Visualization in a Parallel Processing Environment

Robert Haimes*
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology, Cambridge, MA 02139
haimes@orville.mit.edu

and

David E. Edwards†
United Technologies Research Center
East Hartford, CT 06108
edwardde@utrc.utc.com

Abstract

An interactive visualization system **pV3** *(parallel Visual3) is being developed for the investigation of advanced computational methodologies employing visualization and parallel processing for the extraction of information contained in large-scale transient engineering simulations. Visual techniques for extracting information from this data in terms of cutting planes, iso-surfaces, particle tracing and vector fields are included in this system. This paper discusses the current design of* **pV3** *'s software and future applications for visualization in a parallel processing environment.*

1 Introduction

The computer network is becoming US industry's supercomputer. Computational simulations are being developed today that run on clusters of workstations or parallel computers. The next step is to integrate visualization directly into the simulation to view the results as they are generated.

As we approach the 21st Century, the engineering community is witnessing a tremendous growth and availability of large scale computing applied to analysis and design. Many of today's advanced computer simulations create data-sets containing billions of pieces of information. The sheer size of this data results in an exceedingly difficult and time consuming analysis process. The task of interrogation and interpretation of this information is required so that the knowledge contained within the simulation can be extracted. The problem is becoming more significant as improvements in computational performance result in these large scale simulations becoming more commonplace. Today, computational performance is increasing an order of magnitude every 3.5 years. Simulations once used only for exploration are now available more frequently during design and parametric studies.

Traditional interactive visualization is used to probe the data in order to locate and identify physical phenomena, or to identify limitations in the simulation process. However as the frequency of the large scale simulations increase in the design process, new approaches must be developed to enable the design engineer to process the information in a timely fashion. Specifically there needs to be closer integration of the traditional analysis stages (pre-processing, solver and post-processing). One scenario is to employ visualization throughout the simulation process.

For example, viewing the results of a CFD solver, as the solution progresses, has many benefits. If this window to the solution is through a visualization system designed for this class of problem, then that allows debugging of steady-state solvers at a macro scale. Solution quality assessment and run-time intervention are possible if the current results are easily accessible. Also, co-processing visualization of the results of transient applications relieves many resource limitations posed by these cases.

Solution steering is, of course, possible if one can watch the results of an unsteady simulation as they progress. Though it has always been a vision of many CFD researchers, today most 3D transient simulations do not allow solution steering. This is because most systems do not provide co-processing for the CFD application. Any system that requires concurrent visualization with the simulation must resolve the following difficult issues:

Hardware. A 3D transient CFD simulation requires super-computer class hardware (which includes parallel machines or a high-speed network of workstations) in order to

---

*Principal Research Engineer, Member AIAA
†Principal Scientist, Member AIAA

get results in a timely manner. Any concurrent scheme that cannot be run with these machines will not be used. Suggested steering techniques (such as Burnett et. al. [1]) that require recasting the solver in new language assume that the resources required to generate the data are small, or the time invested in the solver is also small and therefore easily rewritten.

Tool-kit. Integrating the visualization to the solver must be simple but general. The visualization tool-kit should provide a complete CFD viewer. Vaziri and Kremenetsky [2] discuss using CM/AVS on a TMC CM-5 for watching the results from a 3D simulation. Steering is possible by the use of constructed widgets within the AVS [3] interface. This work, though innovative, cannot be generally applied. The technique requires a CM-5 and the entire scheme suffers from the data handling inefficiencies associated with AVS.

Coupling. The co-processing scheme must be able to fit in, or adapt, to the solver. Usually a great deal of effort goes towards making the solver run efficiently. The visualization of the results must be subservient to the solution scheme and easily coupled. An interesting example of an attempt to do this is the code **CUMULVS** [4] which uses PVM and AVS as the visualization system. Unfortunately for large scale problems the entire field must be transmitted to the viewer.

The visualization tool-kit **pV3** [5, 6] is an attempt to deal with these problems. It also provides a fairly complete match to the type of analysis done (i.e. exploratory or production).

2 pV3

pV3 was originally designed to provide an interactive platform for the visualization of unsteady Computational Fluid Dynamics (CFD) calculations on unstructured as well as structured meshes. It has been successfully used in cluster environments using PVM for data movement and process management. For the first **pV3** releases the visualization software consisted of two parts, see Fig. 1. The client portion executes on the compute engine(s) closely coupled with the solver. This allows the reduction of the distributed data-set, in-place, to manageable lower-dimensional "extracts". The distilled data is transferred to the interactive server portion of **pV3** (executing on a graphics workstation) where the rendering and display are performed. This unique architecture frees up the compute engine(s) from dealing with the graphics, and reduces the large data-set to a size that can be rapidly transported to the server using standard Ethernet or FDDI connections. These early versions of **pV3** are in use in the parallel CFD community with applications running on clusters of workstations and parallel systems such as the IBM SP2 using a small number processors (tens of nodes).

pV3 has also been used in ways not originally anticipated. These include:

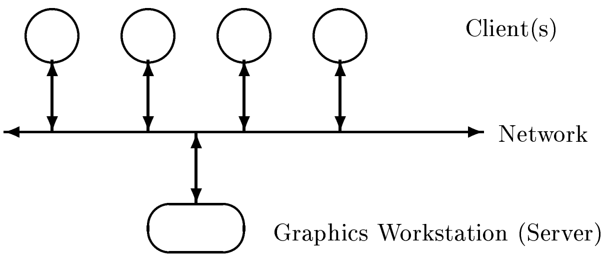

Figure 1: **pV3**'s Original Architecture

- Debugging/Data Validation
  Viewing the domain decomposition is simple, even for unstructured meshes. The internal surfaces can be treated like any external (domain) surface. In addition, the partition number can be loaded as a scalar field variable and then cut in a geometric manner.

  Message-passing can be checked at a macro level. An iso-surface can be selected to pass through any field primitive. If the generated surface does not match at the internal surfaces, then there is a mismatch in updating the field. The physical location of the mismatch can be used to locate the problem within the message-passing code.

  Watching the simulation go unstable and diverge is possible because the visualization is run concurrently with the solver. The physical location and error modes can be informative in locating and correcting problems.

- Efficient Use of Computer Resources
  Some steady-state simulations are now much larger than can be viewed by traditional post-processing CFD visualizers running on desktop workstations. Because **pV3** splits the viewer from the data extraction, the machine that was used for the simulation can also be used for the visualization. Simulations of order 10 million cells can be viewed at the engineers seat with a workstation having a minimal amount of memory (64 Mbytes).

- Quality Control Management
  pV3's ability to *plug-in* to and *unplug* from a long running solver allows the monitoring of the simulation's progress. This permits the early termination of a simulation gone astray. Steering is also possible to attempt to get the simulation back on-track.

Three deficiencies in this interactive distributed software model were reported by users. First, it was noted that the visualization process did not scale well and could not be run on some MPPs (Massively Parallel Processors). Second, the system could not easily be used in an industrial environment as compared to a research environment. In an industrial environment, the engineer will run many simulations during an engineering design or parametric analysis.

This production running of the simulation highlighted an obvious weakness in using interactive visualization for long-running simulations. Finally, since **pV3** is a visualization subsystem that can be integrated into other applications, users wanted more control over the design of the Graphical User Interface (GUI). This control allows **pV3** to have the same *look and feel* as the rest of the analysis suite. These problems and requirements have been addressed by adding new functionality to the **pV3** suite of software.

2.1 Problem – Scaleablity

The message-passing model of **pV3** with PVM, requires that the graphics workstation be accessible off the MPP with PVM's 'group' functionality intact (this is a problem for both the CRAY T3D and the Intel Paragon). Also, for most installations of IBM SP2s the Ethernet interface may be the only avenue to the graphics workstation. This may require that the simulation run on the slower Ethernet network between the nodes instead of the high-speed interconnect (particularly if the solver also uses PVM for message-passing). Not running over the switch poses significant performance penalties in comparison to on-switch bandwidth and user-mode latencies.

