environments and an overview of their typical uses.

4.1.1 The Wand

One of the most frequently used input devices in virtual environments is a "wand" (Figure 4.1). The wand is a convenient interaction device because of its ergonomic shape and the fact that it fits comfortably in a user's hand. Wands have been used in VR applications for navigating through architectural models, for playing virtual sports, and for selecting items from a virtual menu. Wands generally contain a number of buttons that serve as a means of discrete input to the virtual environment. The wand is also position tracked, and therefore can provide the application with information on the wand's position and orientation.

Figure 4.1- A Wand. Image courtesy of Virtual Reality Applications Center.

While the wand is an extremely useful interaction device, it has several drawbacks. One of the drawbacks of the wand is the fact that the user must explicitly remember the function of each button or have a reference key visible within the virtual world. Another drawback is the limited number of buttons available for commands. This causes either limited interaction or overloading of buttons with multiple features, both of which are undesirable.

4.1.2 Static Menus

One method to provide complex menus within surround screen virtual environments is to place a static menu on one of the walls. Using the wand as a laser pointer and pointing at the button on the menu activates the feature of interest. The menus can be hierarchical, meaning that pushing one button brings up an additional menu. This method has been used in several scientific visualization applications including VR-CFD (Kutti, 1998) and VR-XPR (Chu, 2001). However, these menus often occlude the user from seeing data. In addition, interacting with the buttons on these menus by use of the wand as a laser pointer can be

difficult. For these reasons, this type of menuing interface has largely been replaced with other methods.

4.1.2 VUI

The VUI, or Virtual User Interface, developed by Heath (Heath, 1998), is a software toolkit to provide a menu system within a virtual environment that can be dynamically moved and hidden from the user when not in use. The VUI allows users to build 2-D Windows®-based menus within a virtual environment in an extensible object-oriented manner. It was developed with OpenGL (Woo et al., 1997), which made it portable to a variety of different platforms with minimal effort. Users could utilize drag and drop menus within the virtual space, move scrollbars and push buttons on the menus in much the same manner as if the menus were on a desktop. Instead of using a mouse to select the button, users would scroll over buttons with the wand. The primary benefit of this system was that users were already acquainted with these types of menus from their experience with desktop computer menus. An image of a VUI menu is shown in Figure 4.2. The VUI is used in the Virtual Vantage (Diachin et al., 1996) CFD visualization application.

Figure 4.2- Virtual Menu developed with VUI. Image courtesy of Virtual Reality Applications Center.

One drawback of the VUI system is that the menus occlude the user from being able to visualize the entire environment; however, the menus can be hidden when not in use, which reduces the impact of this issue.

4.1.3 Virtual-Tricorder

Wloka (Wloka et al., 1995) developed an interaction scheme based on the Star-trek "tricorder" that took the advantages of a scheme like VUI, and made the menu system function in a more user-centered manner. This system was designed for use primarily with HMDs or BOOM® environments where the user could not physically see their hand because of the VR display. With this system, the user's hand was position tracked and the user held a Logitech 3d mouse. Within the virtual environment, the virtual "tricorder" was mapped to the user's hand at all times. This "tricorder" maintained the same position and orientation in the virtual space as the user's 3d mouse occupied in physical space. The user could push buttons on the wand in their hand, which would subsequently bring up menus in the virtual

environment, similar to the menus used by VUI. This "tricorder" controlled several different parameters, and allowed users to move their hand out of their field of view when not using the menus. In this manner, users could prevent occlusion of objects within the virtual space, but maintain an interactive menu system.

4.1.4 JAIVE

The interaction methods mentioned up to this point function well for a variety of applications, but all have drawbacks when associated with an application that contains a complex menu. Most of these drawbacks are related to the user's ability to use a complex menu within the virtual environment. 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 lead to attempts to integrate palmtop computers into virtual environments as interaction devices. Watsen, Darken and Capps (Watsen et al., 1999) developed a Palm Pilot™-based interaction device that could display quantities of interest such a 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 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 (Hill et al., 2000) extended this system when he developed JAIVE, or Java™ Interface to the Virtual Environment. This interface utilizes a palmtop computer, shown in

Figure 4.3, to handle menu and button interaction on complex hierarchical menus. This computer uses an Intel Celeron® processor running at 266 MHz, containing 32 megabytes of RAM, and a 4.3 gigabyte hard disk.

Figure 4.3 – Intermec Pen*Key 6642 Tablet Computer. Image courtesy of Intermec, Inc.

Hill's implementations used a Java™-based menu on this Windows® computer 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. Changes to menu states are immediately communicated to the VR application. JAIVE makes use of WaveLAN IEEE 802.11 Wireless Ethernet technology to handle the communication between the palmtop computer and the visualization application. The software framework for this system is shown Figure 4.4.

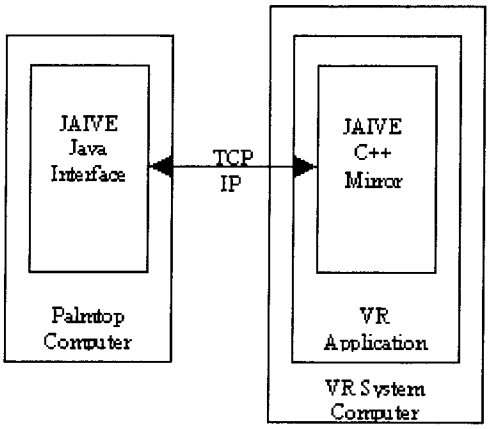


Figure 4.4 – JAIVE Software Framework. Image courtesy of Virtual Reality Applications Center.

The approaches Hill (Hill et al., 2000) and Watsen (Watsen et al., 1999) used 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 palmtop computer allows input and display of numerical or text data in an intuitive manner. Finally, a palmtop computer can act as both an input device and an output device by displaying data of interest.

4.1.5 Speech Recognition

Speech recognition has made rapid advances and is becoming a more stable technology. As speech recognition technology becomes more accurate and reliable, it becomes an attractive candidate for an input device in virtual environments because of the natural interaction capabilities it provides. The most accurate speech recognition applications have been developed for the Windows® operating system, including programs such as ViaVoice™ by IBM or the Microsoft® speech software development kit (SDK)

(Microsoft, 2002). Dorozhkin and Vance (Dorozhkin et al., 2002) developed a speech interface for their spatial mechanisms design program, VRSpatial. Their implementation runs ViaVoice™ on a Windows® 2000 PC. This PC uses the Java™ Speech Application Programming Interface (JSAPI) developed by Sun Microsystems, Inc. and the Common Object Request Broker Architecture (CORBA) (Bolton, 2002) to interface with the IRIX®-based VRSpatial application. This implementation reported excellent results with no training of the speech recognition system and provided a natural interface for complex menu commands.

As an alternative to using a commercial voice recognition package, a speech recognition interface has been developed in unpublished work by Cernak and Sannier at Iowa State University. This speech interface was based on Tcl and the Center for Spoken Language Understanding (CSLU) toolkit (CSLU). CSLU is an application that plugs readily into the Rapid Application Development (RAD) environment and runs on the Windows® platform. In addition, CSLU is free for research use. Rather than having to utilize Java™ and CORBA, the Tcl bindings can be used directly with a TCP/IP network connection to interface with the Windows®-based speech recognition software with the IRIX®-based VR application. This enables users to develop applications quickly with less computational overhead. One drawback of the CSLU toolkit, however, is that to date the performance of the CSLU toolkit is not as good as the performance of commercial speech recognition software.

4.2 VirtualTrace Interaction Methods

VirtualTrace takes a three-pronged approach to supporting interaction methods. This same approach was utilized with success by the IVRESS (Wasfy et al., 2001) CFD application discussed in chapter 2. 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 tasks which the wand is not appropriate, a menu interface was designed for use with a palmtop computer. This provides support for several additional functions, including display of numerical data. Finally, speech recognition was implemented for a variety of tasks. This was designed to enable expert users to quickly control the simulation that they are viewing.

4.2.1 Wand Interaction Methods

A standard wand, similar to the one shown in Figure 4.1 is used to allow users to pick and place things within the virtual environment. The wand serves three main purposes within Virtualtrace. First, the wand is used to allow users to interactively place the particles to be simulated within the environment, or to move them to a different location. Second, the wand is used to define the control points that select volumes for analysis, which will be discussed in more detail in Chapter 5. Finally, the wand is used to locate the cutting planes for analysis, which will also be discussed in Chapter 5. Figure 4.5 shows a user placing particles within the simulation environment using the wand.

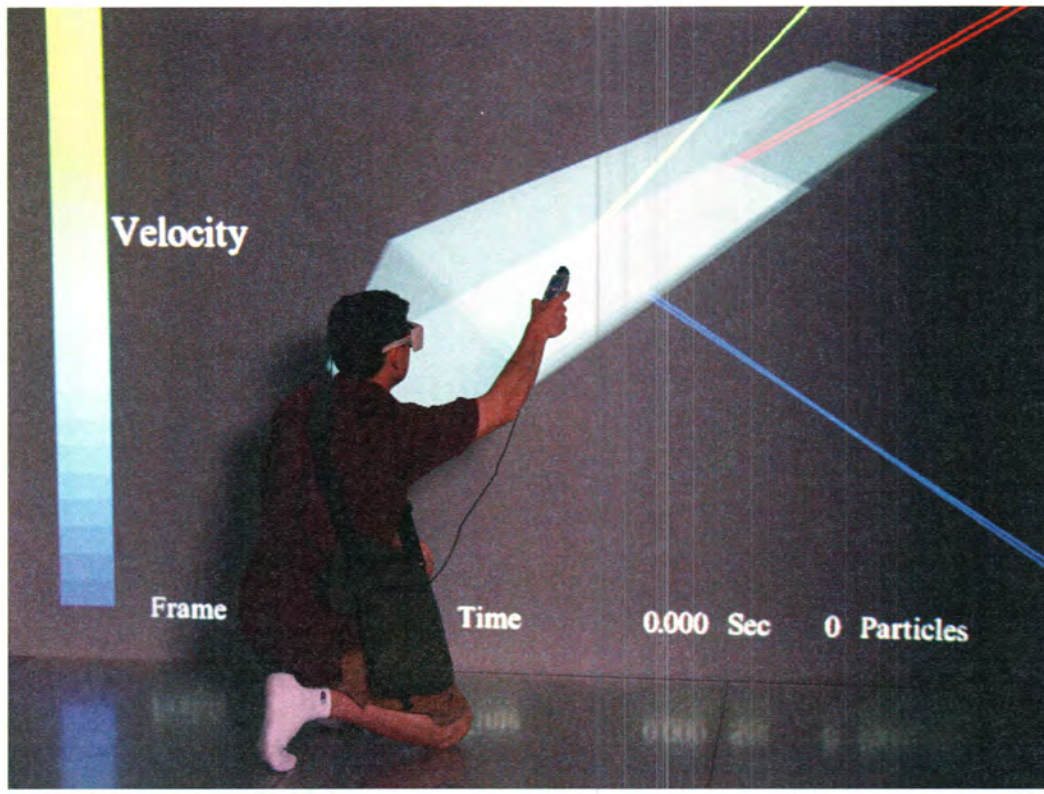


Figure 4.5 – User Placing Particles into the flow field with the wand

4.2.2 Palmtop Menuing System

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 in chapter 5. Therefore, an approach similar to JAIVE (Hill et al., 2000) was chosen.

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® palmtop 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® palmtop shown in Figure 4.4. A picture of a user utilizing the palmtop computer to control the Virtualtrace simulation is shown in Figure 4.6. Another image of the menu on the palmtop system is shown in Figure 4.7. The menu system that was developed provides pop-up menus and a variety of controls including drop down menus, buttons, and sliders.

