Immersive Hierarchical Visualization and Steering for Spectral/hp Element Methods

Jonathan Reiter, R.M. Kirby,
and Joseph J. LaViola Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-01-03
May 2001

Immersive Hierarchical Visualization and Steering for Spectral/hp Element Methods

Jonathan Reiter
Department of Computer Science
Brown University

R.M. Kirby
Division of Applied Mathematics
Brown University

Joseph J. LaViola Jr.
Department of Computer Science
Brown University

Abstract

We present the first steps in our hierarchical visualization and steering research targeted at large time-dependent spectral/hp element methods. This research is built around several observations concerning the structure of spectral/hp data which lead to the development of several hierarchical visualization techniques. These techniques provide an attractive vehicle for building a complete end-to-end hierarchical visualization and steering system. This paper discusses our techniques in conjunction with the Pilot system, a first implementation of these ideas in the form of an immersive hierarchical visualization and steering system for spectral/hp element simulations. Pilot allows the user to control the level of extracted and visualized detail on a local basis throughout the simulation's domain and provides the user with an immersive computational steering interface to interact directly with the simulation. This combination of techniques allows us to use Pilot to visualize the results of large concurrently-running spectral/hp element CFD simulations.

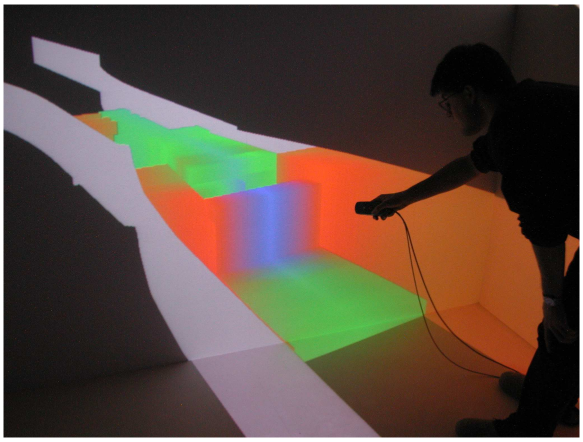

Figure 1: Standing in an immersive virtual environment a user examines flow through a nozzle in a 10 million degree of freedom simulation.

1 Introduction

Simulation scientists are often interested in visualizing the results of large simulations. But many scientifically interesting simulations involve more data than can be presented to the investigator using naive techniques and traditional tools. Further, we would like to enable a user to interact directly with a simulation code and steer the computation as it progresses.

Our approach is to take advantage of the special structure of spectral/hp element simulations and data sets to build a system that is capable of handling larger problems [9]. We present several specific techniques aimed at enabling interaction with large spectral/hp element simulations. The Pilot system was developed as a first implementation of these techniques; Pilot embodies the process of taking the first steps towards building an end-to-end hierarchical visualization and steering system specifically for spectral/hp element simulations.

1.1 Why Spectral/hp Element Methods?

We chose to work with spectral/hp element methods because there are three characteristics of these methods that prove useful for the development of a hierarchical visualization system. Spectral/hp element methods are built around traditional mesh-based spatial subdivision, providing a natural framework for partitioning the computational domain into a set of subdomains. This characteristic allows for selective visualization on an element-by-element basis.

The internal data representation of spectral/hp element methods also holds promise for large-scale visualization. Instead of generating value-at-vertex data like traditional finite element methods, spectral/hp element methods compute vectors of polynomial coefficients, each of which describes the values of an individual data field over the space of one element. The structure of these vectors is hierarchical: by considering more coefficients one gets a more accurate representation. Truncated coefficient vectors still provide a useful data representation, although they exchange space for accuracy. This tradeoff is very similar at a high level to common lossy image compression techniques. This characteristic allows control over the balance between accuracy and data size.

These polynomial coefficient vectors have one more property that is of interest when designing a large scale data visualization system. Because the computational methods produce coefficient vectors describing data fields in a continuous fashion, the visualization system has a free hand in selecting the points that will be sampled to produce visualizations. We can use the polynomial coefficient vectors to generate exact values at any location in the domain; we are not constrained to the vertices of the elements or to any other subset of points within the simulation's domain. This gives the visualization implementor the ability to construct a spatial hierarchy of points at which to evaluate the coefficient vectors - thereby introducing a mechanism to control the balance between visualization accuracy and geometry size.

It is these observations that motivated the design of the Pilot system for the immersive hierarchical visualization and steering of spectral/hp element simulations. Pilot implements our three hierarchical techniques - elemental, spectral and spatial - within a larger visualization and steering system. Pilot focuses on computational fluid dynamics (CFD) simulations and provides an immersive visualization front end in a Cave [3].

1.2 Paper Organization

The remainder of this paper is organized in the following manner. First we explain how our work relates to existing hierarchical and multiresolution visualization techniques. Then we describe our hierarchical techniques in detail.

The second half of the paper is devoted to detailing the Pilot system's architecture and presenting the immersive user interface we developed for interacting with simulations. After presenting the hierarchical methods and system design, we move on to detailing how Pilot has been applied to specific CFD problems and the results we have observed thus far. Finally we discuss areas for future work.

2 Related Work

There have been a number of approaches for building hierarchical and multiresolution data representations. A taxonomy for these methods is provided in [2, 12]. Within this framework, our three presented hierarchies are classified as follows: the elemental and spatial resolution hierarchies are space-based, while our spectral hierarchy is value-based. But the design of each of our presented hierarchical techniques was motivated by the structure of spectral/hp element simulation data.

Many hierarchical systems do not take data values into account at all [8, 10, 15, 19]. These hierarchies are purely geometric in nature because they were designed for rendering efficiency, not scientific data visualization. Techniques based primarily on geometric structure have been proposed for scientific visualization [4]. These methods work by adjusting the number of exact-valued points that are extracted from simulation data. In contrast, our spatial hierarchy selects the number of exact-valued points to generate given the extracted data. We leverage the characteristics of spectral/hp element data to develop a value-based extraction hierarchy that supports space-based hierarchical rendering techniques.

Value-based approaches built around wavelets have been successful in the construction of hierarchical visualization techniques. Trott [18] applies a wavelet transform to CFD data to construct this type of visualization. The spectral/hp element data representation provides similarly transformed data as its initial output; consequently, our approach does not require this additional step. Other wavelet-based approaches exhibit similar behavior [17, 24]. Finally, Grosso [5] presents an automated hierarchical refinement technique for finite element CFD simulation data. However, this technique is not interactive and we want the user to have interactive control over the hierarchies to enable exploratory visualization.

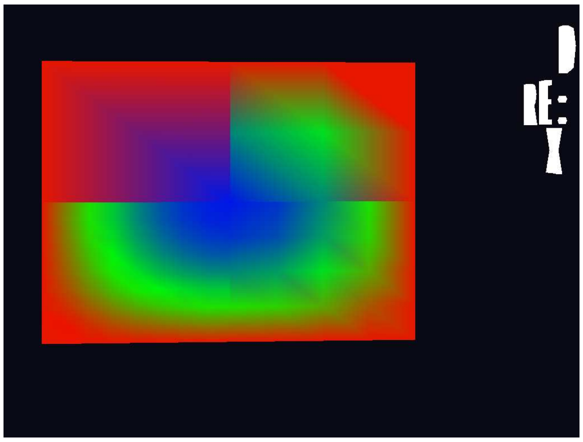

Figure 2: Four elements displayed with different tessellation orders. Moving clockwise from the upper left, the elements have tessellation order 1,2,3 and 4.

3 Hierarchical Data Techniques

Pilot provides the user with three hierarchical techniques to control the tradeoff between accuracy and size for simulation and visualization data. Our first level hierarchy and control mechanism works at the element level. The user has the ability to selectively visualize and extract data from the simulation on a per-element basis.

The second and third hierarchical techniques are dependent on the first. We consider them to operate independently over the set of active elements.[1] These hierarchical mechanisms provide control over data extraction and rendering accuracy. In both cases the user can adjust the balance between data size and accuracy.

3.1 Element Level Hierarchy

Any element-based method lends itself to a simple one level hierarchical representation: selectively enabling or disabling the visualization of different elements. While this is not a particularly complex hierarchical visualization technique, it is an important one. When we talk of large data sets it is often the case that simply displaying the complete spatial discretization of the domain involves more geometry than we can interactively render.

This work targets data sets larger than 1 million degrees of freedom. For these cases simply displaying the complete computational mesh can involve in excess of 1 million triangles. Larger data sets can lead to orders of magnitude more geometry - enough to swamp many hardware rendering systems. In these cases a first-level elemental hierarchy provides a way to visualize data sets whose coarsest representation could otherwise overwhelm even the fastest of visualization systems.

