Scientific Visualization: From Data to Insight

Vijay Natarajan

Scientific visualization refers to the study and design of visual representation of data that arises in science and engineering disciplines. The aim of the visualization is the identification of patterns leading to a better understanding of the underlying phenomena. This article will briefly introduce this topic by first providing the motivation in the form of applications, next describing a couple of well-studied visualization methods and algorithms, outlining some of the current research challenges in this area, and finally presenting a summary of related ongoing research work IISc Bangalore.

Vijay Natarajan is at the Indian Institute of Science, Bangalore. His research interests include scientific visualization, computational geometry, computational topology, and meshing.

1. Scientific Data

Ever increasing processing power and better imaging technology have resulted in the generation of large and feature-rich data (from simulation and experimental studies). The field of visualization relies on the cognitive capability of humans to identify and understand patterns in the data. *Scientific visualization* refers to the visualization of data that has a natural geometric context, where the domain is spatial and hence geometric. Such data often arises within science and engineering disciplines. Examples include weather simulation data that includes quantities such as temperature, pressure, wind speeds, and precipitation measured over a 3D volume, and medical imaging data such as CT scans that is available as a stack of 2D images. *Information visualization* refers to the visualization of data that has no such inherent geometric context [1,2]. Examples of such abstract data include social networks that represent the relationships between different members of an online community and census data.

Keywords
Scientific data, visualization, scalar fields, vector fields.

The field of visualization relies on the cognitive capability of humans to identify and understand patterns in the data. *Scientific visualization* refers to the visualization of data that has a natural geometric context, where the domain is spatial and hence geometric.

2. Applications

Visualization plays a crucial role in several application domains. Below, we describe three select application scenarios. This list is surely not exhaustive. Visualization continues to be used in newer fields and in innovative ways.

2.1 *Medical Visualization*

Various imaging modalities are employed to image the interior of the human body: ultrasound, magnetic resonance imaging (MRI), computed tomography (CT), near-infrared optical tomography (NIR OT). Data is typically available as a collection of density samples over a 3D lattice. Visualization of such data plays an important role in medical diagnosis, detection of tumors, surgery planning, and in robotic surgery. *Figure* 1 shows a visualization of a CT scan of a human head.

2.2 *Bio-Visualization*

The structure of protein molecules determines its function. Visualization of these biomolecules imaged using modalities such as X-ray crystallography, cryo-electron microscopy (cryo-EM), and nuclear magnetic resonance (NMR) plays an important role in understanding the

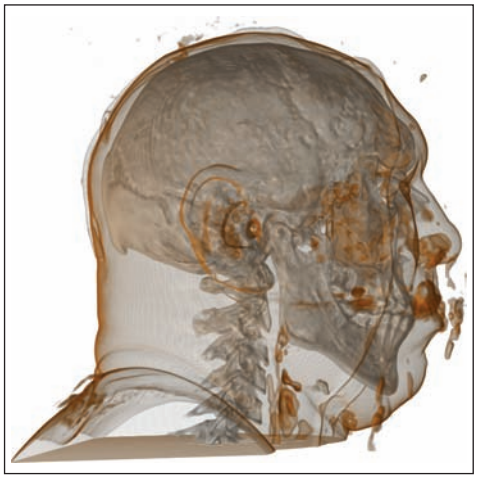


Figure 1. CT scan of a human head visualized using direct volume rendering. Scalar values are mapped to color and transparency resulting in a rendering that highlights important features in the volume such as the skin and the bone structure.

