Visualizing Gyrokinetic Simulations

David Crawford Kwan-Liu Ma [*] Min-Yu Huang
University of California at Davis

Scott Klasky Stephane Ethier
Princeton Plasma Physics Laboratory

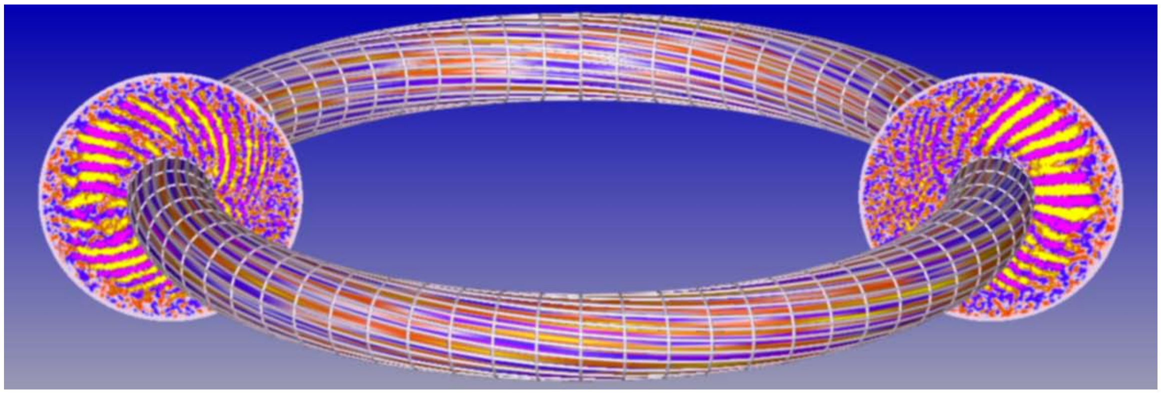
Figure 1: Interactive volume visualization for gyrokinetic simulations presents some unique challenges to existing commercial software packages due to the mesh structure, size, and complexity of the data.

Abstract

The continuing advancement of plasma science is central to realizing fusion as an inexpensive and safe energy source. Gryokinetic simulations of plasmas are fundamental to the understanding of turbulent transport in fusion plasma. This paper discusses the visualization challenges presented by gyrokinetic simulations using magnetic field line following coordinates, and presents an effective solution exploiting programmable graphics hardware to enable interactive volume visualization of 3D plasma flow on a toroidal coordinate system. The new visualization capability can help scientists better understand three-dimensional structures of the modeled phenomena. Both the limitations and future promise of the hardware-accelerated approach are also discussed.

Keywords: graphics hardware, non-rectilinear mesh, plasma physics, scientific visualization, texture methods, volume visualization

1 Introduction

Fusion energy research gives promise to environmental attractive, commercially viable, sustainable energy source for the next century. A large part of the worldwide fusion research effort goes into the development of numerical codes to simulate real experiments. By effectively utilizing the full power of modern supercomputers, fusion codes can simulate the movement of billions of particles over thousands of time steps in complex geometry. Advanced visualization is necessary for extracting the key physics from the large amounts of data generated by these simulations.

To numerically study magnetic confinement and transport physics in the fusion process, the gyrokinetic model [3, 4] has been shown to be effective in modeling turbulent transport and highly efficient on many massively parallel computing platforms. However, existing visualization systems do not provide adequate capability to support the analysis of advanced gyrokinetic simulations of plasma turbulence. These simulations produce enormous amounts of data along a mesh which follows the magnetic field lines. Scientists need to be able to freely vary the visualized parameters and interactively browse the data. Local and global details must be captured at the highest available resolution. Rendering the particle paths is needed in addition to displaying the parameter values in order to gain understanding and new scientific insights from the data.

This paper discusses the visualization challenges presented by gyrokinetic simulations and introduces a new volume rendering strategy leveraging the programmable features of PC graphics hardware to enable interactive visualization of plasma flow in toroidal systems. The interactive 3D visualization technique gives the scientists at the Princeton Plasma Physics Laboratory (PPPL) new ability to examine the structure and evolution of the volumetric, tubular features in their turbulence simulation data. The resulting visualizations also allow scientists to more effectively communicate their findings to others.

2 Gyrokinetic Simulations

Scientists at PPPL have been developing microturbulence codes in fusion to gain an improved understanding of the turbulent transport in magnetically confined toroidal plasmas. Turbulence is believed to be the mechanism primarily responsible for cross-field transport in such systems. Energy transport from the hot and dense core of the plasma to the cold walls of the device greatly exceeded the level predicted by the earlier theory of Coulomb collisions. It is now believed that plasma microturbulence driven by temperature and density gradients are responsible for these enhanced cross-field transport rates. The ability to suppress microturbulence-driven transport may well be the key to a practical magnetic confinement device. Therefore, the size and cost of a fusion reactor is determined in large part by the balance between particle and energy confinement time and fusion self-heating. Plasma turbulence is also a very com-

[*]Department of Computer Science, University of California at Davis, One Shields Avenue, Davis, CA 95616. ma@cs.ucdavis.edu

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

plex nonlinear phenomenon with associated large time and spatial scale separations, similar in many ways to general fluid turbulence.