3.2 Spectral Data Hierarchy

Beyond the simple notion of selecting which elements are of interest, our hierarchical representation takes advantage of the structure of spectral/hp element data to build a deeper data hierarchy around the simulation's internal polynomial

[1] We refer to the set of currently extracted and visualized elements as the "active elements".

coefficient representation. This representation consists of one coefficient vector for each data field for each element. It is these vectors that lend themselves to a hierarchical interpretation.

The basic operation available to the user is the selection of the number of coefficients to extract from the simulation code. While the simulation computes a coefficient vector of fixed size, one can use fewer than all of the coefficients to generate a less accurate representation for the data over an element. In exchange for this loss of accuracy, leaving out coefficients saves space. We maintain the number of coefficients to use on a per-element basis throughout the computational domain. The user can then use this hierarchy to extract more coefficients only for sections of the domain that require a more accurate representation. This increased local accuracy does not force the system to deal with large quantities of data for elements where a coarse representation provides sufficient information.

Our data extraction technique considers information in units of "order". In a one-dimensional system, the addition of one more coefficient increases the order of the polynomial expansion by one degree. In multi-dimensional systems increasing the modal order one degree requires more than one additional coefficient; all coefficients corresponding to polynomial terms of the same degree are required. For example, in two dimensions, if we presently have constant and first-order information as coefficients of $(1,x,y)$ and we desire to increase the order by one degree, we require coefficients to be extracted for the three additional terms (x^2, y^2, and xy). Extracting only one more coefficient would lead to a polynomial expansion which was not complete for a particular polynomial order.

Providing this freedom to the user has consequences. A user may be visualizing two adjacent elements where one is being extracted at 10th order and the other at 5th order. When these imbalances are present the data displayed on the boundary between elements may be discontinuous.

This is not the only potential source of visualization discontinuity here. The simulation that we employ uses the discontinuous Galerkin formulation which does not require continuity - only L^2 boundedness - across element interfaces. The users of this sort of system will likely understand the mathematical formulation of the simulation being visualized, so they are unlikely to be confused by the discontinuity. The visualization issues associated with discontinuity are beyond the scope of this paper.

3.3 Spatial Hierarchy

We consider data on an element-by-element basis and display it using elemental boundary surfaces. These surfaces are polyhedral subsets of the simulation's domain and are visualized as polygon meshes. We would like to provide the user with hierarchical control over these meshes so they can interactively adjust the balance between geometry size and visualization quality.

Our initial set of polygons is just the set of polygons bounding the element under consideration. This provides a functional but coarse spatial discretization on which to display the data values. The extracted polynomial coefficients are used to compute exact values for each vertex, these values are color mapped, and the polygons are fed to the rendering system.

From the simulation's perspective, there is nothing special about the points where we choose to evaluate the coefficients. Spectral/hp element methods provide continuous representations for data fields so we have free reign within an element to pick points.

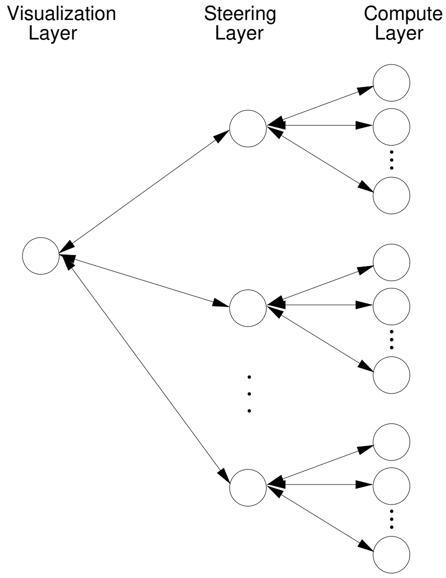

Figure 3: A graph view of the Pilot network architecture. A single visualization processor is attached to an arbitrarily large number of compute processors through the intermediate layer of steering processors. In turn, each steering processor handles data extraction and the computational steering operations for at least one compute processor.

For our initial condition - using only the vertices of an element's bounding polygons - the rendering system will apply an interpolation procedure between adjacent vertices to produce an image. To produce more accurate images we tessellate the bounding polygons to a user specified degree and evaluate the polynomial coefficients at each resulting vertex. This can greatly increase the number of points at which we have exact values, although naive interpolation is still used between the exactly-valued vertices.

