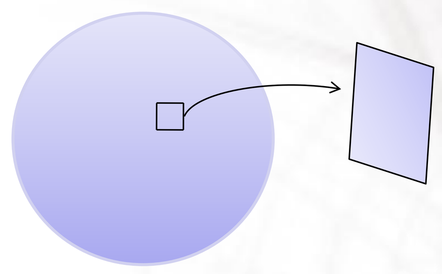
techvista CHENNAI 2008

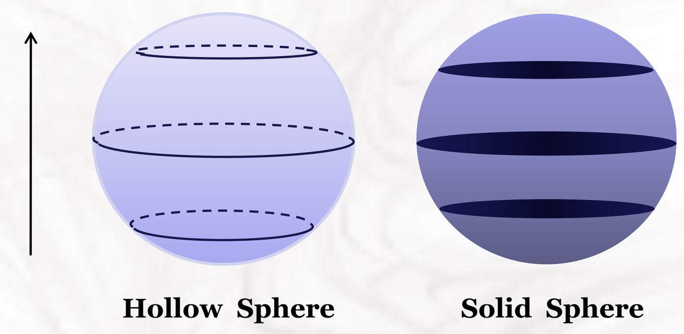
Microsoft Research india

d-manifold



Neighbourhood of every point resembles the Euclidean Space

Level Sets



The preimage of a real value is called a level set

Morse Function

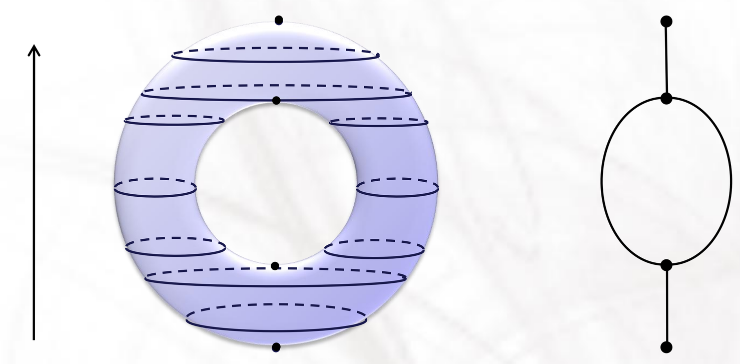
A real valued function defined on a d-manifold, such that

- There are no degenerate critical points
- No two critical points have the same value

Critical Points

Critical Points of a smooth function are points where the gradient becomes zero

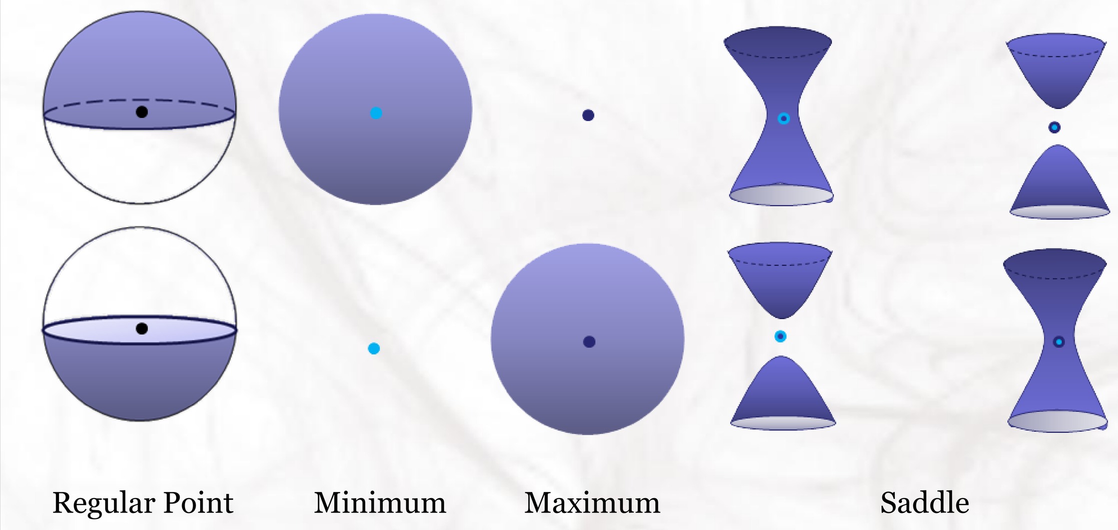
Reeb Graph



The Reeb graph of a function is obtained by contracting each connected component of a level set to a point.