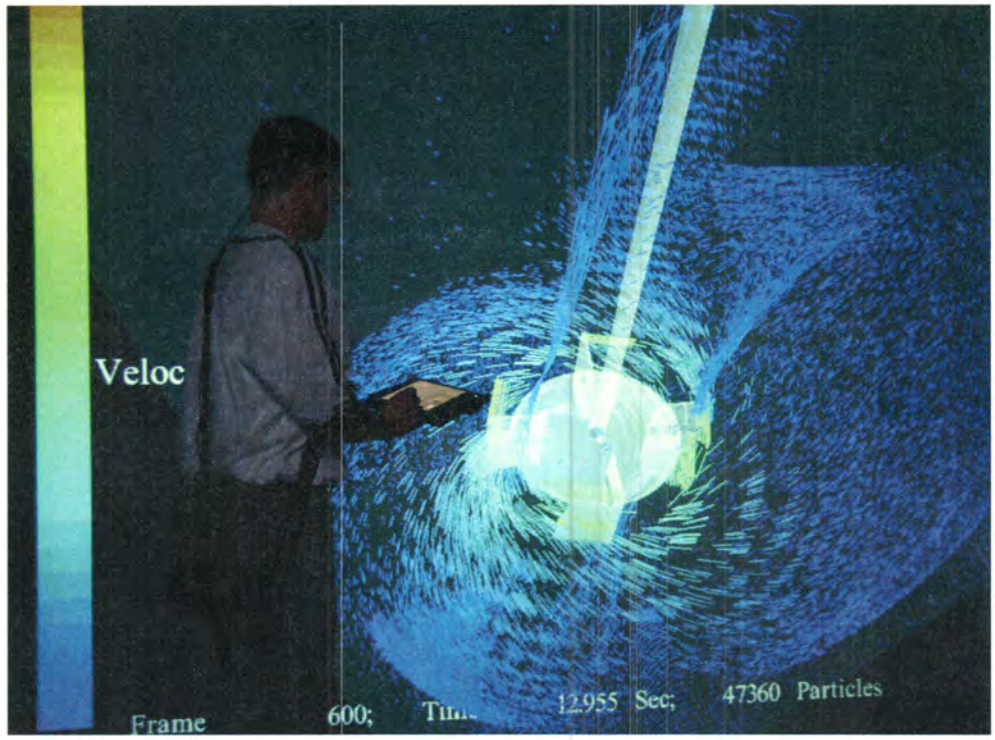


Figure 4.6 – User using palmtop computer to control mixing simulation in virtual environment

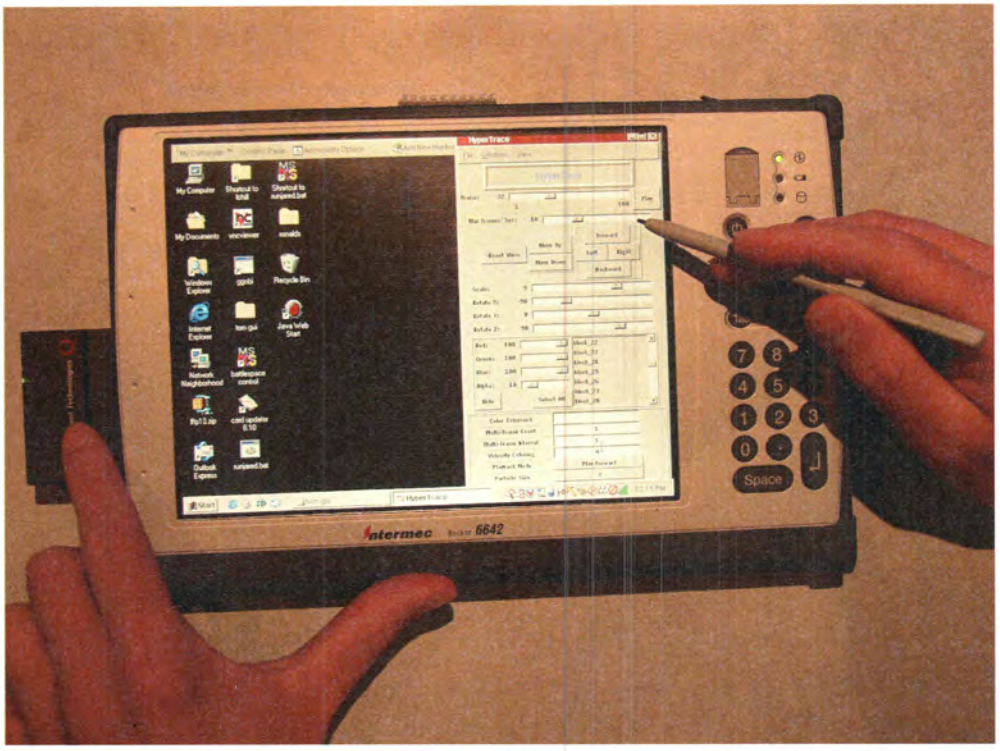


Figure 4.7 – Palmtop menu with HypertraceTM menu active

4.3 VirtualTrace Speech Recognition Implementation

Because the networking code was developed to connect the Windows® Tcl/Tk code to the VR application on IRIX®, the unpublished implementation developed at Iowa State University by Sannier and Cernak was chosen to implement speech recognition. The speech recognition implementation gives rudimentary control over a variety of parameters within the simulation. However, the software's recognition capability is too crude to allow control of some of the more advanced controls in the simulation. The accuracy given by the CSLU toolkit is approximately 70%. While the speech recognition software is undoubtedly a valuable interaction tool, more accurate recognition software must be utilized for more advanced recognition.

The aforementioned three interaction methods chosen for this application provide several different interaction techniques for a range of functionality. The interfaces chosen are designed to make the application simple and intuitive to use for both novice and expert users while maintaining the same menu structure as the desktop version of HyperTrace™.

CHAPTER 5. VIRTUALTRACE DATA EXPLORATION AND INTERROGATION METHODS

Thus far, a significant amount of effort in this thesis has been placed on describing the computation and display of particle traces. However, there has been little mention of methods to interact with the data being displayed. This chapter focuses on the addition of two interaction methods and the integration of existing statistical methods with the VR application.

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 tank, but are essentially surrounded by a storm of particles. It is impossible for users to completely comprehend the data being displayed to them. While 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 chapter two methods are described which were 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. In addition, the statistical capabilities that were developed to enable users to gain a quantitative description of the flow are presented.

5.1 Overview of 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. While 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 (Bowman et al., 1997) 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 that the user can interact with. 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. While 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.

Arn's et al. (Arns et al., 1999) utilized a paint brush 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.

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.

5.1.1 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 (de Berg et al., 2000). 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 3D 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 (O'Rourke, 1998) is modified to run in VR. The theory of the incremental algorithm follows closely from the works of O'Rourke and de Berg (de Berg et al., 2000).

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 H is assumed 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 falls inside the existing convex hull, then the point can safely be discarded. However, if the point does not reside in the convex hull, 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 Equation 5.0

$$\det\begin{bmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{bmatrix} \tag{5.0}$$

where points a, b, and c are the points from the triangular face of the hull, and d is the point being examined. 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 Figure 5.1

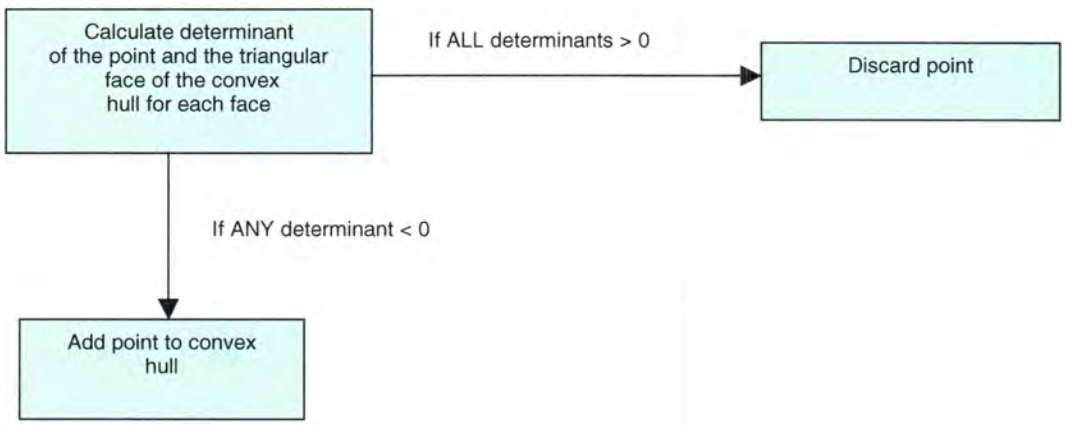


Figure 5.1 – Flowchart of determination process for whether or not a point is inside of a convex hull

If the point lies outside of the existing convex hull, 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 equation 5.0. 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 equation 5.0, 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 5.2 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. Figure 5.2 appears courtesy of Tim Lambert, University of New South Wales.

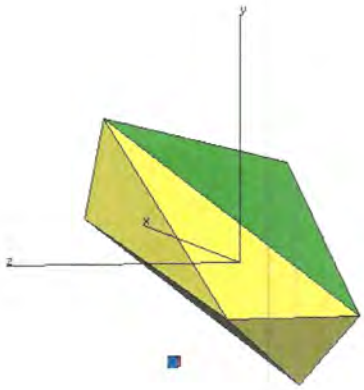


Figure 5.2 - Convex hull. The point in the center of the image is being added. Yellow faces are visible to the point; green faces are invisible to the point.

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 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 Figure 5.2 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 Figure 5.3. Figure 5.4 shows the convex hull shown initially in Figure 5.2 with the new faces added. The new faces are shown in red. Figure 5.4 appears courtesy of Tim Lambert, University of New South Wales.

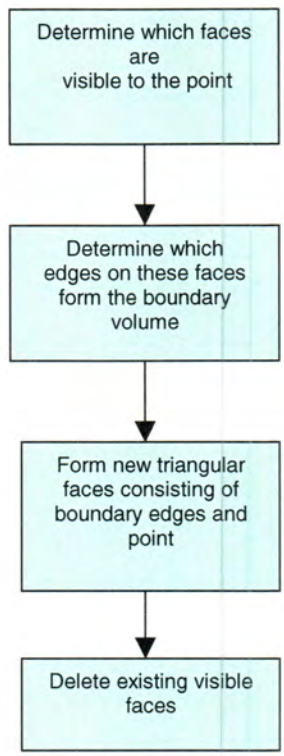


Figure 5.3 – Flowchart for adding a point to a convex hull

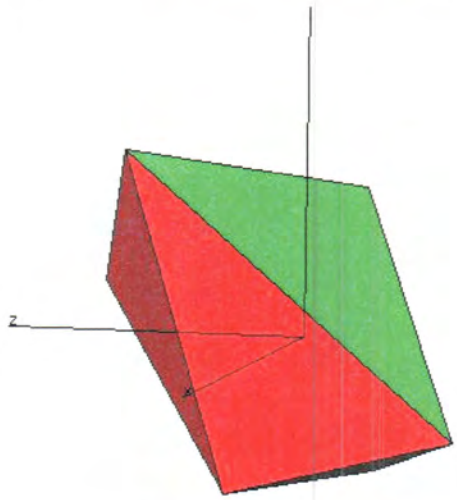


Figure 5.4 - Convex hull with new faces added and existing faces deleted. New faces are shown in red.

5.1.2 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 non-coplanar, since a convex hull cannot be formed from co-planar 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.

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 below in Figure 5.5. A view of the convex hull is shown in Figure 5.6. 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 Figure 5.7. Figures 5.5, 5.6, and 5.7 are courtesy of Dr. Minye Liu, formerly of Cray Research.

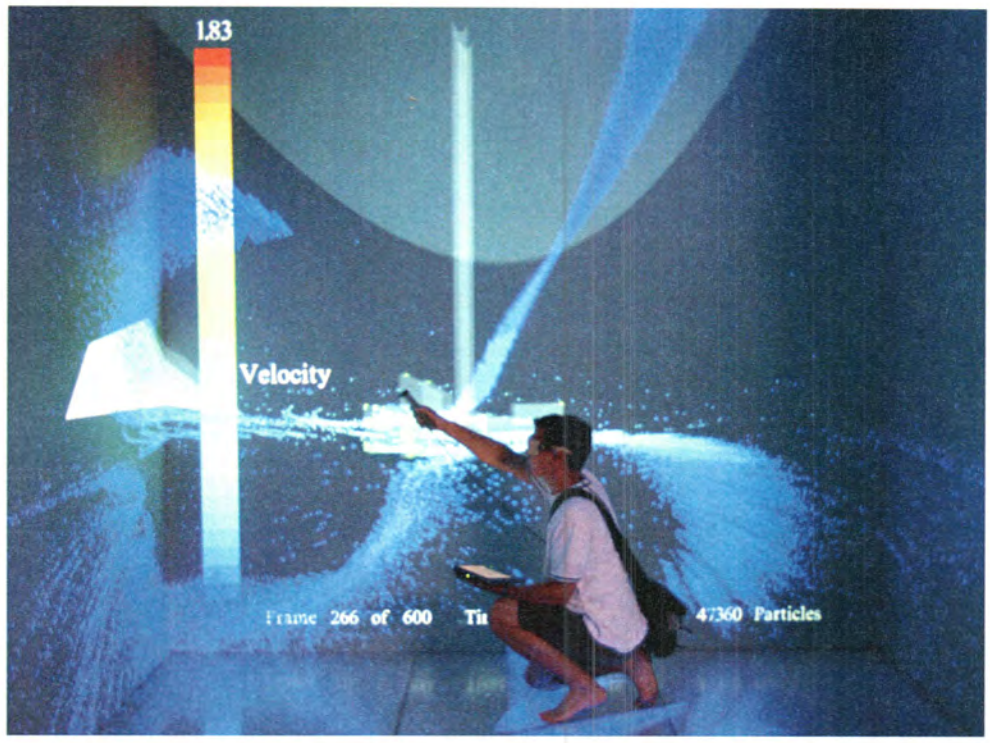


Figure 5.5 – User Creating Convex Hull.