Ethernet, specifically, and TCP/IP communication in general, is the major contributor of performance problems that affect the visualization and therefore the solver when coupled together. Each Ethernet packet may encounter as much as 1 millisecond of latency. **pV3** (at Rev 1.10) required at least two packets per "extract" per client; a request for the data from the server and then the resultant data back from the client. An "extract" may be the simulation results on domain surface, an iso-surface, the results from a geometric cut, a single streamline or the particles currently active. If the job uses 16 clients and the investigator is viewing only 10 "extracts" then 0.32 seconds are wasted in latency alone. The other problem with Ethernet is the lack of bandwidth that is improved with other "wires" used in TCP/IP supported networks (such as 100Base-T, FDDI or HIPPI).

One of the goals of **pV3** is to provide an efficient software environment for distributed CFD co-processing applications that shows continued improvement to high-end scaleability. In order to solve the latency, bandwidth and PVM 'group' problems for MPP platforms, a new module has been added to the **pV3** software suite. This "concentrator" executes on the front-end of the MPP (or a node that has an appropriate off-machine high-speed network interface). All **pV3** messages on the machine use the high-speed interconnect via the MPI protocol. Communication off the MPP is performed by the concentrator using PVM to the graphics workstation in a point to point manner. Off-MPP communication is further improved by the concatenation of smaller messages at the concentrator to avoid the accumulated latency encountered by many packets.

Only a minor change to the **pV3** client-side API was needed to support the addition of this new software component. This change was required because the MPI specification [7] does not define how to start a mix of tasks (the solver components and the concentrator). Some MPI executives (or startup scripts) assume that all the tasks to be started are the same, making the mix impossible.

To enable the concentrator to start, a call was added to the **pV3** client side API that allows one task to be specified as the concentrator (and does not return for that task). Also a new MPI communicator is returned from this call for all other tasks. This communicator does not have the concentrator included and must be used for the solver's communication (instead of MPI_COMM_WORLD). A different network module is required at solver link in order to use MPI (instead of PVM). When the server starts at the graphics workstation, the concentrator acts like a normal **pV3** client, but passes through the requests to the actual clients and transmits back the requested data. When **pV3** reports any client data (such as from the point-probe) the actual client-ID is displayed (not the ID from the concentrator) for proper feedback to the investigator.

Another advantage to this network mode is the ease of startup. Normal clients need PVM functioning at the call to pV_Init. If the PVM daemon is not running at **pV3** initialization, the startup and enrollment fails. This routine returns an error indication. The application programmer can test for this and try to initialize at some later time. The concentrator simplifies the startup situation. It does not require PVM functioning at initialization. Under these circumstances, the concentrator waits for PVM to start then begins checking for the server's existence. The MPI clients never return with the PVM error condition.

The design allows for multiple concentrators with a possible mixture of normal (PVM) clients. Great flexibility is obtained for large and complex solver systems, see Fig 2.

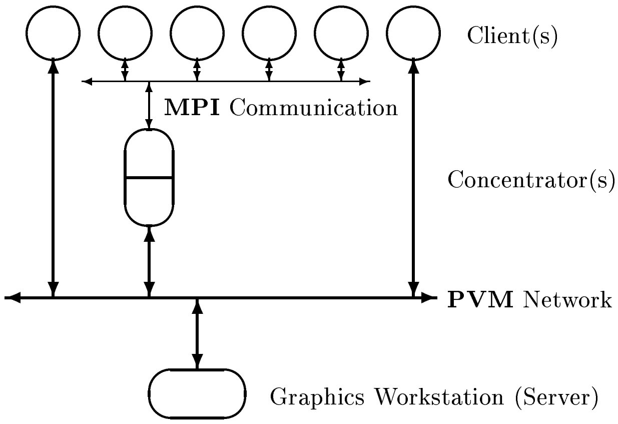

Figure 2: An Example of a More Complex **pV3** Network Setup

At the server side the following changes were made to support concentrators and further improve scaleability:

- Accept concatenated data from concentrators

- Send out collected "extract" requests
  This almost halves the number of packets, giving reduced accumulated latencies and therefore better performance even in pure PVM environments.