Our tessellation procedure simply recursively subdivides each face of each currently displayed element with recursive depth specified by the users "tessellation order" setting for the element under consideration. Increased tessellation order produces more accurate visualizations at the expense of increased geometry size. We give the user control over this balance and enable them to decide how to trade rendering performance for visualization accuracy.

4 Pilot Architecture

The steering contribution of this research is not a novel approach to computational steering per se, but rather the steering that is enabled by our hierarchical data visualization techniques. The problem of interactive data visualization is a necessary prerequisite to computational steering [21]. Our hierarchical visualization work applies to non-steering applications as well; we felt that building a steering system was the most effective way to demonstrate the utility of our visualization techniques. Pilot employs steering techniques similar to other computational steering architectures [6, 7, 11, 13, 20].

The overall layout of Pilot is that of a three layer tree, as depicted in figure 3. We consider the simulation itself to be the first layer, consisting of an arbitrary number of processors. The third layer is the visualization front end. Each of these layers includes network communication hooks that allows them to be attached to Pilot.

In between these two layers the steering layer exists as a parallel program running on any number of processors between 1 and the number of compute processors. Data extraction takes place between the steering and simulation layers. This process involves communication only between a steering processor and each computation processor it steers. Across steering processors this procedure is parallel. The steering processors only communicate with each other to synchronize their extraction in between simulation time steps.

The visualization layer then reads renderable data out of the steering processors and presents it to the user. Interaction with the user is accomplished through the visualization layer, which passes messages back through the steering layer, eventually to modify the simulation's state.

4.1 Compute Layer

The primary function of the compute layer is to generate the data to be visualized and steered. The flow solver corresponds to a particular version $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$, a general purpose parallel spectral/hp element CFD code for simulating incompressible, compressible and plasma flows in unsteady three-dimensional geometries. The major algorithmic developments are described in [9] and [22]. The code uses meshes similar to standard finite element and finite volume meshes, consisting of structured or unstructured grids or a combination of both. The formulation is also similar to those methods, corresponding to Galerkin and discontinuous Galerkin projections for the incompressible and compressible Navier-Stokes equations, respectively. For this work we use the compressible solver because characteristics of its implementation made the modifications we needed easier to implement.

Pilot's data extraction system was designed with the computational needs of large simulations in mind. To this end the extraction mechanism was designed to be minimally intrusive on the simulation. It does little more than copy out arrays of data and make the necessary function calls into the simulation to implement steering. Data is extracted in the spectral/hp element representation internal to the simulation. These collections of data are the hierarchical extracts that form the core data flow of the Pilot system.

This layer connects to the steering layer for two reasons. It exposes simulation data for extraction, and provides an interface for implementing simulation steering. The compute layer maintains minimal state - instead of remembering the current visualization state it answers requests for data from the steering layer.

4.2 Visualization Layer

Data is displayed and user input is collected by the visualization layer. While there are benefits to maintaining the spectral/hp element data representation during data extraction, the visualization system cannot easily handle these coefficients as input. The Pilot visualization system takes raw renderable data as input over the network: collections of triangles with values specified for each vertex.

Pilot's visualization front end only receives geometry corresponding to the pieces of the domain the user has requested, and then only at the user specified resolutions. As the user inputs commands to change the subset of data to

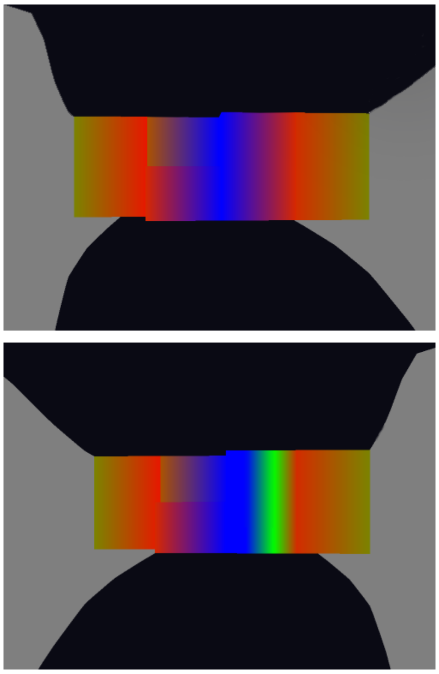

Figure 4: Here we see a cross-sectional cut of flow through a nozzle visualized at different levels of spectral order. The top image uses only a second order representation while the bottom image was generated with a 3rd order representation. At this higher level of detail the boundary layer between jet and near-wall downstream flow is resolved.

be displayed, the visualization system forwards these commands on to the steering processors. That layer remembers the current state and replies with the requested data so it can be displayed.

