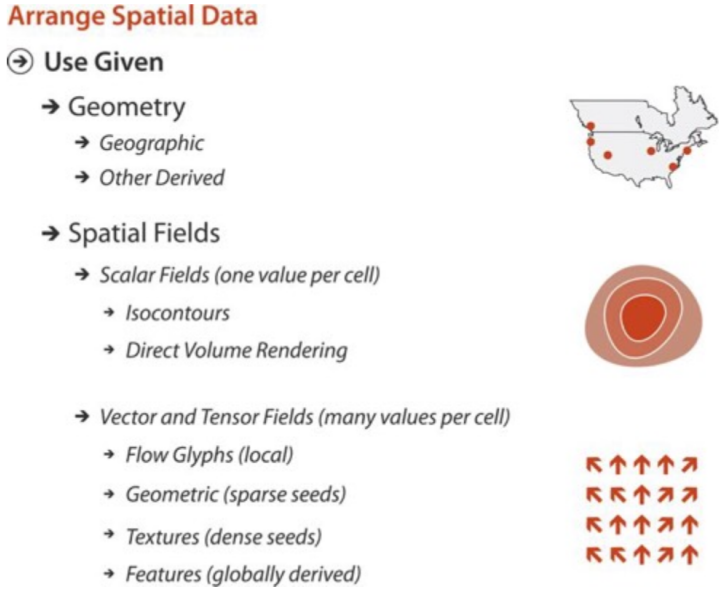
# Chapter 8 Arrange Spatial Data

## 8.1 The Big Picture

For datasets with spatial semantics, the usual choice for *arrange* is to *use* the given spatial information to guide the layout. In this case, the choices of express, *separate, order*, and *align* do not apply because the position channel is not available for directly encoding attributes. The two main spatial data types are geometry, where shape information is directly conveyed by spatial elements that do not necessarily have associated attributes, and spatial fields, where attributes are associated with each cell in the field. Figure 8.1 summarizes the major approaches for arranging these two data types. In a visualization context, geometry data typically either is geographic or has explicitly been derived from some other data type due to a design choice. For scalar fields with one attribute at each field cell, the two main visual encoding idiom families are isocontours and direct volume rendering. For both vector and tensor fields, with multiple attributes at each cell, there are four families of encoding idioms: flow glyphs that show local information, geometric approaches that compute derived geometry from a sparse set of seed points, texture approaches that use a dense set of seeds, and feature approaches where data is derived with global computations using information from the entire spatial field.

**Figure 8.1. Design choices for using given spatial data: geometry or spatial fields.**




## 8.2 Why Use Given?

The common case with spatial data is that the given spatial position is the attribute of primary importance because the central tasks revolve around understanding spatial relationships. In these cases, the right visual encoding choice is to use the provided spatial position as the substrate for the visual layout, rather than to visually encode other attributes with marks using the spatial position channel. This choice may seem obvious from common sense alone. It also follows from the effectiveness principle, since the most effective channel of spatial position is used to show the most important aspect of the data, namely, the spatial relationships between elements in the dataset.

► The expressiveness principle is covered in Section 5.4.1.

Of course, it is possible that datasets with spatial attribute semantics might not have the task involving understanding of spatial relationships as the primary concern. In these cases, the question of which other attributes

to encode with spatial position is once again on the table.

## 8.3 Geometry

Geometric data does not necessarily have attributes associated with it: it conveys shape information directly through the spatial position of its elements. The field of computer graphics addresses the problem of simply drawing geometric data. What makes geometry interesting in a vis context is when it is derived from raw source data as the result of a design decision at the abstraction level. A common source of derived geometry data is **geographic** information about the Earth. Geometry is also frequently derived from computations on spatial fields.

### 8.3.1 Geographic Data

Cartographers have grappled with design choices for the visual representation of geographic spatial data for many hundreds of years. The term **cartographic generalization** is closely related to the term *abstraction* as used in this book: it refers to the set of choices about how to derive an appropriate geometry dataset from raw data so that it is suitable for the intended task of the map users. This concept includes considerations discussed in this book such as filtering, aggregation, and level of detail. For example, a city might be indicated with a point mark in a map drawn at the scale of an entire country, or as an area mark with detailed geometric information showing the shape of its boundaries in a map at the scale of a city and its surrounding suburbs. Cartographic data includes what this book classifies as nonspatial information: for example, population data in the form of a table could be used to size code the point marks representing cities by their population.*

► Filtering, aggregation, and level of detail are discussed in Chapter 13.

* The integration of nonspatial data with base spatial data is referred to as **thematic cartography** in the cartography literature.

**Example: Choropleth Maps**



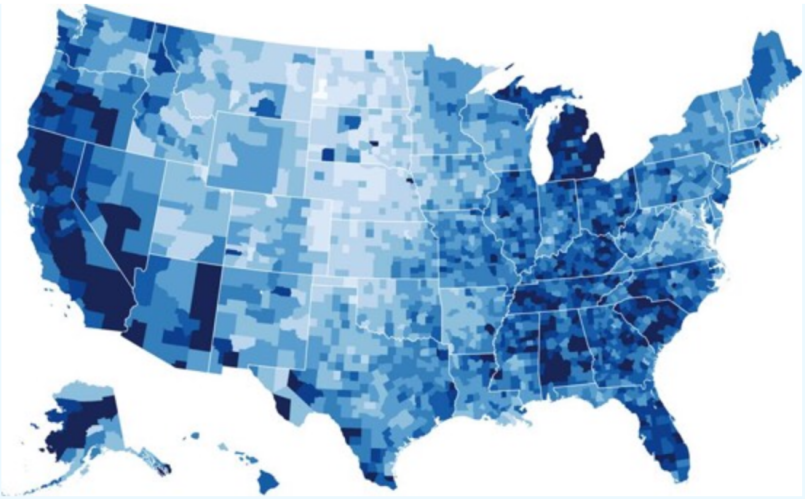
A **choropleth** map shows a quantitative attribute encoded as color over regions delimited as area marks, where the shape of each region is determined by using given geometry. The region shapes might either be provided directly as the base dataset or derived from base data based on cartographic generalization choices. The major design choices for choropleths are how to construct the colormap, and what region boundaries to use.

