Distributed Visualization in Grid Environment

Gilson A. Giraldi Jauvane C. de Oliveira Bruno Schulze
Rodrigo L. S. Silva
Department of Computer Science
National Laboratory for Scientific Computing
Petrópolis, RJ, ZIP 25651-070, Brazil
gilson,jauvane,schulze,rodrigo@lncc.br, http://virtual01.lncc.br

Abstract

In this paper we focus on the application of grid technology for fluid visualization. Specifically, we are interested on distributed visualization using grid. The focused visualization technique is particle tracing. They can be thought as the trajectory of massless *virtual* particles upon the simulated flow. We propose a distributed particle tracing system and its implementation for collaborative visualization environments in computational grids. We decided for Globus as a middleware and offer solutions to minimize communications between application components during computation, a bottleneck for particle tracers in distributed memory environments. Besides, external memory (Out-of-Core) polices are discussed as an alternative approach to reduce this problem.

1 Introduction

Grid Computing provides transparent access to distributed computing resources such as processing, network bandwidth and storage capacity. A single system image is created, offering open distributed processing support, allowing applications development, usage and maintenance. Grid user essentially sees a single, large virtual computer despite of the fact that the pool of resources can be geographically-distributed and connected over world-wide networks [Foster et al., 2002]. At its core, Grid Computing is based on an open set of standards and protocols (i.e., Open Grid Services Architecture: OGSA [Foster et al., 2002]) that enable communication across heterogeneous, geographically dispersed environments.

In this paper we are interested on grid computing solutions for data analysis in Computational Fluid Dynamics (CFD).

The study of fluid flow and its computational simulation is extremely important since there is a wide range of natural phenomena that can be modeling through fluid theory. Some common engineering examples are pumps, fans, turbines, airplanes, ships, rivers, windmills, pipes, and more recently, the hemodynamics of the arterial system [Barnard et al., 1999].

The data sets produced by the simulations may be too large in size to fit completely inside the internal memory of conventional workstations bringing problems for data analysis. Besides, the data exploration can be a very computational expensive task, especially when using scientific visualization methods, a fundamental set of tools for data analysis [Rosenblum et al., 1994].

Moreover, it is desired that applications and networked users (anywhere in the world) can share a graphical representation of the data, see it from their respective points of view communicated with each other, and interact with the *virtual* environment that represents the data [Dam et al., 2000].

It is a suitable scenario for Collaborative Visualization Systems (CVE). Simply stated, these computer systems integrate input/output devices and computational resources to allow one or more networked users to observe and interact with a computer generated scene [Park et al., 2000]. The range of devices can vary from traditional ones (mouse and 2D desktops) to virtual reality devices (data glove and shutter glasses, for example) [Oliveira and Georganas, 2002].

For CFD data visualization supported by a collab-

orative system, we have to take in account that we may have heterogeneous, and geographically-distributed computational resources (data bases, hosts, devices, services, etc.) linked by a net. Users in distant locations will collaborate in a shared environment and access those resources in order to interact with each other and with the data representation. The access must be transparent to mask differences in heterogeneous computer systems with different operating systems and languages. Thus, we are in the context of Grid Computing [Foster and Kesselmany, 1996].

In this paper, we focus on distributed scientific visualization techniques for CFD using grid. We propose a distributed particle tracing system and its implementation. The result of a particle tracing method can be thought as the trajectory of massless *virtual* particles upon the simulated flow [Lane, 1994, Rosenblum et al., 1994, Hamann et al., 1995, Stolk and van Wijk, 1992].

This paper is organized as follows. Firstly, we discuss the simulation and visualization in CFD. Some aspects of the CFD should be clarified before going on to the focused technique. Then, in section 3 we describe some Collaborative Visualization Environments that use grid computing. Scientific Visualization techniques are discussed on section 4. The particle tracing system proposed, as well as the methodology for implementation are presented on section 5. The innovations of the proposed system are discussed on section 6. Finally, we give conclusion on section 7.