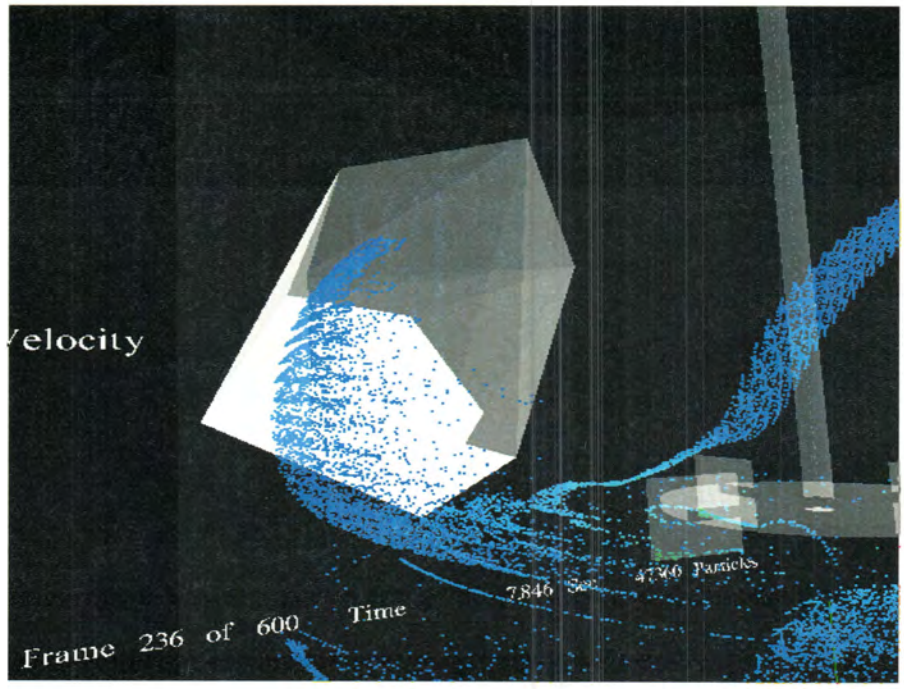


Figure 5.6 – Completed Convex Hull in Virtual Environment

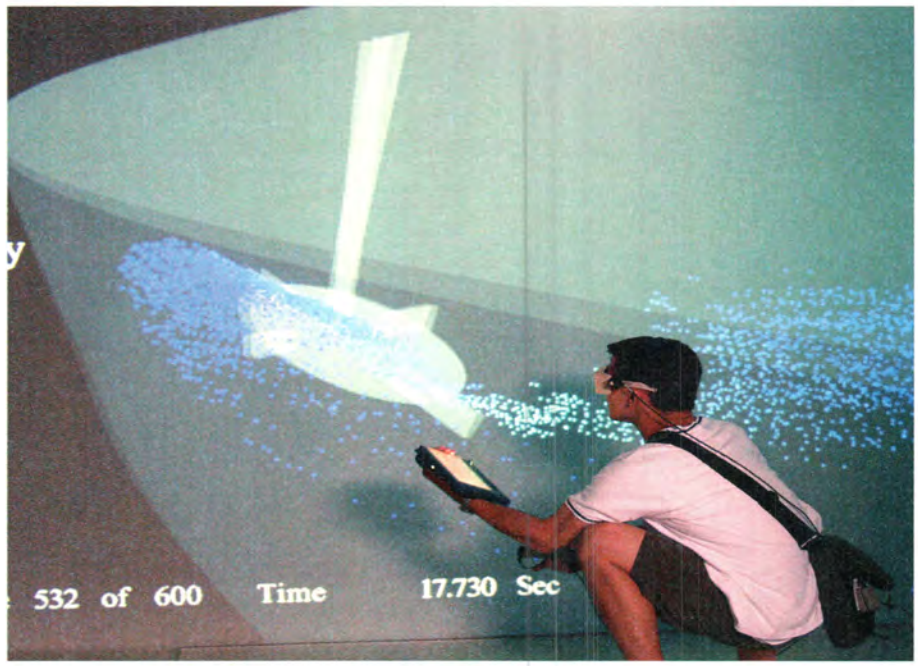


Figure 5.7 – User Visualizing Particles from Selected Volume.

5.2 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.

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 Figure 5.8. The flow field shown in Figure 5.8 is Poiseuille flow, solved numerically using Fluent^{TM} on a 5000 cell grid. The resulting cross sectional plot is shown in Figure 5.9.

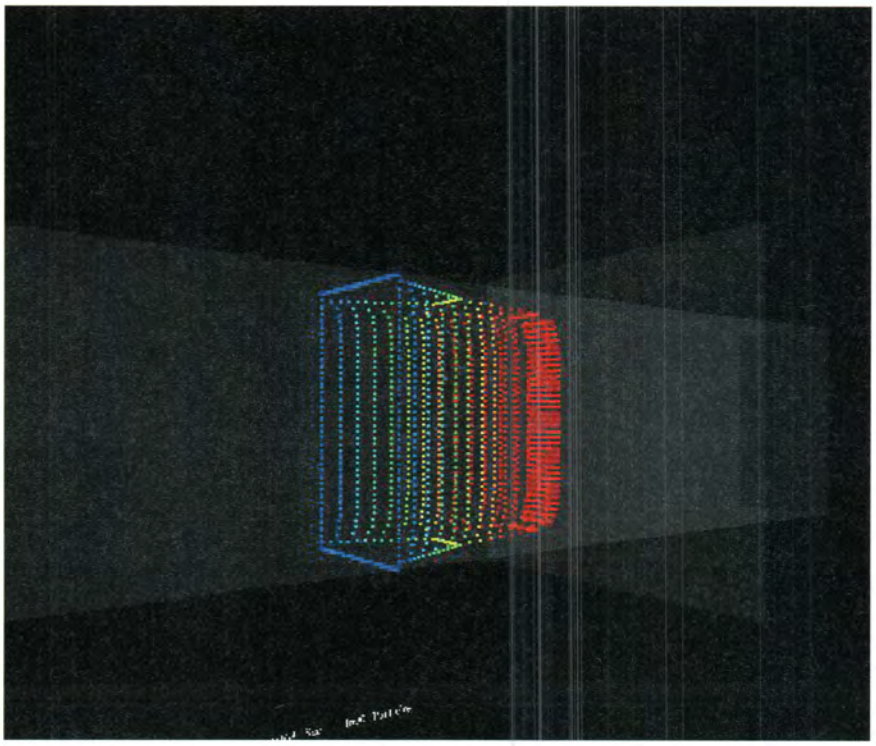


Figure 5.8 – Cutting plane placed into flow field. The cutting plane is visible on the right side of the picture.

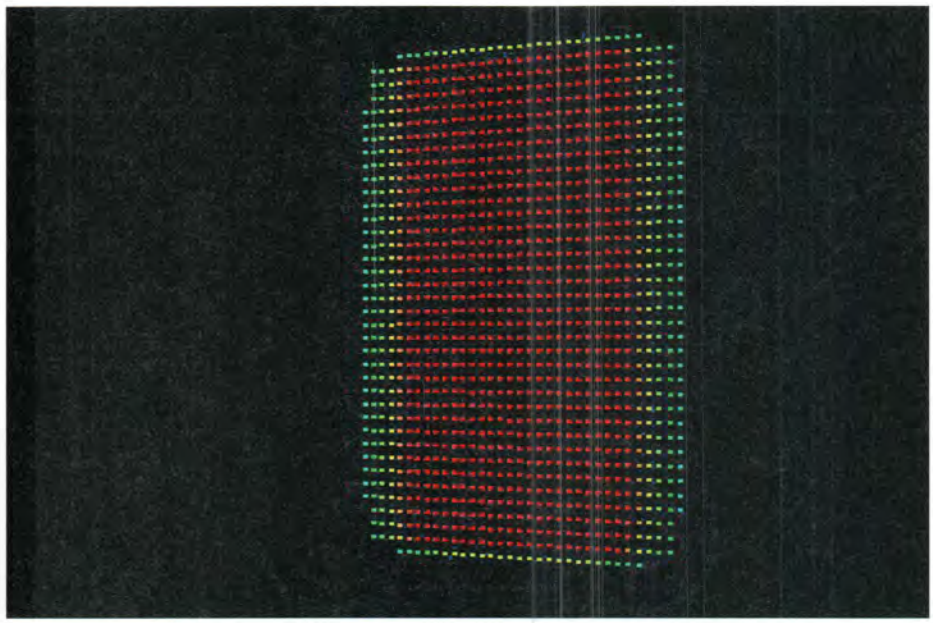


Figure 5.9 – Particle distribution from cutting plane.

5.3 Statistics Computation

As was mentioned in chapter 2, HyperTrace™ was originally developed with statistical capabilities, including the volume coverage of particles within a mixer and the variance between the bins. Volume coverage refers to the number of bins containing at least one particle divided by the total number of bins inside the mixing vessel. The actual number of particles in each bin is ignored for the computation of volume coverage. The variance between the each bin gives a measure of uniformity of the particle distribution between the bins. The original version of HyperTrace™ output the statistics to a text file during the computation of the particle traces for later reference by scientists. However, this method precluded users from interactively viewing the statistics and graphics simultaneously. A more suitable method for interactive interrogation of data would be to have the statistics available within the GUI. In this manner users could observe quantities like the percentage

of tank volume occupied by particles. However, to accomplish this, a major rewrite of the existing statistical code was required. This rewrite was done as part of this research.

To calculate statistics in HyperTrace™, small cubic bins of equal size are placed in the flow field while the particle traces are being computed in the particle tracing application. First, a rectangular grid of bins covering is generated over the entire problem domain. Next, the flow field is used as the reference for the generation and placement of bins. Any bin that does not at least partially fall within the flow field is discarded. Partial bins at the boundaries are considered complete bins. Statistics are then generated at the completion of tracing each frame. This method works well, but cannot provide data to the visualization application because the grid has to be generated with knowledge of the flow field and the binary file contains only particle positions and the boundary geometry. A provisional solution would be to load the flow field into the visualization application, or to write out information about the flow field to a binary file that contains information about the particle traces, but either of these solutions would unnecessarily inflate the amount of memory required to run the application.

To enable the visualization application to generate the grid of bins for statistical calculations independently of the particle tracing application, the visualization application must be able to determine whether or not a bin is located inside or outside of the mixing vessel. Previously it was impossible to determine whether or not the bins were inside or outside of the mixing vessel while the visualization application was running. However, utilizing convex hulls, as introduced earlier in this chapter, allows the application to gain an explicit definition of what is within the boundary geometry and what is outside of the boundary geometry. In this manner, the application can test each bin against the convex hull

of the boundary geometry to determine the bin's location. Bins inside of the flow field are retained, and bins outside of the flow field are discarded. In this manner, the visualization application can generate the bins independently of the particle tracing application. This method was implemented as part of VirtualTrace.

A display on the GUI was developed that displays the salient particle statistical information. This display can be opened at any point during the application to yield statistical information about the particle distribution.

This chapter has presented two visual methods and one quantitative method that allow users to interactively examine the characteristics of particle traces in a virtual environment. First, this chapter presented a novel use of convex hulls to select volumes within a virtual environment, an area of research that has previously been unexplored. Next, this chapter presented the use of cutting planes to show particle distribution within selected areas of the simulation. Finally, the chapter closed with the development of statistical methods using convex hulls.

CHAPTER 6. APPLICATION OF VIRTUALTRACE

This chapter presents how VirtualTrace can enable users to analyze flows within a mixing vessel. This chapter shows two examples, one of a stirred mixing vessel and one of flow through a duct. These two examples allow us to show the use of the interrogative data analysis methods as applied to real applications.

The first application that will be explored is flow within a stirred mixing vessel. Particles are initially placed near the impeller in the bottom of the tank. For this simulation, approximately fifty thousand particles were placed in a cylindrical volume. The particle motion was simulated for 20 seconds and 600 frames (each time-step is considered a frame). This data set, shown in Figures 6.1 and 6.2, was generated by Dr. Minye Liu, formerly of Cray Research. Because of the volume of particles, understanding specific areas of the flow is difficult. Using convex hulls as a selection tool, scientists can observe the particle motion through specific areas of the flow, which would have previously been impossible. Additionally, users can examine the statistical data generated by the particles as they move through the flow on the palmtop computer. Figure 6.1 shows a user examining the data prior to any volume selection taking place. Figure 6.2 shows the user examining the same data set after using a convex hull to select a group of particles. In this image the user sees fewer particles and can make out details that would have been previously impossible to ascertain.

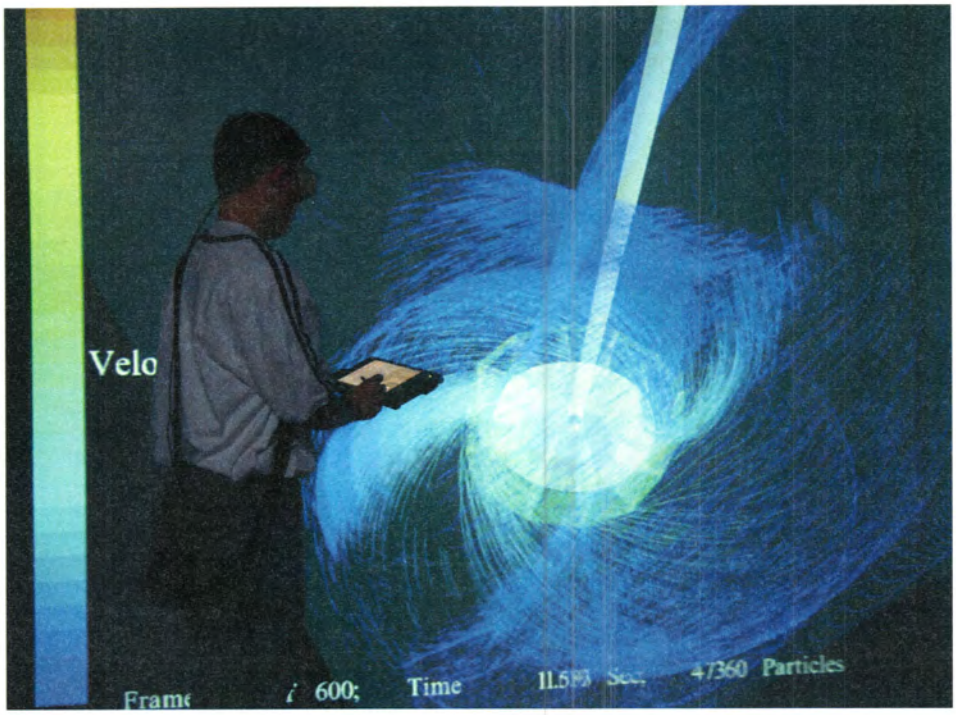


Figure 6.1 – User visualizing chaotic motion through tank.

