Volume 0 (1981), Number 0 pp. 1–26 COMPUTER GRAPHICS *forum*

Graphs in Scientific Visualization: A Survey

Chaoli Wang and Jun Tao

University of Notre Dame
{chaoli.wang, jtao1}@nd.edu

Abstract

Graphs represent general node-link diagrams and have long been utilized in scientific visualization for data organization and management. However, using graphs as a visual representation and interface for navigating and exploring scientific data sets has a much shorter history yet the amount of work along this direction is clearly on the rise in recent years. In this paper, we take a holistic perspective and survey graph-based representations and techniques for scientific visualization. Specifically, we classify these representations and techniques into four categories, namely, partition-wise, relationship-wise, structure-wise, and provenance-wise. We survey related publications in each category, explaining the roles of graphs in related work and highlighting their similarities and differences. At the end, we reexamine these related publications following the graph-based visualization pipeline. We also point out research trends and remaining challenges in graph-based representations and techniques for scientific visualization.

Keywords: *graphs, scientific visualization, user interface and interaction, visual analytics*

Categories and Subject Descriptors (according to ACM CCS): E.1 [Data]: Data Structures—Graphs and networks I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques H.5.2 [Information Interface and Presentation]: User Interfaces—Graphical user interface

1. Introduction

Trees and graphs are well-known structures that use node-link diagrams to represent data relationships. These visual mappings and representations are at the heart of information visualization techniques. Scientific visualization, on the other hand, deals with three-dimensional spatial data that are typically time-varying and multivariate, including scalar and vector quantities. Trees and graphs have long been used to organize and manage scientific data sets for processing and rendering. In these scenarios, they are mostly utilized as an internal data representation.

Over the past decade, we have witnessed the great marriage of scientific visualization and information visualization. Information visualization techniques such as parallel coordinates and treemaps have been successfully utilized to assist a number of scientific visualization tasks including correlation exploration, level-of-detail selection, and transfer function specification. Unlike straightforward applications of parallel coordinates and treemaps that encode co-occurrence and hierarchical relationships among data items, leveraging the more generic and powerful visual means of graphs often requires a fully integrated pipeline of data transformation, representation, and visual mapping.

Recently, there is a flourishing of works that utilize trees and graphs as visual mappings and interfaces to help scientific visualization tasks. These solutions extract relationships from high-dimensional data over space and time, display these relationships as graphs in a low-dimensional space, and allow users to perform queries and make connection to the original data. This transformative way of exploring scientific data holds the promise to address scientific visualization problems whose size, complexity, and need for closely coupled human and machine analysis may make them otherwise intractable.

We classify related publications in scientific visualization that use trees and graphs into four categories: *partition-wise*, *relationship-wise*, *structure-wise*, and *provenance-wise* representations and techniques. Partition-wise graph representations are concerned with partitioning spatiotemporal data toward effective organization and rendering. These tech-

		tree					graph						internal	external			steady data	unsteady data		scalar field			vector field	
			binary tree	quadtree	octree	16-tree	hybrid	directed	undirected	weighted	unweighted	hierarchical	compound			w/o interaction	w interaction	combined view	separate view			univariate	multivariate		others
partition-wise	wavelet tree [GWGS02]				x									x						x			x		
	temporal hierarchical index tree [She98]		x											x						x			x		
	BONO [WVG92]				x									x						x			x		
	multi-dimensional trees [WVG94]	x												x						x				x	
	TSP tree [SCM99]					x								x						x			x		
	WTSP tree [WS04]					x								x						x			x		
	multiresolution video [FJS96]					x								x						x					x
	SPT tree [DCS09]					x								x						x			x		
	binary swap [MPH94]		x											x											x
	2-3 swap [YWM08]	x												x											x
	hierarchical navigation interface [WS05]					x										x	x			x			x		
	LOD map [WS06a]				x											x	x			x			x		
relationship-wise	MDMap [PIB*11]								x		x		x			x	x			x			x		
	iTree [GW13]	x														x	x			x			x		
	Multifield-Graphs [STS06]								x		x		x			x	x			x				x	
	attribute cloud [JBS08]								x		x					x	x			x				x	
	probability association graph [LS16]							x		x						x	x			x				x	
	ε-machine [JS09]							x		x						x	x			x			x		
	TransGraph [GW11]							x		x		x				x	x			x			x		
	Flow Web [XS10]							x		x		x				x	x			x					x
	FlowGraph [MWSJ14]							x		x		x	x			x	x			x					x
	binary connectivity graph [BSL*14]								x		x					x		x							x
	ontology instance graph [KPM*08]							x		x		x				x	x			x			x		
	neuroMap [SBS*13]							x		x			x			x	x			x					x
	VisNEST [NSvA*13]							x		x			x			x	x			x					x
	MobilityGraph [vLBR*16]							x		x		x				x	x			x					x
	TrajGraph [HZY*16]							x		x		x				x	x			x					x
	feature correspondence graph [SSZC94]							x			x			x						x			x		
	Petri Nets [OSBM14]							x			x			x						x			x		
	CompuCell3D [HSS*12]								x		x					x		x		x					x
	cavity graph [PTRV12]								x		x						x	x		x			x		
	propagation graph [WLY*13]							x		x							x	x		x					x
	flow graph [NLS11]							x		x				x						x					x
	access dependency graph [CNLS12]							x		x				x						x			x		
structure-wise	contour spectrum [BPS97]	x															x	x		x			x		
	contour tree interface [CSvdP04]	x															x	x		x			x		
	skeleton tree [GKHS98]	x												x						x			x		
	volume skeleton tree [TTFN05]	x												x						x			x		
	topological landscape [WBP07]								x								x	x		x			x		
	cancellation tree [BPH05]	x												x						x			x		
	topological spine [CLB11]								x								x	x		x			x		
	empty region graphs [CL11]								x					x						x			x		
	topology change graph [SB06]							x									x	x		x			x		
	attributed relational graph [MBB*12]								x		x						x	x		x				x	
	augmented extremum graph [TN13]									x				x						x			x		
	extended BDG [SSW14]							x						x						x			x		
	relation graph [CQC*08]							x									x	x		x			x		
	saddle connectors [TWHS03]							x											x	x					x
	joint contour net [CD14]							x								x		x		x				x	
	simplicial chain graphs [RL14]								x		x						x	x			x			x	
provenance-wise	image graph [Ma99]								x		x						x	x		x			x		
	spreadsheet-like interface [JKM01]							x									x	x							
	derivation graph [JKMG02]								x		x					x		x							x
	volume tree [WS06b]	x															x	x		x			x		
	storyboard [LS08]	x															x	x		x			x		
	event graph [YLRC10]							x									x	x		x			x		
	lineage tree [WWB*14]	x														x		x		x					x
	VisTrails [BCC*05]								x		x						x	x							x
	interaction graph [GS06]								x		x						x	x							x

Figure 1: *Selected tree and graph representations from the four categories.*

niques can be traced back to a large body of early work that aimed at fast rendering of large data sets with limited memory available. Relationship-wise graph techniques encompass similarity-based hierarchical data clustering, various relationship graphs based on correlation, transition or correspondence, as well as general- and special-purpose trees and graphs for parallel and out-of-core visualization algorithms. Structure-wise graph techniques mainly study topological structures of data which can be treated as a special case of relationship-wise graph techniques. We separate them into a different category due to a variety of topological trees and graphs applied to scalar, vector and multifield data visualization. Finally, provenance-wise graph representations deal

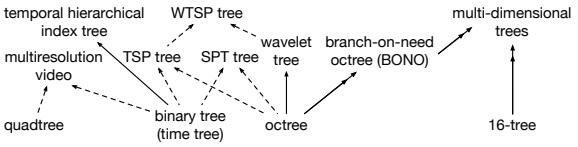


Figure 2: *Various trees for partition-wise graph representations and techniques. A solid line with single arrow head represents a variant, extension, or special case while a solid line with double arrow heads represents a generalization. Dashed lines represent a combination.*

with visualization history and parameter management, storytelling, animation creation, and reproducible visualization.

1.1. Scope of Survey

To collect related publications for this survey, we started with the collection of papers we knew of from our own research, then scanned through three major visualization conferences (*IEEE Visualization Conference*, *Eurographics Conference on Visualization*, *IEEE Pacific Visualization Symposium*) and two major visualization journals (*IEEE Transactions on Visualization and Computer Graphics*, *Computer Graphics Forum*) to identify related papers. In addition, around the periphery of scientific visualization, there are many applications focusing on visual analytics of biological network data and spatiotemporal movement data where graphs naturally play an important role. We therefore expand our search to include three related venues (*IEEE Symposium on Biological Data Visualization*, *Eurographics Workshop on Visual Computing for Biology and Medicine*, and *IEEE Symposium on Visual Analytics Science and Technology*).

The criteria for us to select a paper are that the work is relevant to tree and graph based representations and techniques, and the work is within the context of scientific visualization. Therefore, the large collection of graph visualization papers in information visualization is not included. We also omit the works that transform scientific data into different kinds of scatterplots or projections using techniques such as principal component analysis (PCA) and multidimensional scaling (MDS) since there is no explicit edges defined to construct the graph view. Furthermore, we do not survey graph drawing and layout algorithms, but focus on their application and utilization in scientific visualization. We refer readers to the following survey papers [HMM00, vLKS*11, BBDW14] for graph visualization, large graph analysis, and dynamic graph visualization.

Needless to say, including all relevant publications is beyond the scope of this survey. For example, as a general data structure, the octree has been widely used for partitioning 3D volume data. As such, we only picked a small collection of representative papers in this direction. Furthermore, we gave higher priority to new and interesting use of graph techniques for visual representation and navigation than conventional use of well-known tree and graph structures for data organization and management. Finally, some graph representations, such as Reeb graphs, have found a wider variety of applications in other fields such as computer graphics and computational geometry instead of scientific visualization. We therefore briefly mentioned these representations rather than giving an extensive review.

1.2. Overview of Different Representations

To help readers gain an overview of various trees and graphs surveyed, we list selected tree and graph representations from the four categories in Figure 1, pointing out their respective tree or graph types, representation modes, and kinds of data handled. For partition-wise and structure-wise techniques, we further illustrate the relationships among different trees and graphs in Figures 2 and 3, respectively. We do not present such an illustration for relationship-wise or provenance-wise techniques. This is because they either have a much broader range of topic (relationship-wise) which makes the illustration inadequate or have a much narrower scope of focus (provenance-wise) which makes the illustration unnecessary.

2. Partition-wise Representations and Techniques

Many of the early research efforts in scientific visualization leveraged tree-like structures to organize and manage three-dimensional volumetric data. The main idea is to create a tree hierarchy by recursively subdividing the domain and preparing data with different resolutions for compression, representation and visualization. Such a hierarchical structure allows us to use low resolution data for regions with low importance in order to speed up the subsequent processing and rendering. As a matter of fact, the same idea has been widely used in image and video processing, albeit in lower dimension. One can build such tree structures in any dimension while the most common forms are in 1D to 4D: binary tree (1D), quadtree (2D), octree (3D), and 16-tree (4D). These trees are normally built from 1D signal or time series, 2D image, 2D video or 3D volume, and 3D time-varying volume, respectively. For 2D video data, the third dimension of time t is treated equally as the two spatial dimensions x and y. Likewise, for 3D time-varying volume data, the fourth dimension of time t is treated equally as the three spatial dimensions x, y and z.

In its simplest form, the domain is *evenly* partitioned along each dimension *simultaneously* and the partitioning planes are always aligned with the corresponding coordinate axes. In practice, we often introduce several variations to improve the flexibility, efficiency and effectiveness of such solutions. First of all, the partition could be adaptive. For example, we do not partition a region further if the data within the region meet certain criteria (e.g., the error within the block is less than a given threshold or the block has reached the minimum block size). This saves unnecessary partitions and is thus more efficient. Second, the partition could be uneven or could follow one axis at a time. For the former case,

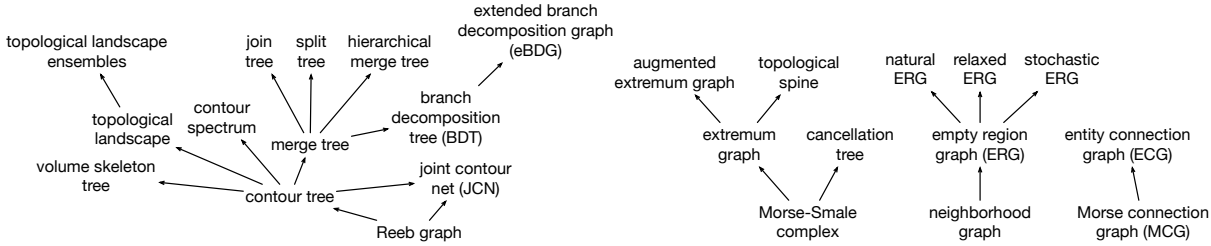


Figure 3: *Various trees and graphs for structure-wise graph representations and techniques. The arrow line represents a variant, extension, or special case.*

we can adjust the partition based on data properties and produce a tighter partitioning, yielding a tree with less hierarchical levels. For the latter case, we would produce a partitioning similar to the *k-d tree*. These strategies are adopted if we want to have a similar number of data elements or items in each leaf node of the tree in order to speed up the subsequent searching or indexing. Third, the partition does not have to be axis-aligned. This leads to *binary space partitioning trees* (BSP trees) which are more general than k-d trees as partitioning planes may have any orientation rather than being aligned with the coordinate axes.

2.1. Standard Tree Structures

In volume visualization, standard octree structures have long been used for data encoding and rendering. An early work of Meagher [Mea82] uses octree encoding for geometric models and volumetric data. Levoy [Lev90] utilized a binary octree to skip empty regions for efficient ray tracing of volume data. Laur and Hanrahan [LH91] stored the volume data using an octree, sorted the cells from back to front, and calculated cell projections for hierarchical splatting. LaMar et al. [LHJ99] used an octree hierarchy for texture-based volume rendering where the octree nodes store volume blocks resampled to a fixed resolution and rendered using 3D texture hardware. Boada et al. [BNS01] presented an octree-based structure for multiresolution volume visualization where the hierarchical texture memory representation and management policy benefits nearly homogeneous regions and regions of lower interest. Carmona and Froehlich [CF11] designed a greedy, real-time cut update algorithm based on split-and-collapse operations for octree-based multiresolution volume raycasting using GPU.

Different data reduction and compression schemes have also been incorporated into octree-based volume visualization. Examples include *Laplacian pyramids* [GY95], *wavelet trees* [GWGS02, GWLS05], application-driven compression [WYM10], and tensor approximation and reconstruction [SIGM*11, SMP13].

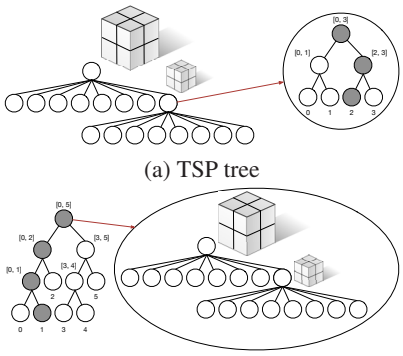
The common way to partition the time dimension is to iteratively partition the time span into half to build a binary tree. For instance, the *temporal hierarchical index tree* [She98] is a structure for creating the isosurface cell search indices from a time-varying field. Cells are characterized based on their extreme values and the variations of the extreme values over time. Cells that have a small amount of variation over time are placed in a single node of the tree that covers the entire time span. Cells with a larger variation are placed in multiple nodes of the tree multiple times, each for a short time span. The temporal hierarchical index tree requires much less storage space and significantly reduces the amount of I/O required to access the indices from the disk at different time steps.

