NFN - Nationales Forschungsnetzwerk

Geometry + Simulation

http://www.gs.jku.at

Isogeometric Segmentation. Part I: Decomposing contractible solids without non-convex edges

B. Jüttler, M. Kapl, Qing Pan

G+S Report No. 7

October 2013

Der Wissenschaftsfonds.

JOHANNES KEPLER UNIVERSITY LINZ

Isogeometric Segmentation. Part I: Decomposing contractible solids without non-convex edges

Bert Jüttler[a], Mario Kapl[a,*], Qing Pan[a]

[a]*Institute of Applied Geometry, Johannes Kepler University, Linz, Austria*

Abstract

We present a novel technique for segmenting a three-dimensional solid with 3-vertex-connected edge graphs consisting of only convex edges into a collection of topological hexahedra. Our method is based on the edge graph, which is defined by the sharp edges between the boundary surfaces of the solid. We repeatedly decompose the solid into smaller solids until all of them belong to certain classes of predefined base solids. The splitting step of the algorithm is based on simple combinatorial and geometric criteria. The extension of our method to solids with non-convex edge graphs of insufficient connectivity will be presented in a forthcoming paper.

Keywords:
Isogeometric analysis, coarse volume segmentation, edge graph, cutting loop, cutting surface

1. Introduction

Isogeometric analysis (IGA) is a novel framework for numerical simulation that relies often on a NURBS volume representation of the computational domain. It ensures the compatibility of the geometry description with the prevailing standard in Computer Aided Design [1, 2]. Additional advantages include higher rates of convergence and increased stability of the simulation results. These beneficial effects are due to the increased smoothness and the higher polynomial degrees of the functions used to represent the simulated phenomena.

However, a NURBS representation of the computational domain, which is often the volume of a solid object or the volume surrounding a solid object, is not provided by a typical CAD model. In connection with the advent of isogeometric analysis, several authors presented algorithms for *creating a NURBS volume representation from a given CAD model*:

*Corresponding author
Email addresses: bert.juettler@jku.at (Bert Jüttler), mario.kapl@jku.at (Mario Kapl), panqing@lsec.cc.ac.cn (Qing Pan)

– Martin et al. [3] describe a method to generate a trivariate B-spline representation from a tetrahedral mesh. First, a volumetric parametrization of the genus-0 input mesh by means of discrete harmonic functions is constructed. This initial parametrization is then used to perform a B-spline volume fitting to obtain a B-spline representation of a generalized cylinder. An extension of this work to more general objects (e.g. to a genus-1 propeller) is presented in [4].

– Another parametrization method for a generalized cylinder-type volume is proposed in [5]. A NURBS parametrization of a swept volume is generated by using a least-squares approach with several penalty terms for controlling the shape of the desired parametrization. Among other applications, this method can be used to generate volume parameterizations for blades of turbines and propellers.

– Xi et al. [6] present a volume parametrization technique for a multi-block object. The parametrization of a single block is constructed by minimizing a quadratic objective function subject to two constraints. While one condition ensures the injectivity of the single B-spline parametrizations, the other condition guarantees C^1-smoothness between the blocks.

– Another volume parametrization method [7] generates first a mapping from the computational domain, which is given by its boundary, to the parameter domain by means of a sequence of harmonic maps. The parametrization of the computational domain is then obtained by a B-spline approximation of the inverse mapping.

– Given a boundary representation of a solid as a T-spline surface, which is assumed to have genus zero and to contain exactly eight extraordinary nodes, the algorithm in [8] constructs a solid T-spline parametrization of the volume.

– Further approaches to volume parametrization are described in [9, 10, 11, 12].

Since many of the existing methods for (NURBS) volume parametrization are restricted to simple objects (cf. [6]) or to decompositions of more complex objects into simple ones, an algorithm for splitting a solid represented by a CAD model into a collection of simpler solids is of interest. In particular, *decompositions into solids that are topologically equivalent to hexahedra or tetrahedra* are desirable, since these objects can be easily parametrized by tensor-product NURBS volume patches.

– The decomposition of a convex polyhedron into a collection of tetrahedra is a well-studied problem [13, 14]. A tetrahedralization of a convex polyhedron can be also generated by barycentric subdivision (cf. [15]), which can be applied to any connected polyhedral complex, see [16].

– For general polyhedra, several methods for decomposing them into smaller convex polyhedra have been studied, e.g. [17, 18, 19]. In contrast to the convex case [13], it is not always possible to obtain a tetrahedralization without adding new vertices, cf. [17].

– A well-established approach to the decomposition of a CAD model is the use of the geometric information that is provided by its features (e.g. sharp edges). Chan et al. [20] describe a volume segmentation algorithm that can be used for prototyping applications. The initial solid is repeatedly decomposed into smaller ones until all resulting models belong to a class of so-called "producible" solid components. It is ensured that the union of the constructed solids represents again the initial object.

