Compression of Triangle Meshes

Geometry Processing
CS 7960

Louis Bavoil
01/17/2006
Outline

• Triangle Meshes
• Compression Techniques
• Edgebreaker for simple meshes
Triangle Meshes

- Mesh = Geometry + Connectivity
  - Geometry = vertices \(\{V\}\)
  - Connectivity = triangles \(\{T\}\)
Uncompressed storage

- Sequence of vertices
  - Triangle 1: XYZ XYZ XYZ
  - Triangle 2: XYZ XYZ XYZ
  - ...

Bytes per triangle: 9x4 = 36 B/T
Assuming 32-bit float per component
Uncompressed storage

- Sequence of vertices
  Bytes per triangle: 9x4 = 36 B/T

- Vertex coordinates and indices
  - Vertex 1: XYZ
  - Vertex 2: XYZ
  - ...
  - Triangle 1: 1 2 3
  - Triangle 2: 2 3 4
  - ...

  Bytes per triangle: 3x4 B/V + 3x4 B/T ≈ 18 B/T

  Assuming |T| ≈ 2|V| and 32-bit integers
Outline

• Triangle Meshes
• Compression Techniques
• Edgebreaker for simple meshes
Motivation

• Bandwidth
  – Can’t store all 3D models locally
    • Remote visualization
    • Massively multiplayer games

• Storage
  – Model complexity grows faster than bandwidth
    • 3D scanners, iso-surface extraction, etc.
Goal

• Compression:
  – Lossless Compression

• In general:
  – Geometry compression is lossy (quantization)
  – Connectivity compression is lossless (unlike simplification)
Vertex/triangle ratio

Euler’s formula:

\[ |T| - |E| + |V| = 2 - 2g - b \]

- \(|T|\) = num triangles
- \(|E|\) = num edges
- \(|V|\) = num vertices
- \(g\) = genus (number of holes)
- \(b\) = number of boundaries
Vertex/triangle ratio

Euler’s formula:

$$|T| - |E| + |V| = 2 - 2g - b$$

Simple Meshes

Homeographic to a half-sphere

- \( g = 0 \) (no hole)
- \( b = 1 \) (one boundary)

\[ \Rightarrow |T| - |E| + |V| = 1 \]
Vertex/triangle ratio

$$|V| = |V_I| + |V_E|$$

$V_E =$ exterior vertex (on a boundary)

$V_I =$ interior vertex

Num external edges = $|V_E|$

Num internal edges = $(3|T| - |V_E|)/2$

$|V_E| = 5$

$|T| = 3$
Vertex/triangle ratio

\[
\begin{align*}
|V| &= |V_I| + |V_E| \\
|E| &= |V_E| + 3|T|/2 + |V_E|/2 \\
|V| - |E| + |T| &= 1
\end{align*}
\]

\[\Rightarrow |T| = 2 |V_I| + |V_E| - 2\]

When \(|V_E| \ll |V_I|\)

\[\Rightarrow |T| \approx 2 |V|\]
Compression

Triangle Mesh

Geometry

Connectivity

|T| = 2|V|

1. Quantization
2. Vertex prediction
3. Entropy coding

Mesh compression
Compression

Triangle Mesh

Geometry
1. Quantization
2. Vertex prediction
3. Entropy coding

Connectivity

$|T| = 2|V|$

Mesh compression
Vertex Compression

• Quantization
  – Instead of using a 32-bit floating point per vertex coordinate, use 16 bit floats or less
  – Use local coordinates relative to the bounding box of object
    • Store the min and max points
    • Scale by \([\text{maxPoint}-\text{minPoint}]\)
    • Translate to \text{minPoint}
Vertex Compression

• Vertex prediction
  – Encoder and decoder use the same predictor
  – Transmit residues (difference \([\text{predicted} - \text{actual}]\))
  – Regular meshes \(\Rightarrow\) small residues

• Vertex predictors
  – Linear combination of 4 ancestors
  – Parallelogram predictor

Example of parallelogram prediction
Vertex Compression

• Entropy Coding
  – Good prediction => residues clustered around zero
  – Compress the residues using Huffman’s code
  – Code most frequent residues with fewer bits

• Example of Huffman coding:
  A=>0 (most frequent symbol/residue)
  B=>10 (second most frequent)
  C=>110…
Bit-precision storage

• Sequence of vertices
  – Constant number of bits per triangle

• Vertices and indices
  – Connectivity = $3 \cdot \log_2 |V| \text{ b/T}$
  – $\log |V| = \text{bad}$
Triangle Strips

- One integer per triangle instead of 3
  - OpenGL strip \[ \Rightarrow 1 + 0.5 \log|V| \frac{b}{T} \]
  - Generalized strip \[ \Rightarrow 2 + 0.5 \log|V| \frac{b}{T} \]
## Connectivity Compression

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Compression (b/T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle Strips</td>
<td>$1 + 0.5 \log</td>
</tr>
<tr>
<td>Deering 1995</td>
<td>$3.75 + 0.062 \log</td>
</tr>
<tr>
<td>Cache 16 decoded vertices</td>
<td></td>
</tr>
<tr>
<td>Gumhold and Strasser 1998</td>
<td>$1.7-2.15 + k.\log</td>
</tr>
<tr>
<td>“Cut-Border”, like Edgebreaker</td>
<td></td>
</tr>
<tr>
<td>Taubin and Rossignac 1998</td>
<td>$&lt; 3$</td>
</tr>
<tr>
<td>“Topological Surgery”</td>
<td></td>
</tr>
<tr>
<td>Touma and Gotsman 1998</td>
<td>$&lt; 1$ for regular meshes</td>
</tr>
<tr>
<td>Valence-based encoding</td>
<td>May exceed 2</td>
</tr>
<tr>
<td>Edgebreaker 1999</td>
<td>$&lt; 2$ for simple meshes</td>
</tr>
</tbody>
</table>
Compression

