



Surfaces

Thomas Funkhouser
Princeton University
COS 426, Spring 2006



3D Object Representations

- Raw data
 - Voxels
 - Point cloud
 - Range image
 - Polygons
- Solids
 - Octree
 - BSP tree
 - CSG
 - Sweep
- Surfaces
 - Mesh
 - Subdivision
 - Parametric
 - Implicit
- High-level structures
 - Scene graph
 - Application specific



Curved Surfaces

- What makes a good surface representation?
 - Accurate
 - Concise
 - Intuitive specification
 - Local support
 - Affine invariant
 - Arbitrary topology
 - Guaranteed continuity
 - Natural parameterization
 - Efficient display
 - Efficient intersections



Curved Surface Representations

- Polygonal meshes
- Subdivision surfaces
- Parametric surfaces
- Implicit surfaces

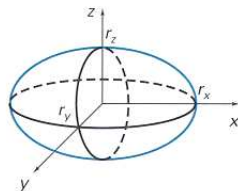


Parametric Surfaces

- Boundary defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$

- Example: ellipsoid

$$\begin{aligned}
 x &= r_x \cos \phi \cos \theta \\
 y &= r_y \cos \phi \sin \theta \\
 z &= r_z \sin \phi
 \end{aligned}$$

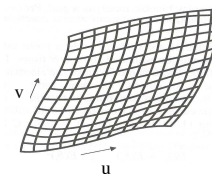


H&B Figure 10.10



Parametric Surfaces

- Advantages:
 - Easy to enumerate points on surface



- Problem:
 - Need piecewise-parametrics surfaces to describe complex shapes

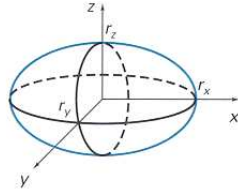
FvDFH Figure 11.42

Parametric Surfaces



- Boundary defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$

- Example: ellipsoid
 - $x = r_x \cos \phi \cos \theta$
 - $y = r_y \cos \phi \sin \theta$
 - $z = r_z \sin \phi$

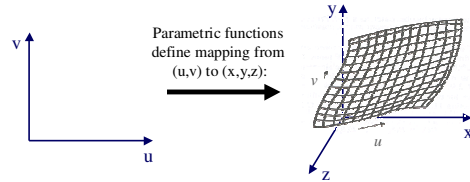


H&B Figure 10.10

Parametric Surfaces



- Boundary defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$

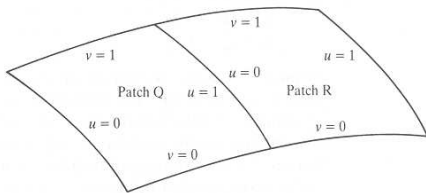


FvDFH Figure 11.42

Piecewise Parametric Surfaces



- Surface is partitioned into parametric patches:



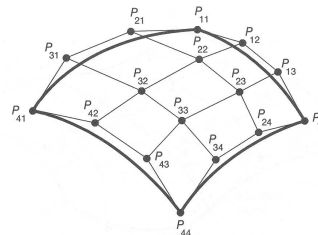
Same ideas as parametric splines!

Watt Figure 6.25

Parametric Patches



- Each patch is defined by blending control points



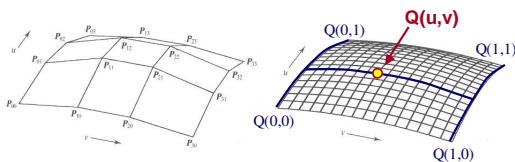
Same ideas as parametric curves!

FvDFH Figure 11.44

Parametric Patches



- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points



Watt Figure 6.21

Parametric Bicubic Patches



- Point $Q(u,v)$ on any patch is defined by combining control points with polynomial blending functions:

$$Q(u,v) = \mathbf{U} \mathbf{M} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}^T \mathbf{V}^T$$

$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1] \quad \mathbf{V} = [v^3 \quad v^2 \quad v \quad 1]$$

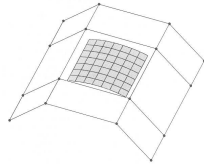
Where M is a matrix describing the blending functions for a parametric cubic curve (e.g., Bezier, B-spline, etc.)

