VECTOR VISUALIZATION
UNIT-3

Contents

Vector Visualization:

Vector Glyphs: Line glyphs, Cone and arrow glyphs, Vector glyphs in 2D

Vector Color Coding: Color coding on 2D surfaces, Displacement Plots

Texture-Based Vector Visualization: Line integral convolution

Domain- Modeling Techniques:

Cutting: Extracting a Brick, Slicing in Structured Datasets, Implicit Function Cutting, Generalized Cutting

Selection: Selecting cells, Thresholding, segmentation, and contouring,

Grid Construction from Scattered Points: Triangulation Methods- Delaunay triangulations, Voronoi diagrams

Vector Visualization

A vector is a tuple of n scalar components v = (v1,...,vn), vi ∈ R. An n-dimensional vector describes, for example, a position, direction, rate of change, or force in $R^{n.}$

Visualization software defines all vectors to have three components. 2D vectors are modelled as 3D vectors with the third (z) component equal to null

Computational Fluid Dynamics

Application for vector visualization is CFD:. CFD simulations are able to predict the time-dependent behavior of compressible 3D fluid flows interacting substances, or species, having different densities and pressures, over complex spatial geometries. The solution of a CFD simulation consists of several datasets, each for a different time step. For each time step, several attributes are computed and stored into the solution dataset, such as velocity, pressure, density, flow divergence, and vorticity.

Divergence: Given a vector field v : R3 → R3, the divergence of $v = (v_x, v_y, v_z)^1$ is the scalar quantity. If v is a flow field that transports mass, div v characterizes the increase or loss of mass at a given point p in the vector field in unit time

Vorticity: The vorticity vector characterizes the speed and direction of rotation of a given vector field at every point. Curl v

Given a vector field v : $R^3 \rightarrow R^3$, the vorticity of v, also called the curl or rotor of v2, is the vector quantity. A vortex is a region where the vector field locally circles around a point called the vortex center.

Vector Glyphs

- Vector Glyphs or vector icon: to visualize vector fields. Mapping technique associates with every sample point of vector dataset.

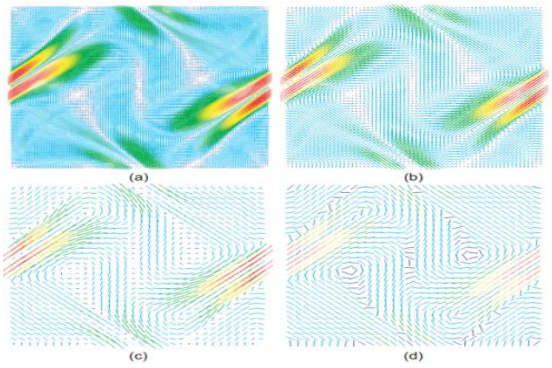
Properties of icon-vector attribute

- Location
- Direction
- Orientation
- Size and color
- Every Glyph is a sign conveys the properties represented by vector

Vector Glyphs: Line Glyphs

Variations of vector glyphs based on the:

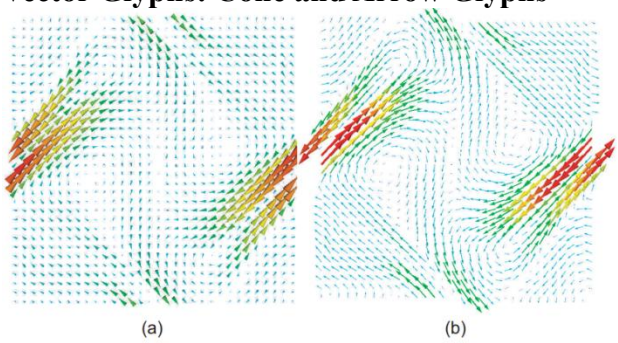
- How many number of glyphs can be displayed on screen area
- How many attributes can displayed per glyph.



Lines show the position, direction and magnitude of a set of vectors. Given dataset with a sampled domain D we associate line l=(x, x+kv(x)) with every sample point xϵD that has a vector attribute v(x) and k represents the scaling factor used to map the vector magnitude to the geometric domain. Oriented line glyphs with spiky appearance are called hedgehogs.

Line glyphs are scaled proportionally to the vector field magnitude, the scaling factor k being proportional to the subsampling rate. First, high-resolution vector datasets must be subsample to visualize hedgehogs. Comparing Figures (a), (b), and (c), the last image is easier to comprehend the vector field where all glyphs have the same, relatively large size are easily perceivable. Use a unique glyph scaling factor k so that all glyphs are locally smaller than the cell size.