Triangle Mesh

Geometry

Connectivity

1. Quantization
2. Vertex prediction
3. Entropy coding

Mesh compression

$|T| = 2|V|$
Outline

• Triangle Meshes
• Compression Techniques
• Edgebreaker for simple meshes
Edgebreaker

• Compression
  - Traverse the triangles in a spiraling order
  - At each step, remove a triangle, and encode:

Output streams:
- Vertex indices as they are visited
- CLERS string encodes connection of triangle with current edge (gate) => CLERS op-codes

Connectivity → Edgebreaker → Vertex indices, CLERS string

SLCCRRCCRRR…
Edgebreaker

• Decompression
  – Start with the same initial edge as for compression
  – Parse the CLERS string and read vertex indices in the same order as compression

• Edgebreaker is a state machine
  – Current gate = current reference edge
  – Current vertex id = next id in vertex stream
Assumptions

• Triangle Mesh
• Manifold with boundary
  – Definition: Every vertex has a neighborhood homomorphic to an open disk or half disk
  – Property: Each edge is bounding one (boundary) or two triangles (interior)
Assumptions

• Triangle Mesh
• Manifold with boundary
• Orientable

Compatible triangles $\Leftrightarrow$ Opposite orientations on common edges
Assumptions

• Triangle Mesh
• Manifold with boundary
• Orientable
Mesh Data Structures

• How to traverse a mesh?
Mesh Data Structures

• How to traverse a mesh?
  – Half-edge data structure
  – Corner Table
Half-edge Data Structure

• Half-edge
  – Edge e \(\Leftrightarrow\) Incident triangle upon e
  – Useful to walk on a mesh

\[
\begin{align*}
\text{h.s} & = \text{opposite half-edge} \\
\text{h.p} & = \text{previous half-edge in triangle} \\
\text{h.n} & = \text{next half-edge in triangle} \\
\text{h.o} & = \text{h.s} \\
\text{h.e} & = \text{h.p} \\
\text{h.v} & = \text{h.n.n.s}
\end{align*}
\]
Two arrays of integers: V and O
Size of each array = 3|T|
Not used for storage, just to walk on a mesh

c.v = V[c]
c.o = O[c]
c.n.v = V[3 c.t + ((c+1) MOD 3)]
c.p.v = V[3 c.t + ((c+2) MOD 3)]
c.l = c.p.o
c.r = c.n.o
Example of Corner Table

# Mesh Data Structures

## Table

<table>
<thead>
<tr>
<th>Half-Edge</th>
<th>Corner Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990’s</td>
<td>Rossignac, et. al. 2001: “3D compression made simple: Edgebreaker on a Corner Table”</td>
</tr>
<tr>
<td>Edge (\Leftrightarrow) Incident Triangle</td>
<td>Vertex (\Leftrightarrow) Incident Triangle</td>
</tr>
<tr>
<td>Every <strong>half-edge</strong> has two vertices and an <strong>opposite half-edge</strong></td>
<td>Every <strong>corner</strong> has a vertex and an <strong>opposite corner</strong> associated to it</td>
</tr>
<tr>
<td>Pointers</td>
<td>Two fixed-size arrays of integers (V and O)</td>
</tr>
</tbody>
</table>
Edgebreaker Compression

• Original version 1999
  – Uses half-edges

• Contributions:
  – Guaranteed < 3 bits / triangle (connectivity)
  – Simple to implement for simple meshes
  homeomorphic to a sphere or to a half-sphere

• Issues:
  – Complex for general triangle meshes
  – Requires two passes to decompress
  – Requires a vertex search for each S symbol
CLERS op-codes

Vertices and triangles can be marked as seen:
\[ v.m = \text{boolean}; \ t.m = \text{boolean} \]
Initially, all \( v.m = 0 \) and \( t.m = 0 \)

\[ \text{C Gate} \]

Red = already visited/mark
Black = not visited yet
CLERS op-codes

- The S and E operations work like parentheses => require a stack

1. Push gate position
2. Explore to the right
3. Then explore to the left

1. Pop
2. If stack empty, return
Encoding

• CLERS => binary
  C = 0
  S = 100
  R = 101
  L = 110
  E = 111
Decompression

• **Multiple Algorithms:**
  – Original Edgebreaker
    • Requires 2 passes to decompress
  – Wrap and Zip
    • Can be done 1 one pass (zip on the fly)
      – Rossignac and Szymczak, "Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker“, 1999
      – Rossignac et. al., “Edgebreaker on a Corner Table”, 2001
  – Spiral Reversi
    • Parse the CLERS string backwards from the end and builds the triangle tree from the leaves. Single pass
Wrap and Zip

• Decompression works by first generating a tree of adjacent triangles.
• Then the free edges of triangles are given an orientation (wrapping).
• Finally similarly oriented edges are joined (zipping).
Wrap and Zip

How to Wrap:

C  L  E  R  S
Wrap and Zip

How to Zip:

• Vertices with adjacent edges pointing inwards should merge those edges.
• Repeat until all edges are zipped.