Solid Modeling

Michael Kazhdan

(601.457/657)

HB 10.15 – 10.17, 10.22
FvDFH 12.1, 12.2, 12.6, 12.7

Marching Cubes, Lorensen et al. 1987
Announcement

• Source code and data files have been updated to include an implementation of vertex/fragment shaders for triangles and directional lights.
Solid Modeling

So far, we have focused on representing models with (triangular) meshes that approximate the surface/boundary of the model.

Advantages:

• Easy to visualize in graphics hardware

Limitations:

• Some models cannot be represented by a boundary
• It can be difficult to intersect two models
Motivation 1

• Some acquisition methods generate solids
  ◦ Example: Medical visualizations

https://www.sciencenewsforstudents.org/

Visible Human
(National Library of Medicine)
Motivation 2

- Some representations require solids
  - Example: FEM simulations

https://www.simscale.com/

[Irving et al., 2007]
Motivation 3

- Some algorithms require solids
  - Example: ray tracing in participating media

http://graphics.stanford.edu/

http://casual-effects.com/
Overview

• Implicit Surfaces
• Voxels
• Quadtrees and Octrees
Implicit Surfaces

Given a real-valued function in 3D, \( F(x, y, z) \), the implicit surface defined by \( F \) is the collection of points for which \( F(x, y, z) = 0 \).

• Example: quadric
  \[
  F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k
  \]
Implicit Surfaces

Given a real-valued function in 3D, \( F(x, y, z) \), the implicit surface defined by \( F \) is the collection of points for which \( F(x, y, z) = 0 \).

• Example: quadric
  \[ F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k \]

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

Ellipsoids

Image courtesy of http://wwwgeom.uiuc.edup/
Implicit Surfaces

Given a real-valued function in 3D, $F(x, y, z)$, the implicit surface defined by $F$ is the collection of points for which $F(x, y, z) = 0$.

- Example: quadric
  - $F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k$

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

Hyperboloids

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

Given a real-valued function in 3D, $F(x, y, z)$, the implicit surface defined by $F$ is the collection of points for which $F(x, y, z) = 0$.

• Example: quadric
  - $F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fzx + 2gx + 2hy + 2jz + k$

\[
\left(\frac{x}{rx}\right)^2 \pm \left(\frac{y}{ry}\right)^2 + 2z = 0
\]

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

Blobby Models

Express the implicit surface as a sum of (signed) Gaussians:

\[ F(x, y, z) = \sum_i F_i(x, y, z) \]

\[ F_i(x, y, z) = \alpha_i e^{-\frac{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}{2\sigma_i^2}} \]

- \((x_i, y_i, z_i)\)
  - Position: center of the Gaussian

- \(\alpha_i\) controls the contribution of the Gaussian
  - Magnitude: How much the Gaussian contributes
  - Sign: Interior vs. exterior

- \(\sigma_i\) controls the width of the Gaussian
  - Magnitude: fall-off of the contribution
Implicit Surfaces

Blobby Models

The more functions we use, the more accurate the reconstruction.

But this could also makes the function more difficult to sample.

Muraki, 1991
Implicit Surfaces

\[ F(x, y, z) = \sum_i F_i(x, y, z) \]

If the functions \( F_i \) are compactly supported, evaluation at a point can be done in sub-linear time.

Chen et al., SIGGRAPH 04
Implicit Surfaces

• Advantages:
  ◦ Easy to test if a point is on the surface
  ◦ Easy to test if a point is inside the surface
  ◦ Easy to intersect two surfaces

• Disadvantages:
  ◦ Hard to describe complex shapes
  ◦ Hard to evaluate complex functions
  ◦ Hard to enumerate points on surface
Overview

• Implicit Surfaces
• Voxels
• Quadtrees and Octrees
Voxels

- Partition space into uniform grid
  - Grid cells are called a voxels (like pixels)

- Each voxel has a value associated to it.
Voxels

- Partition space into uniform grid
  - Grid cells are called a **voxels** (like pixels)

- Each voxel has a value associated to it.
  - Binary Voxel Grids:
    - Value is 0 if the voxel is outside the model
    - Value is 1 if the voxel is inside
Binary Voxel Boolean Operations

• Compare objects voxel by voxel
  ◦ Trivial

```latex
\begin{array}{ccc}
\includegraphics[width=0.3\textwidth]{image1} & \cup & \includegraphics[width=0.3\textwidth]{image2} \\
\includegraphics[width=0.3\textwidth]{image3} & \cap & \includegraphics[width=0.3\textwidth]{image4}
\end{array}
```
Binary Voxel Visualization

- Draw the faces between on and off voxels.
Voxels

- Partition space into uniform grid
  - Grid cells are called *voxels* (like pixels)

- Each voxel has a value associated to it.
  - Binary Voxel Grids:
  - Continuous Voxel Grids:
    - Each voxel stores a continuous value (e.g. density, temperature, color, etc.)
Voxels

- Partition space into uniform grid
  - Grid cells are called *voxels* (like pixels)
- Each voxel has a value associated to it.
  - Binary Voxel Grids:
  - Continuous Voxel Grids:
    » Each voxel stores a continuous value (e.g. density, temperature, color, etc.)
Continuous Voxel Visualization

- Slicing
- Ray-Casting
- Iso-Surface Extraction
Voxel Display

• Slicing
  ◦ Draw 2D image by intersecting voxels with a plane
    » Supported by graphics card with 3D textures
Voxel Display

• Slicing
  ◦ Draw 2D image by intersecting voxels with a plane
    » Supported by graphics card with 3D textures
Voxel Display

- Ray casting
  - Integrate density along rays through pixels
Voxel Display

• Iso-Surface Extraction
  ◦ Treat the voxel grid as a regular sampling of a function $F(x, y, z)$, and extract the iso-surface with $F(x, y, z) = \delta$.  

