Large Scale Scientific Visualization on Cray MPP Architectures

Andrew Johnson and Cory Quammen
Army HPC Research Center / Network Computing Services, Inc.
Minneapolis, Minnesota
ajohn@ahpcrc.org

April 24, 2003

ABSTRACT

We present various methods and computational techniques for visualizing large scientific data sets from remote locations over the Internet. These techniques are based on a client-server concept where the server component is fully implemented in parallel based on the distributed memory programming model using MPI. The client component runs on the user's desktop system, and the two components communicate using standard Internet protocols. These methods and techniques have been incorporated into the "Presto" data visualization software and are used to visualize unstructured mesh data sets such as those found in computational fluid dynamics or computational structural mechanics simulations. Advanced visualization methods such as parallel volume rendering have also been incorporated into the Presto visualizer.

INTRODUCTION

Visualizing and interpreting results of large-scale numerical simulations has become increasingly difficult as the scale of the calculations being performed on today's high performance computing (HPC) architectures increases dramatically. In the recent past, numerical simulations with one or two million mesh elements were considered large. Today, many researchers are routinely performing simulations on parallel HPC architectures with meshes containing anywhere between 10 and 100 million elements or larger. These simulations produce data files on the order of tens to hundreds of Giga-bytes in size, and these data files must be interpreted through scientific visualization. Also, most researchers use HPC architectures which are not local to their university or research lab. Most HPC architectures are found at central computing facilities that maintain and administer a large system which is used by many researchers from many locations.

In a "traditional" method to visualize the results of a numerical simulation, the researcher would download the computed results over the Internet on to their local workstation from the remote HPC systems that generated the data. Once the simulation files are residing on a local disk, a workstation-based visualization program is run which loads the data set into system memory and processes it for visualization. The user can then show the boundaries of the problem, possibly shaded with some simulation variables, or the user could create cross-sections, iso-surfaces, streamlines, etc. to help visualize and interpret the results as they interactively explore the data set. All of these calculations to create/extract these visualization constructs are performed locally on the user's workstation as well as displaying the geometry using OpenGL or some other 3D API. This mode of visualization fails for large numerical simulations due to several factors:

1) Transferring large data files over the Internet takes a significant amount of time.
2) Storing the large data files on a local disk can take a significant amount of space, especially when several separate simulations are being stored.
3) The user's workstation usually does not have enough memory to load the entire data set.
4) The user's local workstation may not have enough computing power to process the entire data set effectively.

To overcome these visualization bottlenecks and to develop a toolkit that fits within a remote HPC model, we have developed the "Presto" data

visualizer [1,2]. This program is able to visualize large unstructured mesh data sets residing on remote HPC architectures using parallel computing methods. Fully unstructured meshes can be processed, and the meshes can contain tetrahedral (4-nodded), hexahedral (8-nodded), pyramid (5-nodded), wedge (6-nodded), and also mixed-element type meshes. Both scalor and vector data can be visualized, as well as time-dependent and moving mesh data sets in order to create animations. Typically, computational fluid dynamics (CFD) data sets are being visualized by Presto, but some computational solid mechanics (CSM) simulations have also been visualized.

Presto is built within a client-server framework which facilitates remote visualization and data processing. The Presto Server component is fully written in parallel based on MPI and runs on the remote HPC architecture where the simulation data was computed. The Presto Client component handles all user interaction and displays all 3D geometry sent to it by the server. The Client is designed to be a very "light" application with very minimal memory requirements so it can be run on most desktop systems. Both components are built to be very portable and run on most if not all HPC architectures and desktop workstations.

A client-server based scientific visualizer that runs on large HPC systems is essential to our AHPCRC and Army Research Laboratory (ARL) partner organizations due to the size and scale of calculations being performed at those respective sites. Our research partners at AHPCRC-Clark Atlanta University have been major users of Presto and are visualizing data located on the AHPCRC's T3E in Minneapolis, MN from their university site in Atlatna, GA, on a daily basis. Our partners at ARL have been using Presto's volume rendering capabilities to create high-quality animations for display in immersive environments. Both of these tasks, unique to Presto, could not be performed with any other application. Presto is currently available to all researchers linked to the AHPCRC data center, and also installed on several computing systems at ARL. Presto is also used by researchers at the US Army Military Academy at West Point.

More details about Presto's client-server framework, the parallel implementation, as well as information about Presto's parallel rendering capabilities are discussed in the next sections.

