SAND2020-11211J

Scientific Visualization: New Techniques in Production Software

By Kenneth Moreland and Hank Childs

The field of visualization encompasses a wide range of techniques, from infographics to isosurfaces. An important subfield called “scientific visualization” is specifically dedicated to data sets with spatial components, i.e., (X, Y, Z) locations. This subfield’s name is inspired by the fact that the data in question often come from the sciences, i.e., physics simulations or sensor networks.

Scientific visualization has a rich history of producing various general-purpose tools, such as ParaView [1], VisIt [6], SciRun [12], Tecplot, FieldView, and EnSight [7]. These tools allow the efforts of relatively few developers to impact numerous stakeholders, with millions of downloads and/or licenses. Furthermore, many other tools—such as MegaMol [8], for the visualization of molecular dynamics—are dedicated to specific scientific domains or data.

Scientific visualization “tool developers” work closely with a significant research community that regularly generates fundamentally new techniques and improvements for existing methods. Unfortunately, these research works often yield prototypes that domain scientists cannot use. However, when such results are shown to be effective, they are ultimately productized and adopted in scientific visualization tools.

Here we discuss the innovative research that experts have recently integrated into scientific visualization software. Specifically, we aim to highlight some of the top directions from the scientific visualization research community that have been translated into usable software for domain scientists. For the sake of brevity, we assume that readers are already familiar with traditional features, such as contouring, pseudocoloring, glyphing, flow tracing, and clipping.

Topology

Recent work in visualization has focused on the adoption of topological methods for data analysis. These analysis methods can operate on large, mesh-based data structures that are commonly used in science simulations. For example, researchers can utilize topological structures—including Morse-Smale complexes and Reeb graphs—for operations like feature tracking, similarity estimation, and segmentation. These techniques are applicable in numerous fields, such as combustion, materials science, chemistry, and astrophysics [16].

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.