– Other feature-based methods that have been described in the literature, (e.g. [21, 22]) decompose polyhedral objects and special curved objects (i.e. objects with planar and cylindrical surfaces) into maximal volumes. In the method described in [21], the maximal volumes are always convex objects, whereas in [22] in some cases the maximal volumes may include objects with a few non-convex edges, too.

– Another approach to the segmentation of a CAD model is the representation as a hexahedral mesh with many hexahedra of approximately uniform size and shape. This is usually referred to as the problem of hex(ahedral) mesh generation. Due to the importance of hex meshes for numerical simulation, this problem has continuously attracted attention over the years. A feature-based algorithm for generating such meshes was introduced in [23], consisting of the following steps. The first phase is devoted to the feature recognition, which provides a guiding frame for the decomposition of the CAD model. Secondly, cutting surfaces are constructed, which split the initial solid into hex-meshable volumes. Further examples of hexahedral meshing algorithms are described in [24, 25, 26, 27, 28, 29, 30].

In contrast to these approaches, our goal is the decomposition of a CAD-model into a *small* number of topological hexahedra, which can be parametrized by single trivariate tensor-product NURBS-patches. More precisely, we consider the following *Isogeometric Segmentation Problem:*

> Given a solid object $\mathcal{S}$ (represented as a CAD model), find a collection of mutually disjoint topological hexahedra $\mathcal{H}_i$ $(i = 1, \ldots, n)$ whose union represents $\mathcal{S}$. The shape of the topological hexahedra need not to be uniform, and the hexahedra are not required to meet face-to-face, thereby allowing T-joints. However, the number n of topological hexahedra should be relatively small.

Each of the topological hexahedra can be represented as a trivariate NURBS volume, which can then be used for performing a numerical simulation using the isogeometric approach. By using a small number of topological hexahedra, it is possible to exploit the regular tensor-product structure on each of them. Since the individual NURBS volumes may not meet face-to-face, advanced techniques (e.g., based on discontinuous Galerkin discretizations) for coupling the isogeometric discretization are required. In the context of IGA, such techniques are currently being investigated in [31].

We present a new splitting algorithm that can be seen as the first step towards solving the IGA segmentation problem for general solid objects. In this paper we shall consider genus-zero objects with only convex edges; the extension to more general solids will be

discussed in a follow-up paper [32]. Future work will also address decompositions that allow a more direct coupling of the various subdomains.

The splitting algorithm is based on the *edge graph* of the given solid. We repeatedly decompose the edge graph into smaller sub-graphs, until all sub-graphs belong to a certain class of predefined base solids. The base solids can then be represented by a collection of topological hexahedra.

We introduce a cost function for identifying the "best" possible splitting in each step. This selection by the cost function is based on simple combinatorial and geometric criteria. In principle, exploring all possibilities would allow to find the decomposition with the smallest number of hexahedra. In practice, however, it is preferable to find a solution that is close to optimal with less computational effort. Several examples will demonstrate that this aim is achieved by the splitting algorithm.

The remainder of the paper is organized as follows. We introduce some basic definitions in Section 2. In particular, we explain the concept of the edge graph of a solid and state the required assumptions. Section 3 describes the main idea of our segmentation algorithm, which is based on the edge graph of the solid. Section 4 provides a theoretical justification for the proposed approach. In order to obtain a near-optimal result, we introduce in Section 5 the concept of the cost function, which provides an automatic possibility to guide the segmentation steps based on simple combinatorial and geometric criteria. Section 6 presents decomposition results for different example solids.

2. Solids and edge graphs

We consider a solid object $\mathcal{S}$ in boundary representation (BRep). It is defined as a collection of edges $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$, faces $F = \{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$, and vertices $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_\ell\}$. The faces are joined to each other along the edges, and the edges meet in vertices. The faces are generally free-form surfaces (e.g., represented as trimmed NURBS patches), and the edges are free-form curves (e.g., represented by NURBS curves).

By considering only the edges and the vertices, we obtain the *edge graph* $\mathcal{G}(\mathcal{S})$ of the solid. Consider the normal plane of an edge at an inner point. It intersects the two neighboring faces in two planar curve segments, and it contains the two normal vectors of these faces. If the two normal vectors do not separate the two tangent vectors of the planar curve segments (oriented such that they point away from the edge), then the edge is said to be convex at this point. An edge that is convex at all inner points is *convex*, otherwise it is *non-convex*.

Throughout the paper we make the following assumptions.

A1. The solid $\mathcal{S}$ is *contractible*, i.e., it is homeomorphic to the unit ball. Consequently, neither voids (i.e. hollow regions within the solid) nor tunnels (holes through the solid) are present.

A2. All edges are convex, any two vertices are connected by at most one edge, and all vertices of any face are mutually different.

The first assumption implies that the edge graph is a *planar graph*, in the sense that it can be embedded into the plane such that the embedded vertices are mutually different and the embedded edges intersect each other in vertices only.

Note that assumption A2 does not imply that the solid $\mathcal{S}$ is convex itself, due to the presence of free-form boundaries.

Solids not satisfying these assumptions can be dealt with by an extension of the methods presented in this paper. This will be the topic of the follow-up paper [32].