The gyrokinetic particle-in-cell (PIC) simulation is based on the gyrophase-averaged Vlasov-Maxwell system of equations for magnetically confined plasmas [3, 4]. Since its inception, major progress has been made in carrying out full-torus (global) microturbulence simulations using PPPL's gyrokinetic Global Toroidal Code (GTC) on massively parallel computers [6]. The PIC method solves the nonlinear partial differential equations through simple linear local operations and is accordingly amenable to multidimensional domain decompositions. The computing time in a PIC code is usually linearly proportional to the number of particles residing in each processor, while the inter-processor communication time is usually less than a few percent of the total computing time. Consequently, GTC has shown excellent scaling up to 1024 processors with a parallel efficiency of up to 98% [1], and has been effectively applied to study the size scaling of reactor-size tokamak plasmas [5].

Understanding the behavior of turbulent plasma transport is not only a grand challenge problem, but it is also a key to enabling a reliable assessment of the requirements for an attractive fusion reactor in the future. Important additional physics could also be incorporated into the code, including an electron model with non-adiabatic dynamics and the vector potential parallel to the magnetic field. This would enable GTC to analyze electromagnetic shear-Alfven physics in tokamaks and stellarators. These improvements would enable exciting new capabilities for running realistic simulations on advanced parallel architectures of turbulence and neoclassical transport in reactor grade plasmas; however, the storage requirements will increase by an order of magnitude. Visualization plays an increasingly important role.

2.1 Characteristics of GTC Data

The mesh used in GTC is unstructured in the poloidal cross section, as shown in Figure 2. The mesh maintains a constant arc distance, $r\Delta\theta$, so the number of mesh points increases as we move out in the radial direction. For example, a typical mesh has 360 radial $\times$ 2560 poloidal $\times$ 64 toroidal points, a total of 59 million points. The mesh *twists* around the torus, and the *twist* is different for each radial tube as you go around the torus. A 3D view of the mesh along with two cross sections are shown in Figure 1

Several parameters, such as density and temperature, are computed at each mesh point for each time step. The largest run of the simulations with 125 million mesh points and one billion particles that has been performed thus far produced 4 TB of data. If all of the particle data were stored, the requirements would increase to over 115 TB. Consequently, intelligent analysis routines are urgently needed to store the most relevant portion of the data from regions of interest. Furthermore, data compression and advanced visualization methods are required.

2.2 Visualization Requirements

A unique diagnostic for tokamak transport would be the visualization of the actual particle trajectories in addition to the 3D rendering of density, velocity, temperature and potential fluctuations generated by the microinstabilities. The particles should be visualized in the presence of the external toroidal magnetic field as well as in the self-consistent perturbed electric and magnetic fields of a torus. When the electromagnetic dynamics are implemented in these global simulations, a formidable visualization challenge will be to track the path of the charged particles as they move through these perturbed trajectories and stochastic regions, and consequently enhance electron transport. In addition, the perturbed magnetic fields can introduce a *tearing* of the equilibrium flux surfaces in local regions.

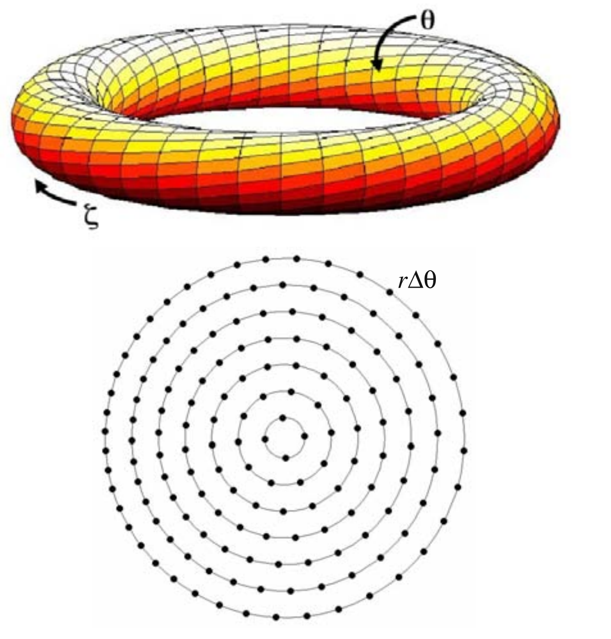

Figure 2: Top: Representation of the field-line following mesh on a flux (magnetic) surface of the system (constant radius). The twist in the filed lines depends on the magnetic equilibrium of the device under study. Bottom: mesh of a poloidal plane (perpendicular section) showing the constant density of points. This mesh rotates as one goes around the torus due to the twisting of the magnetic field lines.

Particle simulation is uniquely equipped for addressing these highly complex problems because it models the behavior of individual particles in toroidal systems. Developing advanced particle visualization software will give scientists the tools to extract key physics insights for these important investigations:

- By following the time evolution of these particles we can gain a better understanding of the physics of particle and energy transport and an answer to the outstanding question of whether the turbulent transport is convective or diffusive in nature.

- The enhanced visualization capabilities may also help unravel the mystery of the experimentally-observed inward pinch of the impurity particles coming from the plasma edge.

- New visualization tools are needed to address the outstanding question of what actually causes the anomalously-high levels of electron transport always observed in tokamak plasmas.

