Robust On-line Computation of Reeb Graphs: Simplicity and Speed

Valerio Pascucci
Giorgio Scorzelli
Peer-Timo Bremer
Ajith Mascarenhas

CASC - LLNL
The Reeb Graph Is the Topological Skeleton of a Geometric Model
The Reeb Graph Is the Contraction of Isocontour Components to Points

- Given a mesh.
The Reeb Graph Is the Contraction of Isocontour Components to Points

- Given a mesh and a function defined on it.
The Reeb Graph is the Contraction of Isocontour Components to Points

- Given a mesh and a function defined on it.
- Consider an isocontour.
The Reeb Graph Is the Contraction of Isocontour Components to Points

- Given a mesh and a function defined on it.
- Consider an isocontour.
The Reeb Graph Is the Contraction of Isocontour Components to Points

- Given a mesh and a function defined on it.
- Consider an isocontour and contract each component.
The Reeb Graph Is the Contraction of Isocontour Components to Points

- Given a mesh and a function defined on it.
- Consider an isocontour and contract each component.
- Repeat for all contours while maintaining adjacency.
The Reeb Graph Is a Fundamental Descriptor of the Topology of Shapes

- Shape matching [Hilaga et al. SIGGRAPH’01]
- Mesh repair [Wood et al. ToG’04]
- Mesh parameterization [Zhang et al. ToG’05]
- Shape skeletons [Lazarus et al. SM’02]
- Morphing [Lee et al. CA&VW ’06]
- Data analysis [Laney et al. Vis’06]
Previous Methods Trade Generality for Better Worst Case Complexity

- Construction from slices [Shinagawa et al. CG&A‘91]
- Contour Trees [Carr et al. SODA ‘00]
- Loops in Reeb graphs [Cole-McLaughlin SoCG’03]
We Propose an Approach Aiming at Generality and Practical Performance

- Input model can be non-manifold.
- Input model of any dimension.
- No need to reorder the input triangles.
- Run in out of core mode.
- Multi-resolution representation.

St. Mathew 372Mt, 486s, 8MB.
Update of the Reeb Graph After Insertion of Each New Element

- Consider a 2D model of known Reeb graph (initial condition = empty graph):
  - Add new vertex.
  - Add new edge.
  - Add new triangle.
On-line Update of the Reeb Graph After Insertion of Each New Element

- maximum
- saddle
- minimum
On-line Update of the Reeb Graph After Insertion of Each New Vertex
On-line Update of the Reeb Graph
After Insertion of Each New Edge
On-line Update of the Reeb Graph After Insertion of Each New Edge
On-line Update of the Reeb Graph After Insertion of Each New Edge
On-line Update of the Reeb Graph After Insertion of Each New Edge
On-line Update of the Reeb Graph After Insertion of Each New Edge
On-line Update of the Reeb Graph After Insertion of Each New Triangle
On-line Update of the Reeb Graph After Insertion of Each New Triangle
On-line Update of the Reeb Graph After Insertion of Each New Triangle
On-line Update of the Reeb Graph After Insertion of Each New Triangle
The Approach is Based on a Few Simple Assumptions on the Input

- Sequence of vertices, edges, and triangles.

- **Indexed mesh**: triangles/edges defined as references to vertices.

- **Finalization**: each vertex knows when it is referenced for the last time (if needed a pass can collect this information easily and fast).
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph

E1 → A0
E4 → A0
E3 → A2
A0 → E1,E4
A2 → E3,E4
Data Structure for Incremental Update of the Reeb Graph

- E1 → A0
- E4 → A0
- E3 → A2
- A0 → E1, E4
- A2 → E3, E4
Data Structure for Incremental Update of the Reeb Graph

E1 → A0
E3 → A2
E4 → A0

A0 → E1, E4
A2 → E3, E4
Data Structure for Incremental Update of the Reeb Graph

E1 $\rightarrow$ A0
E3 $\rightarrow$ A2
E4 $\rightarrow$ A0
E2 $\rightarrow$ A1

A0 $\rightarrow$ E1, E4
A2 $\rightarrow$ E3, E4
A1 $\rightarrow$ E2
Data Structure for Incremental Update of the Reeb Graph

E1 → A0

E4 → A0

E2 → A1

E3 → A2

A0 → E1,E4

A2 → E3,E4

A1 → E2
Data Structure for Incremental Update of the Reeb Graph

E1 → A0
E2 → A1
E3 → A2
E4 → A0
E0 → A4

A0 → E1, E4
A2 → E3, E4
A1 → E2
A4 → E0
Data Structure for Incremental Update of the Reeb Graph

E1 → A0 → E0 → A4
E4 → A0
E3 → A2 → E2 → A1
A4 → E0
A0 → E1, E4
A2 → E3, E4
A1 → E2
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph

E1 → A4
E0 → A4
E4 → A4
E2 → A0
E3 → A2
A4 → E0, E1, E4
A0 → E1, E2, E4
A2 → E3, E4
A1 → E2
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph

- E1 → A4
- E0 → A4
- E3 → A2
- E2 → A0
- A4 → E0, E1
- A0 → E1, E2
- A2 → E2, E3
Data Structure for Incremental Update of the Reeb Graph
Data Structure for Incremental Update of the Reeb Graph

E1 → A4

E3 → A2

A4 → E1

A0 → E1

A2 → E3
Data Structure for Incremental Update of the Reeb Graph
In Out of Core Mode Unnecessary Elements Are Removed from Memory

- From input mesh:
  - Never store triangles.
  - Remove finalized vertices.
  - Remove edges with finalized vertices.

- From Reeb graph:
  - Retire arcs without edge reference.
  - Retire nodes without edge reference and adjacent arcs.
The Input Triangles Do Not Need to Be in Any Particular Order

- Dancer model with 50K triangles.

- Sorted by Z coordinate: 0.11s, 1.8MB.
- Original mesh: 0.2s, 1.8MB.
We Exploit the Locality of an Input Mesh in Cache Friendly Format

David model: 56Mt, 108s, 2.1MB
The Reeb Graph Can Be Simplified by Removing Branches and Loops

Branch: extremum-saddle

Loop: saddle-saddle
The Reeb Graph Can Be Simplified by Removing Branches and Loops
The User Can Choose the Level of Resolution for the Reeb Graph
The Simplification Can Be Used to Develop Shape Signatures
The Scheme Computes Reeb Graphs for Meshes of Any Dimension
The Scheme Computes Reeb Graphs for Meshes of Any Dimension
The Scheme Computes Reeb Graphs for Meshes of Any Dimension

- 2-skeleton
- level sets
- 1-skeleton
- Reeb graph
- cut-away view
The Scheme Computes Reeb Graphs for Meshes of Any Dimension
The Scheme Computes Reeb Graphs for Meshes of Any Dimension

2D:

Sierpinski simplex

3D:

4D:

5D:
Practical Tests Confirm Robustness and Show Good Scalability