Data Visualization (CSCI 627/490)

Isosurfacing & Volume Rendering

Dr. David Koop
Data Wrangling

• Problem 1: Visualizations need data
  • Solution: The Web!

• Problem 2: Data has extra information I don't need
  • Solution: Filter it

• Problem 3: Data is dirty
  • Solution: Clean it up

• Problem 4: Data isn't in the same place
  • Solution: Combine data from different sources

• Problem 5: Data isn't structured correctly
  • Solution: Reorder, map, and nest it
JavaScript Data Wrangling Resources

- Latest version: https://observablehq.com/@berkeleyvis/learn-js-data
- My old version: https://observablehq.com/@dakoop/learn-js-data
- Based on http://learnjsdata.com/
- Good coverage of data wrangling using JavaScript
Assignment 5

- Map of Citi Bike trips
  - Multiple Views
  - Linked Highlighting
  - Filtering
  - Aggregation

- Due Monday, Nov. 23
Projects

• Keep working on implementation
• Be creative, don't copy
• Think about interaction
• Presentations on the last day of class (Dec. 3)
  - Plan to use Blackboard
  - Upload to Blackboard beforehand in case of technical issues
Course Evaluations
Fields in Visualization

Scalar Fields
(Order-0 Tensor Fields)

Vector Fields
(Order-1 Tensor Fields)

Tensor Fields
(Order-2+)

Each point in space has an associated...

Scalar

\[ s_0 \]

Vector

\[
\begin{bmatrix}
  v_0 \\
  v_1 \\
  v_2
\end{bmatrix}
\]

Tensor

\[
\begin{bmatrix}
  \sigma_{00} & \sigma_{01} & \sigma_{02} \\
  \sigma_{10} & \sigma_{11} & \sigma_{12} \\
  \sigma_{20} & \sigma_{21} & \sigma_{22}
\end{bmatrix}
\]
Grids

- Remember we have continuous data and want to sample it in order to understand the **entire** domain
- Possible schemes?

- Geometry: the spatial positions of the data (points)
Grids

• Remember we have continuous data and want to sample it in order to understand the **entire** domain

• Possible schemes?

  - Geometry: the spatial positions of the data (points)
  - Topology: how the points are connected (cells)
  - Type of grid determines how much data needs to be stored for both geometry and topology

• **Grids (Meshes)**
  - Meshes combine positional information (geometry) with topological information (connectivity).
  - Mesh type can differ substantially depending on the way mesh cells are formed.

---

© Weiskopf/Machiraju/Möller
Visualizing Volume (3D) Data

- **2D visualization**
  - slice images (or multi-planar reformating MPR)

- **Indirect**
  - 3D visualization
  - isosurfaces (or surface-shaded display SSD)

- **Direct**
  - 3D visualization
  - (direct volume rendering DVR)
Data

• In this lecture, we will be considering **scalar** data: a single value at each point.
• Our data is always discrete, what is the value of a point not exactly on our grid?
• Need a method to determine what these values are...
Interpolation

Value at 2.2?
Nearest Neighbor Interpolation

Value at 2.2?
Linear Interpolation

Value at 2.2?
Interpolation

- Other schemes:
  - polynomial interpolation
  - splines
  - more…
Dimensions of Data

• 1-Dimension: data along a line
  - Example: temperature along my drive from Massachusetts to Illinois
• 2-Dimensional: data on a plane
  - Example: temperature on the surface of a pond
• 3-Dimensional: data in our normal world (data in a volume)
  - Example: temperature at every point in the room
• Complexity increases as we add dimensions
• Visualization complexity also increases
• Often, want to be able to see phenomena as we see them in real life settings
3D: Voxels and Cells

Voxel vs. Cell model

- Voxel: grid point in center, constant value in voxel
- Cell: grid points at vertices, value within cell varies

Visualizing Volume (3D) Data

- 2D visualization of slice images (or multi-planar reformatting MPR)
  - *Indirect* 3D visualization of isosurfaces (or surface-shaded display SSD)
  - *Direct* 3D visualization (direct volume rendering DVR)

[© Weiskopf/Machiraju/Möller]
Visualizing Volume (3D) Data

(a) An isosurfaced tooth.

(b) Multiple isosurfaces. [J. Kniss, 2002]
Visualizing Volume (3D) Data

(a) 2D slice  
(b) Volume Rendering

[J. Kniss, 2002]
Visualizing Volume (3D) Data

(a) 2D slice  
(b) Volume Rendering

[J. Kniss, 2002]
How have we encoded 3D scalar data before?

Hint: Think about elevation maps
Isolines (2D)

• Isoline: a line that has the same scalar value at all locations
• Example: Topographical Map
Isosurfaces (3D)

- Isosurface: a surface that has the same scalar value at all locations
- Often use multiple isosurfaces to show different levels
How?