2 Simulation and Visualization

To aid in understanding fluid flow problems, researchers have been used computational simulations based on fluid dynamics equations and numerical analysis. To accomplish this task, a cell decomposition of the domain is taken and the differential equations, plus boundary and initial conditions, are discretized upon that decomposition following Finite Difference or Finite Element approaches [Barnard et al., 1999].

Roughly, the mesh behind that cell decomposition can be classified as regular (each cell is a cube or parallelepiped), curvilinear, which can be thought of as a resulting of warping of a regular mesh, and unstructured one, which consist of arbitrary shaped cells with no particular relation to the regular case.

The use of high performance multiprocessor machines and networks of workstations for fluid simulations is a common practice due to the complexity of the problems in nature. Despite of the computational power of state-of-the art supercomputers, the high performance demand of scientific/engineering community in this field still poses challenges to create, store and analysis data [Johnston et al., 2000]. The *Information Power Grid* is an example of a NASA's initiative in to provide a platform for accessing a wide range of heterogeneous, and geographically-distributed computational resources to achieve such goal in the context of aircraft design [Barnard et al., 1999].

The data sets produced by these simulations may be too large in size to fit completely inside the internal memory of conventional workstations bringing problems for data analysis. Besides, the data exploration can be a very computational expensive task, especially when using visualization methods [Rosenblum et al., 1994].

Scientific Visualization is a relatively new computer-based field concerned with techniques that allow scientists to create graphical representations from the results of their computations, as well as to visualize features of interest in a data set obtained through imaging instruments [Rosenblum et al., 1994].

Despite been computational expensive, visualization techniques have become essential in interpreting data because it exploits the dominance of the human visual channel (more than 50 percent of our neurons are devoted to vision) that makes humans experts at using their highly developed pattern-recognition skills to look anomalies. That is way visualization techniques are so important for scientific data analysis.

From the above discussion, we conclude that to design a collaborative visualization system in CFD, we shall consider a grid computing with an infrastructure of distributed clusters, shared memory supercomputers, as well as single processor machines. This is considered in the next section.

3 Collaborative Visualization

In scientific applications, modern research is not conducted alone. Often a team of collaborators works in the same subject, sharing and discussing partial results. Collaborative visualization systems try to address these necessities and can be augmented with virtual reality technology [Pakstas and Komiya, 2002].

The CAVE system is a very known example (Figure 1). The CAVE technology was first developed at the University of Illinois at Chicago (http://www.evl.uic.edu/). It was built a multiperson 10x10x9 foot theater, with images rear-projected on the walls (screens), and projected down onto the floor. Four projectors, one for each screen, are connected to graphics pipes of one or more high-end workstations (Silicon Graphics Onyx Reality Engine 2 or Infinite Reality). Viewers wear Stereographic' liquid crystal shutter glasses (CristalEyes) to view the stereoscopic images. One user's head is tracked with a 6 degree-of-freedom tracking system, and images are generated from that user's viewpoint. A wand (3D equivalent of a mouse) is also tracked.

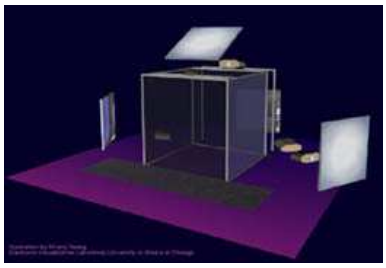

Figure 1: CAVE environment for collaborative visualization.

Despite of CAVE facilities it does not allow a user to share his/her view with a remote user. Next, we describe two systems that offer users that kind of service.

TeleInViVo [Coleman et al., 1996] was developed by the Fraunhofer Center for Computer Graphics research with the sponsorship of the Defense Advanced Research Projects Agency (DARPA) and the US Army Medical Advanced Technology Management Office (MATMO). It is an application that supports collaborative visualization and exploration of volumetric data, including computed tomography, magnetic resonance and PET - Positron-Emission Tomography. The main goal of TeleInViVo is to facilitate diagnosis, medical training, surgery, and therapy planning and treatment, using real-time visualization in a distributed environment.