2.2 Problem – Simulation Production Runs

There is a problem using **pV3** in production environments or for batch execution. The difficulty is that the user may not be around to start the interactive **pV3** server and view the results. Also, even if someone is looking at the data, when the image on the screen is updated, the old information is not saved for later retrieval.

In addition, it has been found that performing certain visualization tools, like streaklines (unsteady particle paths), in a post-processing manner can lead to inaccurate results [8]. The problem with saving away the entire volume of data for post-processing is that the time-step selected for the visualization is based on the available disk space and not the physical time scales of the integration. The **pV3** system does not exhibit this problem because co-processing is fully supported.

To resolve these issues another new module has been added to the **pV3** suite, the 'batch' server. In this case, the client side remains totally unchanged. The coupled CFD solver need not know if the results are currently being viewed or are to be viewed at some later time.

When a batch job starts, the 'batch' **pV3** server is also started. Data is read on where and what tools and probes are to be active and their locations. The results (the tool "extracts") are collected and written to disk for later playback. This is different from the normal post-processing in that the entire volume of data is not written to disk every iteration. The purpose of the batch server is to write the visualization data to disk. Therefore some knowledge of the underlying hardware must be used to determine where to execute this module. The machine running the batch server should be the computer that contains the output file-system. Running the **pV3** batch server so that the output is to a distributed file-system will further burden the network and possibly slow down the solution process.

The end result is something that is not interactive in the placement of tools, but can be thought of as analogous to a wind-tunnel experiment. Some knowledge of the flow must be used in the placement of probes to extract data of interest. If important information is missed (or only found after viewing these results) the 'tunnel' will have to re-run adding (or changing the location) of the probes. However from discussions with designers, it is noted that when production runs are made, 90% of the post-processing is repetitive. For each case, the same information is examined at the same locations.

The 'post-processing' **pV3** viewer module is highly interactive in dealing with time. This is due to the fact that the amount of data has been reduced by orders-of-magnitude. The viewer's performance is further enhanced by overlapping the disk IO with display rendering. This is accomplished via multi-threading. The viewer is the only self-contained module in the suite. It does not require PVM or any other subsystem. Therefore the simulation can be reviewed away from the hardware used in data generation. All that is required is the disk file, the post-processing viewer and the proper graphics workstation.

This mode of operation has these additional advantages:

- The Data is Collected
  The process of patching together field data from a simulation that has gone through a domain decomposition can be time consuming. This is not required for this type of visualization. The data is already collected and deposited in a single file.
- Visualization is Done at the End of the Simulation
  No scripts are needed. All the visualization data has been collected. All that is required, at this point, is to view and attempt to understand the results from the simulation.
- Need for Video Movies is Reduced
  The movie is already made, but using this paradigm the viewpoint and plotting attributes for the collected "extracts" can be interactively changed while playing back the results.

The batch server and post-processing viewer can be thought of as the interactive **pV3** server split into two portions. These parts communicate through the Generic Engineering Data Structure (GEDS) data file. This file format is documented at the byte level and described in the **pV3** distribution.

2.3 Alternative Visualization Server GUIs – pV3-Gold

The default **pV3** server GUI is divided into six different windows. The functions invoked by mouse buttons, keyboard or dialbox input are dependent upon the position of the cursor. Thus, different functions are available to the user depending on which window is active, i.e. contains the cursor. The six different windows are:

- Text Window
  This window is used to input commands by the user and to output information from **pV3**.
- 3D Window
  The 3D window displays data in three-dimensional space.
- 2D Window
  This window is used to display data on a mapped domain or dynamic surface.
- 1D Window
  The 1D window is used to display one dimensional data which is generated by various functions in the 2D window or from mapping streamlines.