For time-varying volume data visualization, Ma and Shen [MS00] used octree encoding and difference encoding for spatial and temporal domain compression, respectively. Linsen et al. [LPD*02] presented a four-dimensional multiresolution approach that supports a hierarchy with spatial and temporal scalability. In their scheme, temporal and spatial dimensions are treated equally in a single hierarchical framework. Similar uniform treatments of space and time have been used for representing and visualizing time-varying 3D vector fields [YWM07, MWSJ14]. Different from the normal way of evenly partitioning the space, the *branch-on-need octree* (BONO) [WVG92] partitions the dimension non-uniformly to avoid empty regions. *Multi-dimensional trees* [WVG94] extend the BONO to multi-dimensional data sets, such as spatiotemporal data.

2.2. Hybrid Tree Structures

Applying the 4D tree or 16-tree to organize time-varying volume data treats time merely as another "spatial" dimension. Although simple to implement, the 16-tree could be ineffective for two reasons. First, there could be large discrepancy between the spatial and temporal dimensions, which makes it difficult to subdivide the spatial and temporal dimensions uniformly without over partitioning. Second, in 16-trees, the temporal and spatial domains are tightly coupled in the hierarchical subdivision. This implies that it could be difficult to choose a flexible combination of spatial and temporal data resolutions from the 4D hierarchy.

To remove the dependency between spatial and temporal resolutions, researchers proposed different hybrid tree structures. Shen et al. [SCM99] introduced the *time-space partitioning tree* (TSP tree), a time-supplemented octree for or-



(a) TSP tree

(b) SPT tree

Figure 4: *Illustration of TSP and SPT trees. (a) an example of the TSP tree with four time steps. (b) an example of the SPT tree with six time steps.*

ganizing a time-varying volume data. As sketched in Figure 4 (a), the skeleton of a TSP tree is a standard complete octree, which recursively subdivides the volume spatially until all subvolumes reach a predefined minimum size. In each of the TSP tree nodes, a binary time tree is created, where each node in the binary time tree represents a spatial subvolume with a different time span. Each time tree node stores the mean voxel value as well as spatial and temporal errors of the subvolume in the given time span. These values are used for tree traversal during the volume rendering process. The unique advantage of TSP tree is that it allows users to request spatial and temporal data resolutions *independently* with separate error thresholds. In this way, a flexible browsing of data at arbitrary spatiotemporal resolutions becomes possible. Wang and Shen [WS04] integrated wavelet transform and compression with the TSP tree and presented the *wavelet-based time-space partitioning tree* (WTSP tree) for managing large-scale time-varying data hierarchically.

When partitioning both space and time domains hierarchically to explore spatial and temporal coherence in time-varying data, one could either partition the spatial domain or the temporal domain first. The TSP tree chooses to partition the spatial domain first in order to maintain the visibility order and spatial locality among the subvolumes during the tree traversal. On the contrary, Finkelstein et al. [FJS96] introduced *multiresolution video* by partitioning the temporal domain first, creating the primary binary time tree. For each time tree node, the spatial domain is then partitioned into a quadtree. Du et al. [DCS09] studied the open question of which domain should be partitioned first for a better data reuse rate. They showed both theoretically and experimentally that partitioning the time domain first is better. They further presented the *space-partitioning time tree* (SPT tree) which has the fully balanced binary time tree as the primary structure and a standard complete octree as the secondary structure, as sketched in Figure 4 (b). Compared to the rendering based on the TSP tree, volume rendering based on the SPT tree has the following advantages. First, it leads to a higher level of details since the algorithm is more likely to select smaller but more subvolumes. Second, it has a higher reuse rate because the algorithm favors higher-level time tree nodes. Third, the algorithm runs faster thanks to the higher reuse rate.

2.3. Parallel Image Compositing

Ma et al. [MPH94] introduced the classical *binary swap* algorithm for parallel image compositing. The algorithm works in multiple stages. It makes use of all processors in each stage of composition in a binary tree fashion. For N processors, the number of stages required is $\log N$. At each stage, the image space is divided into two partitions and each processor takes the responsibility for one of the two partitions. A swapping of the partitions between the two processors is needed, thus the name binary swap. Due to the nature of binary compositing, the binary swap algorithm may not work well when the number of processors is not an exact power of 2. Yu et al. [YWM08] presented a generalized image compositing algorithm called the *2-3 swap* which works with an arbitrary number of processors. This flexibility is achieved by allowing each node in the compositing tree to have either two or three children. The resulting *2-3 tree* is a balanced tree, which defines the structure of the compositing tree and determines the grouping of processors during each stage of image compositing.

2.4. Visual Mapping and Interface

Most partition-wise representations of volumetric data record tree structures *internally* in memory for level-of-detail (LOD) querying and rendering. Only a few research works present such a tree *externally* as a visual mapping as well as an interface for LOD selection. Examples of these external tree mappings include the hierarchical navigation interface [WS05] and LOD map [WS06a] presented by Wang and Shen. Both works utilize the treemap representation. The treemap [Shn92] is a space-filling method for visualizing large hierarchical information (in this case, the current LOD, or a cut through the tree hierarchy). It works by recursively subdividing a given display area based on the hierarchical structure, and alternating between vertical and horizontal subdivisions. The information of an individual node is presented through visual attributes, such as color and size, of its bounding rectangle.

The *hierarchical navigation interface* [WS05] works with the WTSP tree representation of a time-varying data set and has two views. The first view is an overview map illustrating the tree hierarchy in a triangle shape, the error distribution among tree nodes, and the current LOD as a cut through the hierarchy. The second view is the treemap showing the detail node information in the current LOD in an uncluttered view which helps users pinpoint the target regions.

The *LOD map* [WS06a] leverages a treemap to guide mul-

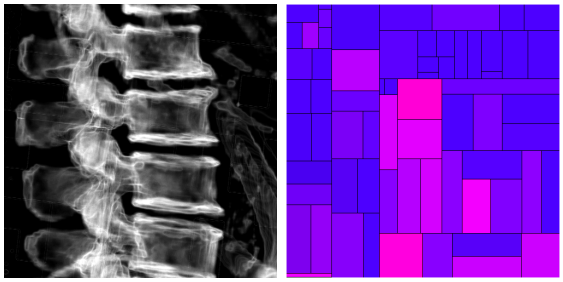


Figure 5: *A zoom to the spine of the visible woman data set and the corresponding LOD map showing the LOD quality. In the LOD map, the contribution of a data block to the final image is mapped to the size of the corresponding rectangle, while the distortion is mapped to the color (magenta to blue for large to small distortion). Images courtesy of Wang and Shen [WS06a] ©2006 IEEE.*

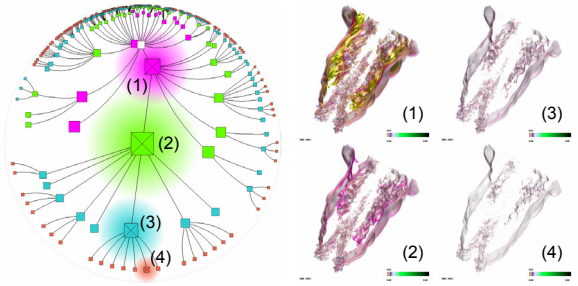


Figure 6: *Level-of-detail exploration of the iTree of the combustion data set. Four nodes at different levels of detail in the tree are selected and their corresponding clusters are highlighted in the volume. Images courtesy of Gu and Wang [GW13] ©2013 IEEE.*

tiresolution volume rendering using the wavelet tree representation. It supports effective LOD volume visualization through a suite of functions including LOD comparison, view comparison, LOD adjustment, and budget control (i.e., giving a LOD of similar quality but with a reduced block budget). An example result of LOD map is shown in Figure 5. The benefits of these explicit visual representations are two folds. First, they provide the *complete* information about the LOD quality of multiresolution data blocks which is impossible to get from the rendering due to the projection of a 3D volume to a 2D image. Second, they also serve as visual interfaces which enable users to interactively adjust the LOD in a guided and controllable fashion.

3. Relationship-wise Representations and Techniques

Relationship is a general term that could refer to any kind of relation among nodes in a tree or graph. Similar to many other applications, the most common way to form a tree-like hierarchy in scientific visualization applications is to either cluster data items from bottom up or partition an object from top down, based on certain similarity or distance measures. For time-varying multivariate data, relationships could mean transition relations, correlation relations, etc. Relationships could also be domain-specific or application-specific. Feature extraction and tracking is an important task for time-varying data visualization and the extracted features are normally connected through a correspondence graph. In addition, different tree and graph structures have also been built for data distribution, task partition, and workload prediction for parallel and out-of-core visualization algorithms.

3.1. Hierarchical Clustering or Partitioning Trees

Linsen et al. [LLRR08] used a cluster tree to represent the level-of-detail of the surfaces constructed from multivariate particle data. They converted the multifield particle volume data to a high-dimensional feature domain, partitioned the domain into cells, and derived the cell density. Then they clustered the cells into to a high density cluster tree where each cluster corresponds to a surface representing the particles within the cluster. Finally, they employed 3D star coordinates to visualize nested density clusters as surfaces.

Patro et al. [PIB*11] designed *MDMap* that uses a state transition graph to explore molecular dynamic simulations. They identified different conformational states by agglomerative clustering to depict the intermediate conformations of the proteins. An inter-cluster distance metric is formulated. During each iteration, the closest clusters are merged until the specified number of clusters remains. In the transition graph, a node represents a state, and an edge between two nodes represents a transition between the two states. The edge thickness indicates the occurrence frequency of the transition. The agglomerative clustering naturally provides a hierarchical structure of clusters. Users can expand a node corresponding to a coarse cluster to reveal refined states.

Ip et al. [IVJ12] utilized a tree-based representation to help users explore a 3D intensity field adaptively. They converted the 3D intensity field into a 2D intensity-gradient histogram and applied the normalized cut algorithm to partition the histogram into segments. By iteratively partitioning the histogram, they created a tree hierarchy which records the level-of-detail partition process. The tree serves as an interface for users to explore the embedded structures. Günther et al. [GRT14] also utilized a tree structure to store streamline clustering information for a 3D vector field. With this tree, users can set opacities for different streamline segments to reduce visual occlusion and clutter and highlight regions of interest. To this end, they first evenly partitioned all streamlines into segments, then used a binary tree to cluster stream segments into a hierarchy. A good streamline visualization result can be achieved by assigning different opacity values to subtrees and optimizing the global opacity.

Gu and Wang [GW13] introduced *iTree* for time-varying data visualization, which integrates data classifying, indexing and compacting into a single framework. They utilized

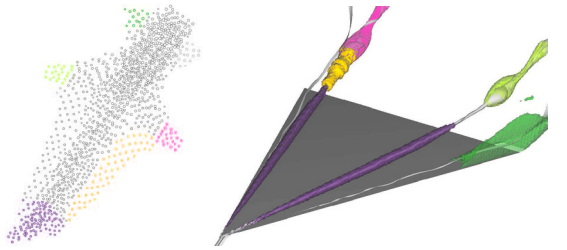
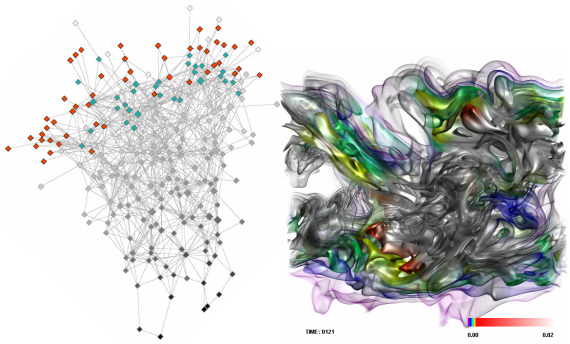


Figure 7: *Analogue brushing with six variables of the multivariate delta wing data set using the attribute cloud. The six variables are the norm of velocity, pressure, density, x-, y-, and z-coordinate of position. Images courtesy of Jänicke et al. [JBS08] ©2008 IEEE.*



(a) (b)

Figure 8: *TransGraph query of the combustion data set. (a) nodes that are only involved in self-transition are in red and the rest of nodes at the same time step are in green. (b) volume highlighting corresponds to the red nodes. Images courtesy of Gu and Wang [GW11] ©2011 IEEE.*

the symbolic aggregate approximation (SAX) for data compacting and indexable SAX (iSAX) for indexing. Although iSAX is already stored internally as a tree structure, it is difficult to visualize and interact due to its wide branching and high depth. To convert iSAX to a user-friendly user interface, they performed level promoting, sibling grouping, and sibling reordering to reduce the width and height of the iSAX tree. The resulting visual interface, iTree, uses a hyperbolic layout for focus+context visualization and provides direct interactions for users to query, search and track the time-varying volumetric data. An example of level-of-detail exploration of the iTree is shown in Figure 6.

3.2. Relationship Graph Encoding and Visualization

Graphs can be used to encode the relationships between variables. In such a graph, a node simply represents a variable and an edge represents the relationships between variables. Therefore, these graphs are typically fully connected. For instance, Qu et al. [QCX*07] built an undirected weighted graph to visualize the correlation between variables. Biswas et al. [BDSW13] leveraged the same kind of graph to encode the mutual information between variables. These two graphs are drawn using a force-directed layout. Wang et al. [WYG*11] studied the information transfer between variables. Since information flow carries directional information, they utilized a directed weighted circular graph to show the information transfer between variables. Oeltze et al. [OFH*11] applied graph visualization techniques for visual analysis of high-dimensional, multi-parameter fluorescence microscopy data. They drew a circular graph where each node represents an affinity reagent while each edge represents two co-occurring affinity reagent bindings.

Besides using a node to represent a variable in the graph, other solutions use a node to represent spatiotemporal data blocks, derived fields, or other quantities. This leads to more refined results of graph relation since a node now represents a group of voxels rather than an entire variable or time step. The graph could be *hierarchical* so that the relationships can be shown in a coarse-to-fine manner. Furthermore, the graph could also be *compound* with different kinds of nodes and edges encoding more complex data relationships.

For multivariate data visualization, Sauber et al. [STS06] developed *Multifield-Graphs* for a complete visualization of scalar fields and their correlations so that features associated with multiple fields can be discovered. In Multifield-Graphs, a node denotes the correlation field among variables. An edge connects a parent and its immediate child if and only if the parent node has one new variable in the correlation field than that of the child node. The graph allows both correlation overview and focused display of certain nodes. A limitation of Multifield-Graph is that the number of nodes in the graph increases exponentially with the number of dimensions. Jänicke et al. [JBS08] transformed multivariate data from their high-dimensional attribute space to a 2D *attribute cloud* by constructing the minimum spanning tree (MST) from sample points in the original high-dimensional space and utilizing a graph layout algorithm to minimize edge crossings in 2D. Through brushing and linking the attribute cloud with the data, users can conveniently connect the abstract attribute space with the original data space. An example of brushing the attribute cloud and the corresponding volume highlighting is shown in Figure 7. Liu and Shen [LS16] used multiple coordinated views to reveal the relationships among scalar fields in multivariate data sets. They derived multi-scalar informativeness and uniqueness from the influence-passivity model, which has been used to analyze social networks. A series of parallel coordinates plots are provided to discover the relationships of values, informativeness, uniqueness and probabilities of each scalar variable. The use of the parallel coordinates plot view and the *probability association graph* view allows users to explore the scalars of interest.