CLIENT-SERVER FRAMEWORK

Large HPC architectures usually have very large and fast work disks that are used to store simulation results. The total storage capacity of these work disks far exceeds those found on even the largest desktop systems. Once a numerical simulation is completed and the results are stored on the work disks of an HPC architecture, it is very difficult and time consuming to transfer and store the results to a user's local system. It is best to leave the simulation results on the HPC system that computed them, and then to use the same HPC system that computed the data to also process the data for visualization.

In most cases, the large HPC system that a user performs a numerical simulation on is in another building, or in many cases, another part of the country. Most HPC systems are located at central sites where many researchers from many locations have access to the computer remotely. The only link between a user's desktop system that resides in their office and the remote HPC system is either the commercial Internet or some other dedicated network connection. This tends to be the working model for most computational researchers.

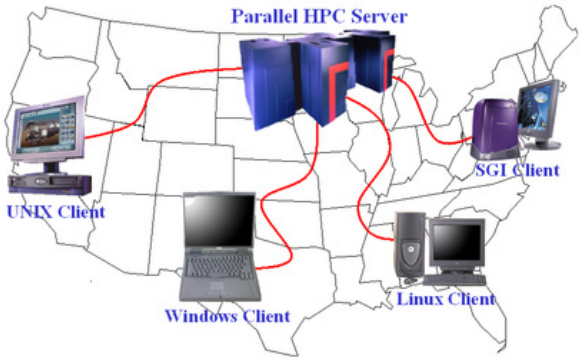

Figure 1: Client-Server implementation framework of the Presto visualizer.

To fit within this framework where visualization of data sets residing on remote architectures is desired, we have built Presto using a client-server

model as shown in Figure 1. We break the application up into two parts called the Presto Server and the Presto Client. The Server runs in parallel on the remote HPC architecture where the simulation data is residing. The Server is responsible for loading the simulation data set into memory, processing it, and creating and/or extracting any "visualization constructs" that the user requests such as a boundary, cross-section, iso-surface, or streamline geometry. This geometry is created by the Server and then sent to the Client component for display. The Server and the Client maintain an Internet connection throughout the entire visualization session where the server responds to requests sent to it by the server. In most cases, geometry such as a cross-section or iso-surface is created by the Server and sent to the Client in a "raw" format, and no geometry compression is currently being used but may be added in the future.

The Client component handles the graphical user interface (GUI) and displays all 3D geometry created by the Server component. Once the geometry is loaded on to the client, no communication between the client and server is required for rotating, scaling or other manipulations of the geometry. The geometry is displayed on the Client using the OpenGL graphics API. 3D graphics capabilities of both UNIX and Windows-based desktop systems are very powerful today and can usually display thousands to millions of polygons very quickly.

The two components of Presto communicate using a traditional socket connection using a special, well defined protocol for sending messages, parameters, and geometry back and fourth. Since TCP/IP sockets are used, a connection can be created between any two points on a network. Fast network transfer rates are desirable, but most commercial Internet connections have proven to be sufficient. Home DSL and modem connections have been tested and have proven to be effective for smaller data sets.

This client-server framework for visualizing large remote data sets works because for visualization, surfaces are created and displayed such as boundaries, cross-sections and iso-surfaces. The amount of data needed to represent a surface using polygons is an order-of-magnitude smaller than the 3D volumetric data which the numerical simulation results represent. Whereas sending the original 3D volumetric data (i.e. the entire data set) across the network is very time consuming, sending surface geometry across the network has much fewer requirements.

PARALLEL IMPLEMENTATION

Most large HPC architectures are distributed-memory parallel computers. These types of supercomputers have the large memory and increased computational power needed to solve today's numerical simulations. To take advantage of this type of HPC architecture, the Presto Server component is written entirely in parallel using the MPI distributed-memory programming library. The Presto Server is fully scalable in terms of memory used as the number of processors are increased. If a user wishes to visualize a larger data set, they simply use more processors. Typically, between 4 and 32 are used for visualization, but Presto has been tested using up to 1024 processors of a Cray T3E. Since the Presto Server is written entirely in C and uses MPI for message-passing, the code is portable to almost any HPC architecture such as the Cray T3E, IBM-SP, SGI Origin servers, and PC clusters.

