Optimized Prediction for Geometry Compression of Triangle Meshes

Dan Chen
Yi-Jen Chiang
Nasir Memon
Xiaolin Wu
Polytechnic University, NY, USA
(DCC 05, March 2005)

Graphics Compression for 3D Triangle Meshes

- Graphics compression is an emerging need for storing, transmitting, and visualizing large graphics models.
- 3D triangle mesh:
  - The most common type of graphics models
  - Two components of information:
    - geometry – 3D coordinates of mesh vertices
    - connectivity – edges & triangles connecting vertices

Previous Work

- Lots of results in connectivity compression... (see paper)
- Best connectivity compression results: 1.5 – 4 bits per vertex on an average
  - e.g. [Taubin-Rossignac 98], [Touma-Gotsman 98], [Rossignac 99], [Alliez-Desbrun 01]
- Geometry compression results are not equally impressive
  - Usually quantize each coordinate to a 10-bit or 12-bit integer (30 or 36 bits/vertex in raw data)
    - Typical results: 40—50% of raw data (12—18 bits/vertex)
      e.g. [Deering 95], [Karni-Gotsman 00], [Taubin-Rossignac 98], [Touma-Gotsman 98]

Previous Work: Geometry Compression (1)

- Flipping method [Touma-Gotsman 98]
  - Dominant, widely considered state of the art: adopted to the MPEG-4 standard for mesh geometry coding
  - Traverse triangles by connectivity coder; predict new vertex position of new triangle by flipping using parallelogram rule
  - Drawback: triangle traversal ignores the geometry of the model

- Other extensions of flipping
  - [Isenburg-Alliez 02]: beyond triangle meshes
  - [Isenburg-Gumbhold 03]: out-of-core method for meshes larger than main memory
  * Do not address the drawback

Geometry compression is by far the dominating bottleneck!!
Previous Work: Geometry Compression (2)

- Prediction tree method [Kronrod-Gotsman 02]
  - Only previous work trying to optimize the flipping prediction error
  - Formulate the problem as finding an optimal cover tree
  - Take the dual graph of the triangle mesh, span the mesh triangles (nodes in dual graph) until all vertices are covered, with min total dual-edge cost (prediction error)
  - Heuristic solution; improves the flipping approach

- Sub-optimal:
  - May cover vertices more than once
  - Cannot visit a triangle from a vertex-adjacent neighbor

Our New Algorithm

- Try to optimize the flipping prediction error
  - New formulation: finding a constrained minimum spanning tree on a new graph G (G is not the dual graph)
  - Span each vertex exactly once (vs. cover more than once)
  - Can visit a triangle from vertex-adjacent neighbor (vs. cannot)
  - Improves the prediction tree method by up to 33.2%

- Overview: 3 major technical components
  - Problem formulation: finding a constrained minimum spanning tree (CMST) on the graph G
  - Heuristic algorithm to find an approximate CMST on G
  - Algorithm to traverse CMST in another pass, build a pseudo-CMST & collect left-over triangles in the same pass, and finish both geometry and connectivity coding

Problem Formulation

- Observation: many possible ways of flipping for a vertex
  - Each flipping pair \((x, y)\) gives a possible flipping

- Form a graph \(G\):
  * nodes—mesh vertices; edges—connect all flipping pairs, edge cost = prediction error
  * \((y', y) = (x', x) \Rightarrow G \text{ is undirected} \)* minimum spanning tree on \(G\)
Final Problem Formulation

- In graph $G$, each edge $(x, y)$ has constraint vertices $a, b$
- Constrained minimum spanning tree $T$ on $G$: $T$ admits a traversal where each $(x, y)$ is visited only after visiting $a, b$

![Example of CMST $T$]

Heuristic Algorithm for CMST

- Modify Prim’s algorithm for an approx. CMST $T$
  - For each edge $(x, y)$ of $G$, make bidirectional links between $(x, y)$ and its constraint vertices $a, b$
  - Initially, include 3 vertices of a triangle to $T$ for initial prediction
  - Use a priority queue $Q$ to maintain vertices not yet added to $T$
  - Key $(x)$: min cost of adding $x$ to $T$, initially infinity; key $(x) \leftarrow \min \{ \text{cost} (x, y) \mid (x, y) \text{ is valid, i.e., } y, a, b \text{ already in } T\}$
  - While $Q$ is not empty do
    - $v \leftarrow \text{Extract-min} (Q)$; include $v$ to $T$
    - Update key values of vertices influenced by $v$
      - candidates for newly valid edges: (i) edges incident on $v$; (ii) edges with $v$ a constraint vertex
  - If key $(v) = \text{infinity}$, then start a new tree (rarely occurred)
  - Cost $(T)$ is very close to the cost of unconstrained MST (unachievable lower bound)

Pseudo-CMST and Final Encoding

- The approx. CMST $T$ admits a valid traversal by the order we grow $T$
- This order grows the boundary of the patch of current $T$ arbitrarily---very expensive to encode

- Idea: each triangle has at most 3 edges to flip
- Traverse $T$ in another pass; build a pseudo CMST $T_p$ & collect left-over triangles
  - (i) recursively traverse $t_1$; (ii) recursively traverse $t_2$; (iii) collect $t_3, t_4$ if all vertices visited
- Step (i): if $t_1$ is visited, ignore $t_1$; else
  - If $v$ unvisited: (a) $e \text{ in } T$: predict $v$ by $e$, add $v, e$ to $T_p$, recurse from $t_1$
    (b) $e \text{ not in } T$: ignore ($v, t_1$ will be visited later by other paths)
  - If $v$ visited: add $e$ to $T_p$ with no cost (pseudo-edge), recurse from $t_1$

Summary: Algorithm Steps

1. Form graph $G$
2. Compute an approximate CMST
3. Compute a pseudo-CMST & collect left-over triangles, finish geometry & connectivity coding
Experiments

- 12 datasets commonly used in literature
  - size: small --- moderately large
  - feature: smooth --- with significantly many sharp corners
- Vertex coordinates are quantized to 12-bit integers
- Compare first-order entropy of prediction errors of:
  - constrained MST (CMST) vs. unconstrained MST (lower bound, though unachievable)
  - pseudo-CMST vs. flipping [Touma-Gotsman 98] (code available from web)
  - prediction tree [Kronrod-Gotsman 02] (from paper)

Datasets (1)

Datasets (2)

Results: Statistics Summary

- CMST vs. unconstrained MST (lower bound):
  - In most cases: CMST is within 10% of MST
  - On an average: within 17.4%
- Pseudo-CMST vs. flipping & prediction tree (PT):
  - Pseudo-CMST: 8.2—20.41 bits per vertex (b/v)
    Cf. original: 36 b/v
  - Gain over flipping: up to 55.45% (> 32% on an average)
  - Gain over PT: up to 33.17% (> 18% on an average)
  - Also, Pseudo-CMST is very close to original CMST
Conclusions

- Novel geometry compression technique via optimized flipping prediction
- Novel problem formulation & optimization methods
- Geometry oriented, integrating both geometry & connectivity coding
- Large improvements:
  55.45% over flipping; 33.17% over prediction tree

Extension

Tetrahedral meshes (volume data)
[Chen-Chiang-Memon-Wu]

Open Problem


Acknowledgments

- C. Touma and C. Gotsman for the Flipping code
- C. Gotsman, Princeton Graphics Database and Stanford Graphics Lab for the test datasets

- National Science Foundation (NSF)
  (CAREER CCR-0093373, ACI-0118915, ITR CCR-0081964, CCR-0208678)