To address these visualization needs, the techniques that we and others have developed for visualizing 3D field lines [13, 11, 12], particle data [14, 9], and time-varying data [8, 7, 15, 10] are applicable. In this paper, we focus on the need to visualize the scalar Maxwell potential data. In the past, scientists at PPPL were mainly using AVS/Express for making cross-section and isosurface visualization of the potential data. As shown in Figure 3, one example highlights the positive and negative potential, and the other reveals the tubular structures. However, the commercial software tool AVS/Express currently lacks support for volume rendering irregular-grid data and does not take advantage of the advanced features of current graphics hardware, making the process of creating visualization rather tedious and time consuming. The rest of this

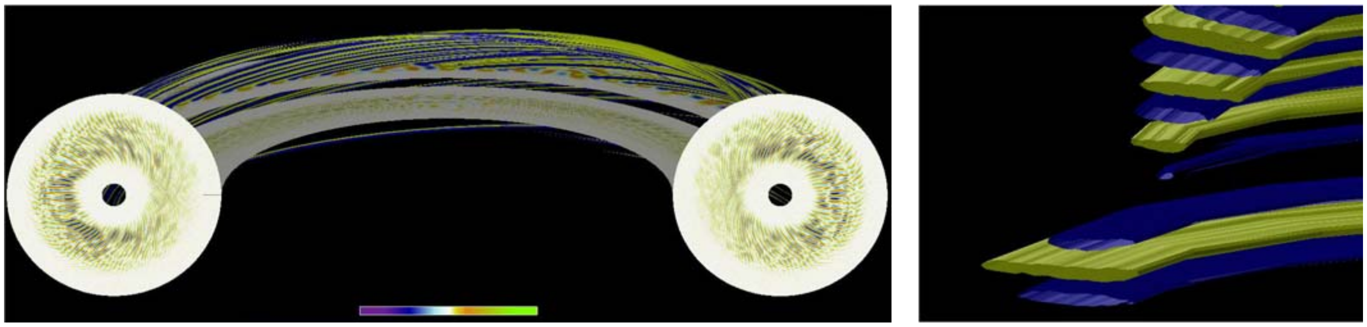

Figure 3: GTC data visualization made using AVS/Express. Left: The scalar potential. Right: Isosurface of the tubular structures.

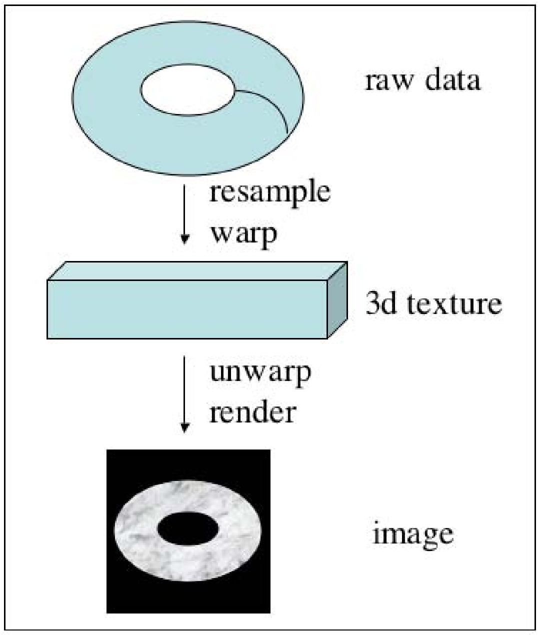

Figure 4: Steps taken to generate an image of the GTC volume data.

paper presents a new hardware-accelerated rendering technique designed specifically for interactive volume visualization of the 3D potential data on the twisted mesh.

3 A Hardware-Accelerated Volume Visualization Technique

The 3D texture hardware support of commodity graphics hardware makes possible real time volume rendering. Rendering is performed by drawing a set of view-aligned polygon slices that sample a 3D texture containing the volume data. These slices are composited using hardware alpha blending to derive the final image. However, the graphics hardware is designed around linear interpolation of planar data. The GTC volume data is not stored in such a manner so it cannot be rendered directly. To maintain interactive visualization, we have developed a rendering technique that makes use of mixed-coordinate system textures and vertex and fragment shaders. We have studied various quality and speed tradeoffs, including using a data-space versus color-space texture and mapping time-critical aspects of the fragment shader to textures. Figure 4 shows the main steps taken to generate an image of the GTC volume data.

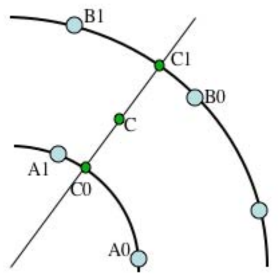

Figure 5: Cartesian point C generated by first doing arclength-wise linear interpolation between A0-A1, and B0-B1, to find the interpolated data values at C0 and C1. The ρ location of C is then used to linearly interpolate between C0-C1 to find the data value at the desired texture coordinate.

3.1 The Toroidal Data Storage

The toroidal data to be visualized is structured into poloidal planes using the minor radial (ρ) and minor angular (θ) coordinates. These planes are arrayed evenly along the major angular coordinate (ζ). The mesh structure of the disks is made to approach even spacing in the plane. This is accomplished by ordering the plane into rings of increasing number of points to maintain arclength density (see Figure 2); the starting θ of a ring and number of points per ring are also stored. The data values associated with a location are thus stored in a 2D array of dimension $\mathbf{D} \times \mathbf{p}$ where $\mathbf{D}$ is the number of disks and $\mathbf{p}$ is the number of points per disk.

3.2 Preprocessing

A primary capability of current graphics hardware for volume visualization is doing linear interpolation of textures. Thus, an optimal coordinate system would be one that closely matched the data storage and facilitated the use of the texture interpolator. This is approached here by using an unwrapped square-toroid, which is a rectangular prism, with its two equal sides corresponding to disk-aligned Cartesian planes ($\mathbf{s}$, $\mathbf{t}$) and its third coordinate ($\mathbf{u}$) mapping directly to ζ.

The object texture is generated by taking Cartesian samples of each poloidal plane created by polar interpolation of the nearest normalized data points (see Figure 5). As the interpolation is bilinear in the original coordinate system, linear error propagation is maintained. Duplicates of the first and last disks are stored as the borders of the opposite ends of the texture to allow for continuity in the interpolation.

