Mesh and Mesh Simplification

Qixing Huang
Mar. 21st 2018
Mesh Data Structures
Data Structures

• What should be stored?
  – Geometry: 3D coordinates
  – Attributes
    • e.g. normal, color, texture coordinate
    • Per vertex, per face, per edge
  – Connectivity
    • Adjacency relationships
Data Structures

• What should it support?
  – Rendering
  – Geometry queries
    • What are the vertices of face #2?
    • Is vertex A adjacent to vertex H?
    • Which faces are adjacent to face #1?
  – Modifications
    • Remove/add a vertex/face
    • Vertex split, edge collapse
Data Structures

• How good is a data structure?
  – Time to construct (preprocessing)
  – Time to answer a query
  – Time to perform an operation
  – Space complexity
  – Redundancy
Mesh Data Structures

- Face Set
- Shared Vertex
- Half Edge
- Face Based Connectivity
- Edge Based Connectivity
- Adjacency Matrix
- Corner Table
## Face Set

### TRIANGLES

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[10 20 30]</td>
<td>[40 5 20]</td>
<td>[10 4 3]</td>
</tr>
</tbody>
</table>

- Simple
- STL File
- No connectivity
- Redundancy
**Shared Vertex**

<table>
<thead>
<tr>
<th>TRIANGLES</th>
<th>VERTICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex Index</td>
<td>Vertex Coord.</td>
</tr>
<tr>
<td>2</td>
<td>[40 5 20]</td>
</tr>
<tr>
<td>1</td>
<td>[10 20 30]</td>
</tr>
<tr>
<td>3</td>
<td>[10 4 3]</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

- Connectivity
- No neighborhood
Shared Vertex

```
<table>
<thead>
<tr>
<th>TRIANGLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 3 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VERTICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1 [20 100]</td>
</tr>
<tr>
<td>v2 [19 200]</td>
</tr>
<tr>
<td>v3 [14 150]</td>
</tr>
<tr>
<td>.</td>
</tr>
<tr>
<td>.</td>
</tr>
<tr>
<td>.</td>
</tr>
</tbody>
</table>
```
What are the vertices of face $f_1$?
- $O(1)$ - first triplet from face list
Shared Vertex

- Are vertices $v_1$ and $v_5$ adjacent?
  - Requires a full pass over all faces
Half Edge Data Structure

- **Vertex stores**
  - Position
  - 1 outgoing halfedge

![Diagram of vertex with an outgoing halfedge]
Half Edge Data Structure

- **Halfedge stores**
  - 1 origin vertex index
  - 1 incident face index
  - `next`, `prev`, twin halfedge indices
Half Edge Data Structure

• Face stores
  – 1 adjacent halfedge index
Half Edge Data Structure

- Neighborhood Traversal

a) b) c) d)
Face Based Connectivity

- **Vertex:**
  - position
  - 1 adjacent face index

- **Face:**
  - 3 vertex indices
  - 3 neighboring face indices

- No (explicit) edge information
Edge Based Connectivity

• **Vertex**
  - position
  - 1 adjacent edge index

• **Edge**
  - 2 vertex indices
  - 2 neighboring face indices
  - 4 edges

• **Face**
  - 1 edge index

• No edge orientation information
• Adjacency Matrix “A”
• If there is an edge between \( v_i \) & \( v_j \) then \( A_{ij} = 1 \)
Adjacency Matrix

- Symmetric for undirected simple graphs
- \((A^n)_{ij} = \# \) paths of length \(n\) from \(v_i\) to \(v_j\)
- Pros:
  - Can represent non-manifold meshes
- Cons:
  - No connection between a vertex and its adjacent faces
Corner Table

- Corner is a vertex with one of its incident triangles
Corner Table

- Corner is a vertex with one of its incident triangles

Corner - c
Corner Table

• Corner is a vertex with one of its incident triangles

Corner - c
Triangle - c.t
Corner Table

- **Corner** is a vertex with one of its incident triangles

  Corner - c
  Triangle - c.t
  Vertex - c.v
• Corner is a vertex with one of its incident triangles
  Corner - c
  Triangle - c.t
  Vertex - c.v
  Next corner in c.t (ccw) - c.n
• Corner is a vertex with one of its incident triangles

Corner - c
Triangle - c.t
Vertex - c.v
Next corner in c.t (ccw) - c.n
Previous corner - c.p (== c.n.n)
Corner Table

- Corner is a vertex with one of its incident triangles

Corner - c
Triangle - c.t
Vertex - c.v
Next corner in c.t (ccw) - c.n
Previous corner - c.p (== c.n.n)
Corner opposite c - c.o
Edge E opposite c not incident on c.v
Triangle T adjacent to c.t across E
c.o.v vertex of T that is not incident on E
Corner Table