The Reeb graph expresses the evolution of connected components of level sets as a graph whose nodes correspond to critical points of the function.

Critical Vertices



Behaviour of isosurfaces at various types of vertices

Reeb Graphs

Harish Doraiswamy Vijay Natarajan

Visualization & Graphics Lab
Dept. Of Computer Science & Automation
Indian Institute Of Science

Abstract

Isosurface or level sets are used extensively to visualize three and higher dimensional scientific data. The Reeb graph tracks topology changes in level sets of a scalar function, and therefore serves as a useful user interface for selecting meaningful level sets. Besides visualization, Reeb graphs also find applications in geometric modeling and shape matching. We describe two algorithms for constructing the Reeb graph of a smooth function defined over manifolds in any dimension. The first algorithm maintains connected components of level sets as a dynamic graph and constructs the Reeb graph in $O(n\ log(n) + n\ log(g)(log\ log(g))^3)$ time for three-dimensional input, where n is the number of triangles in the tetrahedral mesh representing the input volume and g is the maximum genus over all level sets of the function. Our algorithm extends to higher dimensions where we construct Reeb graphs in $O(n\ log(n)(log\ log(n))^3)$ time. This is a significant improvement over the previously known $O(n^2)$ algorithm.

In a complementary approach, we design a near-optimal two step algorithm that is simple and easy to implement. This algorithm identifies critical points of the input function in the first step, and connects the critical points in the second step to obtain the Reeb graph. Experimental results show that our two-step algorithm is an order of magnitude faster than existing methods. We also develop methods to simplify the Reeb graph, which aids in removing noise and unimportant features from the input, and produce a feature-directed layout of the Reeb graph, which helps users explore their data effectively.

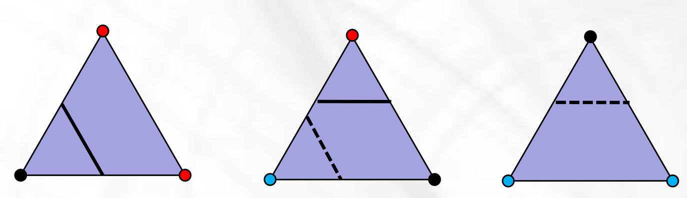
The Sweep Algorithm

(The conventional approach)

1. Sort vertices in increasing order of function value
2. **Starting from the smallest vertex, maintain the level set at a function value infinitesimally above the vertex under consideration**
3. Update the Reeb graph depending on the changes in components of the level set

Mantaining Level Sets

Edges of the level sets



- Vertex under consideration
- Vertex with lower function value
- Vertex with higher function value

—— Edge to be added to the level set
------ Edge to be removed from the level set

Given an input mesh with n triangles, $2n$ insert and delete operations are performed to maintain the level sets by the sweep algorithm.

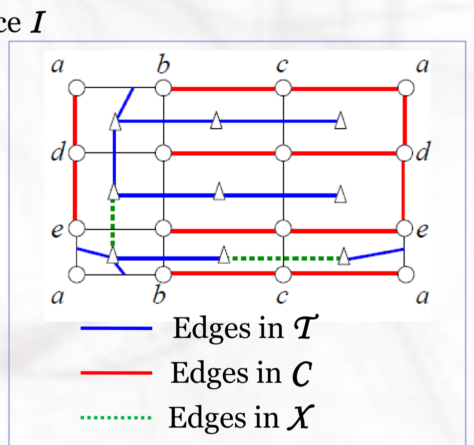
3-manifolds

We use a tree-cotree partition $(\mathcal{T},\mathcal{C},\mathcal{X})$ to store the isosurface I [Eppstein, 2003]

$\mathcal{T}$ – Minimum spanning tree
$\mathcal{C}$ – Maximum spanning Cotree
$\mathcal{X}$ – The edges of I not in $\mathcal{T}$ and $\mathcal{C}$ (size equal to twice the genus of the surface)

- Supports insert and delete operations in $O(log\ n)$ and $O(log\ n + log(g)(log\ log(g))^3)$ time respectively.

Sweep algorithm for 3-manifolds runs in $O(n\ log(n) + n\ log(g)(log\ log(g))^3)$ time



Tree-Cotree partition of a torus that is cut open

d-manifolds

The fully-dynamic graph connectivity algorithm is used to maintain the level sets of a d-manifold [Thorup, 2000]

- Supports insert and delete operations in $O(log(n)(loglog(n))^3)$ time respectively.
- Supports connectivity queries in $O(log(n)/logloglog(n))$ time respectively.