The visualization system is built around Sense8's World Toolkit (WTK) [23] , and communicates via standard WTK callback mechanisms.

4.3 Steering Layer

In between the compute and visualization layers is the steering software, performing those functions that are necessary to convert between rendering and simulation representations. The steering processors in Pilot use a double buffered system to insure that only complete extractions of data are ever exposed to the visualization side. A similar approach is employed to insure that the steering commands entered for a particular time step of the simulation are passed back to the simulation as a single block.

For visualization, the steering processors have two responsibilities: remembering the current extraction state - which elements should be displayed and and what resolutions - and converting the simulation's internal representation into the (vertex,value) tuples that the visualization system takes as input. By interactively adjusting the extracted resolution of data, the user is controlling the accuracy of the conversion between representations that runs on the steering processors. And as we would expect, a more accurate conversion requires both more space and time.

The role the steering processors play for computational steering operations is similar: they simply convert steering commands from the representation used by the visualization system to one the simulation can handle and forward that data to the compute layer.

It is important to state that both of these conversions - for visualization and computational steering - are run in parallel on the steering processors with no communication among them. Since the steering processors are nearly autonomous we expect to see good scalability for highly parallel simulations - those using 100s of processors or more.

5 User Interface

5.1 Environment

Pilot's user interface is implemented in the Cave facility at Brown University's Technology Center for Advanced Scientific Visualization and Computation. This approach is similar to that taken by [14]. We chose to use an immersive display for three reasons. First, there is excellent connectivity between the large compute servers we planned to use for running the CFD simulations and the graphics systems driving the Cave. Second, as already stated, the ability to easily select locations in space makes the implementation of many of Pilot's features easier. And lastly, we subscribe to the hypothesis that users will be better able to understand large data sets in immersive environments than on the desktop [16].

5.2 User Interaction Tools

All user interaction in Pilot is accomplished through three dimensional widgets in a virtual environment. Hand-following tools are used for the three basic visualization-changing operations:

- selecting which elements to display
- adjusting the extracted order of an element
- adjusting the tessellation order of an element

In each case, the user selects a tool, presses one of two buttons (on/off or increase/decrease) on a wand, and drags the device around the virtual space. The operation selected by the user is applied to every element the user drags the tool through while the button is depressed.

Steering is implemented using a single tool, a virtual laser pointer, and a collection of changeable three dimensional text fields. Pilot includes support for interactively steering three scalar values in the coupled simulation: Reynolds number, time step and wall temperature. These values are displayed on the right wall of the Cave in 3D text and maintain a constant position relative to the user (i.e. they are always located on the right wall independent of the user's navigation around the virtual space).

This interface is also used to control visualization parameters in some cases. The user can change the currently displayed scalar field using this same laser pointer on another piece of 3D text where the increase and decrease controls cycle through the list of available fields.

The various user interface components are all implemented entirely within the virtual space. We feel this characteristic is essential; if users can explore data in an immersive setting they should not be forced to go back to the keyboard when it comes time to change parameters.

5.2.1 Color Mapping Technique

It is unclear how to handle maintaining a value to color map for concurrently generated time-dependent data. Each new time step provides a new range of values, possibly wholly outside the value range for the previously displayed time step. Pilot allows the user to change which elements are being displayed from time step to time step. There is no guarantee that value ranges will even be similar as the user selects different sets of elements.

We experimented with an adaptive color mapping technique that automatically expanded its range of value as new data arrived. This approach had the unexpected side effect that with new maxima and minima in the data, coloring could change significantly even in regions where the data values remained constant. This behavior can present the illusion of features due only to a changing color map.

To counter this problem we adopted a user-controlled remapping procedure. A new user interface tool was added to the visualization system that performs only one operation: resetting the color map to handle the range of currently display data with a constant padding slight variations. This approach allows the user to turn on the elements of interest and, at any time in the visualization process, choose to adjust the color map to fit the currently selected data.

This approach has the advantage that automatic color map changes do not occur and the problem of illusory changes disappears. On the other hand it means that if new data comes in with a new time step that lies outside the current mapping range, it will not be mapped properly. Our system assigns the nearest value to values outside the extrema of the map: the color assigned to the maximum or minimum mapped value. Data outside the current range will appear entirely in one color.

