Automatic Decomposition of 3D Solids into Contractible Pieces Using Reeb Graphs

Birgit Strodthoff, Bert Jüttler

Institute of Applied Geometry, Johannes Kepler University, Linz, Austria

Abstract

This work is motivated by the need to generate volumetric spline models for isogeometric analysis. There exist numerous constructions of volumetric spline models that represent contractible solids. We present a novel decomposition algorithm that splits general solids into pieces that can be dealt with by these existing methods. More precisely, we present a method to automatically decompose solid objects in boundary representation into pieces with fewer or no tunnels by cutting them with auxiliary surfaces. The segmentation is guided by a reduced form of the object's boundary and volume Reeb graphs with respect to several Morse functions, the level sets of which define the cutting surfaces. Special attention is paid to the selection of suitable cutting surfaces, where we employ a quality criterion to avoid the creation of badly shaped pieces.

Keywords: Object decomposition, segmentation, genus reduction, Reeb graphs, isogeometric analysis

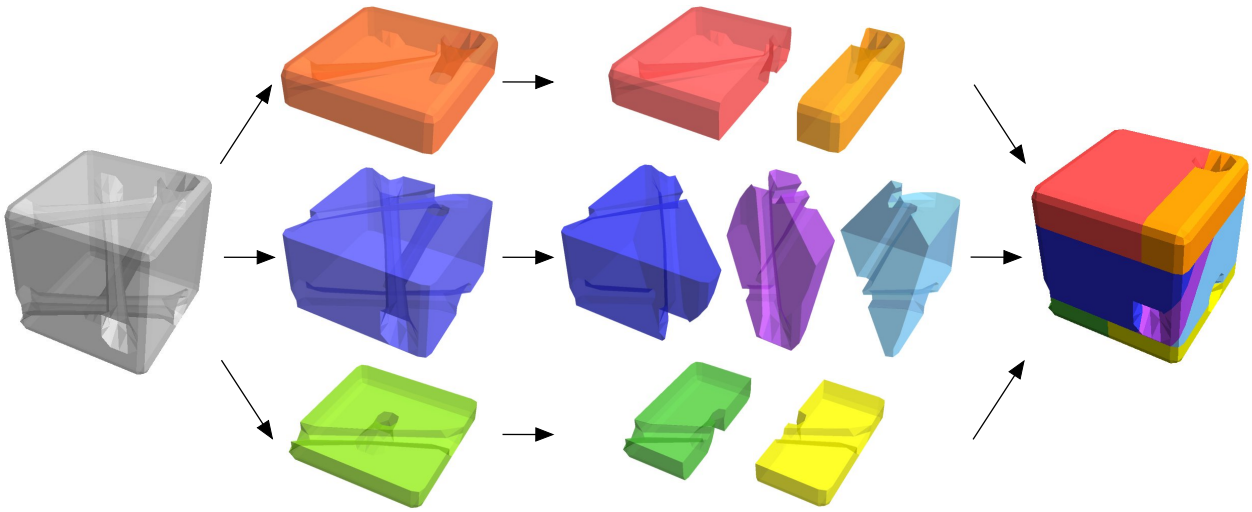


Figure 1: Segmentation of a cube with several straight tunnels using cutting surfaces that have been automatically selected by our algorithm, based on boundary and volume Reeb graphs. For a more detailed description, the reader is referred to Section 3.1.

1. Introduction

We describe the goals of our splitting algorithm in this section and discuss related work.

1.1. Outline and motivation

Given a solid object, which is represented by a boundary surface, we present a new algorithm for automatically reducing the number of tunnels through the object (which is closely related to the genus of the boundary surface), see Fig. 1. As an input, we use a triangular surface mesh of a 3D-object, from which only local information about neighboring triangles is readily accessible. For analyzing, comparing or decomposing solid objects, some global structural information is often needed. Many approaches have been used to construct representations that provide such information, for example by identifying tunnels in the shape, by pooling similar triangles into region patches to get a more compact representation, or by creating a skeletal structure to represent the shape. We will follow the latter approach, computing Reeb graphs of the object with respect to simple linear Morse functions in order to capture information about the shape of the object and its boundaries at the same time.

The Reeb graphs are then used to guide the decomposition of the object. In each iteration of the algorithm, planar cutting surfaces in the level sets of the current Morse function are inserted into the object if they simplify the shape. Only a small number of cutting surfaces are inserted until the object is split into separate, contractible pieces (i.e., each piece can be continuously shrunk to a point), see Fig. 1 for a representative result. A quality

criterion is applied in order to select good cuts, which, for example, form near-orthogonal angles with the object's surface, see Fig. 2.

In principle it is possible to resolve any tunnels in the object in this way, provided that the Morse functions are chosen appropriately. For straight tunnels in the object, using linear functions is sufficient. For cutting open more complicated tunnels in the object, however, more elaborate Morse functions would be required.

The decomposition of a solid into large contractible pieces may serve several purposes, but our work was motivated almost exclusively by requirements from isogeometric analysis. In order to perform simulations on a given object using isogeometric analysis, a parameterization by NURBS volumes needs to be generated [27]. There exist numerous constructions of volumetric spline models that represent contractible solids, e.g. [14, 26, 40, 44]. We present a novel decomposition algorithm that splits general solids into pieces that can be dealt with by these existing methods.

The recent survey article [33] describes the entire process, covering boundary surface (re-) construction, solid segmentation into topological hexahedra, parameterization and isogeometric simulation. In the second step of this process chain, the object has to be split into hexahedral pieces. For the case of contractible solids, such a splitting is investigated in [17, 28]. In order to extend the splitting algorithm to more general solids, an initial segmentation into contractible solids is required. The algorithm, which is described in this paper, provides the preprocessing step, which was still missing. This has substantially enlarged the class of feasible input data for the isogeometric segmentation pipeline [33]. For instance, the manual preprocessing step needed when dealing with the TERRIFIC demonstrator (Example A, see Fig. 14 and Table 1) has become obsolete.

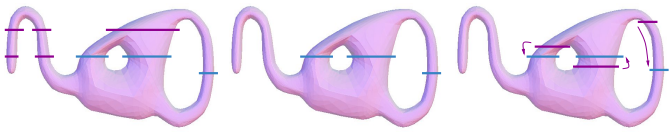


Figure 2: Goals of the decomposition. We want to use few cuts, avoiding over-segmentation as in the left picture. However, as a result, the object should be segmented into separate pieces, so if we insert any of the cutting surfaces shown in the central picture, we need to insert the others as well. Finally, we try to use high-quality cutting surfaces, see also Section 3.4.

1.2. Related work

There is a vast literature devoted to the decomposition of solid objects or their surfaces into (nearly) convex pieces, or into meaningful parts reflecting the human perception of the object, see [35] for an overview of decomposition algorithms. These decompositions are typically computed by repeatedly splitting in concave surface areas (such as [20, 23]) or by growing surface regions with similar characteristics [30]. The choice of cuts is often guided by some distance measures on the surface [19, 39], or using skeletal structures of the object [22, 24, 34, 39, 43].

Pants decompositions segment a surface into patches of genus zero with three boundary components [6, 21, 15]. While this is a topological approach as the one presented in this paper, it decomposes the surface of the object, not the object itself. It is often possible to extend surface decompositions to volume decompositions by inserting cutting surfaces, although this is not the case in general. Even for a double torus, which possesses two surface decompositions into pairs of pants, only one of them can be extended to a volume decomposition, see also Section 2.3.

Morse theory allows to extract topological information about a manifold by analyzing a scalar-valued function (called the Morse function), which is defined on it. For example, Morse (-Smale) complexes decompose an underlying 2-manifold into quadrangular pieces of similar flow patterns, see e.g. [8, 13, 11, 25, 29]. Even though these complexes have been employed for generating hex-dominant meshes of 3-manifolds with boundary [25], the cells are generic crystals in this situation, thereby limiting the applicability to segmentation into topological hexahedra. Moreover, rather complicated defining functions (like the eigenfunctions of certain differential operators) are used in order to produce a meaningful segmentation, resulting in rather high computational costs of these methods.