The parallel implementation of the Presto Server mirrors that used in our finite element parallel flow solvers which have been under development and in use since the early 1990's [1,3-5]. When a data set is loaded by the Presto Server, the unstructured mesh is read from disk and distributed on to the parallel processors in a "default" arangement. Presto then calls the ParMetis [6] parallel mesh partitioner to obtain an optimal distribution of the mesh elements. The mesh elements are then re-distributed amongst the processors in order to realize that optimal distribution. The nodal points of the mesh are then aligned with the element partitioning and distributed accordingly amongst the processors. Finally, other data structures are computed which tie the mesh partitions together and help facilitate visualization. This mesh/data distribution has been used for unstructured-mesh flow solvers for many years and has proven to be very efficient,

scales well, and also minimizes the amount of inter-processor communication required.

When a geometric object is requested by the Client component such as a boundary, cross-section, or iso-surface, that geometry is created/extracted from the 3D mesh in parallel. Each processor derives a piece of the surface if that construct happens to intersect with its mesh elements. After each processor creates/derives a piece of the surface geometry, messages and data are exchanged to tie the surface together across the processors to form a "complete" or "coupled" representation of the surface. The surfaces are always stored on the parallel processors, but are also gathered together and sent to the Client component for display. By always storing a surface geometry amongst the parallel processors, if the user requests a small change such as displaying a different variable on a surface, the geometry has already been created and stored, and only a minimal amount of information needs to be collected and sent to the Client component for a display update. Mesh partitioning of a tactical unmanned aerial vehicle (TUAV) data set is shown in Figure 2. In this figure can be seen the distribution of the boundary surface amongst 200 parallel processors. There are 43,000 triangular polygons in the geometry depicted in Figure 2, and we typically see surface geometry being created by the Presto Server anywhere between 10,000 and 500,000 thousand polygons or more.

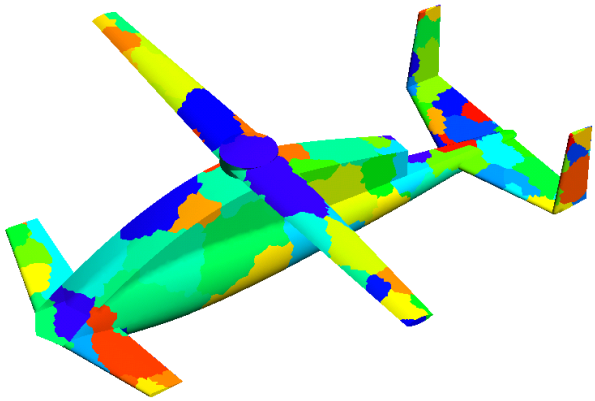

Figure 2: Parallel mesh distribution as seen on the surface of a tactical unmanned aerial vehicle.

As stated earlier, the parallel implementation of the Server component is fully scalable in terms of memory. Unstructured, tetrahedral element meshes containing between 20 and 30 million elements are routinely visualized using around 32 processors of a Cray T3E with 512 Mega-bytes of memory per processor. Using 8 processors, a user can typically visualize unstructured meshes containing up to roughly 4 or 5 million elements. On the Cray T3E, 64-bit integers are used, so on architectures using the more traditional 32-bit integers, larger data sets can be visualized on the same number of processors. In an extreme benchmark case, an unstructured mesh containing 1 billion tetrahedral elements has been visualized using 1024 processors of the Cray T3E.

PARALLEL RENDERING CAPABILITIES

For special cases and other advanced visualization capabilities, we have built two separate parallel rendering engines directly into the Presto Server component.

For extremely large data sets such as the 1 billion element mesh mentioned in the previous section, the number of polygons generated to represents a boundary, cross-section, and/or iso-surface can be too large to be sent over the Internet and processed by the user's local workstation effectively. In the 1 billion element data set, roughly 10 million triangular polygons were generated to represent a single boundary object and a single iso-surface. Creation and storage of this many polygons or more by the parallel Server component is not a difficulty if enough processors are used, but it is the transfer and rendering of these polygons on a desktop system which becomes difficult.

For these cases, we have built a "low-memory" option for Presto. Under this mode-of-operation, after a surface geometry is created/extracted by the parallel Server component, the polygons are left on the parallel processors and only a bounding box or a feature-angle description of the geometry is downloaded to the Client component for display and interactive manipulation. When the user wishes to actually "render" a detailed image of the geometry, a request is sent to the Server component which then performs polygon rendering in parallel. The rendering algorithm is very basic and is based on ray tracing. Since each processor controls only a piece of the overall

geometry, each processor renders their polygons. All of the images created by each processor are then composited together based on depth information using a single MPI command to form a final image of the geometry. This image is then sent to the Client component for display. While giving up on some user interactivity when using this mode-of-operation, a user can now visualize a data set of almost any size from a simple desktop system.