Figure 8.2 shows an example of US unemployment rates from 2008 with a segmented sequential colormap. The white-to-blue colormap has a sequence of nine levels with monotonically decreasing luminance. The region granularity is counties within states.

► Sequential colormaps are covered in Section 10.3.2.

► The problem of spatial aggregation and its relationship to region boundaries is covered in Section 13.4.2.

**Figure 8.2. Choropleth map showing regions as area marks using given geometry, where a quantitative attribute is encoded with color.**



*From http://bl.ocks.org/mbostock/4060606.*

| Idiom | Choropleth Map |
|---|---|
| **What: Data** | Geographic geometry data. Table with one quantitative attribute per region. |
| **How: Encode** | Space: use given geometry for area mark boundaries. Color: sequential segmented colormap. |

### 8.3.2 Other Derived Geometry

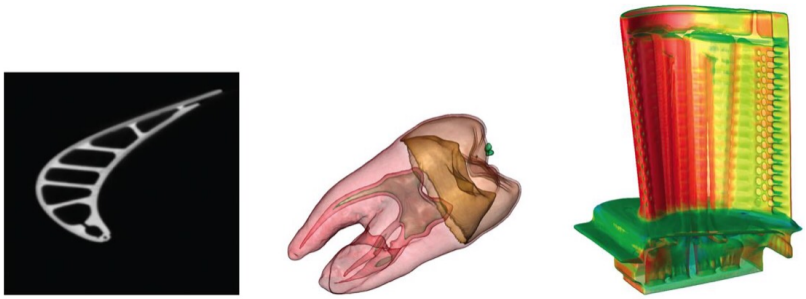
Geometry data used in vis can also arise from spatial data that is not geographic. It is frequently derived through computations on spatial fields, as discussed below.

## 8.4 Scalar Fields: One Value

A scalar spatial field has a single value associated with each spatially defined cell. Scalar fields are often collected through medical imaging, where the measured value is radio-opacity in the case of computed tomography (CT) scans and proton density in the case of magnetic resonance imaging (MRI) scans.

There are three major families of idioms for visually encoding scalar fields: slicing, as shown in Figure 8.3(a); isocontours, as in shown Figure 8.3(b); and direct volume rendering, as shown in Figure 8.3(c). With the **isocontours** idiom, the derived data of lower-dimensional surface geometry is computed and then is shown using standard computer graphics techniques: typically 2D isosurfaces for a 3D field, or 1D isolines for a 2D field. With the **direct volume rendering** idiom, the computation to generate an image from a particular 3D viewpoint makes use of all of the information in the full 3D spatial field. With the **slicing** idiom, information about only two dimensions at once is shown as an image; the slice might be aligned with the original axes of the spatial field or could have an arbitrary orientation in 3D space. In all of these cases, geometric navigation is the usual approach to interaction. The idioms can be combined, for example, by providing an interactively controllable widget for selecting the position and orientation of a slice embedded within direct volume rendering view.

**Figure 8.3. Spatial scalar fields shown with three different idioms. (a) A single 2D slice of a turbine blade dataset. (b) Multiple semitransparent isosurfaces of a 3D tooth dataset. (c) Direct volume rendering of the entire 3D turbine dataset.**



(a) (b) (c)

*From [Kniss 02, Figures 1.2 and 2.1b].*

► Slicing is also covered in Section 11.6.1, in the context of other idioms for attribute reduction.

► Section 11.5 covers geometric navigation.

### 8.4.1 Isocontours

A set of **isolines**, namely, lines that represent the contours of a particular level of the scalar value, can be derived from a scalar spatial field.* The isolines will occur far apart in regions of slow change and close together in regions of fast change but will never overlap; thus, contours for many different values can be shown simultaneously without excessive visual clutter. Color coding the regions between the contours with a sequential colormap yields a **contour plot**, as shown in Figure 6.9(c).

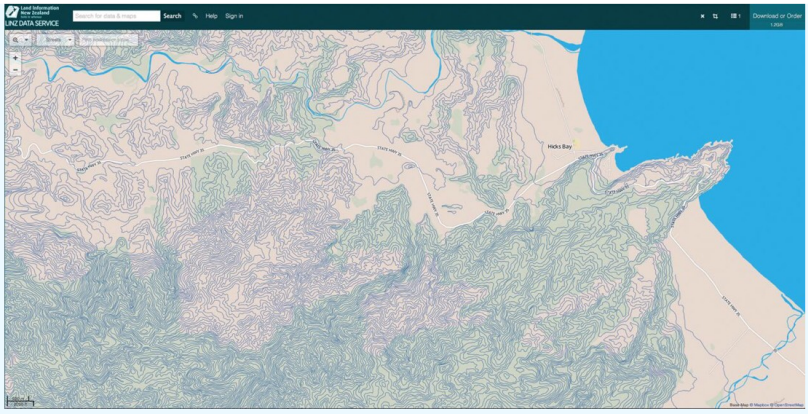
* Synonyms for *isolines are* **contour lines** and **isopleths**.