Our aim is to reduce the number of tunnels through a solid object (and therefore the genus of its boundary surface) by cutting it open along cutting surfaces. To get a contractible solid it suffices to perform one such cut for opening each of the tunnels. However, we use additional cuts in order to obtain pieces that can be separated from each other. More precisely, we guarantee that the two copies of each cutting surface belong to different pieces. We do not aim at generating the minimum number of such pieces. Instead we select "good" cuts according to a certain quality measure.

While the volume decomposition algorithms mentioned above often implicitly create pieces with fewer or no tunnels (in fact, convex pieces are necessarily tunnel-free), we are not aware of many algorithms specifically addressing a decomposition into contractible pieces. The close relation between the genus of a surface and its Reeb graph is investigated in [3]. Extending this idea, Reeb graphs were used to detect handle and tunnel loops of an object in boundary representation [7]. The present paper is inspired by two works [10, 38] which insert cutting surfaces in order to get objects with cycle-free Reeb graphs. This reduces the genus, but some tunnels may remain in the object. Also, the quality of the inserted cutting surfaces has not been considered so far. See Section 2.2 for a more detailed comparison of our work with these two approaches.

Our decomposition is guided by Reeb graphs (see e.g. [2, 12, 16, 18, 41]) with respect to one or several Morse functions. The Reeb graph of object surfaces has been used repeatedly for analyzing and decomposing the boundary surface [1, 39, 42], typically using the average geodesic distance function [1, 42] or geodesic distances to feature

points [39].

Several algorithms are described in the literature that compute the Reeb graph of a surface for a given surface description, or the Reeb graph of a volume for a given volumetric description (e.g. [5, 9, 31, 32, 38]). These algorithms allow for a rather general choice of defining functions. However, if a three-dimensional manifold is given in a boundary description only, a volumetric description (such as a tetrahedral mesh) has to be generated first to compute the Reeb graph of the object using these approaches. The Reeb graph of the object and the Reeb graph of its boundary surface are generally different.

To avoid the generation of a volumetric description we compute both Reeb graphs using an algorithm that works with the boundary representation, see [36, 37] and Section 2.4. This provides substantial computational advantages, since the generation of a volume description is costly, and a boundary description is, typically, "smaller" with respect to data volume. It should be noted, however, that the boundary-based construction limits the choice of the Morse function.

1.3. Overview

We will start by considering the different Reeb graphs of an object with respect to a given function, their relation to the genus of the object and how they can be applied to guide decomposition algorithms. We will also describe shortly how the Reeb graph of the object volume can be constructed efficiently from a boundary representation under certain restrictions on the defining Morse function.

After that, we will present our volume decomposition algorithm in Section 3. For different linear defining functions, level set surfaces up to a certain quality threshold are repeatedly inserted in order to open tunnels. In contrast to other segmentation algorithms which are guided by Reeb graphs of the object's boundary surface, we use the Reeb graph of the volume itself. Otherwise, we would actually get a decomposition of the object's surface, and it may be impossible to extend it to a volume decomposition, see Section 2.3. Finally we will give some results of the decomposition algorithm.

2. Reeb graphs

We describe Reeb graphs and discuss their role in our decomposition algorithm and their construction from a boundary mesh of an object.

2.1. Definitions

We begin by recalling the definition of Reeb graphs (see also [2] for a more detailed and general introduction).

Definition 1. Consider a scalar-valued function f defined on a d-manifold (or a 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, see Fig. 3.

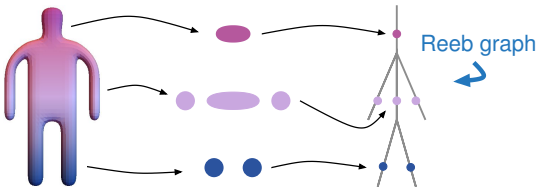


Figure 3: Simple example for the Reeb graph. Left: object with color-encoded height function. Center: level sets at three different function values. Right: the resulting Reeb graph.

For a solid object, two different Reeb graphs can be considered, see also Fig. 4. On the one hand, one can consider a Morse function f defined on the object considered as 3-manifold with boundary. We will call the Reeb graph generated by this function the *volume Reeb graph (VRG)*. On the other hand, consider the restriction of f to the boundary surface of the object. The level set components of that function consist of curves on the boundary of the object. The Reeb graph tracing these level set components will be denoted as *boundary Reeb graph (BRG)* in the following. For simplicity we will use the same symbol f both for the function and for its restriction to the object's boundary surface.

We consider only functions f that do not possess critical points in the interior of the solid, i.e., each of the level sets intersects the solid's boundary. For instance, this is satisfied for all linear Morse functions. The VRG and the BRG are closely related in this case.

Firstly, every vertex of the VRG is then also a vertex of the BRG, since whenever two level set components meet, their outer boundary components become connected also.

Secondly, the edges of the VRG can be labeled by the edges of the BRG that represent the boundary curves of the level set component. The boundary components may change within an edge of the VRG. Thus, edges may consist of several segments with different labels. Similarly, one could define a labeled boundary Reeb graph where edge segments are labeled by the corresponding volume-edges.

Definition 2. The *boundary component-labeled volume Reeb graph (VRG$^+$)* consists of the edges of the VRG which are subdivided into segments of constant boundary components and labeled by the edges of the BRG which represent these boundaries.

This definition is illustrated by Fig. 4.

While the VRG does not recognize tunnels appearing in level sets, and the BRG does not contain the information which boundary components bound the same level set component, the labeled VRG combines the structural information of both graphs, see Fig. 5. Thus, it may be an interesting object for shape analysis in itself.

We will use the labels stored in the VRG$^+$ to access the boundary components of the level set component that is represented by an edge of the VRG. In this way, we can consistently compute cutting surfaces using only a boundary representation of the object, even if these surfaces possess holes. In the following figures, we will, however, show

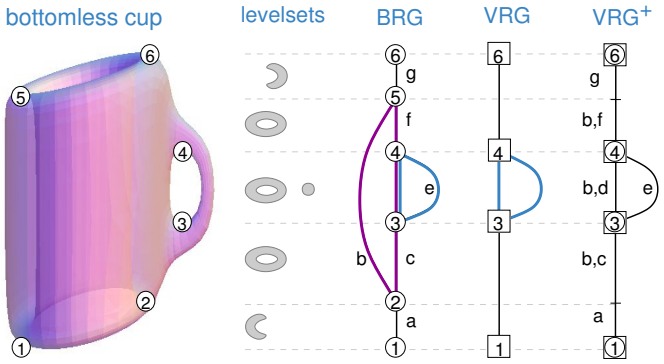


Figure 4: Boundary Reeb graph (BRG), volume Reeb graph (VRG) and their combination (VRG$^+$) with respect to the height function which maps every point to its last coordinate. The VRG contains only one cycle corresponding to the handle of the bottomless cup. In the BRG, also the vertical tunnel induces a cycle, since the boundary curve splits into an inner and an outer component. The VRG$^+$ combines the information of the other two graphs, see Definition 2.