For time-varying data visualization, Jänicke and Scheuer-

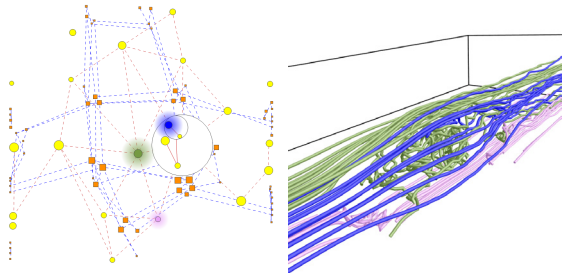
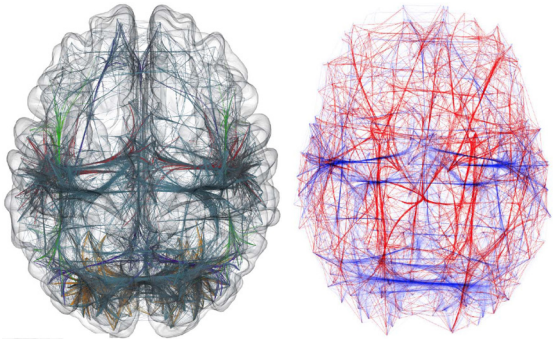


Figure 9: *Selecting three L-nodes in the car flow data set using FlowGraph to capture the main flow structure passing through the car. Images courtesy of Ma et al. [MWS13] ©2013 IEEE.*



(a) (b)

Figure 10: *Brain connectivity graphs. (a) the whole-brain group-level connectivity with the bundled graph. (b) the comparison between functional (blue) and anatomical (red) connectivity. Images courtesy of Böttger et al. [BSL*14] ©2014 IEEE.*

mann [JS09] introduced a directed graph representation named the ε-*machine*. Leveraging a light-cone structure, they defined a causal state as all the required information in a given position to predict its future. Transition probabilities among states can be derived based on transition frequencies. In the ε-machine, a node represents a causal state and a directed edge connecting two nodes represents their transition probability. Gu and Wang [GW11] designed *TransGraph* to visualize the transition relationships in a time-varying volumetric data set. In TransGraph, a node denotes a state, i.e., a spatiotemporal region, and a directed edge between two nodes indicates their transition probability. TransGraph organizes the states hierarchically and the resulting hierarchical graph enables relationship overview and detail exploration of a 4D time-varying volume data set in a single 2D graph view. Figure 8 gives such an example. Gu et al. [GWP*16] further applied a mining approach to obtain more insights from TransGraph and the underlying time-varying volumetric data set. First, they applied a graph simplification scheme to reduce visual clutter by replacing graph features (fans, connectors and cliques) with symbols. Second, they detected communities to recognize node groups with more frequent connections. Unlike a clique, a community of nodes does not require that every pair of nodes is connected by an edge. Instead, it places a looser restriction, so that most of the nodes in a community are adjacent to each other, but few connections exist between nodes from different communities. Third, they provided visual recommendations to highlight similar nodes or communities when a node or a community is selected by users.

For flow visualization, Xu and Shen [XS10] introduced the *Flow Web*, which provides an overview of the 3D flow field and help users locate regions of interest. To construct the Flow Web, they first partitioned the volume into regions using an octree. Then they performed random seeding in each region to trace streamlines. In the Flow Web, a node represents a region and a weighted edge between two regions indicates the number of streamlines going through them. To allow for easy understanding of the underlying flow structures, they created hierarchical Flow Webs by splitting or merging nodes and edges. Rather than only considering streamline clusters or spatial regions as nodes, *FlowGraph* developed by Ma et al. [MWS13] integrates both streamline clusters or spatial regions as nodes and thus presents a more complete picture. As a *compound* graph, FlowGraph consists of two kinds of nodes (L-nodes and R-nodes) and three kinds of edges (L-L edges, R-R edges, and L-R edges) where 'L' denotes streamline and 'R' denotes spatial region. To construct the hierarchical graph, they clustered streamlines hierarchically from bottom up and partitioned spatial regions recursively from top down. A set of functions is designed to enable hierarchical exploration of streamline clusters, spatial regions and their interconnection, detail comparison among streamline clusters, and close examination of spatial regions. Figure 9 shows an example of exploring interesting flow patterns using FlowGraph. Ma et al. [MWSJ14] further extended FlowGraph to analyze and visualize the relationships between pathlines and spatiotemporal regions in 3D unsteady flow fields.

3.3. Relationship Graph for Biomedical Data

Studying the anatomical and functional connectivity within a brain is an important topic in neuroscience. The derived connectivity graphs are undirected and unweighted. Beyer et al. [BAAK*13] presented *ConnectomeExplorer*, where a *connectivity graph* was used to visualize the connections among axons or dendrites. In the 2D connectivity graph view, nodes correspond to axons or dendrites, and edges represent synapses between them. Böttger et al. [BSL*14] used a *binary graph* in conjunction with an edge bundling technique to show the group-level connectivity information within a whole brain. In the binary graph, nodes represent cortical parcels and edges indicate strongly connected nodes. In Figure 10, two binary bundled graphs show the whole-brain connectivity information. Unlike previous examples of

iTree, attribute cloud, TransGraph and FlowGraph shown in Figures 6 to 9, the connectivity graphs are drawn in the 3D spatial domain so the contextual brain information is retained. The edge bundling technique is applied to produce a less cluttered view.

Kuß et al. [KPM*08] proposed an *ontology instance graph* for interactive exploration of hierarchical neuroanatomical structures. The neuroanatomical atlases are organized as a hierarchical graph. Users can interact with this graph to query the corresponding 3D geometries of an atlas. Instead of displaying only the queried geometries, they rendered all geometries with different transparencies. The transparencies of different geometries represent how important the structures are, based on their relationships to the queried structure. The importance value of a geometry is determined by the depth of its corresponding node in the traversal tree rooted at the queried node. This provides the queried result in a focus+context manner, where the queried geometries are the focus and other geometries provide the context.

Sorger et al. [SBS*13] designed *neuroMap*, an interactive graph in the form of a circuit-style wiring diagram that renders the brain of a fruit fly and its interconnections. They presented an anatomical layout to emulate an abstract view of the fruit fly's brain. Three different types of nodes, namely, cell body nodes, arborization nodes, and neuropil nodes, are designed to represent different structures. To demonstrate the connection among different elements, they used projection edges connecting cell body nodes and arborization nodes as well as overlap edges among arborization nodes.

Nowke et al. [NSvA*13] introduced *VisNEST* for users to inspect connections among brain regions and neural activities. VisNEST consists of three views: a connectivity view to visualize how each brain area is connected to others; a flux view to show the time-dependent neural activities between brain areas, where the sizes of nodes and edges are used to encoded the self-flux and flux between areas, respectively; and a population view to display activities and connectivities within each area.

3.4. Relationship Graph for Movement Data

Spatiotemporal movement data (e.g., traffic or trajectory data) are often encoded by graphs as well. For example, Guo [Guo09] visualized the population migration through flows using a *location-to-location graph*. Each node in the graph represents a location, and an edge indicates the flow between two locations. A spatially constrained graph partitioning method is applied to construct a hierarchy of spatial regions, and the flows are then aggregated at the regional level. Multiple variables are associated with the flow, such as the number of migrants for different age and income levels. Multivariate clustering is performed for flow clustering. A separate parallel coordinates visualization is provided, so that users can brush this view to filter the flows.

von Landesberger et al. [vLBR*16] proposed *MobilityGraph* to visualize mass mobility dynamics. The spatial *situations*, i.e., the presence and flows of people at certain time intervals, are represented by graphs. Similar to the work of Guo [Guo09], they first simplified the spatial situations by aggregating neighboring places. Then, they applied a density-based clustering to group the places into regions based on spatial closeness and flow magnitude. The clustering parameters can be adjusted interactively to obtain graphs with various levels of detail. To obtain the common patterns of situations, they performed a temporal clustering on the simplified situations based on their similarities. A feature vector consisting of a set of flow magnitudes is generated to characterize each situation. The similarity between two situations is measured by the difference between the corresponding feature vectors. For temporal clusters, they employed a calendar view to show the relationships among situations along with one representative situation in each cluster. Figure 11 shows an example of MobilityGraph.

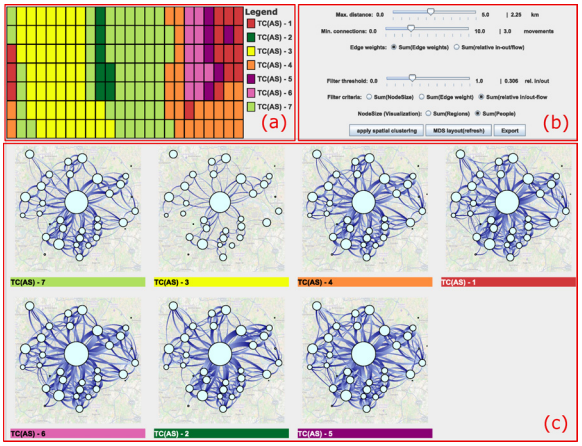


Figure 11: *Clusters of MobilityGraph showing the flows of people. (a) the calendar view of temporal clusters. (b) the interface for obtaining graphs with various levels of detail. (c) the cluster overview showing the representative graph for each cluster. Image courtesy of von Landesberger et al. [vLBR*16] ©2016 IEEE.*

Huang et al. [HZY*16] designed *TrajGraph* to study urban network centralities using taxi trajectory data. Each node in the graph represents a road segment, and an edge exists between two nodes if the corresponding road segments are connected. They applied a graph partitioning method to generate the region-level graph from the original street-level graph. Unlike other approaches, a region in this graph indicates a connected set of streets, which is not necessarily a spatial cell. Different partitions can be obtained based on different vertex weight assignments, including the length of streets, number of taxis, average travel time, and average speed. They measured the centrality or betweenness to characterize the time-varying importance of regions. Three

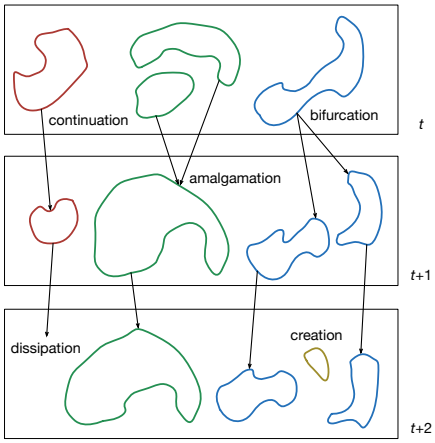


Figure 12: *Different feature tracking events: continuation, bifurcation and amalgamation for matched features, and creation and dissipation for unmatched ones.*

different views are provided to visualize the centralities: a node-link graph view, a map view, and a temporal information view.

3.5. Feature Correspondence and Tracking

An important task for time-varying data analysis and visualization is feature extraction and tracking. An early work by Samtaney et al. [SSZC94] extracts features (i.e., regions that fulfill certain criteria) and makes correspondence in neighboring time steps. The features are matched through their attributes such as the centroid, volume and mass. For the neighboring two time steps, they compared every feature pair for a possible match. If there is a match, then the two features are corresponded based on the relation of continuation. After removing all the continuous features, they compared the combination of multiple features to test for bifurcation or amalgamation cases. The remaining unmatched features are considered to be dissipation or creation. Figure 12 illustrates these five different events commonly identified in feature tracking. The evolution of features over time can be mapped to a directed acyclic graph (DAG) which records feature correspondence.

A limitation of the work by Samtaney et al. [SSZC94] is that since all feature combinations need to be considered for the decision of bifurcation and amalgamation, the computation cost remains high. To reduce the cost of comparing all the feature combinations, Silver and Wang [SW97] utilized octrees to store all the features, which allows fast identification of feature overlapping in space. Spatial overlap is used to determine feature matching. For a selected feature in one time step, they found out those candidate features in the next time step which overlap it. Then they applied a normalized volume difference test to choose the best matching from the candidates.

Commonly used approaches for feature extraction include region growing with a certain threshold or extracting the isosurfaces with a given isovalue. Woodring and Shen [WS09] found the features at each time step using a k-means clustering algorithm. For each time step, they first generated k clusters based on the input time-activity curves within a time interval. Then they created a directed graph that connects the clusters over time. To find the links between features, they estimated the probability of transferring one feature to another by computing the distance of their time histograms.

The recent work of Ozer et al. [OSBM14] advances the state-of-the-art feature tracking by leveraging *Petri Nets* to model and detect activities in scientific visualization. Petri Nets are graph-based techniques that model and visualize various types of behaviors. The Petri Nets model includes places (types of possible object states), transitions (conditions or actions between places), and directed arcs (connections between places and transitions). Based on the results of feature extraction and tracking, they utilized Petri Nets to model the activity of interest and ran the algorithm for hypothesis validation.

For biological data, Heiland et al. [HSS*12] presented *CompuCell3D*, a system that identifies individual cells and visualizes their connections using a connectivity graph. They provided multiple example models to perform cell sorting simulations. Cell sorting is a biological procedure where cells adhere to one another to varying degrees depending on their cell types, leading to a certain pattern. During the cell sorting simulation, the positions of cells may change to reflect the adhesion among the cells. CompuCell3D creates a graph representation for any time step of the simulation and provides two types of layouts to visualize the graph. Using the cell's centers of mass as the positions of nodes, one layout represents the spatial distribution of different types of cells. Using the other spring layout, the spatial positions of nodes emphasize the neighboring connectivities of cells.

Parulek et al. [PTRV12] proposed to extract cavities in molecular simulations and represent them and their connections using a *cavity graph*. Unlike most of the graph-based techniques which generate a 2D layout of the graph, the cavity graph is embedded in the molecular, which provides the contextual information. For each connected component in the graph, they computed three graph component measures: the longest path between any two nodes, the average of the shortest paths between the nodes, and the average of the degrees of all the nodes. For effective exploration, two accompanying scatterplots of the graph components are provided to support brushing and linking. One scatterplot depicts a selected graph component measure against time, and the other depicts two selected graph component measures against each other over all time steps of the molecular simulation. Figure 13 demonstrates an example of using these scatterplots to identify a graph component.