• Given an **isovalue**, we want to draw the isocontours corresponding to that value
• Remember we only have values defined at grid points
• How do we get isolines or isosurfaces from that data?
• Can we use the ideas from interpolation?
Generating Isolines (Isovalue = 5)

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>9</th>
<th>7</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

[R. Wenger, 2013]
Generating Isolines
Generating Isolines

[Figure 2.4. (a) 2D scalar grid. (b) Black vertices are positive. Vertex \( v \) with scalar value \( s_v \) is positive if \( s_v > \alpha \) and negative if \( s_v < \beta \). Note that \( s_v = \alpha \) for one grid vertex. (c) Isocontour with vertices at edge midpoints (before linear interpolation). (d) Isocontour with isovalue 5.]

The isocontour lookup table, Table, contains sixteen entries, one for each configuration. Each entry, Table \([\kappa]\), is a list of the \( \mathbb{E}^+/-\kappa \) pairs.

In Figure 2.3 the isocontour edges are drawn connecting the midpoints of each square edge. This is for illustration purposes only. The geometric locations of the isocontour vertices are not defined by the lookup table.

The isocontour lookup table is constructed on the unit square with vertices \((0, 0), (1, 0), (0, 1), (1, 1)\). To construct the isocontour in grid square \((i, j)\), we have to map pairs of unit square edges to pairs of square \((i, j)\) edges. Each vertex \( v = (v_x, v_y)\) of the unit square maps to \( v + (i, j) = (v_x + i, v_y + j)\). Each edge \( e \) of the unit square with endpoints \((v, v')\) maps to edge \( e + (i, j) = (v + (i, j), v' + (i, j))\). Finally, each edge pair \((e_1, e_2)\) maps to \((e_1 + (i, j), e_2 + (i, j))\).

The endpoints of the isocontour edges are the isocontour vertices. To map each isocontour edge to a geometric line segment, we use linear interpolation to generate isolines.
Generating Isolines

(a) Scalar grid. (b) The +/- grid. (c) Midpoint vertices. (d) Isocontour.

Figure 2.4. (a) 2D scalar grid. (b) Black vertices are positive. Vertex $v$ with scalar value $s_v$ is positive if $s_v > 5$ and negative if $s_v < 5$. Note that $s_v = 5$ for one grid vertex. (c) Isocontour with vertices at edge midpoints (before linear interpolation). (d) Isocontour with isovalue 5.

The isocontour lookup table, Table, contains sixteen entries, one for each configuration. Each entry, Table[$\kappa$] is a list of the $E^+/E^-$ pairs.

In Figure 2.3 the isocontour edges are drawn connecting the midpoints of each square edge. This is for illustration purposes only. The geometric locations of the isocontour vertices are not defined by the lookup table.

The isocontour lookup table is constructed on the unit square with vertices $(0,0)$, $(1,0)$, $(0,1)$, $(1,1)$. To construct the isocontour in grid square $(i,j)$, we have to map pairs of unit square edges to pairs of square edges. Each vertex $v = (v_x, v_y)$ of the unit square maps to $v + (i,j) = (v_x + i, v_y + j)$. Each edge $e$ of the unit square with endpoints $(v, v')$ maps to edge $e + (i,j) = (v + (i,j), v' + (i,j))$. Finally, each edge pair $(e_1, e_2)$ maps to $(e_1 + (i,j), e_2 + (i,j))$.

The endpoints of the isocontour edges are the isocontour vertices. To map each isocontour edge to a geometric line segment, we use linear interpolation to...
Marching Squares

2. Marching Cubes and Variants

Figure 2.10. Red, positive regions and blue, negative regions for each square configuration. The green isocontour is part of the positive region. Black vertices are positive.

Proof of Properties 1 & 2:
The Marching Squares isocontour consists of a finite set of line segments, so it is piecewise linear. These line segments intersect only at their endpoints and thus form a triangulation of the isocontour. The endpoints of these line segments lie on the grid edges, confirming Property 2. □

Property 3. The isocontour intersects every bipolar grid edge at exactly one point.

Property 4. The isocontour does not intersect any negative or strictly positive grid edges.

Proof of Properties 3 & 4:
Each isocontour edge is contained in a grid square. Since the grid squares are convex, only isocontour edges with endpoints (vertices) on the grid edge intersect the grid edge. If the grid edge has one positive and one negative endpoint, the unique location of the isocontour vertex on the grid edge is determined by linear interpolation. Thus the isocontour intersects a bipolar grid edge at only one point.

If the grid edge is negative or strictly positive, then no isocontour vertex lies on the grid edge. Thus the isocontour does not intersect negative or strictly positive grid edges. □

Within each grid square the isocontour partitions the grid square into two regions. Let the positive region for a grid square \( c \) be the set of points which can be reached by a path \( \zeta \) from a positive vertex. More precisely, a point \( p \) is in the positive region of \( c \) if there is some path \( \zeta \subset c \) connecting \( p \) to a positive vertex of \( c \) such that the interior of \( \zeta \) does not intersect the isocontour. A point \( p \) is in the negative region of \( c \) if there is some path \( \zeta \subset c \) connecting \( p \) to a negative vertex of \( c \) such that \( \zeta \) does not intersect the isocontour. Since any path \( \zeta \subset c \) from a positive to a negative vertex must intersect the isocontour, the positive and negative regions form a partition of the square \( c \). Figure 2.10 illustrates the positive and negative regions, colored red and blue, respectively, for each square configuration.