B-Spline Patches



$$Q(u, v) = \mathbf{U} \mathbf{M}_{\text{B-Spline}} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}_{\text{B-Spline}}^T \mathbf{V}$$

$$\mathbf{M}_{\text{B-Spline}} = \begin{bmatrix} -1/6 & 1/2 & -1/2 & 1/6 \\ 1/2 & -1 & 1/2 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1/6 & 2/3 & 1/6 & 0 \end{bmatrix}$$



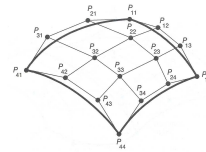
Watt Figure 6.28

Bezier Patches



$$Q(u, v) = \mathbf{U} \mathbf{M}_{\text{Bezier}} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}_{\text{Bezier}}^T \mathbf{V}$$

$$\mathbf{M}_{\text{Bezier}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

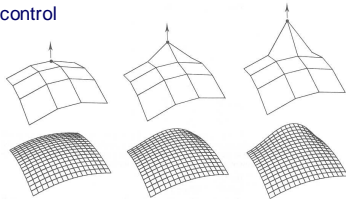


FvDFH Figure 11.42

Bezier Patches



- Properties:
 - Interpolates four corner points
 - Convex hull
 - Local control

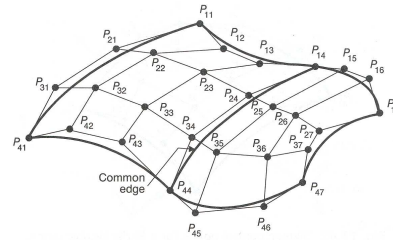


Watt Figure 6.22

Bezier Surfaces



- Continuity constraints are similar to the ones for Bezier splines

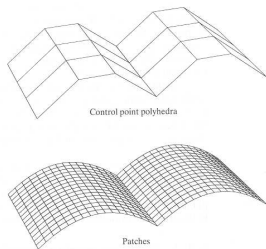


FvDFH Figure 11.43

Bezier Surfaces



- C^0 continuity requires aligning boundary curves

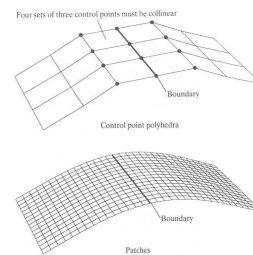


Watt Figure 6.26a

Bezier Surfaces



- C^1 continuity requires aligning boundary curves and derivatives



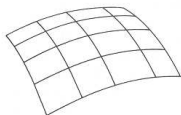
Watt Figure 6.26b

Drawing Bezier Surfaces



- Simple approach is to loop through uniformly spaced increments of u and v

```
DrawSurface(void)
{
  for (int i = 0; i < imax; i++) {
    float u = u_min + i * ustep;
    for (int j = 0; j < jmax; j++) {
      float v = v_min + j * vstep;
      DrawQuadrilateral(...);
    }
  }
}
```



Watt Figure 6.32

Drawing Bezier Surfaces



- Better approach is to use adaptive subdivision:

```
DrawSurface(surface)
{
  if Flat (surface, epsilon) {
    DrawQuadrilateral(surface);
  }
  else {
    SubdivideSurface(surface, ...);
    DrawSurface(surfaceLL);
    DrawSurface(surfaceLR);
    DrawSurface(surfaceRR);
  }
}
```



Uniform subdivision



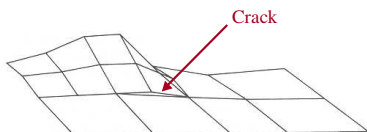
Adaptive subdivision

Watt Figure 6.32

Drawing Bezier Surfaces



- One problem with adaptive subdivision is avoiding cracks at boundaries between patches at different subdivision levels



Avoid these cracks by adding extra vertices and triangulating quadrilaterals whose neighbors are subdivided to a finer level

Watt Figure 6.33

Parametric Surfaces



- Advantages:
 - Easy to enumerate points on surface
 - Possible to describe complex shapes
- Disadvantages:
 - Control mesh must be quadrilaterals
 - Continuity constraints difficult to maintain
 - Hard to find intersections

Curved Surface Representations

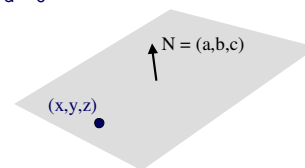


- Polygonal meshes
- Subdivision surfaces
- Parametric surfaces
- Implicit surfaces

Implicit Surfaces



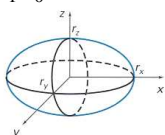
- Boundary defined by implicit function:
 - $f(x, y, z) = 0$
- Example: linear (plane)
 - $ax + by + cz + d = 0$