Sweep algorithm for 3-manifolds runs in $O(n\ log(n)(log\ log(n))^3)$ time

Two-Step Algorithm

1. Find the critical points of the input mesh
2. Connect the critical points to obtain the Reeb graph

Step I – Finding Critical Points

Link of a vertex – Mesh induced by its adjacent vertices
Lower Link – Mesh induced by its adjacent vertices that have a lower function value
Upper Link – Mesh induced by its adjacent vertices that have a higher function value

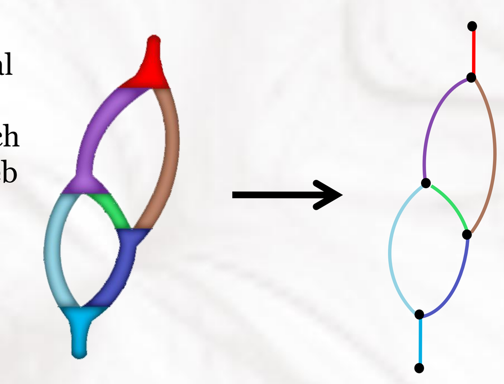
A vertex is **regular** if its upper link and lower link have 1 component. All other vertices are **critical**

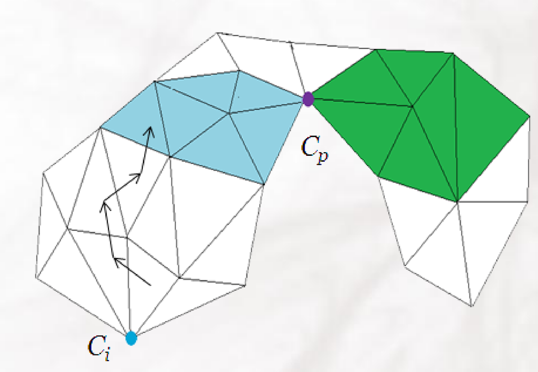
a. Compute the upper and lower link for each vertex in the input mesh
b. Classify the vertex as regular or critical depending on the number components in its upper and lower link

Step II – Connecting Critical Points

We map each arc of the Reeb graph to an **interval volume** between two critical level sets

a. Compute the critical level set for each critical point
b. Trace the interval volumes starting from each critical level set to obtain the arcs in the Reeb graph





Tracing the interval volumes is accomplished by using the **ls-graph** - a dual graph that stores the adjacencies between triangles. The edges in this graph traces the level set components as function value is increased.

Running Time – $O(n + \ell + t\ log\ t)$

n – number of triangles in the input
ℓ – size of critical level sets
t – number of critical points

Near-Optimal
Size of ℓ is usually $O(n)$ in practice, hence running time close to the lower bound $O(n + t\ log\ t)$

Generic
Works without modification for d-manifolds ($d \geq 2$), and for non-manifold meshes

Output Sensitive
The running time depends of the number of critical points of the function, and the size of critical level sets, which is indicative of the importance of features in the data

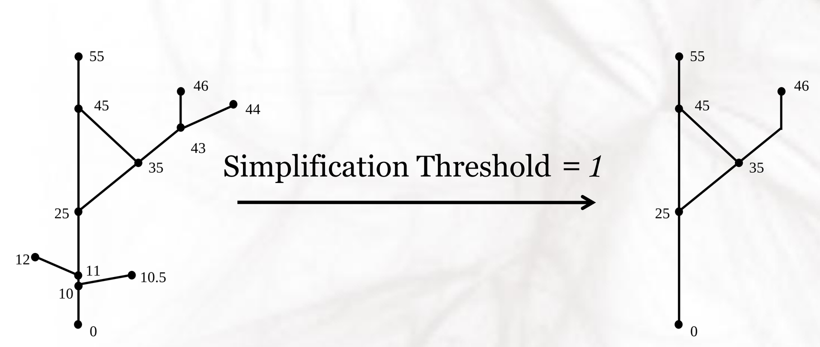
Reeb Graph Simplification

We simply the Reeb graph using the notion of **Extended Persistence** – which denotes the life time of a level set component