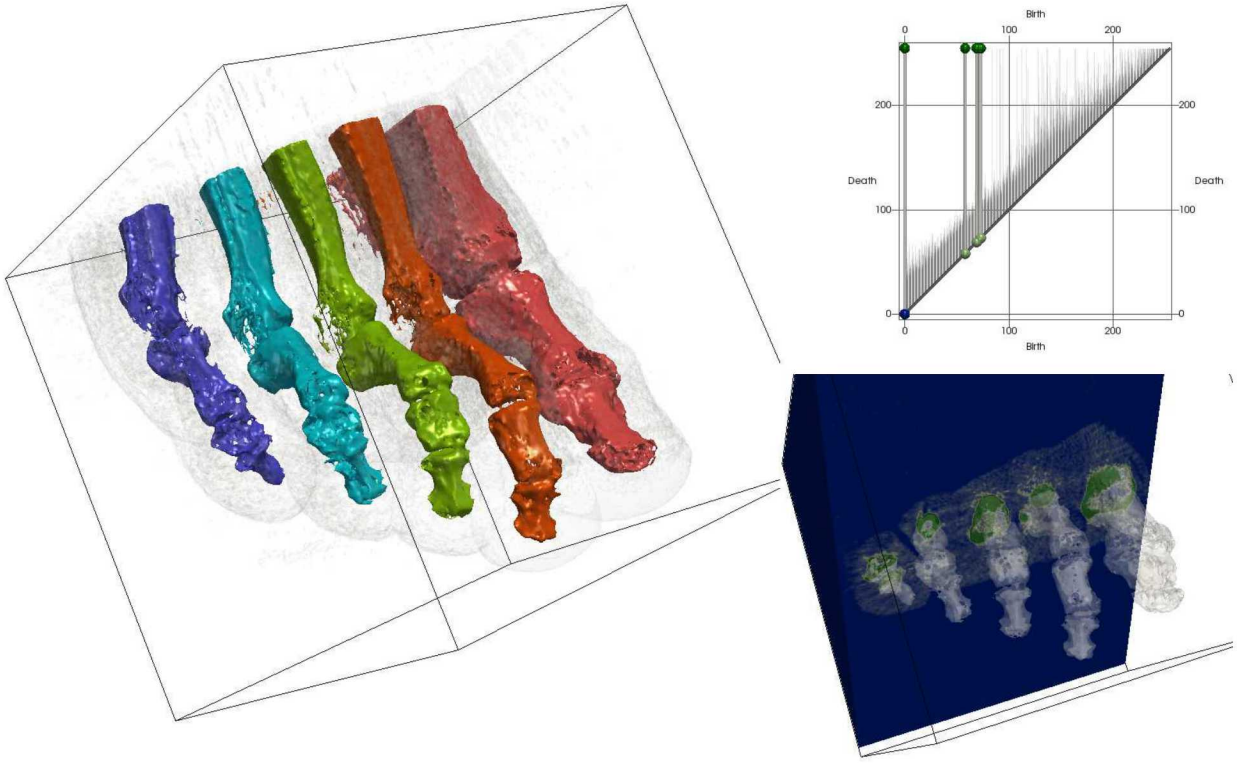


Figure 1. Use of the Topology ToolKit (TTK) to identify and separate bones in a CT scan. Image generated with the TTK tutorial.

The Topology ToolKit[1] (TTK) [16] has increased the accessibility of topological data analysis. TTK is a library with many topological analysis functions that can integrate into other software. It also provides plugins that assimilate its methods into existing software tools (see Figure 1).

Advanced Flow

Recent advances in parallel hardware and visualization software facilitate the practicality of increasingly more flow visualization techniques. Traditionally, animating particle motion and plotting particle trajectories (i.e., streamlines or pathlines) have been the most common flow visualization procedures. However, recent advances have enabled the calculation of many more particle trajectories and thus new types of flow visualizations.

[1] https://topology-tool-kit.github.io/

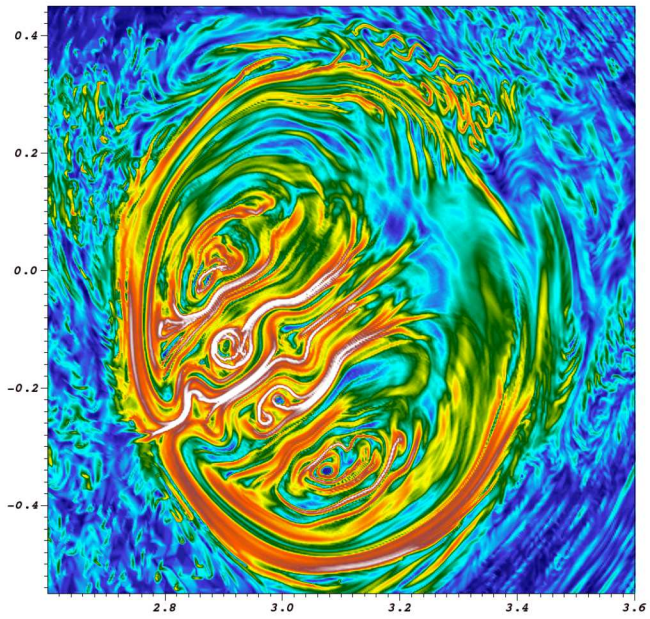


Figure 2. A finite-time Lyapunov exponent (FTLE) for a tokamak simulation, which determines how much flow separates. Blue and cyan areas are less turbulent, while red and white areas are more turbulent. Data courtesy of Linda Sugiyama.

Finite-time Lyapunov exponents (FTLE) comprise a noteworthy flow visualization technique that is derived from the tracing of many particle trajectories. This method produces a new scalar field that measures flow separation. To successfully operate, it considers neighborhoods and places multiple particles within a neighborhood. If the particles separate significantly, the scalar field for the neighborhood in question is assigned a high value; if the particles remain close together, the corresponding scalar field is assigned a low value. Figure 2 depicts an example of an FTLE.

Many additional flow techniques use particle trajectories as a foundational step. For instance, stream surfaces operate by seeding particles along a line (or curve) and constructing a surface from the resulting trajectories. Poincaré analysis considers topological structures that form when a particle repeatedly circulates through a volume. Other techniques illuminate underlying Lagrangian coherent structures.

Ray Tracing

Three-dimensional (3D) rendering has remained a staple of scientific visualization since graphics hardware became available. However, most rendering in scientific visualization utilizes approximations by independently drawing each small piece. This practice misses “global” effects like shadows, reflections, and diffuse lighting conditions.

