Isosurfaces

Lecture 7

February 6, 2020
7. Isosurfaces

Feb 6, 2020
Isosurfaces In Scientific Visualization
Isosurfaces In Scientific Visualization
Properties of Isocontours

Pre-image of scalar value
- Applicable in any dimension
- Manifolds of codimension 1

Closed (except at boundaries)

Nested: different values do not cross
- Can consider the zero-set case:
  \[ f(x, y) = k \iff f(x, y) - k = 0 \]

Normals given by gradient vector of \( f (\nabla f) \)
Contours in 2D

Assign geometric primitives (line segments) to individual cells (process one cell at a time)
Contours in 2D

Assign geometric primitives (line segments) to individual cells (process one cell at a time)

Consider sign of the values at vertices
Contours in 2D

Assign geometric primitives (line segments) to individual cells (process one cell at a time)

Consider sign of the values at vertices

Intersections occur on edges with sign change
Contours in 2D

Assign geometric primitives (line segments) to individual cells (process one cell at a time)

Consider sign of the values at vertices

Intersections occur on edges with sign change

Determine exact position of intersection
Contours in 2D

Assign geometric primitives (line segments) to individual cells (process one cell at a time)

Consider sign of the values at vertices

Intersections occur on edges with sign change

Determine exact position of intersection

Interpolate along grid edges
Contours in 2D

Idea: primitives must cross every grid line connecting two grid points of opposite sign

- Get cell
- Identify grid lines w/cross
- Find crossings
- Interpolate along grid lines
- Primitives naturally chain together
Questions

How many grid lines with crossings can there be?

What are the different configurations (adjacencies) of +/- grid points?
## Cases

<table>
<thead>
<tr>
<th>Case</th>
<th>Polarity</th>
<th>Rotation</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No crossing</td>
<td>x2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Singlet</td>
<td>x2</td>
<td>x4</td>
<td>8</td>
</tr>
<tr>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double adj</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double opp</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16 = 2^4
Ambiguities

How to form the lines?

- Red circle with an X
- Green circle with an X
- Red circle with a blank
- Green circle with a blank
- Red circle with an X
- Green circle with an X
Ambiguities

Right or wrong?
Isosurfacing

Goal: given a big 3D block of numbers ("scalars"), create a picture
Slicing shows data, but not its 3D shape
Solution: Isosurfacing
A Little Math...

Scalar volume:  
\[ f : D \subset \mathbb{R}^3 \to \mathbb{R} \]
\[ (x, y, z) \mapsto f(x, y, z) \]

Want to find  
\[ S_v = \{(x, y, z) | f(x, y, z) = v\} \]

= "All the locations where the value of \( f \) is \( v \)"

\( S_v \): isosurface of \( f \) at \( v \)

In 2D: isocontours (some path)

In 3D: isosurface

Why are isosurfaces useful?
Notations

Volume of data

Each voxel transformed to unit cube
Trilinear Interpolation

(Recall Lecture 4…) 

In a cuboid (axis parallel)

general formula

\[ \phi(x, y, z) = axyz + bxy + cxz + dyz + ex + fy + gz + h \]

with local coordinates

\[
\begin{align*}
P &= P_1 + u(P_2 - P_1) + v(P_4 - P_1) + w(P_5 - P_1) \\
&\quad + uv(P_1 - P_2 + P_3 - P_4) \\
&\quad + uw(P_1 - P_2 + P_6 - P_5) \\
&\quad + vw(P_1 - P_4 + P_8 - P_5) \\
&\quad + uvw(P_1 - P_2 + P_3 - P_4 + P_5 - P_6 + P_7 - P_8)
\end{align*}
\]
Isosurfacing
Increasing the Threshold
Isosurface Construction

For simplicity, we shall work with zero level isosurface, and denote positive vertices as \( \bullet \).

There are 8 vertices, each can be positive or negative - so there are \( 2^8 = 256 \) different cases.
Straightforward Cases

There is no portion of the isosurface inside the cube!

(Convexity)
Isosurface Construction

One Positive Vertex

Intersections with edges found by inverse linear interpolation (as in contouring)
Inverse Linear Interpolation

The linear interpolation formula gives value of $f$ at specified point $t$:

$$f^* = (1 - t)f_1 + tf_2$$

Inverse linear interpolation gives value of $t$ at which $f$ takes a specified value $f^*$

$$t = \frac{f^* - f_1}{f_2 - f_1}$$
Joining edge intersections across faces forms a triangle as part of the isosurface.
Isosurface Construction

Two Positive Vertices at Opposite Vertices
Isosurface Construction

One can work through all 256 cases in this way - although it quickly becomes apparent that many cases are similar.

For example:

- 2 cases where all are positive, or all negative, give no isosurface
- 16 cases where one vertex has opposite sign from all the rest

In fact, there are only 15 topologically distinct configurations
Canonical Cases

The 256 possible configurations can be grouped into these 15 canonical cases on the basis of complementarity (swapping positive and negative) and rotational symmetry.

The advantage of doing this is for ease of implementation - we just need to code 15 cases not 256.
Isosurface Construction

In some configurations, just one triangle forms the isosurface

In other configurations ...