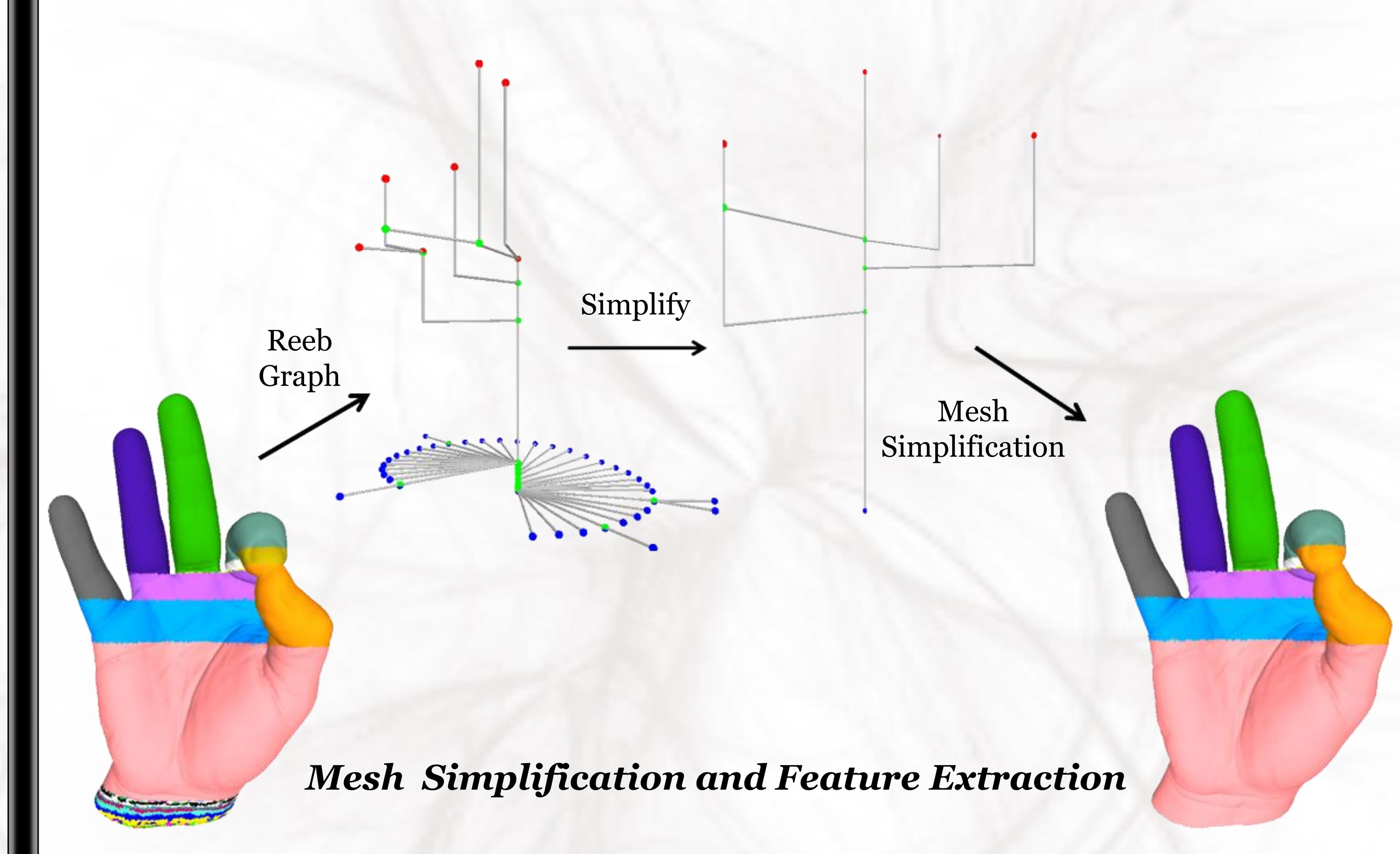
Simplification Procedure

1. Repeat until Reeb graph cannot be simplified
   a. While there exists a leaf/loop that can be pruned
      i. Prune the leaf/loop
   b. Remove every degree 2 node