- Corner is a vertex with one of its incident triangles
  - Corner - c
  - Triangle - c.t
  - Vertex - c.v
  - Next corner in c.t (ccw) - c.n
  - Previous corner - c.p (== c.n.n)
  - Corner opposite c - c.o
    - Edge E opposite c not incident on c.v
    - Triangle T adjacent to c.t across E c.o.v
      - vertex of T that is not incident on E
  - Right corner - c.r - corner opposite c.n (== c.n.o)
Corner Table

- Corner is a vertex with one of its incident triangles

  Corner - c
  Triangle - c.t
  Vertex - c.v
  Next corner in c.t (ccw) - c.n
  Previous corner - c.p (== c.n.n)
  Corner opposite c - c.o
    Edge E opposite c not incident on c.v
    Triangle T adjacent to c.t across E
      c.o.v vertex of T that is not incident on E
  Right corner - c.r - corner opposite c.n (== c.n.o)
  Left corner - c.l (== c.p.o == c.n.n.o)
Corner Table

- Corner is a vertex with one of its incident triangles

<table>
<thead>
<tr>
<th>Corner</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle</td>
<td>c.t</td>
</tr>
<tr>
<td>Vertex</td>
<td>c.v</td>
</tr>
<tr>
<td>Next corner in c.t (ccw)</td>
<td>c.n</td>
</tr>
<tr>
<td>Previous corner</td>
<td>c.p (== c.n.n)</td>
</tr>
<tr>
<td>Corner opposite c</td>
<td>c.o</td>
</tr>
</tbody>
</table>
  - Edge E opposite c not incident on c.v
  - Triangle T adjacent to c.t across E
    - c.o.v vertex of T that is not incident on E
| Right corner | c.r     |
| Left corner  | c.l (== c.p.o == c.n.n.o) |
Corner Table

- Corner is a vertex with one of its incident triangles

<table>
<thead>
<tr>
<th>corner</th>
<th>c.v</th>
<th>c.t</th>
<th>c.n</th>
<th>c.p</th>
<th>c.o</th>
<th>c.r</th>
<th>c.l</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>v₁</td>
<td>f₁</td>
<td>c₂</td>
<td>c₃</td>
<td>c₆</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c₂</td>
<td>v₂</td>
<td>f₁</td>
<td>c₃</td>
<td>c₁</td>
<td></td>
<td></td>
<td>c₆</td>
</tr>
<tr>
<td>c₃</td>
<td>v₃</td>
<td>f₁</td>
<td>c₁</td>
<td>c₂</td>
<td>c₆</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c₄</td>
<td>v₃</td>
<td>f₂</td>
<td>c₅</td>
<td>c₆</td>
<td></td>
<td>c₇</td>
<td>c₁</td>
</tr>
<tr>
<td>c₅</td>
<td>v₂</td>
<td>f₂</td>
<td>c₆</td>
<td>c₄</td>
<td>c₇</td>
<td>c₁</td>
<td></td>
</tr>
<tr>
<td>c₆</td>
<td>v₄</td>
<td>f₂</td>
<td>c₄</td>
<td>c₅</td>
<td>c₁</td>
<td></td>
<td>c₇</td>
</tr>
</tbody>
</table>
Corner Table

• Store:
  – Corner table
  – For each vertex - a list of all its corners
• Corner number $j*3-2$, $j*3-1$ and $j*3$ match face number $j$
• What are the vertices of face #3?
  – Check c.v of corners 9, 8, 7
Corner Table

• Are vertices 2 and 6 adjacent?
  – Scan all corners of vertex 2, check if c.p.v or c.n.v are 6
Which faces are adjacent to vertex 3?
- Check c.t of all corners of vertex 3
Corner Table

• One ring neighbors of vertex $v_4$?
  – Get the corners $c_6 \ c_8 \ c_{10}$ of this vertex
  – Go to $c_i.n.v$ and $c_i.p.v$ for $i = 6, 8, 10$.
  – Remove duplicates
Corner Table

• Pros:
  – All queries in $O(1)$ time
  – Most operations are $O(1)$
  – Convenient for rendering

• Cons:
  – Only triangular, manifold meshes
  – Redundancy
Multiple Simplification
Applications

• Oversampled 3D scan data

~150k triangles

~80k triangles
Applications

- Overtessellation: E.g. iso-surface extraction
Applications

• Multi-resolution hierarchies for
  – efficient geometry processing
  – level-of-detail (LOD) rendering
Applications

• Adaptation to hardware capabilities
Size-Quality Tradeoff
Problem Statement

• Given: $M = (V,F)$
• Find: $M' = (V',F')$ such that
  – $|V'| = n < |V|$ and $d(M, M')$ is minimal, or
  – $d(M, M') < \epsilon$ and $|V'|$ is minimal

• Respect additional fairness criteria
  – Normal deviation, triangle shape, scalar attributes, etc.
Mesh Decimation Methods

