Computation of Layered Reeb Graphs*

B. Strodthoff †, B. Jüttler†

Abstract

Reeb graphs represent the topological structure of a manifold based on a scalar-valued, sufficiently smooth function defined on it. The use of more than one function leads to Reeb spaces, which are thus able to capture more features of an object. The structure of the Reeb space of a 3-manifold with boundary with respect to two scalar-valued functions is captured by the layered Reeb graph. We present an efficient algorithm for computing such layered Reeb graphs, using only a boundary representation of the underlying manifold.

1 Introduction

In this section, we will recall some definitions and properties from previous works.

1.1 Reeb graphs and -spaces

We begin by recalling the definition of Reeb graphs (see also [1] for an introduction).

Definition 1 *Consider a scalar-valued function f defined on a d-manifold (or manifold with boundary) M. Points mapped to the same function value form a* level set*, connected parts of a level set are called* level set components*. The* Reeb graph *of M with respect to f is obtained by contracting every level set component to a point, maintaining adjacency between level sets.*

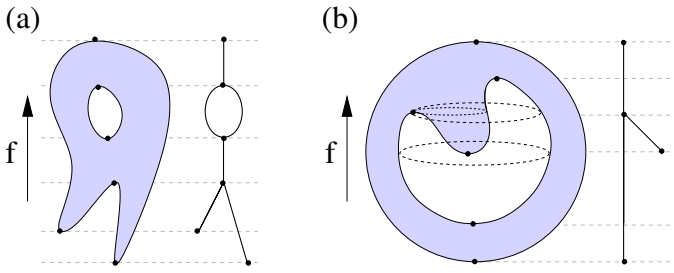

Figure 1: Reeb graphs with respect to the height function, which maps each point to its last Cartesian coordinate, of (a) a 2-manifold in the plane, and (b) a 3-manifold in space, containing an inner void.

*Supported by ESF Programme EuroGIGA-Voronoi
†Johannes Kepler University, Linz, Austria

In the following, we will restrict ourselves to the cases $d = 2$ and $d = 3$ as shown in Figure 1, and assume that M is embeddable in $\mathbb{R}^3$.

Reeb spaces were considered in 2008 by [4], generalizing Reeb graphs by considering two functions f and g on the manifold M.

Definition 2 *Consider two scalar-valued functions f and g on a 3-manifold M with boundary. Here,* level sets *consist of all points with constant f- and g-value. The* Reeb space *of M with respect to f and g is obtained by contracting every level set component to a point, maintaining adjacency between level sets.*

1.2 The layered Reeb graph

In [6], we introduced Layered Reeb graphs as a discrete representation for the structure of a Reeb space.

Definition 3 *For a constant value c attained by f, the Reeb graph of the level set $\{f = c\}$ with respect to g is called a* level set Reeb graph*.*

To obtain the layered Reeb graph*, the arcs of the Reeb graph with respect to f are subdivided into parts of equivalent level set Reeb graphs. Then, the corresponding level set Reeb graphs are added to these parts as a secondary structure, see Figure 2.*

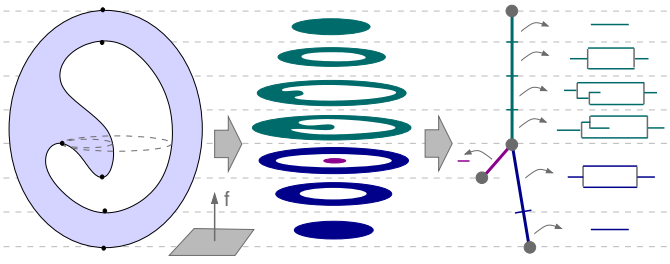

Figure 2: Layered Reeb graph. Left: vertical cut through a 3-manifold. Middle: level sets of f. Right: layered Reeb graph.

Since a point of a level set Reeb graph represents all points with constant f *and* g-value, the Reeb space basically consists of the level set Reeb graphs for all f values, stacked with intact adjacency. Thus, the layered Reeb graph captures the Reeb space's structure by grouping parts with equivalent level set Reeb graphs.