The construction of the edge graph from a given 3D geometry will not be discussed in this paper. In most cases, the edge graph of the solid can be derived directly from the CAD data. For a solid that is represented by a triangular mesh, the edge graph can be generated by detecting the sharp edges of the triangular mesh, possibly followed by cleaning and repairing steps.

We recall the following definition [33]:

Definition 1. An (edge) graph $\mathcal{G}$ is said to be *k-vertex-connected* if it has at least $k+1$ vertices and it remains connected after removing any set of $k-1$ vertices.

We will consider solids that satisfy the following additional assumption (again, the more general case will be addressed in [32]):

A3. The edge graph is 3-vertex-connected.

According to the Steinitz Theorem in polyhedral combinatorics, any planar graph that satisfies Assumption A3 can be obtained from the edges and vertices of a convex polyhedron, and the graphs satisfying Assumption A3 are therefore called *polyhedral* graphs [34, 35].

Figure 1 shows three examples of solids, their edge graphs and the associated planar embeddings. In these examples, which will be used throughout this paper, the boundary faces are represented as triangular meshes, and the edge graphs were generated by an edge detection method.

If two vertices or edges belong to the same face, then we say that they *share* a common face. For future reference we state the following observation.

Lemma 2. *The edge graph is 3-vertex-connected if and only if any two non-neighboring vertices of any face do not share another face.*

PROOF. We show that the negations of the two statements are equivalent.

First we consider an edge graph that possesses a face $\mathbf{f}_k$ with two non-neighboring vertices $\mathbf{v}_i, \mathbf{v}_j$ sharing this face and another face $\mathbf{f}_\ell$, cf. Figure 2, left. The two vertices can then be connected by two additional edges in the graph (shown in blue), one within each of the two faces. The loop formed by these two edges splits the edge graph into two disjoint subsets. Consequently, after removing the two vertices, the graph splits into two sub-graphs, thereby contradicting the assumption that it is 3-vertex-connected.

On the other hand, consider an edge graph that is not 3-vertex-connected. Clearly, it is at least 2-vertex-connected; otherwise Assumption A2 would be violated, since the

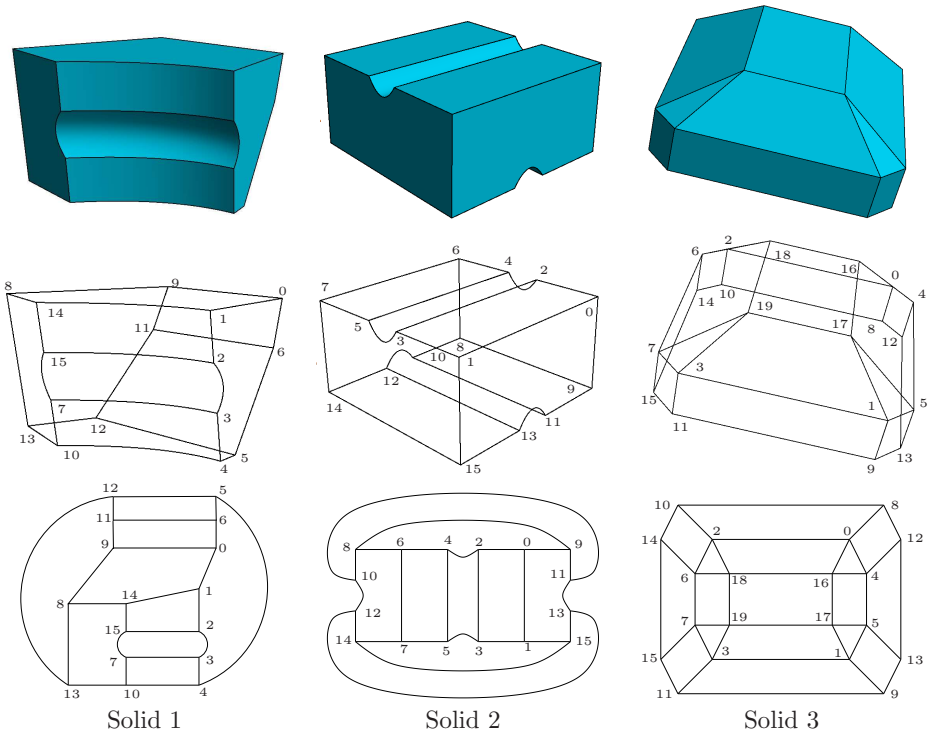

Figure 1: First row: Three solids with sharp edges. Second row: The associated edge graphs with only convex edges. Third row: The corresponding planar embeddings.