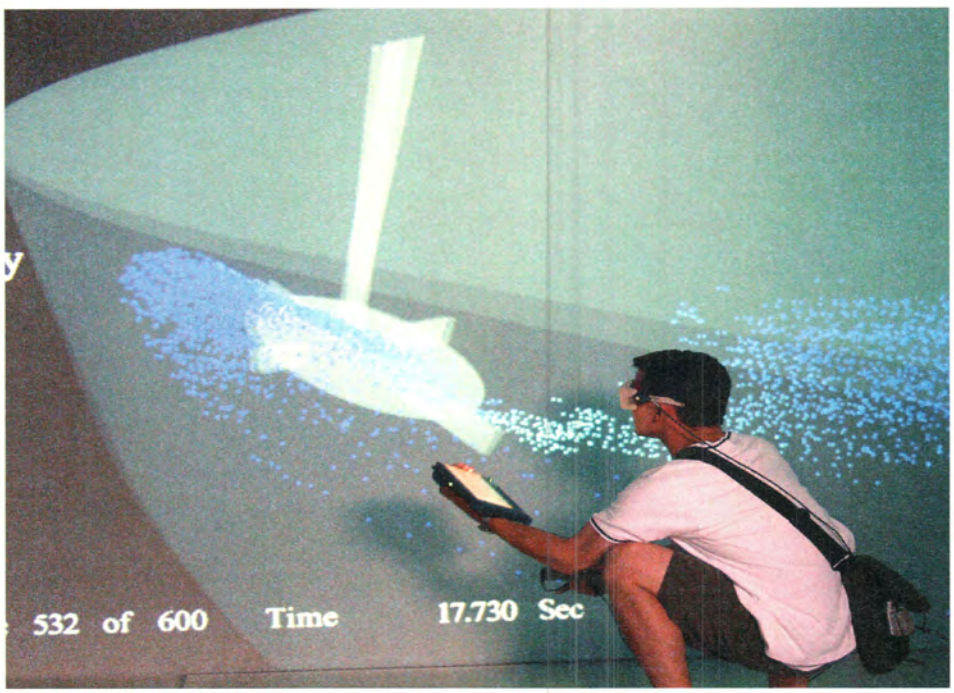


Figure 6.2 – User visualizing particles selected from a convex hull.

The second application explored is Poiseuille flow through a rectangular duct. Poiseuille flow is a flow driven by a pressure difference. Fluid flows from the region of high pressure to the region of lower pressure. Exact solutions for Poiseuille flow through several geometries are published in books by Fay (Fay, 1994) and Munson, et al.(Munson et al., 1998). This flow was numerically calculated using FluentTM on a 5000-cell grid. Figure 6.3 shows particles moving through the duct in the parabolic velocity profile for which Poiseuille flow is known. While no mixing takes place due to the nature of the flow, users can interactively place particles within the flow and visualize the particles subsequent motion. Additionally, users can utilize the cutting plane tool to gain a clear insight into the distribution of particles at the duct exit. Figure 6.4 shows a cutting plane placed at the exit of the flow. This plane shows the particle velocity profile as the particles exit the flow. While this flow is conceptually simple, it shows how the analysis tools can be applied to complex flows in static mixers to determine the particle distribution at the flow exit. Cutting planes were applied to several complex flows through static mixers to gain an increased understanding of the particle distribution at the mixer exit. However, due to the proprietary nature of these mixers the results cannot be displayed.

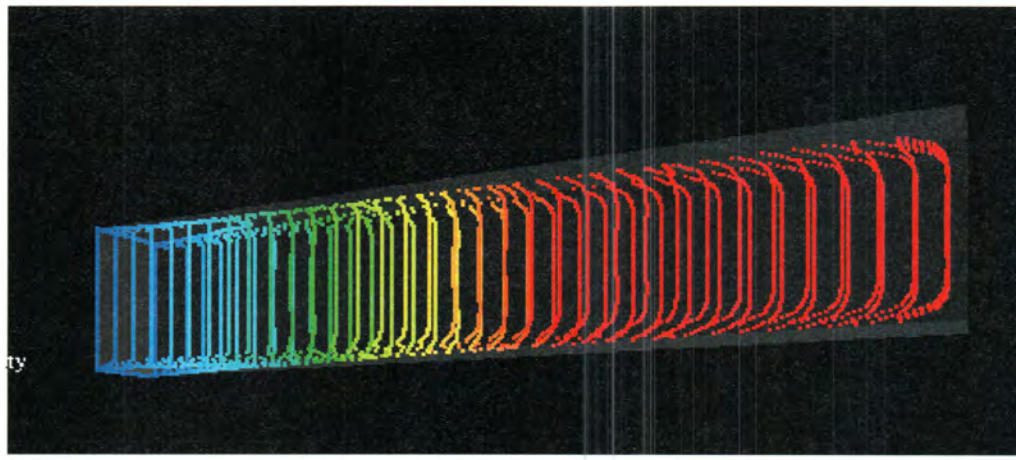


Figure 6.3 – Particle motion through Poiseuille flow

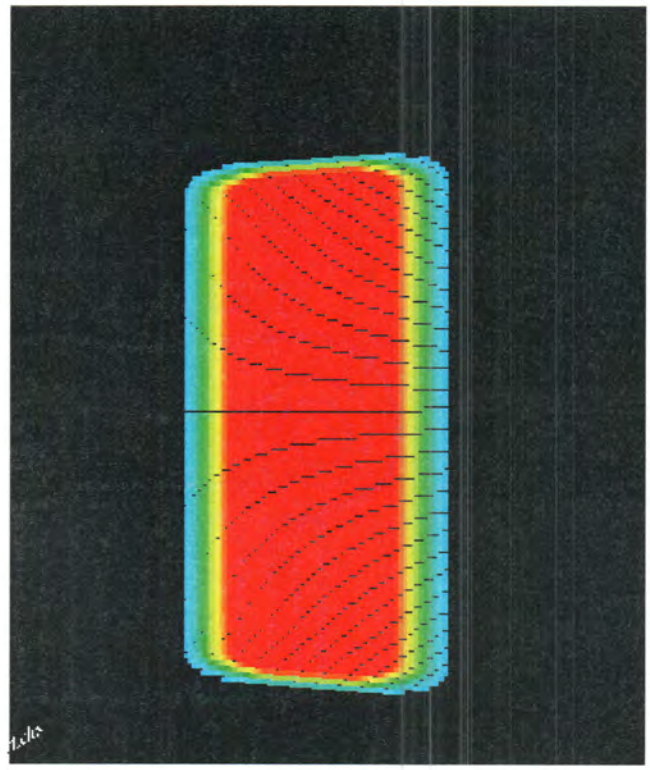


Figure 6.4- Cross-sectional plot of particle intersections with cutting plane in Poiseuille flow

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. Users who had previous experience using HyperTrace™ felt

immediately comfortable with the user interface, particularly because they were already familiar with the command environment. Additionally, users felt like the immersion added to their perception of scientific data.

CHAPTER 7. CONCLUSIONS AND RECOMMENDATIONS

This thesis has presented work that resulted in the development of software platform for interactive analysis of computational mixing data. The application of VR to mixing data has been shown to overcome the limitations posed by traditional 2D graphics. In correlation to this, the initial reaction of users has been very positive. Most users have felt that the VR version of this software gave them an increased understanding of the mixing processes. The goal of extending the original HyperTrace™ software was achieved by the following:

- Creating a distributed parallel version of the particle tracing software
- Enabling interactive tracing of particles in the virtual environment
- Facilitating particle subset selection in the simulation to better understand the mixing devices
- Permitting the placement of cutting planes into the flow field and the viewing the resulting cross sectional profiles
- Migrating the menu system to a palmtop computer
- Adding statistics support to the visualization software

This research has opened many areas for further development. While the current hybrid programming methodology shows no improvement over the standard distributed programming methods, further research could be carried out into how the software scales to additional compute nodes. The results suggest that the hybrid code may prove to be superior to the MPI only code on additional nodes. In addition, load balancing could be more thoroughly investigated to see if there is the possibility of any computational improvement by the use of adaptive load balancing techniques. Finally, there are other open source MPI libraries besides MPICH that could be investigated. The work of Ong and Farrell (Ong et al.,

2000) has shown that there are promising alternatives to MPICH (Gropp et al., 1996) that may deliver superior performance currently in development.

Calculation of the stretching of the fluid elements and stretching statistics could result in increased understanding of the flow characteristics. Research by de la Cruz, et al. (de La Cruz et al., 2001) has shown a unique method of visualizing stretching in 3D that could be implemented.

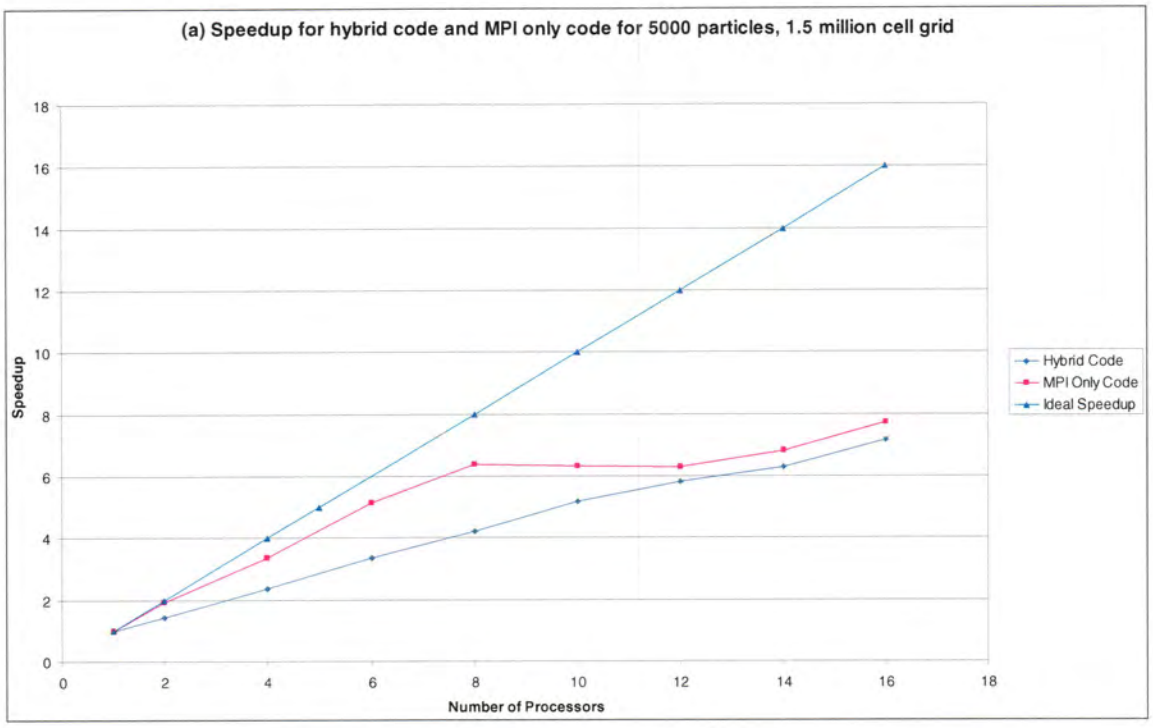
The particle tracing software could be modified to be able to read in additional data file types. While the currently supported EnsightTM data type is robust and widely supported, there would be significant advantages to supporting other file types.

The software could also be modified to color the particles by a variety of scalar values, as opposed to the current method of coloring only by velocity. Particles could be colored by temperature, turbulent kinetic energy, or other quantities of interest.

Another future research area could be to make use of the work of Hartling (Hartling et al., 2000) that allows users from separate virtual environments to see the data simultaneously and interact with the data together, even though they might physically reside at two different locations. This work could also be modified to incorporate the work of Blom, et al. (Blom et al., 2002) to enable more than one viewer to have the image drawn in the correct perspective. While it is currently possible for more than one user to stand inside the C4 or C6, the views looks distorted for all users except the user wearing the head tracked glasses. Blom has developed a method of multiplexing stereo-glasses for two users that allows each user to achieve a tracked perspective without interfering with the other users view.

APPENDIX. PERFORMANCE RESULTS

This appendix contains the results of visualization techniques tested on the 16 processor Linux® cluster.