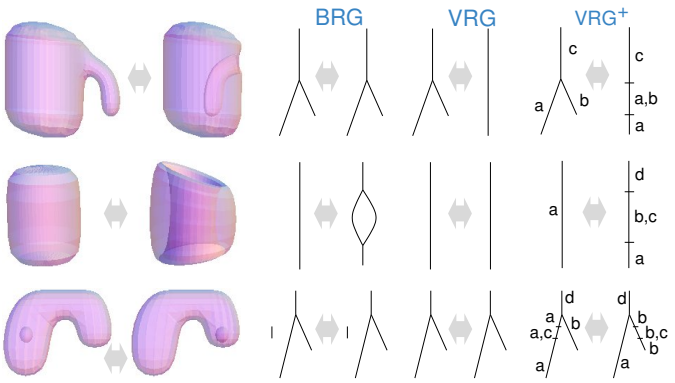


Figure 5: Comparison of the boundary Reeb graph (BRG), the volume Reeb graph (VRG) and the boundary component-labeled volume Reeb graph (VRG$^+$). The objects in the first line have the same BRG, the objects of the second line induce the same VRG, and the objects in the last line cannot be distinguished using either BRG or VRG. The VRG$^+$ correctly distinguishes the objects in each case, and thus contains more information than the separate objects.

the VRG without these labels in order not to overload the pictures.

2.2. Reeb graphs and the genus

Similar to [38], we use the fact that every tunnel in the object induces a cycle in the BRG. For reducing the genus of the object, we seek to reduce the number of cycles in the BRG. For any given Morse function f, there are two types of tunnels, see also [38]. We denote them as resolvable tunnels and non-resolvable tunnels:

- *Resolvable tunnels* induce a cycle in the BRG and in the VRG, such as the handle of the bottomless cup in Fig. 4.
- *Non-resolvable tunnels* only induce a cycle in the BRG, such as the vertical tunnel in the bottomless cup.

Similar to [10, 38], we seek to remove the cycles in the VRG by cutting the object with level sets of the Morse function. The aim of those works is to reduce the construction of the (volume) Reeb graph to the much more efficient construction of the contour tree, its loop-free equivalent. In [38], the top vertices of loops in the VRG are identified, and cuts are inserted in a lower incident edge of these vertices. This approach could easily be adapted to achieve a splitting of the object into separate pieces with loop-free Reeb graphs by cutting all lower incident edges of the loop-top vertices instead of just one per top vertex, similar to [10]. It should be noted that no attempts were made to optimize the quality of the inserted cuts, since they are only inserted symbolically. Thus, the algorithm would sometimes generate badly shaped cutting surfaces, see Fig. 6.

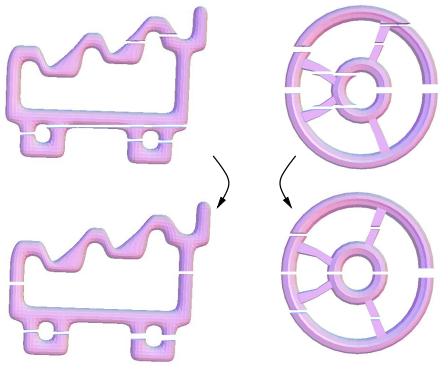


Figure 6: Difference between the results following the approach by Tierny et al. [38] (top row) and our results (bottom row).

Our aim is quite different, since we want to find *good* cuts, for which the cutting surfaces create well-defined intersections with the surface of the object, forming near-orthogonal angles, where possible. Besides, we do not only want to generate a cycle-free VRG. In order to cut all tunnels of the object, we also want to achieve a cycle-free BRG. However, non-resolvable tunnels cannot be opened using f-level sets. Whichever horizontal cut we add in the bottomless cup in Figure 4, the vertical tunnel will still be present in both resulting pieces. Therefore, we will reduce the number of cycles in the VRG and repeatedly apply the same technique to Reeb graphs with respect to several Morse functions f.

2.3. Reeb graphs for object decompositions

For simple objects, surface decompositions based on the BRG can often be extended to satisfactory decompositions of the volume. In general, however, a surface segmentation may contain patches describing the boundary surfaces of tunnels or deep cavities, and thus does not induce a valid volume decomposition, see Figures 7 and 8. In the presented examples, the height function is used to define the Reeb graph. While choosing a different function, e.g. the average geodesic distance function, may lead to better decompositions, one cannot rule out the situation that occurs in the second example in Fig. 7.

In contrast, decomposition algorithms based on the VRG always give valid volume segmentations, e.g. [43]. However, the computation of the VRG is typically more expensive than the BRG-construction, requiring a volume

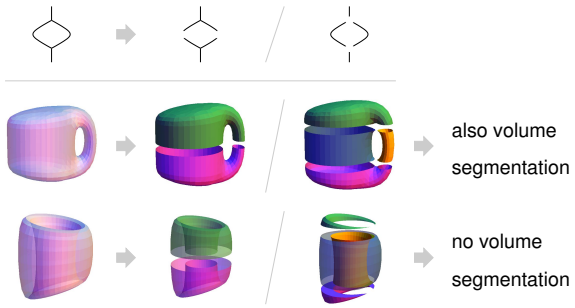


Figure 7: As demonstrated by an example with respect to the height function, segmentations of an object's surface do not necessarily extend to satisfactory volume segmentations. There are two natural ways to decompose the BRG shown in the first line (which is the same for both objects). Each of them leads to meaningful surface segmentations of both shown objects, which can be used e.g. for parameterizing the surface. However, only for the first object can these decompositions be extended to a satisfactory volume segmentation. For the second object, the tunnel in the object is not resolved in the induced volume segmentations. Additionally, two different surface patches have to be combined to one volume piece in the second decomposition.

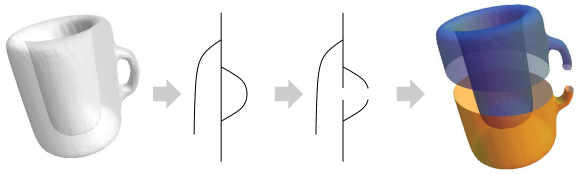


Figure 8: In this example, the information captured by the BRG does not suffice to guide a volume decomposition of the cup-object. It contains no information about the position of the inner boundary surface relative to the outer surface. Therefore, when inserting a cut, we do not know which other parts of the boundary have to be considered.

mesh of the object. Besides, it is not possible to decide whether an object is tunnel-free using its VRG alone, since not every tunnel in the object induces a loop in the VRG.

We apply the VRG$^+$ to obtain a decomposition of the object and we generate the VRG$^+$ from a boundary representation. The VRG guides the volume decomposition, while the edge labels give efficient access to the boundary components of an inserted cutting surface. While cutting along level sets of the defining functions as in [10, 38], we aim at cuts complying with a certain quality criterion to achieve nicely shaped pieces. Finally, we repeatedly use this approach for different defining functions, where the BRG information contained in the VRG$^+$ helps to identify new defining functions for following iterations by providing information about remaining tunnels in the object.

2.4. Computing the VRG$^+$ from a boundary representation

The VRG$^+$ is computed efficiently from a triangulated boundary mesh of the given object, using an adaptation of the algorithm presented in [37], see Algorithm 1. We use a sweep algorithm where the status structure contains all segments of VRG$^+$ -edges at the current function level. Each segment is represented by one or more points on the mesh, one for each boundary curve of the associated level set component for the function on the solid. In particular, we choose points on edges of the triangulation.

The format of the status structure (i.e., the number of segments or the number of boundary components for each segment) changes only at events. These events are the vertices of the VRG$^+$ and of the BRG. Those always lie in critical vertices of the mesh, i.e. in local minima, maxima and saddle points with respect to f, and can thus be located by considering their incident triangles. In between two events, corresponding entries of the status structure can be connected by a sequence of edges of the surface triangulation with monotonically increasing (or decreasing) f-values. We will refer to the process of generating such a connection as "growing of edges".