• Vertex clustering

• Incremental decimation

• Remeshing
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
  – Hierarchical approach
  – Top-down or bottom-up

• 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 $q$

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

\[
dist(q, p)^2 = (q^T p)^2 = p^T (qq^T) p =: p^T Q_q p
\]

\[
Q_q = \begin{bmatrix}
    a^2 & ab & ac & ad \\
    ab & b^2 & bc & bd \\
    ac & bc & c^2 & cd \\
    ad & bd & cd & d^2
\end{bmatrix}
\]
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:

$$\begin{bmatrix}
q_{11} & q_{12} & q_{13} & q_{14} \\
q_{21} & q_{22} & q_{23} & q_{24} \\
q_{31} & q_{32} & q_{33} & q_{34} \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
p^*_1 \\
p^*_2 \\
p^*_3 \\
p^*_4
\end{bmatrix} =
\begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix}$$
Comparison

average

median

error quadric
Vertex Clustering

- Cluster Generation
- Computing a representative

- Mesh generation
  - Clusters $p \leftrightarrow \{p_0, \ldots, p_n\}$, $q \leftrightarrow \{q_0, \ldots, q_m\}$

- Topology changes
Vertex Clustering

• Cluster Generation

• Computing a representative

• Mesh generation
  – Clusters $p\rightarrow \{p_0, \ldots, p_n\}$, $q\rightarrow \{q_0, \ldots, q_m\}$
  – Connect $(p, q)$ if there was an edge $(p_i, q_i)$

• Topology changes
Vertex Clustering

- Cluster Generation
- Computing a representative
- Mesh generation

- Topology changes
  - If different sheets pass through one cell
  - Can be non-manifold
Incremental Decimation
Incremental Decimation

500K

50K

5K

0.5K
Incremental Decimation

• General Setup

• Decimation operators

• Error metrics

• Fairness criteria
General Setup

• Repeat:
  – Pick mesh region
  – Apply decimation operator

• Until no further reduction possible
Greedy Optimization

• For each region
  – evaluate quality after decimation
  – enqueue(quality, region)

• Repeat:
  – get best mesh region from queue
  – apply decimation operator
  – update queue

• Until no further reduction possible
Global Error Control

• For each region
  – evaluate quality after decimation
  – enqueue(quality, region)

• Repeat:
  – get best mesh region from queue
  – If error < eps
    • Apply decimation operator
    • Update queue

• Until no further reduction possible
Incremental Decimation

- General Setup
- Decimation operators
- Error metrics
- Fairness criteria
Decimation Operators

• What is a “region”?

• What are the DOF for re-triangulation?

• Classification
  – Topology-changing vs. topology-preserving
  – Subsampling vs. filtering
  – Inverse operation -> progressive meshes [Hoppe et al....]
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

- Merge two adjacent vertices
- Define new vertex position
  - Continuous 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
Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...

[Diagram of a polygon with black and blue lines]
Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...

\[
\frac{r_1}{e_1} < \frac{r_2}{e_2}
\]
Incremental Decimation

- General Setup
- Decimation operators
- Error metrics
- Fairness criteria
Local Error Metrics

- Local distance to mesh
  - Compute average plane
  - No comparison to *original* geometry
Global Error Metrics

- Error quadrics
  - Squared distance to planes at vertex
  - No bound on true error

\[ p_i^T Q_i p_i = 0, \quad i = \{1, 2\} \]

\[ Q_3 = Q_1 + Q_2 \]

Solve \[ v_3^T Q_3 v_3 = \min \]

\[ < \varepsilon \quad \rightarrow \quad \text{ok} \]
Incremental Decimation

- General Setup
- Decimation operators
- Error metrics
- Fairness criteria
Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...

[3D model of a meshed object]
Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...

Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...


Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...

Fairness Criteria

• Rate quality of decimation operation
  – Approximation error
  – Triangle shape
  – Dihedral angles
  – Valence balance
  – ...

[Diagram of geometric shapes]
Comparison

• Vertex clustering
  – fast, but difficult to control simplified mesh
  – Topology changes, non-manifold meshes
  – Global error bound, but often not close to optimum

• Incremental decimation with quadratic error metrics
  – good trade-off between mesh quality and speed
  – explicit control over mesh topology
  – restricting normal deviation improves mesh quality
Remeshing
Remeshing

Given a 3D mesh, find a “better” discrete representation of the underlying surface
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
What is a good mesh?

• Equal edge lengths
• Equilateral triangles
• Valence close to 6
• Uniform vs. adaptive sampling
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
- Alignment to curvature lines
- Isotropic vs. anisotropic
What is a good mesh?

- Equal edge lengths
- Equilateral triangles
- Valence close to 6
- Uniform vs. adaptive sampling
- Feature preservation
- Alignment to curvature lines
- Isotropic vs. anisotropic
- Triangles vs. quadrangles