- Key Window
  This window displays the color scheme used in the 3D and 2D windows.
- Dialbox Window
  The dialbox window serves several purposes. The first purpose is to display the functions associated with each of the dials of the dialbox. The second role of this window is to display/modify the state of the surface database. The database lists all of the domain and static surfaces and other extracts such as streamlines.

The ability to modify **pV3**'s user interface was not available to the software engineer until recently. Since **pV3** is a low level X-lib application, control is driven by X-events. In general, all X-events are collected from the X-event queue and internal state information is adjusted appropriately. When all events are exhausted, any changes to state that require getting data from the application cause the appropriate routines to be called. After this information collection phase is done, the windows that require updating are redrawn in the data rendering phase. Then the event queue is checked for more user interaction, and this continues for the life of the session.

One of **pV3**'s goals of allowing alternate control is to maintain the current action as the default. This allows the programmer to either modify the effects of an event or completely take over the event. First, an X-event is pulled off the X-event queue. Next a programmer-supplied routine (a call-back) is invoked with the event data. This routine has the option of passing the event on, performing some action based on the event, changing the event or having the event ignored. The process is then repeated.

There is also a set of **pV3** server/viewer callable routines that allow the internal state to be changed directly.

An example of the use of this new functionality can be found in **pV3-Gold**, an alternative **pV3** visualization server being developed under the NASA Affordable High Performance Computing Project [9]. This interface is a derivative of an alternative GUI developed for **Visual3** [10].The user interface of the **pV3-Gold** visualization server is an event-driven interface. An event is defined as an occurrence caused by user input through either mouse, keyboard or input devices attached to serial ports. An event is detected by the user interface, which then updates information in the data structures and calls the appropriate procedure in **pV3**. The selected visualization procedure will then read the necessary information from the data structures and construct the visual objects that are displayed in the 1D, 2D or 3D **pV3** graphics windows.

The Motif GUI used by **pV3-Gold** is made up of windows consisting of a collection of widgets. A widget is a pre-built procedure that operates independently of the calling application except through prearranged interactions. Widgets used in developing the GUI include main window, pop-up menu, pull-down menu, push button, slide bar, label, separator, dialog box, and radio button. The primary focus in the physical layout of this system's graphical interface is to make efficient use of all available screen space.

Five buttons have been integrated on the main panel to control five of the original **pV3** windows (excluding the text window). This allows the user to have the option of using either the original GUI or the new Motif GUI. It was decided that **pV3**'s 3D, 2D and 1D graphic windows would only be visible when in actual use by the system (the user can activate them if needed). The positions of these windows have been rearranged slightly from the default interface. A panel for viewing controls is placed at the lower left portion of the display. The panel is used to control the view in either the 3D or 2D windows as well as to control the positioning of the cutting plane. All other control panels appear on the right side of the display.

The same **pV3-Gold** GUI can also be applied to the 'post-processing' **pV3** viewer module for consistency.

3 Multi-Disciplinary Visualization

When using computational analysis in design problems for complex systems, simulations using multi-physics are invoked. The classic example of this multi-disciplinary analysis in aeronautics is fluids coupled with structures. The 3D space is discretized into a mesh for the structural analysis and possibly many grids for the fluid volume. These regions only co-exist at the surface boundary. This is the area where the coupling takes place.

There has been little work done on the visualization of results from these types of simulations. A notable exception is **MDV** [11], a testbed built on the visualization system **SuperGlue** [12]. This was an ambitious project coupling object-oriented data handling with artificial intelligence for visualization extraction. Unfortunately, the project never got out of the conceptual phase.

Therefore, when a multi-physics problem is run, each discipline is usually treated separately and visualized alone. **pV3** now supports these types of runs in a more integrated fashion.

Multi-disciplinary cases are defined at the client pool when the **pV3** server starts. A client defines its discipline by setting a character string at **pV3** client-side initialization. All clients belonging to the same discipline have the same field variables (i.e. scalar, vector, and etc.) and must define the same programmer-defined cuts.

