Computers & Graphics 46 (2015) 186–197

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SMI 2014

Layered Reeb graphs for three-dimensional manifolds in boundary representation

B. Strodthoff*, B. Jüttler

Johannes Kepler University, Linz, Austria

ARTICLE INFO

Article history:
Received 24 June 2014
Received in revised form
25 September 2014
Accepted 27 September 2014
Available online 7 October 2014

Keywords:
Reeb graph
Reeb space
Boundary representation
3D solid

ABSTRACT

Reeb graphs are topological graphs originating in Morse theory, which represent the topological structure of a manifold by contracting the level set components of a scalar-valued function defined on it. The generalization to several functions leads to Reeb spaces, which are thus able to capture more features of an object. We introduce the layered Reeb graph as a discrete representation for Reeb spaces of 3D solids (embedded three-dimensional manifolds with boundary) with respect to two scalar-valued functions. After that we present an efficient algorithm for computing the layered Reeb graph, which uses only a boundary representation of the underlying three-dimensional manifold. This leads to substantial computational advantages if the manifold is given in a boundary representation, since no volumetric representation has to be constructed. However, this algorithm is applicable only if the defining functions satisfy certain conditions.

1. Introduction

A *Reeb graph* [1] is a topological data structure which is able to capture the topology of shapes. Considering a scalar-valued function defined on the domain of interest, the level sets of this function may consist of several components, which evolve while sweeping through the function values. The Reeb graph encodes the evolution of these level set components, which are sometimes referred to as *contours* in the literature, by collapsing each contour to a point. Typical applications of Reeb graphs include shape abstraction [2–4], shape recognition [5–9] and shape decomposition [10,11], but they are also applied e.g. to loop detection [12], landscape modeling [13], to guide hex-meshing [14] or in the analysis of trajectories [15] and point data [16].

Generalizing this approach to vector-valued defining functions (or, equivalently, to several scalar-valued functions) leads to *Reeb spaces* [17], which are thus able to capture more features of a shape. However, they are a more complicated structure. Consider for example the Reeb space of a three-dimensional manifold with respect to two functions. In general it consists of several connected surface patches, which makes its computation and storage more difficult. As our first goal, we propose the layered Reeb graph for this situation, which encodes the information captured by the Reeb space using two layers of Reeb graphs, see Fig. 1.

We demonstrate that the Reeb graph of a general solid object does not capture the topology of the object. In contrast, the Reeb space with respect to two functions, and therefore also the layered Reeb graph proposed in this paper, is able to do so. Besides, the layered Reeb graph can easily be embedded into the object. From this embedding, a skeletal structure of the object can be derived. To our knowledge, no algorithm for the computation of Reeb spaces has been proposed in the literature, and our algorithm for the layered Reeb graph is a first step in this direction. We expect that the Reeb space will find various applications in shape recognition, classification and decomposition.

The existing literature on Reeb graphs describes several algorithms, which compute the Reeb graph of a manifold with boundary with respect to a given scalar-valued function, typically using a simplex mesh of the manifold as input, or using a voxel-based description, as in [18]. However, if a three-dimensional manifold is given in a boundary representation, a volumetric representation 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.

As our second goal, we describe a construction algorithm for the layered Reeb graph (and, implicitly, for the Reeb graph), which uses only a boundary representation of the manifold. This leads to substantial computational advantages, since the generation of a volume representation is costly, and a boundary representation typically has a smaller data volume, e.g. comparing the number of elements in a surface- and volume mesh. However, the class of defining functions has to be restricted.

* Corresponding author.
E-mail address: birgit.strodthoff@jku.at (B. Strodthoff).