Lindow et al. [LBBH12] presented a dynamic molecular

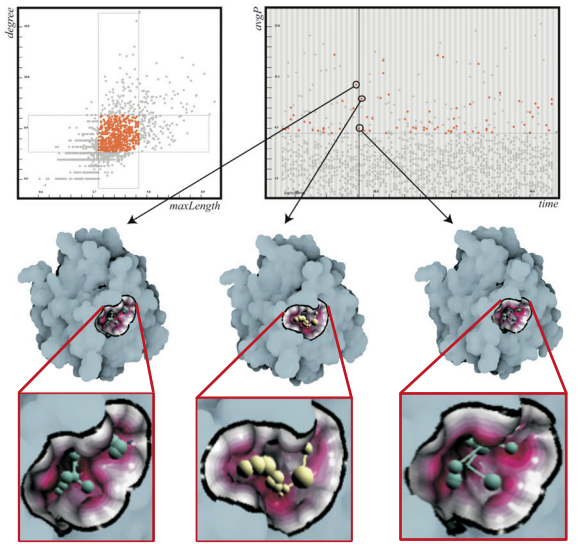


Figure 13: *The cavity graph that presents the cavities and their connections in the context of the protein. A graph component group of medium maxLength and medium degree values is selected from the top-left scatterplot. The cavities are then identified from the top-right scatterplot in three consecutive time steps. Image courtesy of Parulek et al. [PTRV12] ©2012 IEEE.*

visualization in the same context as the work by Parulek et al. [PTRV12]. Similarly, static molecular path components are identified at each time step and represented by graph components. But instead of using a scatterplot over time to brush and link certain static paths, they visualized dynamic paths consisting of a large number of components that are accumulated over time. In this way, splits and merges are shown more clearly since the static paths from different time steps are displayed together. The dynamic paths color the components by time, so that the development of paths can be tracked by following color transition.

For traffic data, Wang et al. [WLY*13] analyzed traffic jams and their propagation. They first detected traffic jam events in the trajectory data and then constructed a *propagation graph*, where each node in the graph represents a traffic jam event. There is a directed link between two events pointing from the earlier event to the later one, if they are temporally overlapping and spatially connected. Other than using the spatial view to visualize the traffic densities, they provided multiple views to explore the relationships among the propagation graphs: the graph list view for showing the traffic jam propagation graphs one-by-one in some sorted manner; the multi-faceted filter view for filtering the propagation graphs by time and size; and the graph projection view for demonstrating the relationships among graph clusters, where graphs in the same cluster share a similar topology.

3.6. Parallel Processing and Visualization

Trees and graphs have also been designed to facilitate data distribution and task partitioning for high-performance parallel visualization applications. For parallel visualization of 3D unsteady flow fields, Yu et al. [YWM07] constructed a hierarchical 4D representation of the spatiotemporal data so that pathline tracing can be treated as streamline tracing in 4D. They built an adaptive grid for hierarchical vector field clustering. The clustering merges neighboring grid cells of similar patterns and yields a binary cluster tree. At runtime, they traversed the binary cluster tree to obtain the seeds from the clustering of the adaptive grid representation. The streamlines are then traced in parallel in the original 4D vector data to generate pathline results.

For parallel feature extraction and tracking, Wang et al. [WYM13] presented a solution that extracts local partial features at individual processors and then integrates partial features based on the connectivity information to extract and track features in parallel. Specifically, for each block, they created a *local connectivity tree* with six children corresponding to its six spatial neighboring blocks. The processor associated with its assigned block communicates with its neighbors to exchange the centroids and the minimal and maximal region boundaries of the features. Such information is used to match the features with the neighbors in the current time step. Then, based on the local connectivity trees, they created a *global connectivity graph* to store the information of all the features in that time step. This process repeats for the subsequent time steps. Since features normally change little in neighboring time steps, feature movements could be identified through their movements between processors in the global connectivity graph.

For parallel streamline generation, Nouanesengsy et al. [NLS11] designed the *flow graph*, a directed weighted graph to estimate the workload for each data block. They followed the work of Flow Web [XS10] for blockwise data partitioning and random seed placement. While the Flow Web counts the number of particles traveling through the boundaries of blocks, the flow graph calculates the corresponding probabilities and treats them as edge weights. The resulting graph guides the estimation of particle movements. The flow graph only stores the information of traveling through blocks. No information of traveling within each block, such as the lengths of streamlines in the block, is kept. During each round of tracing, the algorithm also considers the average number of tracing steps per particle for more accurate workload estimation and balancing.

3.7. Out-of-Core Field Line Tracing

For out-of-core streamline tracing, Chen et al. [CXLS12] developed the *access dependency graph* (ADG) to guide data reorganization with the goal of reducing the I/O miss ratio. In an ADG, a node represents a spatial data block, and an edge connects two blocks if there are particles traveling between them. Besides spatially adjacent blocks, the ADG also

considers non-adjacent blocks if they are on the same flow path of a particle. A linear ordering of data block files is required for I/O performance optimization so that when a file is needed and uploaded to main memory for streamline tracing, its neighboring files would also be fetched to reduce the I/O cost. Therefore, an appropriate reordering of files could reduce the miss ratio. Unlike the flow graph [NLS11] which only uses one-hop prediction, the ADG considers N-hop in the prediction and the final ADG is the union of graphs G_1 to G_N. They formulated the optimal layout problem as a graph linear arrangement problem. The miss ratio is minimized when the sum of distances in the file for data blocks along the same flow paths is minimal.

Chen et al. [CNLS12] further extended the ADG for out-of-core pathline tracing. Each node in the ADG is called a time block which consists of the blocks at the same location in a time period. A directed edge connects two time blocks if there exists pathlines passing through them. Each edge is assigned a weight equal to the probability of seeds moving from one block to the other. The finite-time Lyapunov exponent (FTLE) helps users study the existence of the Lagrangian coherence structures (LCS) by quantifying the separation of flows. However, in order to produce a high-resolution FTLE field, we need to trace a pathline from every grid point and at every time step, which is very time-consuming. Chen and Shen [CS13] presented an out-of-core solution for efficient FTLE and pathline computation through reducing I/O cost and maximizing data reusing. They first utilized the ADG to estimate the tracing probabilities between neighboring blocks and then applied the discrete-time Markov chains on the ADG to predict the probabilities between all the blocks. Finally, they performed seed scheduling to select and order the seeds for tracing at each round to maximize the usage of data blocks.

4. Structure-wise Representations and Techniques

Scientific visualization often deals with discrete data sets with values sampled at grid points in a volume. Extracting the structural information from discrete data is critically important toward efficient understanding of the underlying data. For instance, there is a large body of work on the model-based visualization of vascular structures in medical visualization [PO08]. Vasculature is represented and analyzed by means of a *branching graph* [HPSP01].

Topology refers to a structure imposed upon a data set that essentially characterizes properties of space such as convergence, connectedness and continuity. Topology-based methods provide a concise and rigorous description of the overall structure and lead to mathematically sound tools for processing, exploration and visualization of scientific data sets. Topological methods based on *Morse theory* include several abstract representations for studying scalar fields: contour trees, Reeb graphs, and Morse-Smale complexes. Morse theory shows that topological changes in scalar field data defined on manifolds occur at distinct isolated points, i.e., *critical points*. The *Reeb graph* shows the evolution of individual contours using these critical points and their relationships. The *contour tree* is a representation that records changes in the topology of the *level sets* (i.e., isocontours) of a scalar field. It is a special case of the Reeb graph where the graph forms a tree structure for simply connected domains. The *Morse-Smale complex* is a partition of the domain into cells according to the gradient of the scalar field. In this survey, we restrict our attention to contour trees since they are most widely used in scientific visualization applications.

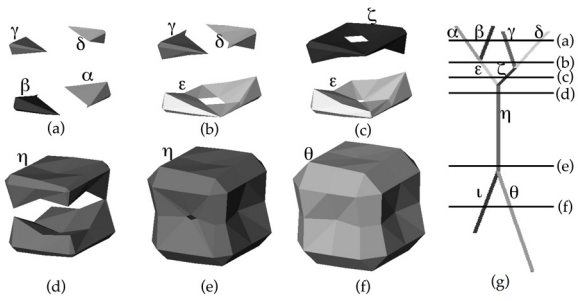


Figure 14: *The contour tree with horizontal lines corresponding to level sets (a) to (f) produced at six distinct points during sweep from high to low isovalue. Note that in (f), contour* η *is separated into invisible inner* (ι) *contour and visible outer* (θ) *contour. Image courtesy of Carr [Car04].*

The contour tree captures the topological evolution of an isosurface as the isovalue changes. Since the domain is simply connected, it does not contain loops. As shown in Figure 14, in a contour tree, nodes represent critical points where the number of components varies. Critical points have three classes: *minima*, *maxima* and *saddles*. Leaf nodes are *extrema* (i.e., minima and maxima) representing the creation or deletion of components. Interior nodes are saddles representing the joining or splitting of two or more components. An arc represents contours between critical points, i.e., contours which do not change topology as the isovalue varies between critical values. The value difference of two nodes along an arc is called *persistence* which measures the importance of the corresponding topological features.

The contour tree can be created by first scanning the data set twice: one pass to create the *join tree* and the other pass to create the *split tree*, and then merging the join and split trees together [CSA03]. A *merge tree* (i.e., a join or split tree) is a substructure of contour trees that tracks either merges or splits of the isocontours.

4.1. Contour Trees and Reeb Graphs

As an abstraction of a scalar field that encodes the nesting relationships of isosurfaces, the contour tree has been used to accelerate isosurface extraction, to identify salient isovalues, and to guide exploratory visualization.

Bajaj et al. [BPS97] introduced the use of *contour spec-*

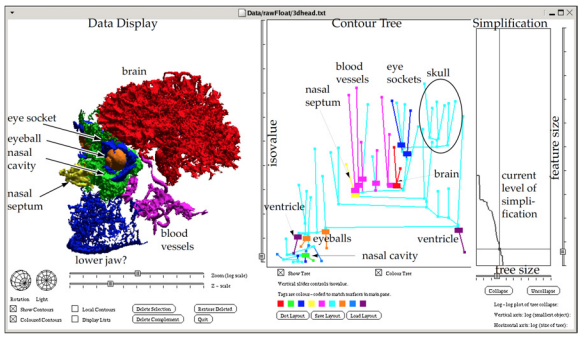


Figure 15: *A flexible isosurface chosen from a simplified contour tree. The rightmost pane controls the amount of simplification of the contour tree shown immediately to its left. The interface supports interactive exploration of structures that are hidden in the conventional view. Image courtesy of Carr et al. [CSvdP04] ©2004 IEEE.*

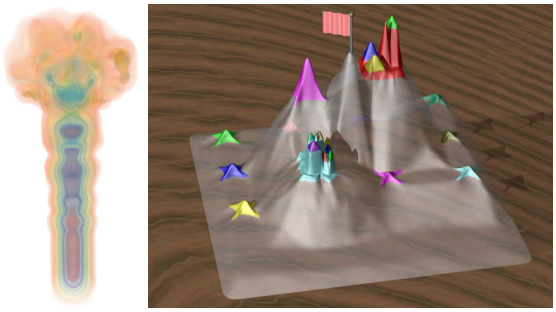


Figure 16: *The topological landscape of the fuel data set. Peaks correspond to maxima and valleys correspond to minima. The main mountain is presented in a semi-transparent gray to facilitate the visualization of deep valleys and its center is marked by a flag. Images courtesy of Weber et al. [WBP07] ©2007 IEEE.*

trum for exploring complex scalar fields. The contour spectrum provides an interface which plots properties such as isosurface area and enclosed volume, along with the contour tree, for users to explore interesting isosurfaces. Carr et al. [CSvdP04] designed an effective contour tree simplification algorithm that computes local geometric measures for individual contours and uses them to suppress minor topological features in the data. The contour tree is simplified with two operations: leaf pruning and vertex reduction. As shown in Figure 15, they presented a flexible isosurface interface to explore individual contours interactively. Carr et al. [CSvdP10] presented an interface that allows users to select an individual contour for manipulation. The contour can be selected from the contour tree or the isosurface display. A set of functions such as contour removal, contour evolution, and contour tracking is provided. These functions are realized based on the attaching of isosurface seeds (i.e., path seeds) to each edge of the contour tree so that individual contours can be extracted on demand.

Galvani et al. [GKHS98] presented a solution for volume animation using the *skeleton tree*. They extracted skeletal voxels and connected them to form a graph based on spatial closeness and value similarity. The minimum spanning tree is extracted from the graph to yield the skeleton tree, which suggests the shape of the object. Using line segments to connect skeletal voxels, one can generate a skeletal structure that is amenable for intuitive motion and deformation.

Takeshima et al. [TTFN05] utilized the *volume skeleton tree* to define topological attributes as additional input for specifying multidimensional transfer functions. Their settings of transfer functions are based on fixed topological indices, such as depth of topological nesting. Weber et al. [WDC*07] generalized the work of Takeshima et al. [TTFN05] by allowing the user to assign independent transfer functions to topologically distinct features. These features do not need to share the same topological indices. They segmented a volume into regions where each region corresponds to a branch of a hierarchical contour tree decomposition, and applied a separate transfer function to each region. Zhou and Takatsuka [ZT09] used the contour tree as a visual index to volume segments and utilized topological attributes for automatic transfer function specification. They employed the opacity residue flow model to contour tree branches and provided user interfaces for generating the transfer function.

Pascucci et al. [PSBM07] proposed an approach to compute Reeb graphs for simplicial complexes of any dimension. In practice, this kind of approach is usually applied to 3D meshes for geometry modeling. In a Reeb graph, nodes represent the critical points, i.e., maxima, minima and saddles. Edges indicate the connection of these critical points. The Reeb graph can be considered as a representation where the contours of a function are contracted into points. They described an online noise removal method to deal with the often noisy real-world data. This procedure removes the extremum-saddle pairs based on the persistence values of a series of arcs connecting them. They further simplified the Reeb graph to provide a coarse-to-fine hierarchy.

Weber et al. [WBP07] introduced the *topological landscape*, a 2D terrain with the same topology as a given high-dimensional data set for easy understanding the topological structure. They first extracted the contour tree from a scalar function, then constructed its branch decomposition [PCMS05], and finally recursively created a terrain with the same topological structure as the original data. The topological landscape also preserves the persistence and volume of each feature. It provides a powerful interface to complement existing contour tree based techniques. Figure 16 shows an example of topological landscape. Based on the topological landscape metaphor, Harvey and Wang [HW10] presented *topological landscape ensembles*, a collection of 2D terrain

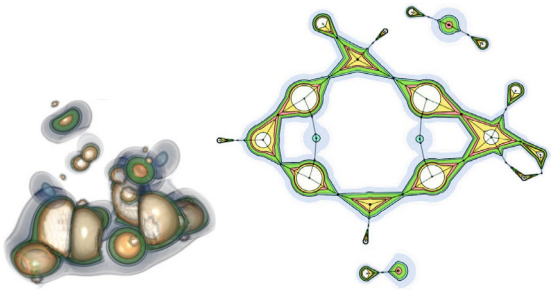


Figure 17: *The topological spine of the neghip data set. The spine preserves the symmetry and cyclic structure of the molecule. Images courtesy of Correa et al. [CLB11] ©2011 IEEE.*

models which preserves the contour tree of the input high-dimensional scalar field. They leveraged the treemap layout [Shn92] to construct the terrain layout and provided a simple interface for users to explore the terrain model.

Doraiswamy et al. [DFD*14] formalized the relationship of regions in traffic data using topological persistence over a scalar function. The scalar function is usually but not constrained to the densities of traffic over a spatial domain. In their graph, each node is a critical point of the scalar function. Considering a sweep of the function in the decreasing order of function value, both the nodes and their connectivities vary according to the sweep. They captured the sub-level and super-level changes of connectivity using the split tree and join tree, respectively. In addition, the minima and maxima of the function are used to represent the events in the data. Each minimum or maximum corresponds to a subgraph, and their relationships are measured by subgraph similarities. They grouped the events based on the similarities for querying using the event group index.