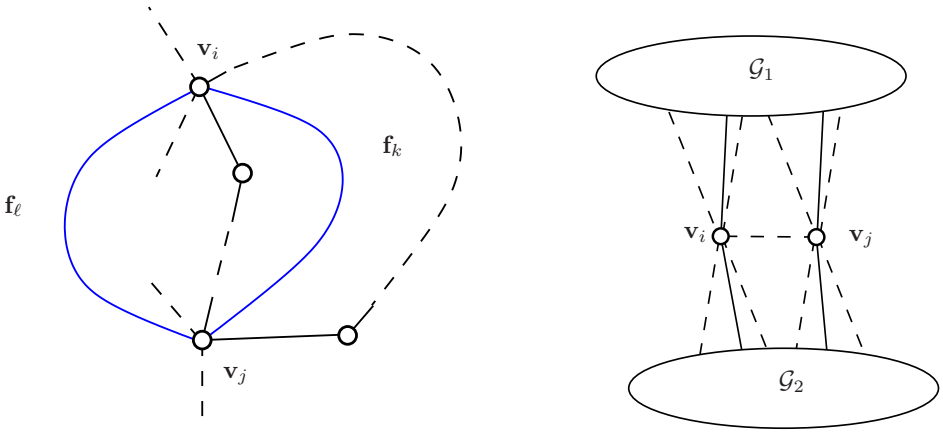

Figure 2: Left: An edge graph that possesses a face with two non-neighboring vertices sharing two faces is not 3-vertex connected. Right: A non-3-vertex-connected graph possess a face with two non-neighboring vertices sharing more than one face.

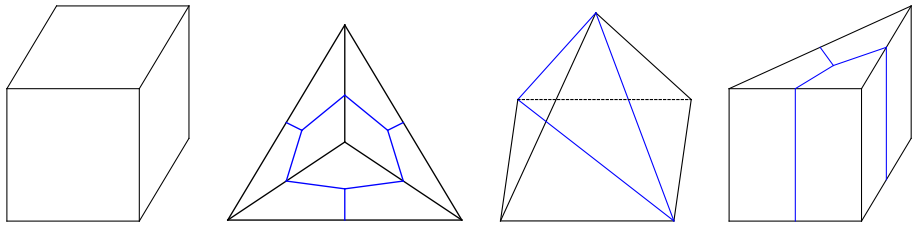

Figure 3: Four types of base solids. Each base solid can be represented by a mesh of topological hexahedra.

vertex whose deletion splits the graph would appear in the same face twice. Consider the situation in Figure 2, right. Deleting the two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ splits the graph into two disjoint components $\mathcal{G}_1$ and $\mathcal{G}_2$. The dashed lines represent edges that may or may not be present. Even when connected by an edge, the two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ are non-neighboring vertices of the outer face but they share more than one face. However, $\mathbf{v}_i$ and $\mathbf{v}_j$ cannot be connected by more than one edge, because of Assumption A2. Thus, if a graph is not 3-vertex-connected then there exist two non-neighboring vertices of a face that share more than one face. □

3. Solid splitting algorithm

We say that a solid is a *topological hexahedron* if its edge graph is equivalent to the edge graph of a cube and if all edges are convex. Our goal is to generate a decomposition of the given solid into a collection of such topological hexahedra.

In order to achieve this goal, we split the solid and its edge graph into two solids with smaller edge graphs. More precisely, the edge graphs of the two solids contain only vertices of the original edge graph, but they may possess additional edges.

We apply this decomposition step repeatedly to each resulting solid and associated edge graph, until we arrive at a sufficiently simple solid, which we will call a *base solid*.

In this paper we use four types of base solids, see Figure 3. In addition to topological hexahedra, we also allow topological tetrahedra, pyramids, and prisms. These are defined in the same way as topological hexahedra.

In principle, using only tetrahedra would be sufficient, since each tetrahedron can be split into four topological hexahedra. In order to obtain a *small* number of topological hexahedra, however, it is advantageous to consider hexahedra as base solids, too. Otherwise, even a solid that is already a topological hexahedron will be split into six tetrahedra, each creating four hexahedra.

Finally, we also added pyramids and prisms as base solids in order to simplify the presentation of the examples below. Before describing our approach in more detail, we introduce several definitions.

Definition 3. An *auxiliary edge* is an additional edge in the edge graph that connects two non-adjacent vertices on a face. Any two non-adjacent vertices of a face can be joined by an auxiliary edge.

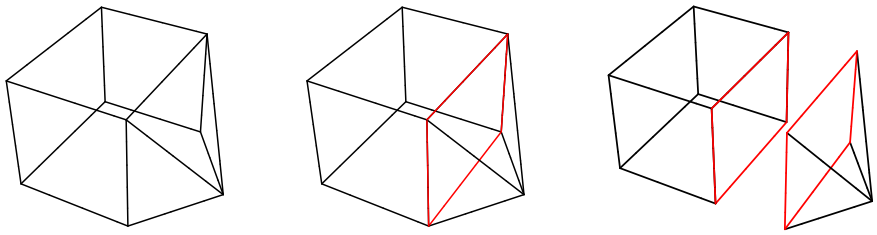

Figure 4: Left: The initial edge graph of a solid. Center: A valid cutting loop with one auxiliary edge and three existing edges (red edges). Right: The resulting two smaller edge graphs with the cutting surface.