Vector Glyphs: Cone and Arrow Glyphs



Complex shapes like cones and arrow can be used as glyphs, had an advantage which convey signed direction. Line glyphs convey unsigned direction. Takes more space to draw but require low resolution data set. Can encode more attributes than vector field. Used in situations where correlations between several scalar and vector fields-CFD Simulations. By shading the line glyph from full color at the glyph origin to the background color at the line tip, a visual effect similar to a thin arrow can be obtained without the need for extra screen space. A 3D CFD solution consisting of flow velocity, vorticity, divergence and material density, pressure, and temperature offers 3 + 3 + 1 + 1 + 1 + 1 = 10 attributes per dataset point.

Cone and arrow glyphs: Vector glyphs in 2D surfaces.

Consider a zoomed-in detail showing a hedgehog plot over a single cell of a 2D vector field.

The vector field variation over the displayed cell is quite small, the displayed arrow glyphs and arrive at the conclusion that the vector field has an upper-right direction and orientation, and increases in magnitude in this direction. Glyph techniques produce a purely discrete visualization. The task is made more difficult when we have to interpolate between directions and orientations, as in the case of vector glyphs.

Vector Color Coding

Develop dense visualizations for vector fields, similar to the color-mapped surfaces used for scalar fields. Similar to scalar color mapping, vector color coding associates a color with every point of a given surface on defined dataset. The color is used to encode the vector orientation and direction attributes. Colors in the HSV system can be visualized using a so-called color wheel,

Every distinct hue corresponds to a different angle of the color wheel: red is 0◦, magenta is 60◦, blue is 120◦, cyan is 180◦, green is 240◦, and yellow is 300◦. Saturation is represented as the distance from the wheel center to a given color point.Value is usually represented as a separate one-dimensional “luminance” parameter, since the color wheel can encode only two distinct parameters

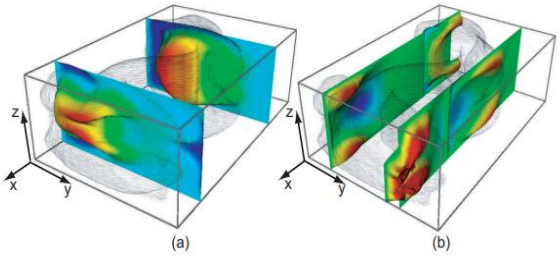
Color coding on 2D surfaces

Vector color coding for 2D vector fields proceeds by assuming a color wheel of unit radius. Under these conditions, every vector is represented by the color it points to, if we place it at the center of the color wheel. The vector orientation is encoded in the hue and the vector length in the value. The saturation parameter is set to one, i.e., we use only fully saturated colors. The color coding process is applied for every point of the dataset, similarly to the scalar color coding, either via texture or polygon color interpolation. Low-vector-magnitude regions can be easily detected as dark (low value) areas, whereas high-vector-magnitude regions show up as brightly colored areas. The inverse mapping from hue to vector orientation takes quite some time to be learned, so users have to be trained extensively to interpret such images. The directional color coding, we can also directly encode the vector components v_x, v_y, v_z into colors. In this setting, a 3D vector field is visualized by three separate scalar color-mapped fields. The user must visually correlate the same locations in three color images to get insight into the vector data at that location. Identify the location of the same spatial point in three different images, mentally performing three separate color-to-scalar mappings independently

Displacement Plots

The vector glyph with the origin at some point p can be seen as the trajectory p would follow in v(p) over a short time interval Δt. The vector glyph shows both the start and end points of the trajectory, i.e., p and p + v(p)Δt, respectively.

Displacement plots take a different approach by showing only the end points of such trajectories. Given a surface S ∈ D inside the domain D of a vector field, where S is discretized as a set of sample points p_i, a displacement plot of S is a new surface S` given by the set of sample points $p_i` = p_i + kv`(p_i)$



Natural interpretation:

The effect of displacing, or wrapping a given surface in the vector field are known as wrapped plots.

Parameter settings: Values that are too small, on the other hand, do not show the warping effect strongly enough so that it is recognizable in the visualization and it lets the users map it back visually to a displacement value.