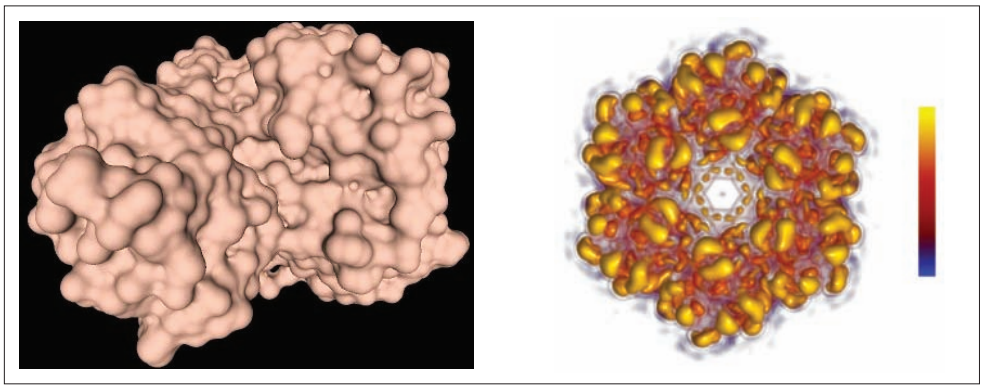


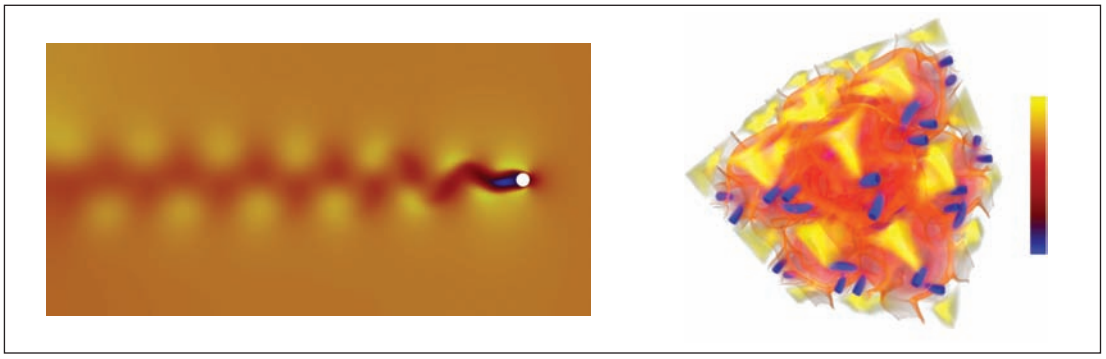
Figure 2. Left: Molecular surface of a protein (PDB ID: 1DKF) extracted given the location of all atoms. Right: Volume visualization of a haemoglobin molecule that is imaged using NMR.

geometric structure of the protein. *Figure* 2 shows the molecular surface of a protein whose structure is available from the protein data bank (PDB, http://www.rcsb.org) and a volume visualization of a haemoglobin molecule imaged using NMR.

2.3 *Flow Visualization*

Fluid flow is studied via simulations and experimental measurements in aerospace engineering, automobile engineering, meteorology/weather modeling, civil engineering, etc. Visualization plays an important role in locating features in these data. Often, these features are not well defined and hence cannot be identified using automated techniques. For example, *Figure* 3 is a visualization of data obtained from a flow simulation that shows the turbulent structures in the fluid flow. Extracting these features using automated methods is a difficult task.

Figure 3. Turbulent structures in a 2D laminar flow simulation (left) and a 3D Taylor–Green vortex flow simulation (right).



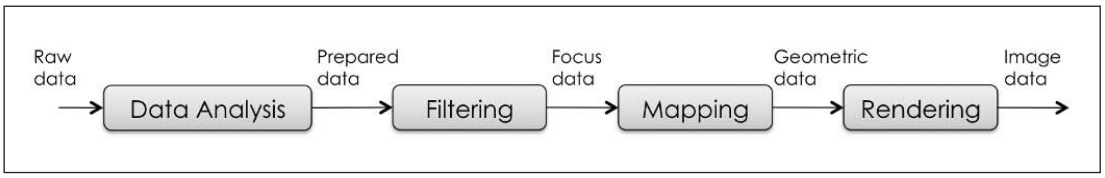


Figure 4. Visualization pipeline. Raw data is converted into image data (pixels) in four stages.

3. Visualization Pipeline

The visualization process is often described as a pipeline that contains four major stages. The pipeline converts *raw data* into *image data* (pixels) in four stages, (see *Figure* 4).