[R. Wenger, 2013]
Ambiguous Configurations

- There are some cases for which we cannot tell which way to draw the isolines…
Ambiguous Configurations

- Either works for marching squares, this isn't the case for 3D

![Diagrams showing ambiguous configurations in 2D and 3D.](image-url)
3D: Marching Cubes

- Same idea, more cases [Lorensen and Cline, 1987]
Incompatible Choices

- If we have ambiguous cases where we choose differently for each cell, the surfaces will not match up correctly—there are holes.
- Fix with the asymptotic decider [Nielson and Hamann, 1991]
Marching Cubes Algorithm

• For each cell:
  - Classify each vertex as inside or outside ($\geq$, $<$) — 0 or 1
  - Take the eight vertex classifications as a bit string
  - Use the bit string as a lookup into a table to get edges
  - Interpolate to get actual edge locations
  - Compute gradients
  - Resolve ambiguities
• Render a bunch of triangles: easy for graphics cards
Multiple Isosurfaces

- Topographical maps have multiple isolines to show elevation trends
- Problem in 3D? **Occlusion**
- Solution? Transparent surfaces
- Issues:
  - Think about color in order to make each surface visible
  - Compositing: how do colors "add up" with multiple surfaces
  - How to determine good isovalues?

[J. Kniss, 2002]
Volume Rendering
Volume Rendering vs. Isosurfacing

(a) Direct volume rendered  (b) Isosurface rendered

[Kindlmann, 1998]
(Direct) Volume Rendering

- Isosurfacing: compute a surface (triangles) and use standard computer graphics to render the triangles
- Volume rendering: compute the pixels shown directly from the volume information
- Why?
  - No need to figure out precise isosurface boundaries
  - Can work better for data with noise or uncertainty
  - Greater control over appearance based on values
Types of Volume Rendering Algorithms

- Ray casting
  - Similar to ray tracing, but use rays from the viewer

- Splatting:
  - Object-order, voxels splat onto the image plane

- Shear Warp:
  - Object-space, slice-based, parallel viewing rays

- Texture-Based:
  - 2D Slices: stack of texture maps
  - 3D Textures

[via Möller]
Volume Ray Casting

Image Plane

Data Set

Eye

[Levine]
Volume Ray Casting

Image Plane

Eye

Data Set

[Levine]
**How?**

- Approximate volume rendering integral: light absorption & emission
- Sample at regular intervals along each ray
- Trilinear interpolation: linear interpolation along each axes (x,y,z)

- Not the only possibility, also "object order" techniques like splatting or texture-based and combinations like shear-warp
Compositing

• Need **one pixel** from all values along the ray

• Q: How do we "add up" all of those values along the ray?

• A: Compositing!

• Different types of compositing
  - First: like isosurfacing, first intersection at a certain intensity
  - Max intensity: choose highest val
  - Average: mean intensity (density, like x-rays)
  - Accumulate: each voxel has some contribution

[Levine and Weiskopf/Machiraju/Möller]
Volume Ray Casting

Image Plane

Data Set

Eye

[Levine]
Volume Ray Casting

Image Plane

Eye

Data Set

[Levine]
Compositing

- Need **one pixel** from all values along the ray
- Q: How do we "add up" all of those values along the ray?
- A: Compositing!
- Different types of compositing
  - First: like isosurfacing, first intersection at a certain intensity
  - Max intensity: choose highest val
  - Average: mean intensity (density, like x-rays)
  - Accumulate: each voxel has some contribution

[Levine and Weiskopf/Machiraju/Möller]
Types of Compositing

- max intensity
- accumulate
- average
- first

Intensity vs. depth

[Levine and Weiskopf/Machiraju/Möller]
Types of Compositing

- max intensity
- accumulate
- average
- first

[Levine and Weiskopf/Machiraju/Möller]
Types of Compositing

max intensity

accumulate

average

first

depth

[Levine and Weiskopf/Machiraju/Möller]
Types of Compositing

- max intensity
- accumulate
- average
- first

D. Koop, CSCI 627/490, Fall 2020

[Levine and Weiskopf/Machiraju/Möller]
Accumulation

• If we're not just calculating a single number (max, average) or a position (first), how do we determine the accumulation?

• Assume each value has an associated color (c) and opacity (α)

• Over operator (back-to-front):
  \[
  c = α_f \cdot c_f + (1-α_f) \cdot α_b \cdot c_b
  \]
  \[
  α = α_f + (1-α_f) \cdot α_b
  \]

• Order is important!

Blue Last

Blue First
Transfer Functions

- Where do the colors and opacities come from?
- Idea is that each voxel emits/absorbs light based on its scalar value
- ...but users get to choose how that happens
- x-axis: color region definitions, y-axis: opacity
Transfer Function Design

• Transfer function **design** is non-trivial!
• Lots of tools to help visualization designers to create good transfer functions
• Histograms, more attributes than just value like gradient magnitude
Multidimensional Transfer Functions
Multidimensional Transfer Functions