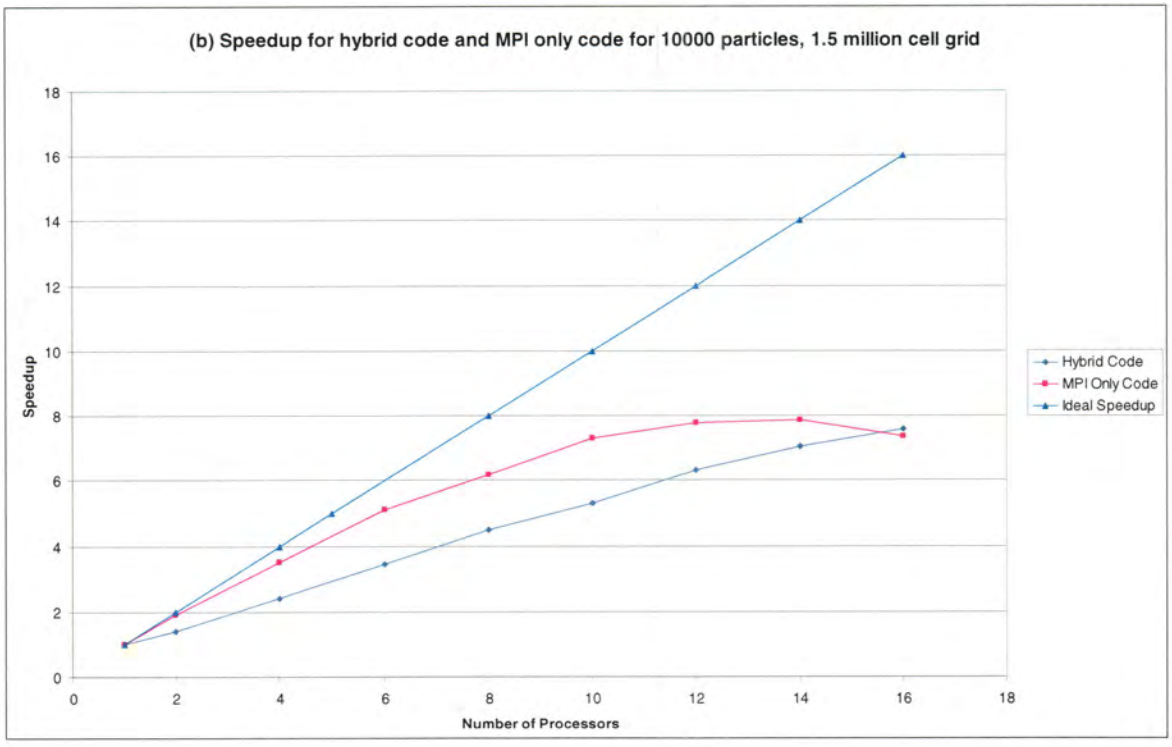


Figure A.1 - Speedup vs. number of processors for hybrid code and MPI only code for 5000 particles (a) and 10000 particles (b) on a 1.5-million cell grid.

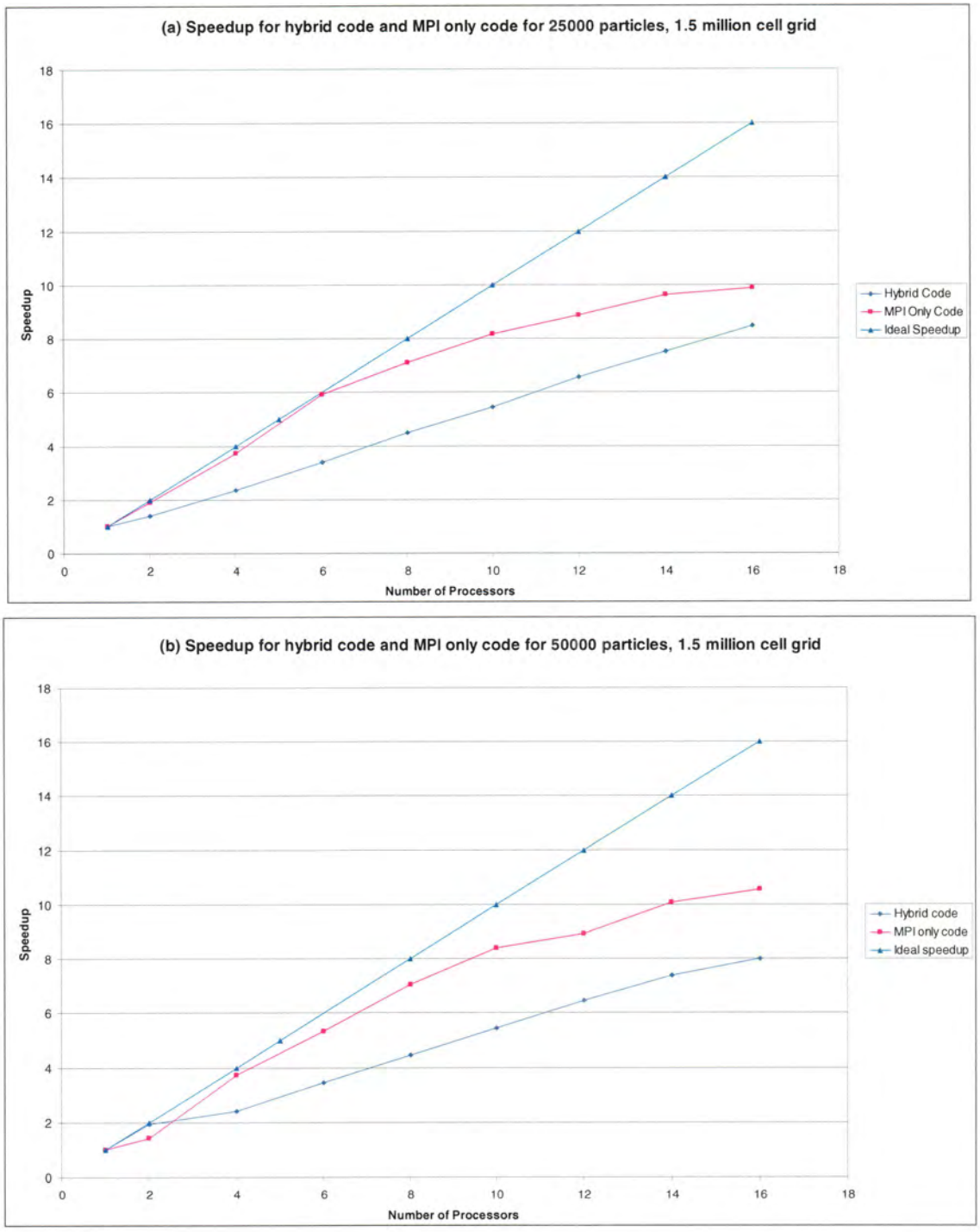
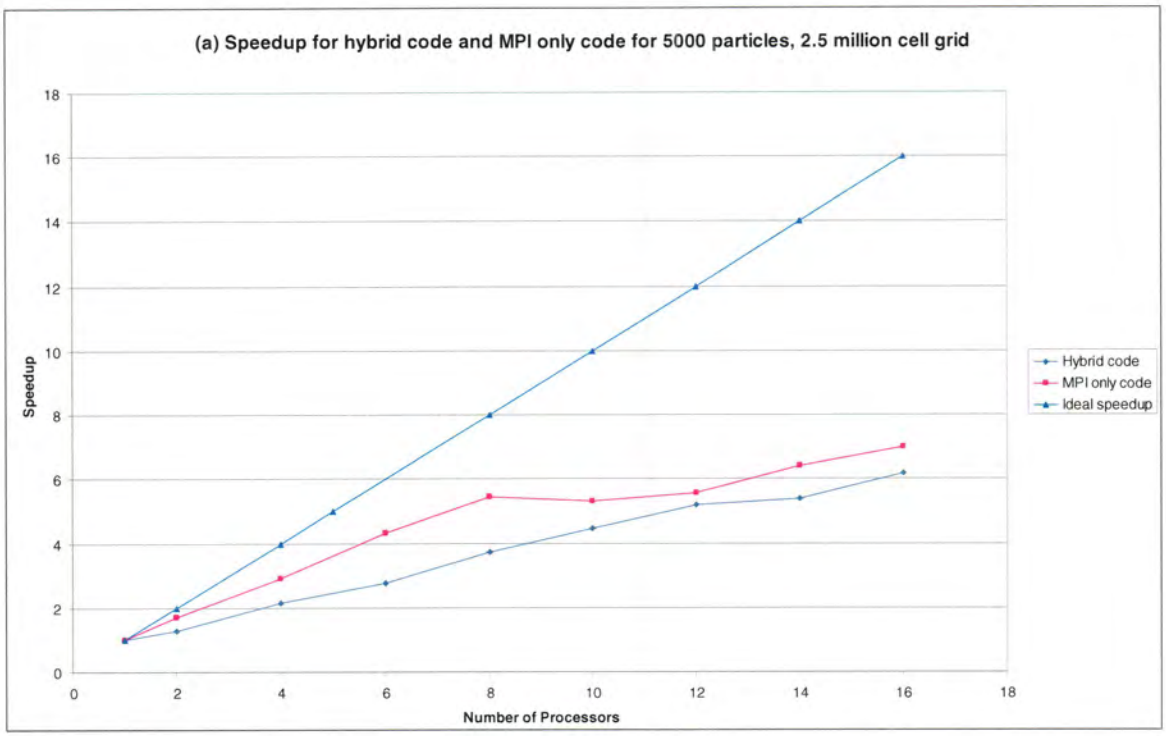


Figure A.2 - Speedup vs. number of processors for hybrid code and MPI only code for 25000 particles (a) and 50000 particles (b) on a 1.5 million-cell grid.



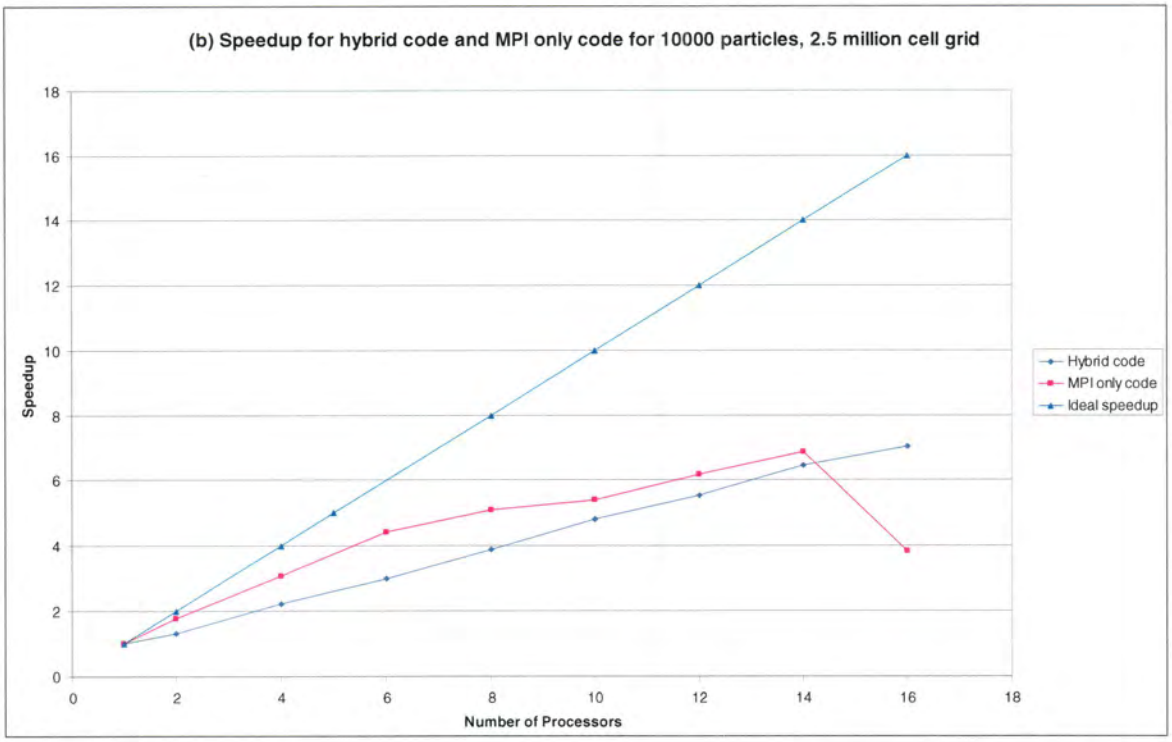
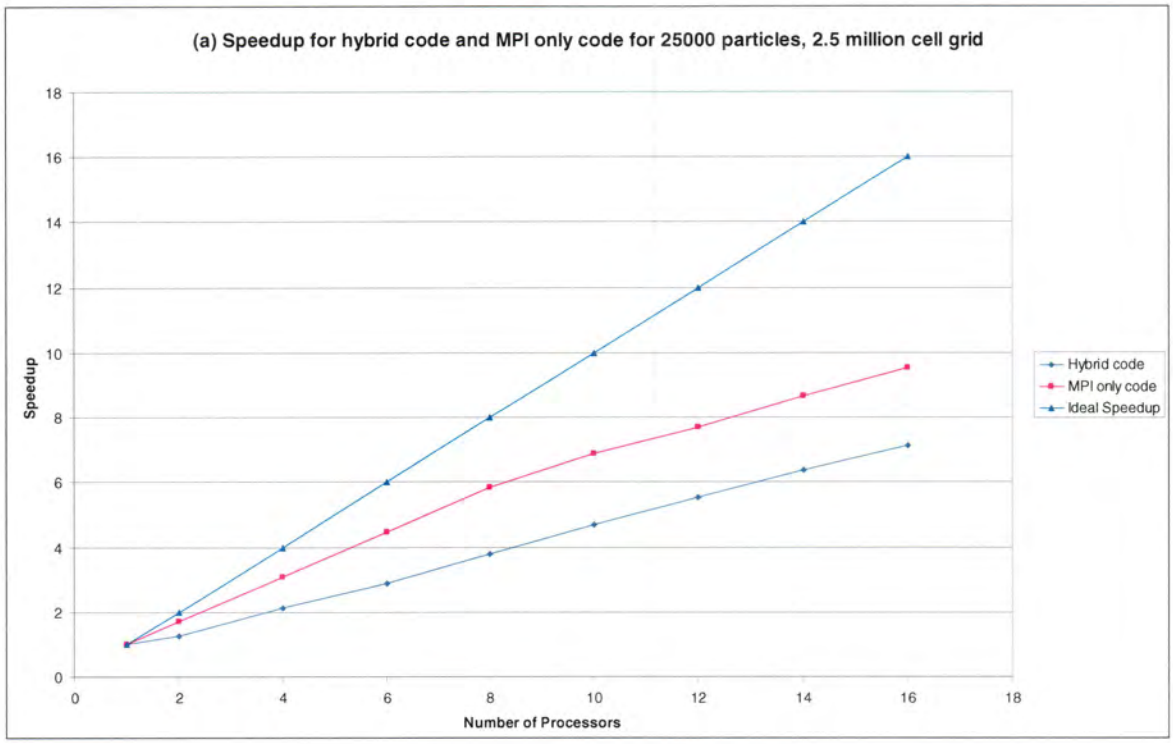


Figure A.3 - Speedup vs. number of processors for hybrid code and MPI only code for 5000 particles (a) and 10000 particles (b) on a 2.5 million-cell grid.