Collaborative environments have been enhanced with the merging of audio and video with collaborative virtual reality, data-mining and scanned scenarios. The extended concept - Tele-Immersion - is the ultimate synthesis of computer vision techniques (real-time scanned scene reconstruction), networking and graphics [Leigh et al., 1998].

The Tele-Immersive Data Explorer (TIDE) is such a system designed for scientific visualization purposes [Sawant et al., 2000]. It is a framework for groups of scientists, each at a geographically disparate location, collectively participate in a data analysis session, in a virtual environment. The data being analyzed can be stored on data servers, which are at a different location from the clients.

Also, development systems are available. That is the case of CAVERNsoft [Park et al., 2000], a toolkit the rapid creation of tele-immersive applications with the synthesis of not only virtual environments and multimedia techniques but also the access of supercomputing resources and massive data stores, which are connected over high-speed world-wide networks. Globus [Foster and Kesselmany, 1996] is the middleware used in CAVERNsoft G2 (*middleware* is a layer between the operating system and the applications that provides open distributed processing support).

Globus Toolkit and Web services were the two technologies on which Foster at al. [Foster et al., 2002] build to define the Open Grid Services Architecture (OGSA). This architecture offers interfaces to discover services, create dynamic service instances, manage service life-time, notify events and manage the Grid environment. These capabilities are fundamental for collaborative visualization systems implementation because the basic design criteria include: the use of multiple protocols (TCP,UDP, HTTP, etc.), supporting both a distributed shared memory and a message passing model, event generation and handling mechanism, low-level access to networking calls as well as high-level abstractions, security and support for performance monitoring.

The next step is to consider the visualization techniques of interest.

4 Visualization Techniques

The techniques in scientific visualization can be classified according to the data type they manage: Scalar fields ($F : D \subset \Re^3 \rightarrow \Re$), vector fields ($F(x)$ is a vector, $x \in D \subset \Re^3$) and tensor fields compose the usual range of data types in this field.

Henceforth, we have methods for scalar fields visualization (isosurface generation and volume rendering, colormap, etc.), vector fields visualization (field lines generation, particle tracing, topology of vector fields, LIC, among others) and techniques for tensor fields (topology and hyperstreamlines) [Rosenblum et al., 1994].

Once in computational grids the resources may be geographically-distributed, we must consider in this discussion the visualization techniques that can be implemented efficiently in distributed memory environments.

Among those cited methods, isosurfaces and volume rendering are the most proper for this king of architecture [Chiang et al., 2001]. Although they are not in the core of this proposal, it is interesting to see some details of them for further analysis.

In volume rendering, the visualization model is based on the concept of extracting the essential content of a 3D data field by *virtual* rays passing the field [Rosenblum et al., 1994]. These rays can interact with the data according to artificial physical laws designed to enhance structures of interest inside. These laws can be summarized in a transport equation of the form:

$$\frac{dI}{ds} = -\sigma(s) I(s) + g(s), \qquad (1)$$

where s is a distance over the ray, I is the scalar field to be visualized, σ is the extinction coefficient, and $g(s)$ represents generalized sources [Rosenblum et al., 1994]. Figure 2 pictures the basic idea behind the model.

Isosurface extraction methods work differently [Rosenblum et al., 1994, Chiang et al., 2001]. Given a value q and a scalar field F, the isosurface q is defined as the set: $C(q) = \{x \in \Re^3, F(x) = q\}$.

As well as volume rendering, the result is a bi-dimensional image out of the three-dimensional data. However, in this case all data cells are first

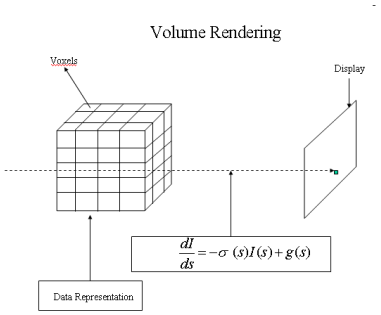

Figure 2: Volume Rendering: virtual rays passing the field, interacting with data and give the final image.

visited to identify cells that intersect the isosurface $C(q)$. Then, the necessary polygon(s) to represent the portion of the isosurface within the cell is generated and stored. Up to the end of this process, the obtained set of polygons gives a piecewise linear representation of the isosurface.