Parameter settings

- First set the displacement factor k.
  - Values that are too large would warp the input surface too much, which can easily lead to self-intersecting surfaces. Large warp factors shift the displaced surface far away from its actual location.
  - Values too small do not show the warping effect strongly which recognizable in the visualization
- Second is the shape and position of the surface to be warped.
  - Planar surfaces choice is for displacement plots
  - geometric objects can be used to create displacement plots
  - The visual difference between the expected shape of the original object and the perceived (deformed) shape serves as a cue for the vector magnitude.

Texture-Based Vector Visualization

Discrete visualizations cannot convey information about every point of a given dataset domain- Create a texture signal that encodes the direction and magnitude of vector field in various parameters like luminance, graininess, color and pattern structure. To encode the vector direction in the texture parameters is to use the texture graininess.

Line integral convolution

Consider an input texture N consisting of small-scale black-and-white noise defined over the domain of the 2D vector field. For each pixel p of this domain, we trace a streamline S(p, s) upstream and downstream through p for some maximal distance L. Here, s parameterizes the streamline. Next, we set the value T (p) of the output texture T at the current location p to be the weighted sum of the values of the input noise texture N measured along the streamline S(p) that passes through p. As a weighting function k(s) : R → R+, we can use a Gaussian $k(s) = e^{-s^2}$, or other functions that are 1 at the origin and decay smoothly and symmetrically from the origin until they reach near-zero values at the maximal distance L. The obtained value T (p) is T (p) =

$$T(p) = \frac{\int_{-L}^{L} N(S(p,s))k(s)ds}{\int_{-L}^{L} k(s)ds}.$$

The denominator normalizes the weight factors for an arbitrary value of L .Applying this equation for all pixels using streamline lengths L of several pixels then we obtain a texture T whose pixel colors exhibit little variation among streamline and strong variation between neighbouring stream lines. Intuitively, we can think of this process as blurring, or filtering, the noise image along the streamlines with a set of filters k(s) that are aligned with the streamlines. The filtering operation can be seen as a convolution of the noise and filter functions N and k. Hence, this process is also known in the visualization field as line integral convolution.

Domain Modeling Techniques

Cutting: Extracting a Brick, Slicing in Structured Datasets, Implicit Function Cutting, Generalized Cutting
Selection: Selecting cells, Thresholding, segmentation, and contouring,
Grid Construction from Scattered Points: Triangulation Methods- Delaunay triangulations, Voronoi diagrams

Domain- Modeling Techniques

By domain-modeling techniques-operations on datasets that modify the sampling domain representation-grid.This modification does not change the reconstructed function, so the meaning of the data attributes stays the same, even though their internal representation may change. This can modify the actual values of the data attributes stored on a given grid-for example-resampling

Cutting

Cutting methods are domain-modeling techniques that map the data from a given source domain to a target subdomain.Consider some function f defined on a domain D is represented by a sampled "source" dataset $D_s = (\{p_i\}, \{c_i\}, \{f_i\}, \{\Phi_i\})$. Cutting the domain D with the domain D' means, resampling f from D to D'. This implies creating a new "target" dataset $D'_s = (\{p_i'\}, \{c'_i\}, \{f'_i\}, \{\Phi'_i\})$, $D'_s = (\{p_i'\}, \{c'_i\}, \{f'_i\}, \{\Phi'_i\})$. The grid points $\{p'_i\}$, cells $\{c'_i\}$, and interpolation functions $\{\Phi'_i\}$ of the target dataset are all user specified, the user to say where to resample the source dataset.The cutting operation has several properties.

First, the target domain is assumed to be a subset of the source domain. More exactly, we assume the points $\{p_i'\}$ of the target dataset to be contained in the cells $\{c_i'\}$ of the source dataset. Use convex cells in our datasets (see Section 3.4), this means that all cells $\{c'_i\}$ in the target dataset are also contained in the cells $\{c_i\}$ of the source dataset

A **second** property of cutting is that the dimensionality of the source and target datasets, and hence the interpolation functions Φ_i and Φ'_I of the two, need not be the same. $Target_{dim} <= Source_{dim}$

Cutting: Extracting a Brick:

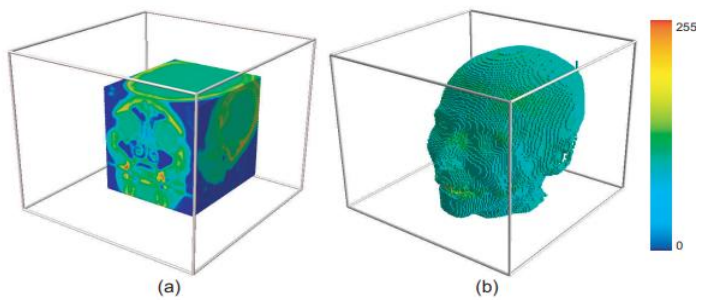
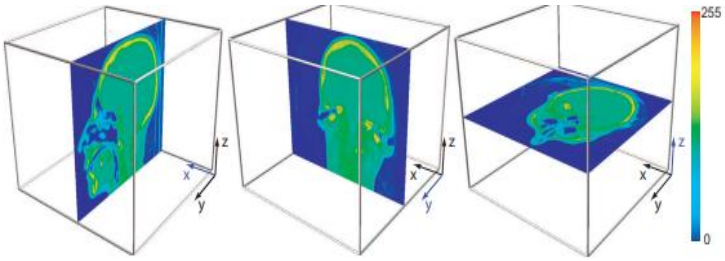


Figure 8.1. (a) Brick extraction. (b) Selection of cells with scalar value above 50.

Extracting a brick, also called bricking or extracting a volume of interest (VOI), is a cutting operation that produces a target dataset with the same dimensionality as the source dataset. The target grid points are a subset of the source grid points, $\{p'_i\} \in \{p_i\}$.Uniform, structured, and rectilinear grids arrange their sample points in a regular axis-aligned lattice where bricking takes

advantage to implement cutting operation. For a d-dimensional dataset, we can identify every sample point by d integers n1,...,nd, called structured coordinates. Hence, we can easily specify the target domain as an axis-aligned "brick" contained in the source dataset, defined by its minimum and maximum integer coordinates (m1, M1),...,(md, Md), where $1 < mi < Mi < ni$ for all $i \in [1, d]$. This set of structured coordinates is called the brick extent. Given a dataset that has uniform, structured and rectilinear grid that produce a new dataset that has the same grid type. In the target dataset, copy all points, cells and corresponding data attributes that fall within the specified brick extent.

Cutting: Slicing in Structured Datasets:



We define a slice as all grid points that have one of the structured integer coordinates $n_1,...,n_d$ equal. Extracting a slice can be seen as a bricking operation where the brick extent $(m1, M_1),...,(m_d, M_d)$ is equal to the grid extent for $d - 1$ of the dimensions, except for the slicing axis s, where $m_s = M_s$. Slicing a d-dimensional dataset generates a $d - 1$ dimensional dataset.

Cutting: Implicit Function Cutting

To cut an arbitrary dataset with a given lower-dimensional domain. A simple, yet powerful way to specify the cutting domain is to use implicit functions. Given some function $\varphi : D \rightarrow R$, where D is the domain of the source dataset, we define the target, or cutting, domain as all points $p \in D$ for which $\varphi(p) = 0$.

- First, we compute a scalar dataset D_{cut} that has the same grid as the source dataset and that evaluates φ.
- Second, we compute a contour of D_{cut} for the value zero

Implicit equation of a plane $Ax+By+ Cz+D = 0$ with appropriate coefficients A, B, C, and D. By changing the coefficients, we can obtain slice planes oriented at arbitrary angles

Cutting: Generalized Cutting

Resample the source dataset attributes on this unstructured grid, using one of the available forms of interpolation (e.g., constant or linear), and we obtain the desired result

The source attributes are interpolated at the locations of the target grid vertices (if the target dataset uses linear interpolation) or target grid cell centers (if the target dataset uses constant interpolation)

Selection

Selection methods extract the data from a source dataset based on data properties.

Cutting **enforces various geometrical and/or topological properties** on the target domain, since the target grid is specified by the user, but cannot explicitly enforce any properties on the data values, as these are fully specified by the source dataset. Selection explicitly specifies which data values **we are interested in**, but cannot enforce, a certain topology and/or geometry of the shape or connectivity of the extracted dataset domain. Selection produces just a set of sample points and/or cells from the source dataset for which the data-based **selection criterion** holds. The simplest variant of selection produces a domain D' that contains just the sample points whose data values meet the selection criterion $D' = \{p \in D | s(p) = true\}$. Here, $s : D \rightarrow B$ is a boolean function representing the user-specified selection operation based on the attributes of the point p

Selection: Selecting cells

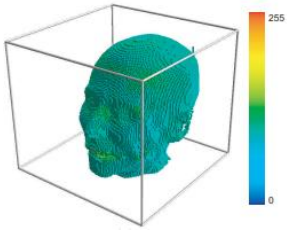
- To extract cells, there are several ways to apply the selection criterion on a cell.