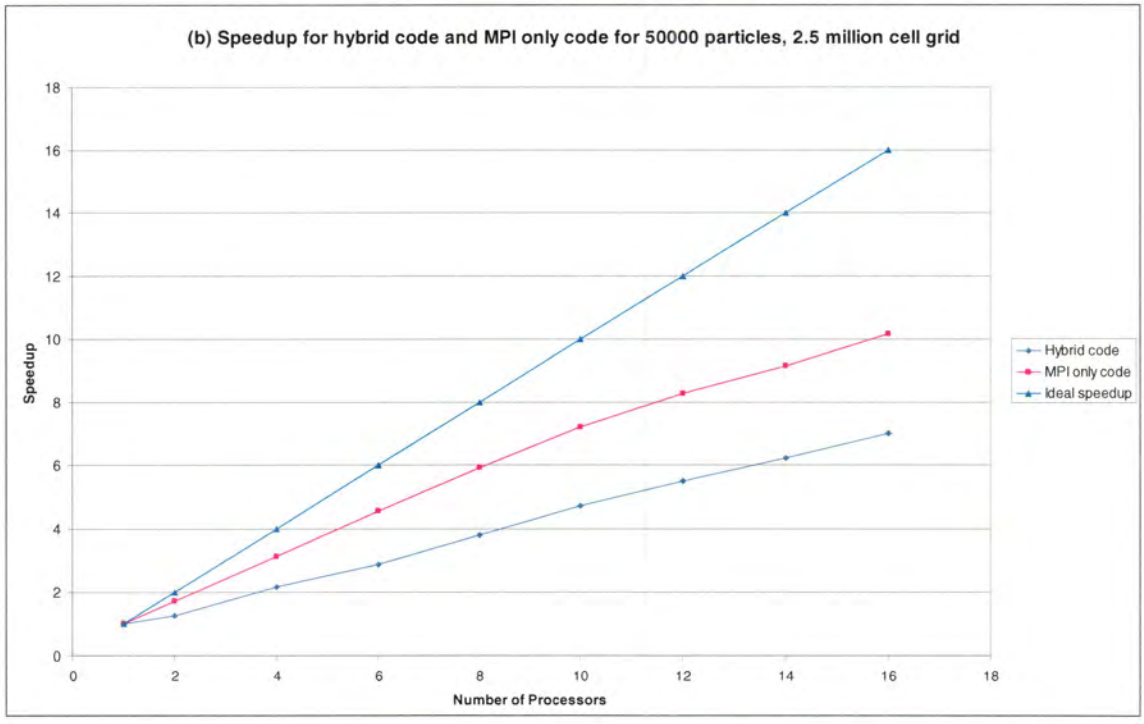
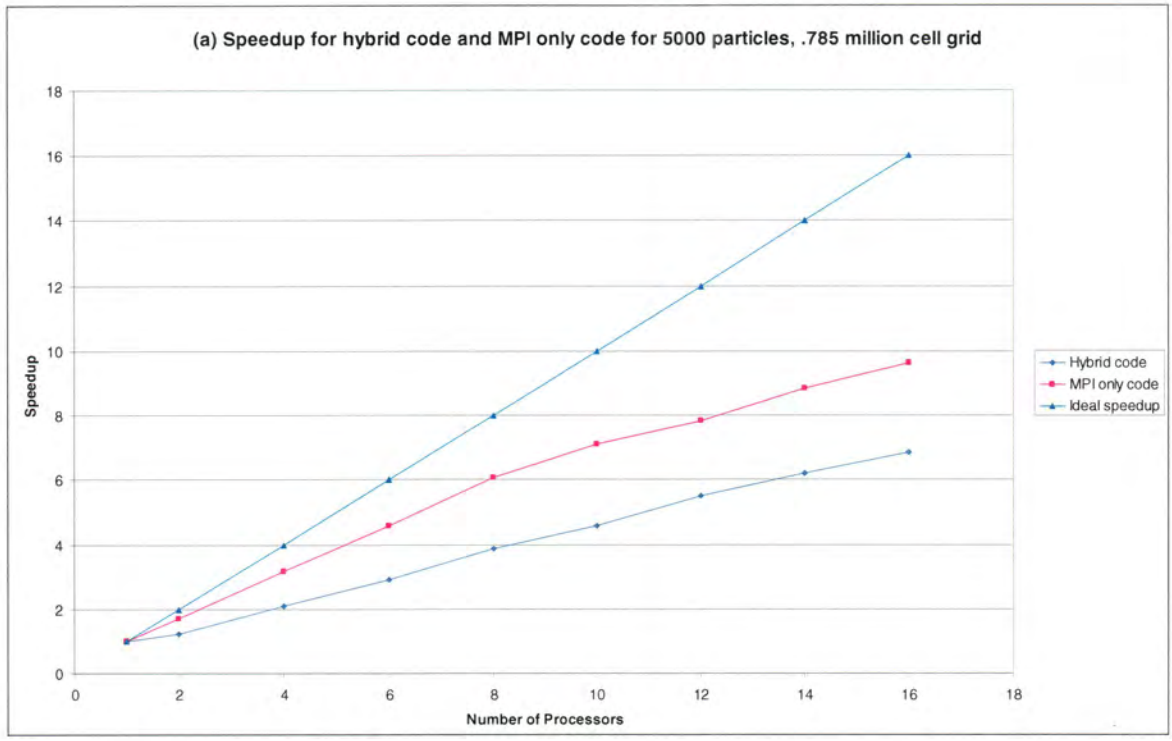


Figure A.4 - Speedup vs. number of processors for hybrid code and MPI only code for 25000 particles (a) and 50000 particles (b) on a 2.5 million-cell grid.



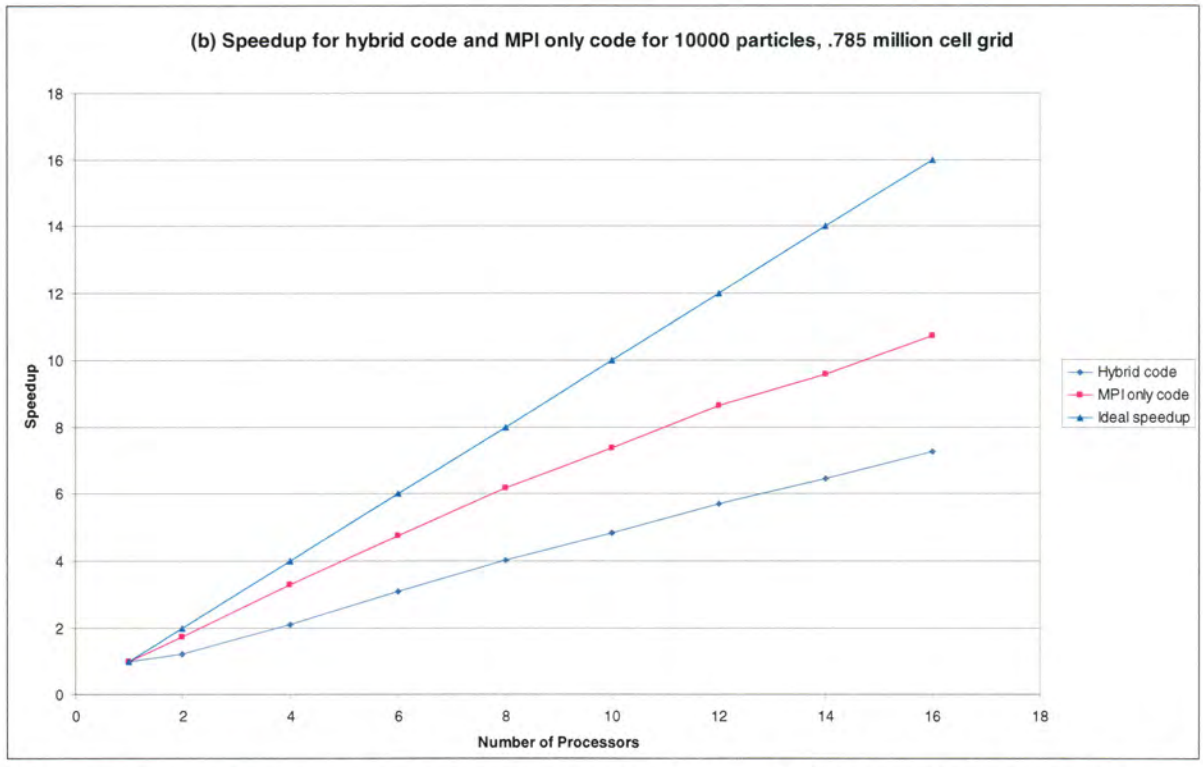


Figure A.5 - Speedup vs. number of processors for hybrid code and MPI only code for 5000 particles (a) and 10000 particles (b) on a 785000-cell grid.