In volume rendering, each ray contribution can be computed independently (see Figure 2). The same is true for each portion of an isosurface. Hence, both these methods can be efficiently implemented in distributed memory machines [Ma, 1995, Lombeyda et al., 2000, Rosenblum et al., 1994].

The implementation of the other methods cited, for distributed memory machines, depends on special considerations.

In this paper we are interested on particle tracing methods [Lane, 1994, Rosenblum et al., 1994, Hamann et al., 1995, Stolk and van Wijk, 1992]. They can be mathematically defined by an initial value problem [Rosenblum et al., 1994, van Wijk, 2002]:

$$\frac{dx}{dt} = F(x,t), \quad x(0) = P_0, \qquad (2)$$

where $F : \Re^3 \times \Re_+ \rightarrow \Re$, is a time-dependent vector field (velocity, for example).

The solution of the above problem for a set of initial conditions gives a set of (integral) curves which can be interpreted as the trajectory of massless particles upon the flow defined by the field $F(x,t)$. Other particle tracing methods can be used (streamlines, streaklines, etc.) through slight modification of the equation (2) [Rosenblum et al., 1994].

The problem (2) in general does not have analyt-

ical solution. Thus numerical methods must be used [van Wijk, 2002, Barnard et al., 1999]. These methods basically generate the particle trajectory step-by-step following some scheme for field interpolation inside cells [Hamann et al., 1995].

4.1 Particle Tracing in Grid

The particles are independent each other. Thus, the method is easily implemented for shared memory machines [Lane, 1994, Rosenblum et al., 1994].

For distributed memory machines special considerations arise.

Firstly, it is not efficiently in general to send the whole data set to all nodes due to its amount. Thus, we need to partition the original data set into subsets which, following a general nomenclature, we will call **meta-cells** [Chiang et al., 1998].

Figure 3 represents such a sub-division and helps to understand the main challenge in this case. In this figure we represent the meta-cells, the particle positions at $t = 0$ and $t = n\Delta t$, where Δt is the time step used for numerical integration of the equation (2). It is supposed that each sub-set of data will be sent to a grid node that is the host of the corresponding process.

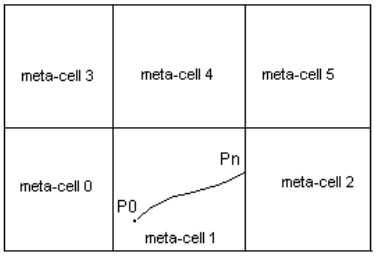

Figure 3: Meta-cell subdivision and particle position P_0 and $P_{n\Delta t}$. The last position belongs to the meta-cell boundary. Thus processes communication is required.

In this case, a particle trajectory (solution of problem (2)) found the meta-cell boundary. Thus, the corresponding processor is not able to continue the integration. Hence, the process should sent a message to a master with the last position found. If we consider that, for practical applications, we may have more than thousands of particles, the communication between processors would be a serious bottleneck.

For some applications (see bellow) it is possible to use problem symmetries to find a meta-cell subdivision that reduces this problem. However, for unstructured meshes (one application of our interest) it is difficult to prevent such problem because we are not able to know in advance the domain region that will probably encompasses a particle trajectory.

The search for methods to efficiently address this problem is the starting point of our research project described next.

5 Research Proposal

We aim to implement a particle tracing system that will be distributed over a computational grid. The proposal starts with a centralized architecture with a *master process* that start the *slaves* and keeps the *directory service* (grid nodes address, particles positions at $t = 0$, meta-cell structures, for each meta-cell the grid nod assigned, etc.).

We emphasize that the sub-division scheme is static in this proposal; that is, at the initialization time, meta-cells are created and each one sent to a grid node. We are not going to consider the possibility of a processor to require to the master another tile of data at run time.