A *cutting loop* is a simple closed loop of existing or auxiliary edges of the edge graph, such that no two edges of the loop belong to the same face.

A *cutting surface* is a multi-sided surface patch whose boundary curves are the existing or auxiliary edges of a cutting loop. It is a newly created surface patch inside the solid. We say that the cutting surface is *well-defined* if it can be used to split the given solid object into two smaller solids.

Due to the assumptions regarding the cutting loop, a well-defined cutting surface for each cutting loop exists. Indeed, the tangent planes of the cutting surface can be chosen such that the solid is subdivided into two sub-solids by the cutting surface. However, if two neighboring edges of the cutting loop shared a common face, then the tangent plane of the cutting surface at the corresponding common vertex would touch this face and the cutting surface would not be well-defined. This fact motivates the requirement concerning the edges in the definition of the cutting loop.

Clearly, there is always an infinite number of possible cutting surfaces for a given cutting loop. All these surface patches possess the same boundary curves.

As a first example, Figure 4 visualizes a simple edge graph of a solid with a valid cutting loop, the cutting surface and the two resulting smaller edge graphs.

In order to be able to formulate our recursive splitting algorithm, we will introduce another notion.

Definition 4. A cutting loop is said to be *valid* for the edge graph $\mathcal{G}$ of a given solid if any associated cutting surface decomposes the given solid into two smaller solids that again satisfy Assumption A3.

First we present a characterization of valid cutting loops.

Proposition 5. *A cutting loop is valid if and only if it contains at least three vertices and if all pairs of non-neighboring vertices do not share a face of the edge graph* $\mathcal{G}$*.*

PROOF. If the cutting loop is valid, then each of the two edge graphs of the solids, which are obtained by splitting the given solid using an associated cutting surface, is again 3-vertex-connected. In these two graphs, any two vertices $\mathbf{v}_i, \mathbf{v}_j$ of the cutting loop lie on the

face that corresponds to the cutting surface. According to Lemma 2, if the two vertices are not neighbors, then they must not share any other face of the edge graphs, since these two edge graphs are both 3-vertex-connected. Since all other faces of the edge graphs are (parts of) faces of the original edge graph $\mathcal{G}$ we conclude that all pairs of non-neighboring vertices of the cutting loop do not share a face of the original solid $\mathcal{G}$.

On the other hand, consider two non-neighboring vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ of a face of one of the two sub-solids that are obtained after splitting. We need to show that they do not share any other face, provided that all pairs of non-neighboring vertices of the cutting loop do not share a face. If one of the two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ does not belong to the cutting loop, then this is implied by the fact that the original solid satisfies Assumption A3. If both belong to the cutting loop, however, then this is exactly the condition that characterizes the cutting loop in the proposition. □

Based on this characterization for the validity of a cutting loop, Section 4 will discuss the existence of a valid cutting loop in more detail.

Now we are ready to formulate the *solid splitting algorithm* SplitSolid. The algorithm splits the edge graph of the solid into a collection of topological base solids, which correspond to hexahedra, tetrahedra, pyramids and prisms, see Figure 3. Each of these solids can be easily decomposed into a collection of topological hexahedra. More precisely, we obtain 4 hexahedra for a tetrahedron, n hexahedra for a prism with a n-sided polygonal base and $n-2$ tetrahedra (and hence $4n-8$ hexahedra) for a pyramid with a n-sided polygonal base.

Algorithm SplitSolid: Splitting the edge graph of the solid

```
1: procedure SplitSolid(graph G)
2:     if G is a base solid then
3:         return G and/or its subdivision into topological hexahedra
4:     else
5:         find the set L of all possible valid cutting loops
6:         ChooseCuttingLoop(L)
7:         decompose G into sub-graphs G1 and G2
8:         return SplitSolid(G1) and SplitSolid(G2)
9:     end if
10: end procedure
```

The selection of a valid cutting loop, performed by the function ChooseCuttingLoop, will be explained in Section 5.

Note that any edge graph with n vertices ($n \geq 4$), all possessing valency three, could be used as a base solid, since this polyhedron can be split into n hexahedra by midpoint subdivision. In the following, this segmentation is referred to as "simple" decomposition. In Section 6, we will show that our splitting algorithm often leads to a reduced number of topological hexahedra when compared to the "simple" decomposition (see Table 2).

The following Section 4 will also address the termination of this algorithm.

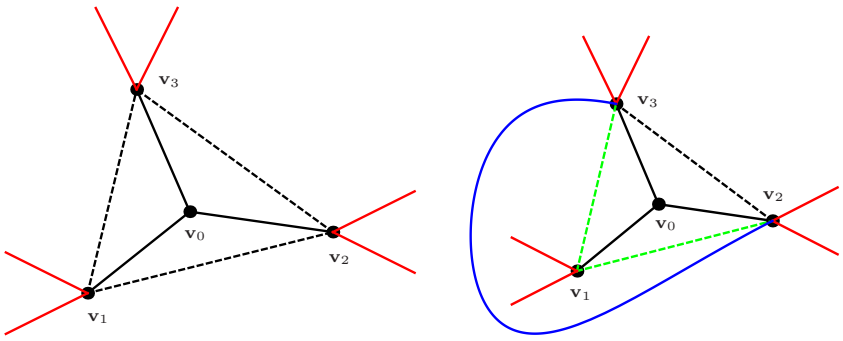

Figure 5: An extended 1-ring neighborhood of an arbitrary vertex $\mathbf{v}_0$ having valency three. For more detail, we refer to the first case in the proof of Theorem 6.