Data Analysis: The data is prepared for visualization in the first stage by converting raw data into *processed data*. For example, missing values may be filled in by interpolation, denoising operators may be applied to correct erroneous measurements, or a smoothing operator may be applied to make the data amenable for visualization. This stage is typically computer centered and requires minimal user interaction.

The visualization pipeline converts raw data into image data in four stages.

Filtering: This stage identifies the portions of the data that are of interest by applying user-directed filters resulting in a *focus data*.

Mapping: This stage essentially maps the focus data into geometric primitives together with attributes like color, texture, size, etc. This is the crucial stage of the pipeline because it determines the expressiveness and effectiveness of the visualization. The methods described in this article belong to this stage.

Rendering: The final stage of the pipeline converts the *geometric data* into image data using techniques and methods from computer graphics.

4. Classical Techniques

Each scientific discipline has adopted and developed its own set of specialized methods for visualizing data. In

contrast, the scientific visualization community focusses on the development of visualization techniques, tools, and technology that are widely applicable across different science and engineering disciplines and to a larger class of data sets. We will describe two classical techniques that have such wide applicability: isosurface extraction and streamline generation.

4.1 *Isosurfaces*

Data from measurement devices and simulations is often available as scalar value samples over a domain. These data sets are represented as scalar functions that map each point in space to a real value. Given a sample of scalar values over a domain, different interpolation techniques are used to derive a continuous and, if necessary, smooth representation of the function. For example, X-ray crystallographers compute the electron density at various points of a molecular crystal using diffraction measurements from X-rays bouncing off the crystal. It is essential to know the electron density to perform structure-related studies of the molecule. The electron density is a scalar function typically defined on a subset of the three-dimensional Euclidean space, $\mathbb{R}^3$. Another example is magnetic resonance imaging (MRI), a popular technique used to take pictures of different slices of the human body. The proton density available from the MRI scan over a slice is mapped to a gray-scale image and studied by radiologists to detect tumors. The scalar function in this case is the density defined on a set of two-dimensional planes stacked together and can be viewed as a function defined on a subset of $\mathbb{R}^3$.

A classical method for visualizing 2D and 3D scalar functions is via extraction and interactive exploration of *isosurfaces*.

The domain is often represented by a lattice grid and the samples available at lattice points are interpolated within each cell of the grid. The resulting scalar function, also called a *scalar field*, is bilinear (for 2D data) or trilinear (for 3D data) within each cell, (see *Figure* 5). The value of the function within a cell is defined as

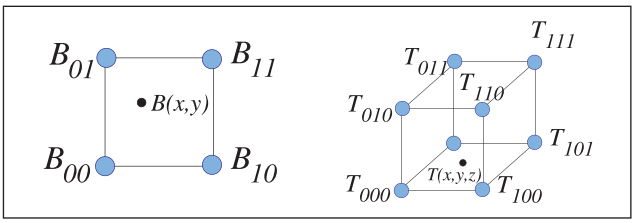


Figure 5. Grid cells. Bilinear interpolation within a 2D cell (left) and trilinear interpolation within a 3D cell (right).

$$
\begin{aligned}
B(x,y) &= B_{00}(1-x)(1-y) + B_{10}x(1-y) \\
&+ B_{01}(1-x)y + B_{11}xy\ , \\
T(x,y,z) &= T_{000}(1-x)(1-y)(1-z) \\
&+ T_{100}x(1-y)(1-z) \\
&\quad + T_{010}(1-x)y(1-z) \\
&+ T_{110}xy(1-z) \\
&\quad + T_{001}(1-x)(1-y)z + T_{101}x(1-y)z \\
&+ T_{011}(1-x)yz + T_{111}xyz.
\end{aligned}
$$

Note that the value of B at a point (x, y) can be computed by linear interpolation along the x-axis followed by linear interpolation along the y-axis.