1.3 Boundary-based construction of Layered Reeb graphs

Several algorithms are described in the literature which compute the Reeb graph of a surface for a given surface description or the Reeb graph of a three-dimensional domain for a given volumetric description (like e.g. [8, 2, 5]). These algorithms typically allow for a rather general choice of defining functions. However, if a three-manifold is given in a boundary description, a volumetric description has to be generated to apply these approaches, since the Reeb graph of the manifold and the Reeb graph of its boundary surface are, in general, different objects.

To avoid this, we describe a construction algorithm which uses only the manifold's boundary description. This leads to substantial computational advantages, since the generation of a volume description is costly, and a boundary description is, typically, "smaller" (e.g. comparing the number of elements in a surface- and volume mesh). However, the class of defining functions has to be restricted.

The boundary-based construction algorithm for Reeb graphs of 3-manifolds with respect to the height function was introduced in [7]. In [6], we extended this approach to the computation of layered Reeb graphs, and defined a feasible function class for the defining functions. In short, the functions have to be chosen such that all changes in level set Reeb graphs are induced by the boundary. Additionally, some basic geometric decisions have to be available inside the level sets. The following lemma gives a sufficient condition for the boundary-based construction to work, see [6] for details.

Lemma 1 *Assume* $\nabla f \times \nabla g \neq 0$. *Additionally, assume that another function* h *is available such that* $det(\nabla f, \nabla g, \nabla h) \neq 0$. *Then, the layered Reeb graph with respect to* f *and* g *is determined by function values of* f, g *and* h *on the boundary of* M.

In this setup, f, g and h form a reparametrization of space.

1.4 Jacobi sets

For functions fulfilling the condition in Lemma 1, changes in the level set Reeb graph only occur on the boundary. Jacobi sets, as introduced by [3], will help us identify those points on the boundary where changes occur.

Definition 4 *Let* $\bar{f}$ *and* $\bar{g}$ *denote the restrictions of* f *and* g *to the boundary surface of* M. *The* Jacobi set *of* $\bar{f}$ *and* $\bar{g}$ *consists of all points with* $\nabla \bar{f} \times \nabla \bar{g} = 0$, *with* ∇ *denoting the gradient operator on the boundary surface of* M.

In our setup, the Jacobi set is a network of curves, where all vertices have even degree [3]. When sweeping through the level sets with respect to f, the level set Reeb graphs' vertices move along the arcs of the Jacobi set [6], see Figure 3. Thus, changes in the level set Reeb graphs occur when sweeping past a vertex or local extremum of the Jacobi set, marked by white dots in the figure.

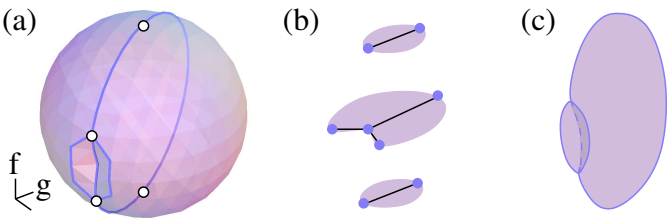

Figure 3: For a simple example: (a) Object and Jacobi set (b) level sets with level set Reeb graphs and (c) Reeb space.

2 Boundary-based construction algorithm

We will now describe an improved version of the algorithm that was sketched in [6].

The layered Reeb graph is computed by sweeping through the level sets with respect to f, see Algorithm 1. First, all points of M are identified where changes occur in the structure of level sets of f or in their level set Reeb graphs. We will call these points *events* in the following. Then, we sweep through the level sets of f, starting at the lowest value. Information about the current level set is stored in the *status*. At each event, the *event handler* decides which level set components in the status are influenced by the event, and implements these changes. We will now describe these elements in more detail.

Algorithm 1 computeLayeredReebGraph

```
find all events
sort events by increasing f-values
for all events e do
    climb status-components to the f-level of e
    identify components of status influenced by e
    adapt status
end for
```

2.1 Status

The status contains information about the current level set by storing a list of all its components. Each level set component maintains a list containing all components of its boundary, where each boundary component stores references to the Jacobi curves it intersects. Additionally, each level set component knows its current level set Reeb graph.

2.2 Events

Considering the information stored in the status, there are basically four types of events, where changes in the status occur:

① Events where the number or connectivity of level set components changes.

② Events where boundary components of a level set component changes.