The design of the server (and post-processing viewer) for multi-disciplinary cases is that there be a current discipline and that while the discipline is active, all user interaction effects that discipline. This can be thought of as multiple visualization sessions (one for each discipline) with one set of output windows. This means that each discipline has its current field variables, with associated color maps and visualization state. Because data displayed in the 2D and 1D windows reflect some further examination of the 3D data, these windows only show the filtering from the current discipline (though each discipline's state can indicate output to these windows). Switching between disciplines will update the data drawn to reflect the state of the current discipline.

The following information is 'global' and therefore not part of any discipline's state:

- Display State
  There is one set of windows and they are used regardless of the discipline. This is also true for the lighting used in the 3D Window.

- View State
  The viewing transformation matrix is global and not part of current discipline. This maintains a constant view as the disciplines are changed.

- Planar Cuts
  The dynamic planar cut is the only tool that allows cross-discipline examination. The 2D window will display all the disciplines cut by the plane using their current scalar colored with their own map. If any (non-current) discipline has a *mapped* domain surface, iso-surface or a programmed cut on when the planar cut is activated these tools are turned off.

  A line probe will only display the scalar along the line for the current discipline.

- Time Color Map
  There is only one color map used for displaying time (on particles).

4 Future Applications of Visualization in a Parallel Environment

If the trend in computational performance continues, within 10 years our computers will be 1000 times faster than today's models. At the same time, network bandwidth is increasing from the current Ethernet speeds of 10 megabits per second to higher speeds of at least 100 megabits per second due to advanced technologies such as ATM or high-speed Ethernet. Speeding up the network communication will be driven in a large part due to the increasing integration of the World Wide Web, telecommunication, and multi-media into each industry's intranet. However, the engineering community will benefit from these improvements since it will be possible to simulate more complex and realistic engineering models. Potential applications where visualization in a parallel environment can have a significant impact are described below.

4.1 Multi-disciplinary Simulations

The visualization of multi-disciplinary cases is a new field. **pV3**'s current abilities are limited to displaying the disciplines' results simultaneously in the same window. A discipline can be turned on or off so that volumes that may obscure the target information can be removed. As stated above, the shared surfaces are usually where the physical coupling takes place. If we are to use the visualization to attempt to understand the results, it is important to see the impact of this coupling. This may require *hooks* (a window) into the coupling procedure(s). Being able to display this *2D* information and data derived from the procedure is important. Much work is required to attempt to standardize the coupling so that a set of *hooks* for the coupling (and therefore the visualization) is possible.

A positive side effect of being able to dissect and view the multi-disciplinary coupling is the ability to debug the procedure. Anomalies can be quickly spotted when graphically displayed. Interpolation schemes can be checked on the target surface discretization.

4.2 System Design

NASA has focused on using the High Performance Computing Initiative to conduct research in the simulation of entire systems of an aircraft or gas turbine engine. Visualization will play a significant role in these systems simulations. Besides viewing specific information of system components, it can also be applied for examining the system in a global sense. Interactive visualization techniques will be used to analyze the system where visual representations of the system will be used to navigate between system components. As the capability of system simulation becomes available the visualization metaphor will change. Areas where new visualization techniques will be developed for system simulation include:

- Team Collaboration – Cooperative Visualization
  There will be a move from developing visualization for a single user to a collaborative team of designers who will analyze the results from remote locations. The cooperative visualization will be coupled with a multi-media video conference application facilitating the collaboration. The visualization system will allow view locking and have soft-cursors so that features in the simulation can be located and pointed out to the team.

- System Steering
  This will allow the designers to vary and assess the parameters affecting the simulation system performance in a near real-time environment.

- Zooming
  Zooming is a function that allows simulations of differing fidelities or dimensions to couple (and therefore communicate). This is a requirement for multi-disciplinary runs where the physics of each discipline may not match. The visualization software needs to be able to deal with zooming in a similar manner that the components use in the system simulation.