A classical method for visualizing 2D and 3D scalar functions is via extraction and interactive exploration of *isosurfaces*. An isosurface[1], also called a *level set*, is the pre-image of a scalar value.

[1] We use the term isosurface loosely to refer to level sets, which could be 1D curves or 2D surfaces.

4.2 *Marching Cubes Algorithm*

A popular algorithm that computes the isosurface is called the Marching Cubes algorithm, which iterates over all cells of the input grid, computes the isosurface within a cell using a linear approximation, and stitches together the pieces from all cells into the resulting isosurface. We will first describe a simple version of this algorithm focussing on 2D data[2]. *Figure* 6 illustrates the result of the marching cubes algorithm applied on a 6×6 grid with scalar values specified at lattice points. The algorithm processes the input one cell at a time. A cell is classified as *active* if the isocontour passes through it. Active cells are determined by simply checking if the

[2] We use the term 'marching cubes' generically to refer to the algorithm working on input in any dimension. The algorithm for 2D input is often referred to as the Marching Squares algorithm.

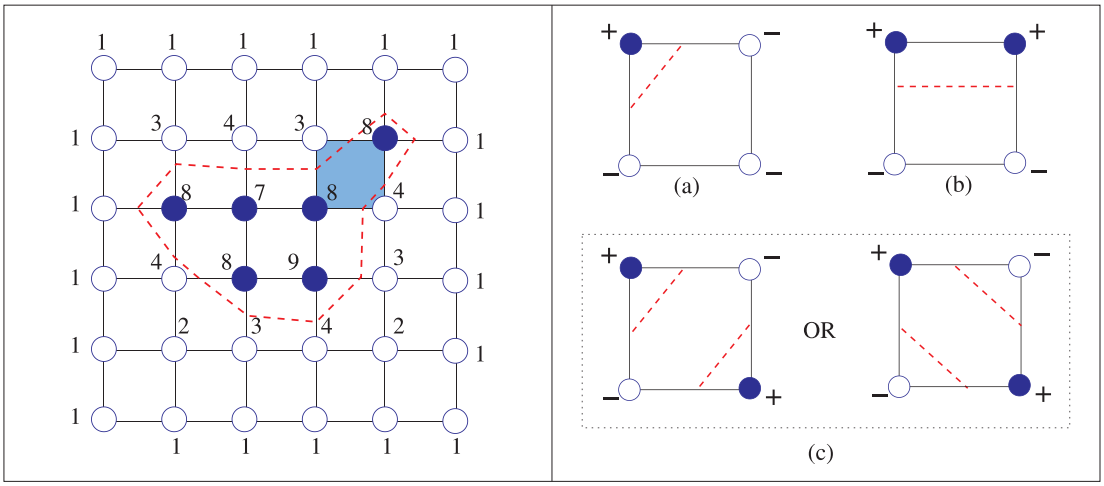


Figure 6 (left). Level set extracted by the marching cubes algorithm for an isovalue of 5. The level set (red dashed line) is computed within each cell using the case table. Note that if the ambiguous case (shaded cell) were handled differently, the resulting level set would consist of two components instead of one.

Figure 7 (right). Case table for the 2D marching cubes algorithm. Case (c) is ambiguous because there are two possible templates for the level set.

isovalue lies within the range of values assumed by the scalar function B in the cell. Since B is a bilinear function, it attains the minimum (m) and maximum (M) values at vertices of the cell. An active cell is identified by checking if the isovalue lies in the interval $[m, M]$.

The algorithm further processes a cell only if it is active. A cell is classified as active depending on the number of vertices of the cell whose function value is lesser than the isovalue. A vertex (lattice point) is positive ('+') if the value of B at the vertex is greater than the isovalue and negative ('–') if the value of B at the vertex is smaller than the isovalue. Assuming that the function value at all vertices is different from the isovalue[3], there are 2^4 different possible configurations. Two configurations, where all vertices are '+' or all vertices are '–', correspond to inactive cells. Active cells are classified into three unique configurations, as shown in *Figure* 7. Other configurations are either obtained by rotation or interchanging the labels. The marching cubes algorithm queries this *case table* to generate the level set within the cell. The case table contains templates for the level sets corresponding to each configuration.

