Constructing Reeb Graphs using Cylinder Maps

Harish Doraiswamy
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India
harishd@csa.iisc.ernet.in

Aneesh Sood
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India
aneesh.sood@gmail.com

Vijay Natarajan
Department of Computer Science and Automation
Supercomputer Education and Research Centre
Indian Institute of Science
Bangalore 560012, India
vijayn@csa.iisc.ernet.in

ABSTRACT

The Reeb graph of a scalar function represents the evolution of the topology of its level sets. In this video, we describe a near-optimal output-sensitive algorithm for computing the Reeb graph of scalar functions defined over manifolds. Key to the simplicity and efficiency of the algorithm is an alternate definition of the Reeb graph that considers equivalence classes of level sets instead of individual level sets. The algorithm works in two steps. The first step locates all critical points of the function in the domain. Arcs in the Reeb graph are computed in the second step using a simple search procedure that works on a small subset of the domain that corresponds to a pair of critical points. The algorithm is also able to handle non-manifold domains.

Categories and Subject Descriptors: I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages, and systems

General Terms: Algorithms

1. INTRODUCTION

The Reeb graph of a scalar function is obtained by mapping each connected component of its level sets to a point. Level set components that contain critical points of the function map to nodes of the graph. The abstract representation of level-set topology within the Reeb graph enables development of simple and efficient methods for modeling objects and visualizing scientific data. They serve as an effective user interface for selecting meaningful level sets [1] and for designing transfer functions for volume rendering [8].

In this video, we illustrate an efficient two-step algorithm[†] for computing the Reeb graph of a piecewise-linear (PL) function in $O(n+l+t\log t)$ time, where n is the number of triangles in the input mesh, t is the number of critical points of the function, and l is the size (number of edges) of all critical level sets. For a comparison of our algorithm with existing approaches we refer the reader to the paper where we proposed the algorithm [4].

2. BACKGROUND

Let $\mathbb{M}^d$ denote a d-manifold with or without boundary. A smooth, real-valued function $f : \mathbb{M}^d \to \mathbb{R}$ is called a *Morse function* if it satisfies the following conditions [3]:

1. All critical points of f are non-degenerate and lie in the interior of $\mathbb{M}^d$.
2. All critical points of the restriction of f to the boundary of $\mathbb{M}^d$ are non-degenerate.
3. All critical values are distinct *i.e.*, $f(p) \neq f(q)$ for all critical points $p \neq q$.

The above conditions typically do not hold in practice for PL functions. However, simulated perturbation of the function [5, Section 1.4] ensures that no two critical values are equal. A total order on the vertices helps in consistently identifying the vertex with the higher function value between a pair of vertices.

Critical points and level sets. Critical points of a smooth function are exactly where the gradient becomes zero. Critical points of a PL function are located at vertices of the mesh [2].

The preimage of a real value is called a *level set*. The level set of a regular value is a $(d-1)$-manifold with or without boundary, possibly containing multiple connected components. We are interested in the evolution of level sets against increasing function value. Significant topological changes occur at critical points, whereas topology of the level set is preserved across regular points [6].

The *link* of a vertex consists of all vertices adjacent to it and the induced edges, triangles, and higher-order simplices. Adjacent vertices with lower function value and their induced simplices constitute the *lower link*, whereas the adjacent vertices with higher function value and their induced simplices constitute the *upper link*. In the context of Reeb graphs, we are only interested in critical points that modify the number of level set components. So, it is sufficient to count the number of connected components of the lower / upper link for identifying these critical points. Given a critical point c_i, call the level set $f^{-1}(f(c_i))$ as a *critical level set*.

Reeb graph. The *Reeb graph* of f is obtained by contracting each connected component of a level set to a point [7]. The Reeb graph expresses the evolution of connected components of level sets as a graph whose nodes correspond to critical points of the function. Nodes corresponding to minima and maxima have degree one. A node that corresponds to a saddle has degree three if the saddle merges / splits level set components. Genus modifying saddles do not alter the number of level set components. They are optionally included into the Reeb graph as degree two nodes.

The above description of the Reeb graph focuses on the mapping between individual level set components and nodes or points within arcs of the graph. We propose the use of an alternate but equivalent mapping, where nodes are mapped to components of critical level sets, and each arc is mapped to a *cylinder*, which is a collection of regular level set components that are topologically equivalent to each other. The advantage of our proposed alternate map is that a

[†]The algorithm is described in a paper that appeared in the Proceedings of the International Symposium on Algorithms and Computation [4].

SCG'10, June 13–16, 2010, Snowbird, Utah, USA.
ACM 978-1-4503-0016-2/10/06.

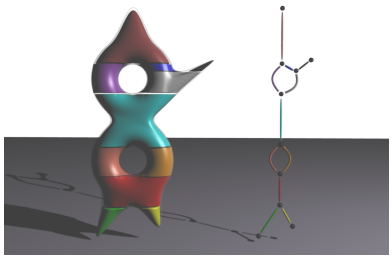

Figure 1: Reeb graph of the height function defined on a solid 2-torus. Reeb graph tracks the topology of level sets.

simple and efficient algorithm to compute the Reeb graph follows immediately from the mapping.