One important consideration is that of resampling density versus memory usage. If the data is sampled sparsely, then interpolation error will increase. If the samples are too dense, then memory is wasted on samples that do not contribute to adding more information. In the case of linear error propagation, the optimum density should occur when the number density of new sample points per unit area is equal to the number density of the original data points per unit area. A regularly sampled grid containing about twice as many points results in a 3D texture that can be rendered very efficiently. In the case of data containing 91 rings per poloidal plane and 32499 points per plane, a 205×205 regularly sampled grid will result in an average area per sample point similar to that of the original data. Thus, a more memory aligned 256×256 grid can also faithfully cover 30,000-50,000 points on the original structure grid with similar sample density. A 512×512 grid can cover a case containing approximately 200,000 original points per plane.

Note that the ζ to $\mathbf{u}$ mapping works because the angular ζ distortion is linear with respect to the major radial coordinate and independent of the $\mathbf{t}$ ('height') coordinate. Therefore, angular distortion is conserved across the linear interpolation of the texture system, and finally undone when a given texture coordinate is remapped to world space. Figure 6 compares the original point data and the resampled data on a poloidal plane. The 256×256 resampled points capture the fine features in the original data.

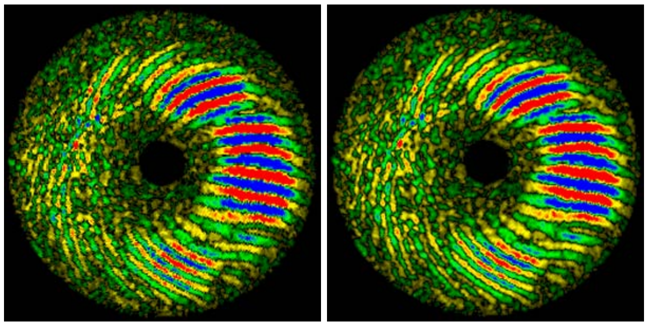

Figure 6: Left: Original point samples. Right: 256×256 resampled points. The two planes match quite closely.

3.3 Rendering

For rendering, view-aligned slices are passed to the graphics pipe, which then proceed to be processed by the vertex and fragment programs, which were were written in nVidia's Cg (C for graphics) language. The primary functionality of the rendering stage is a coordinate transformation from the rectangular object coordinate system to the mixed coordinate system of the texture data object. This can be described as $(x,y,z) \rightarrow (\sqrt{x^2+y^2}, \text{atan}(y,x), z) \equiv (r,\zeta,z)$ (see Figure 7), where (r,ζ,z) is then normalized to the final texture coordinates $\mathbf{T}_{stu}$. At various points, checks are made to verify that the sample point is within the data; otherwise the sample is discarded. This transformation is necessary because when samples are taken along view-aligned slice planes they do not vary linearly across the texture (see Figure 8). The data value is then retrieved from the 3D texture using the calculated normalized coordinates. Finally, this data value is passed as the coordinate to the colormap to give the final output color.

Rendering can be done in either data space or color space. In data space, resampling is done by interpolating the data texture, and then the color and opacity values are looked up using the resampled data value. In color space, we render classified volume directly by interpolating the RGBA texture. Data space rendering usually generates more accurate visualization while color space rendering is

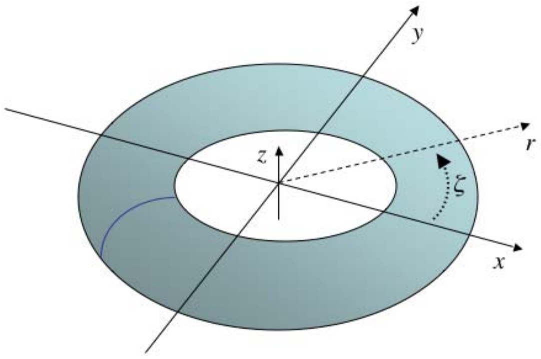

Figure 7: The mixed-coordinate system.

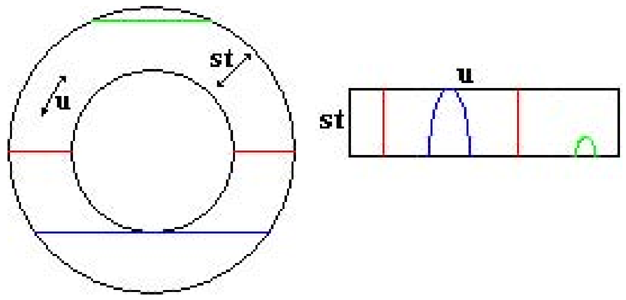

Figure 8: Left: Mapping between object slice planes. Right: Corresponding locations in the texture object.

faster. However, color space visualization often suffers from the blurriness introduced by color interpolation. Figure 9 compares data-space rendering with color-space rendering. Data space rendering displays clearer tubular structures.

4 Optimizations

The fragment shader is neither at its fastest nor at its highest quality. Various optional optimizations are possible and applied in different combinations to meet different performance requirements.

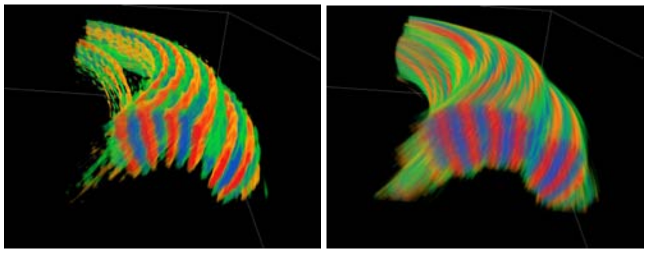

Figure 9: Left: Data-space visualization of positive (blue/green) and negative (red/yellow) potential values. Right: Color-space visualization. Color space rendering is faster but it adds some blurriness to the picture.