The other advanced parallel rendering capability that we have built into the Presto Server component is parallel volume rendering. Volume rendering is an advanced capability which has been typically used in the medical field to view MRI or CAT scan data [7]. In cases such as these where volume rendering has been used, the computational grid is highly structured based on 3D voxel data. Applying volume rendering to an unstructured grid such as those composed of arbitrary tetrahedral elements is computationally much more expensive to perform and has not been used much in the past.

As with the surface rendering capabilities of Presto described above, at any point during a visualization session, the user can request a volume-rendered image of the data. Each processor then independently performs volume rendering based on a ray-tracing algorithm on the piece of the mesh assigned to it. This volume data (color and depth information) is then assembled in parallel to form a single final image. This image is then downloaded to the Client component for display. Typically, a volume-rendered image of a CFD data set can be created anywhere between 10 seconds and 3 minutes depending on the size of the image. Further details about the parallel volume rendering algorithms built within the Presto Server will be presented at a future date, but an example volume-rendered image of airflow past a cargo aircraft is shown in Figure 3 where the field-variable depicted is velocity magnitude. The data set depicted in Figure 3 is based on an unstructured mesh consisting of 243 million tetrahedral elements.

EXAMPLES

Three computational fluid dynamics simulations are visualized in the examples presented here. The meshes used for all simulations were generated by our in-house automatic mesh generator [4,8] to create unstructured meshes composed of tetrahedral elements. The time-accurate numerical simulations were performed in parallel using our stabilized finite element flow solver [1,3-5] on either the Cray T3E or X1. The data sets were all visualized remotely with Presto on the Cray T3E.

Figure 3 shows a volume-rendered image of airflow past a cargo aircraft in a take-off configuration (i.e. high angle of attack, and extended flaps). The variable shown is velocity magnitude, and blue colors represent low velocity while orange/red colors represent high (close to free-stream) velocity. This data set is based on an unstructured mesh composed of 243 million tetrahedral elements, and was visualized using 350 processors of the Cray T3E. In this case, the mesh itself takes-up 10 Giga-bytes of memory, and each of the 1000 data file written to disk takes-up 1.3 Giga-bytes. The volume rendering highlights many of the airflow characteristics such as the large separation region behind the wing, as well as the complex turbulent flow field.

Figures 4 and 5 show a rendering of pressure on the surface of a TUAV design (reds indicate high pressure, blue indicate low pressure), and the velocity magnitude at a cross-section near the wings of the aircraft. The Presto GUI can also be seen in these figures. The cross section seen in Figure 5 also demonstrates Presto's ability to create actual line contours. The mesh used for this simulation of airflow past a TUAV contains roughly 2 million tetrahedral elements and was solved on the Cray T3E.

Figure 6 shows velocity vectors at a cross-section of airflow past an advanced parachute design. The velocity vectors in this figure highlight the complexity of the flow field in the wake of the parachute. The mesh for the parachute simulation contains roughly 1.7 million tetrahedral elements and was solved on the Cray T3E.

Finally, Figure 7 shows the TUAV simulation again with an iso-surface of velocity magnitude. In this figure, rather than showing a solid iso-surface, a degraded wire-frame geometry mode is used to isolate and highlight the three-dimensional features of the surface. The TUAV design, which would have been obscured with a solid iso-surface, can also be seen in this figure.

CONCLUDING REMARKS

We have presented various details about or strategy for visualizing large data sets residing on remote HPC parallel architectures. They include a client-server implementation framework, parallel computing methods, and parallel rendering capabilities. These methods have been incorporated within the Presto data visualizer which has been in use by various researchers since 2000. This visualization tool has proven to be very effective at allowing researchers at the AHPCRC and ARL to visualize their large unstructured mesh data sets from their local desktop systems (UNIX or Windows –based) connected to various remote HPC systems such as the Cray T3E, IBM-SP, SGI systems, and PC-based clusters.

Future work on these methods and the Presto visualization toolkit will include expanding its capabilities, adding more features, implementing native structured-mesh capabilities, improving and expanding the volume rendering capabilities, and increased automation and data encryption of the client-server data stream.