3. THE REEB GRAPH ALGORITHM

We now describe an algorithm [4] that computes the Reeb graph of a PL function f defined on a 3-manifold. We assume that the input manifold is represented by a triangulated mesh, the function is sampled at vertices, and linearly interpolated within each simplex.

The algorithm follows from the alternate mapping described in the previous section. It consists of two steps:

1. Locating critical points in the domain and sorting them based on function value.
2. Identifying pairs of critical points that define cylinders and inserting the corresponding arcs in the Reeb graph.

The link of a vertex in a 3-manifold is a triangulation of a sphere. The vertex is *regular* if it has exactly one lower link component and one upper link component. All other vertices are *critical*. A critical point is a *maximum* if the upper link is empty and a *minimum* if the lower link is empty. Else, it is classified as a saddle. We perform a breadth first search over the link to count the number of components of the upper and lower links.

Level set and cylinder representation. The 3-manifold is represented by a tetrahedral mesh. A level set of the input scalar function is generically a surface that is represented by a collection of triangles. However, the algorithm requires only the 2-skeleton (vertices, edges, and triangles) representation of the domain and the 1-skeleton (vertices and edges) representation of level sets. This is because the 1-skeleton captures the connectivity of level sets, which is exactly what we aim to abstract into the Reeb graph. Edges in a level set of f lie within unique triangles in the domain. So, a level set is represented by a collection of triangles in the input mesh. Cylinders are also represented as a collection of mesh triangles. Specifically, the cylinder bounded by two critical level set components is represented by triangles that contain the intermediate level set components.

Connecting the critical points. We compute arcs in the Reeb graph by tracing paths within each cylinder. Let $\{c_1, c_2, \ldots, c_t\}$ be the ordered set of critical points with function values $\{f_1, f_2, \ldots, f_t\}$ and $f_x < f_y$ whenever $x < y$. Let L_i denote the set of triangles containing the components of the critical level set $f^{-1}(f_i)$ that is modified by c_i. The i^{th} iteration of the algorithm connects c_i with a set of critical points c_p ($f_p > f_i$).

Each component of the upper link corresponds to a potential new arc in the Reeb graph that connects c_i with a higher critical point. We trace the cylinders bounded below by a level set component of c_i in the i^{th} iteration of the algorithm. Starting from a triangle incident on an upper link component of c_i, we march to an adjacent triangle that is incident on a vertex with a higher function value. This traversal is equivalent to tracing a path within a cylinder. We continue the traversal until we reach a triangle that contains an edge of the level set $f^{-1}(f_{i+1})$. We insert an arc into the Reeb graph between nodes corresponding to c_i and c_{i+1} if this triangle is part of the critical level set L_{i+1}. If the triangle does not belong to L_{i+1}, we continue the traversal until it reaches a higher critical level set. If a search initiated from two components of the upper link of c_i reaches the same component of a critical level set, then c_i is declared a genus modifying saddle and the Reeb graph remains unaffected. We obtain the Reeb graph of f once all critical points are processed.

Discussion. Tracking the connected components of the level set requires only a 1-skeleton representation, which can be extracted from the 2-skeleton of the input mesh. So, the algorithm works directly on the 2-skeleton representation of d-manifolds ($d \geq 2$). In the case of non-manifolds, we relax the definition of critical points to include all vertices that modify the topology of the level set. Candidate critical points are again located by counting the number of connected components of the lower and upper link.

Our algorithm in the worst case has an $O(n + l + t \log t)$ running time, where n is the number of triangles in the input, t is the number of critical points of the input PL function and l is the total size of all critical level sets. Though it is possible in theory that $l = O(n^2)$, we notice that l is usually $O(n)$ in practice.

4. ABOUT THE VIDEO

The video animates the various stages of our algorithm for computing the Reeb graph of the height function defined on a solid 2-torus shown in Figure 1. Models were created and rendered using *Blender* (http://www.blender.org) and our implementation of the algorithm. The volume rendered images showing Reeb graphs for various datasets was created using *VTK* (http://www.vtk.org). The video was edited using Blender.

Acknowledgements. Harish Doraiswamy is supported by Infosys Technologies Ltd., Bangalore, under the Infosys Fellowship Award. This work was also supported by the Department of Science and Technology, India, under Grant SR/S3/EECE/048/2007.

5. REFERENCES

[1] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *Proc. IEEE Conf. Visualization*, pages 167–173, 1997.

[2] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *Am. Math. Monthly*, 77:475–485, 1970.

[3] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. *Disc. Comput. Geom.*, 32(2):231–244, 2004.

[4] H. Doraiswamy and V. Natarajan. Efficient output-sensitive construction of Reeb graphs. In *Proc. Intl. Symp. Algorithms and Computation*, pages 557–568, 2008.

[5] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge Univ. Press, England, 2001.

[6] Y. Matsumoto. *An Introduction to Morse Theory*. Amer. Math. Soc., 2002. Translated from Japanese by K. Hudson and M. Saito.

[7] G. Reeb. Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie ses Séances, Paris*, 222:847–849, 1946.

[8] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 13(2):330–341, 2007.