4.1 Quality Improvements

Aligning the texture samples to the data disks removes some artifacts. However, it requires two accesses each of the data and colormap textures: pre-alpha multiplying each temporary color by its alpha value, and linear interpolation between the colors; followed by the final alpha to obtain the output color.

Using more view-aligned slices can increase image quality but opacity must be appropriately corrected. Adding in the opacity correction formula allows for consistency across sampling densities at a slight performance loss.

In the texture coordinate function, adding a check against the minimum and maximum minor radii before normalizing $\mathbf{T}_{st}$ allows for clipping the object by varying the input parameters for those radii. This comes at the cost of an if-statement branch in the fragment program.

Lighting can help illustrate the shape of 3D structures and their spatial relationships. To including lighting, normalized gradient direction of each voxel must be either precomputed and stored or calculated on the fly, which requires additional texture lookup. Our current implementation does not include lighting.

4.2 Speed Improvements

A color-space interpretation of the data can be a speed increase of mixed blessings. On the positive side, it allows the pre-multiplication stage to be moved into the texture object creation phase and it removes the need for accessing an additional colormap. However, since it has to update the texture object whenever the RGB or alpha maps change, it adds a delay when trying to modify the transfer functions. Additionally, it can add sometimes enlightening, but counter-intuitive artifacts to the image. As an example, consider a two-color transfer function. In data-space, all output will be, for example, red or blue. In color-space however, if there are two paths of data near to each other, one red and one blue, then the region between them will be interpolated to be purple. A similar problem occurs with transparency functions that are non-monotonic.

Because color-space does not require a colortable lookup, another optimization is available in this mode. When the data values are first interpolated into the **s**-**t**-**u** coordinate system, the borders can be set to a value that is outside of the [0, 1] range required for data-space. Then when the color object is generated, the alpha value of the borders can then be forced to zero, disregarding the colormap entirely. Then the texture border clamping capabilities can be taken advantage of, allowing the check of whether or not the fragment is inside the toroid to be removed. Because the branching capability of current hardware is such that the program takes as long as if sections of code were not skipped, it can currently be faster to just process a fragment with a zero alpha than to test if the point is outside of the object.

An additional increase in framerate can come from changing the arctangent function call to a two-dimensional texture. This texture can be quickly procedurally generated at run-time for a one-time cost and then used in place of the "atan2" Cg function call. This performance increase is primarily because the arctangent function is not native to the current generation of graphics cards and instead is expanded to around an additional 30 lines of GPU assembly code. The quality trade-off comes from a blockiness that is visible at close viewpoints when the texture lookup-function is used. Tests indicate that this is only minimally dependent upon the texture size itself.

One speed improvement that can allow even higher sampling densities while maintaining a faster framerate is cropping the viewing area to a subset of the bounding box. This is done at zero cost by changing the parameters for the borders in the creation of the view-aligned slices instead of putting bounds checking in as an additional geometry step in the shaders.

5 Test Results

We have tested our hardware accelerated rendering technique on an Intel Xeon 3.06 GHz machine with 4GB of RAM and an nVidia GeForce 5900 Ultra graphics card. Recorded framerates in Table 1 were achieved using a 500×500-pixel display window. The test data set consists of 64 disks and 32,449 points per disk so a 256×256×64 regular grid is sufficient. The largest increases of framerate came from activating the texture-based arctangent. In addition, removing disk-aligned sampling can increase framerate by another 30-40%, which is not revealed in this table.

Other saving can be obtained by processing fewer fragments. The default mode (G) uses geometry clipping and branching. Additional optimization (Gd) is possible by restructuring function returns to remove an if-statement in the texture coordinate generation code. As if-statement overhead is reduced in new cards, such an optimization will become obsolete. Further saving can be achieved by avoiding branching. This is done with texture boarder extension, making the space outside the toroidal region transparent rather than performing geometry clipping. We call it alpha clipping (Ga).

Table 1: Performance: frames per second (relative speedup). Far: 500×500 window, but entire torus does not fill the screen. Close: window filled primarily by one segment of the torus. Top: a default view looking at the ring of the torus where the torus fills the window.

Gfx Options	Far	Close	Top
D G A	4.28 (1)	.45 (1)	.48 (1)
D Gd A	4.86 (1.14)	0.55 (1.19)	.56 (1.16)
D G At	9.87 (2.31)	1.22 (2.69)	1.11 (2.28)
C G A	4.42 (1.03)	0.49 (1.08)	.51 (1.06)
C G At	7.26 (1.7)	1.35 (2.98)	1.14 (2.36)
C Gd At	7.65 (1.79)	1.45 (3.19)	1.19 (2.46)
C Ga At	14.48 (3.38)	3.41 (7.54)	2.98 (6.15)

Keys for Gfx options:

C: color space	D: data space
A: assembly arctan	At: texture-based arctan
G: geometry clipping	Gd: early-discard geometry
Ga: alpha clipping	

Figure 10 shows images generated with data-space rendering The image on the left was made with direct computing of *atan* values while on the right with table lookup. Figure 11 shows images generated with color-space rendering. The image on the left was made with disk-aligned sampling while on the right with full hardware sampling. Note that using table lookup can lead to 50-75% speedup in rendering time but the image results, as shown, are comparable.