4. Existence of a valid cutting loop

The following statements guarantee that our splitting algorithm works for all solids that satisfy Assumption A3.

Theorem 6. If the edge graph $\mathcal{G}$ is not the edge graph of a topological tetrahedron (which is the complete graph K_4), then at least one valid cutting loop exists.

PROOF. We distinguish between two cases.

First case. We consider edge graphs where all vertices possess valency three. According to the Steinitz Theorem, these edge graphs can be obtained from a convex polyhedron with planar faces, where all vertices have valency 3. We may then pick one of the vertices and split the polyhedron into the tetrahedron formed by that vertex and its three neighbors and the remaining convex polyhedron. More precisely, we proceed as follows.

We pick an arbitrary vertex, denoted by $\mathbf{v}_0$, and connect the three neighboring vertices by a loop of existing and/or auxiliary edges, see Figure 5, left. The dashed lines represent either existing or auxiliary edges, and the red lines represent edges that may or may not exist.

First we observe that at least one of the red edges must exist, since the entire edge graph does not represent a tetrahedron. Now we conclude that at least one red edge for each of the three vertices $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ exists, since the graph is 3-connected. If for one of the three vertices a red edge would not exist, then deleting the other two vertices would split the graph into two disconnected sub-graphs.

Now we assume that two of these three edges belong to the same face. We consider the situation in Figure 5, right, and assume that the two green edges share a face. Consequently, we can join the vertices $\mathbf{v}_2$ and $\mathbf{v}_3$ by an auxiliary edge (shown in blue) across this face. Deleting $\mathbf{v}_1$ would then split the graph into two disconnected sub-graphs, thereby contradicting the assumption that the graph is 3-vertex connected.

Since the cutting loop consists of only 3 vertices, it is automatically valid, see Proposition 5.

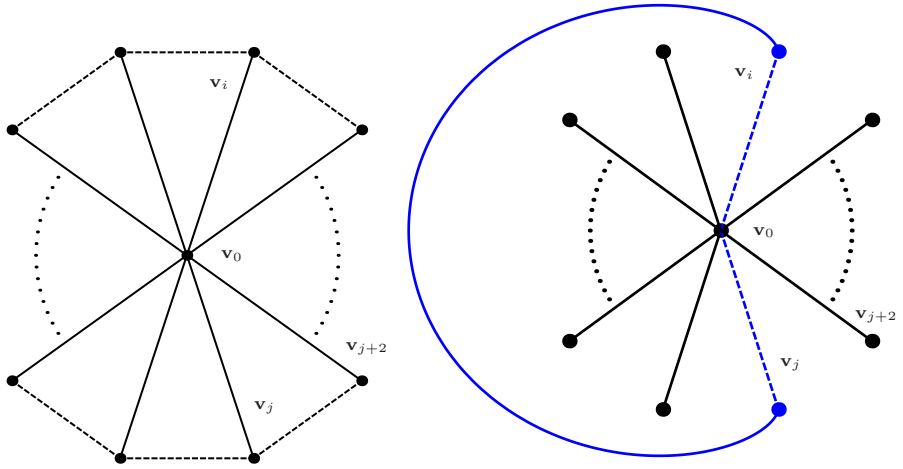

Figure 6: A part of the 1-ring neighborhood of a vertex $\mathbf{v}_0$ with a valency greater than three. For more detail, we refer to the second case in the proof of Theorem 6.

Second case. We consider an edge graph $\mathcal{G}$ where at least one vertex has a valency greater than three. We pick one of these vertices and denote it by $\mathbf{v}_0$. The n neighbors, where $n > 3$ is the valency of $\mathbf{v}_0$, can be connected by a closed loop of existing and/or auxiliary edges, see Figure 6, left. If all pairs of non-neighboring vertices of this loop do not share any face, then the same is true for the edges of the loop, and we found a cutting loop. Clearly, this loop is then also valid according to Proposition 5.

Otherwise, if two vertices, say $\mathbf{v}_i$ and $\mathbf{v}_j$ share a face, then we connect them by an existing or auxiliary edge of this face and consider the loop with the vertices $\mathbf{v}_i$, $\mathbf{v}_j$ and $\mathbf{v}_0$, see Figure 6, right. We show that this loop is then a cutting loop. It is then automatically also valid, since it consists of only three vertices, see again Proposition 5.

