Surface Representations
Polygonal Meshes
Volumetric Models
Merging Multiple Views

EE-148 3D Photography

Caltech Spring 2001

Gabriel Taubin

Implicit surfaces

- Set of zeros of a function
  - { (x,y,z) : f(x,y,z) = 0 }
- Good for boolean operations (CSG)
- Difficult to render (ray-tracing)
- Iso-surface
  - Function defined by piecewise function
  - Volumetric mesh
  - 1 function value per vertex
- Iso-surface algorithm
  - Conversion to triangle or polygon mesh representation

Implicit surfaces

- Can be used to represent the probability that a point belongs to a surface
  - Occupancy grid
- Can be used to integrate multiple measurements
- Can be used to merge multiple 3D scans

Piecewise Linear Functions

- Triangle : Barycentric coordinates
  - Triangle / Tetrahedron / Simplex
- Every point in 3D can be written as a unique affine combination of 4 non-coplanar points (affine basis)
- Every linear function in 3D can be specified by its values at the 4 vertices of an affine basis
- A piecewise-linear function is specified in 3D by its values at the vertices of a tetrahedral mesh (volumetric).

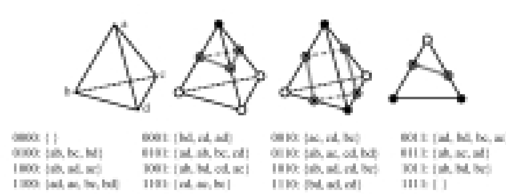
Affine bases / Linear function

$$p = \lambda_0\ p_0 + \lambda_1\ p_1 + \lambda_2\ p_2 + \lambda_3\ p_3$$

$$\begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p \\ 1 \end{bmatrix}$$

$$f(p) = \lambda_0\ f(p_0) + \lambda_1\ f(p_1) + \lambda_2\ f(p_2) + \lambda_3\ f(p_3)$$

Implicit Linear Surfaces / Curves



Iso-surfaces on tetrahedral meshes

- Piecewise linear function defined on vertices of tetrahedral mesh f(i)
- For each edge (i,j) such that f(i)f(j)<0
  - create a surface vertex v(i,j)
- For each tetrahedron (i,j,k,l)
  - Skip if all vertices are positive or negative
  - Else if 3 positive or 3 negative create a triangle
  - Else (if 2 positive and 2 negative) create two trianges
- Output triangle mesh is IndexedFaceSet
- Is it a manifold mesh ? Why ?

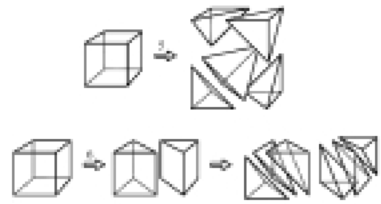


Figure 24. Five- and Six-Decompositions of the Cube

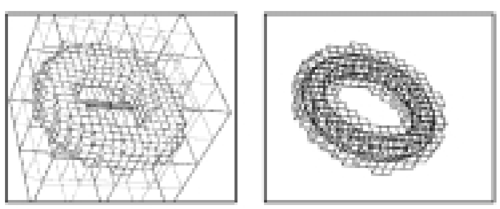
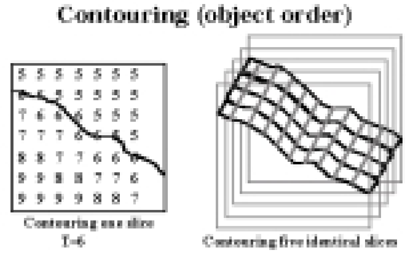


Figure 4. Subdivision Enclosing a Torus

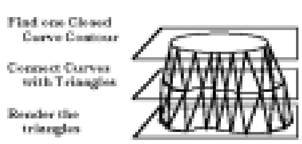
Contouring (object order)



- Assumes Volume Contains Thin Boundary Surfaces
- Classify All Cells as Inside, Outside, or "On" The Surface
- Fit Constant-value Surfaces to All "On" Cells
- Render Surfaces

Connecting Slices

Connecting Slices (cont.)



Iso-surfaces on hexahedral meshes

- Function defined on vertices of regular grid
- For each edge (i,j) such that f(i)f(j)<0
  - create a surface vertex v(i,j)
- For each intersecting cube
  - Polygonize intersection
- Output triangle mesh is IndexedFaceSet
- Is it a manifold mesh ? Why ?
- Main problem: storage
- Solution: do not represent the mesh explicitly

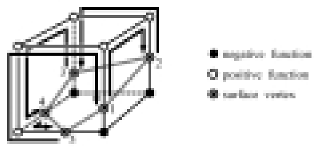
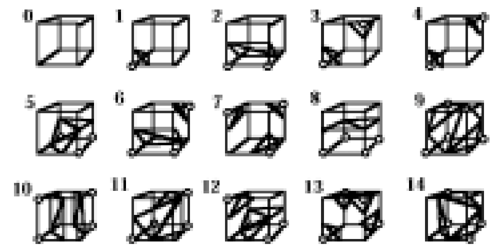


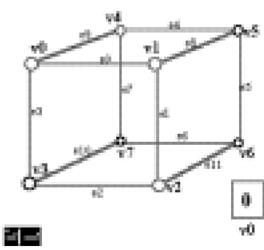
Figure 17. Cubical Polygonization

Edge Insertion Table

256 Cases Reduce to 15



Constructing an Index into the Edge Intersection Table



0	0	0	1	0	1	1	1
v0	v1	v2	v3	v4	v5	v6	v7

Marching Cubes Algorithm

1. User Specifies Threshold Value
2. Read Four Slices Into Memory
3. Scan Middle Two Slices and Create a Cell
4. Classify Eight Vertices. Construct Index Number.
5. Use Index to Look Up List of Edges
6. Find 3 Surf/Edge Intersections via Linear Interpolation
7. Calculate Unit Normal (Gradient) at 3 Intersections
8. Output the Triangle Vertices and Vertex Normals

Interpolation

- Linear interpolation
- Triangle : Barycentric coordinates
  - Triangle
  - Tetrahedron
- Quadrilateral ?
  - Bi-linear interpolation
- Cube ?
  - Tri-linear interpolation

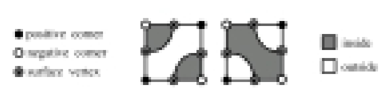


Figure 21. Alternate Surface Vertex Connections for a Face

six arrangements of ambiguous faces within a cube are shown below.

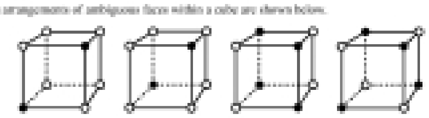
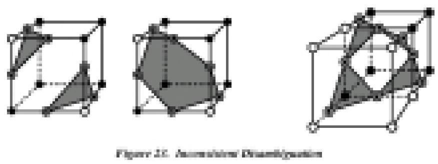


Figure 22. Ambiguous Corner Configurations for a Cube



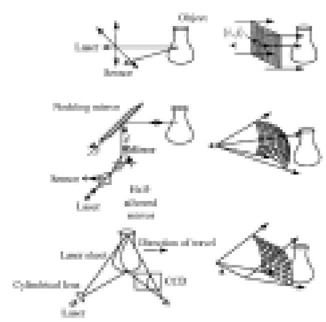
Building meshes from data

- Range images
  - Rectangular array of depths
  - Texture image (optional)
- Connect 4 contiguous vertices as a Q
- What is a hole ?
- What is a discontinuity ?
- Noise ?
- Be conservative
- Merge multiple views

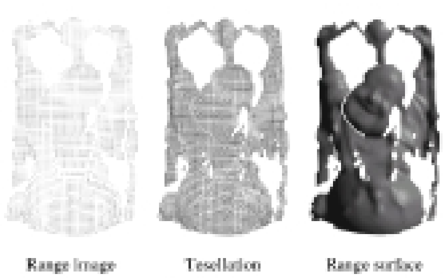
Registration and Merging

- **Range images vs. point clouds**
- **Registration**
- **Reconstruction from point clouds**
- **Reconstruction from range images**
- **Modeling appearance**

Range Image Sampling Pattern

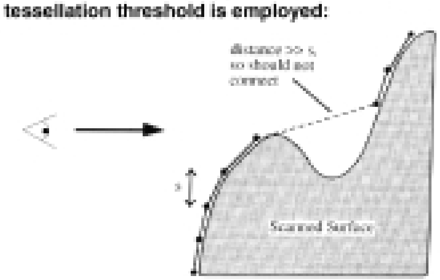


Range Surface



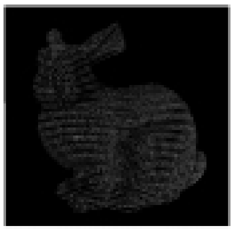
Tessellation Threshold

To avoid "prematurely aggressive" reconstruction, a tessellation threshold is employed:



Overlapping range images

We can view the entire set of range data as a point cloud or as a group of overlapping range surfaces.



Registration

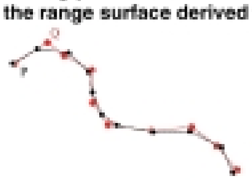
Given two overlapping range scans, we wish to solve for the rigid transformation, *T*, that minimizes the distance between them.

Registration

An approximation to the distance between range scans is:

$$E = \sum_{i}^{N_p} \left\| Tq_i - p_i \right\|^2$$

Where the q_i are samples from scan Q and the p_i are the *corresponding* points of scan P. These points may lay on the range surface derived from P.

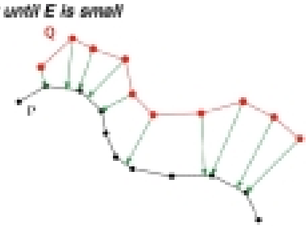


Registration

- Closed form solution if correspondence is known
- This is often not the case
- Iterative Closest Point Algorithm
  - Choose points in first mesh
  - Find closest points in second mesh
  - Solve for optimal transformation
  - Iterate until convergence

ICP

- *Identify nearest points*
- *Compute the optimal T*
- *Repeat until E is small*



Merging multiple range images

- Surface based
  - Zippering
  - Ball-Pivoting Algorithm (BPA)
- Volume based
  - Each range image defines a signed inside-outside function
  - Integrate multiple range images into a single inside-outside function
  - Compute iso-surface