The following items will be implemented.

a) *Specifies meta-cell structure by using prior knowledge about the flow and mesh symmetries:* For some applications, the solution of the flow simulation is determined on a structured body-fitted, 3D, curvilinear computational mesh. Thus, it can be constructed numerical schemes in accordance with the physical symmetry of the flow. Figure 4 pictures an example for aircraft design.

Those symmetries can be used to steer the meta-cell construction. In the given example, it is expected that we must partition the data set in curvilinear tiles following the aircraft structure.

b) *Out-of-Core techniques:*We have to break the data-set into several meta-cells. Their sizes should not be too small in order to minimize communica-

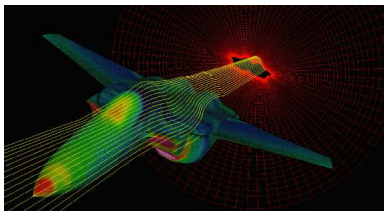

Figure 4: Symmetries in aircraft design can be used to steer the meta-cell construction.

tions between processes of the application due to the problem pictured on Figure 3. But what happens if the meta-cell size is larger than the main memory of the corresponding grid node?

Such problem can be avoided by taking into account the size of the smallest memory available in the pool of grid nodes. However, as we are going to use a meta-cell structure to distribute data over the grid, it is straightforward to use a similar structure to address this problem.

Specifically, we shall consider out-of-core methods; that is, techniques to deal with large data sets that do not fit completely in main memory [Shyh-Kuang Ueng and Ma, 1997, Chiang et al., 1998, Chiang et al., 2001]. Once a process receives a meta-cell larger than the main memory available, it starts to construct a (local) meta-cell structure with the goal of retrieving only the portion(s) of interest during the particle(s) path generation [Shyh-Kuang Ueng and Ma, 1997]. This local structure may be totally independent from the global one that the master keeps.

The data structure behind the sub-division is very important to efficiently retrieve data from disk. Octree and R-trees variants, among others, have been used for this purpose in the field of out-of-core algorithms for visualization [Shyh-Kuang Ueng and Ma, 1997, Chiang et al., 1998, Chiang et al., 2001].

c) *Particle Buffer:* A set of particles is assigned to each slave at the initialization. A slave will send new particles positions to the master only after all particles assigned to it undergoes a pre-defined number of integration steps or touch the partition boundary.

d) *Physically based heuristics to find out efficient meta-cell structures:* The heuristics used in item (a) are based on geometric elements. There are no such drives when the simulation takes place in unstructured meshes. Computational hemodynamics is a case-study of interest. (see references in http://virtual01.lncc.br).

In this field, given a surface that represents the inner (outer) artery wall, the complex phenomena associated with the blood flow can be simulated. The artery system of interest is the main one. The corresponding arterial tree has also a topological pattern, and there are rheological features that characterize the arteries as well as the blood. May be, up to some scale, such elements can inspire specific heuristics to steer the meta-cell structure construction.

(e) *Explore non centralized distributed methodologies:* In this case, each process must has a directory service. Thus, each process can send particles positions of another process. Such policy has the advantage of cutting the number of messages in half. However, there is an additional requirement per grid node to keep the directory service in memory.

(f) *Utilization of Autonomous Agents as a model to explore the data set:* For our context of interest, an autonomous agent can be considered as *a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.*

It is common that researches take in account some heuristics when analyzing the data sets generated. For example, Banks and Singer [Banks and Singer, 1995] uses the fact that vortices (roughly defined as rotating region inside a fluid) are characterized by high vorticity and low pressure.

That kind of prior knowledge could be incorporated in the agent's *agenda.* Thus, when the agent senses that its neighborhood satisfies the pre-defined rules, it could start self replications.

