3D Modeling
Implicit Surfaces

Shandong University
Spring 2013
3D Object Representations

• **Raw data**
  – Point cloud
  – Range image
  – Polygon soup

• **Surfaces**
  – Mesh
  – Subdivision
  – Parametric
  – Implicit

• **Solids**
  – Voxels
  – BSP tree
  – CSG
  – Sweep

• **High-level structures**
  – Scene graph
  – Skeleton
  – Application specific
Implicit Surfaces

- Represent surface with function over all space
Implicit Surfaces

- Surface defined implicitly by function
Implicit Surfaces

- Surface defined implicitly by function:
  - $f(x, y, z) = 0$ (on surface)
  - $f(x, y, z) < 0$ (inside)
  - $f(x, y, z) > 0$ (outside)
Implicit Surfaces

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

Inside:
\[ f(x,y,z) < 0 \]

Outside:
\[ f(x,y,z) > 0 \]
Implicit Surfaces

- Normals defined by partial derivatives
  - normal(x, y, z) = (df/dx, df/dy, df/dz)
(1) Efficient check for whether point is inside

- Evaluate $f(x,y,z)$ to see if point is inside/outside/on

\[
\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 - 1 = 0
\]
Implicit Surface Properties

(2) Efficient surface intersections

– Substitute to find intersections

Ray: \( P = P_0 + tV \)

Sphere: \( |P - O|^2 - r^2 = 0 \)

Substituting for \( P \), we get:

\[ |P_0 + tV - O|^2 - r^2 = 0 \]

Solve quadratic equation:

\[ at^2 + bt + c = 0 \]

where:

\[ a = 1 \]
\[ b = 2V \cdot (P_0 - O) \]
\[ c = |P_0 - C|^2 - r^2 = 0 \]
(3) Efficient boolean operations (CSG)

- Union, difference, intersect
Implicit Surface Properties

(3) Efficient boolean operations (CSG)

- Union, difference, intersect
Implicit Surface Properties

(4) Efficient topology changes
   – Surface is not represented explicitly!
Comparison to Parametric Surfaces

- **Implicit**
  - Efficient intersections & topology changes
- **Parametric**
  - Efficient “marching” along surface & rendering

**Equations:**

- Equiangular parametric (transcendental trigonometric)
  \[ p = (\cos(\alpha), \sin(\alpha)), \alpha \in [0, 2\pi] \]

- Non-equiangular parametric (rational)
  \[ p = (\pm(1-t^2)/(1+t^2), 2t/(1+t^2)), t \in [-1, 1] \]

- Implicit
  \[ p_x^2 + p_y^2 - 1 = 0 \]
Implicit Surface Representations

• How do we define implicit function?
  – Algebraics
  – Voxels
  – Basis functions
Algebraic Surfaces

• Implicit function is polynomial
  \[ f(x,y,z) = ax^d + by^d + cz^d + dx^{d-1}y + ex^{d-1}z + fy^{d-1}x + \ldots \]

\[
\left( \frac{x}{r_x} \right)^2 + \left( \frac{y}{r_y} \right)^2 + \left( \frac{z}{r_z} \right)^2 - 1 = 0
\]
Algebraic Surfaces

• Most common form: quadrics
  \[ f(x,y,z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k \]

• Examples
  – Ellipsoid (Sphere)
  – Paraboloid
  – Hyperboloid
Algebraic Surfaces

- Most common form: quadrics
  \[ f(x,y,z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k \]

- Examples

\[
\left( \frac{x}{r_x} \right)^2 + \left( \frac{y}{r_y} \right)^2 + \left( \frac{z}{r_z} \right)^2 - 1 = 0
\]

Ellipsoids

Image courtesy of http://www.geom.uiuc.edu/
Algebraic Surfaces

• Most common form: quadrics
  \[ f(x,y,z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k \]

• Examples

\[ \left( \frac{x}{r_x} \right)^2 + \left( \frac{y}{r_y} \right)^2 - \left( \frac{z}{r_z} \right)^2 \pm 1 = 0 \]

Hyperboloids

Image courtesy of http://www.geom.uiuc.edu/
Algebraic Surfaces

• Most common form: quadrics
  \[ f(x,y,z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k \]

• Examples

Paraboloids

Image courtesy of http://www.geom.uiuc.edu/
Algebraic Surfaces

- Higher degree algebraics

Cubic

Quartic

Degree six
• Intersection
  – Intersection of degree \( m \) and \( n \) algebraic surfaces yields curve with degree \( mn \)
Algebraic Surfaces

- Function extends to infinity
  - Must trim to get desired patch (this is difficult!)
Implicit Surface Representations

• How do we define implicit function?
  – Algebraics
  – Voxels
  – Basis functions
Voxels

- Regular array of 3D samples (like image)
  - Samples are called voxels ("volume pixels")
Voxels

- Example CT Data
Voxels

- Example isosurfaces

SUNY Stony Brook

Princeton University
Voxels

- Regular array of 3D samples (like image)
  - Apply reconstruction filter to determine \( f(x,y,z) \)
  - Isosurface at \( f(x,y,z) = 0 \) defines surface
Voxels

- Iso-surface extraction algorithm
  - e.g., Marching cubes
Voxels

- Iso-surface extraction algorithm
  - e.g., Marching cubes (15 cases)
Voxel Storage

- $O(n^3)$ storage for $n \times n \times n$ grid
  - 1 billion voxels for 1000 x 1000 x 1000
Implicit Surface Representations

• How do we define implicit function?
  – Algebraics
  – Voxels
  – Basis functions
Basis Functions

- Implicit function is sum of basis functions
- Example:

\[ f(P) = a_0 e^{-b_0 d(P,P_0)^2} + a_1 e^{-b_1 d(P,P_1)^2} \]
Implicit Surface Summary

Advantages:
- Easy to test if point is on surface
- Easy to compute intersections/unions/differences
- Easy to handle topological changes

Disadvantages:
- Hard to describe sharp features
- Hard to enumerate points on surface
  » Slow rendering
<table>
<thead>
<tr>
<th>Feature</th>
<th>Polygon Mesh</th>
<th>Subdivision Surface</th>
<th>Parametric Surface</th>
<th>Implicit Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Concise</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Intuitive specification</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Local support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Affine invariant</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Arbitrary topology</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Guaranteed continuity</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Natural parameterization</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Efficient display</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Efficient intersections</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
For simplicity, we shall work with zero level isosurface, and denote positive vertices as

There are EIGHT vertices, each can be positive or negative - so there are $2^8 = 256$ different cases!
These two are easy...

There is no portion of the isosurface inside the cube!
Intersections with edges found by inverse linear interpolation
(as in contouring)
Joining edge intersections across faces forms a triangle as part of the isosurface.
Isosurface Construction - Positive Vertices at Opposite Corners
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
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
• The classic algorithm is called Marching Cubes – find isosurface in one cube, then next and so on, marching from cube to cube
Isosurfacing by 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
  – 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?
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

As in 2D case, this removes ambiguities…

Cost is greater number of triangles which increases rendering time