4.2. Neighborhood Graphs

Neighborhood graphs connect points based on their proximity information to create a geometric structure of the data. Well-known examples of neighborhood graphs include the *k-NN graph*, *relative neighborhood graph* [JT92], and *Delaunay triangulation*.

Bremer et al. [BPH05] presented the *cancellation tree* which simplifies a Morse-Smale complex of a function's topology by successively canceling pairs of critical points. Through effective encoding the cancellations, they produced an adaptive topology-based multiresolution representation of the function. This leads to a concise subset of the Morse-Smale complex that connects maxima or minima in a tree. Gyulassy et al. [GNP*06] proposed a topological approach to simplify Morse-Smale complexes derived from 3D scalar functions. This approach is based on the cancellation as well. They descried two types of cancellation to remove critical point pairs between two saddles, or between a saddle and an extremum. They ordered the pairs of critical points by the persistence values, so that those with smaller persistence values will be canceled first. The persistence value of a pair of critical points is defined as the absolute difference of the scalar values of the corresponding critical points.

Correa et al. [CLB11] introduced *topological spines*, a visual representation that preserves both topological and structural properties of a scalar field. Topological spines are based on *extremum graphs* that describe the connectivity of the spine by connecting critical points along steepest ascending (or descending) lines. Topological spines are enhanced visual representations of extremum graphs with geometric and contour nesting information. They not only abstract the shape and structure of complex 3D scalar data, but also present 2D maps to facilitate feature selection and tracking. As shown in Figure 17, compared with other representations such as the contour tree and Morse-Smale complex, topological spines are better suited for spatial reasoning due to the preservation of the topology and locality of extrema and the nesting structure of the surrounding contours.

Correa and Lindström [CL11] studied the *empty region graphs* (ERGs) to improve the robustness of neighborhood-based analysis methods. In an ERG, two points are connected if a canonical region around them does not contain any other point. Examples of ERGs are the *nearest neighbor graph*, *relative neighborhood graph*, *Gabriel graph*, *diamond graph*, and β-*skeletons* [CL11]. They generalized ERGs to natural ERGs (which guarantee certain neighborhood and space partitioning properties), relaxed ERGs (a variation which is less sensitive to the sparsity and distribution of samples), and stochastic ERGs (a generalization that introduces the likelihood of samples being neighbors).

4.3. Topological Feature Tracking

Soho and Bajaj [SB06] applied the contour tree for time-varying contour tracking. They first constructed the contour tree at every time step and computed the correspondence information among contour trees across time steps given an isovalue. The *topology change graph* is constructed by creating a node for every intersection point and connecting each pair of intersection points where their representative contours correspond to each other. This graph allows detection of significant topological and geometric changes in time-varying isosurfaces. It also serves as an interface for users to segment, track and visualize the evolution of selected contour components over time.

Bremer et al. [BWT*11] constructed a *hierarchical merge tree* for each time step with augmented attributes using a streaming algorithm and then created tracking graphs to capture the temporal evolution of features. As shown in Figure 18, a linked-view interface is provided to explore the time evolution of the graph along with segmented data features. Widanagamaachchi et al. [WCB12] built a compact *meta-graph* to store the sequence of feature families and to encode all possible tracking graphs and relevant attributes. At

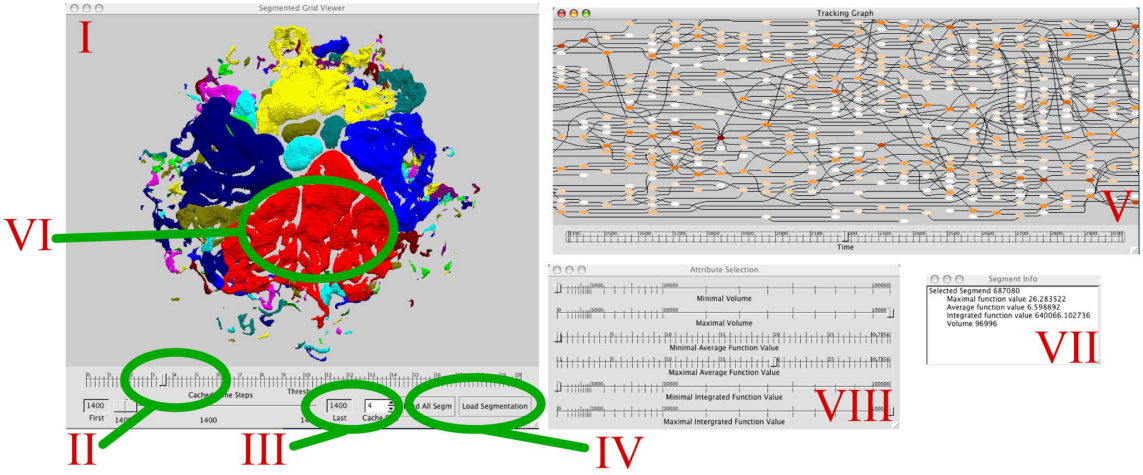


Figure 18: *(I) 3D display of the segmentation of burning cells from the turbulent combustion data set with various interface controls (II to IV). (V) interactive display of the tracking graph. A selected node is highlighted in (VI) and its attribute information is displayed in (VII). (VIII) is for subsection of segments based on attribute values. Image courtesy of Bremer et al. [BWT*11] ©2011 IEEE.*

run time, they interactively extracted, filtered and simplified a *dynamic tracking graph* from the meta-graph to explore the temporal evolution of features. The tracking graph also provides a user interface that is integrated into the 3D view of current feature sets for brushing and linking.

For multivariate feature tracking, Bennett et al. [BKL*11] encoded the set of all possible flow features by precomputing merge trees augmented with attributes. The resulting multiresolution feature hierarchy enables flexible and interactive feature analysis, allowing the exploration of the entire feature family without accessing the original data. McLendon et al. [MBB*12] developed an *attributed relational graph* (ARG) to capture the structural relationships defined by multiple variables. In the ARG, nodes represent features defined by an arbitrary number of variables, and edges encode the relationship between these features both within and across time steps. They presented a query-based search interface for users to identify events of interest characterized by subgraph-isomorphism search heuristics.

4.4. Symmetry Detection and Structure Comparison

Thomas and Natarajan [TN11, TN13] studied the symmetry in scalar field topology. *Symmetry* in scalar fields refers to different parts of the data that are invariant in the scalar field distribution. In [TN11], they analyzed the topology of level sets in a scalar field using the contour tree to detect symmetric patterns. With a similarity measure for comparing subtrees, they identified subtrees in the contour tree that are similar for grouping and extracted regions corresponding to a common group as symmetric components. In [TN13], they presented the *augmented extremum graph* and designed a symmetry detection algorithm based on robust distance estimation. The augmented extremum graph is based on the extremum graph [CLB11]. It captures both topological and geometric information of the scalar field and enables computationally efficient detection of symmetry. This method improves the previous work [TN11] in that it is aware of the underlying geometry and can detect global symmetry even in the presence of significant noise.

Schneider et al. [SWC*08] presented a solution for interactive comparison of scalar fields using isosurfaces. They defined features as the largest contour segmentation after topological simplification and compared features based on their spatial overlap. They designed the *similarity browser* to show the similarities between features and used the contour trees for navigation. They also combined the contour tree with the λ_2 algorithm to detect and compare features in fluid-flow related scalar fields. Saikia et al. [SSW14] studied the topological structures defined by their sub-level or super-level sets for identifying repeating structures. They introduced the *extended branch decomposition graph* (eBDG) which represents a forest of *branch decomposition trees* (BDTs) [PCMS05] where each of the BDTs is computed from a subtree of the merge tree. The eBDG is leveraged to finding similar structures in the same data set, detecting periodic patterns in different time steps, and comparing the topology in different data sets.

Chan et al. [CQC*08] presented a relation-aware pipeline for volume exploration. They employed region connection calculus (RCC) to define relations between structures in a volume and represented these relations as a *relation graph* for understanding and navigation. In their relation graph, nodes represent segments (i.e., homogenous regions) and links represent the spatial relations between segments. Un-

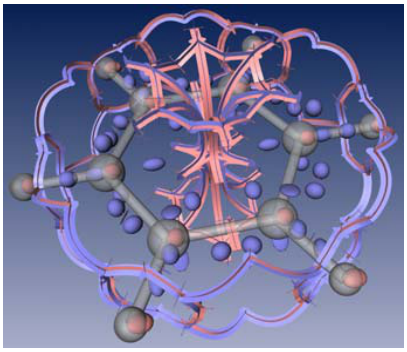


Figure 19: *Visualization of the topological skeleton along with saddle connectors. A double flow ribbon approach is applied to visualize the orientation of the separation surfaces in the neighborhood of the saddle connector. Image courtesy of Theisel et al. [TWHS03] ©2003 IEEE.*

like the contour tree which focuses on the nesting relations of isosurfaces, the relation graph focuses on structure segments and considers a more complete set of spatial relations (separate, touch, overlap and enclose).

4.5. Vector Field Topology

Topological analysis of vector fields is based on the critical points and their connections through integration of special streamlines called *separatrices* [HH89]. The resulting topological skeletons enable users to visually comprehend the structure of the vector field by partitioning the domain into subregions of uniform flow behaviors.

Visualization of the topological skeleton of a 3D vector field requires simultaneous display of a large number of streamsurfaces, which easily leads to visual clutter. To address this issue, Theisel et al. [TWHS03] presented *saddle connectors* that create sparse visual representations by representing separation surfaces as a finite number of streamlines. These streamlines are the intersection curves of separation surfaces, and start and end in saddle points. Figure 19 shows an example of the topological skeleton with saddle connectors.

Based on Morse decompositions, Chen et al. [CML*07] presented the *Morse connection graphs* (MCGs) and their refinement *entity connection graphs* (ECGs) to extract and visualize boundary flow topology on surfaces. MCG augments the vector field skeleton by addressing *periodic orbits*. Szymczak and Sipeki [SS13] presented an algorithm for drawing the MCG for topologically rich vector fields. The proposed visual representation of the MCG preserves the spatial relationships between its arcs and nodes and highlights the coherence between connecting trajectories.

The *transition graph* [SZ12] and *super-transition graph* [Szy11] have been proposed for robust and stable Morse decompositions for piecewise constant vector fields. The difference between these two graphs is that a transition graph represents trajectories of a single piecewise linear vector field, while a super-transition graph represents trajectories of all feasible vector fields.

4.6. Multifield Topology

Compared with univariate scalar field topology, much less work has been done on topological analysis and visualization of multivariate scalar fields. Carr and Duke [CD14] introduced *joint contour nets* (JCNs) which generalize the contour tree analysis for univariate scalar fields to multifields. The JCN is an approximation of the Reeb space of an arbitrary number of variables. The quantized contour trees of the single fields can be extracted from the JCN using a quotient graph algorithm. Before the entire concept of JCN was published, Duke et al. [DCK*12] also described an application of using graph-based techniques to visualize the JCN and analyze simulation data from nuclear physics, which leads to the insights into the detection of nuclear scission. Using multiple examples, they discussed practical methods for visualizing the JCN from many aspects, including effectiveness and aesthetic criteria. Huettenberger et al. [HHC*13] extended the topological structures from single scalar field to multifields using the concepts of Pareto optimality and Pareto dominance. For multivariate point clouds, Riech et al. [RL14] introduced the *simplicial chain graphs* as a visual metaphor of the inhomogeneous topological structure. They calculated the persistent homology of the data set and derived a localized description of simplicial chains, from which a graph structure was created to obtain the structural information.

5. Provenance-wise Representations and Techniques

Provenance refers to the lineage of an item. In the context of scientific workflow, it refers to all the information necessary to reproduce a certain piece of data. Although not a great deal of work relating to provenance- or history-wise graph representations and techniques has been done, this direction of research has its distinct foci and unique purposes. It dates back to the concept of "visualizing visualizations" [Ma00] which aims at better managing and exploring visualization processes and results. Later works extended this idea to storytelling generation, animation creation, and simulation parameter space exploration. For provenance visualization, the graphs produced resemble flowcharts widely used in designing and documenting complex processes or programs.

5.1. Visualizing Visualizations

The process of visual data exploration contains a wealth of information: parameters, results, history, as well as relationships among them. To learn lessons and share experiences, the process itself can be stored, tracked and analyzed. It can also be incorporated into, and thus becomes a part of the user interface of a visualization system. The work of *image graphs* by Ma [Ma99] was the first that visualizes the visualization process. As shown in Figure 20, image graphs

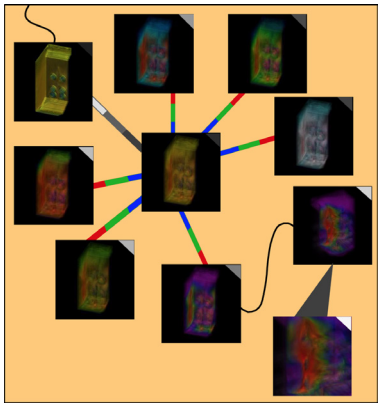


Figure 20: *Image graph for exploring the furnace data set. The user first searches for an appropriate color transfer function before deriving the desirable visualization shown in the lower right image. Image courtesy of Ma [Ma99] ©1999 IEEE.*

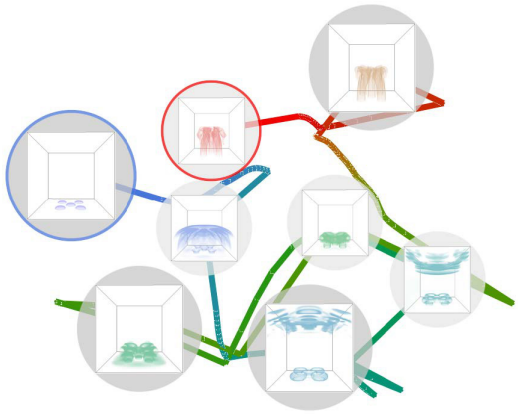


Figure 21: *A storyboard of the time-varying five jets data set. Blue to green to red are for early to middle to later time steps. Image courtesy of Lu and Shen [LS08] ©2008 IEEE.*

represent not only the results but also the process of data visualization. Each node in an image graph records an image and the corresponding visualization parameters used to produce it. Each edge in the graph shows the change in rendering parameters (e.g., color, opacity, rotation, zoom, shading, sampling, etc.) between the two nodes it connects.

Jankun-Kelly and Ma [JKM01] introduced a *spreadsheet-like interface* for visualization exploration. The interface displays the visualization parameter space and presents a clear correspondence between parameters and results through tabular organization. Jankun-Kelly et al. [JKMG02] further formalized a general model of the visualization exploration process. They designed the *derivation graph*, a collection of directed acyclic graphs to represent relationships between parameter value derivations. Each node in the graph represents a single session result. A directed edge exists between two nodes if and only if the node with the outgoing edge derives the node with the incoming edge.