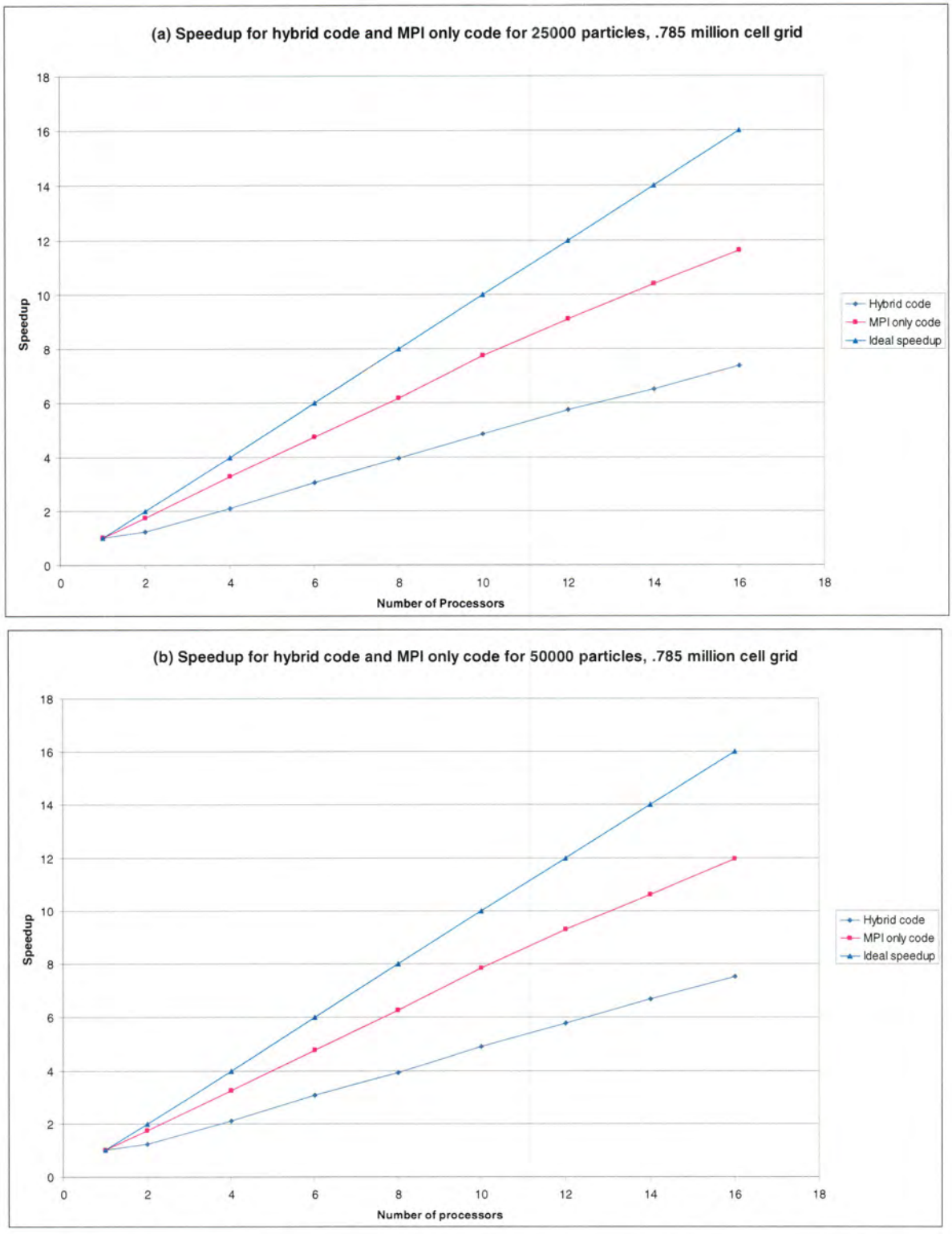
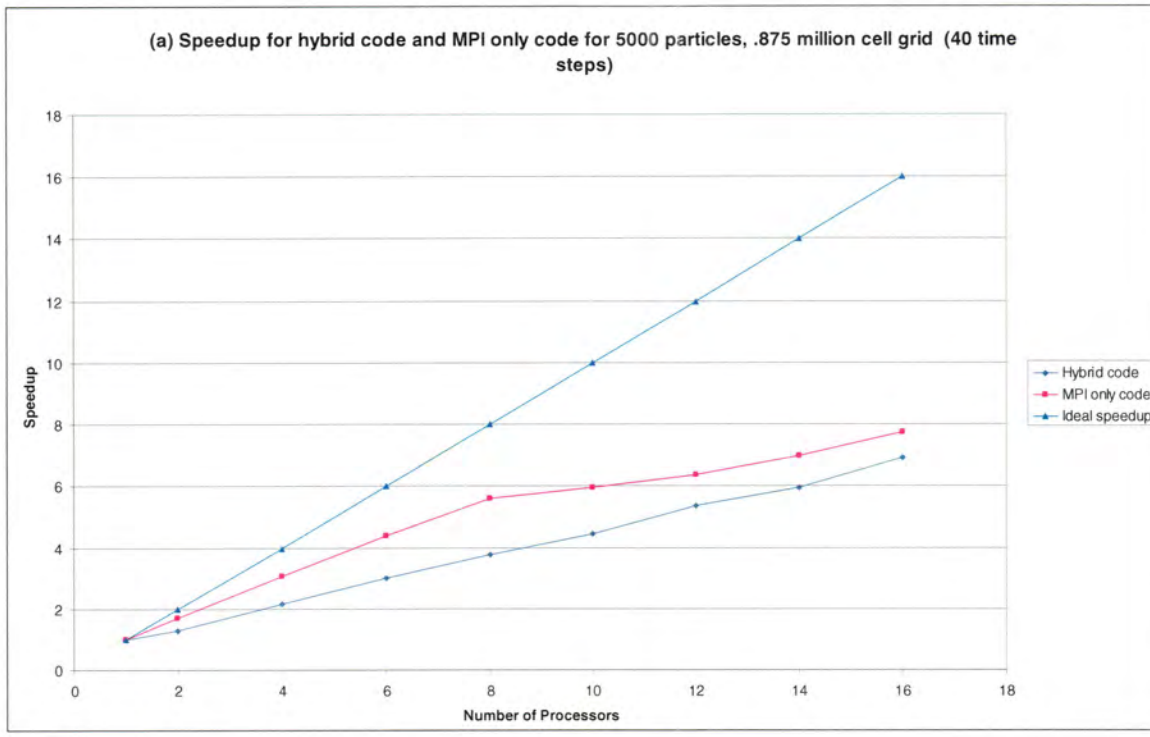


Figure A.6 - Speedup vs. number of processors for hybrid code and MPI only code for 25000 particles (a) and 50000 particles (b) on a 785000-cell grid.



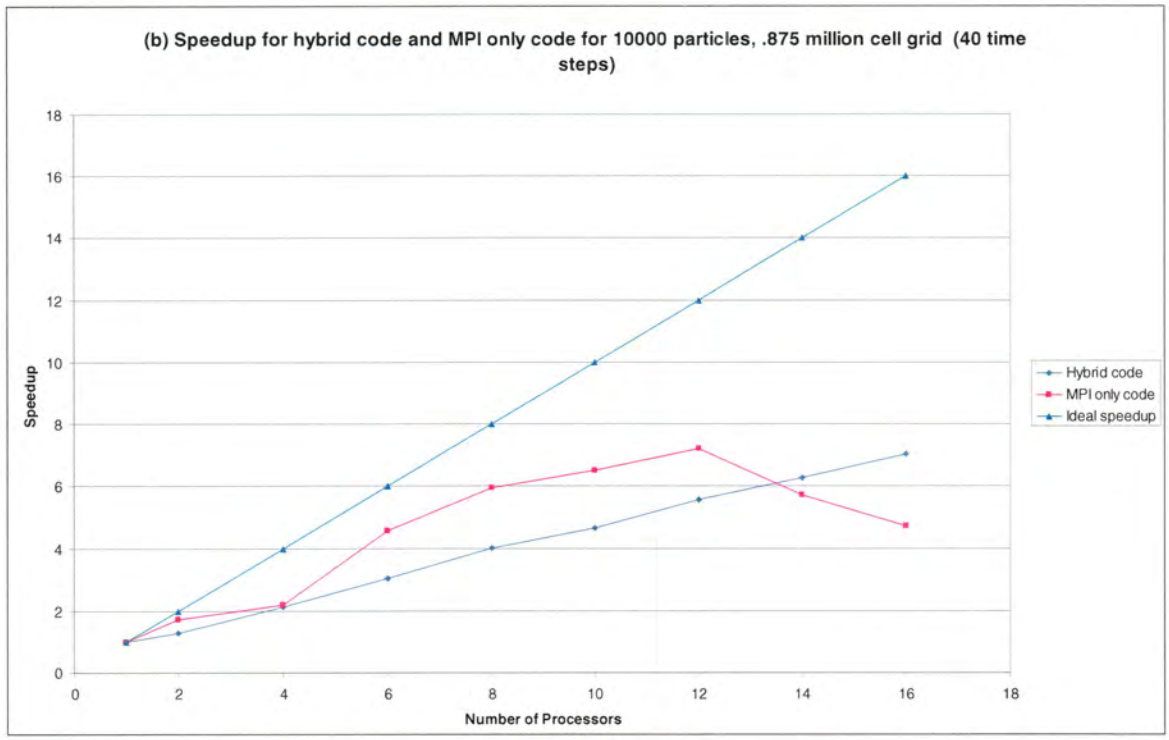
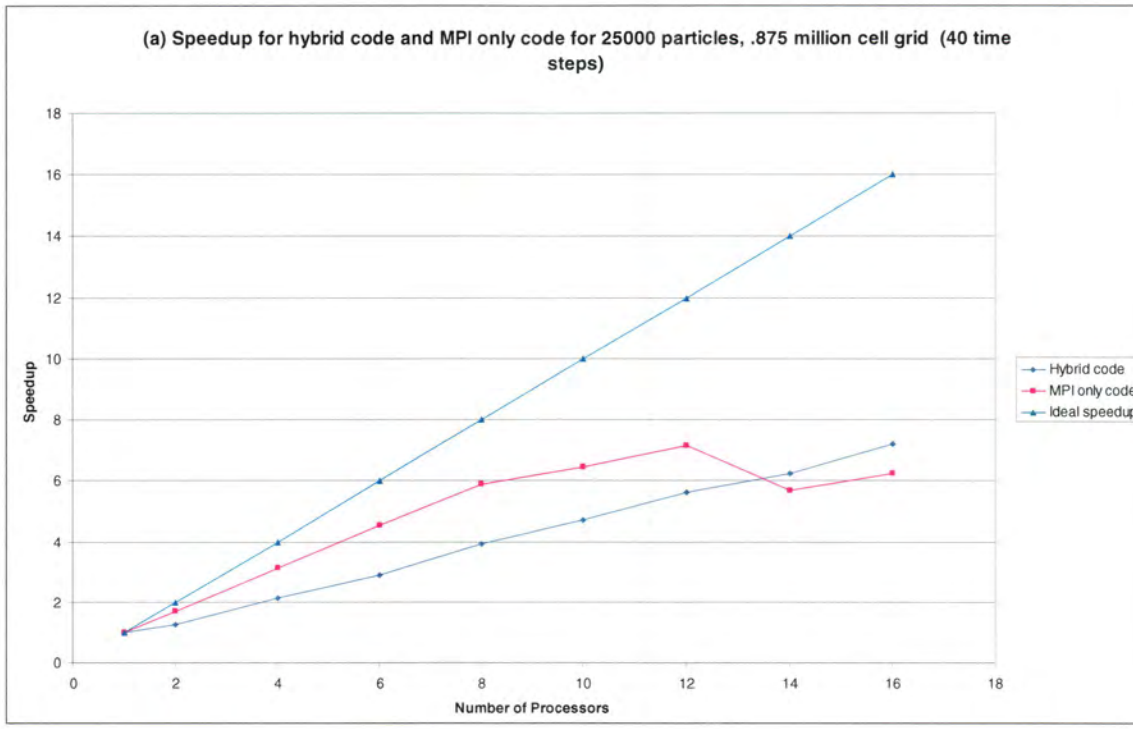


Figure A.7 - Speedup vs. number of processors for hybrid code and MPI only code for 5000 particles (a) and 10000 particles (b) on an unsteady 875000-cell grid.



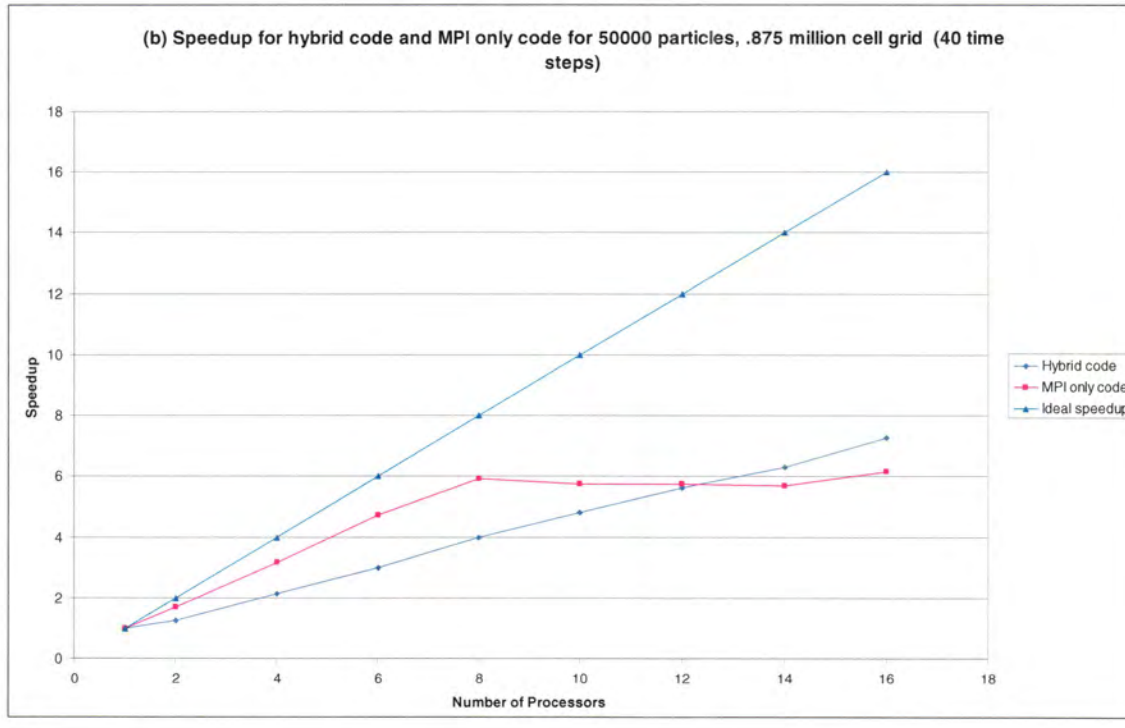


Figure A.8 - Speedup vs. number of processors for hybrid code and MPI only code for 25000 particles (a) and 50000 particles (b) on an unsteady 875000-cell grid.

REFERENCES

Arns, L., Cook, D. and Cruz-Neira, C., 1999, "The benefits of statistical visualization in an immersive environment," *Virtual Reality*, Houston, TX, 88 -95.

Bakker, A., Haidari, A. and Marshall, E., 2001, "Design Reactors via CFD," Chemical Engineering Process, December.

Bakker, A., Haidari, A. and Oshinowo, L., 2001, "Realize Greater Benefits from CFD," Chemical Engineering Process, May.

Bierbaum, A., 2000, *VR Juggler: A Virtual Platform for Virtual Reality Application Development*, Master's Thesis, Iowa State University, Ames, IA.