This solution worked well, but we believe additional research is required to develop a more robust solution.

5.3 A Networked UI

All of our non-steering user interaction tools require computing which element contains an arbitrary location in space selected by the user. But the visualization system in Pilot does not have any information about the grid structure of the computation - it only has lists of graphical objects to display. We do not want to transmit the entire computational mesh to the visualization software as it is unnecessary and potentially large. We would prefer to run this computation on a processor that needs the information anyway, and ideally in a parallel setting to accelerate the search.

Pilot achieves all of these goals by implementing interactive widgets in two pieces and across two different Pilot network layers. The user interface simply reports a list of locations and actions to all of the pilot processors. Those processors are responsible for searching their local subdomain - thereby searching the entire domain in parallel - to map locations to elements.

With this information Pilot can forward a list of elements and actions to the relevant compute processors. This technique allows us to retain the desired interactive behavior of the widgets by not burdening the visualization system with costly searches and still allows the user to interact with very large computations.

The cost of this technique is latency. After the visualization software reports the lists of locations and actions to Pilot, it does not get new graphics immediately. The requested changes are not included in the visualization until the next time step of data runs through the system. Pilot makes a conscious choice to trade this sort of latency for the ability to interact with larger simulations.

6 Results

After designing and implementing these visualization techniques, we used our system for the visualization and steering of current research CFD simulations. Since this work targets large simulations, we chose to test Pilot at four points along the path to large data. In each case we assessed the ability of Pilot to handle the given simulation and visualization using an SGI Onyx2 InfiniteReality for visualization and at most 64 processors of an IBM SP2 for simulation.

6.1 Duct Flow: Small Simulation

Our small simulation is flow through a duct. This simulation has roughly ten thousand degrees of freedom and the fluid dynamics are well understood. Our hierarchical techniques were not needed at this level since the data set is small enough to enable full resolution visualization.

However, the fluid behavior for this example is simple enough to explore our steering functionality. The user has the ability to interactively change the Reynolds number, wall temperature and simulation time step size through Pilot. We were able to see the effects of these steering operations clearly with our visualization system. This determination was carried out by the developers of $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$.

6.2 Aerofoil: Mid-Size Simulation

Our next step up was an aerofoil simulation with roughly one million degrees of freedom. In this case the wing geometry itself consisted of nearly ten thousand triangles. Displaying all of the data would require between one and ten million triangles, overwhelming the capabilities of our rendering system for interactive purposes.

This case presents the first instance where our hierarchical methods truly enabled interactive visualization; specifically, the spatial and elemental hierarchical representations proved valuable. We could handle transporting all of the data over the network among the various Pilot components since the data set size was still only 8 MB per time step. [2] The effect of extracting a lower order representation was negligible here; reducing the quantity of geometry was the primary issue.

For this sized simulation we found that our steering operations produced no visibly apparent results. This is due to a confluence two factors: the time scale on which the phenomenon changes and the amount of resources available to us for running the simulation.

6.3 Nozzle: Large Simulation

To test the utility of the spectral data hierarchy we moved up one order of magnitude to a 10 million degree of freedom simulation. This simulation generates 80 MB per time step. While transporting this sized data set is not a problem for a user willing to wait a few seconds, we would prefer to not introduce that additional lag between the simulation generating results and the user viewing the corresponding visualization.

As predicted, decreasing the accuracy of the extracted representation did improve performance. The simulation itself required several seconds to compute each new time step, and our hierarchical techniques allowed us to extract the data within this window. The spectral hierarchy provided a modest improvement for this sized data set.

The elemental and spatial hierarchies proved essential. An individual feature in this simulation - a jet of fluid for example - can occupy several hundred elements. Displaying this many elements with a dense spatial discretization can easily generate several hundred thousand triangles, overwhelming the rendering system's ability to generate graphics quickly enough to maintain the immersion.

6.4 F-15: Very Large Simulation

Our largest test simulation was an F-15 airplane. This is an over 100 million degree of freedom simulation - nearly 1 GB of data per time step - with roughly 250,000 elements. Now, in addition to swamping the rendering system with triangles, we can swamp our communications system with data. When we tried to extract data at full resolution from this simulation it took longer than the computation itself.