Algorithm 1 computeVRG$^+$

```
for all vertices v of mesh do
    if v is critical with respect to f then
        trace oriented level set polygons through v
        create a VRG+ -vertex V(v)
        add V to vertex_list
    end if
end for
sort vertex_list by increasing f-value
initialize the status structure as an empty list
for all V in vertex_list do
    for all status structure entries do
        grow edges until reaching f-value of V
    end for
    identify influenced edge segments in status structure
    update status structure and VRG+
end for
```

In order to identify whether an edge segment in the status structure is influenced by a vertex V, we use the level set polygons through V. Typically, the level set polygons are, at some point, intersected by the mesh-edges representing the edge segment. Only in a minimum V that is no vertex of the VRG, we trace level set polygons starting from the edge segments in the status structure and check which one surrounds V.

It should be noted that this algorithm imposes restrictions on the choice of the defining function f, since it requires certain operations inside f-level sets to be available. For example, we need to be able to determine the orientation of level set polygons and to decide whether a minimum is surrounded by a level set polygon or not. Since we are using linear functions here, this is always satisfied because all operations in f-level sets are broken down into operations in the plane.

*2.5. The reduced volume Reeb graph (*VRG$_\circ$, VRG$^+_\circ$ *)*

As described in Section 2.2, the number of tunnels through an object is related to the number of cycles in the Reeb graphs. Other branches of the Reeb graph which result in tree subgraphs only complicate the graph without providing additional topological information. Therefore, we will simplify the Reeb graph such that it contains only the cycle-information which we will use in our cutting algorithm. To this end, we perform three steps on the VRG, see also Fig. 9.

- Iteratively remove all terminal vertices of the VRG.

- Merge edges if they meet in a vertex of valency two. This may produce *loop edges*, i.e. edges with coinciding start- and end-vertex such as the edge from and to vertex 8 in Fig. 9.
- Split those loop edges by re-inserting a vertex removed in the previous step. We also split other edges where the interval of f-values spanned by the edge is much larger than the interval spanned by its vertices, such as the lower edge connecting vertices 6 and 7 in Fig. 9.

Note that we will not actually delete and merge edges in the data structure of the VRG. Instead, a new graph object is introduced for the $\mathrm{VRG}_\circ$, where each edge knows which edges of the VRG it consists of. Fig. 10 shows an example where reducing the Reeb graph gives a much more compact object.

In the shape decomposition algorithm, we will use the reduced version of the boundary component-labeled volume Reeb graph, denoted by $\mathrm{VRG}_\circ^+$. Since we will allow only one cut per edge of the $\mathrm{VRG}_\circ^+$, the third step is essential to find sufficiently many candidates for cutting surfaces and also to improve the quality of the selected ones.

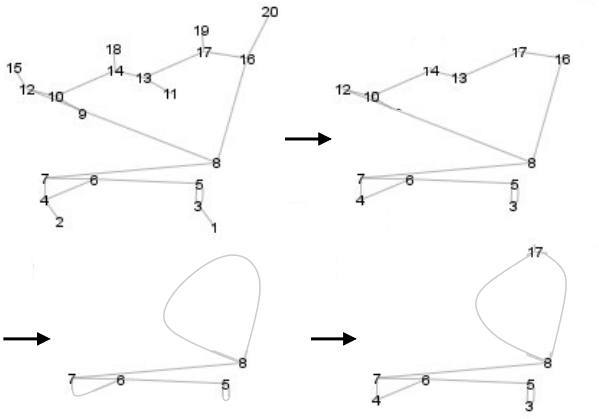


Figure 9: The three steps of reducing the VRG: Removing terminal vertices, merging edges and splitting loops.

3. The cutting algorithm

We begin by giving an overview of the overall algorithm, see Algorithm 2. The algorithm has been fully implemented in C++ based on the Computational Geometry Algorithms Library CGAL, see `www.cgal.org`.

In every iteration of the algorithm, we first choose a function f (see Section 3.1) and a quality bound q. In the beginning, we will set the quality bound rather high, aiming at high-quality cutting surfaces (see Section 3.2 for the description of our quality measure). In subsequent iterations, we decrease the quality bound to make sure that sufficiently many cuts are eventually accepted[1]. The

[1]Instead of this "greedy" approach it would be an option to employ more sophisticated strategies also. We did not yet explore this possibility, since we did not encounter examples where chosen cuts force the use of low quality cutting surfaces at a later stage of the segmentation algorithm. Currently, the main limitations of the method are caused by the choice of the Morse functions.

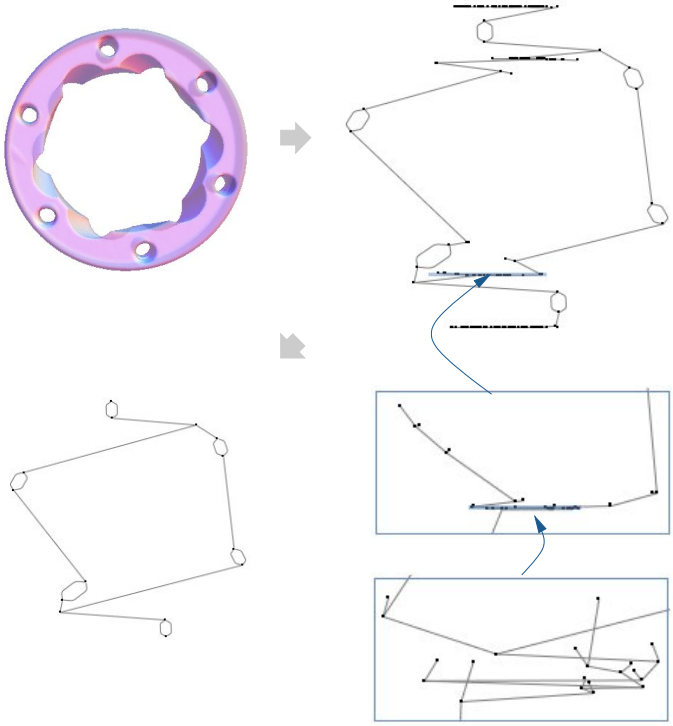


Figure 10: Efficiency of the reduced Reeb graph for the rolling stage model from the AIMATSHAPE shape repository shown on the top left. On the right, the volume Reeb graph with respect to the height function is shown with two zoomed-in areas. Due to small features on the object surface, the graph consists of 419 vertices and 202 edges. The reduced graph shown on the bottom left contains only 14 vertices and 20 edges while still representing the shape of the object. This reduction leads to computational advantages when analyzing connectivity of the graph.

$\mathrm{VRG}_\circ^+$ is computed using the boundary-based construction algorithm as described in Section 2.4.

For every edge e of the reduced volume Reeb graph $\mathrm{VRG}_\circ^+$, we compute cutting surfaces at $N \cdot k_e$ equally spaced intermediate f-levels, where the number N is defined by the user and k_e is the number of edges of the VRG^+ that have been merged into e. We choose the best candidate and sort the edges of the $\mathrm{VRG}_\circ^+$ by the quality of these candidate cutting surfaces.

Then, we select edges with good cuts if their removal simplifies the $\mathrm{VRG}_\circ^+$. Some of these good cuts are rejected in the following step if they do not split the object. In these two decisions, we analyze whether or not the edge forms part of a cycle in the graph, i.e., if the end vertices of the edge are connected elsewhere in the graph, see Section 3.3 for details.

Finally, the remaining good cuts are implemented in the mesh, see Section 3.4.

3.1. Choosing the function f

We will use a combination of three approaches.