http://dx.doi.org/10.1016/j.cag.2014.09.026
0097-8493/

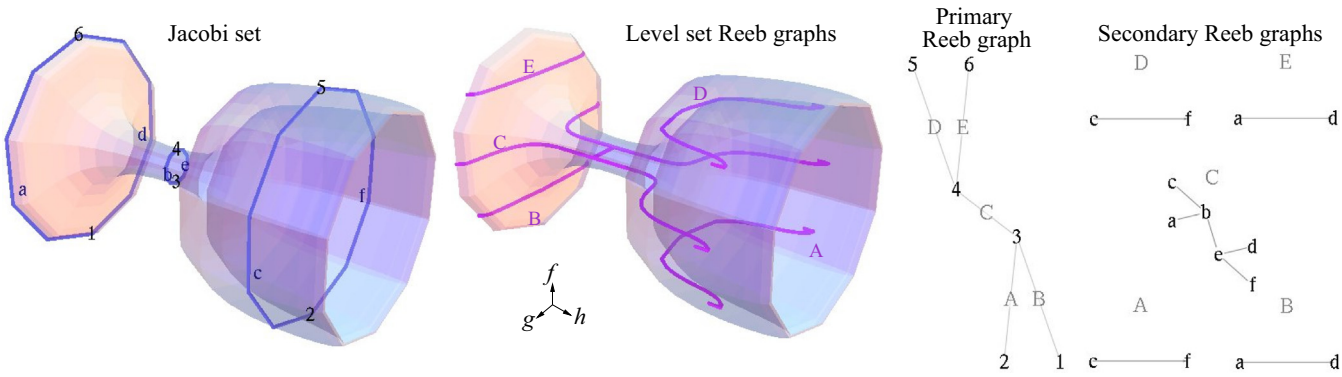

Fig. 1. Example of a layered Reeb graph. Left: Object with the blue curves indicating the Jacobi set, which guides our construction, see Section 5. Center-left: For each level set component of the height function, a Reeb graph with respect to a second function is shown, which is embedded into the level set, see Definition 3. First, the level set components *A* and *B* occur, which are joined to form *C*, and so on. Center-right: The Reeb graph with respect to the height function, where arcs are labeled by the capital letters of the corresponding level set component. Right: The Reeb graphs of each level set component with respect to a second function. The primary and secondary graphs together form the layered Reeb graph, see also Definition 3. For a detailed explanation of the labels, see Section 7.1. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

The remainder of this paper is structured as follows. In the next section, we will review related work on Reeb graphs before defining the layered Reeb graph in Section 3. In Section 4 we identify the conditions on the defining functions which we will need to construct the layered Reeb graph from a boundary representation. After that, we study the Jacobi set and its relevance for the layered Reeb graph in Section 5. Finally, our construction algorithm is described in Section 6 and we present some results in Section 7.

2. Related work

Several algorithms for the computation of Reeb graphs are described in the literature, see e.g. Biasotti et al. [19,20] or Edelsbrunner and Harer [21] for overviews.

The first known algorithm by Shinagawa and Kunii [22] constructs the Reeb graph in any dimension by explicitly *maintaining the level sets* of the function while sweeping through the function values. The level sets are updated at every vertex, which results in an $O(n^2)$ runtime. This approach was improved by using more efficient data structures for storing the level sets, by Cole-McLaughlin et al. [23] and Brandolini and Piastra [24] for two-dimensional manifolds and by Doraiswamy and Natarajan [25] and Parsa [26] for higher dimensions.

Sampling-based algorithms analyze the evolution of level sets by dissecting the domain of interest at certain (e.g. uniformly distributed) function levels and analyzing their connectivities, e.g. by Hilaga et al. [8], Attene et al. [2], Berretti et al. [10] and Biasotti et al. [6]. They are, in principle, able to produce a multilevel representation of a shape by decreasing the distance between analyzed function levels in every step. However, they are not guaranteed to compute the correct Reeb graph, since they may loose important features if their resolution is chosen too coarse unless an additional check is included.

Other approaches exploit the vital relation between the Reeb graph and the *critical points* of the defining function. Patanè et al. [27] successively slice a given two-dimensional manifold at critical function levels and extract the adjacencies between the saddle points by flooding through surface triangles. Berretti et al. [11] successively merge level sets that are represented by the same arc in the Reeb graph. Other authors first identify all critical points and then find connections between them using monotone paths. For example, Doraiswamy and Natarajan [28] use paths of adjacent triangles of a simplex mesh of arbitrary dimension, and Strodthoff et al. [29] use monotone edge paths to construct the Reeb graph of a three-dimensional manifold in boundary representation.

The loop-free variant of Reeb graphs, called *contour trees*, can be constructed more efficiently than the Reeb graph, and there are many applicable algorithms described in the literature. For example, Carr et al. [30] first sweep through the function values twice to construct the split- and join-tree, which are then put together to form the contour tree. Chiang et al. [31] improved this approach by avoiding to sort all input vertices. Instead, they first identify component-critical points, which they connect to split- and join-trees using monotone paths. This results in a construction time of $O(n+t \log t)$ with t denoting the number of component-critical points.

Due to the efficient possibilities for constructing contour trees, there have been efforts to reduce the construction of Reeb graphs to the computation of contour trees. Tierny et al. [32] use the Reeb graph of an object to find all loops in the object, and introduce cuts to open the loops of the Reeb graph. Thus they produce one manifold with a loop-free Reeb graph, which can be handled by a contour tree algorithm. Similarly Doraiswamy and Natarajan [33] identify saddle points where they cut the object into several parts with loop-free Reeb graphs to which the algorithm of [30] is applied individually.

Some recent computation algorithms for the Reeb graph do not fit into one of these frameworks. Pascucci et al. [34] introduce an *on-line algorithm*, which constructs the Reeb graph of a simplex mesh while streaming the input data. This algorithm performs well in practice and is applicable to large meshes since it does not require the whole input to be stored in memory. Harvey et al. [35] propose a *randomized algorithm* for simplex meshes which computes the Reeb graph in $O(n \log n)$ expected time by successively collapsing triangles in random order. Furthermore, Dey and Wang [36] study the approximation of the Reeb graph for a manifold given by a point sample, and a simplified Reeb graph was introduced in [24].

Reeb graphs are defined via a scalar function on a given domain of interest. The extension to several functions leads to *Reeb spaces*. They were introduced by Edelsbrunner et al. [17], but appear to be little researched by now. Reeb spaces are able to capture more features of an object than Reeb graphs, which makes them an interesting object of study. However, as mentioned before, they possess a more complicated structure than Reeb graphs, and are, therefore, harder to handle. We introduce a different representation for a special class of Reeb spaces using two layers of Reeb graphs, which we call *layered Reeb graphs*, and an algorithm to compute this object.