If the two dashed blue edge were on the same face, then we could draw a curve connecting $\mathbf{v}_0$ with itself that does not intersect any other edge but encircles some of the other vertices. This, however, contradicts the assumption that $\mathcal{G}$ is 3-vertex-connected, since removing $\mathbf{v}_0$ would split the graph into two disconnected components.

Also, if one dashed edge, say $(\mathbf{v}_0, \mathbf{v}_i)$, and the one blue non-dashed edge were on the same face, then we could create an auxiliary edge connecting vertex $\mathbf{v}_0$ and $\mathbf{v}_j$ across that face. The loop formed by the two existing and auxiliary edges between $\mathbf{v}_0$ and $\mathbf{v}_j$ then encircles some of the remaining vertices and deleting both $\mathbf{v}_0$ and $\mathbf{v}_j$ splits the graph into two disconnected components, thus violating the assumption that the graph is 3-vertex-connected.

Summing up, we can find at least one valid cutting loop in all cases. □

Corollary 7. *Any solid that is not topologically equivalent to a tetrahedron can be segmented into a collection of topological hexahedra using algorithm* SplitSolid.

PROOF. In each recursive step of the algorithm we will find at least one valid cutting loop. Moreover, the splitting step does not introduce additional vertices, and the number of

vertices in the two sub-solids is always less than in the original solids. Consequently, after finitely many steps we arrive at a collection of base solids, which can then be subdivided into topological hexahedra. □

In particular, if we allow only tetrahedra as base solids, then we are able to decompose the edge graph of the solid into a mesh of (topological) tetrahedra without adding new vertices. This would be similar to the tetrahedralization algorithm for convex polyhedra in [13].

Our main goal, however, is to obtain a subdivision with a *small* number of topological hexahedra. This will be achieved with the help of a suitable cost function.

5. Cost-based splitting algorithm

The procedure ChooseCuttingLoop provides a simple possibility to automatically select a valid cutting loop in the splitting algorithm. This is achieved by combining simple combinatorial and geometric criteria.

Algorithm ChooseCuttingLoop: Selection of the cutting loop

```
1: procedure ChooseCuttingLoop(set L of valid cutting loops)
2:     compute for each cutting loop λ the value ν with the help of the sequence ω
3:     choose a cutting loop λ_max that realizes the highest value ν
4:     return λ_max
5: end procedure
```

Let $\mathcal{L}$ be the set of all possible valid cutting loops for the given edge graph $\mathcal{G}$ of the solid. For each cutting loop $\lambda \in \mathcal{L}$, we compute a value, depending on the length and on the number of auxiliary edges of the cutting loop λ. In detail, we compute the value

$$\nu(\lambda) = \omega_n - m\ p,$$

where ω_n is a value that depends on the length n (i.e., the number of vertices) of the cutting loop, m is the number of auxiliary edges of the cutting loop and p is the cost of introducing an auxiliary edge.

As a simple extension, one might introduce additional geometry-related terms in this cost function. These terms could measure the deviation of the cutting surface from a plane, thereby encouraging planar cutting surfaces, and similar geometric criteria.

Based on the cost function we establish a ranking list of the valid cutting loops. We choose a cutting loop that realizes the highest value of ν. If two or more have the same highest value, then we select one loop randomly.

We call the sequence $\omega = (\omega_3, \ldots, \omega_s|p)$ for $s > 3$, which controls the cost function, the *decomposition sequence*. The number s specifies the maximum length of the cutting loops to be considered by the algorithm. The decomposition sequence is specified by the user in advance. Table 1 shows three different instances of possible decomposition sequences, which we used in our examples in Figure 7-9. In all examples, the length of the valid cutting loops was restricted to at most $s = 7$.

	ω_3	ω_4	ω_5	ω_6	ω_7	p
decomposition 1	50	100	100	100	100	0
decomposition 2	50	100	150	200	250	50
decomposition 3	50	100	50	80	30	50

Table 1: Instances of possible decomposition sequences $\omega = (\omega_3, \omega_4, \omega_5, \omega_6, \omega_7|p)$, which are used for decomposing the Solid 1, Solid 2 and Solid 3 in Figure 7, 8 and 9, respectively.

The selection of the valid cutting loop is performed by algorithm ChooseCutting-Loop.

As a straightforward modification of the algorithm, one might combine two or even more splitting steps and consider the total costs, in order to obtain a globally optimal splitting of a solid. However, the number of possible cuts grows exponentially with the number of cutting steps that are considered simultaneously.

6. Examples

We present several examples of possible volume segmentations for different geometries, see Figure 7-10. In detail, we choose the three solids from Figure 1 as starting point of the segmentation. Each of them is decomposed with the help of the cost-based splitting algorithm by using the three decomposition sequences ω given in Table 1.

Figure 1 depicts the initial geometries, the resulting edge graphs and the corresponding planar embeddings. The resulting decompositions of the edge graphs are shown in Figures 7-9. In each step of the splitting algorithm, a cutting loop was automatically selected (red loop), which consists of existing (dashed) and auxiliary edges (non-dashed).