5.1 Methodology

Collaborative Virtual Environments: We aim at allowing a group of researches to collaboratively visualize data sets. Such collaboration will be implemented through a Collaborative Virtual Environments (CVE), where a communication layer is to be added to the visualization tool in order to allow local actions to be transmitted to the other parties, as well as to mirror the incoming actions which are

to be arriving from them. CVEs will be exploited initially in its simplest form of desktops augmented with virtual reality devices (data glove and HMD). Next, we intend to pushing forward to an immersive CVE in a CAVE environment.

Performance Evaluation: We expect to perform simulation experiments in a tool such as Opnet (http://www.opnet.com). Such simulation will aim at validating the architecture used, also resulting in the fine tuning of such architecture.

Selected components: We shall use Java3D (http://java.sun.com) for 3D visualization of particles. This would allow a platform independent implementation. Java 3D has been chosen due to both, the platform independency, as well as the fact that it transparently handles stereoscopic rendering of a 3D scene without the need to have much changes in the code.

Grid Resources: The LNCC Grid Project (http://netra01.lncc.br).

Middleware: In this work, we have decided to use Globus in our implementation (http://www.globus.org). As we already pointed out, Globus has been used not only to design collaborative visualization systems [Barnard et al., 1999, Park et al., 2000] but also to support distributed computational simulations in CFD [Barnard et al., 1999]. Thus, we expect to achieve support for both efficient resource/data access and task distribution.

6 Scientific Contributions

From the viewpoint of particle tracing methods for distributed memory environments meta-cells size should not be too small in order to minimize communications between processes. But, if the meta-cell size is larger than the main memory of a grid node, we will have performance problems. The bottleneck in this case, the smallest memory available, would be a serious limitation. The first contribution of our proposal is the application of out-of-core methods to avoid this problem.

Besides, the proposed architecture can support not only particle tracers but also isosurface and volume rendering methods described on section 4. It is just a matter of choosing a suitable meta-cell structure, like the one proposed in [Chiang et al., 2001].

On the other hand, dedicated clusters with closely coupled nodes and ad-hoc clusters, where computing nodes are volunteered and loosely coupled, can be integrated through Globus middleware. Another contribution of our proposal is to allow users to take advantage of the computational power of such integration.

7 Conclusions

The visualization of data sets in fluid flow is important for scientific and engineering applications. The data sets size (Gigabytes or Terabytes) poses challenges to the visualization methods.

In this paper we propose a distributed particle tracing system and its implementation for collaborative visualization environments in computational grids. We decided for Globus as a middleware, Java3D as graphics development language and out-of-core techniques to simplify the efficient design of meta-cell partitions.

The proposed architecture can support not only particle tracers but also isosurface and volume rendering methods, among others.

References

[Banks and Singer, 1995] Banks, D. and Singer, B. (1995). A predictor-corrector technique for visualizing unsteady flow. *IEEE Trans. on Visualization and Comp. Graphics*, 1(2).

[Barnard et al., 1999] Barnard, S., R.Biswas, Saini, S., der Wijngaart, R. V., Yarrow, M., and Zechtzer, L. (1999). Large-scale distributed computational fluid dynamics on the information power grid using globus. In *The 7th Symposium on the Frontiers of Massively Parallel Computation*, http://www.ipg.nasa.gov.

[Chiang et al., 2001] Chiang, Y.-J., Farias, R., Silva, C., and Wei, B. (2001). A unified infrastructure for parallel out-of-core isosurface and volume rendering of unstructured grids. In *IEEE*

2001 Symposium on Parallel and Large-Data Visualization and Graphics.

[Chiang et al., 1998] Chiang, Y.-J., Silva, C., and Schroeder, W. (1998). Interactive out-of-core isosurface extraction. In *IEEE Visualization '98*, pages 167–174.

[Coleman et al., 1996] Coleman, J., Goettsch, A., Savchenko, A., Kollmann, H., Wang, K., Klement, E., and Bono, P. (1996). Teleinvivo: Towards collaborative volume visualization environments. *Computers& Graphics*, 20(6):801–811.

[Dam et al., 2000] Dam, A. V., Forsberg, A., Laidlaw, D., LaViola, J., and Simpson, R. (2000). Immersive vr for scientific visualization: A progress report. *IEEE Computer Graphics and Applications*, pages 26–52.