- In the beginning, a collection of functions is given, or estimated from the object (using e.g. the three principal axes of the object or the coordinate axes).
- If all given functions have been used and non-resolvable tunnels remain, we can heuristically esti-

Algorithm 2 cutObject

```
while not finished do
    //Preparation, see Sections 3.1 and 3.2
    Choose next function f from Morse function list M.
    Decrease quality bound q if f was used with q before.
    Compute VRG_o^+ .
    for all edges of VRG_o^+ do
        compute a good cut candidate (using the labels).
    end for
    //Choose cuts, see Section 3.3
    sort edges by the quality of their cut candidates.
    cut_edges ← empty list
    for all edges e of VRG_o^+ , starting at best edge do
        if quality of e > q and in_cycle(e,1) then
            add e to cut_edges.
        end if
    end for
    //Reject some chosen cuts, see Section 3.3
    for all edges e in cut_edges do
        if in_cycle(e,0) then
            remove e from cut_edges.
        end if
    end for
    //Implement the cuts, see Section 3.4
    for all edges e in cut_edges do
        implement cut candidate of e.
    end for
    if cut_edges is empty then
        Generate new Morse function(s) as described in
        Section 3.1 and append them to M
    end if
end while
```

mate new functions to specifically cut certain non-resolvable tunnels. This will be described in more detail below.

- If there are still unresolved tunnels, it would be possible to use randomly generated functions, such as linear functions with randomly chosen gradients. This option is reported for the sake of completeness but has not been used in our examples.
- We decrease the quality bound if all functions have been used with the current one, see Algorithm 2. Alternatively one may also opt to decrease the quality bound after performing a certain number of iterations.

The success of the decomposition algorithm depends strongly on the selection of the functions. A simple theoretical guarantee can be obtained by considering linear functions: When considering sufficiently many linear functions, the segmentation is always successful provided that the object has only straight tunnels (i.e., tunnels where a straight line can be inserted without touching the object's boundary). While it would in principle be possible to rely on randomly generated functions, we found it more efficient to use a heuristic method to estimate suitable functions using the second approach in the above list. In the remainder of this section we describe this approach in more detail and discuss its limitations.

When analyzing the object using a function f, we may encounter non-resolvable tunnels which cannot be opened using f-level sets. However, the BRG with respect to f allows us to access the inner boundary component of the non-resolvable tunnel and its boundary curves at the top and bottom end of the corresponding edge of the BRG. Estimating the centers of these top and bottom curves, we can thus approximate the axis of the tunnel. We are then able to cut straight tunnels in the next pass of the algorithm, using a linear function with a gradient orthogonal to the tunnel axis. Since straight tunnels are frequently encountered in engineering objects, this heuristic criterion turns out to be very helpful.

An example is shown in Fig. 11, where we visualize suitable gradients of linear functions for each of the three holes. If several straight tunnels are present, we try to use Morse functions that can deal with at least two of them by choosing the cross product of the axes' direction vectors as gradient. In the example it suffices to use only one linear function, since the holes' axis are all parallel to a single plane.

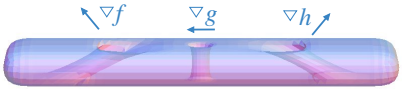


Figure 11: Object with three non-resolvable tunnels with respect to the height function. For each of them, we can generate suitable functions (f, g, h) for the next iteration.

In the example shown in Fig. 1, this heuristic function choice criterion is used. First, the given coordinate functions are applied. Thus, the object is split using two horizontal cuts in the first step. For the resulting top and bottom piece, vertical cuts are able to produce tunnel-free pieces. In the central piece, however, two tunnels remain which cannot be resolved using the coordinate functions. Thus, a linear function is generated whose gradient is simultaneously orthogonal to the axes of both tunnels.

Fig. 12 demonstrates the limitation of this approach in dealing with bent tunnels. In the first column, the axis-aligned cube contains a horizontal curved tunnel, which can be resolved using the given height function. The second cube contains a similar tunnel which is no longer aligned with a coordinate plane. Thus, it cannot be resolved using the given coordinate axis. Also, since the tunnel is not straight, the algorithm fails to estimate a good new function for opening the tunnel, and thus no de-

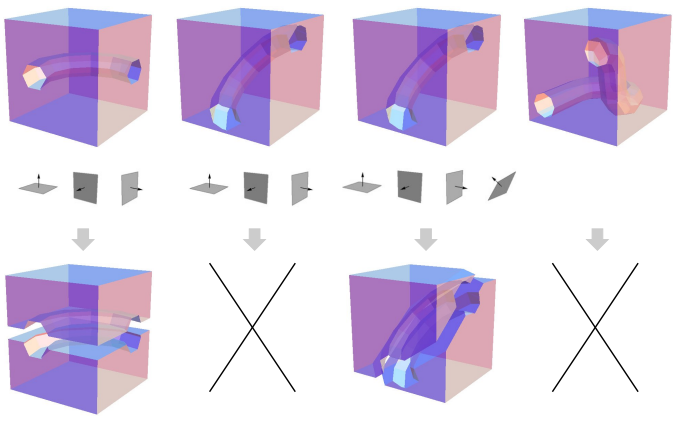


Figure 12: Limitation of the method in opening curved tunnels in an axis-aligned cube.

composition is generated. If, however, a suitable defining function is included in the given functions (third column), the cube is decomposed into contractible pieces. The tunnel in the cube in the last column cannot be opened using one planar cut. Therefore, whichever functions would be given as an input, the presented algorithm cannot decompose this object. In this case, one would either have to cut the cube a few times to simplify the tunnel, or include non-linear functions, where finding a suitable function for the given tunnel would remain a challenging problem.

3.2. The quality of cuts

The object is always cut by a level set of the current function f. We prefer cuts that form near-orthogonal angles with the object surface. Let T_c be the set of surface triangles intersected by a cut c. If n_t denotes the unit normal vector of a triangle t and F is the unit vector in the direction of ∇f, then the quality of the cut c is determined by

$$q(c) = 1 - \max_{t \in T_c}(|n_t.F|),$$

where $0 \leq q \leq 1$ and bigger values of q are preferred: If all triangles are orthogonal to the cutting surface, then their normals are orthogonal to F and $q(c)$ evaluates to 1. On the other hand, if one intersected triangle makes a very shallow angle with the cutting surface, the scalar product approaches 1 and $q(c)$ becomes zero.

This specific choice of the quality criterion is motivated by the ultimate goal of our work, namely to obtain segments that can be parameterized by NURBS. In this case it is advantageous to obtain transversal intersections along the entire intersection curve, since this is beneficial for the parameterization quality. Even a few places with shallow intersection angles might spoil the results. Other applications of the segmentation algorithm may call for a different choice of the quality criterion. In particular, this also applies to the case of noisy meshes, where the evaluation of our quality criterion may be unreliable and using other measures (based on averaging) will make the method more robust.

In the initial sorting of the $\mathrm{VRG}^+_\circ$-edges by the quality of their cut candidates, we additionally compare the number of boundary components of the cutting surface. We list all cutting surfaces without holes first, sorted by their cutting quality, then all cuts with one inner boundary component, etc. In this way, cutting surfaces with inner boundaries are only used if there is no way to achieve a similar reduction of the $\mathrm{VRG}^+_\circ$ using only hole-free cuts of sufficient quality.

3.3. Choosing the cuts

For $k \in \{0, 1\}$ and an edge e, the test in_cycle(e, k) returns true iff there is a cycle in the $\mathrm{VRG}^+_\circ$ containing e and at most k edges (apart from e itself) which have been marked as cut edges up to this point. This can be checked using a simple region growing algorithm on the vertices of the $\mathrm{VRG}^+_\circ$.

We first use this test with $k = 1$ to decide if cutting an edge e of the $\mathrm{VRG}^+_\circ$ simplifies the graph. At this point, we check for cycles containing up to one previously cut edge, since we do not only want to open tunnels, but to construct separate objects. In a following step, some of the chosen edges are rejected if they do not split the object, using the test with $k = 0$.