③ Events where Jacobi references of a boundary component change.

④ Events where a level set Reeb graph changes.

One can observe that ① ⊂ ② ⊂ ③ ⊂ ④. In the following, we will categorize an event by the smallest set it belongs to, see Figure 4.

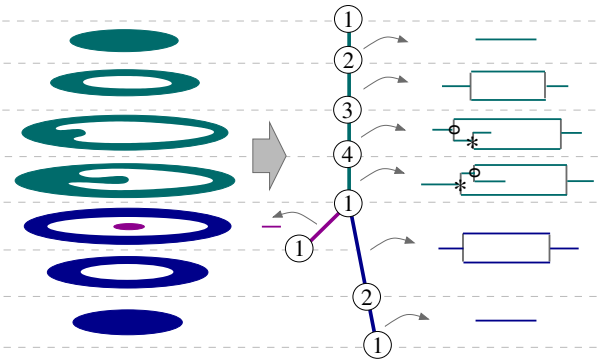

Figure 4: Layered Reeb graph of Figure 2, including categorization of events.

Events of the first three types occur in vertices or local extrema of the Jacobi set. They can be identified efficiently once the Jacobi set has been determined.

Type ④ occurs if two vertices of the level set Reeb graph (or, equivalently, the Jacobi arcs they move on) swap their order with respect to g. In Figure 4, the two vertices marked by ∘ and ∗ in the level set Reeb graph are swapped, and the graph connectivity changes. To find these events, we do a preliminary sweep through the Jacobi set, maintaining a list of intersected Jacobi arcs, sorted by the g-value of their intersection with the f-level set.

2.3 Event handler

In every event, the corresponding event handler first *associates* the event to the status, i.e. it identifies all elements of the status which are influenced by the event. If the event is not a local minimum of the Jacobi set, its lower Jacobi arcs can be looked up in the status, directly. For local minima of the Jacobi set, which are not local minima of the surface, we can trace a curve along the f level set to the next intersection with a Jacobi arc. In order to determine whether a level set component contains a local minimum of the surface, its boundary curves are traced starting from their Jacobi references.

Afterwards, the event handler *adapts* the status. The necessary changes to Jacobi references, level set- and boundary components are rather straightforward. Thus, we would like to concentrate on the implementation of the changes to the level set Reeb graphs in the next section.

2.4 Handling level set Reeb graphs

In the brute force approach as outlined in [6], the level set Reeb graph of a level set component is recomputed between each two successive events on the corresponding arc of the Reeb graph. In order to do this, the level set's boundary curves are traced at an intermediate f-level. Then, a sweep with respect to g provides the level set Reeb graph, see e.g. Figure 5 for some results.

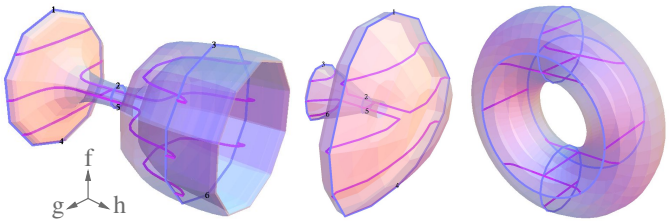

Figure 5: Objects with embedded level set graphs

This approach is obviously very costly. The computation time for the level set Reeb graphs easily dominates the total computation time. For many events it is, however, rather straightforward how to adapt the level set Reeb graph to reflect the level set component's structure above the event. Implementing the adaptation rules for non-degenerate cases of event types ③, ④ and some cases of type ②, we can reduce the number of recomputed graphs by about 90%.

As first example, consider events of type ④, where two Jacobi arcs swap their relative position with respect to g. There are no changes if the swapped Jacobi arcs belong to different level set components. Otherwise, the corresponding vertices of the level set Reeb graph are swapped in its ordered vertex list. Additionally, some incident arcs are exchanged between the two vertices in case there is an arc connecting them. This can only occur if both vertices have valency three, see Figure 6 for typical examples.

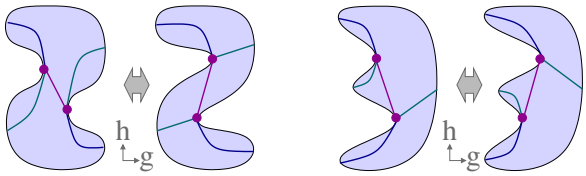

Figure 6: Changes in an event of type ④.