Implicit Surfaces

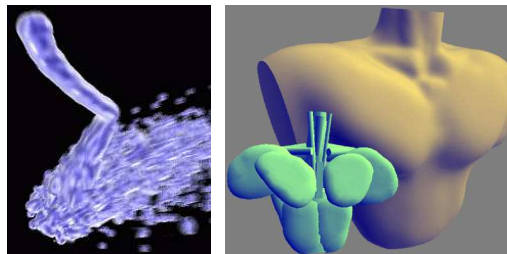


- Example: quadric
 $f(x,y,z)=ax^2+by^2+cz^2+2dxy+2eyz+2fxz+2gx+2hy+2jz+k$
- Common quadric surfaces:
 - Sphere
 - Ellipsoid $\rightarrow \left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 - 1 = 0$
 - Torus
 - Paraboloid
 - Hyperboloid



H&B Figure 10.10

Implicit surface examples



MaxMan Bloby Object

Skin [Markosian99]

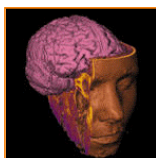
Implicit Surface Examples



- Voxels describe density at regular sample points
 - Surface may not be described explicitly



Visible Human
(National Library of Medicine)



SUNY Stony Brook

Implicit Surfaces



- Advantages:
 - Easy to test if point is on surface
 - Easy to intersect two surfaces
 - Easy to compute z given x and y
- Disadvantages:
 - Hard to describe specific complex shapes
 - Hard to enumerate points on surface

Summary



Feature	Polygonal Mesh	Implicit Surface	Parametric Surface	Subdivision Surface
Accurate	No	Yes	Yes	Yes
Concise	No	Yes	Yes	Yes
Intuitive specification	No	No	Yes	No
Local support	Yes	No	Yes	Yes
Affine invariant	Yes	Yes	Yes	Yes
Arbitrary topology	Yes	No	No	Yes
Guaranteed continuity	No	Yes	Yes	Yes
Natural parameterization	No	No	Yes	No
Efficient display	Yes	No	Yes	Yes
Efficient intersections	No	Yes	No	No

3D Object Representations

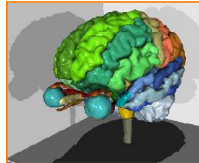


- Raw data
 - Point cloud
 - Range image
 - Polygon soup
- Surfaces
 - Mesh
 - Subdivision
 - Parametric
 - Implicit
- Solids
 - Voxels
 - BSP tree
 - CSG
 - Sweep
- High-level structures
 - Scene graph
 - Application specific

Solid Modeling Representations



- What makes a good solid representation?
 - Accurate
 - Concise
 - Affine invariant
 - Easy acquisition
 - Guaranteed validity
 - Efficient boolean operations
 - Efficient display



Lorensen

Solid Modeling Representations



- Voxels
- Quadrees & Octrees
- Binary space partitions
- Constructive solid geometry

Solid Modeling Representations

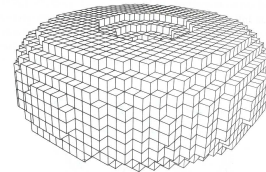


- **Voxels**
- Quadrees & Octrees
- Binary space partitions
- Constructive solid geometry

Voxels



- Partition space into uniform grid
 - Grid cells are called a *voxels* (like pixels)
- Store properties of solid object with each voxel
 - Occupancy
 - Color
 - Density
 - Temperature
 - etc.

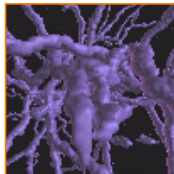


FvDFH Figure 12.20

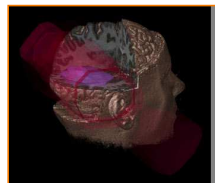
Voxel Acquisition



- Scanning devices
 - MRI
 - CAT
- Simulation
 - FEM



SUNY Stony Brook

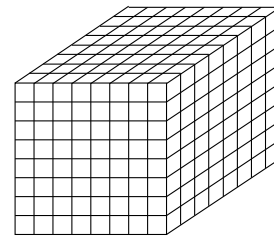


Stanford University

Voxel Storage



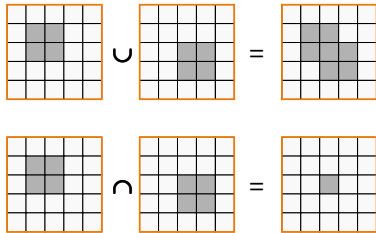
- $O(n^3)$ storage for $n \times n \times n$ grid
 - 1 billion voxels for $1000 \times 1000 \times 1000$



Voxel Boolean Operations



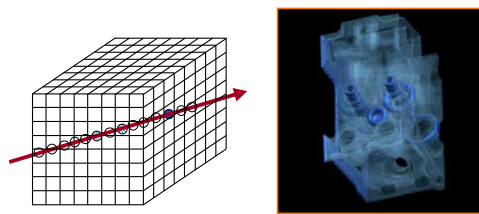
- Compare objects voxel by voxel
 - Trivial



Voxel Display



- Ray casting
 - Integrate density along rays through pixels

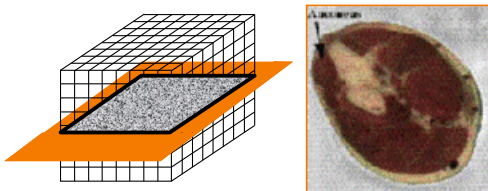


Engine Block
Stanford University

Voxel Display



- Slicing
 - Draw 2D image resulting from intersecting voxels with a plane

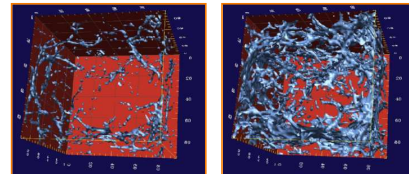


Visible Human
(National Library of Medicine)

Voxel Display

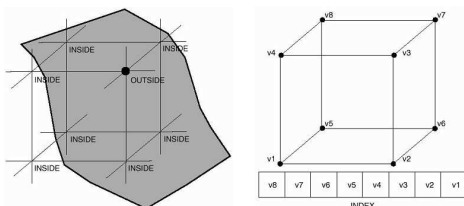


- Isosurface rendering
 - Render surfaces bounding volumetric regions of constant value (e.g., density)

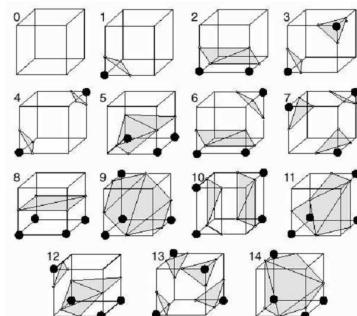


Isosurface Visualization
Princeton University

Marching Cubes



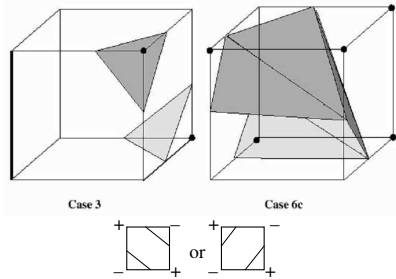
Marching Cubes (15 Cases)



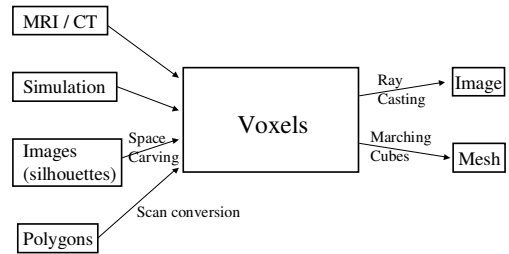
Marching Cubes



- Sometimes have to match neighbors



Voxels



Voxels



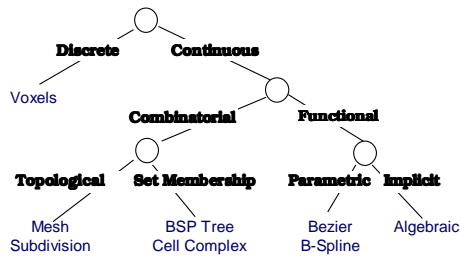
- Advantages
 - Simple, intuitive, unambiguous
 - Same complexity for all objects
 - Natural acquisition for some applications
 - Trivial boolean operations
- Disadvantages
 - Approximate
 - Not affine invariant
 - Large storage requirements
 - Expensive display

Summary



	Voxels
Accurate	No
Concise	No
Affine invariant	No
Easy acquisition	Some
Guaranteed validity	Yes
Efficient boolean operations	Yes
Efficient display	No

Taxonomy of 3D Representations



Naylor