The level set template for each cell is generated by computing the *isopoints* lying on grid edges where the value

[3] Degenerate cases such as when the function value at a vertex is equal to the isovalue can be handled explicitly via a case analysis or by assuming that the label is '+' in this case.

of B equals the isovalue. Since B is linear along the axis, a grid edge whose end points have labels '+' and '–' will contain an isopoint whose location is determined by solving a linear equation. A straight line segment joining the isopoints is an approximation of the level set curve within the cell. Case (c) in *Figure* 7 is ambiguous because there are two legal ways to connect the isopoints. There are several known approaches to resolve this ambiguity. Nielson and Hamann [3] presented an approach that identified the correct pairs of isopoints to connect by locating the saddle point of the bilinear function. A simple but naive approach is to consistently choose one of the two templates, say the one on the left in *Figure* 7.

The output of the algorithm is essentially a collection of edges representing the level set. Note that level set edges from adjacent cells will share an end point and hence the output is a collection of continuous piecewise linear curves. Marching cubes algorithm for 3D input is essentially the same as described above except for the case table, which consists of 15 unique configurations. The interested reader may refer to the book by Hansen and Johnson for additional details on this algorithm [4].

4.3 *Streamlines*

Data from flow simulations often consists of both scalar-valued and vector-valued data. For example, weather simulations produce scalar data such as pressure, temperature, and precipitation levels and vector data such as wind velocity. Similar to scalar data, the domain for the vector data is also often represented using a lattice grid. The vector data is available as a collection of d-dimensional vector-valued samples at lattice points on the domain. The vector values are linearly interpolated coordinate-wise in the interior of the grid cells resulting in a vector-valued function, called a *vector field*.

Direct visualization of the vector field essentially involves designing a glyph to represent the vector at each lattice point, or a subset of lattice points (see *Figure* 8). The lengths of the arrow in the figure represents the magnitude and its orientation represents the direction of the vector value at the point. Clearly, this visualization does not scale to large data due to visual clutter.

A streamline is an extremal curve whose tangent at any point is equal to the vector value at the point.

A popular vector field visualization method involves the computation of *streamlines*. A streamline is an extremal curve whose tangent at any point is equal to the vector value at the point. It is computed via numerical integration beginning at multiple seed points within the domain. The seed points are often carefully chosen to generate either equally spaced streamlines or to generate streamlines that represent key features in the vector field.

Given a vector field V, a streamline can be expressed as an integral

$$p(t) = \int_0^t \vec{v}(t)\,\mathrm{d}t,$$

where $p_0 = p(0)$ is the seed point. Given a seed point, the streamline can be computed via numerical integration using Euler's forward method. Numerical computation begins with the seed point and uses discrete steps to compute points on the curve. The point p_{i+1} is computed by displacing p_i along the instantaneous velocity vector at p_i by a distance δt,

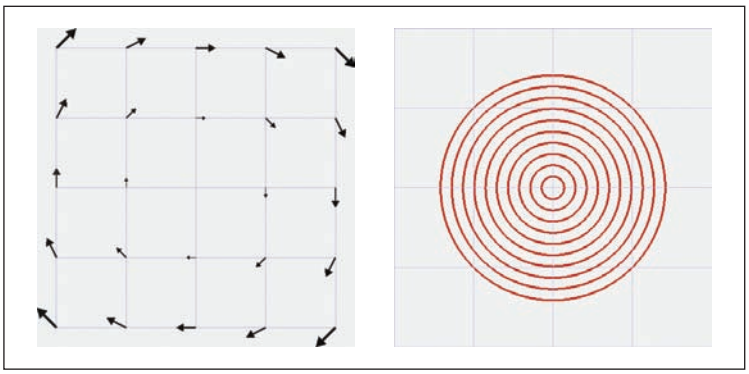


Figure 8. Vector field visualization. A rotational vector field visualized using glyphs (left) and streamlines (right). The length of the glyph is proportional to the magnitude of the velocity. The streamline is computed given 10 seed points using Runge–Kutta method of order 2.

$$p_{i+1} = p_i + \vec{v}_i \delta t.$$

Euler's method may not be accurate enough for all applications. The errors from numerical integration are particularly high in regions with high velocity magnitude. For example, given a rotational vector field as in *Figure* 8, streamlines computed using Euler's method are almost always spirals and not circles. Higher order methods such as the Runge–Kutta technique guarantee better accuracy at the expense of computational cost. Using the Runge–Kutta method of order 2, the streamline is computed as

$$p_{i+1} = p_i + \frac{\delta t}{2}(\vec{v_i} + \vec{v}_{i+1}).$$

If the vector field is constant over time, then the streamlines are equivalent to *particle traces* and *streaklines*, which are two additional popular techniques for visualization of vector fields. A particle trace follows the trajectory of a particle under the influence of the, possibly time-varying, vector field. A streakline is the set of points corresponding to all particles that passes through a given point on the domain. In other words, dye that is injected at a fixed point will extend along a streakline. 3D vector fields may also be visualized using glyphs, streamlines, and streamsurfaces. However, occlusion and visual clutter limit the applicability of these techniques. Nevertheless, they are often employed to gain insight into the vector field. The interested reader may refer to the book by Hansen and Johnson for a survey of vector field visualization techniques [4].

5. Research Challenges

Scientific visualization is a young field[4] and there are several fundamental problems that remain open. Below, we list three research challenges that are of current interest. The article by Johnson [5] and the NSF/NIH

[4] The field of scientific visualization was established by a report by the National Science Foundation, USA, titled *Visualization in Scientific Computing* in 1987.

report on *Visualization Research Challenges* [6] discuss these problems and other challenges in greater detail.

5.1 *Large Data Visualization*

Data generated from current and next generation simulations and experiments are in the order of petabytes and in some cases exabytes. Existing methods for data visualization, and in general for data mining and data analysis, do not scale to such large data sets. The success of these science and engineering experiments depends crucially on the methods used to process the data to extract information and discover patterns. Visualization of these large data sets requires an interdisciplinary and unified effort by bringing together techniques from databases for large data management, data mining for large data analysis, high performance computing for processing the data, and statistics for presenting useful summaries. New algorithms are also required for high quality rendering and interactive exploration of these large data sets at interactive speeds via effective utilization of the powerful graphics processors and multi-core CPUs that are available today at affordable rates.

5.2 *Topological Methods*

Simulations and experiments in science and engineering are generating data that is increasingly complex in terms of the number of features in addition to being large in size. New methods for feature detection, extraction, comparison, and tracking are required for effective visualization and exploration of these feature-rich data sets. Classical visualization techniques and methods as described in the previous section are not effective when applied to such large and feature-rich data. Topology-based methods, on the other hand, provide abstract representations of features in the data and may be more effective [7]. Consider the example of isosurface-based visualization and exploration of scalar fields. For large data sets, the size of the isosurface and visual clutter are

New methods for feature detection, extraction, comparison, and tracking are required for effective visualization and exploration of these feature-rich data sets.

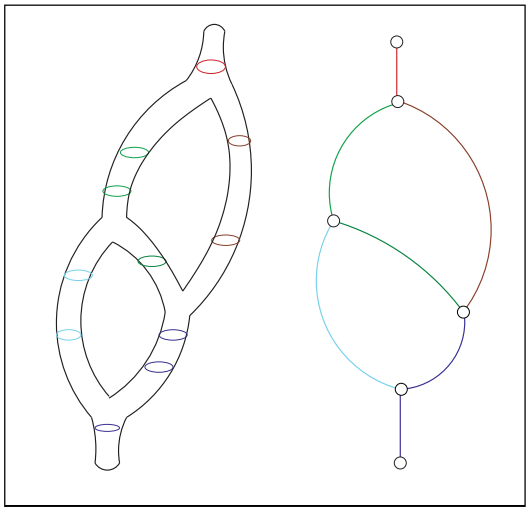


Figure 9. Reeb graph of a height function defined on a surface (left) with two tunnels. Reeb graph (right) tracks the topology of level sets.