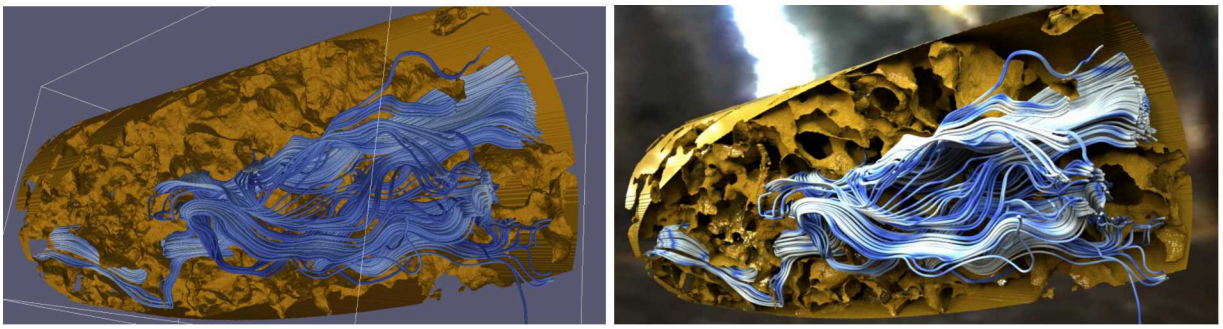


Figure 3. Use of ray trace rendering to improve visualization of a Florida groundwater core sample. 3a. Traditional raster image. 3b. Ray cast image. The porous media is much more discernable in the ray cast image than the traditional raster image. Images courtesy of Paul Navratil and Carson Brownlee. Data courtesy of Michael Sukop, Sadé Garcia, and Kevin Cunningham.

Another approach to 3D rendering is ray tracing, which traces the path of light as it bounces off objects. Recent improvements in ray tracing software—as well as increases in computational power—make interactive scientific visualization practical. Intel's OSPRay[2] [17] and NVIDIA's OptiX[3] [11] are software libraries that provide quick and realistic ray-traced rendering. Several visualization tools, including ParaView,[4] VisIt,[5] and VMD,[6] are integrating these new rendering capabilities.

In Situ Visualization

An increasingly major bottleneck on large, parallel machines is the speed at which data can be written out — the fraction of data that can be written to disk storage is often unacceptably small. To bypass this problem, simulations are turning to *in situ* visualization [5, 13], wherein the visualization is run as part of the simulation; in this process, data does not need to be written to disk storage.

Several libraries exist for *in situ* visualization of computational simulations. First, two major post hoc tools can deliver their capabilities in *in situ* form: ParaView provides Catalyst[7] [4] and VisIt offers Libsim[8] [18]. Furthermore, libraries are emerging that are devoted entirely to *in situ*. Ascent[9] is one such library [9], with foci on flyweight processing (application programming interface (API), memory usage, binary size, execution time) and support for modern supercomputers (central processing units, graphics processing units, etc. via the VTK-m[10]

[2] https://www.ospray.org/
[3] https://developer.nvidia.com/optix
[4] https://blog.kitware.com/virtual-tour-and-high-quality-visualization-with-paraview-5-6-ospray/
[5] https://tacc.github.io/visitOSPRay/
[6] https://www.ks.uiuc.edu/Research/vmd/vmd-1.9.3/
[7] https://www.paraview.org/in-situ
[8] https://www.visitusers.org/index.php?title=Libsim_Batch
[9] https://ascent.readthedocs.io/en/latest
[10] http://m.vtk.org/index.php/Main_Page

library [10]). SENSEI[11] is another example [3], with an emphasis on tool and method portability (i.e., providing an API that can access other *in situ* libraries) and proximity portability (i.e., run on the current resources or in transit resources).