**Example: Topographic Terrain Maps**

Topographic terrain maps are a familiar example of isolines in widespread use by the general public. They show the contours of equal elevation above sea level layered on top of the spatial substrate of a geographic map. Figure 8.4 shows contours every 10 meters, with nearly 80 levels in total. Small closed contours indicate mountain peaks, and the flat regions near sea level have no lines at all.

| Idiom | Topographic Terrain Map |
|---|---|
| **What: Data** | 2D spatial field; geographic data. |
| **What: Derived** | Geometry: set of isolines computed from field. |
| **How: Encode** | Use given geographic data geometry of points, lines, and region marks. Use derived geometry as line marks (blue). |
| **Why: Tasks** | Query shape. |
| **Scale** | Dozens of contour levels. |

**Figure 8.4. Topographic terrain map, with isolines in blue.**




*From https://data.linz.govt.nz/layer/768-nz-mainland-contours-topo-150k.*

The idiom of **isosurfaces** transforms a 3D scalar spatial field into one or more derived 2D surfaces that represent the contours of a particular level of the scalar value. The resulting surface is usually shown with interactive 3D navigation controls for changing the viewpoint using rotation, zooming, and translation.

► Spatial navigation is discussed further in Section 11.5.

In the 3D case, simply showing all of the contour surfaces for dozens of values at once is not feasible, because the outer contour surfaces would occlude all of the inner ones. Thus, one crucial question is how to determine which level will produce the most useful result. Exploration is frequently supported by providing dynamic controls for changing the chosen level on the fly, for example, with a slider that allows the user to quickly change the contour value from the minimum to the maximum value within the dataset.

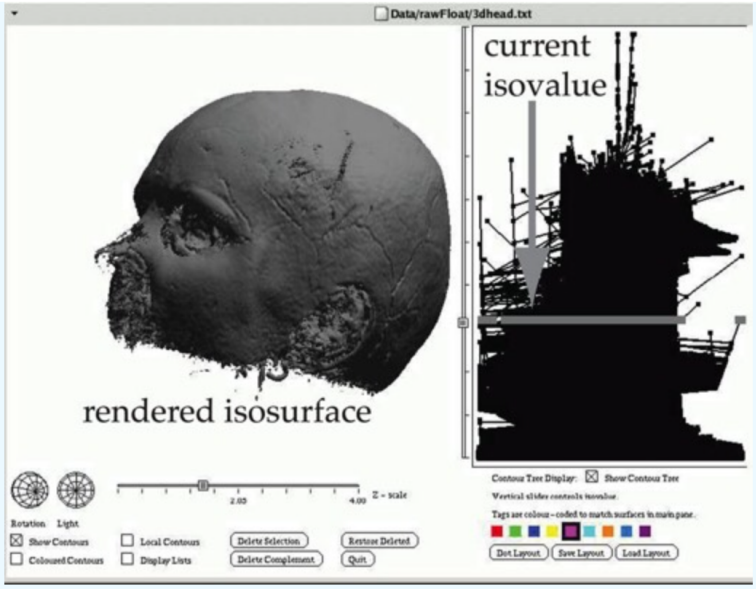
With careful use of colors and transparency, several isosurfaces can be shown at once. Figure 8.3(c) shows a 3D spatial field of a human tooth with five distinguishable isosurfaces.

**Example: Flexible Isosurfaces**

The flexible isosurfaces idiom uses one more level of derived data, the *simplified contour tree*, to help users find structure that would be hidden with the standard single-level approach. There may be multiple disconnected isosurfaces for a given value: as the value changes, individual components could appear, join or split, or disappear. The **contour tree** tracks this evolution explicitly, showing how the connected isosurface components change their nesting structure. The full tree is very complex, as shown in Figure 8.5; there are over 1.5 million edges for the head dataset. Careful simplification of the tree yields a manageable result of under 100 edges, as shown in Figure 8.6. Using this structure for filtering and coloring via multiple coordiated views supports interactive exploration. Figure 8.6 shows several meaningful structures within the head that have been identified through this kind of exploration; seeing them all within the same 3D view allows users to understand both their shape and their relative position to each other.

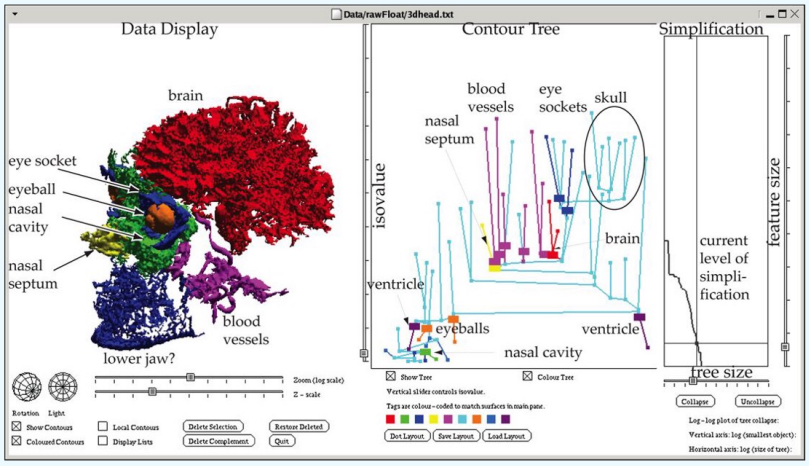
► Filtering is discussed in Section 13.3.2 and coordinating multiple views is discussed in Section 12.3.

**Figure 8.5. A full contour tree with over 1.5 million edges does not help the user explore isosurfaces.**




*From [Carr et al. 04, Figure 1].*

**Figure 8.6. The flexible isosurfaces idiom uses the simplified contour tree of under 100 edges to help users identify meaningful structure.**




*From [Carr et al. 04, Figure 1].*

| Idiom | Flexible Isosurfaces |
|---|---|
| **What: Data** | Spatial field. |
| **What: Derived** | Geometry: surfaces. Tree: simplified contour tree. |
| **How: Encode** | Surfaces: use given. Tree: line marks, vertical spatial position encodes isovalue. |
| **Why: Tasks** | Query shape. |
| **Scale** | One dozen contour levels. |

## 8.4.2 Direct Volume Rendering

The **direct volume rendering** idiom creates an image directly from the information contained within the scalar spatial field, without deriving an intermediate geometric representation of a surface. The algorithmic issues

involved in the computation are complex; a great deal of work has been devoted to the question of how to carry it out efficiently and correctly.