The value of volume rendering is that it can show scientists all of the tubular structure which is inside of the dataset. Figure 12 shows direct volume rendering of the tubular structure in color space as a result of interactive cutting away and enhancement by editing transfer function. This timeslice was after the zonal flow, which means that the tubular structure generated in the "middle" timeslices, as shown in Figure 3, gets torn apart into these threads. Figure 13 displays similar feature enhancement for three selected timeslices which are rendered in data space instead. When such a feature extraction and exploration can be made interactively and conveniently, the scientists' opportunity for discovery is greatly enhanced.

Finally, Figure 14 shows four selected timeslices, in which we see the onset of turbulence, where eventually the eddy structure is formed in the later time steps. These images are generated in color space. It is very difficult to look at this using isosurfaces, compared to using volume rendering, because the time required to do nested isosurfaces for large GTC datasets is prohibitive. It can take over minutes per timeslice on a comparable PC.

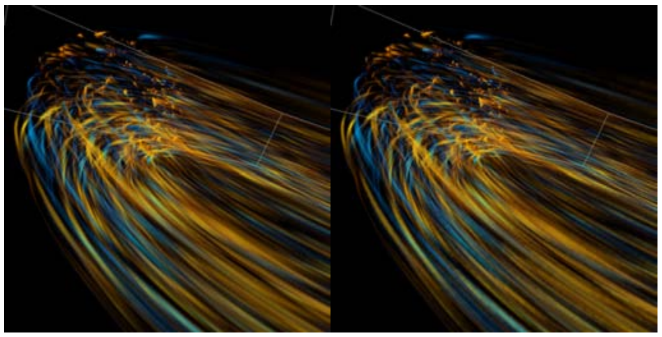

Figure 10: Data-space visualization of potential plasma flows. Left: Using the hardware-assembly expanded *atan* function. Right: Using a 2D lookup texture for *atan*.

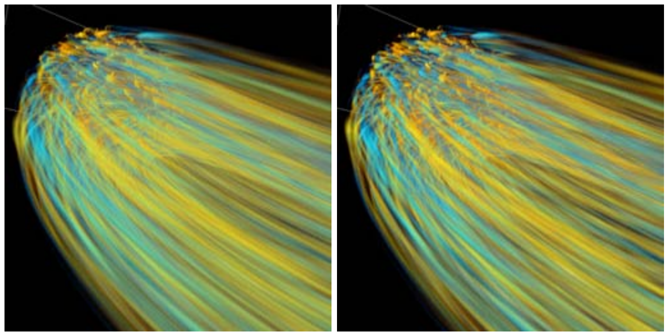

Figure 11: Color-space view using hardware *atan*. Left: Disk-aligned sampling. Right: Full hardware sampling.

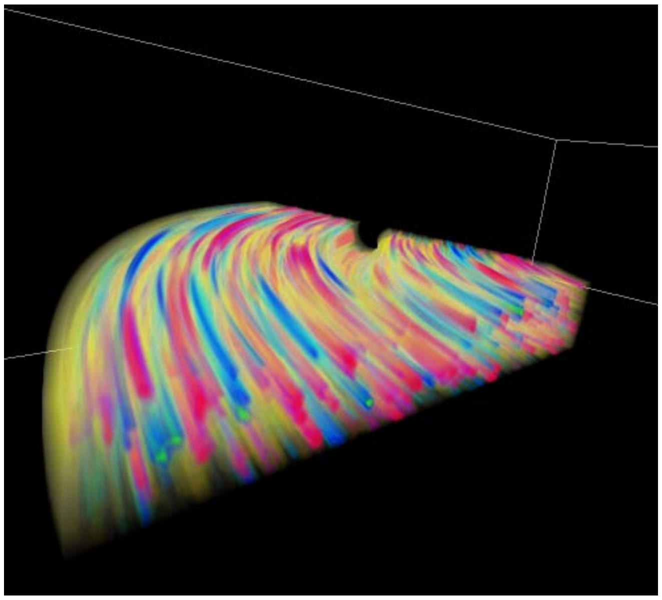

Figure 12: Volume rendering of the tubular structure with enhancement. Positive potential in blue and negative potential in pink. The volume is largely cut away to reveal a middle section.