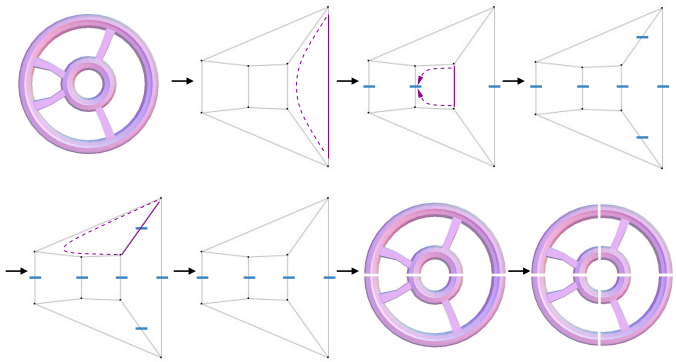


Figure 13: Choosing the cuts. First row: Candidate cuts better than the quality bound are chosen if they are contained in a cycle with at most one previously cut edge. Second row: Previously chosen cut edges are rejected if the graph contains a path of non-selected edges connecting the two ends of the edge. In a second pass of the algorithm, better cutting surfaces are generated to open the remaining tunnels using a different Morse function.

We describe the steps of the algorithm for a simple example, see Fig. 13. Among the available cuts, we first test the edge with the highest quality cutting surface, see second image. Since it is contained in a cycle, we add it to cut_edges, similarly the second and third tested edge. For the edge tested in the third image, we find a cycle containing one previously cut edge. Thus, the edge is cut as well in order to split the object into two pieces. Two more cuts are chosen, then the algorithm stops because the subsequent cuts are not better than the quality bound.

In the following step, the in_cycle(e, k) test with $k = 0$ is applied to each of the cut_edges. For example, the edge checked in the fifth image in Fig. 13 is contained in a cycle containing no other cut edges. Thus, this cut would insert a surface inside one of the resulting pieces instead of splitting the object. Therefore, it is rejected, and only the four central cuts are implemented in the first iteration of the algorithm[2].

After another pass of the algorithm using a different Morse function f, the final result shown in the last image of Fig. 13 is obtained. Fig. 14 shows the results obtained by this procedure for two additional examples.

3.4. Implementing the cuts

We work with a triangulated surface description of the object and with linear Morse functions. In order to execute a cut for an edge e of the $\mathrm{VRG}^+_\circ$, we choose a function value in the function span of e and trace all boundary

[2]If one is only interested in opening the tunnels and does not require the result to consist of separate objects, the algorithm should be modified as follows. Firstly, good cuts are only chosen if they are contained in a cycle without any previously cut objects, so the edge checked in the third image in Fig. 13 is not cut. Secondly, the step of rejecting some chosen edges must be skipped, since it would reject all previously inserted cuts in this case.

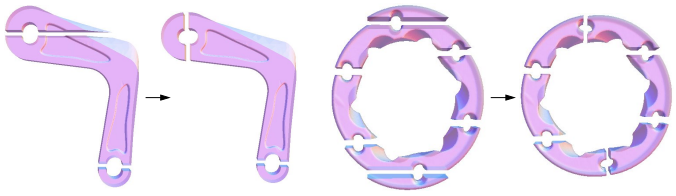


Figure 14: The effect of using different quality bounds on two more objects, the TERRIFIC demonstrator, data courtesy of Siemens AG (left) and the Rolling Stage model from the AIMATSHAPE shape repository (right). Only resolvable tunnels are detected, so all loops could be cut using the same function in the first image. If some cuts are rejected in the first step, a better decomposition is achieved after another pass of the algorithm with a different function.

polygons of the corresponding level set component, starting from references on the surface which are stored during the construction of the Reeb graphs.

In every point of such a polygon, an edge of the surface triangulation is split into two segments. Then, the incident faces have to be re-triangulated, introducing new edges into the mesh. If the same facet is intersected by other intersection polygons that have been computed already, those polygons may have to be updated before their next use, adding intersection points on the newly introduced edges.

Finally, a triangulation of the inserted cutting surface is generated. Since we use only linear functions so far and the cutting surfaces are planar, this can be achieved without adding new vertices to the mesh. We use the algorithm described in Chapter 3 of [4].

After all chosen cuts have been implemented, the separate polyhedron pieces are extracted using a simple surface flooding algorithm. Implementing the cuts may introduce very thin or small triangles, for example if the mesh is cut very close to original mesh vertices. In the resulting polyhedron pieces, we therefore check for edges and facets whose size is in the order of machine precision and carefully eliminate them from the mesh.

3.5. Results

Fig. 15 visualizes the decomposition procedure for the Rolling Stage object. First, the height function f_1 is used to insert horizontal cutting surfaces. Four cutting surfaces comply with the given quality bound $q_1 = 0.5$, decomposing the object into two pieces with remaining tunnels. These are analyzed with a new function f_2, which is chosen as an arbitrary function orthogonal to f_1. Each part is decomposed into one contractible piece and one part with a remaining tunnel. The algorithm tries to estimate more defining functions f, but does not generate cutting surfaces complying with the quality bound. Thus the previously tested functions are used once more on the remaining pieces with a quality bound $q_2 = 0$ (basically accepting all cuts that simplify the graph). Now, the height function resolves the remaining tunnels. Note that while the labels of the VRG$^+$ are not noted here, they are used for the construction of cutting surface to access all necessary boundary components.

Fig. 16 visualizes the decomposition of the Stanford bunny which was pierced four times. Since the tunnels are all narrow, long and not aligned with a coordinate plane, none of them can be opened using the coordinate functions. The algorithm thus automatically determines the axis of each non-resolvable tunnel and estimates suitable functions. Firstly, a function is generated whose gradient is orthogonal to the two non-resolvable tunnels in the bottom right. Using this function, the blue and green piece are separated from the magenta piece. Secondly, for the magenta-colored piece, the algorithm generates another function with a gradient orthogonal to the remaining tunnels' axes. Thus, the magenta piece is split into two more pieces.

The result of the described decomposition algorithm applied to several different meshes is shown in Table 1. The first row describes the input data. The given object is shown along with the number of triangles in its surface mesh and the number of tunnels of the object which are resolved by the decomposition. If several values for the quality bound q_i are given, at first the higher value is used with all given and generated functions, before resorting to the lower value. If necessary, the same functions are then used once more with quality bound q_i set to zero, accepting all cuts. Since, in the selection of cutting surfaces, those with good quality are tested first, the algorithm still generates reasonable cuts. The given linear functions f_j are depicted by their gradient vector and a level set. The leftmost function of each example is tested first, after using the rightmost function additional functions may be generated (e.g. in example G) before returning to the first function with a smaller quality bound.

The second row describes the output of the decomposition algorithm. Alongside a picture of the decomposition, the number of contractible pieces and the number of inserted cutting surfaces is given. Furthermore, the quality of the worst used cutting surface (q_{min}) and the average cutting-quality (q_{avg}) are specified as well as the time taken for the whole computation.

Examples B and C demonstrate the effect of swapping the order of the given defining functions. Also examples E and F describe two different decompositions of one object, where the decomposition quality in example F is significantly better due to the additional given functions. Examples G and H give additional information about the decomposition shown in Figures 1 and 16, respectively.

Example I (the "knotty" object from Cindy Grimm's web page[3]) is a failure case, since the non-straight tunnel in the model's interior cannot be opened by using a linear Morse function. The algorithm reduces the genus of the solid but does not return a decomposition into contractible pieces. Fortunately, objects of this type do not appear in typical mechanical engineering applications, but they may well be present in other fields.