4.3 Automatic Feature Extraction

Automated feature detection and identification procedures need to be developed for the examination of 3D transient simulations. Today, visualization procedures are used interactively to identify physical phenomena contained within the data. The knowledge required to identify the various phenomena (physical and numerical) contained within various types of fluid dynamic flows should be built into a knowledge base that could be used by an expert system to automatically locate, detect and identify phenomena in dynamic fluid data. The benefits of developing these automated feature identification systems are 1) automation procedures reduces time of engineer required to analyze data, 2) experience and knowledge of experts is made available to less experienced users, and 3) experience is accumulated.

In the past, feature extraction and identification were interesting concepts, but not required to understand the underlying physics of a steady flow field. This is because the results of the more traditional tools like iso-surfaces, cuts and streamlines were more interactive and easily abstracted so they could be represented to the investigator. These tools worked and properly conveyed the collected information at the expense of much user interaction. For unsteady flow-fields, the investigator does not have the luxury of spending much time scanning only one "snap-shot" in the simulation. Automated assistance is required in pointing out areas of potential interest in the flow. This must not require a heavy compute burden (the visualization should not significantly slow down the solution procedure). And methods must be developed to abstract the feature and display it in a manner that makes sense physically. Some of the important features for fluid flow are:

- Shocks
  The display of shocks is simple; a shock is a surface in 3-space. As the solution progresses the investigator can view the changing shape of the shock surfaces.

- Vortex Cores
  Finding these features is important for flow regimes that are vortex dominated (most of these are unsteady) such as flow over delta wings and flow through turbine cascades. Tracking the core can give insight into controlling unsteady lift and fluctuating loadings due to core/surface interactions.

  Some success has been made recently [13]. This particular algorithm has been designed so that no serial operations are required, it is parallel, deterministic (with no 'knobs'), and the output is minimal. The method operates on a cell at a time in the domain and disjoint lines are created where the core of swirling flow is found. Only these line segments need to be sent over the network to be displayed, reducing the entire vector field to a tiny amount of data.

- Regions of Recirculation
  Recirculation is a difficult feature to locate, but a simple one to visualize. A surface exists that separates the flow (in steady-state) so that no streamlines seeded from one side of this surface penetrate the other side. Some work has been done in locating this feature by computing the stream function. Also it is possible to use vector field topology to find the extent of this region and then draw a series of streamlines connecting the critical points. These lines can be tessellated to create this surface.

  These methods do not work for transient problems. Like a series of instantaneous streamlines can be misleading in unsteady flow regimes, using techniques based on streamlines will not represent the regions of *older* fluid. The concept that appears most promising is Residence Time. This is the Eulerian view of unsteady particle tracing (a Lagrangian operation). A simple P.D.E. can be solved on the same mesh along with the flow solver. (NOTE: This is only possible when performing co-processing do to the same time limit constraints.) An iso-surface can be generated through the result so that regions of *old* fluid can be separated from newer fluid elements.

A suite of feature extraction algorithms need to be identified and constructed for disciplines other than fluids. Only with a complete arsenal of these tools can complex transient physical systems be analyzed and understood.

4.4 Navigational Maps

There are a number of applications that do not require interactive processing of the results of a simulation when there is the ability to extract the physical features off-line. The feature techniques listed above can be used to abstract and make cartoons or *navigational maps* of the data in the simulation. This is analogous to weather maps, where the data is simplified and drawn in a way that conveys fronts, storms and circulation patterns. *Navigational maps* can be used in the following scenarios:

- Parametric Study of Steady-State Runs
  The result of each run can be processed through the feature extractor and a thumb-nail image made. A list of features can be cataloged along with this image. The investigator can then compare the input conditions to the resultant features. Transitions, when the solution maps from one set of features to another, can be noted.

- Data-base Archiving and Retrieval
  The input data as well as the catalogue of the resultant features can be placed in a relational data-base. If desired, the entire solution and/or the graphical representations (images) can also be included. At this point the data-base can be used to answer questions like; With these input conditions, what output would I expect? Or even the other way around – Under what

set of input conditions would I expect to see these features?