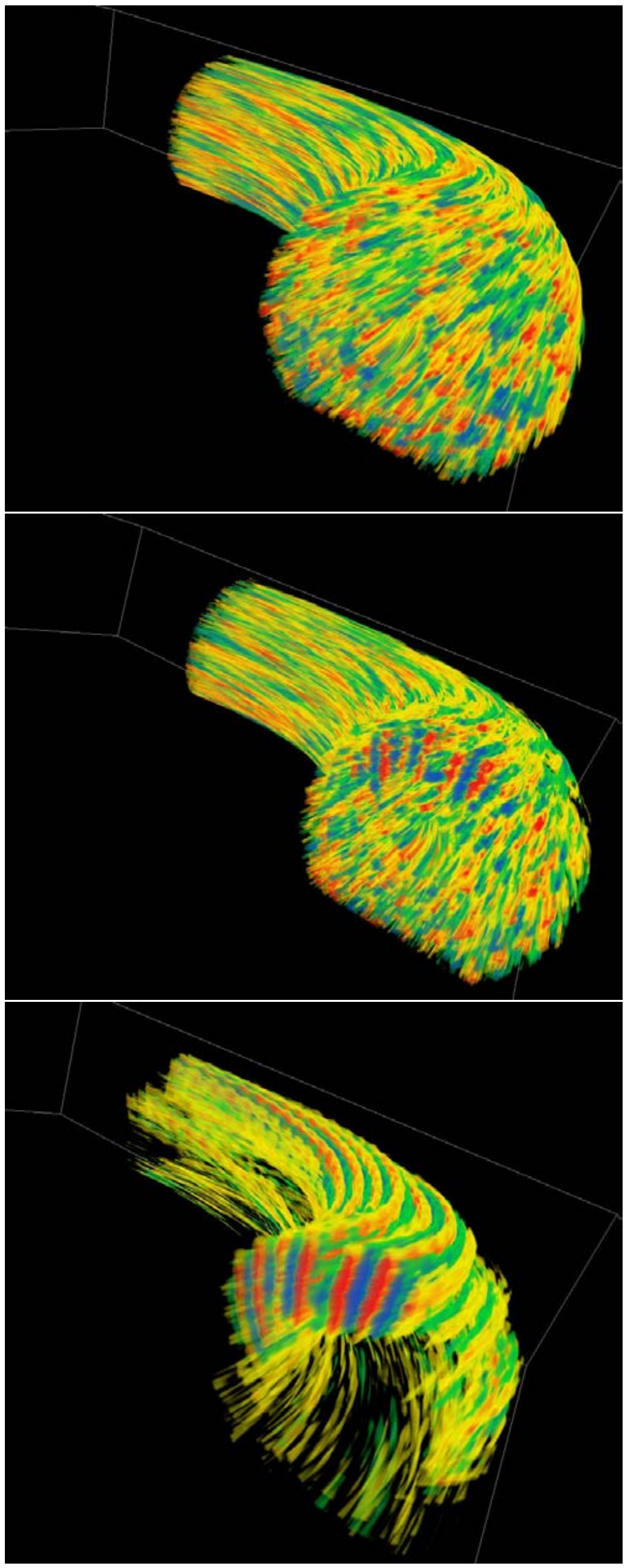

Figure 13: Visualization of the tubular structure for three selected time steps. Positive potential in blue and green and negative potential in red and yellow.

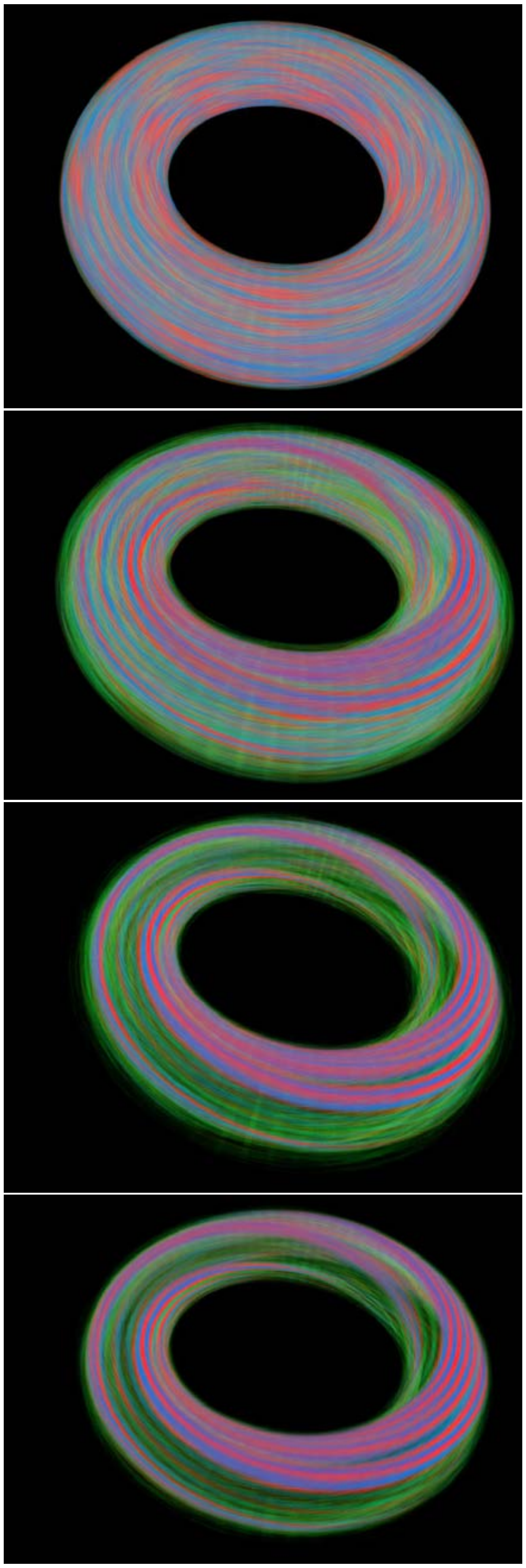

Figure 14: Four selected time steps from top to bottom. We see the onset of turbulence, where eventually the eddy structure is formed in the later time steps. Positive potential values are mapped to blue and green and negative values to red.

6 Discussion

The gyrokinetic data has been previously analyzed using isosurfaces, but not with volume visualization techniques. With isosurface visualization, only a selection of data values would be seen, possibly leaving out critical and or surprising information that could have been available with a volume visualization. This program was brought forward as an option for a different style of analysis. Both techniques have pros and cons, leading to different but complementary uses. Direct volume rendering gives holistic viewing of the data to see all of the tubular structure in ways impractical for isosurface techniques. The time required for isosurface creation can become prohibitively expensive as datasets increase in size and detail but isosurface visualization scales between display sizes with little penalty.