4. Conclusion

We presented a novel algorithm for the decomposition of solid objects into contractible pieces. Given a Morse

[3] `web.engr.oregonstate.edu/~grimmc/meshes.php`

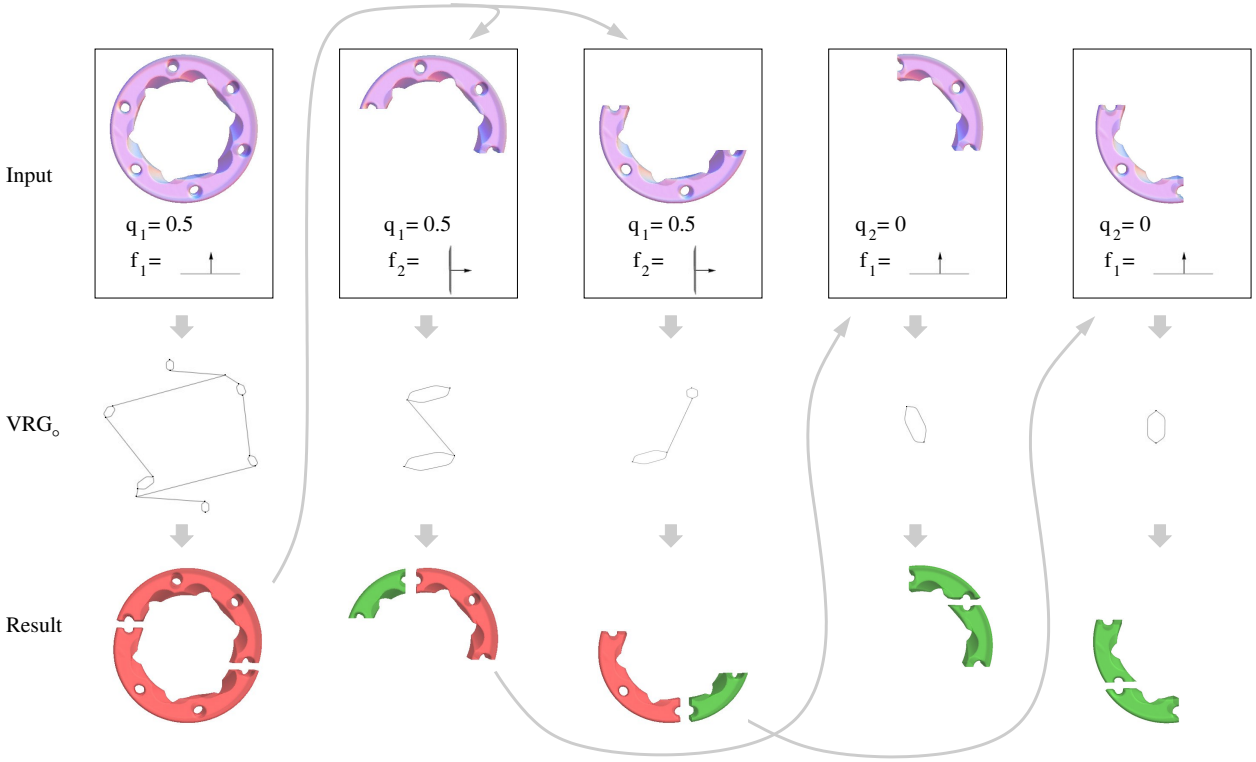


Figure 15: Decomposition of the Rolling Stage model from the AIMATSHAPE shape repository with approximately 400 000 triangles. The worst inserted cutting surface has quality $q = 0.15$, the total computation takes 13 seconds.

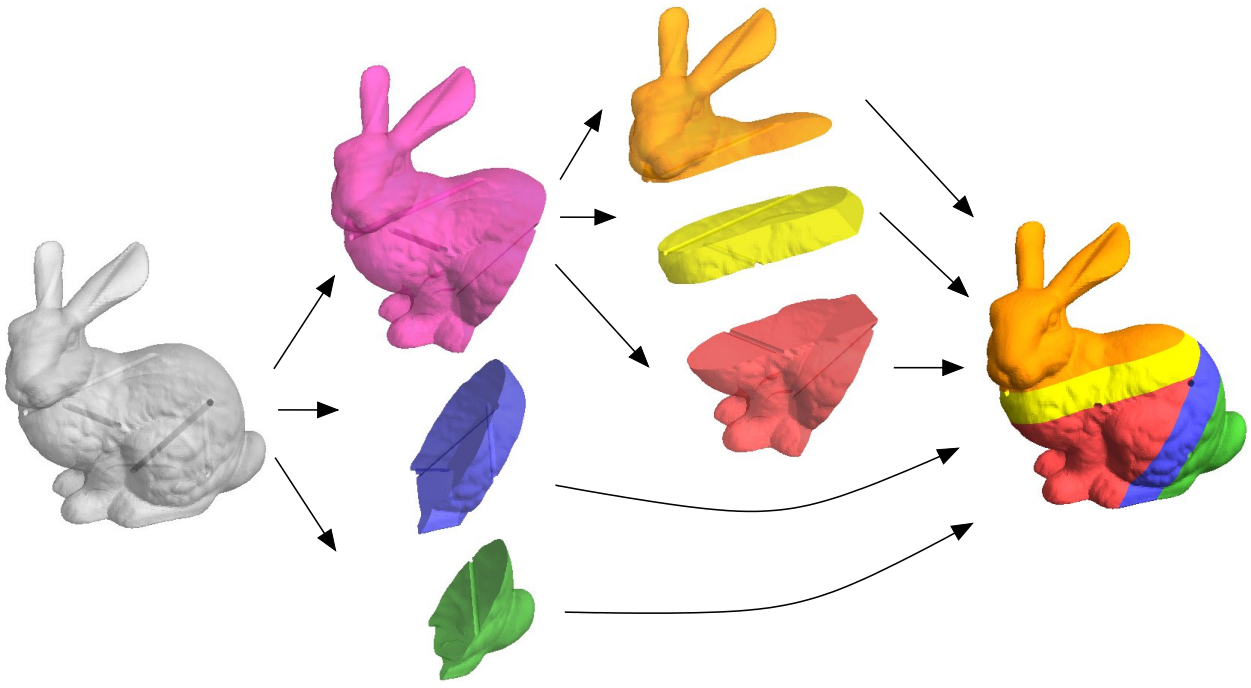
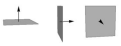
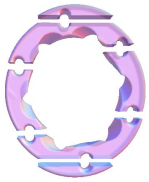
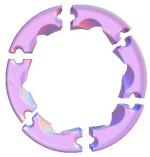
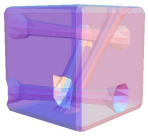


Figure 16: Decomposition of a Stanford bunny with four non-resolvable tunnels, using only functions which were generated automatically by estimating the tunnels' axes, using the heuristic method described in Section 3.1.

Table 1: Results of the decomposition algorithm.

example	A	B	C	D	E	F
object						
triangles	3672	1088	1088	228	382242	382242
tunnels	2	5	5	2	7	7
q_i	0.5, 0	0.5, 0	0.5, 0	0.5, 0	0	0.5, 0
f_j						
result						
pieces	3	4	4	3	6	6
cuts	4	8	8	4	12	12
q_{min}	0.84	0.85	0.52	0.70	0.08	0.50
q_{avg}	0.87	0.90	0.73	0.76	0.33	0.69
time	0.05s	0.02s	0.01s	0.01s	8.43s	11.14s

G	H	I
		
3136	76476	1200
4	4	2
0	0.5, 0	0.5, 0
		
		
7	5	2
12	8	2
0.20	0.12	0.19
0.42	0.30	0.23
0.08s	20.05s	0.55s

function f we combine the Reeb graphs of the object's volume and of its boundary surface. We then examine all edges of a reduced version of this graph. If their removal helps us to split the object – and this can be decided using the Reeb graph – then we cut it with a level set of f.

In order to optimize the quality of the resulting pieces, we first compute a good cutting surface for each edge, and we prefer the edges that provide the best ones. Additionally, cuts are only inserted up to a certain quality bound. In a final step, some of the chosen cuts are rejected if they do not split the object.