A crucial visual encoding design choice with direct volume rendering is picking the **transfer function** that maps changes in the scalar value to opacity and color. Finding the right transfer function manually often requires considerable trial and error because features of interest in the spatial field can be difficult to isolate: uninteresting regions in space may contain the same range of data values as interesting ones.

**Example: Multidimensional Transfer Functions**

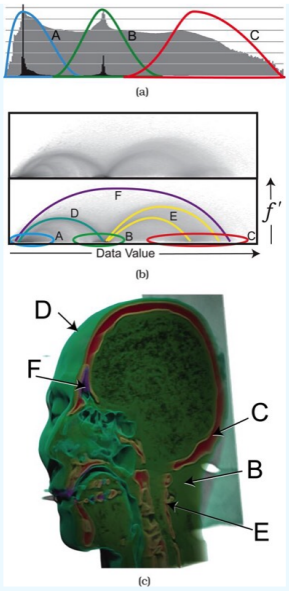
The Simian system [Kniss 02, Kniss et al. 05] uses a derived space and a set of interactive widgets for specifying regions within it to help the user construct multidimensional transfer functions. The horizontal axis of this space corresponds to the data value of the scalar function. The vertical axis corresponds to the magnitude of the gradient,[1] the direction of fastest change, so that regions of high change can be distinguished from homogeneous regions. Figure 8.7(a) shows the information that can be considered part of a standard 1D transfer function: the histogram of the data values. The histogram shows both the linear scale values in black, and the log scale values in gray. In this view, only the basic three materials can be distinguished from each other: (A) air, (B) soft tissue, and (C) bone. Figure 8.7(b) shows that more information can be seen in the 2D joint histogram of the full derived space, where the vertial axis shows the gradient magnitude. This view is like a heatmap with very small area marks of one pixel each, where each cell shows a count of how many values occur within it using a grayscale colormap. In this view, boundaries between the basic surfaces also form distinguishable structures. Figure 8.7(c) presents a volume rendering of a head dataset using the resulting 2D transfer function, showing examples of the base materials and these three boundaries: (D) air–tissue, (E) tissue–bone, and (F) air–bone. A cutting plane has been positioned to show the internal structure of the head.

► The histogram visual encoding idiom is covered in Section 13.4.1.

► Cutting planes are covered in Section 11.6.2.

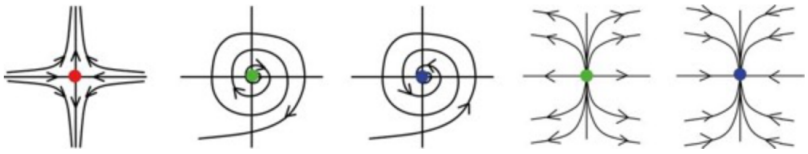
| Idiom | Multidimensional Transfer Functions |
|---|---|
| **What: Data** | 3D spatial field. |
| **What: Derived** | 3D spatial field: gradient of original field. |
| **What: Derived** | Table: two key attributes, values binned from min to max for both data and derived data. One derived quantitative value attribute (item count per bin). |
| **How: Encode** | 3D view: use given spatial field data, color and opacity from multidimensional transfer function. Joint histogram view: area marks in 2D matrix alignment, grayscale sequential colormap. |

**Figure 8.6. Simian allows users to construct multidimensional transfer functions for direct volume rendering using a derived space. (a) The standard 1D histogram can show the three basic materials: (A) air, (B) soft tissue, and (C) bone. (b) The full 2D derived space allows material boundaries to be distinguished as well. (c) Volume rendering of head dataset using the resulting 2D transfer function, showing material boundaries of (D) air–tissue, (E) tissue–bone, and (F) air–bone.**




*From [Kniss et al. 05, Figure 9.1].*

**Figure 8.7. The main types of critical points in a flow field: saddle, circulating sinks, circulating sources, noncirculating sinks, and noncirculating sources.**



*From [Tricoche et al. 02, Figure 1].*

1 Mathematically, the gradient is the first derivative.

## 8.5 Vector Fields: Multiple Values

Vector field datasets are often associated with the application domain of computational fluid dynamics (CFD), as the outcome of flow simulations or measurements. Flow vis in particular deals with a specific kind of vector field, a velocity field, that contains information about both direction and magnitude at each cell. The three common cases are purely 2D spatial fields, purely 3D spatial fields, and the intermediate case of flow on a 2D surface embedded within 3D space. Time-varying flow datasets are called **unsteady**, as opposed to **steady** flows where the behavior does not change over time.