Woodring and Shen [WS06b] proposed a volume shader for users to easily select and operate on many data volumes to create comparative visualization. They showed the contextual visualization of the volume shader by converting it to a *volume tree*. In a volume tree, nodes are operators and input data fields, and edges are input and output connections between operators. The visualization result is the root, and all input fields are leaves. One operational step forward in forming the final result is indicated by one level of the tree. The context is defined as the suboperations (i.e., subtrees of the volume tree) that form the final result.

Mayerich et al. [MBTR11] formulated a metric to compare hierarchical trees generated by different tracking and segmentation algorithms using neuron and microvascular data sets. In the context of neuron data sets, nodes in this kind of trees represent end points or branch points of fibers, and edges represent fiber filaments. The metric evaluates both geometry differences and connectivity differences. These difference values are mapped to colors of nodes and edges for visual tracking.

5.2. Storytelling, Animation and Simulation

Storytelling provides a powerful means for exploration and communication of data. Ma et al. [MLF*12] referred to *scientific storytelling* as telling stories using scientific data. Wohlfart and Hauser [WH07] introduced a *story model* that includes story nodes and story transitions. Story nodes are major steps or milestones in which a story briefly halts (for interactive exploration by the story consumer, for example) and then resumes. Story transitions connect story nodes smoothly leading from one node to the next.

Lu and Shen [LS08] presented an *interactive storyboard* for time-varying data visualization. As shown in Figure 21, the storyboard displays sample images and line drawings in a clear 2D layout to summarize complex data dynamics, such as relevancies and differences, in a concise and effective manner. Yu et al. [YLRC10] designed an *event graph* for automatic animation of time-varying data. The event graph embeds a tree-like structure and includes nodes, tree links, and relation links. Nodes represent event features from several aspects and at different scales. Tree links indicate the child and parent relationships of nodes belonging to the same feature aspect. Relation links indicate the similarities of time durations of nodes from different feature aspects. Balabanian et al. [BVG10] explored the combination of hierarchical data space and 3D spatial space. The input volumetric data set is segmented and hierarchically organized. They utilized a tree as a guiding structure for visual exploration of relationships among segmented components. The spatial characteristics of the data were integrated within the abstract view.

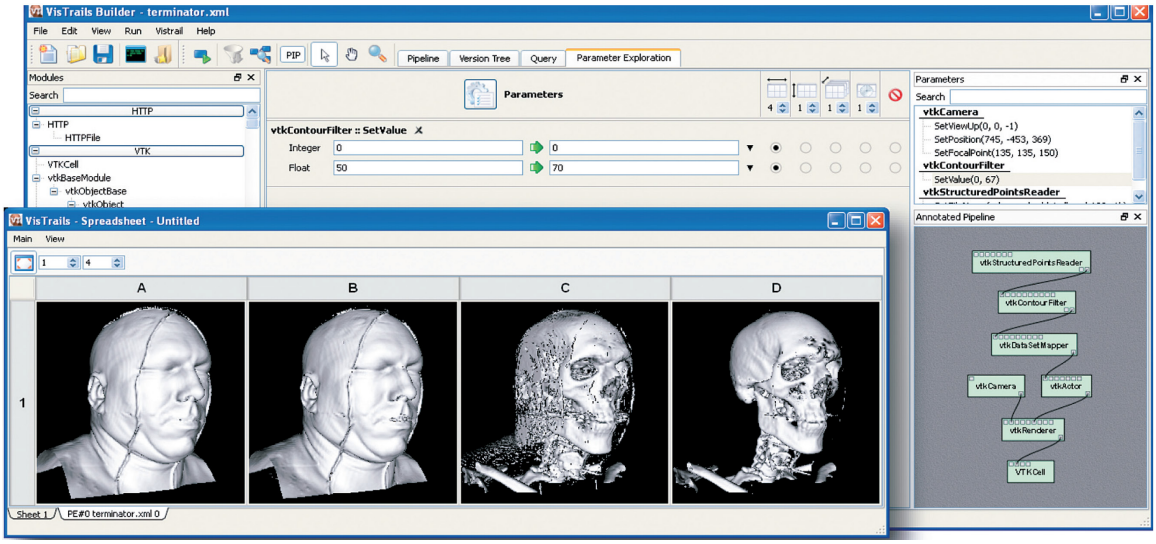


Figure 22: *The VisTrails builder and spreadsheet show the dataflow and visualization products. Image courtesy of Silva et al. [SFC07] ©2007 IEEE.*

For biological data, Wait et al. [WWB*14] used a *lineage tree* to depict the development of proliferating cells. The parent-daughter relations among the proliferating cells are tracked in consecutive time steps, and the cell that appears earlier is assigned to be the parent. Unlike traditional lineage trees that communicate information such as the cell cycle time and the number of progenies, this work focuses on revealing the spatial relationships between stem cells and their nutrient source. In their lineage trees, the vertical lines are perturbed and the x-axis is used to reflect the distance between a particular stem cell and its nearest vasculature voxel.

Timelines are commonly used in animation control and parameter exploration. Related work in this direction includes the timeline-based mixer (including multiple tracks and a template chooser) for creating animations for volume visualization [AWM10] and the timeline-based graph for exploring simulation parameter space [BM10]. For traffic data, Wang et al. [WYL*14] visualized the traffic over a long period through dynamic graph animation. In this context, each spatial cell is represented by a node, and the traffic between two cells is visualized as an edge. For each cell, the traffic speed is indicated by node color. In addition, the '>' sign is used inside the node to show the flow direction, and the number of '>' signs represents the relative flow volume passing the cell. To explore the graph over time, they created the pixel table view over location and time for different properties, such as abnormality, flow volume, and traffic speed. With this view, users can locate a certain time point with desired properties. Moreover, they created an intuitive animation to show vehicle movements in the local animation view. They also used the link/route filtering view to compare flow patterns on upstream/downstream links of a central cell.

5.3. Provenance Visualization

Typical visualization solutions only require final results. Besides final results, provenance visualization also includes capabilities for visualizing intermediate or partial results, derivation processes, and any information associated with used sources. Provenance techniques can facilitate the comprehension, verification and reproduction of scientific results by providing access to information about the sources and methods used to derive them.

The most notable example of provenance visualization is *VisTrails* [BCC*05, SFC07], a system that provides infrastructure for data exploration and visualization through workflows. An example of VisTrails builder and spreadsheet is shown in Figure 22. Provenance information managed by VisTrails refers to the modifications (e.g., addition, deletion or replacement) or history of changes made to a particular workflow in order to derive a new workflow. VisTrails renders this history of modifications as a tree-like structure where nodes represent a version of some workflow and edges represent the modification applied to a workflow in order to derive a new workflow. Upon accessing a particular node of the *provenance tree*, users are provided with a rendering of the scientific product which is generated as a result of a particular workflow associated with the node. Due to the clear separation between the specification of a pipeline and its execution instances, VisTrails features powerful scripting capabilities and provides a scalable mechanism for generating a large number of visualizations.

Groth and Streefkerk [GS06] presented a conceptual interaction model to support provenance and annotation for visual exploration systems. In their *interaction graph*, nodes

represent measurable states of the visualization system and edges denote transitions between states. States of the system are generically captured in the model and transitions might contain discrete interactions, such as zooming, translation, rotation, etc. Their model concisely captures state changes made by the user so that the recall of the steps taken to achieve the visual representation can be retrieved. By articulating with annotations, the prototypes implemented support a wide variety of knowledge discovery tasks as well as collaborative discovery and recall of past explorations.

6. Summary

Throughout this survey, we classify graph-based solutions into four categories. Nevertheless, we are also aware that there are other ways to slice and dice the papers covered in this survey. In this section, we revisit these papers and reveal their relationships from a different perspective. We divide the pipeline of generating and utilizing a graph-based visualization into three stages: *construction*, *organization* and *exploration*. For each stage, we summarize the most commonly adopted strategies with representative works. This summary serves a different purpose than the four categories. The four categories classify different approaches according to their underlying structures, which helps readers obtain the inner-connections among them. This summary, on the other hand, emphasizes what have been done for each key stage of graph-based visualization. It may provide a better guideline for researchers to apply such a technique to their own work.

6.1. Graph Construction

Graph construction consists of the definition and identification of nodes and edges. Node definition states what are the basic elements in a graph, and edge definition states the relationships to explore among these basic elements.

Node Generation. There are two major strategies to generate the set of nodes in scientific visualization: spatial/spatiotemporal partition and feature detection. Partition-wise techniques and some relationship-wise techniques adopt domain partitioning to generate nodes. For partition-wise techniques, we usually leverage partitioning trees over the spatial domain [Mea82] or hybrid partitioning trees over the space-time domain [SCM99]. The generated nodes are hierarchically organized with parent-child relationships. On the other hand, it is common to start with a uniform partition for relationship-wise techniques [GW11]. The nodes represent non-overlapping regions and are without a hierarchy at this point. Many techniques create hierarchical structures in a later stage by clustering nodes or partitioning the graph. An alternative approach to generate the set of nodes is to extract features in data and represent them as nodes. Many relationship-wise techniques [SSZC94, SW97, WS09, OSBM14] adopt this strategy, although the actual methods to detect features vary. We refer readers to Section 3.5 for details. Structure-wise techniques share more common sense in terms of node definition. They usually detect critical points and treat them as nodes [BPS97, CSvdP04, PSBM07, WBP07].

Edge Definition. The definition of edges is interrelated with that of nodes. For partition-wise techniques, we straightforwardly derive edges from the parent-child relationships among nodes. Similarly, in most structure-wise techniques, edge definitions come directly from the concepts behind them, e.g., Reeb graphs and Morse-Smale complexes. In structure-wise techniques, an edge usually indicates the connectivity, e.g., the neighborhood relationship [CL11]. In relationship-wise techniques, edge definitions are more diversified. Edges may be generated by defining the neighborhood relationship of two nodes using their spatial distance [PTRV12] or their distance in the feature space, such as transition probability [JS09, GW11]. They may also originate from the hierarchical structure by clustering the nodes [IVJ12, GRT14].

6.2. Graph Organization

Graph organization is needed since graphs generated at the construction stage may not be ideal. Larger data sets lead to larger graphs, which undoubtedly hinders the observation of graph structures as well as encoded data information. In addition, scientific data sets are often complicated and contain high-dimensional information. Therefore, ordinary graph attributes, e.g., node positions and edge connections, are not enough to present all the information. To deal with these issues, recent works take an extra step to organize the graph. Some techniques build a hierarchy on the nodes to *simplify* the graph for clearer observation, while others develop a more expressive representation to *enrich* the encoded information, which is often application-specific.

Simplification. In relationship-wise techniques, the most common way to simplify a graph is to group the nodes and form a hierarchy. We can perform node grouping bottom up by agglomerative clustering [PIB*11] or top down by graph partitioning [IVJ12]. The complete hierarchy can be captured by tree-based representations [LLRR08, IVJ12, GRT14], where all clusters at different levels are displayed. An alternative strategy is to display the nodes corresponding to coarse clusters at the beginning. Fine clusters are only available when users expand a node [GW11, MWS13]. In structure-wise techniques, it is also possible to simplify the graph by canceling critical point pairs [BPH05, GNP*06] and build a coarse-to-fine hierarchy [PSBM07]. Note that this simplification removes noise that is less persistent. When information beyond node connectivity is encoded, the associated information should be aggregated for presentation at a higher level. This is usually the case for relationship-wise techniques. For example, in the migration data sets, a node represents the population in a region, and an edge represents the volume of migrants between two regions. When the nodes are grouped, an aggregation is performed to sum the population in a group of regions and the number of migrants between groups of regions [Guo09, vLBR*16].

Besides explicit aggregation, information is sometimes aggregated implicitly, e.g., via edge bundling. In Figure 10 (a), the group-level connectivity is shown clearly with the bundled graph.

Enrichment. The enrichment is designed to encode the information that is not captured by ordinary graph attributes, e.g., node size and edge weight. Whether enrichment is necessary often depends on the application rather than the underlying approach to construct the graph. The entire graph may be redesigned for intuitive understanding: treemaps were used to encode the level-of-detail quality in multiresolution volume rendering [WS05, WS06a]; topological landscapes [WBP07] indicate the scalar values associated with critical points; and topological spines [CLB11] display nesting structures to demonstrate the surrounding contours of critical points. Other techniques use symbols to encode information on nodes: Sorger et al. [SBS*13] designed neuroMap to visualize a fruit fly's brain with different symbols representing cell bodies, arborizations, projections, overlapping information, and neuropils; Wang et al. [WYL*14] placed multiple '>' signs in nodes to indicate the traffic direction and volume. In addition, graph layouts can be utilized to convey useful information. The graphs can be embedded in the volume to provide contextual information [BSL*14, Guo09, PTRV12]. Wait et al. [WWB*14] perturbed the vertical lines of a lineage tree to reflect the distance between a stem cell and vessels.

6.3. Graph Exploration

Direct exploration of graphs includes zooming and focus+context techniques for large graphs, brushing of nodes and edges, and collapsing and expansion of subtrees, etc. Another commonly adopted strategy for graph exploration is to design *multiple coordinated views*. Those views usually plot the information about nodes which is not immediately available from the graph view, so that users can select nodes or graph components that fulfill certain properties. Examples include parallel coordinates [LS16], scatterplots [PTRV12], pixel table views [WYL*14], and other hybrid views [WLY*13, HZY*16, NSvA*13]. A more recent direction leverages *graph mining* solutions which directly mine the graph itself for knowledge discovery. For instance, Gu et al. [GWP*16] detected features and communities on the graph. Thomas and Natarajan [TN11, TN13] defined a similarity measure of subtrees to detect symmetry structures. Huang et al. [HZY*16] partitioned the graph according to the user-specified scalar property. Researchers also developed graph similarity measures and clustering solutions to capture high-level information [vLBR*16, WLY*13].

7. Research Trends

Throughout this survey, we have observed several research trends in graph-based representations and techniques for scientific visualization:

From Internal to External Representations. Early work on tree and graph structures focused on data representation and organization, mainly for adaptive processing and rendering. These representations were mostly internal, addressing issues such as data reduction and reuse, memory and I/O efficiency, and rendering performance. Over the years, these representations have gradually shifted from internal to external. Most external representations display the graph in a separate 2D view to avoid occlusion rather than superimposing the graph in the original 3D spatial data view. These external representations go beyond the traditional boundary of scientific visualization and incorporate information visualization techniques toward effective analysis of scientific data. Besides showing an overview of the data and relationships as abstract visual graphs, they also serve as interfaces for user interaction. In conjunction with the original data view via brushing and linking, users are able to compare relationships, track changes, and gain a more complete view and flexible control over data navigation and exploration.

From Simple to Complex Graphs. Graph representations are moving from simple to complex, small to large, often with hierarchical, sometime compound structures. This is partly because the research in scientific visualization has moved from univariate to multivariate data, from scalar to vector fields, and from steady to unsteady data. As the data get larger and more complicated, the corresponding graphs need to consider more advanced forms to encode various data relationships. Another reason that contributes to this trend lies in the growing need to tackle big data with coarse-to-fine analysis capability. Such a graph representation should allow users to not only gain a quick global overview but also identify local features and patterns, in a way that should be more simpler and easier to achieve in the transformed graph view than in the original data view.