In principle, this procedure is able to eliminate all resolvable tunnels with respect to the given Morse function, and this would be achieved when using no (or a very low) quality bound. In practice, not all resolvable tunnels may be opened since the necessary cuts for opening them may not pass the quality test. In addition, all non-resolvable tunnels with respect to the Morse function remain. Typically, these tunnels can be dealt with by repeating the procedure with a different function f. When using sufficiently many linear functions, we can guarantee to open all straight tunnels. In fact, it is possible to segment any object without voids when considering sufficiently complex Morse functions, but the construction of such functions may be challenging. Also, there is a trade-off between the complexity of the Morse function f and the computational difficulties that arise when implementing the operations in its level sets, which are described in the last paragraph of Section 2.4. Piecewise linear Morse functions might lead to a good compromise between the conflicting requirements of segmentation and Reeb graph computation.

In our examples, the presented method generates very satisfactory decompositions of the given objects. Using a heuristic criterion, we can always find suitable cuts to open straight tunnels. Currently, our implementation automatically generates segmentations for objects with only straight tunnels, or tunnels with a very large diameter compared to their length. Bent tunnels may be opened automatically as well, if suitable functions are given or generated. The user can control the quality of the resulting shapes by using different starting functions or different bounds for accepting cutting surfaces.

The choice of suitable Morse functions remains an interesting topic for future research. So far, we have used only linear functions. In order to allow curved cutting surfaces, the approach could be generalized to nonlinear functions, provided that they allow for a boundary-based construction of the Reeb graphs. In addition, it could be worthwhile to explore alternative approaches, that are not

based on Morse theory, using structures such as the medial axis, which contain more geometric information than the Reeb graph.

References

[1] S. Berretti, A. del Bimbo, and P. Pala. 3D Mesh decomposition using Reeb graphs. *Image and Vision Computing*, 27:1540–1544, 2009.

[2] 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.

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

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry*. Springer, 2008.

[5] V. de Silva, E. Munch, and A. Patel. Categorified Reeb graphs. *Discrete and Computational Geometry*, 55:854–906, 2016.

[6] E. de Verdière and F. Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. In *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 478–490. Springer Berlin Heidelberg, 2004.

[7] T. K. Dey, F. Fan, and Y. Wang. An efficient computation of handle and tunnel loops via Reeb graphs. *ACM Trans. Graph.*, 32(4):32:1–32:10, 2013.

[8] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066, 2006.

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

[10] H. Doraiswamy and V. Natarajan. Computing Reeb graphs as a union of contour trees. *IEEE Trans. Vis. Comp. Graph.*, 19(2):249–262, 2013.

[11] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Proc. Symp. Comput. Geom.*, pages 361–370. ACM, 2003.

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

[13] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical Morse complexes for piecewise linear 2-manifolds. In *Proc. Symp. Comput. Geom.*, pages 70–79. ACM, 2001.

[14] J. Gravesen, A. Evgrafov, D. Nguyen, and P. Nørtoft. Planar parametrization in isogeometric analysis. In *Mathematical Methods for Curves and Surfaces*, pages 189–212. Springer, 2014.

[15] M. Hajij, T. Dey, and X. Li. Segmenting a surface mesh into pants using Morse theory. *Graphical Models*, 88:12–21, 2016.

[16] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. SIGGRAPH Conf. Computer Graphics*, pages 203–212. ACM, 2001.

[17] B. Jüttler, M. Kapl, D.-M. Nguyen, Q. Pan, and M. Pauley. Isogeometric segmentation: The case of contractible solids without non-convex edges. *Computer-Aided Design*, 57:74–90, 2014.

[18] N. Karmakar, A. Biswas, and P. Bhowmick. Reeb graph based segmentation of articulated components of 3D digital objects. *Theoretical Computer Science*, 624:25–40, 2016.

[19] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *Vis. Comp.*, 21(8-10):649–658, 2005.

[20] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Mesh scissoring with minima rule and part salience. *Comput. Aided Geom. Des.*, 22:444–465, 2005.

[21] X. Li, X. Gu, and H. Qin. Surface matching using consistent pants decomposition. In *Proc. Symp. on Solid and Physical Modeling*, pages 125–136. ACM, 2008.

[22] X. Li, T. W. Woon, T. S. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proc. Symp. on Interactive 3D Graphics*, pages 35–42. ACM, 2001.

[23] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra and its applications. *Comput. Aided Geom. Des.*, 25:503–522, 2008.

[24] J.-M. Lien, J. Keyser, and N. M. Amato. Simultaneous shape decomposition and skeletonization. In *Proc. Symp. on Solid and Physical Modeling*, pages 219–228. ACM, 2006.

[25] R. Ling, J. Huang, B. Jüttler, F. Sun, H. Bao, and W. Wang. Spectral quadrangulation with feature curve alignment and element size control. *ACM Trans. Graph.*, 34(1), 2014. Article no. 11.

[26] T. Martin, E. Cohen, and R. M. Kirby. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. *Computer Aided Geometric Design*, 26(6):648–664, 2009.

[27] F. Massarwi and G. Elber. A B-spline based framework for volumetric object modeling. *Computer-Aided Design*, 78:36 – 47, 2016.

[28] D.-M. Nguyen, M. Pauley, and B. Jüttler. Isogeometric segmentation. Part II: On the segmentability of contractible solids with non-convex edges. *Graphical Models*, 76(5):426–439, 2014. Geometric Modeling and Processing 2014.

[29] X. Ni, M. Garland, and J. C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Trans. Graph.*, 23(3):613–622, 2004.

[30] L. Papaleo and L. De Floriani. Semantic-based segmentation and annotation of 3d models. In *Image Analysis and Processing - ICIAP 2009*, volume 5716 of *Lecture Notes in Computer Science*, pages 103–112. Springer Berlin Heidelberg, 2009.

[31] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: Simplicity and speed. *ACM Trans. Graph.*, 26(3), 2007. Article no. 58.

[32] G. Patanè, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the Reeb graph. *IEEE Trans. Vis. Comp. Graph.*, 15(4):583–595, 2009.

[33] M. Pauley, D.-M. Nguyen, D. Mayer, J. Špeh, O. Weeger, and B. Jüttler. The isogeometric segmentation pipeline. In B. Jüttler and B. Simeon, editors, *Isogeometric Analysis and Applications 2014*, volume 107 of *LNCSE*, pages 51–72. Springer, 2015.

[34] D. Reniers and A. Telea. Skeleton-based hierarchical shape segmentation. In *Proc. Int. Conf. on Shape Modeling and Applications*, pages 179–188. IEEE, 2007.

[35] A. Shamir. A survey on mesh segmentation techniques. *Comp. Graph. For.*, 27(6):1539–1556, 2008.

[36] B. Strodthoff and B. Jüttler. Layered Reeb graphs for three-dimensional manifolds in boundary representation. *Computers & Graphics*, 46:186–197, 2015.

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

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

[39] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Topology driven 3d mesh hierarchical segmentation. In *Proc. Int. Conf. on Shape Modeling and Applications*, pages 215–220. IEEE, 2007.

[40] G. Xu, B. Mourrain, R. Duvigneau, and A. Galligo. Optimal analysis-aware parameterization of computational domain in 3d isogeometric analysis. *Computer-Aided Design*, 45(4):812 – 821, 2013.

[41] Y. Yasui, S. McMains, and T. Glau. Pool segmentation for predicting water traps. *Journal of Manufacturing Systems*, 37:494–504, 2014.

[42] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24:1–27, 2005.

[43] X. Zhang, J. Liu, Z. Li, and M. Jaeger. Volume decomposition and hierarchical skeletonization. In *Proc. SIGGRAPH Int. Conf. on Virtual-Reality Continuum and Its Applications in Industry*, pages 17:1–17:6. ACM, 2008.

[44] Y. Zhang, W. Wang, and T. J. R. Hughes. Solid T-spline construction from boundary representations for genus-zero geometry. *Computer Methods in Applied Mechanics and Engineering*, 2012.