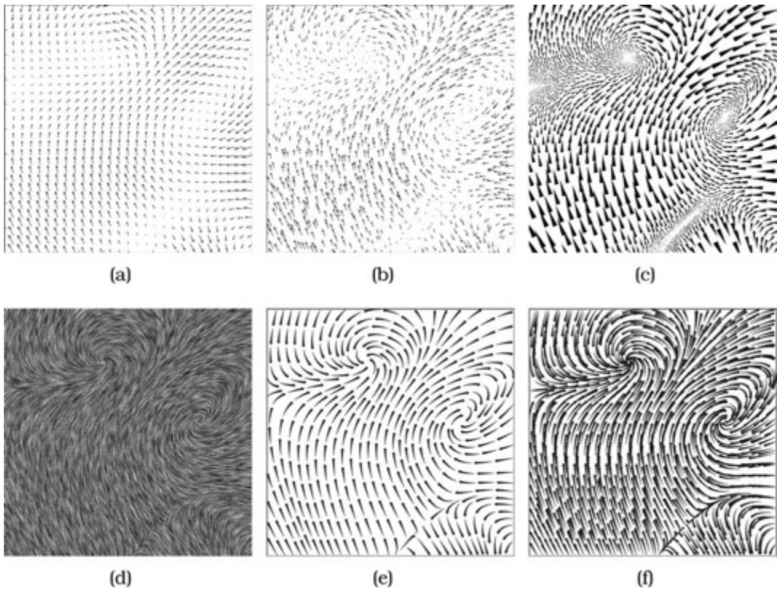
One of the features of interest in flows are the **critical points**, the points in a flow field where the velocity vanishes. They are classified by the behavior of the flow in their neighborhoods: the three main types are attracting **sources**, repelling **sinks**, and **saddle points** that attract from one direction and repel from another.[2] Also, sources and sinks may or may not have circulation around them.* Figure 8.7 shows these five types of critical points.

* In flow vis, a source or sink with no circulation around it is called a **node**, and one with circulation is called a **focus. I** avoid these overloaded terms; in this book, I reserve *node* and *link* for network data and *focus+context* for the family of idioms that embed such information together in a single view.

There are four major families of vector field spatial visual encoding idioms. The *flow glyph* idioms show local information at each cell. There are two major methods based on the derived data of tracing particle trajectories, either the *geometric flow* approach using a sparse set of seed points or the *texture flow* approach with a dense set of

seeds. The *feature flow* approach uses global computation across the entire field to explicitly detect features, and these derived features are usually visually encoded with glyphs or geometry. Finally, a vector field can be reduced to a scalar field, allowing any of the scalar field idioms covered in the previous section to be used, such as direct volume rendering or isocontouring.

**Figure 8.8. An empirical study compared human response to six different 2D flow vis idioms. (a) arrow glyphs on a regular grid. (b) arrow glyphs on a jittered grid. (c) triangular wedge glyphs inspired by oil painting strokes. (d) dense texture-based Line Integral Convolution (LIC). (e) curved arrow glyphs with image-guided streamline seeding. (f) curved arrow glyphs with regular grid streamline seeding.**




*From [Laidlaw et al. 05, Figure 1].*

Laidlaw et al. conducted an empirical study comparing six visual encoding idioms for 2D vector fields [Laidlaw et al. 05]. Figures 8.8(a), 8.8(b), and 8.8(c) show local glyph idioms, Figure 8.8(d) shows a dense texture idiom, and Figures 8.8(e) and 8.8(f) show geometric idioms. The three tasks considered were finding all of the critical points and identifying their types; identifying what type of critical point is at a specific location; and predicting where a particle starting at a specified point will end up being transported.* While none of the idioms outperformed all of the others for all tasks, the two local glyph idioms using arrows fared worst.

* The technical term for the transport of a particle within a fluid is **advection**.

### 8.5.1 Flow Glyphs



The **flow glyph** idioms show local information about a cell in the field using an object with internal substructure; one of the most basic choices is an arrow, as shown in Figure 8.8(a). An arrow glyph encodes magnitude with the length of the stem, direction with arrow orientation, and disambiguates directionality with the arrowhead on one side of the stem. In addition to the visual encoding of the glyphs themselves, another key design choice with this idiom is how many glyphs to show: a glyph for each cell in the field, or only a small subset. A limitation of glyph-based approaches is the problem of occlusion in 3D fields.

2 A fourth possible type is a *center* where the flow is perfectly circular, but this type is less important in practice.

### 8.5.2 Geometric Flow

The **geometric flow** idioms compute derived geometric data from the original field using trajectories computed from a sparse set of seed points and then directly show the derived geometry. One major algorithmic issue is how to compute the trajectories.* A crucial design choice is the seeding strategy: poor choices result in visual clutter and occlusion problems, but a well-chosen strategy supports inspection of both 2D and 3D fields. In the 3D case, geometric navigation is a useful interaction idiom that helps with the shape and structure understanding tasks.

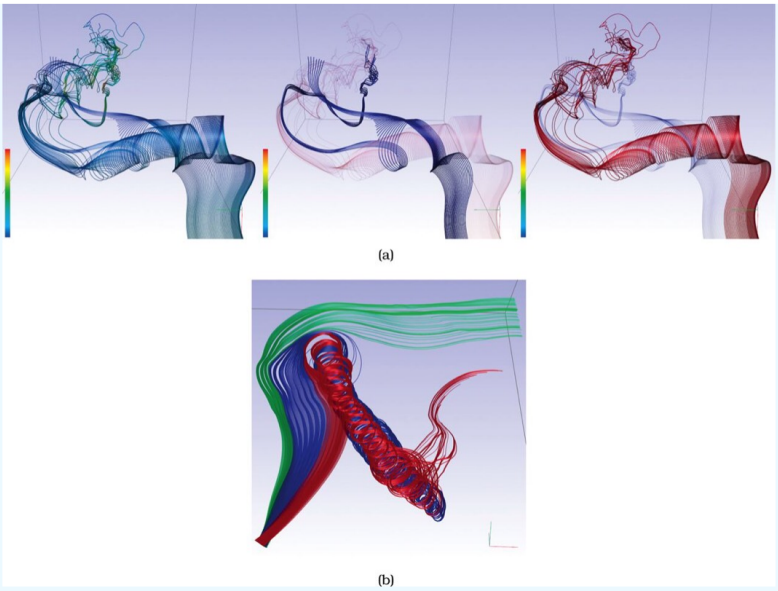
* The geometric approaches typically approximate using numerical integration, and so this idiom is sometimes called **integration-based flow**.