Blom, K., Lindahl, G. and Cruz-Neira, C., 2002, "Multiple Active Viewers in Projection Based Immersive Environments," *6th International Immersice Projection Technology Symposium (IPT 2002). Proceedings*, Orlando, FL, CDROM.

Boku, T., Yoshikawa, S., Sato, M., Hoover, C. G. and Hoover, W. G., 2001, "Implementation and Performance evaluation of SPAM particle code with OpenMP-MPI Hybrid Programming," *Proceedings of the Third European Workshop on OpenMP*, Sep. 8-9, Barcelona, Spain.

Bolton, F., 2002, *Pure CORBA*, SAMS Publishing, Indiapolis, IN.

Bowman, D. A. and Hodges, L. F., 1997, "An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments," *Symposium on Interactive 3D Graphics*, April 27-30, Providence, RI.

Brooks Jr., F. P., 1999, "What's Real About Virtual Reality?," IEEE Computer Graphics and Applications, 19(6), Nov/Dec, pp. 16-27.

Bruckschen, R., Kuester, F., Hamann, B. and Joy, K. I., 2001, "Real-time out-of-core visualization of particle traces," *Parallel and Large-Data Visualization and Graphics, 2001. Proceedings. IEEE 2001 Symposium on*, Oct. 22-23, San Diego, CA, pp. 45-50.

Bryden, K., Ashlock, D., Cruz-Neira, C., Dornan, J. and Liu, S., 2000, "Interactive Design of Fluid Systems in a Virtual Environment," *Fourth International Immersive Projection Technology Workshop (IPT 2000). Proceedings*, June 19-20, Ames, IA, CDROM.

Bryson, S., 1992, "Virtual Environments in Scientific Visualization," *Compcon Spring 1992. Thirty-Seventh IEEE Computer Society International Conference, Digest of Papers*, Feb. 24-28, pp. 460-461.

Bryson, S., 1996, "Virtual Environments in Scientific Visualization," Communications of the ACM, 39(5), pp. 62-71.

Bryson, S. and Gerald-Yamasaki, M., 1992, "The Distributed Virtual Windtuneel," *Supercomputing '92. Proceedings*, Nov. 16-20, Minneapolis, MN, pp. 275-284.

Bryson, S., Johan, S., Globus, A., Meyer, T. and McEwen, C., 1995, "Initial User Reaction to the Virtual Windtunnel," *AIAA 95-0114, 33rd AIAA Aerospace Sciences Meeting*, Reno, NV.

Bryson, S. and Levit, C., 1992, "The Virtual Wind Tunnel," IEEE Computer Graphics and Applications, 12(4), July, pp. 25-34.

Cappello, F. and Richard, O., 1999, "Performance Characteristics of a Network of Commodity Multiprocessors for the NAS Benchmarks Using a Hybrid Memory Model," *Parallel Architectures and Compilation Techniques, 1999. Proceedings. 1999 International Conference on*, Oct. 12-16, Newport Beach, CA, pp. 108-116.

Cappello, F., Richard, O. and Etiemble, D., 2000, "Investigating the Performance of Two Programming Models for Clusters of SMP PC," *High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium on*, Jan. 8-12, Toulouse, France, pp. 349-359.

Chandra, R., Menon, R., Dagum, L., Kohr, D., Maydan, D. and McDonald, J., 2000, *Parallel Programming in OpenMP*, 300, Morgan Kaufmann, San Diego, CA.

Chipperfield, K. A., 2002, *Interactive Stress Reanalysis in Virtual Reality*, Doctoral Dissertation, Iowa State University, Ames, IA.

Chu, J. T., 2001, *VR-XPR: A rapid prototyping toolkit and framework for developing high performance virtual reality applications*, Master's Thesis, Iowa State University, Ames, IA.

Cruz-Neira, C., Sandon, D. J. and DeFanti, T. A., 1993, "Surround Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," *20th Annual Conference on Computer Graphics and Interactive Techniques*, August 2-6, Anaheim, CA, pp. 135-142.

Center for Spoken Language Understanding, Oregon Health and Science University, "Center for Spoken Language Understanding (CSLU) Toolkit", http://cslu.cse.ogi.edu/, 24 June, 2002.

Dagum, L. and Menon, R., 1998, "OpenMP: An Industry-Standard API for Shared-Memory Programming," IEEE Computational Science and Engineering, 5(1), Jan.-Mar., pp. 46-55.

de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopt, O., 2000, *Computational Geometry Algorithms and Applications*, pp. 235-249, Springer-Verlag, Berlin, Germany.

de La Cruz, L. M., Garcia, I., Godoy, V. and Ramos, E., 2001, "Case Study: Parallel Lagrangian Visualization Applied to Natural Convection Flows," *EEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics, 2001. Proceedings.*, San Diego, CA, pp. 41-44.

Diachin, D., Freitag, L., Heath, D., Herzog, J. and Michels, W., 1996, "Interactive Computational Models of Particle Dynamics Using Virtual Reality," *Virtual Reality in Manufacturing Research and Education. Proceedings*, Oct. 7-8, Chicago, IL.

Diachin, D., Freitag, L., Heath, D., Herzog, J. and Michels, W., 1998, "Interactive Simulation and Visualization of Massless, Massed, and Evaporating Particles," Preprint ANL/MCS-P713-0498, Mathematics and Computer Science Division, Argonne National Laboratory, May.

Dorozhkin, D. V. and Vance, J. M., 2002, "Implementing Speech Recognition in Virtual Reality," *ASME 2002 Design Engineering Technical Conference. To Appear In.*, Montreal, Canada.

Dowd, K. and Severance, C., 1998, *High Performance Computing*, O'Reilly and Associates, Sebastoppol, CA.

NASA Advanced Supercomputing Division, "Flow Analysis Software Toolkit", http://www.nas.nasa.gov/FAST/, 11 June, 2002.

Fay, J. A., 1994, *Introduction to Fluid Mechanics*, MIT Press, Cambridge, MA.

Foster-Johnson, E., 1997, *Graphical Applications with Tcl & Tk, 2nd Edition*, M & T Books, Foster City, CA.

Gerald-Yamasaki, M. J., 1997, "Recent Advances in Visualization for Fluid Dynamics," *AIAA-97-2086,13th Annual Computational Fluid Dynamics Conference*, June, Snowmass, CO.

Gropp, W., Lusk, E., Doss, N. and Skjellum, A., 1996, "A high-performance, portable implmentation of the Message Passing Interface standard," Parallel Computing, 22(6), September, pp. 789-828.

Gropp, W. D. and Lusk, E., 1996, "User's Guide for MPICH, a Portable Implementation of MPI," ANL 96/6, Mathematics and Computer Science Division, Argonne National Laboratory.

Hartling, P. and Cruz-Neira, C., 2000, "Octopus: A Cross-Platform API for Enabling Distributed Virtual Reality Applications," *Fourth International Immersive Projection Technology Workshop (IPT 2000). Proceedings*, June 19-20, Ames, IA, CDROM.

Heath, D. J., 1998, "Virtual User Interface (VUI): A Windowing System for VR," *Second International Immersive Projection Technology Workshop (IPT 1998). Proceedings*, May 11-12, Ames, IA, CDROM.

Hill, L. C. and Cruz-Neira, C., 2000, "Palmtop Interaction Methods for Immersive Projection Technology Systems," *Fourth International Immersive Technology Projection Technology Workshop (IPT 2000). Proceedings*, June 19-20, Ames, IA, CDROM.

Kutti, S., 1998, *Applying Virtual Reality to Computationl Fluid Dynamics Post-Processing*, Master's Thesis, Iowa State University, Ames, IA.

Lamberto, D. J., Alvarez, M. M. and Muzzio, F. J., 2001, "Computational Analysis of Regular and Chaotic Mixing in a Stirred Tank Reactor," Chemical Engineering Science, 56 pp. 4887-4899.

Lane, D. A., 1993, "Visualization of Time-Dependant Flow Fields," *IEEE Visualization '93. Proceedings*, October, Washington, DC, pp. 32-38.

Lane, D. A., 1994, "UFAT - A Particle Tracer for Time-Dependant Flow Fields," *IEEE Visualization '94. Proceedings*, October, Washington, DC, pp. 257-264.

Lane, D. A., 1995, "Parallelizing a Practicle Tracer for Flow Visualization," *Seventh SIAM Conference on Parallel Processing for Scientific Computing. Proceedings*, February 15-17, San Francisco, CA, 784-789.

LaRoche, R. D., 1998, "The Use of Computational Fluid Dynamics for Chemical Process Engineering," *Presented at Chemputers Europe 4*, February, Barcelona, Spain.

Liu, M., Abdreasen, C. D., LaRoche, R. D. and Choudhary, S., 1998, "Visualization and Quantification of Laminar Flow Mixing in a Stirred Tank Reactor," *International Conference on Mixing an Crystalization*, April 22-24, Tioman Island, Malaysia.

Majumdar, A., 2000, "Parallel Performance Study of Monte Carlo Photon Transport Code on Shared-, Distributed-, and Distributed-Shared-Memory Architectures," *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, May 1-5, Cancun, Mexico, pp. 93-99.

Malevsky, A. V., Yuen, D. A. and Jordan, K. E., 1992, "Simulation of Particle Mixing by Turbulent Convective Flows on the Connection Machine," *IEEE Supercomputing 1992. Proceedings*, Nov. 16-20, Minneapolis, MN, pp. 294-300.

Martin, K. W. and Lorensen, B., 1998, *The Visualization Toolkit*, Prentice Hall PTR, Upper Saddle River, NJ.

Microsoft Corporation, "Microsoft Speech Software Development Kit", http://www.microsoft.com/speech, June 17, 2002.

Munson, B. R., Young, D. F. and Okiishi, T. H., 1998, *Fundamentals of Fluid Mechanics*, pp. 367-369, John Wiley & Sons, New York, NY.

Nichols, B., Buttlar, D. and Farrell, J. P., 1996, *Pthreads Programming*, O'Reilly & Asociates, Sebastopol, CA.

Olson, E. C., 2002, *Cluster Juggler: PC Cluster Virtual Reality*, Master's Thesis, Iowa State University, Ames, IA.

Ong, H. and Farrell, P. A., 2000, "Performance Comparison of LAM/MPI, MPICH, and MVICH on a Linux Cluster Connected by a Gigabit Ethernet Network," *4th Annual Linux Showcase and Conference. Proceedings*, Oct. 10-14, Atlanta, GA.

O'Rourke, J., 1998, *Computational Geometry in C*, pp. 101-153, Cambridge University Press, New York, NY.

Schulz, M., Reck, F., BartelHeimer, W. and Ertl, T., 1999, "Interactive Visualization of Fluid Dynamics Simulations in Locally Refined Cartesian Grids," *IEEE Visualization '99. Proceedings*, Oct. 24-29, San Francisco, CA, pp. 413-553.

Shahnawaz, V. J., 2000, *Visualization and approximation of post processed computational fluid dynamics data in a virtual environment*, Master's Thesis, Iowa State University, Ames, IA.

Smith, L., 1999, "Development and Performance of a Mixed OpenMP/MPI Quantum Monte Carlo Code," *First European Workshop on OpenMP. Proceedings*, Sept. 30 - Oct. 1, Lund, Sweden.

Snir, M., Otto, S., Huss-Lederman, S., Walker, D. and Dongarra, J., 2001, *MPI The Complete Reference*, MIT Press, Cambridge, MA.

Stevens, W. R., 1998, *Unix Network Programming: Networking APIs - Sockets and XTI*, Prentice Hall, Upper Saddle River, NJ.

Stuart, R., 2001, *The Design of Virtual Environments*, Fort Lee, NJ.

Tannehill, J. C., Anderson, D. A. and Pletcher, R. H., 1997, *Computational Fluid Mechanics and Heat Transfer*, Taylor & Francis, Washington, DC.

Van Dam, A., LaViola Jr., J. J., Forsberg, A. S., Laidlaw, D. H. and Simpson, R. M., 2000, "Immersive VR for Scientific Visualization: A Progress Report," *IEEE Visualization 2000. Proceedings*, Salt Lake City, UT, pp. 457-460, 589.

Wasfy, T. M. and Noor, A. K., 2001, "Visualization of CFD results in immersive virtual environments," Advances in Engineering Software, 32 pp. 717-730.

Wasfy, T. M. and Noor, A. K., 2002, "Rule-based natural-language interface for virtual environments," Advances in Engineering Software, 33 pp. 155-168.

Watsen, K., Darken, R. P. and Capps, M. V., 1999, "A Handheld Computer as an Interaction Device to a Virtual Environment," *Third International Immersive Projection Technology Workshop (IPT 99). Proceedings*, May 10-11, 1999, Stuttgart, Germany, CD ROM.

Wloka, M. and Greenfield, E., 1995, "The Virtual Tricorder: A Uniform Interface for Virtual Reality," *Eighth Annual Symposium on User Interface Software and Technology. Proceedings*, Nov. 15-17, Pittsburgh, PA, pp. 39-40.

Woo, M., Neider, J. and Davis, T., 1997, *OpenGL Programming Guide, 2nd Edition*, Addison-Wesley, California.

Yeh, T. P. and Vance, J. M., 1998, "Apply Virtual Reality Techniques to Sensitivity-Based Structural Shape Design," ASME J. Mech. Des., 120(4), December, pp. 612-619.