With a large enough suite of cases it may be possible to construct simpler, surrogate, models of the simulation.

- MPEG digital movies from Transient Applications
  The total size of the information required to communicate the results from an unsteady simulation can be reduced even further from the GEDS file to an MPEG movie (MPEG is a digitial movie format currently used on the World Wide Web). This can either be in an iconic (cartoon) form or a movie created from the **pV3** post-processing viewer.

4.5 The World Wide Web

The technologies developed that allow **pV3** to efficiently function can be applied to the Web. **pV3** is network based and works because the amount of data passed over the "wire" is minimized to only what is needed to satisfy the request. It has been shown that replacing the underlying network interface is possible (i.e. MPI instead of PVM). It is not difficult to imagine replacing **pV3**'s network interface with one supported by the World Wide Web. Then by supplying a viewer (or modifying an existing server) to accept the data, one could do scientific visualization supported by **pV3**'s client-side solver coupling and the Web.

5 Conclusions

An interactive visualization system **pV3** (parallel Visual3) is being developed for the investigation of advanced computational methodologies employing visualization and parallel processing for the extraction of information contained in large-scale transient engineering simulations. Visual techniques for extracting information from the data in terms of cutting planes, iso-surfaces, particle tracing and vector fields are included in this system. This paper discussed the current design of the **pV3** system. In addition potential future applications for visualization in a parallel processing environment were described.

Acknowledgments

This work was partially sponsored by NAS at NASA Ames Research Center with Michael Gerald-Yamasaki as the Technical Monitor. The Affordable High Performance Computing Project out of NASA Lewis Research Center (Theresa Babrauckas, Technical Monitor) also contributed to this work. Additional support came from the IBM SUR Project and the IBM UUP Project.

References

[1] M. Burnett, R. Hossli, T. Pulliam, B. VanVoorst, and X. Yang. Toward Visual Programming Languages for Steering Scientific Computations. IEEE Computational Science and Engineering, 1994.

[2] A. Vaziri and M. Kremenetsky. Visualization and Tracking of Parallel CFD Simulations. Proceedings of HPC '95, Society of Computer Simulation, 1995.

[3] C. Upson, T. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz and A. van Dam. The Application Visualization System: A Computer Environment for Scientific Visualization. IEEE Computer Graphics and Applications, July, 1989.

[4] J. Kohl and P. Papadopoulos. The Design of **CUMULVS**: Philosophy and Implementation. PVM User's Group Meeting, 1996.

[5] R. Haimes. **pV3**: A Distributed System for Large-Scale Unsteady Visualization. AIAA Paper 94-0321, 1994.

[6] R. Haimes. Unsteady Visualization of Grand Challenge Size CFD Problems: Traditional Post-Processing vs. Co-Processing. Proceedings of ICASE/LeRC Symposium on Visualizing Time-Varying Data, 1995.

[7] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. May 5, 1994.

[8] D. Darmofal and R. Haimes. An Analysis of 3-D Particle Path Integration Algorithms. Journal of Computational Physics, Vol. 123, pp. 182-195, Jan. 1996.

[9] D. Wright and T. Babrauckas. Affordable High Performance Computing (AHPC) Project. Presented at the CAS '96 Workshop, Aug. 1996.

[10] D. Edwards and R. Haimes. **RAVEN**: A Rapid Visualization System for Interpretation of Engineering Data. AIAA Paper 94-0204, 1994.

[11] K. Miceli. **MDV**: An Intelligent System for Multidisciplinary Visualization. Presented at the AIAA Aerospace Sciences Meeting, 1994.

[12] J. Hultquist and E. Raible. **SuperGlue**: A Programming Environment for Scientific Visualization. Proceedings of IEEE Visualization '92, pp. 243-251, 1992.

[13] D. Sujudi and R. Haimes. Identification of Swirling Flow in 3-D Vector Fields. AIAA Paper 95-1715, 1995.