...there can be several triangles

...or a polygon with 4, 5 or 6 points which can be triangulated

A software implementation will have separate code for each configuration
Marching Cubes Algorithm

Step 1: Classify the eight vertices relative to the isosurface value

8-bit index: 1 (+); 0 (-)

Code identifies edges intersected:
V1V4; V1V5; V2V3; V2V6; V5V8; V7V8; V4V8
Marching Cubes Algorithm

Step 2: Look up table which identifies the canonical configuration

For example:

- 00000000 Configuration 0
- 10000000 Configuration 1
- 01000000 Configuration 1

256 entries in table

- ... 11000001 Configuration 6
- ... 11111111 Configuration 0
Marching Cubes Algorithm

Step 3: Inverse linear interpolation along the identified edges will locate the intersection points

Step 4: The canonical configuration will determine how the pieces of the isosurface are created (0, 1, 2, 3 or 4 triangles)

Step 5: Pass triangles to renderer for display

Algorithm marches from cube to cube between slices, and then from slice to slice to produce a smoothly triangulated surface
Case 12

Three positive vertices on the bottom plane; and one positive vertex on the top plane, directly above the single negative on the bottom plane.

Solution…?
Three positive vertices on the bottom plane; and one positive vertex on the top plane, directly above the single negative on the bottom plane.

Solution…?
Marching Cubes Algorithm

Advantages

isosurfaces good for extracting boundary layers
surface defined as triangles in 3D - well-known
rendering techniques available for lighting, shading and viewing ... with hardware support

Disadvantages

shows only a slice of data
topological ambiguities?
Topological Ambiguities

Marching cubes suffers from exactly the same problems that we saw in contouring.

Case 3: Triangles are chosen to slice off the positive vertices - but could they have been drawn another way?
Ambiguities on Faces

On the front face, we have exactly the same ambiguity problem we had with contouring.

We can determine which pair of intersections to connect by looking at value at saddle point ($\nabla f = 0$).
Ambiguities on Faces

Trouble occurs because:

trilinear interpolant is only linear along the edges on a face, it becomes a bilinear function ... and for correct topology we must join the correct pair of intersections

Case 3 has two triangle pieces cutting off corners!
Ambiguities on Faces

Trouble occurs because:

trilinear interpolant is only linear along the edges on a face, it becomes a bilinear function ... and for correct topology we must join the correct pair of intersections

Case 3 has two triangle pieces cutting off corners!

.. but here is another interpretation!

6 configurations include ambiguous faces
Holes in Isosurfaces

Because of the ambiguity, early implementations which did not allow for this could leave holes ("cracks") where cells join.

Cases 12 and 3 in adjoining cells can cause holes.
Resolving the Face Ambiguities

Using the saddle point method to determine the correct behavior on a face generates sub-cases for each of the 6 ambiguous configurations which sub-case is chosen depends on the value of the saddle-point on the face note that some configurations have several ambiguous faces so many subcases arise - e.g., see config 13! if we do not extend the 15 cases there is chance of holes appearing in surface
Trilinear Interpolant

The trilinear function:

\[ f(x,y,z) = f_{000}(1-x)(1-y)(1-z) + f_{100}x(1-y)(1-z) + f_{010}(1-x)y(1-z) + f_{001}(1-x)(1-y)z + f_{110}xy(1-z) + f_{101}x(1-y)z + f_{011}(1-x)yz + f_{111}xyz \]

is deceptively complex!

For example, the isosurface of

\[ f(x,y,z) = 0 \]

is a cubic surface
Accurate Isosurface of Trilinear Interpolant

True isosurface of a trilinear function is a curved surface

cf. contouring where contours are hyperbola

We are in fact approximating by the triangles shown
Interior Ambiguities

In some cases there can also be ambiguities in the interior.

Consider case where opposite corners are positive.

Two possibilities: separated or tunnel.
Resolving the Interior Ambiguity

Decided by value at body saddle point

\[ \nabla f = 0 \text{ i.e., } \frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = \frac{\partial f}{\partial z} = 0 \]

Negative: two separate shells
Positive: tunnel
Marching Tetrahedra

As in contouring, another solution is to divide into simpler shapes - here they are tetrahedra.

24 tetrahedra in all

Value at centre = average of vertex values

Fit linear function in each tetrahedron:

\[ f(x,y,z) = a + bx + cy + dz \]

Isosurface of linear function is triangle.
Marching Tetrahedra

A disadvantage of the ‘24’ marching tetrahedra is the large number of triangles which are created - slowing down the rendering time.

There are versions that just use 5 tetrahedra.
References

Original marching cubes algorithm
   Lorensen and Cline (1987)

Face ambiguities
   Nielson and Hamann (1992)

Interior ambiguities
   Chernyaev (1995)

Accurate marching cubes
   Lopes and Brodlie (2003)
The Span Space

Livnat, Shen, Johnson 1996

**Given:** Data cells in 8D

**Past (active list):** Intervals in a 1D Value space

**New:**
- Points in the 2D Span Space
- Benefit: Points do not exhibit any spatial relationships
The Span Space

Search

Find all the points
minimum < isovalue
isovalue < maximum
Semi-infinite area
Quadrant

min = max

Isovalue
Minimum
The Span Space

Search for rectangles using Kd-tree

$O(n \log(n))$ to build

Search Complexity

$O(\sqrt{n+k})$

Recursively divide each axis along median
Interval Tree

Cignoni et al, 1997