REFERENCES

1. A. Johnson, "Automatic mesh generation, geometric modeling, and visualization tool development efforts at the Army HPC Research Center / NetworkCS", *Proceedings of the DoD HPCMP Users Group Conference 2000*, Albuquerque, New Mexico (2000).
2. A. Johnson, C. Quammen, "Presto Visualizer 2.0, Parallel scientific visualization of remote data sets: User Guide", Tech Report, Army HPC Research Center / NetworkCS, Inc., 2002.
3. M. Behr, A. Johnson, J. Kennedy, S. Mittal, and T. Tezduyar, "Computation of incompressible flows with implicit finite element implementations on the Connection Machine", *Computer Methods in Applied Mechanics and Engineering*, **108** (1993) 99 – 118.
4. A. Johnson and T. Tezduyar, "Parallel computation of incompressible flows with complex geometries", *International Journal for Numerical Methods in Fluids*, **24** (1997) 1321 – 1340.
5. A. Johnson, "Computational fluid dynamics applications on the Cray X1 architecture: Experiences, algorithms, and performance analysis", *Cray User Group Conference 2003 Proceedings*, May 2003.
6. G. Karypis and V. Kumar, 'METIS: unstructured graph partitioning and sparse matrix ordering systems', *Tech. Report*, Department of Computer Science, University of Minnesota, 1995; METIS is available on the WWW.
7. B. Lichtenbelt, R. Crane, and S. Naqvi, *Introduction to Volume Rendering*, Prentice Hall, 1998.
8. A. Johnson and T. Tezduyar, "Advanced mesh generation and update methods for 3D flow simulations", *Computational Mechanics*, **23** (1999) 130 – 143.

ACKNOWLEDGEMENTS

The research reported in this document was performed in connection with contract DAAD19-03-D-0001 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or positions, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

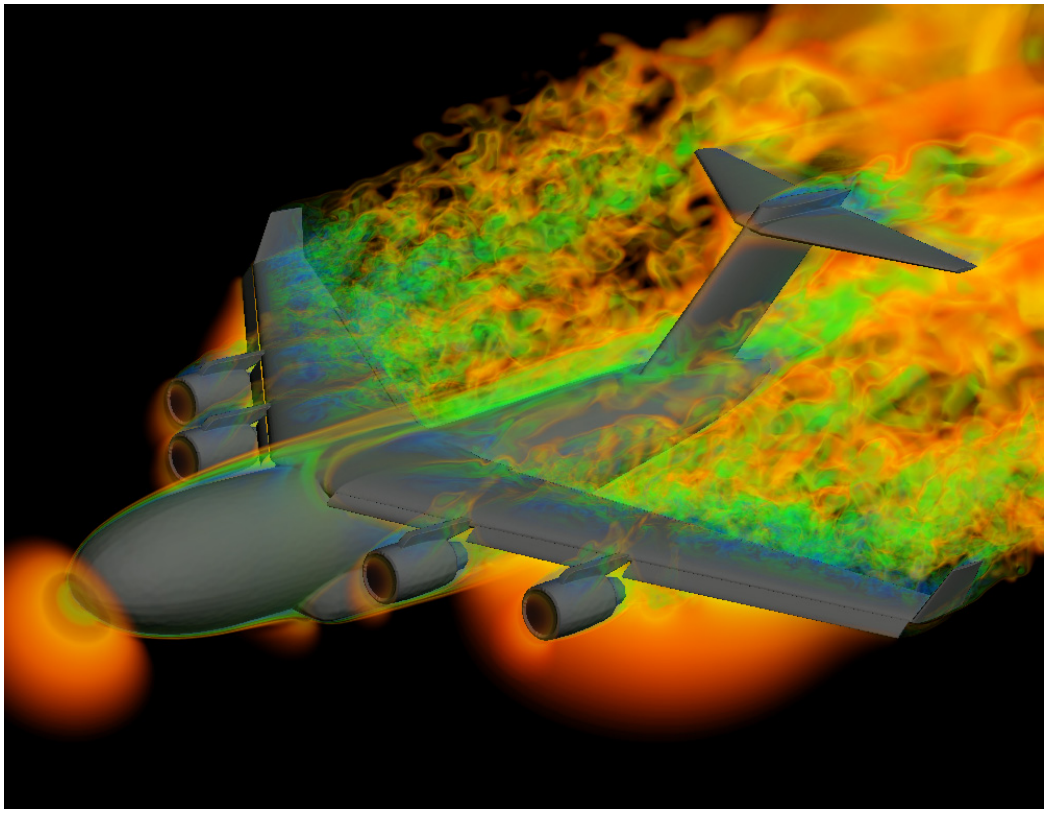

Figure 3: Volume-rendered image representing velocity magnitude of airflow past a cargo aircraft. The data set depicted here is based on an unstructured mesh containing 243 million tetrahedral elements. The solution was computed on a Cray X1 using 28 processors.

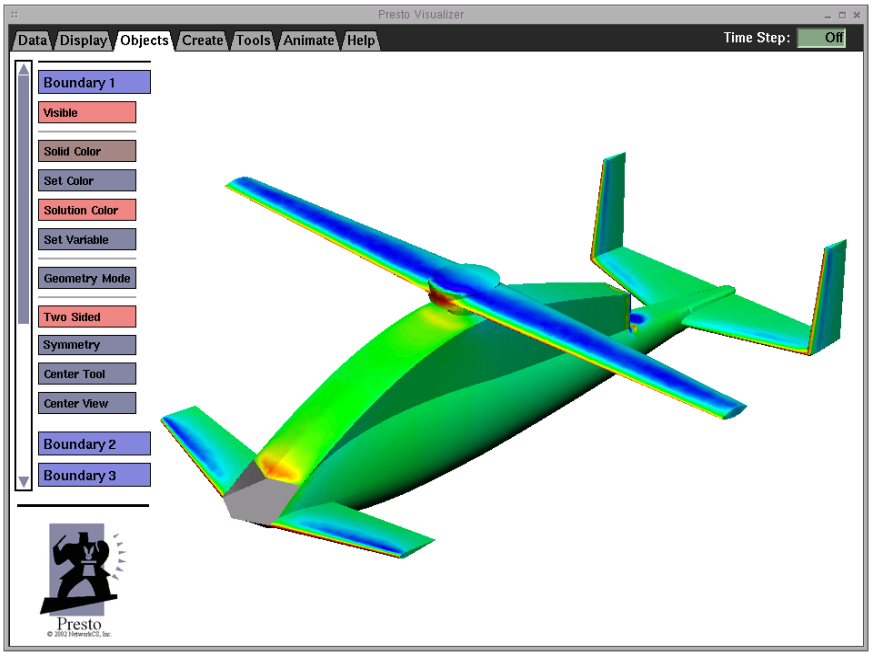

Figure 4: Pressure on the surface of a TUAV design. Red colors indicate high pressure while blue colors represent low pressure.

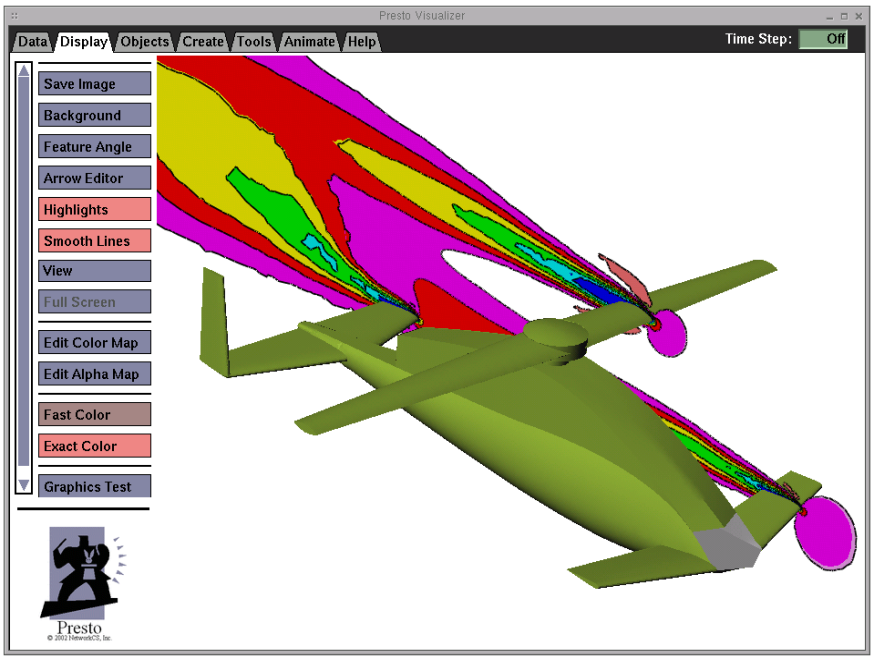

Figure 5: Cross-section depicting velocity magnitude shown around a TUAV design.