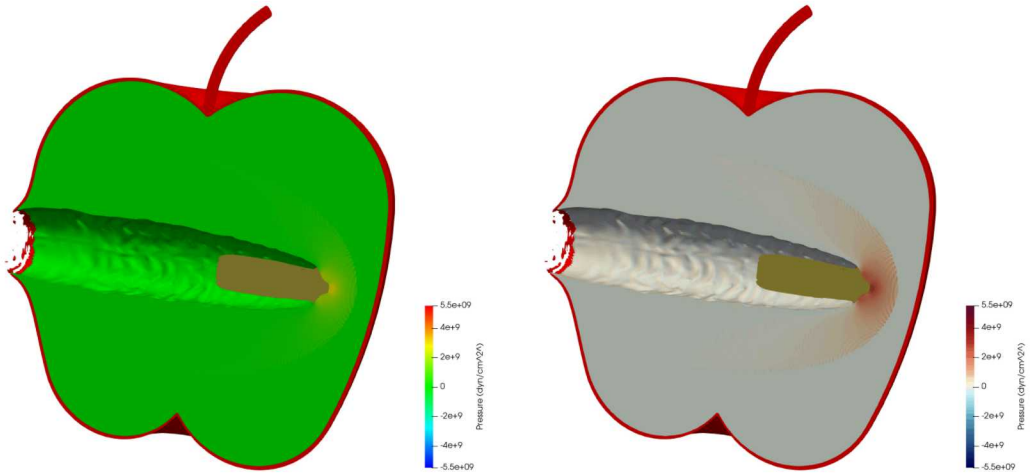
Image Exploration

A challenge of *in situ* visualization is that users cannot change the visualization once it is generated. Incorrectly setting parameters can hide important phenomena from the camera and thus cause them to be missed. One possible solution involves taking a scattershot approach that performs numerous visualizations of the same data, then organizing these visualization results into a navigable image database.

Cinema[12] is a community project that provides a specification for the organization of an assortment of visualization results [2]. Cinema databases, which comprise image files and text metadata files, are simple to generate, and Ascent, Libsim, and Catalyst directly support their creation. One can then explore these databases via a web browser or programs that run on a desktop.

Color Perception

Most scientific visualization users will recognize the red, yellow, green, and blue colors that are painted on objects to represent data. These classic colors are derived from the natural physical properties of light and can yield some attractive images. Unfortunately, research in human perception suggests that the colors are not ideal for data representation [15]; human vision is complex and does not respond proportionally to changes in light intensity and wavelength. Consequently, use of these rainbow colors can obfuscate the visualization data.



[11] https://sensei-insitu.org
[12] https://cinemascience.github.io

Figure 4. These two images depict identical data wherein only the colors that map numbers vary. 4a. The use of physical rainbow colors works to hide the pressure wave. 4b. The pressure wave is easily visible in the perceptual colors. Data courtesy of Jason Wilke.

Recent work in visualization has built color map functions that are based on models of human perception of color. These new color maps better represent the data they encode (see Figure 4). Perceptual color maps are now easily accessible in many visualization tools.

Web Delivery

Most general-purpose scientific visualization programs require that software be installed on a user's computer. But recent years have seen an increased interest in using the web as a deployment platform for scientific visualization tools. Libraries like VTK.js and ParaViewWeb adapt standard scientific visualization libraries to simplify the building of active web pages, enabling the creation of full-featured applications that run in a web browser. Tapestry[13] focuses on nimble delivery, including embedding in general web pages [14]. The user directs new visualizations via web browser interactions, with renderings on the cloud then placed in the browser as if they were generated locally.

In conclusion, scientific visualization researchers have been active in forming newly discovered tools into usable software products. We hope to have educated readers about recent improvements in scientific visualization that are useable today.

Acknowledgments

We wish to thank Chris Johnson (University of Utah) for his encouragement and support of this paper. We also thank Paul Navratil for providing example images. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy's (DOE) Office of Science and National Nuclear Security Administration (NNSA). Sandia National Laboratories is a multi-mission laboratory managed and operated by the National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for NNSA under contract DE-NA0003525. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. DOE or the U.S. government.

References

[1] Ahrens, J.; Geveci, B. & Law, C. (2005). ParaView: An End-User Tool for Large Data Visualization. *Visualization Handbook*, Elesvier.

[13] https://seelabutk.github.io/tapestry/

[2] Ahrens, J.; Jourdain, S.; O'Leary, P.; Patchett, J.; Rogers, D. H. & Petersen, M. (2014). An Image-based Approach to Extreme Scale In Situ Visualization and Analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 424-434.