The geometric flow idioms are based on intuitions from physical experiments that can be conducted in real-world settings such as wind tunnels, and the simpler cases all have direct physical analogs. The trajectory that a specific particle will follow is called a **streamline** for a steady field and a **pathline** for an unsteady (time-varying) field. The physical analogy is the path that a single ball would follow as time passes. In contrast, a **streakline** traces all the particles that pass through a specific point in space; the analogy is a trail of smoke particles released at different times from the same spot. A **timeline** is formed by connecting a front of pathlines over time: the analogy is placing several balls at the same time at different locations along a curve, and tracing the path between them at a later time step. All of these geometric structures have counterparts one dimension higher, formed by seeding from a curve rather than from a single point: **stream surfaces, path surfaces, streak surfaces**. Similarly, **time surfaces** are a generalization that is formed by connecting particles released from a surface rather than a curve.

**Example: Similarity-Clustered Streamlines**

Figure 8.9 shows a seeding strategy for streamlines and pathlines based on a derived similarity measure, proposed by McLoughlin et al. [McLoughlin et al. 13]. First, the derived geometry data of streamlines or pathlines is computed from the original 3D vector field. A set of derived attributes is computed for each streamline or pathline: curvature, namely, the curve's deviation from a straight line; torsion, namely, how much the curve bends out of its plane; and tortuosity, namely, how twisted the curve is. These three attributes are combined with a complex algorithm to form a fourth derived attribute, the line's *signature*. These signatures are used to construct a similarity matrix, and that is in turn used to create a cluster hierarchy. The user can interactively filter which lines are seeded according to cluster membership so that as much detail as possible is preserved. The streamline or pathline spatial geometry is drawn in 3D. Each line is colored according to its cluster membership, and the user has interactive control of how many clusters to show. The user can also select a cluster to emphasize as a foreground layer with high opacity, where the others are drawn in low opacity to form a translucent background layer. Figure 8.9(a) shows three views: all of the streamlines at full opacity, the purple cluster emphasized, and the red cluster emphasized. Figure 8.9(b) shows an unsteady field, with three clusters of pathlines. The interaction idiom of geometric 3D navigation allows the user to rotate to any desired viewpoint.

**Figure 8.9. Geometric flow vis idioms showing a sparse set of particle trajectories, with seeding and coloring according to similarity. (a) Streamlines: all clusters equally opaque; purple cluster emphasized; red cluster emphasized. (b) Pathlines, colored by three clusters.**




*From [McLoughlin et al. 13, Figures 7 and 11c].*

► Filtering is covered in Section 13.3.

► Layering is covered in Section 12.5.

| Idiom | Similarity-Clustered Streamlines |
|---|---|
| **What: Data** | Spatial field: 3D vector field. |
| **What: Derived** | Geometry: streamlines or pathlines. |
| **What: Derived** | One attribute per streamline/pathline (signature). |
| **What: Derived** | Cluster hierarchy of streamlines/pathlines. |
| **How: Encode** | Use derived geometry of lines, color, and opacity according to cluster. |
| **Why: Tasks** | Find features, query shape. |
| **Scale** | Field: millions of samples. Geometry: hundreds of streamlines. |

### 8.5.3 Texture Flow

The **texture flow** idioms also rely on particle tracing, but with dense coverage across the entire field rather than from a carefully selected set of seed points.* They are most commonly used for 2D fields or fields on 2D surfaces. Figure 8.8(d) shows an example of the Line Integral Convolution (LIC) idiom, where white noise is smeared according to particle flow [Cabral and Leedom 93].

* The name of **texture** arises from a set of data structures and algorithms in computer graphics that efficiently manipulate high-resolution images without intermediate geometric representations; these operations are supported in hardware on modern machines.

### 8.5.4 Feature Flow

The **feature flow** vis idioms rely on global computations across the entire vector field to explicitly locate all instances of specific structures of interest, such as critical points, vortices, and shock waves.* The goal is to partition the field into subregions where the qualitative behavior is similar. The resulting derived data is then

directly visually encoded with one of the previously described flow idioms, for a geometric representation or a glyph showing each feature. In contrast, the previous idioms are intended to help the user to infer the existence of these structures, but they are not necessarily shown directly. A major challenge of feature-based flow vis is the algorithmic problem of computationally locating these structures efficiently and correctly.

* An alternative name for *feature-based flow* is **topological flow** vis.

## 8.6 Tensor Fields: Many Values

Flow vis is concerned with both vector and tensor data. Tensor fields typically contain a matrix at each cell in the field, capturing more complex structure than what can be expressed in a vector field.[3] Tensor fields can measure properties such as stress, conductivity, curvature, and diffusivity. One example of a tensor field is diffusion tensor data, where the extent to which the rate of water diffusion varies as a function of direction is measured with magnetic resonance imaging. This kind of medical imaging is often used to study the architecture of the human brain and find abnormalities.

All of the idiom families used for vector fields are also used for tensor fields: local *glyphs*, sparse *geometry*, dense *textures*, and explicitly derived *features*.

One major family of idioms for visually encoding tensor fields is **tensor glyphs**, where local information at cells in the field is shown by controlling the shape, orientation, and appearance of a base geometric shape. Just as with vector glyphs, another design choice is whether to show a glyph in all cells or only a carefully chosen subset. While the glyph idiom is the same fundamental design choice for both tensor and vector glyphs, tensor glyphs necessarily have a more complex geometric structure because they must encode more information.

**Example: Ellipsoid Tensor Glyphs**

Tensor quantities can be naturally decomposed into orientation and shape information; these quantities can be visually encoded with a 3D glyph.[4] A shape may be **isotropic**, where each direction is the same, or **anisotropic**, where there is a directional asymmetry. For diffusion in biological tissue, anisotropy occurs when the water moves through tissue faster in some directions than in others; Figure 8.10 shows a physical example of the 2D case where two different kinds of paper are stained with ink. There is isotropic diffusion through Kleenex, where the ink spreads at the same rate in all directions as shown in Figure 8.10(a), whereas the newspaper has a preferred direction where the ink moves faster with anisotropic diffusion as shown in Figure 8.10(b).