[Foster et al., 2002] Foster, I., Kesselman, C., Nick, J., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed system integration. Technical report, www.globus.org/research/papers/ogsa.pdf.

[Foster and Kesselmany, 1996] Foster, I. and Kesselmany, C. (1996). Globus: A metacomputing infrastructure toolkit. Technical report, http://www.globus.org/.

[Hamann et al., 1995] Hamann, B., Wu, D., and II, R. J. M. (1995). On particle path generation based on quadrilinear interpolation and bernstein-bzier polynomials. *IEEE Trans. on Visualization and Comp. Graph.*, 1(3).

[Johnston et al., 2000] Johnston, W. E., Gannon, D., Tanner, B. N. L. A., Thigpen, B., and Woo, A. (2000). Computing and data grids for science and engineering. pages 70–71, http://www.sc2000.org/.

[Lane, 1994] Lane, D. (1994). Parallelizing a particle tracer for flow visualization. Technical report, http://www.nas.nasa.gov.

[Leigh et al., 1998] Leigh, J., Park, K., Kenyon, R., and Wong, H.-Y. (1998). Preliminary star tap tele-immersion experiments between chicago and singapore. In *HPCAsia 98*, http://www.evl.uic.edu.

[Lombeyda et al., 2000] Lombeyda, S., Aivazis, M., and Rajan, M. (2000). Making remote tools available for visualization of large data sets: Parallel isosurface calculation and rendering of large data sets in iris explorer. In *Visualization Development Environments 2000 Proceedings.*

[Ma, 1995] Ma, K. L. (1995). Parallel volume raycasting for unstructured-grid data on distributed-memory architectures. In *IEEE Parallel Rendering Symposium*, pages 23–30.

[Oliveira and Georganas, 2002] Oliveira, J. C. and Georganas, N. D. (2002). Velvet: An adaptive hybrid architecture for very large virtual environments. In *Proceedings of IEEE International Conference on Communications.*

[Pakstas and Komiya, 2002] Pakstas, A. and Komiya, R., editors (2002). *Virtual Reality Technologies for Future Telecommunications Systems.* John Wiley & Sons, LTD.

[Park et al., 2000] Park, K., Cho, Y., Krishnaprasad, N., Scharver, C., Lewis, M., Leigh, J., and Johnson, A. (2000). Cavernsoft g2: a toolkit for high performance tele-immersive collaboration. In *ACM 7th Annual Symposium on Virtual Reality Software & Technology (VRST)*, http://www.evl.uic.edu/.

[Rosenblum et al., 1994] Rosenblum, L., Earnshaw, R., Encarnacao, J., Hagen, H., Kaufman, A., Klimenko, S., Nielson, G., Post, F., and Thalmann, D. (1994). *Scientific Visualization: Advances and Challenges.* Academic Press.

[Sawant et al., 2000] Sawant, N., Scharver, C., Leigh, J., Johnson, A., Reinhart, G., Creel, E., Batchu, S., Bailey, S., and Grossman, R. (2000). The tele-immersive data explorer: A distributed architecture for collaborative interactive visualization of large data-sets. In *Proceedings of the Fourth International Immersive Projection Technology Workshop*, http://www.evl.uic.edu.

[Shyh-Kuang Ueng and Ma, 1997] Shyh-Kuang Ueng, K. S. and Ma, K. L. (1997). Out-of-core streamline visualization on large unstructured meshes. *IEEE Trans. on Visualization and Comp. Graph.*, 3(4).

[Stolk and van Wijk, 1992] Stolk, J. and van Wijk, J. J. (1992). Surface-particles for 3d flow visualization. In Post, F. H. and Hin, A. J. S., editors, *Advances in Scientific Visualization*, pages 119–130. Springer, Berlin, Heidelberg.

[van Wijk, 2002] van Wijk, J. J. (2002). Image based flow visualization. *ACM Transactions on Graphics (TOG)*, 21(3):745–754.