[3] Ayachit, U.; Bauer, A.; Duque, E. P. N.; Eisenhauer, G.; Ferrier, N.; Gu, J.; Jansen, K. E.; Loring, B.; Lukić, Z.; Menon, S.; Morozov, D.; O'Leary, P.; Ranjan, R.; Rasquin, M.; Stone, C. P.; Vishwanath, V.; Weber, G. H.; Whitlock, B.; Wolf, M.; Wu, K. J. & Bethel, E. W. (2016). Performance Analysis, Design Considerations, and Applications of Extreme-Scale In Situ Infrastructures. In *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*.

[4] Ayachit, U.; Bauer, A.; Geveci, B.; O'Leary, P.; Moreland, K.; Fabian, N. & Mauldin, J. (2015). ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*, 25-29.

[5] Bennett, J.C., Childs, H., Garth, C., & Hentschel, B. (2019). In situ visualization for computational science (Dagstuhl seminar 18271). *Dagstuhl Reports*, 8(7), 1-43.

[6] Childs, H.; Brugger, E.; Whitlock, B.; Meredith, J.; Ahern, S.; Pugmire, D.; Biagas, K.; Miller, M.; Harrison, C.; Weber, G. H.; Krishnan, H.; Fogal, T.; Sanderson, A.; Garth, C.; Bethel, E. W.; Camp, D.; Rübel, O.; Durant, M.; Favre, J. M. & Navrátil, P. (2012). VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data, *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, CRC Press/Francis--Taylor Group, 357-372.

[7] Frank, R. & Krogh, M. F. (2013). The EnSight Visualization Application. *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, CRC Press/Francis--Taylor Group, 429-442.

[8] Gralka, P.; Becher, M.; Braun, M.; Frieß, F.; Müller, C.; Rau, T.; Schatz, K.; Schulz, C.; Krone, M.; Reina, G. & Ertl, T. (2019). MegaMol – A comprehensive prototyping framework for visualizations. *The European Physical Journal Special Topics*, 227, 1817-1829.

[9] Larsen, M.; Ahrens, J.; Ayachit, U.; Brugger, E.; Childs, H.; Geveci, B. & Harrison, C. (2017). The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman. In *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization (ISAV)*, 42-46.

[10] Moreland, K.; Sewell, C.; Usher, W.; Lo, L.-T.; Meredith, J.; Pugmire, D.; Kress, J.; Schroots, H.; Ma, K.-L.; Childs, H.; Larsen, M.; Chen, C.-M.; Maynard, R. & Geveci, B. (2016). VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications*, 36, 48-58.

[11] Parker, S. G.; Friedrich, H.; Luebke, D.; Morley, K.; Bigler, J.; Hoberock, J.; McAllister, D.; Robison, A.; Dietrich, A.; Humphreys, G.; McGuire, M. & Stich, M. (2013). GPU ray tracing. *Communications of the ACM*, 56, 93-101.

[12] Parker, S. G. & Johnson, C. R. (1995). SCIRun: A scientific programming environment for computational steering. *Proceedings ACM/IEEE Conference on Supercomputing*.

[13] Peterka, T., Bard, D., Bennett, J., Bethel, E.W., Oldfield, R., Pouchard, L., Sweeney, C., & Wolf, M. (2019). *ASCR workshop on in situ data management: enabling scientific discovery from diverse data sources*.

[14] Raji, M.; Hota, A.; Hobson, T. & Huang, J. (2020). Scientific Visualization as a Microservice. *IEEE Transactions on Visualization and Computer Graphics*, 26, 1760-1774.

[15] Rogowitz, B., & Treinish, L. (1998). Data visualization: the end of the rainbow. *IEEE Spect*., *35*(12), 52-59.

[16] Tierny, J., Favelier, G., Levine, J.A., Gueunet, Ch., & Michaux, M. (2018). The Topology ToolKit. *IEEE Trans. Visual. Comp. Graph*.

[17] Wald, I.; Johnson, G.; Amstutz, J.; Brownlee, C.; Knoll, A.; Jeffers, J.; Günther, J. & Navratil, P. (2017). OSPRay -- A CPU ray tracing framework for scientific visualization. *IEEE Trans. on Visual. and Comp. Grap*.

[18] Whitlock, B.; Favre, J. M. & Meredith, J. S. (2011). Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System. *Eurographics Symposium on Parallel Graphics and Visualization*.