From Straightforward to Advanced Solutions. As graphs derived from scientific data sets get larger and more complex, more advanced graph techniques from information visualization and graph drawing have been applied for effective visualization and interaction. From simple fixed graph layouts such as circular layouts to flexible force-directed layouts, from straightforward graph visualization to advanced spectral layout and layered graphs, from single graph to hierarchical and compound graph drawing, from steady to dynamic graphs, this trend will remain active and we expect more applications of state-of-the-art graph visualization techniques to investigate scientific data sets. We also expect novel graph interaction and navigation solutions to be presented in order to handle the ever-growing graphs in terms of both size and complexity.

8. Remaining Challenges

Remaining challenges for graph-based representations and techniques in scientific visualization include the following:

Graph Simplification and Mining. As the graphs derived from scientific data sets get larger and more complex,

there is a pressing need to process and present the graphs in a simplified form for easy human understanding and navigation. Techniques such as graph simplification will be very helpful to reduce the workload, both visually and manually, of users toward cost-effective analysis. When the data set is large and the relationships are complex, it might be simply impossible for users to find out community structures and track features over time in the resulting graph. Graph mining solutions could automatically identify communities and hotspots for detecting trends and anomalies, and align multiple graphs or subgraphs for finding common features and distinct patterns. These solutions will be more efficient and effective than simply asking users to extract relationships via standard brushing and linking techniques. Such graph functions represent a significant step forward and will be in great demand for big data visualization.

Online Streaming and In-Situ Graphs. Most current graph algorithms and techniques presented for scientific visualization, especially for those represented in a visual form for navigation and interaction, are built offline and on a single machine. Solutions for streaming, parallel and in-situ scenarios have not been explored. To fully leverage the aggregated computing power, we need to not only develop online dynamic graph solutions for extreme-scale scientific data but also incorporate graph generation, visualization and interaction into parallel and in-situ visualization. Much work remains to be done in order to demonstrate the feasibility, capability and scalability of graph-based techniques for analyzing and visualizing scientific data sets of terabytes, petabytes and beyond.

Graph Customization and Feedback Loop. In Section 6, we divide the graph-based visualization pipeline into three stages: *construction*, *organization* and *exploration*. At each stage, specific designs have been developed to deal with the diversified data and various needs from different applications and users. Scientific data sets are multivariate and time-varying, and therefore, multiple aspects of data need to be visualized. This means that no single graph encoding is sufficient to meet all the needs. In the graph exploration stage, a general solution can hardly accommodate the need to customize the exploration procedure. Because once the graph is constructed, the underlying node-link structure is usually not subject to change. As a result, the amount of customization that can be performed in the exploration stage is limited. To address this issue, we envision an additional *feedback* stage at the end of the pipeline which receives feedback during the exploration and updates the graph accordingly. Unlike the existing approaches, where users usually interact with nodes and edges, this feedback loop interacts with the entire graph by providing the mechanism to flexibly transform from one graph form to another. Achieving this will surely require long-term research efforts.

Evaluation and Knowledge Discovery. Since graphs are abstract representations extracted from scientific data sets, it is imperative to show that the added graph view would not increase much the burden for users to understand the underlying data but rather facilitate such a process. Early work on graph-based representations and techniques seldom included user evaluation. Recent work that presented visual graphs for navigation and exploration often incorporated ad-hoc feedback from domain experts or scientists. More rigorous evaluations are needed in order to verify that graph-based solutions indeed help scientists in their visual analysis and discovery of knowledge previously unknown or unable to get without the graphs. As transformation-based solutions for scientific visualization get increasingly popular, formal evaluation of general graph-based techniques for scientific visualization becomes necessary. This evaluation presents quite a few challenges ranging from evaluation design, experiment to analysis as many current solutions are restricted to one kind of scientific data or even tied together with a particular scientific application. Nevertheless, existing guidelines and practices for graph evaluation from information visualization offer practical guidance to achieve this goal.

9. Conclusions

We have presented a survey of graph-based representations and techniques for scientific visualization. The survey reviews related work in four categories, reexamines related publications in the graph-based visualization pipeline, and points out research trends and remaining challenges. A notable trend in this research is to develop graph-based visual representations and interfaces for scientific visualization, targeting large-scale time-varying multivariate scalar and vector field data. We believe that this direction of research is still on the rise as a number of new graph mappings and interactions have been introduced recently.

Many challenges remain to be solved which are related to big scientific data visualization. Novel graph-based solutions for streaming, parallel and in-situ settings are in great demand. Completing the feedback loop and evaluating graph-based techniques for scientific visualization are only in their infancy and much remains to be explored. We believe that the success of graph-based techniques will fundamentally change the tools and feature sets we have at hand to perform scientific visualization on a regular basis. Eventually, such graph views would become commonplace for researchers and vendors to integrate and provide in the scientific visualization workflow.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation through grants IIS-1456763 and IIS-1455886. The authors would like to thank Yi Gu for helping the identification of related publications.

References

[AWM10] AKIBA H., WANG C., MA K.-L.: AniViz: A template-based animation tool for volume visualization. *IEEE Computer Graphics and Applications 30*, 5 (2010), 61–71. 18

[BAAK*13] BEYER J., AL-AWAMI A., KASTHURI N., LICHTMAN J. W., PFISTER H., HADWIGER M.: ConnectomeExplorer: Query-guided visual analysis of large volumetric neuroscience data. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2868–2877. 8

[BBDW14] BECK F., BURCH M., DIEHL S., WEISKOPF D.: The state of the art in visualizing dynamic graphs. In *Eurographics Conference on Visualization - State-of-the-Art Reports* (2014). 3

[BCC*05] BAVOIL L., CALLAHAN S. P., CROSSNO P. J., FREIRE J., SCHEIDEGGER C. E., SILVA C. T., VO H. T.: VisTrails: Enabling interactive multiple-view visualizations. In *Proceedings of IEEE Visualization Conference* (2005), pp. 135–142. 18

[BDSW13] BISWAS A., DUTTA S., SHEN H.-W., WOODRING J.: An information-aware framework for exploring multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2683–2692. 7

[BKL*11] BENNETT J. C., KRISHNAMOORTHY V., LIU S., GROUT R. W., HAWKES E. R., CHEN J. H., SHEPHERD J., PASCUCCI V., BREMER P.-T.: Feature-based statistical analysis of combustion simulation data. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1822–1831. 15

[BM10] BRUCKNER S., MÖLLER T.: Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1468–1476. 18

[BNS01] BOADA I., NAVAZO I., SCOPIGNO R.: Multiresolution volume visualization with a texture-based octree. *The Visual Computer 17*, 3 (2001), 185–197. 4

[BPH05] BREMER P.-T., PASCUCCI V., HAMANN B.: Maximizing adaptivity in hierarchical topological models. In *Proceedings of International Conference on Shape Modeling and Applications* (2005), pp. 298–307. 14, 19

[BPS97] BAJAJ C. L., PASCUCCI V., SCHIKORE D. R.: The contour spectrum. In *Proceedings of IEEE Visualization Conference* (1997), pp. 167–173. 12, 19

[BSL*14] BÖTTGER J., SCHÄFER A., LOHMANN G., VILLRINGER A., MARGULIES D. S.: Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 471–480. 8, 20

[BVG10] BALABANIAN J.-P., VIOLA I., GRÖLLER M. E.: Interactive illustrative visualization of hierarchical volume data. In *Proceedings of Graphics Interface* (2010), pp. 137–144. 17

[BWT*11] BREMER P.-T., WEBER G., TIERNY J., PASCUCCI V., DAY M., BELL J.: Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics 17*, 9 (2011), 1307–1324. 14, 15

[Car04] CARR H.: *Topological Manipulation of Isosurfaces*. PhD thesis, The University of British Columbia, 2004. 12

[CD14] CARR H., DUKE D.: Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics 20*, 8 (2014), 1100–1113. 16

[CF11] CARMONA R., FROEHLICH B.: Error-controlled real-time cut updates for multi-resolution volume rendering. *Computers & Graphics 35*, 4 (2011), 931–944. 4

[CL11] CORREA C. D., LINDSTRÖM P.: Towards robust topology of sparsely sampled data. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1852–1861. 14, 19

[CLB11] CORREA C. D., LINDSTRÖM P., BREMER P.-T.: Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1842–1851. 14, 15, 20

[CML*07] CHEN G., MISCHAIKOW K., LARAMEE R. S., PILARCZYK P., ZHANG E.: Vector field editing and periodic orbit extraction using Morse decomposition. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 769–785. 16

[CNLS12] CHEN C.-M., NOUANESENGSY B., LEE T.-Y., SHEN H.-W.: Flow-guided file layout for out-of-core pathline computation. In *Proceedings of IEEE Symposium on large Data Analysis and Visualization* (2012), pp. 109–112. 12

[CQC*08] CHAN M.-Y., QU H., CHUNG K.-K., MAK W.-H., WU Y.: Relation-aware volume exploration pipeline. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1683–1690. 15

[CS13] CHEN C.-M., SHEN H.-W.: Graph-based seed scheduling for out-of-core FTLE and pathline computation. In *Proceedings of IEEE Symposium on Large Data Analysis and Visualization* (2013), pp. 15–23. 12

[CSA03] CARR H., SNOEYINK J., AXEN U.: Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications 24*, 2 (2003), 75–94. 12

[CSvdP04] CARR H., SNOEYINK J., VAN DE PANNE M.: Simplifying flexible isosurfaces using local geometric measures. In *Proceedings of IEEE Visualization Conference* (2004), pp. 497–504. 13, 19

[CSvdP10] CARR H., SNOEYINK J., VAN DE PANNE M.: Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry 43*, 1 (2010), 42–58. 13

[CXLS12] CHEN C.-M., XU L., LEE T.-Y., SHEN H.-W.: A flow-guided file layout for out-of-core streamline computation. In *Proceedings of IEEE Pacific Visualization Symposium* (2012), pp. 145–152. 11

[DCK*12] DUKE D., CARR H., KNOLL A., SCHUNCK N., NAM H. A., STASZCZAK A.: Visualizing nuclear scission through a multifield extension of topological analysis. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2033–2040. 16

[DCS09] DU Z., CHIANG Y.-J., SHEN H.-W.: Out-of-core volume rendering for time-varying fields using a space-partitioning time (SPT) tree. In *Proceedings of IEEE Pacific Visualization Symposium* (2009), pp. 73–80. 5

[DFD*14] DORAISWAMY H., FERREIRA N., DAMOULAS T., FREIRE J., SILVA C. T.: Using topological analysis to support event-guided exploration in urban data. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 2634–2643. 14

[FJS96] FINKELSTEIN A., JACOBS C. E., SALESIN D. H.: Multiresolution video. In *Proceedings of ACM SIGGRAPH Conference* (1996), pp. 281–290. 5

[GKHS98] GAGVANI N., KENCHAMMANA-HOSEKOTE D., SILVER D.: Volume animation using the skeleton tree. In *Proceedings of IEEE Symposium on Volume Visualization* (1998), pp. 47–53. 13

[GNP*06] GYULASSY A., NATARAJAN V., PASCUCCI V., BREMER P.-T., HAMANN B.: A topological approach to simplification of three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 474–484. 14, 19

[GRT14] GÜNTHER T., RÖSSL C., THEISEL H.: Hierarchical opacity optimization for sets of 3D line fields. *Computer Graphics Forum 33*, 2 (2014), 507–516. 6, 19

[GS06] GROTH D. P., STREEFKERK K.: Provenance and annotation for visual exploration systems. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1500–1510. 18

[Guo09] GUO D.: Flow mapping and multivariate visualization of large spatial interaction data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1041–1048. 9, 19, 20

[GW11] GU Y., WANG C.: TransGraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2015–2024. 7, 8, 19

[GW13] GU Y., WANG C.: iTree: Exploring time-varying data using indexable tree. In *Proceedings of IEEE Pacific Visualization Symposium* (2013), pp. 137–144. 6

[GWGS02] GUTHE S., WAND M., GONSER J., STRASSER W.: Interactive rendering of large volume data sets. In *Proceedings of IEEE Visualization Conference* (2002), pp. 53–60. 4

[GWLS05] GAO J., WANG C., LI L., SHEN H.-W.: A parallel multiresolution volume rendering algorithm for large data visualization. *Parallel Computing 31*, 2 (2005), 185–204. 4

[GWP*16] GU Y., WANG C., PETERKA T., JACOB R., KIM S. H.: Mining graphs for understanding time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 965–974. 8, 20

[GY95] GHAVAMNIA M. H., YANG X. D.: Direct rendering of Laplacian pyramid compressed volume data. In *Proceedings of IEEE Visualization Conference* (1995), pp. 192–199. 4

[HH89] HELMAN J. L., HESSELINK L.: Representation and display of vector field topology in fluid flow data sets. *IEEE Computer 22*, 8 (1989), 27–36. 16

[HHC*13] HUETTENBERGER L., HEINE C., CARR H., SCHEUERMANN G., GARTH C.: Towards multifield scalar topology based on pareto optimality. *Computer Graphics Forum 32*, 3 (2013), 341–350. 16

[HMM00] HERMAN I., MELANCON G., MARSHALL M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (2000), 24–43. 3

[HPSP01] HAHN H., PREIM B., SELLE D., PEITGEN H.-O.: Visualization and interaction techniques for the exploration of vascular structures. In *Proceedings of IEEE Visualization Conference* (2001), pp. 395–402. 12

[HSS*12] HEILAND R., SHIRINIFARD A., SWAT M., THOMAS G. L., SLUKA J., LUMSDAINE A., ZAITLEN B., GLAZIER J. A.: Visualizing cells and their connectivity graphs for CompuCell3D. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2012), pp. 85–90. 10

[HW10] HARVEY W., WANG Y.: Topological landscape ensembles for visualization of scalar-valued functions. *Computer Graphics Forum 29*, 3 (2010), 993–1002. 13

[HZY*16] HUANG X., ZHAO Y., YANG J., MA C., ZHANG C., YE X.: TrajGraph: A graph-based visual analytics approach to studying urban network centrality using taxi trajectory data. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 160–169. 9, 20

[IVJ12] IP C. Y., VARSHNEY A., JAJA J.: Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2355–2363. 6, 19

[JBS08] JÄNICKE H., BÖTTINGER M., SCHEUERMANN G.: Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1459–1466. 7

[JKM01] JANKUN-KELLY T. J., MA K.-L.: Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 275–287. 17

[JKMG02] JANKUN-KELLY T., MA K.-L., GERTZ M.: A model for the visualization exploration process. In *Proceedings of IEEE Visualization Conference* (2002), pp. 323–330. 17

[JS09] JÄNICKE H., SCHEUERMANN G.: Steady visualization of the dynamics in fluids using epsilon-machines. *Computers & Graphics 33*, 5 (2009), 597–606. 8, 19

[JT92] JAROMCZYK J. W., TOUSSAINT G. T.: Relative neighborhood graphs and their relatives. *Proceedings of the IEEE 80*, 9 (1992), 1502–1517. 14

[KPM*08] KUSS A., PROHASKA S., MEYER B., RYBAK J., HEGE H.-C.: Ontology-based visualization of hierarchical neuroanatomical structures. In *Eurographics Workshop on Visual Computing for Biomedicine* (2008), pp. 177–184. 9