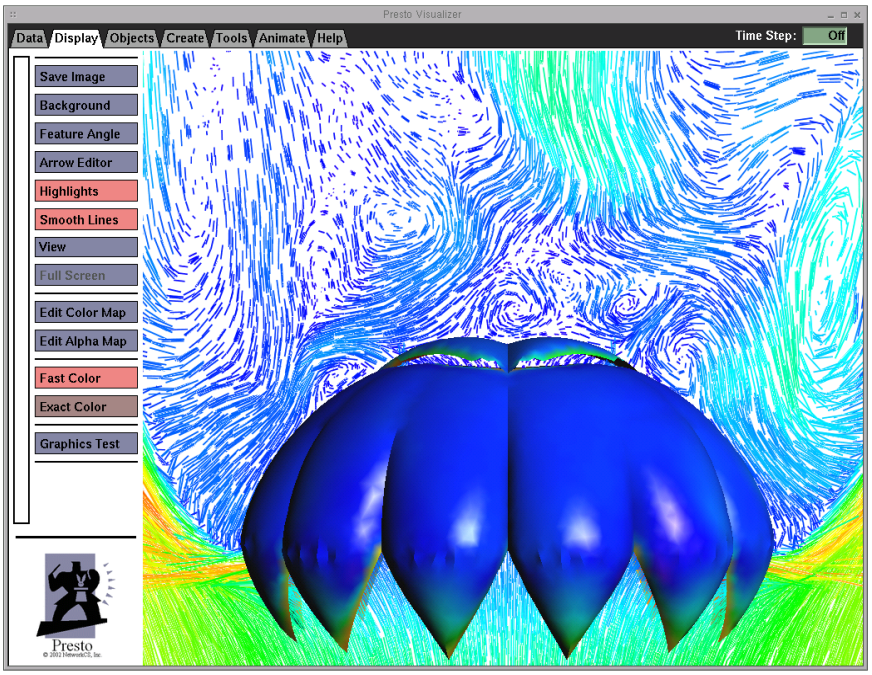

Figure 6: Cross-section depicting velocity vectors shown around an advanced parachute design.

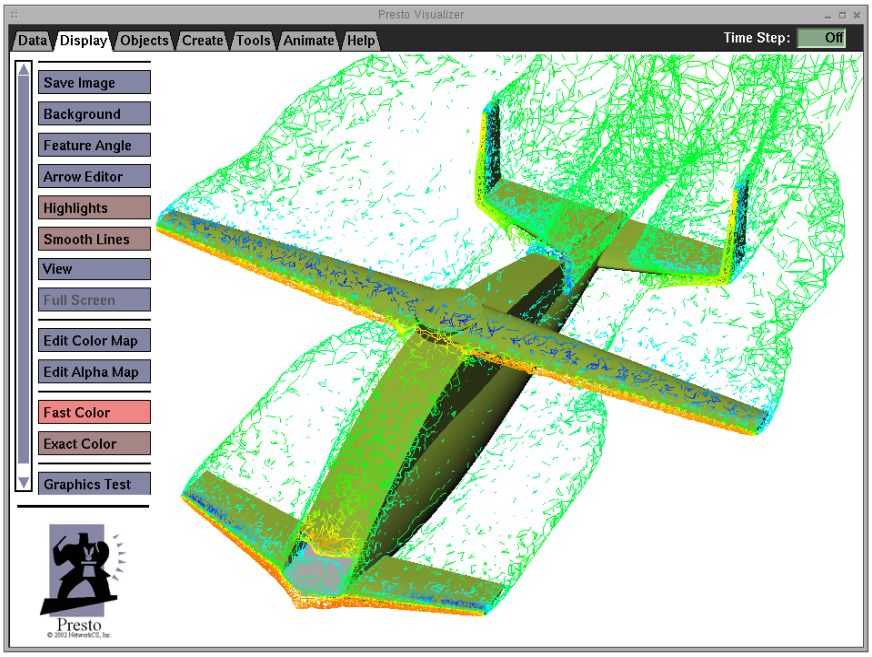

Figure 7. Iso-surface of velocity magnitude shown in a sparsely-populated wire-frame mode.