two factors that hinder the effective exploration of the scalar field. The Reeb graph [8] is a topological structure that tracks the evolution of the topology of isosurfaces. *Figure* 9 shows the Reeb graph for a simple scalar field. It is a succinct and abstract representation of features, defined in terms of topology or connectivity, in the data and also aids in controlled feature simplification. Further, the Reeb graph serves as an interface and facilitates feature-directed exploration of the data. Several other topological structures have been studied in the context of visualization of scalar and vector fields: Morse–Smale complexes, Jacobi sets, persistence diagrams, separatrix structures, and Morse connection graphs. Open problems include the design of novel topological structures, efficient computation of these structures, demonstrating their applicability to visualization either directly or in conjunction with other geometry- or statistics-based methods.

Topology-based methods provide abstract representations of features in the data and generate effective visualizations.

5.3 *Multi-field Visualization*

Simulation studies often produce data corresponding to multiple quantities – for example, a weather simulation will produce temperature, pressure, precipitation and wind speed distributions besides other field variables.

Scientists study the relationship between these quantities with the aim of understanding the underlying phenomena. The visualization techniques described in the previous section focussed on a single field variable (isosurfaces for scalar fields or streamlines for vector fields). Novel techniques are required to visualize multiple scalar and vector fields together with the interactions between them. Recent efforts have employed varying approaches including fusion of different visual channels, design of sophisticated glyphs, explicit visualization of the interactions between the fields via the computation of one or more derived fields, and feature analysis and comparison. While these results are impressive and interesting they are still few in number and address only some of the known open issues. Several fundamental challenges remain open in multi-field visualization and it remains an important direction for future research activities.

6. Scientific Visualization, IISc

We conclude with a brief description of research in scientific visualization at the Indian Institute of Science (IISc), Bangalore. The Visualization and Graphics Laboratory (VGL) at IISc, Bangalore serves as a facility for conducting research in the areas of scientific visualization, geometry processing, computational topology, and computer graphics. Current topics of research include topological modeling for visualization, identification of symmetry in scientific data, development of feature-directed visualization methods, development of integrated simulation and visualization frameworks, and applications of visualization in life sciences and engineering. *Figures* 10 and 11 provide a glimpse of research on symmetry identification and topology-directed interactive exploration of volumetric data.

The Visualization and Graphics Laboratory (VGL) at IISc, Bangalore serves as a facility for conducting research in the areas of scientific visualization, geometry processing, computational topology, and computer graphics.

Several ongoing research projects focus on the construction of novel geometric and topological structures over scientific data. These structures aid the analysis and