Volume visualization has different aspects affecting visualization time, including texture creation time and the pixel count of the viewing window. Texture creation time is linear with respect to not dataset size, but texture size, which can be adjusted as detail requires. And with current graphics hardware, high quality textures can be held on-card, at which point texture size is nearly decoupled from rendering speed. As this program is shader-limited, render time is proportional to the number of pixels and the sample plane density. Thus for a given image, the current implementation would not work well for large displays or powerwalls. However, it does have application as a possible monitoring tool for viewing datasets as time slices of the simulation are generated.

While interactive volume visualization is attractive, the current hardware accelerated approach suffers from two types of artifacts. The first type of artifacts is due to insufficient sampling rate in the the front and back of the volume; depending on how the view-aligned slices intersect with the poloidal data slices, different level of blurriness may appear in those regions. The second type is mainly transfer-function dependent, but has also to do with the crude hardware interpolation employed. The former can be slightly alleviated by employing more view-aligned slices but not be completely removed. The latter would require accurate interpolation following the tubular structure which is not currently practical to do in hardware due to the increased complexity of the shader program. Nevertheless, the interactivity possible still makes the hardware rendering approach attractive since scientists can freely vary transfer functions and view to tell what are artifacts and what are physical features.

7 Conclusions

The GTC particle-in-cell nonlinear microturbulence code has utilized the full computing power of the modern supercomputers to produce outstanding scientific results. Our ultimate goal is to address the major visualization challenges associated with the increasingly large data sets from GTC. In this paper we present our experience in making use of commodity graphics hardware to address the irregular grid problem. We show interactive volume visualization of the data on a single PC. This new capability offers scientists the power to examine the volumetric, tubular structures in the turbulence data.

For large meshes, like the largest one consisting of 59 million points, a much larger texture would be needed to retain accuracy. Our hardware accelerated technique can be implemented on a PC cluster to render texture of that size at interactive rates. We envision a multiresolution approach to the large data problem. That is, the scientist will switch between interactive browsing in the temporal, spatial and variable domains of the data and a close-up view of some region of interest by resampling on the fly the corresponding subset of the highest resolution data. Furthermore, these visualization techniques are valuable for most of the datasets produced in fusion simulations, since they work in toroidal coordinates.

Future work includes temporal-space animation and the incorporation of particle visualization along with volume visualization. A parallel data streaming approach has been developed to efficiently transfer terabytes of time-varying data generated by the Gyrokinetic code for data analysis and visualization [2]. We will set up a parallel rendering pipeline coupled with the data streaming method for simultaneous visualization of volume and particle data.

ACKNOWLEDGMENTS

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE award), ACI 0222991, and ACI 0325934 (ITR). the U.S. Department of Energy under Memorandum Agreements No. DE-FC02-01ER41202 (SciDAC program), No. B523578 (ASCI VIEWS), and the LANL/UC CARE program.

REFERENCES

[1] S. Ethier, Z. Lin, J. Lewanowski, W. Wang, W. W. Lee, T. Hahm, and W. M. Tang. Gyrokinetic toroidal code: A 3d parallel particle-in-cell code to study microturbulence in magnetized plasmas. In *Proceedings of the CCP02 meeting of American Physical Society, APS/DCOMP*, 2002.

[2] S. Klasky, S. Ethier, Z. Lin, K. Martins, D. McCune, and R. Samtaney. Grid-based parallel data streaming implemented for the gyrokinetic toridal code. In *Proceedings of Supercomputing 2003 Conference*, 2003.

[3] W. W. Lee. Gyrokinetic approach in particle simulation. *Physics of Fluids*, 26(2):556–562, February 1983.

[4] W. W. Lee. Gyrokinetic particle simulation model. *Journal of Computational Physics*, 72:243–269, 1987.

[5] Z. Lin, S. Ethier, T. Hahm, and W. M. Tang. Size scaling of turbulent transport in magnetically confined plasmas. *Phyica. Review Letters*, 88:195004–1–195004–4, May 2002.

[6] Z. Lin, T. Hahm, W. W. Lee, W. M. Tang, and R. White. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, 281:1835–1837, September 1998.

[7] E. Lum, K.-L. Ma, and J. Clyne. Texture hardware assisted rendering of time-varying volume data. In *Proceedings of Visualization 2001 Conference*, pages 263–270, October 2001.

[8] K.-L. Ma and D. Camp. High performance visualization of time-varying volume data over a wide-area network. In *Proceedings of Supercomputing 2000 Conference*, November 2000.

[9] K.-L. Ma, G. Schussman, B. Wilson, K. Ko, J. Qiang, and R. Ryne. Advanced visualization technology for terascale particle accelerator simulations. In *Proceedings of Supercomputing 2002 Conference*, November 2002.

[10] J. Schneider and R. Westermann. Compression domain volume rendering. In *Proceedings of IEEE Visualization 2003 Conference*.

[11] G. Schussman and K.-L. Ma. Visualizing tokamak magnetic field line data. In *Proceedings of IEEE Visualization 2000 Conference*, pages 501–504, October 2000.

[12] G. Schussman and K.-L. Ma. Scalable self-orienting surfaces: A compact, texture-enhanced representation for interactive visualization of 3d vector fields. In *Proceedings of the Pacific Graphics 2002 Conference*, pages 356–365, October 2002.

[13] D. Stalling, M. Zockler, and H.-C. Hege. Fast display of illuminated field lines. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):118–128, April 1997.

[14] B. Wilson, K.-L. Ma, and P. McCormick. A hardware-assisted hybrid rendering technique for interactive volume visualization. In *Proceedings of 2002 Volume Visualization and Graphics Symposium*, pages 123–130, 2002.

[15] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumes. In *Proceedings of IEEE Visualization 2003 Conference*.