- A cell can meet the selection criterion if one of its vertices, all vertices, or its center point meet the selection criterion as defined for a point.
- The one-vertex criterion produces more cells, essentially selecting cells that are neighbors of the ones produced by the all-vertex selection criterion.
- The center point criterion is equivalent to applying the one-point selection criterion on a slightly different sampling grid.
- If cells are selected in the output dataset, we assume these to have the same interpolation functions as in the input dataset, since we just copied them from the input dataset.
- The output dataset is assumed to have the same interpolation functions as the input dataset, since it is essentially just a subset of the input points and/or cells.
- Depending on what we want to select (points or cells), we iterate over all the input dataset's points or cells, apply the selection criterion, and copy the elements that pass the criterion to the output dataset, including their data attributes as well

Selection: Thresholding, segmentation

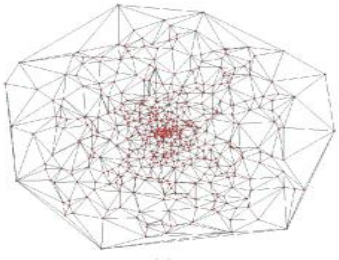
Selection based on the scalar value being larger or equal (or smaller or equal) than a given threshold s_0 produces one (or more, depending on the data monotonicity) compact subsets of the input dataset, also called threshold sets. Such an operation is also known as thresholding or segmentation. A variant of segmentation tests the scalar value against a given value range $[s_{min}, s_{max}]$. scalar values that represent the luminance, or intensity, of an image, selecting data points based on the derivative values is related to edge-detection methods. Selection methods that implement are essentially local methods, in the sense that they treat each point or cell of the dataset separately. Nonlocal selection function can be described as a function $s : D \rightarrow D$, where $s(D) = D' \subset D$ is the result of the selection applied on the domain D. Nonlocal selection operations occur when we must enforce the connectivity of the resulting domain D'."select all connected components $D'_i \subset D$ from an input domain D where the scalar values exceed some threshold s_{min} and whose size $|D'_i|$ exceeds some minimal size τ_{min}.

Selection: contouring



Selecting all cells in a dataset whose data values are equal to a given target value τ is conceptually equivalent to producing a piecewise constant approximation of the contour at value τ. The contour is approximated by a set of cells, which gives it the blocky appearance visible in Figure.The marching squares and marching cubes algorithms will compute the same isosurface, but use a piecewise linear approximation. The contour is approximated by a set of planes (in 3D) or lines (in 2D)

Grid Construction from Scattered Points: **Triangulation Methods- Delaunay triangulations:**



Triangulation methods are most-used class of methods for constructing grids from scattered points. Given a set of points p_i (sometimes also called sites), a triangulation method produces a grid (p_i, c_i) by generating a set of cells c_i that have the sample points pi as vertices. The cells c_i form a tiling of the convex hull of the point set $\{p_i\}$. In other words, triangulation methods produce a grid that samples a domain D identical to the convex hull of the triangulated point set.

Delaunay triangulations

This method generates triangular cells c_i for a set of 2D points $p_i \in R^2$ and tetrahedra for a set of 3D points $p_i \in R^3$.A Delaunay triangulation of a point set consists of a set of triangles that covers the convex hull of the point set. An important property of a Delaunay triangulation is that no point from the input point set $\{p_i\}$ lies in the circumscribed circle of any triangle in the triangulation. Triangulations that obey this property are called conforming Delaunay triangulations. We define piecewise linear basis functions over the triangles contained in the unstructured grid generated by the Delaunay triangulation, and use these functions to interpolate the vertex data values. The point density is higher in the center, which causes the creation of smaller triangles in that area.

Grid Construction from Scattered Points: Voronoi diagrams

For every Delaunay triangulation, there exists an associated geometric structure called a Voronoi diagram.A Voronoi diagram consists of a set of convex polygonal cells in 2D and polyhedral cells in 3D.The vertices of the Voronoi cells are the centers of the circumscribed circles of the triangles present in the associated Delaunay triangulation. The edges of the Voronoi cells are line segments contained in the lines perpendicular to, and passing through, the midpoints of the edges of the triangles present in the associated Delaunay triangulation. The centers of the Voronoi cells are the vertices of the Delaunay triangulation, i.e., the given scattered points.Every location x in a Voronoi diagram is included in the Voronoi cell that has as center. Voronoi diagrams can be used to quickly find the closest point p from a given scattered point set to a given test location x e closest point p in the input point set $\{p_i\}$.