 Iso-Value = $\delta_1$  
 Iso-Value = $\delta_2$
Marching Cubes Algorithm
[Lorensen and Cline, ’87]

- Iso-Surfaces analog with 2D grid
  - Assume each grid location has scalar value
  - If one of the vertices of an edge has value larger than $\delta$ and the other has value less than $\delta$, find the point on the edge whose linear interpolation is equal to $\delta$.
  - Connect the new edge points with line segments.

Note that the number of edges on which we insert vertices must be even.
Marching Cubes Algorithm
[ Lorensen and Cline, ’87 ]

- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case

Note that certain configurations are ambiguous.
Marching Cubes Algorithm [Lorensen and Cline, ’87]

- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case
  - Combine the segments from the different grid cells
Marching Cubes Algorithm [Lorensen and Cline, ’87]

- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case
  - Combine the segments from the different grid cells

As long as the geometry shared by adjacent cells (e.g. position of iso-vertices) is defined by values along the shared edge, adjacent cells will define consistent (connected) segments.
Marching Cubes Algorithm
[Lorensen and Cline, ’87]

Assigning iso-vertex position (linear):

If we have a function with $f(0) = a$ and $f(1) = b$, we can fit a linear interpolant:

$$f(x) = a + (b - a)x$$

Then for the function to have value $\delta$:

$$\delta = f(x) = a + (b - a)x$$

$$\delta - a \quad \downarrow$$

$$\frac{\delta - a}{b - a} = x$$
Marching Cubes Algorithm [Lorensen and Cline, ’87]

Assigning iso-vertex position (cubic):

If we also know $f(-1)$ and $f(2)$ we can fit a Cardinal B-spline to the four values and find the root(s) of the cubic polynomial in the range $[0,1]$. 

Note: Since we are using an interpolating spline, we are guaranteed to find an odd number of roots in the interval.  
⇒ Same number of iso-vertices as linear interpolation. 

The polynomial may have 3 roots in the range $[0,1]$. 
⇒ We need to choose the zero-crossing consistently.
Marching Cubes Algorithm
[Lorensen and Cline, ’87]

- Iso-Surface with 3D grid
  - Break up into the $2^8 = 256$ different possible cases
  - Assign a rule for surface extraction in each case
  - Combine the patches from the different grid cells

Back in the day, a table of $2^8$ configurations was too much to store in memory. Leveraging symmetry, [Lorensen and Cline, 1987] reduced it to 15 cases.
Marching Cubes Algorithm (Informally)

- Inductively:
  - Dimension 1: consider each edge in turn
    » Get edge \( \delta \)-crossings: iso-vertices
  - Dimension 2: consider each face in turn
    » Get face iso-edges: iso-polygon(s)
  - Dimension 3: triangulate the iso-polygon(s)

Note:
For this to give a seamless surface, we have to resolve the ambiguous 2D cases consistently.
Voxels

Continuous voxel grids are 3D images. Operations that we applied to 2D images can also be applied to voxel grids:

- Sampling
- Contrast
- Edge detection
- Smoothing
Voxels

Continuous voxel grids are 3D images. Operations that we applied to 2D images can also be applied to voxel grids:

- Sampling
- Contrast
- Edge detection
- Smoothing

Demo
Voxels

• Advantages
  ○ Simple
  ○ Same complexity for all objects
  ○ Natural acquisition for some applications
  ○ Trivial boolean operations

• Disadvantages
  ○ Approximate
  ○ Not affine invariant
  ○ Large storage requirements
  ○ Expensive display
Solid Modeling Representations

- Implicit Surfaces
- Voxels
- Quadtrees & Octrees
Quadtrees (2D) & Octrees (3D)

- Refine resolution of voxels hierarchically
Quadtrees(2D) and Octrees(3D)

Cell ordering:
1. bottom left
2. bottom right
3. top left
4. top right

- Expected complexity: number of nodes \(\cong\) to perimeter or surface area
Quadtree Boolean Operations

Cell ordering:
1. bottom left
2. bottom right
3. top left
4. top right

\[ A \cup B \quad A \cap B \]

FvDFH Figure 12.24
Quadtree Boolean Operations

Cell ordering:
1. bottom left
2. bottom right
3. top left
4. top right

If the operation results in all child nodes being marked empty/full

\[ \downarrow \]

Remove the children and mark the parent.
Octree Display

• Extend voxel methods
  ◦ Slicing
  ◦ Ray casting
  ◦ Iso-surface extraction

How to define positions of zero-crossings along edges shared by cells at different resolutions?
Octree Display

• Extend voxel methods
  ◦ Slicing
  ◦ Ray casting
  ◦ Iso-surface extraction

How to handle the situation when vertices on one side of a face do not exist on the other?
Octree Display

- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction

General Approach:
Copy from the finer faces to the coarser one.
Octree Display

• Extend voxel methods
  ○ Slicing
  ○ Ray casting
  ○ Iso-surface extraction

Octree adapted to surface curvature
Octree Display

- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction