Geometric Modeling

Sources of 3D data

- Modeling programs
  - Autodesk
  - CATIA
  - Bryce
  - Pro/E
- 3D Scanning
  - LiDAR/Range Scanning
- Photogrammetry
  - Structure from motion
- Procedural methods
  - Program creates model algorithmically
- Typing it in....
In 1998 Marc Levoy of Stanford scanned several of Michelangelo sculptures
David at 1mm resolution
St. Matthew at 290µm resolution
about 200M triangles
What does “scanning resolution” mean?
Triangle Meshes

Triangles: fundamental rendering primitive of modern GPUs

Common Modeling Operation

Tessellation: split a surface into a set of polygons

First step is often to project to 2D
....then apply one of many 2D triangulation algorithms

Triangulation tessellates using triangles

Consider the problem of simply triangulating a 2D polygon....
Recursive Brute Force...

- Pick two vertices
- If an edge between them is a valid choice, split polygon
- Apply recursively until can’t split anymore...

How do you decide if an edge is valid?

What is the worst case running time for n vertices?
Tessellation

Recursive Brute Force...

Pick two vertices
If an edge between them is a valid choice, split polygon
Apply recursively until can’t split anymore...

How do you decide if an edge is valid?

Test if the edge intersects or overlaps any polygon edge.
Test for inclusion inside polygon

What is the worst case running time for n vertices?

$O(n^3)$

Bernard Chazelle [1991] any simple polygon can be triangulated in linear time
Scattered Data Triangulation – State of the Art

Delaunay Triangulation

for set P of points in a plane there is a triangulation DT(P) such that no point is inside the circumcircle of any triangle in DT(P).

DT(P) maximizes the minimum angle in the triangulation often used for scientific applications

O(n lg n) incremental algorithm
Good Meshes

**Manifold:**
1. Every edge connects exactly two faces
2. Vertex neighborhood is “disk-like”

**Orientable:** Consistent normals

**Watertight:** Orientable + Manifold

**Boundary:** Some edges bound only one face

**Ordering:** Vertices in CCW order when viewed from normal
2-Manifold Meshes

Disk-shaped neighborhoods

non-manifolds
Mesh Characteristics

- Single component, closed, triangular, orientable manifold
- Multiple components
- With boundaries
- Not only triangles
- Not orientable
- Non manifold
Genus

Genus:
Half the maximal number of closed paths that do not disconnect the mesh (= the number of holes)
Euler Formula

For a closed (no boundary), manifold, connected surface mesh
\[ V - E + F = 2(1 - G) \]

- **V** = number of vertices
- **E** = number of edges
- **F** = number of faces
- **G** = genus (number of holes in the surface)

A **2-manifold** is a surface (locally like a plane)
For Closed 2-Manifold Polygonal Meshes

\[ V + F - E = \chi \]

\( V = 8 \)
\( E = 12 \)
\( F = 6 \)
\[ \chi = 8 + 6 - 12 = 2 \]

\( V = 3890 \)
\( E = 11664 \)
\( F = 7776 \)
\[ \chi = 2 \]
...and if they are triangle meshes

- **Triangle** mesh statistics
  \[ E \approx 3V \]
  \[ F \approx 2V \]

- Avg. valence \( \approx 6 \)
  *Show using Euler Formula*
Mesh Data Structures

Need to store
- Geometry
- Connectivity

Can be used as file formats or internal formats

Considerations
- Space
- Efficient operations

Mesh processing has different requirements than rendering
  Example: Smoothing by averaging a vertex with neighbor vertices
Face Set (STL)

- face:
  - 3 positions

<table>
<thead>
<tr>
<th>Triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_{11} )</td>
</tr>
<tr>
<td>( x_{12} )</td>
</tr>
<tr>
<td>( x_{13} )</td>
</tr>
<tr>
<td>( x_{21} )</td>
</tr>
<tr>
<td>( x_{22} )</td>
</tr>
<tr>
<td>( x_{23} )</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>( x_{F1} )</td>
</tr>
<tr>
<td>( x_{F2} )</td>
</tr>
<tr>
<td>( x_{F3} )</td>
</tr>
</tbody>
</table>

36 B/f = 72 B/v
no connectivity!
Indexed Face Set (OBJ)

- **vertex:**
  - position

- **face:**
  - vertex indices

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Triangles</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$ $y_1$ $z_1$</td>
<td>$v_{11}$ $v_{12}$ $v_{13}$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$x_v$ $y_v$ $z_v$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td></td>
<td>$\cdots$</td>
</tr>
<tr>
<td></td>
<td>$v_{F1}$ $v_{F2}$ $v_{F3}$</td>
</tr>
</tbody>
</table>

$12 \text{ B/v} + 12 \text{ B/f} = 36 \text{ B/v}$

no neighborhood info
Halfedge-Based Connectivity

- vertex
  - position
  - 1 halfedge

- halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges

- face
  - 1 halfedge

96 to 144 B/v
class HalfEdge {
    HalfEdge *opp;
    Vertex *end;
    Face *left;
    HalfEdge *next;
};

HalfEdge e;
class HalfEdge {
    HalfEdge *opp;
    Vertex *end;
    Face *left;
    HalfEdge *next;
};

HalfEdge e;
Half Edge

class HalfEdge {
    HalfEdge *opp;
    Vertex *end;
    Face *left;
    HalfEdge *next;
};

HalfEdge e;

e->start() = e->opp()->end();
class HalfEdge {
    HalfEdge *opp;
    Vertex *end;
    Face *left;
    HalfEdge *next;
};

HalfEdge e;

e->right() = e->opp()->left();
Half Edge

class HalfEdge {
    HalfEdge *opp;
    Vertex *end;
    Face *left;
    HalfEdge *next;
};

HalfEdge e;

Can walk around left face until e(->next)^n = e
**Vertex Star:** A set consisting of
- the vertex
- its incident edges,
- neighboring vertices

**1-Ring:** A set consisting of all incident
- Edges
- Faces
- Neighboring vertices
1-Ring Traversal

1. Start at vertex
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
1-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...
Mesh Smoothing
Mesh Processing Pipeline

Scan  Reconstruct  Clean  Remesh
Mesh Quality

- Visual inspection of “sensitive” attributes
  - Specular shading
  - Reflection lines
  - Curvature
Motivation

• Filter out high frequency noise
Mesh Smoothing
(aka Denoising, Filtering, Fairing)

**Input:** Noisy mesh (scanned or other)
**Output:** Smooth mesh
**How:** Filter out high frequency noise
Laplacian Smoothing

An easier problem: How to smooth a curve?

\[ \mathbf{p}_i = (x_i, y_i) \]

\[ \mathbf{p}_i = \frac{\mathbf{p}_{i-1} + \mathbf{p}_{i+1}}{2} - \mathbf{p}_i \]

\[ L(\mathbf{p}_i) = \frac{1}{2} (\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2} (\mathbf{p}_{i-1} - \mathbf{p}_i) \]
Laplacian Smoothing

An easier problem: How to smooth a curve?

Finite difference discretization of second derivative
= Laplace operator in one dimension
Laplacian Smoothing

Algorithm:
Repeat for $m$ iterations (for non boundary points):

$$p_i \leftarrow p_i + \lambda L(p_i)$$

For which $\lambda$?

$$0 < \lambda < 1$$

Closed curve converges to?
Single point
Laplacian Smoothing on Meshes

Same as for curves:

\[ p_i^{(t+1)} = p_i^{(t)} + \lambda \Delta p_i^{(t)} \]

What is \( \Delta p_i \)?

\[
\frac{1}{2} (p_{i+1} + p_{i-1}) - p_i
\]

\[
\frac{1}{|N_i|} \left( \sum_{j \in N_i} p_j \right) - p_i
\]
Laplacian Smoothing on Meshes
Problem - Shrinkage

Repeated iterations of Laplacian smoothing shrinks the mesh
Taubin Smoothing

Iterate:

\[ p_i \leftarrow p_i + \lambda \Delta p_i \]
\[ p_i \leftarrow p_i + \mu \Delta p_i \]

with \( \lambda > 0 \) and \( \mu < 0 \)

Shrink
Inflate

From Taubin, Siggraph 1995
Laplacian Smoothing

\[ p_i^{(t+1)} = p_i^{(t)} + \lambda \Delta p_i^{(t)} \]

\[ \Delta p_i = \text{mean curvature normal} \]

mean curvature flow
Laplace Operator Discretization
The Problem

The result should not depend on triangle sizes

From Desbrun et al., Siggraph 1999
Laplace Operator Discretization

What Went Wrong?

Back to curves:

$$\frac{1}{2}(p_{i+1} + p_{i-1}) - p_i$$

Same weight for both neighbors, although one is closer
Laplace Operator Discretization

The Solution

Use a weighted average to define $\Delta$

Which weights?

$$w_{ij} = \frac{1}{l_{ij}}, \quad w_{ik} = \frac{1}{l_{ik}}$$

$$L(p_i) = \frac{w_{ij}p_j + w_{ik}p_k}{w_{ij} + w_{ik}} - p_i$$

Straight curves will be invariant to smoothing
Laplace Operator Discretization

Cotangent Weights

Use a weighted average to define $\Delta$

Which weights?

$$w_{ij} = \frac{h_{ij}^1 + h_{ij}^2}{l_{ij}} = \frac{1}{2} \left( \cot \alpha_{ij} + \cot \beta_{ij} \right)$$

$$L(p_i) = \frac{1}{\sum w_{ij} j \in N_i} \sum w_{ij} (p_j - p_i)$$

Planar meshes will be invariant to smoothing
Contangent Weights

Figure 4: Left: uniform (red) and cotangent (green) Laplacian vectors for a vertex $v_i$ and its (in this case planar) 1-ring, as well as the angles used in Eqn. 4 for one $v_j$. Bottom right: the effect of flattening $v_i$ into the 1-ring plane. While the cotangent Laplacian vanishes, the uniform Laplacian generally does not. Right top: the Voronoi region $A(v_i)$ around a vertex.
Smoothing with the Cotangent Laplacian

From Desbrun et al., Siggraph 1999
Alternative: Bilateral Smoothing

- Displace vertex along a normal
- Displacement computed from neighboring vertices
  - Weights based both on spatial distance and normal direction of neighbor
- Non-iterative...more scalable
MLS Smoothing of Point Sets

- Fit a plane to a neighborhood of points with normals
- Construct a bivariate polynomial $g$ over that plane based on
  - $f_i$: distances of points to the plane
  - weight related to distance
  - weight related to normal deviation
- Minimizes

$$
\sum_{i=1}^{n}(g(x_i, y_i) - f_i)^2 \theta(||p_i - q||) \phi(n_i, n_q)
$$

- Move a point along the normal to $g$
Morton Coding for Point Set

Figure 5.6: A Z-order traversal of a unit-cube using a uniform grid
Changing Neighborhood Size

Figure 5.7: Effect of varying neighborhood size during smoothing
Mesh Processing

- Input Data
  - Range-Scan
  - CAD

- Removal of topological and geometrical errors

- Analysis of surface quality

- Surface smoothing for noise removal
Mesh Processing

Simplification for complexity reduction

Parameterization

Remeshing for improving mesh quality
Mesh Processing

Freeform and multiresolution modeling

Deformation and editing

Extracting shape structure
Mesh Processing Pipeline

- We will look at an application at the end of the pipeline
- *Simplification* is a type of remeshing
- Current state-of-the-art developed over the last decade...
Simplification: Applications

- Oversampled 3D scan data

~150k triangles

~80k triangles
Simplification: Applications

- Over tessellation: E.g. iso-surface extraction
Simplification: Applications

- Multi-resolution hierarchies for
  - efficient geometry processing
  - level-of-detail (LOD) rendering
Mesh Decimation Methods

- Vertex clustering
- Incremental decimation
- Resampling
- Mesh approximation
Vertex Clustering

- Cluster Generation
- Computing a representative
- Mesh generation
- Topology changes
Vertex Clustering

- Cluster Generation
  - Uniform 3D grid
  - Map vertices to cluster cells

- Computing a representative

- Mesh generation

- Topology changes
Vertex Clustering

- Cluster Generation
- Computing a representative
  - Average/median vertex position
  - Error quadrics
- Mesh generation
- Topology changes
Computing a Representative

Average vertex position
Computing a Representative

Median vertex position
Computing a Representative

Error quadrics
Error Quadrics

- Patch is expected to be piecewise flat
- Minimize distance to neighboring triangles’ planes
Error Quadrics

- Squared distance of point $p$ to plane $q$: remember plane equation: $ax + by + cz + d = 0$

$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$\text{dist}(q, p)^2 = (q^T p)^2 = p^T (qq^T)p =: p^T Q_q p$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$
Error Quadrics

• Sum distances to planes $q_i$ of vertex neighboring triangles:

$$\sum_i \text{dist}(q_i, p)^2 = \sum_i p^T Q_{q_i} p = p^T \left( \sum_i Q_{q_i} \right) p =: p^T Q_p p$$

• Point $p^*$ that minimizes the error satisfies:

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} p^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$
Comparison

average

median

error quadric
Vertex Clustering

- Cluster Generation
- Computing a representative
- Mesh generation
  - Clusters $p \leftrightarrow \{p_0, ..., p_n\}$, $q \leftrightarrow \{q_0, ..., q_m\}$
  - Connect $(p, q)$ if there was an edge $(p_i, q_j)$
- Topology changes
Vertex Clustering

- Cluster Generation
- Computing a representative
- Mesh generation
- Topology changes
  - If different sheets pass through one cell
  - Can be non-manifold
Outline

• Applications

• Problem Statement

• Mesh Decimation Methods
  – Vertex Clustering
  – **Incremental Decimation**
  – Extensions
Incremental Decimation
Vertex Removal

Select a vertex to be eliminated
Vertex Removal

Select all triangles sharing this vertex
Vertex Removal

Remove the selected triangles, creating the hole
Vertex Removal

Fill the hole with new triangles
Decimation Operators

- Remove vertex
- Re-triangulate hole
  - Combinatorial degrees of freedom
Decimation Operators

- Collapse edge into one end point
  - Special case of vertex removal
  - Special case of edge collapse
- No degrees of freedom
- Separates global optimization from local optimization
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Half-Edge Collapse
Decimation Operators

- Merge two adjacent vertices
- Define new vertex position
  - Continuous degrees of freedom
  - Filter along the way
Edge Collapse

Removing an edge…
- Merges two vertices into one vertex
- Removes two faces
- Mesh still consists of triangles

Which edges should be removed first?
Where should the new vertices go?
Quadric Error Metric

Find $Q$ for each vertex $v$

$$Q_v = \Sigma Q_i$$

(for adjacent polygons $i$)

Create a priority queue of edge collapses:

$\rightarrow$ Each collapse would create new vertex $v_{new}$

$\rightarrow$ $Q_{v_{new}} = Q_{v1} + Q_{v2}$

$\rightarrow$ Choose collapse with min $Q(v_{new})$

$$v_{12} = -\begin{bmatrix} A^2 & AB & AC \\ AB & B^2 & BC \\ AC & BC & C^2 \end{bmatrix}^{-1} \begin{bmatrix} AD \\ BD \\ CD \end{bmatrix}$$
Examples