194
195

**Figure 8.10. 2D diffusion illustrated with ink and paper. (a) Isotropic Kleenex. (b) Anisotropic newspaper.**

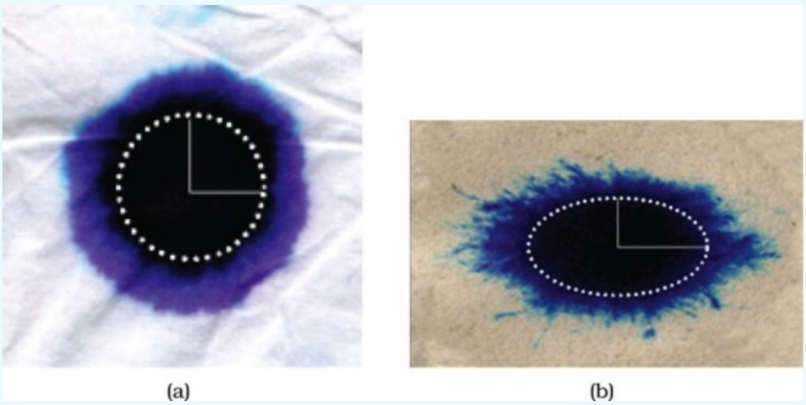
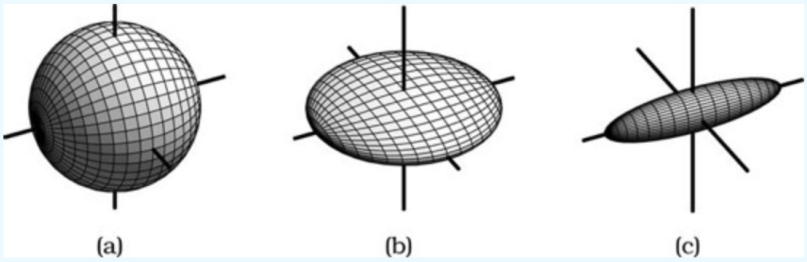


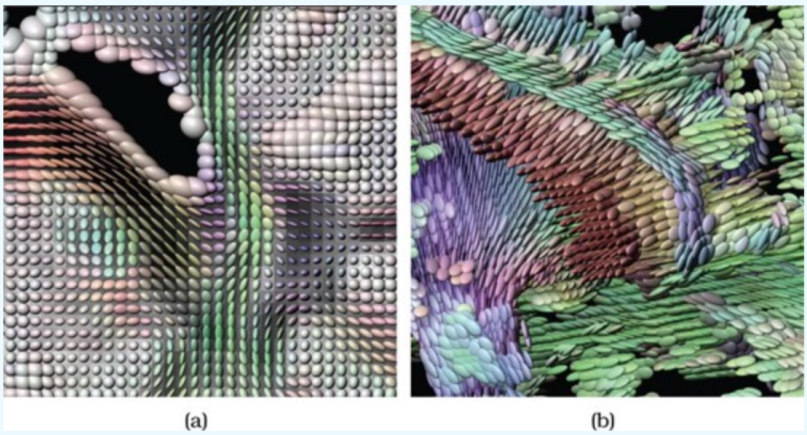
Figure 8.11 shows the three basic shapes that are possible in 3D. The fully isotropic case is a perfect sphere, as in Figure 8.11 (a); the partially anisotropic planar case is a sphere flattened in only one direction, as in Figure 8.11 (b); and the completely anisotropic linear case is flattened differently each of two directions to become a cigar-shaped ellipsoid, as in Figure 8.11(c). One way to encode this shape information in a 3D glyph is with an ellipsoid, where the direction that it points is an intuitive way to encode the orientation. Figure 8.12(a) shows using ellipsoid glyphs to inspect a 2D slice of a tensor field with the orientation attributes also used for coloring. In the 3D case shown in Figure 8.12(b), the isotropic glyphs are filtered out so that the anisotropic regions are visible.

**Figure 8.11. Ellipsoid glyphs can show three basic shapes. (a) Isotropic: sphere. (b) Partially anisotropic: planar. (c) Fully anisotropic: linear.**




*From [Kindlmann 04, Figure 1].*

**Figure 8.12. Ellipsoid glyphs show shape and orientation of tensors at each cell in a field. (a) 2D slice. (b) 3D field, with isotropic glyphs filtered out.**




*From [Kindlmann 04, Figures 10a and 11 a].*

Ellipsoid tensor glyphs have the weakness that different glyphs cannot be disambiguated from a single viewpoint; superquadric tensor glyphs are a more sophisticated approach that resolve this ambiguity [Kindlmann 04].

| Idiom | Ellipsoid Tensor Glyphs |
|---|---|
| **What: Data** | Spatial field: 3D tensor field. |
| **What: Derived** | Three quantitative attributes: tensor shape. Three vectors: tensor orientation. |
| **How: Encode** | Glyph showing six derived attributes, color and opacity according to cluster. |

The **geometric tensor flow** visual encoding idioms are based on similar intuitions as in the vector case, by computing sparse derived geometry such as hyperstreamlines or tensorlines; the same situation holds for the

**texture tensor flow** idioms. Similarly, **feature tensor flow** idioms explicitly detect features in tensor fields, where simpler cases that occur in vector fields are generalized to the more complex possibilities of tensor fields.