[LBBH12] LINDOW N., BAUM D., BONDAR A., HEGE H.: Dynamic channels in biomolecular systems: Path analysis and visualization. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2012), pp. 99–106. 10

[Lev90] LEVOY M.: Efficient ray tracing of volume data. *ACM Transactions on Graphics 9*, 3 (1990), 245–261. 4

[LH91] LAUR D., HANRAHAN P.: Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Proceedings of ACM SIGGRAPH Conference* (1991), pp. 285–288. 4

[LHJ99] LAMAR E., HAMANN B., JOY K. I.: Multiresolution techniques for interactive texture-based volume visualization. In *Proceedings of IEEE Visualization Conference* (1999), pp. 355–362. 4

[LLRR08] LINSEN L., LONG T. V., ROSENTHAL P., ROSSWOG S.: Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1483–1490. 6, 19

[LPD*02] LINSEN L., PASCUCCI V., DUCHAINEAU M. A., HAMANN B., JOY K. I.: Hierarchical representation of time-varying volume data with $\sqrt[4]{2}$ subdivision and quadrilinear B-spline wavelets. In *Proceedings of Pacific Conference on Computer Graphics and Applications* (2002), pp. 346–355. 4

[LS08] LU A., SHEN H.-W.: Interactive storyboard for overall time-varying data visualization. In *Proceedings of IEEE Pacific Visualization Symposium* (2008), pp. 143–150. 17

[LS16] LIU X., SHEN H.: Association analysis for visual exploration of multivariate scientific data sets. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 955–964. 7, 20

[Ma99] MA K.-L.: Image graphs - a novel approach to visual data exploration. In *Proceedings of IEEE Visualization Conference* (1999), pp. 81–88. 16, 17

[Ma00] MA K.-L.: Visualizing visualizations: User interfaces for managing and exploring scientific visualization data. *IEEE Computer Graphics and Applications 20*, 5 (2000), 16–19. 16

[MBB*12] MCLENDON W. C., BANSAL G., BREMER P.-T., CHEN J., KOLLA H., BENNETT J. C.: On the use of graph search techniques for the analysis of extreme-scale combustion simulation data. In *Proceedings of IEEE Symposium on Large-Scale Data Analysis and Visualization* (2012), pp. 57–63. 15

[MBTR11] MAYERICH D., BJORNSSON C., TAYLOR J., ROYSAM B.: Metrics for comparing explicit representations of interconnected biological networks. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2011), pp. 79–86. 17

[Mea82] MEAGHER D. J.: Geometric modeling using octree encoding. *Computer Graphics and Image Processing 19*, 2 (1982), 129–147. 4, 19

[MLF*12] MA K.-L., LIAO I., FRAZIER J., HAUSER H., KOSTIS H.-N.: Scientific storytelling using visualization. *IEEE Computer Graphics and Applications 32*, 1 (2012), 12–19. 17

[MPH94] MA K.-L., PAINTER J. S., HANSEN C. D.: Parallel volume rendering using binary-swap compositing. *IEEE Computer Graphics and Applications 14*, 4 (1994), 59–67. 5

[MS00] MA K.-L., SHEN H.-W.: Compression and accelerated rendering of time-varying data. In *Proceedings of International Computer Symposium - Workshop on Computer Graphics and Virtual Reality* (2000), pp. 82–89. 4

[MWS13] MA J., WANG C., SHENE C.-K.: FlowGraph: A compound hierarchical graph for flow field exploration. In *Proceedings of IEEE Pacific Visualization Symposium* (2013), pp. 233–240. 8, 19

[MWSJ14] MA J., WANG C., SHENE C.-K., JIANG J.: A graph-based interface for visual analytics of 3D streamlines and pathlines. *IEEE Transactions on Visualization and Computer Graphics 20*, 8 (2014), 1127–1140. 4, 8

[NLS11] NOUANESENGSY B., LEE T.-Y., SHEN H.-W.: Load-balanced parallel streamline generation on large scale vector fields. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1785–1794. 11, 12

[NSvA*13] NOWKE C., SCHMIDT M., VAN ALBADA S. J., EPPLER J. M., BAKKER R., DIESRNANN M., HENTSCHEL B., KUHLEN T.: VisNEST - interactive analysis of neural activity data. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2013), pp. 65–72. 9, 20

[OFH*11] OELTZE S., FREILER W., HILLERT R., DOLEISCH H., PREIM B., SCHUBERT W.: Interactive, graph-based visual analysis of high-dimensional, multi-parameter fluorescence microscopy data in toponomics. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 1882–1891. 7

[OSBM14] OZER S., SILVER D., BEMIS K., MARTIN P.: Activity detection in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 337–390. 10, 19

[PCMS05] PASCUCCI V., COLE-MCLAUGHLIN K., SCORZELLI G.: *Multi-Resolution Computation and Presentation of Contour Trees*. Tech. Rep. UCRL-PROC-208680, Lawrence Livermore National Laboratory, 2005. 13, 15

[PIB*11] PATRO R., IP C. Y., BISTA S., CHO S. S., THIRUMALAI D., VARSHNEY A.: MDMap: A system for data-driven layout and exploration of molecular dynamics simulations. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2011), pp. 111–118. 6, 19

[PO08] PREIM B., OELTZE S.: 3D visualization of vasculature: An overview. In *Visualization in Medicine and Life Sciences*, Mathematics and Visualization. Springer Berlin Heidelberg, 2008, pp. 39–59. 12

[PSBM07] PASCUCCI V., SCORZELLI G., BREMER P.-T., MASCARENHAS A.: Robust on-line computation of Reeb graphs: Simplicity and speed. *ACM Transactions on Graphics 26*, 3 (2007). 13, 19

[PTRV12] PARULEK J., TURKAY C., REUTER N., VIOLA I.: Implicit surfaces for interactive graph based cavity analysis of molecular simulations. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2012), pp. 115–122. 10, 11, 19, 20

[QCX*07] QU H., CHAN W.-Y., XU A., CHUNG K.-L., LAU K.-H., GUO P.: Visual analysis of the air pollution problem in Hong Kong. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1408–1415. 7

[RL14] RIECH B., LEITTE H.: Structural analysis of multivariate point clouds using simplicial chains. *Computer Graphics Forum 33*, 8 (2014), 28–37. 16

[SB06] SOHN B.-S., BAJAJ C. L.: Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics 12*, 1 (2006), 14–25. 14

[SBS*13] SORGER J., BUHLER K., SCHULZE F., LIU T., DICKSON B.: neuroMap - interactive graph-visualization of the fruit fly's neural circuit. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2013), pp. 73–80. 9, 20

[SCM99] SHEN H.-W., CHIANG L.-J., MA K.-L.: A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. In *Proceedings of IEEE Visualization Conference* (1999), pp. 371–377. 4, 19

[SFC07] SILVA C. T., FREIRE J., CALLAHAN S. P.: Provenance for visualizations: Reproducibility and beyond. *IEEE Computing in Science & Engineering 9*, 5 (2007), 82–89. 18

[She98] SHEN H.-W.: Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In *Proceedings of IEEE Visualization Conference* (1998), pp. 159–166. 4

[Shn92] SHNEIDERMAN B.: Tree visualization with tree-maps: A 2D space-filling approach. *ACM Transactions on Graphics 11*, 1 (1992), 92–99. 5, 14

[SIGM*11] SUTER S. K., IGLESIAS GUITIÁN J. A., MARTON F., AGUS M., ELSENER A., ZOLLIKOFER C. P. E., GOPI M., GOBBETTI E., PAJAROLA R.: Interactive multiscale tensor reconstruction for multiresolution volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2135–2143. 4

[SMP13] SUTER S. K., MAKHYNIA M., PAJAROLA R.: TAMRESH - tensor approximation multiresolution hierarchy for interactive volume visualization. *Computer Graphics Forum 32*, 3 (2013), 151–160. 4

[SS13] SZYMCZAK A., SIPEKI L.: Visualization of Morse connection graphs for topologically rich 2D vector fields. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2763–2772. 16

[SSW14] SAIKIA H., SEIDEL H.-P., WEINKAUF T.: Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum 33*, 3 (2014), 41–50. 15

[SSZC94] SAMTANEY R., SILVER D., ZABUSKY N., CAO J.: Visualizing features and tacking their evolution. *IEEE Computer 27*, 7 (1994), 20–27. 10, 19

[STS06] SAUBER N., THEISEL H., SEIDEL H.-P.: Multifield-Graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 917–924. 7

[SW97] SILVER D., WANG X.: Tracking and visualizing turbulent 3D features. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 129–141. 10, 19

[SWC*08] SCHNEIDER D., WIEBEL A., CARR H., HLAWITSCHKA M., SCHEUERMANN G.: Interactive comparison of

scalar fields based on largest contours with applications to flow visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1475–1482. 15

[SZ12] SZYMCZAK A., ZHANG E.: Robust Morse decompositions of piecewise constant vector fields. *IEEE Transactions on Visualization and Computer Graphics 18*, 6 (2012), 851–860. 16

[Szy11] SZYMCZAK A.: Stable Morse decompositions for piecewise constant vector fields on surfaces. *Computer Graphics Forum 30*, 3 (2011), 851–860. 16

[TN11] THOMAS D. M., NATARAJAN V.: Symmetry in scalar field topology. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2035–2044. 15, 20

[TN13] THOMAS D. M., NATARAJAN V.: Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2663–2672. 15, 20

[TTFN05] TAKESHIMA Y., TAKAHASHI S., FUJISHIRO I., NIELSON G. M.: Introducing topological attributes for objective-based visualization of simulated datasets. In *Proceedings of International Workshop on Volume Graphics* (2005), pp. 137–236. 13

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proceedings of IEEE Visualization Conference* (2003), pp. 225–232. 16

[vLBR*16] VON LANDESBERGER T., BRODKORB F., ROSKOSCH P., ANDRIENKO G., ANDRIENKO N., KEREN A.: MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 11–20. 9, 19, 20

[vLKS*11] VON LANDESBERGER T., KUIJPER A., SCHRECK T., KOHLHAMMER J., VAN WIJK J. J., FEKETE J.-D., FELLNER D. W.: Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum 30*, 6 (2011), 1719–1749. 3

[WBP07] WEBER G. H., BREMER P.-T., PASCUCCI V.: Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1416–1423. 13, 19, 20

[WCB12] WIDANAGAMAACHCHI W., CHRISTENSENT C., BREMER P.-T.: Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *Proceedings of IEEE Symposium on Large Data Analysis and Visualization* (2012), pp. 9–17. 14

[WDC*07] WEBER G. H., DILLARD S. E., CARR H., PASCUCCI V., HAMANN B.: Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 330–341. 13

[WH07] WOHLFART M., HAUSER H.: Story telling for presentation in volume visualization. In *Proceedings of Joint Eurographics - IEEE VGTC Symposium on Visualization* (2007), pp. 91–98. 17

[WLY*13] WANG Z., LU M., YUAN X., ZHANG J., VAN DE WETERING H.: Visual traffic jam analysis based on trajectory data. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2159–2168. 11, 20

[WS04] WANG C., SHEN H.-W.: *A Framework for Rendering Large Time-Varying Data Using Wavelet-Based Time-Space Partitioning (WTSP) Tree*. Tech. Rep. OSU-CISRC-1/04-TR05, The Ohio State University, 2004. 5

[WS05] WANG C., SHEN H.-W.: Hierarchical navigation interface: Leveraging multiple coordinated views for level-of-detail multiresolution volume rendering of large scientific data sets. In *Proceedings of International Conference on Information Visualisation* (2005), pp. 259–267. 5, 20

[WS06a] WANG C., SHEN H.-W.: LOD map - a visual interface for navigating multiresolution volume visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1029–1036. 5, 6, 20

[WS06b] WOODRING J., SHEN H.-W.: Multi-variate, time-varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 909–916. 17

[WS09] WOODRING J., SHEN H.-W.: Semi-automatic time-series transfer functions via temporal clustering and sequencing. *Computer Graphics Forum 28*, 3 (2009), 791–798. 10, 19

[WVG92] WILHELMS J., VAN GELDER A.: Octrees for faster isosurface generation. *ACM Transactions on Graphics 11*, 3 (1992), 201–227. 4

[WVG94] WILHELMS J., VAN GELDER A.: Multi-dimensional trees for controlled volume rendering and compression. In *Proceedings of IEEE Symposium on Volume Visualization* (1994), pp. 27–34. 4

[WWB*14] WAIT E., WINTER M., BJORNSSON C., KOKOVAY E., WANG Y., GODERIE S., TEMPLE S., COHEN A. R.: Visualization and correction of automated segmentation, tracking and lineaging from 5-D stem cell image sequences. *BMC Bioinformatics 15*, 1 (2014), 1–14. 18, 20

[WYG*11] WANG C., YU H., GROUT R. W., MA K.-L., CHEN J. H.: Analyzing information transfer in time-varying multivariate data. In *Proceedings of IEEE Pacific Visualization Symposium* (2011), pp. 99–106. 7

[WYL*14] WANG Z., YE T., LU M., YUAN X., QU H., YUAN J., WU Q.: Visual exploration of sparse traffic trajectory data. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1813–1822. 18, 20

[WYM10] WANG C., YU H., MA K.-L.: Application-driven compression for visualizing large-scale time-varying data. *IEEE Computer Graphics and Applications 30*, 1 (2010), 59–69. 4

[WYM13] WANG Y., YU H., MA K.-L.: Scalable parallel feature extraction and tracking for large time-varying 3D volume data. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization* (2013), pp. 17–24. 11

[XS10] XU L., SHEN H.-W.: Flow web: A graph based user interface for 3D flow field exploration. In *Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis* (2010). 8, 11

[YLRC10] YU L., LU A., RIBARSKY W., CHEN W.: Automatic animation for time-varying data visualization. *Computer Graphics Forum 29*, 7 (2010), 2271–2280. 17

[YWM07] YU H., WANG C., MA K.-L.: Parallel hierarchical visualization of large time-varying 3D vector fields. In *Proceedings of ACM/IEEE Conference on Supercomputing* (2007). 4, 11

[YWM08] YU H., WANG C., MA K.-L.: Massively parallel volume rendering using 2-3 swap image compositing. In *Proceedings of ACM/IEEE Supercomputing Conference* (2008). 5

[ZT09] ZHOU J., TAKATSUKA M.: Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1481–1488. 13

Biography

Chaoli Wang is currently an associate professor of computer science and engineering at University of Notre Dame. He received a Ph.D. degree in computer and information science from The Ohio State University in 2006. He was a postdoctoral researcher at University of California, Davis from 2007 to 2009, and an assistant professor of computer science at Michigan Technological University from 2009 to 2014. Dr. Wang's main research interest is scientific visualization, in particular on the topics of time-varying multivariate data visualization, flow visualization, and information-theoretic algorithms and graph-based techniques for big data analytics. He has published extensively in visual analysis of scientific data sets leveraging various tree and graph representations. Dr. Wang is a recipient of the U.S. National Science Foundation CAREER Award.

Jun Tao is currently a postdoctoral researcher at University of Notre Dame. He received a Ph.D. degree in computer science from Michigan Technological University in 2015. Dr. Tao's major research interest is scientific visualization, especially on applying information theory, optimization techniques, and topological analysis to flow visualization and multivariate data exploration. He is also interested in graph-based visualization, image collection visualization, and software visualization.