The spatial hierarchy cannot help this problem because it only effects the amount of generated geometry, not the size of the extracted data. While our elemental extraction control helps here, even small subsets of the data approach 100 MB. Our spectral data hierarchy proved useful in this case by the user to see a coarse visualization with several updates per second.

For this size simulation we were still able to use our steering tools, but we were not able to quickly observe changes in the fluid's behavior. This is not at all surprising given our available computational resources and we only undertook this experiment to demonstrate the scalability of our steering system.

7 Future Work

Several points in the current implementation of Pilot deserve further thought and work. Our set of hierarchies has proved useful for controlling both data and geometry size, but it seems likely that one could do a better job on both fronts by developing adaptive tessellation algorithms to generate geometry in a way that requires fewer vertices and coefficients to achieve the same error between data and visualization.

With an intelligent tessellation algorithm it would be possible to develop hierarchical versions of other common visualization widgets like cut planes. In the current system of regular tessellation, cut planes would be rendered with linear interpolation across element boundaries - a clear source of substantial error. Proper hierarchical tools would tessellate adaptively around element boundaries to minimize the error in the visualization.

It is also likely that isosurface and streamline algorithms can be developed that exploit the structure of spectral data. For streamlines, there is potential in the hierarchical data representation for developing time-critical streamline algorithms that can trade off the accuracy of the sampled data points for faster update rates [1]. One can also imagine developing time-critical isosurface algorithms along the same lines - or moving in the opposite direction to develop high-order accurate isosurfaces that will undoubtedly take more time to compute.

All of these ideas will generate more geometry for the visualization system to render. And as we have already

[2] Data points in $\mathcal{N}\varepsilon\kappa\mathcal{T}\alpha r$ are stored as double precision floating point values.

stated, the dynamic nature of the geometry in Pilot makes it difficult to apply render-accelerating techniques. If the geometry transmitted to the visualization system already included structures like triangle strips, rendering could be significantly faster. Such a procedure would have the nice property that geometry optimization is done in parallel across the steering processors. Of course it would also involve including visualization-specific techniques in parts of Pilot other than the visualization front end. This increases complexity and potentially decreases the extent to which Pilot can be attached to other front end systems.

The current user interface is far from being ready to present to scientists as a tool for doing CFD research. We see that significant user interface research is needed to help build a better Pilot in two areas. The latency introduced by spreading the user interface across two layers of the network introduces some "dead time" between a user's visualization modifying action and the change actually appearing in the visualization. This time can probably be used productively in some way; at the least we should develop a method for providing the user with status updates indicating which commands are still being processed and how much more processing time they will require.

Our second area of user interface work concerns the system in a broader sense. From the use of one Cave wall for steering through the spatial point-and-click approach we use to select locations in space, our current user interface was developed to explore the possible utility of our hierarchical visualization techniques. Now that we have accomplished this first goal, the user interface should be revisited with the usability requirements of our target end users - simulation scientists - in mind.

Lastly, we would like to extend our spectral hierarchy to steer the simulation itself. Giving the user interactive control over the computed order on a per-element basis would provide even greater control over the simulation. Along these lines, we would also like to allow the user to interactively modify the computational mesh by changing the geometry of the domain. Both of these changes involve making substantial modifications to the simulation and are the subjects of longer-term simulation research.

8 Acknowledgements

We acknowledge the support and computational resources provided to us by the Brown University Technology Center for Advanced Scientific Computing and Visualization. All authors acknowledge the support and advice of our advisor Prof. Andries van Dam. The second author acknowledges the support and advice of his Ph.D. thesis advisor, Prof. George Em. Karniadakis. We thank Dr. Tim Warburton for his assistance, Igor Pivkin for providing mesh assistance to us, and Andrew Forsberg for his input towards the scientific visualization issues in the research. We also thank Robert Zeleznik for his comments on a draft of the paper. Finally, we thank Mark Oribello for his assistance in the production of our video.

References

[1] Steve Bryson and Sandy Johan, *Time Management Simultaneity and Time-Critical Computation in Interactive Unsteady Visualization Environments*, Proceedings of Visualization '96, 1996, pp. 255–261.

[2] Paolo Cignoni, Paola Marino, Claudio Montani, Enrico Puppo, and Roberto Scopigno, *Speeding Up Isosurface Extraction Using Interval Trees*, IEEE Transactions on Visualization and Computer Graphics **3** (1997), no. 2, 158–170.