The leaves/loops that can be simplified are stored in a priority queue

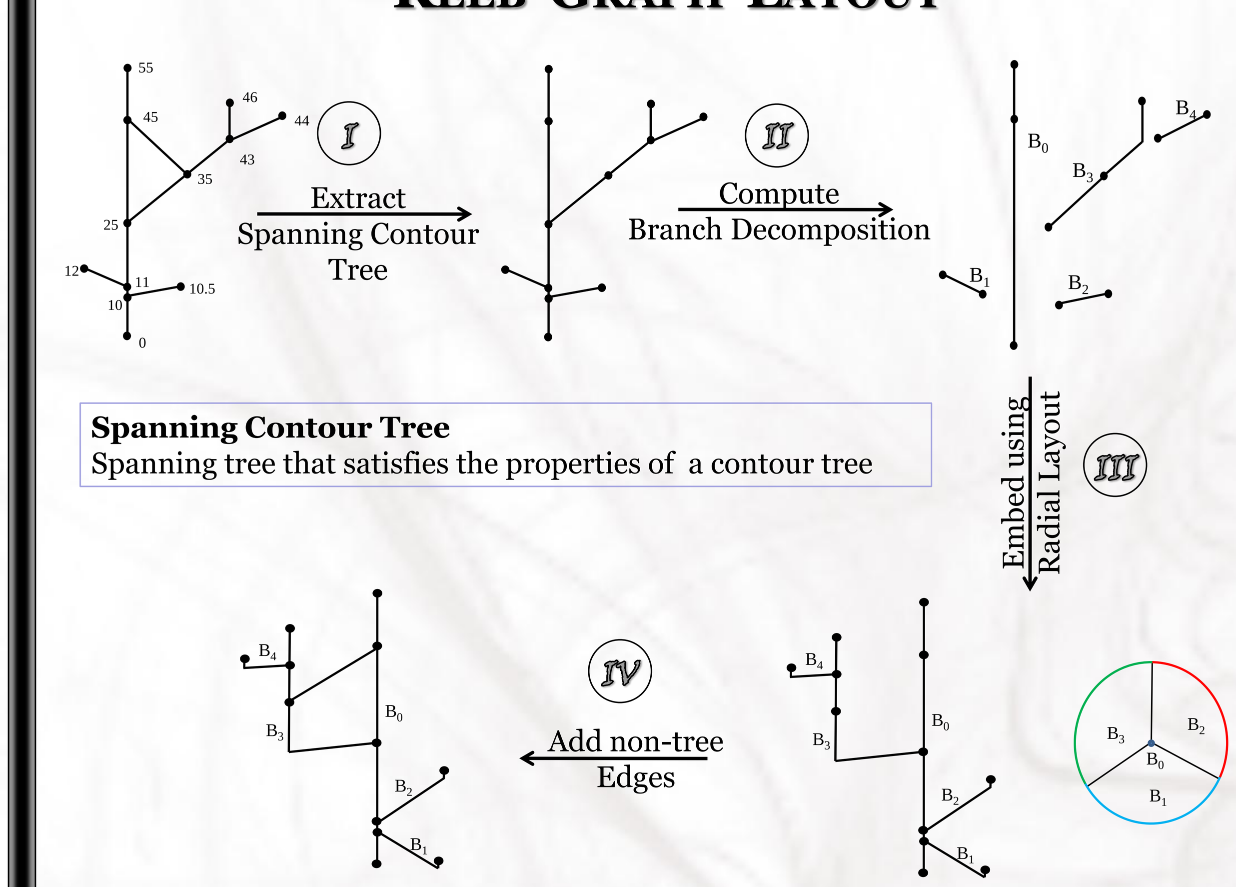


An Application



Mesh Simplification and Feature Extraction

Reeb Graph Layout



Publications

Efficient output-sensitive construction of Reeb graphs
Harish Doraiswamy and Vijay Natarajan
ISAAC '08: Proc. Intl. Symp. Algorithms and Computation, 2008.

Efficient algorithms for computing Reeb graphs
Harish Doraiswamy and Vijay Natarajan

References

David Eppstein.
Dynamic generators of topologically embedded graphs, in: *SODA'03: Proc. Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

Kree Cole-Mclaughlin, Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci.
Loops in Reeb graphs of 2-manifolds, *Discrete and Computational Geometry*, 32 (2), 2004, 231-244.

Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas.
Robust on-line computation and presentation of Reeb graphs: simplicity and speed, *ACM Transactions on Graphics*, 26 (3), 2007, 58.

Mikkel Thorup.
Near-optimal fully-dynamic graph connectivity, in: *STOC '00: Proceedings of the thirty-second Annual ACM Symposium on Theory of Computing*, 2000.