3 Mathematically, in the 3D case second-order tensors are 3 × 3 matrices that may be symmetric or nonsymmetric.

4 Mathematically, the shape information can be computed from the eigenvalues and the orientation from the eigenvectors.

## 8.7 Further Reading

**History** Thematic cartography, where statistical data is overlaid on geographic maps, blossomed in the 19th century. Choropleth maps, where shading is used to show a variable of interest, were introduced, as were dot maps and proportional symbol maps. The history of thematic cartography, including choropleth maps, is documented at the extensive web site http://www.datavis.ca/milestones [Friendly 08].

**Cartography** A more scholarly but still accessible historical review of thematic cartography is structured around the ideas of marks and channels [MacEachren 79]; MacEachren's full-length book contains a deep analysis of cartographic representation, visualization, and design with respect to both cognition and semiotics [MacEachren 95]. Slocum's textbook on cartography is a good general reference for the vis audience [Slocum et al. 08].

**Spatial Fields** One overview chapter covers a broad set of spatial field visual encoding and interaction idioms [Schroeder and Martin 05]; another covers isosurfaces and direct volume rendering in particular [Kaufman and Mueller 05].

**Isosurfaces** Edmond Halley presented isolines in 1686 and contour plots in 1701. The standard algorithm for creating isosurfaces is Marching Cubes, proposed in 1987 [Lorensen and Cline 87]; a survey covers some of the immense amount of followup work that has occurred since then [Newman and Yi 06]. Flexible isosurfaces are discussed in a paper [Carr et al. 04].

**Direct Volume Rendering** The *Real-Time Volume Graphics* book is an excellent springboard for further investigation of direct volume rendering [Engel et al. 06]. The foundational algorithm papers both appeared in 1988 from two independent sources: Pixar [Drebin et al. 88], and UNC Chapel Hill [Levoy 88]. The Simian system supports multidimensional transfer function construction [Kniss 02, Kniss et al. 05].

**Vector Fields** An overview chapter provides a good introduction to flow vis [Weiskopf and Erlebacher 05]. A series of state-of-the-art reports provide more detailed discussion of three flow vis idioms families: geometric [McLouglin et al. 10], texture based [Laramee et al. 04], and feature based [Post et al. 03]. The foundational algorithm for texture-based flow vis is Line Integral Convolution (LIC) [Cabral and Leedom 93].

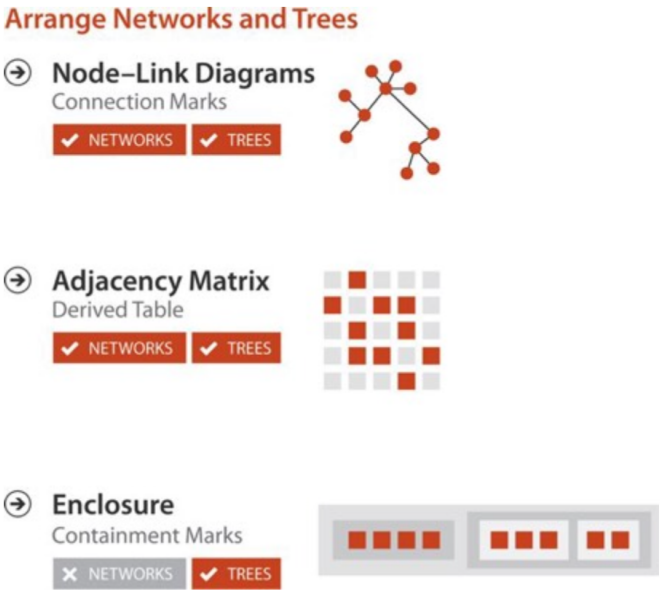
**Tensor Fields** The edited collection *Visualization and Processing of Tensor Fields* contains 25 chapters on different aspects of tensor field vis, providing a thorough overview [Weickert and Hagen 06]. One of these chapters is a good introduction to diffusion tensor imaging in particular [Vilanova et al. 06], including a comparison between ellipsoid tensor glyphs and superquadric tensor glyphs [Kindlmann 04].

# Chapter 9 Arrange Networks and Trees

## 9.1 The Big Picture

This chapter covers design choices for arranging network data in space, summarized in Figure 9.1. The node–link diagram family of visual encoding idioms uses the connection channel, where marks represent links rather than nodes. The second major family of network encoding idioms are matrix views that directly show adjacency relationships. Tree structure can be shown with the containment channel, where enclosing link marks show hierarchical relationships through nesting.

### Figure 9.1. Design choices for arranging networks.




## 9.2 Connection: Link Marks

The most common visual encoding idiom for tree and network data is with **node–link diagrams**, where nodes are drawn as point marks and the links connecting them are drawn as line marks. This idiom uses connection marks to indicate the relationships between items. Figure 9.2 shows two examples of trees laid out as node–link diagrams. Figure 9.2(a) shows a tiny tree of 24 nodes laid out with a triangular vertical node–link layout, with the root on the top and the leaves on the bottom. In addition to the connection marks, it uses vertical spatial position channel to show the depth in the tree. The horizontal spatial position of a node does not directly encode any attributes. It is an artifact of the layout algorithm's calculations to ensure maximum possible information density while guaranteeing that there are no edge crossings or node overlaps [Buchheim et al. 02].

Figure 9.2(b) shows a small tree of a few hundred nodes laid out with a spline radial layout. This layout uses essentially the same algorithm for density without overlap, but the visual encoding is radial rather than rectilinear: the depth of the tree is encoded as distance away from the center of the circle. Also, the links of the graph are drawn as smoothly curving **splines** rather than as straight lines.

Figure 9.3(a) shows a larger tree of 5161 nodes laid out as a rectangular horizontal node–link diagram, with the root on the left and the leaves stretching out to the right. The edges are colored with a purple to orange continuous colormap according to the Strahler centrality metric discussed in Section 3.7.2. The spatial layout is fundamentally the same as the triangular one, but from this zoomed-out position the edges within a subtree form a single