[3] C. Cruz-Neira, D. J. Sandin, and T. A. Defanti, *Surround-Screen Projection Based Virtual Reality: The Design and Implementation of the CAVE*, Proceedings of SIGGRAPH '93, 1993, pp. 135–142.

[4] Lori A. Freitag and Raymond M. Loy, *Adaptive, Multiresolution Visualization of Large Data Sets using a Distributed Memory Octree*, Proceedings of SC99: High Performance Networking and Computing (Portland, OR), ACM Press and IEEE Computer Society Press, 1999.

[5] Roberto Grosso, Christoph Lürig, and Thomas Ertl, *The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data*, Proceedings of Visualization '97, 1997, pp. 387–394.

[6] Weiming Gu, Greg Eisenhauer, Eileen Kraemer, Karsten Schwan, John T. Stasko, Jeffrey Vetter, and Nirupama Mallavarupu, *Falcon: On-line Monitoring and Steering of Large-Scale Parallel Programs*, Proceedings of the 5th Symposium of the Frontiers of Massively Parallel Computing, McLean, VA,, 1995, pp. 422–429.

[7] R. Haimes, *pV3: A Distributed System for Large-Scale Unsteady CFD Visualization*, In AIAA 32nd Aerospace Scienced Meeting and Exhibit, number AIAA 94-0321, 1994.

[8] Hughes Hoppe, *Progressive Meshes*, Proceedings of SIGGRAPH 96, ACM Press, 1996, pp. 99–108.

[9] G. E. Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for CFD*, Oxford University Press, January 1999.

[10] R. Klein, G. Liebach, and W. Strasser, *Mesh Reduction with Error Control*, Proceedings of Visualization '96, 1996, pp. 311–318.

[11] J. Kohl, P. Papadopoulos, and G. Geist, *Cumulvs: Collaborative Infrastructure for Developing Distributed Simulations*, Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing, 1997.

[12] Kwan-Liu Ma, Michael Cox, Steve Parker, and William J. Schroeder, *System Design for Visualizing Large-Scale Scientific Data*, SIGGRAPH '99 Course Notes No. 9, 1999.

[13] Michelle Miller, Charles Hansen, and Christopher R. Johnson, *Simulation Steering with SCIRun in a Distributed Environment*, PARA, 1998, pp. 366–376.

[14] Jurriaan Mulder, Robert van Liere, and Jack van Wijk, *Computational Steering in the CAVE*, Future Generation Computer Systems **13** (1998), no. 2.

[15] W. Schroder, J.A. Zarge, and W.E. Lorenson, *Decimation of Triangle Meshes*, Proceedings of SIGGRAPH 92, ACM Press, 1992, pp. 65–70.

[16] Paul H. Smith and John van Rosendale eds., *Data and Visualization Corridors*, Technical Report CACR-164. Center for Advanced Computing Research, California Institute of Technology, September 1998.

[17] H. Tao and R. J. Moorhead, *Progressive Transmission of Scientific Data Using Biorthogonal Wavelet Transform*, Proceeding of Visualization '94, 1994, pp. 93–99.

[18] Aaron Trott, Robert Moorhead, and John McGinley, *Wavelets Applied to Lossless Compression and Progressive Transmission of Floating Point Data in 3-D Curvilinear Grids*, Proceedings of Visualization '96, 1996, pp. 385–388.

[19] Greg Turk, *Re-Tiling Polygonal Surfaces*, Proceedings of SIGGRAPH 92, ACM Press, 1992, pp. 55–64.

[20] Jarke J. van Wijk, *CSE: A Modular Architecture for Computational Steering*, Proceedings of the 7th Eurographics Workshop on Visualization in Scientific Computing, 1996.

[21] Jeffrey Vetter and Karsten Schwan, *Models for Computational Steering*, Tech. Report GIT-CC-95-39, Georgia Institute of Technology, 1996.

[22] T. C. Warburton, *Spectral/hp Methods on Polymorphic Multi-Domains: Algorithms and Applications*, Ph.D. thesis, Brown University, Division of Applied Mathematics, 1999.

[23] *World Toolkit. http://www.sense8.com/*, 2001.

[24] Zhifan Zhu, Raghu Machiraju, Bryan Fry, and Robert Moorhead, *Wavelet-based Multiresolutional Representation of Computational Field Simulation Datasets*, Proceeding of Visualization '97, 1997, pp. 151–158.