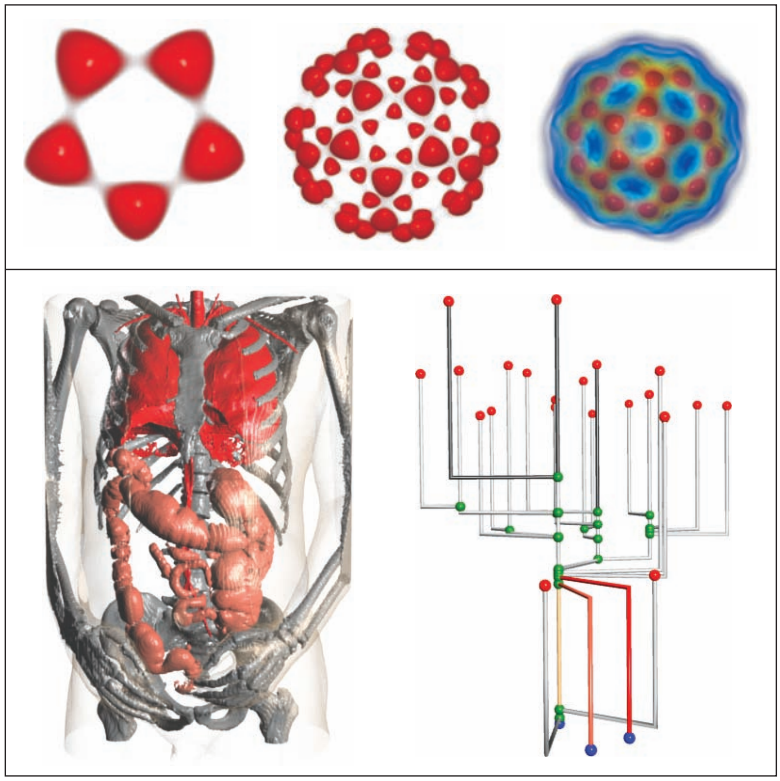


Figure 10 (top). Symmetry identification. The buckyball contains 60 carbon atoms. Symmetric structures are identified at different resolutions from the electron density distribution around the buckyball molecule.
Figure 11 (bottom). Topology-directed volume exploration. Left: Volume rendering of the torso of the visible human data set (courtesy NIH). Right: Visualization of the Reeb graph that tracks the topology of level sets of the input. Arcs of the Reeb graph correspond to features of interest. Individual arcs can be selected using a visualization tool. Volumetric regions corresponding to the selected arcs can be rendered using an explicitly defined transfer function, which maps scalar values to color and opacity.

visualization of data by automatically locating interesting features. Work done in VGL has led to the development of robust and computationally efficient combinatorial algorithms to construct key topological structures: Morse–Smale complex, Reeb graph, and Jacobi set. Members of the lab also actively work together with application domain scientists to ensure that the technology is transferred and applied to the study of interesting and challenging problems within that discipline. More

details about individual research projects including publications and software is available at http://vgl.serc.iisc.ernet.in.

Acknowledgements

The volume visualizations in this article were generated by Harish Doraiswamy and Dilip Thomas. I thank all members of the Visualization and Graphics Lab, IISc for various discussions.

Suggested Reading

[1] **M O Ward, G Grinstein and D Keim, *Interactive Data Visualization: Foundations, Techniques, and Applications*, CRC Press, 2010.**

[2] **C Ware, *Information Visualization: Perception for Design*, Morgan Kaufmann, 2004.**

[3] **G M Nielson and B Hamann, The asymptotic decider: resolving the ambiguity in marching cubes, *Proc. IEEE Conf. Visualization*, pp. 83–91, 1991.**

[4] **C D Hansen and C R Johnson, *Visualization Handbook*. Academic Press, 2004.**

[5] **C R Johnson, Top Scientific Visualization Research Problems, IEEE Computer Graphics and Applications, Vol.24, No.4, pp.13–17, 2004.**

[6] **C R Johnson, R Moorhead, T Munzner, H Pfister, P Rheingans, and T S Yoo, *NIH/NSF Visualization Research Challenges Report*, IEEE Computer Society Press, 2006.**

[7] **Y Matsumoto, *An Introduction to Morse Theory*, Translated from Japanese by K Hudson and M Saito, *Amer.Math.Soc.*, 2002.**

[8] **H Doraiswamy and V Natarajan, Computing Reeb graphs as a union of contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 2012, PrePrints.**

Several general purpose visualization tools have been developed by researchers and are made available for free [9,10,11,12,13].

[9] **ParaView. http://www.paraview.org**

[10] **W Schroeder, K Martin, and Bill Lorensen, The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, Kitware Inc. Publishers, 2004.**

[11] **SCIRun. http://www.scirun.org**

[12] **VisIt Visualization Tool. http://wci.llnl.gov/codes/visit/**

[13] **VTK - The Visualization Toolkit. http://www.vtk.org**

Address for Correspondence
Vijay Natarajan
Department of Computer Science and Automation
Supercomputer Education and Research Centre
Indian Institute of Science
Bangalore 560 012, India.
Email: vijayn@csa.iisc.ernet.in