As second example, consider a local minimum of the Jacobi set of type ③. Here, an existing arc is split to attach a new arc. To determine the split arc, we trace the boundary curve to the next intersected Jacobi arc. Starting from the corresponding vertex, we follow the level set Reeb graph to the g level of the event point, see Figure 7.

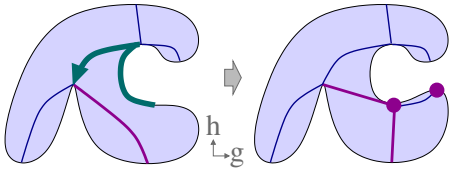

Figure 7: Changes in local minimum of the Jacobi set of type ③: The purple arc is split, and a new arc is attached. The green arrow marks the searching route.

2.5 Further examples

We implemented the sketched algorithm for a piecewise linear setup: The manifold is given as triangular surface mesh and the piecewise linear approximations of the functions are considered.

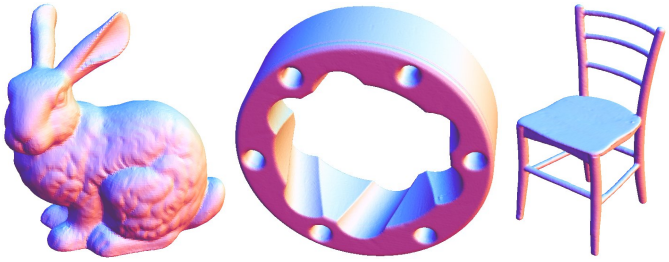

Figure 8: Considered test objects in Table 1: Stanford bunny and Rolling stage- and Wooden chair model from aimatshape shape repository.

For each of the three meshes in Figure 8, the layered Reeb graph of the object and of its complement are computed using six different combinations of the coordinate functions. Of these in total 12 setups per object, we represent the smallest, the most complicated, and an intermediate result in Table 1.

	triangles	events	adapts	time 1	time 2
Stanford bunny	69 664	2 669	2 402	4.0s	0.4s
		4 030	2 941	7.3s	1.7s
		13 265	12 634	53.0s	2.8s
rolling stage	382 242	3 995	3 766	18.6s	2.3s
		6 569	5 583	16.8s	2.5s
		63 809	60 756	376.3s	27.8s
wooden chair	408 398	4 697	4 110	15.8s	2.2s
		14 222	12 983	90.9s	9.6s
		56 593	53 177	298.3s	21.3s

Table 1: Some results for three function setups per object. Number of events, the number of events which can be handled by an adaption rule, time for the brute force algorithm and time for the adaptive algorithm.

3 Conclusion

After recalling the definition of the layered Reeb graph, we outlined an efficient algorithm for its computation in a piecewise linear setup. In contrast to the brute force algorithm presented earlier, this algorithm adapts level set Reeb graphs at events instead of recomputing them from scratch.

References

[1] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theor. Computer Science*, 392(1-3):5–22, 2008.

[2] H. Doraiswamy and V. Natarajan. Efficient algorithms for computing Reeb graphs. *Computational Geometry*, 42(6-7):606–616, 2009.

[3] H. Edelsbrunner and J. Harer. Jacobi sets of multiple Morse functions. *Foundations of Computational Mathematics, Minneapolis 2002*, pages 35–57, 2004.

[4] H. Edelsbrunner, J. Harer, and A. K. Patel. Reeb spaces of piecewise linear mappings. In *Proc. Sympos. on Comput. Geom.*, pages 242–250. ACM, 2008.

[5] G. Patané, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the Reeb graph. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):583–595, 2009.

[6] B. Strodthoff and B. Jüttler. Layered reeb graphs of a spatial domain. In *Booklet of Abstracts of EuroCG*, pages 21–24, 2013.

[7] B. Strodthoff, M. Schifko, and B. Jüttler. Horizontal decomposition of triangulated solids for the simulation of dip-coating processes. *Computer Aided Design*, 43:1891–1901, 2011.

[8] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Trans. on Visualization and Computer Graphics*, 15(6):1177–1184, 2009.