Table 2 summarizes the resulting number of topological hexahedra for the different decompositions of the three solids, compared with the corresponding "simple" decompositions where possible. In all cases, the third decomposition sequence was the best choice for the segmentation in the sense of generating the minimal number of resulting topological hexahedra. This may be due to the fact that four-sided cutting surfaces were encouraged by this decomposition sequence. In Figure 10, we present the segmentation results for this decomposition sequence for the three solids.

	Solid 1	Solid 2	Solid 3
"simple" decomposition	16	16	n/a
decomposition 1	16	31	18
decomposition 2	16	22	10
decomposition 3	8	3	10

Table 2: Number of topological hexahedra for the resulting segmentations of the solids in Figure 7-9 for the three different decomposition sequences given in Table 1, compared with the corresponding "simple" decompositions if possible. For Solid 3 the "simple" decomposition is not available, since not all vertices of its edge graph have valency three.

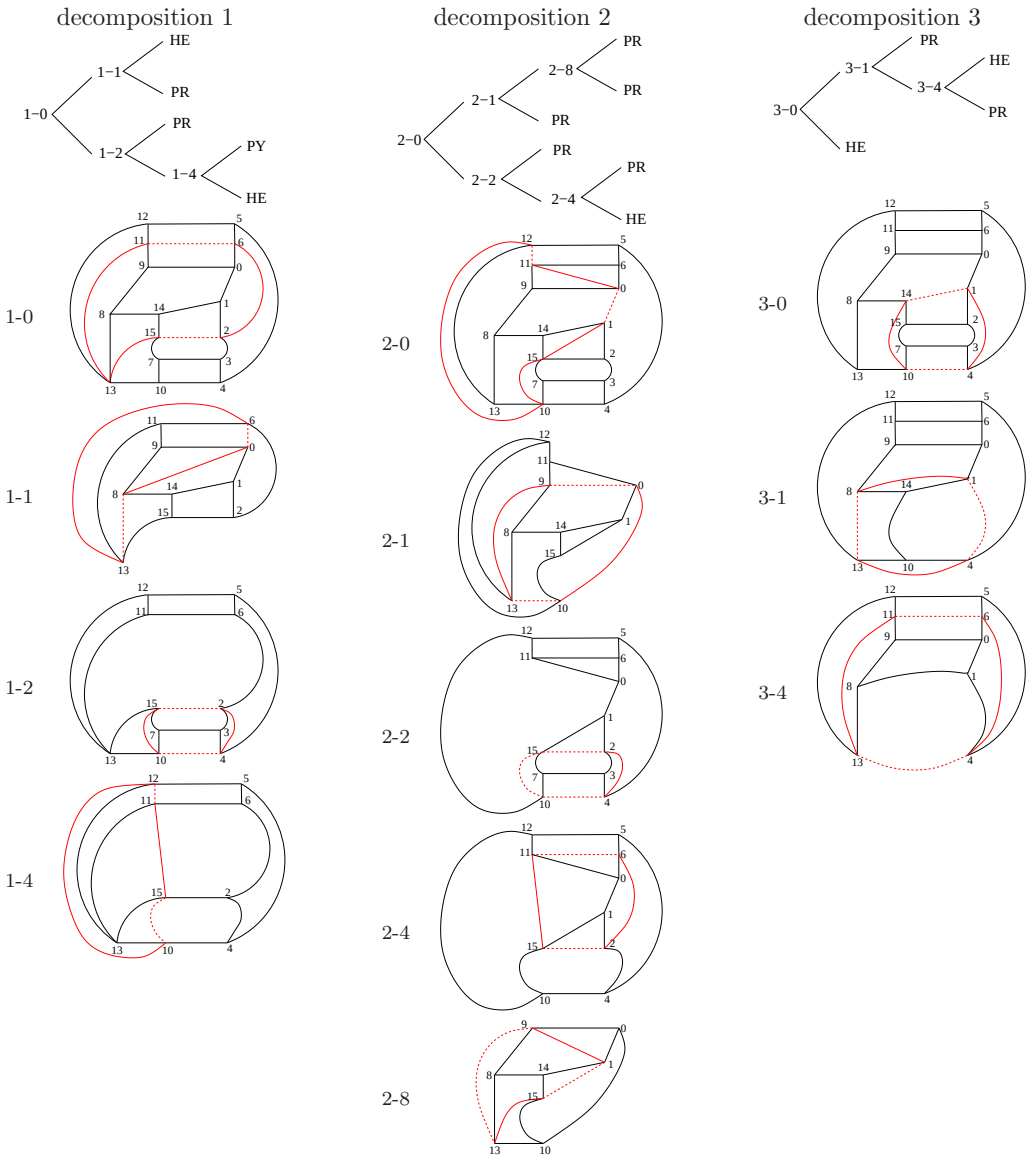

Figure 7: Three different decomposition results (i.e. the planar embeddings of the resulting edge graphs) for Solid 1, see Figure 1, by using the three different decomposition sequences ω given in Table 1. The red loops in each step are the selected cutting loops with the existing (dashed) and the auxiliary edges (non-dashed).

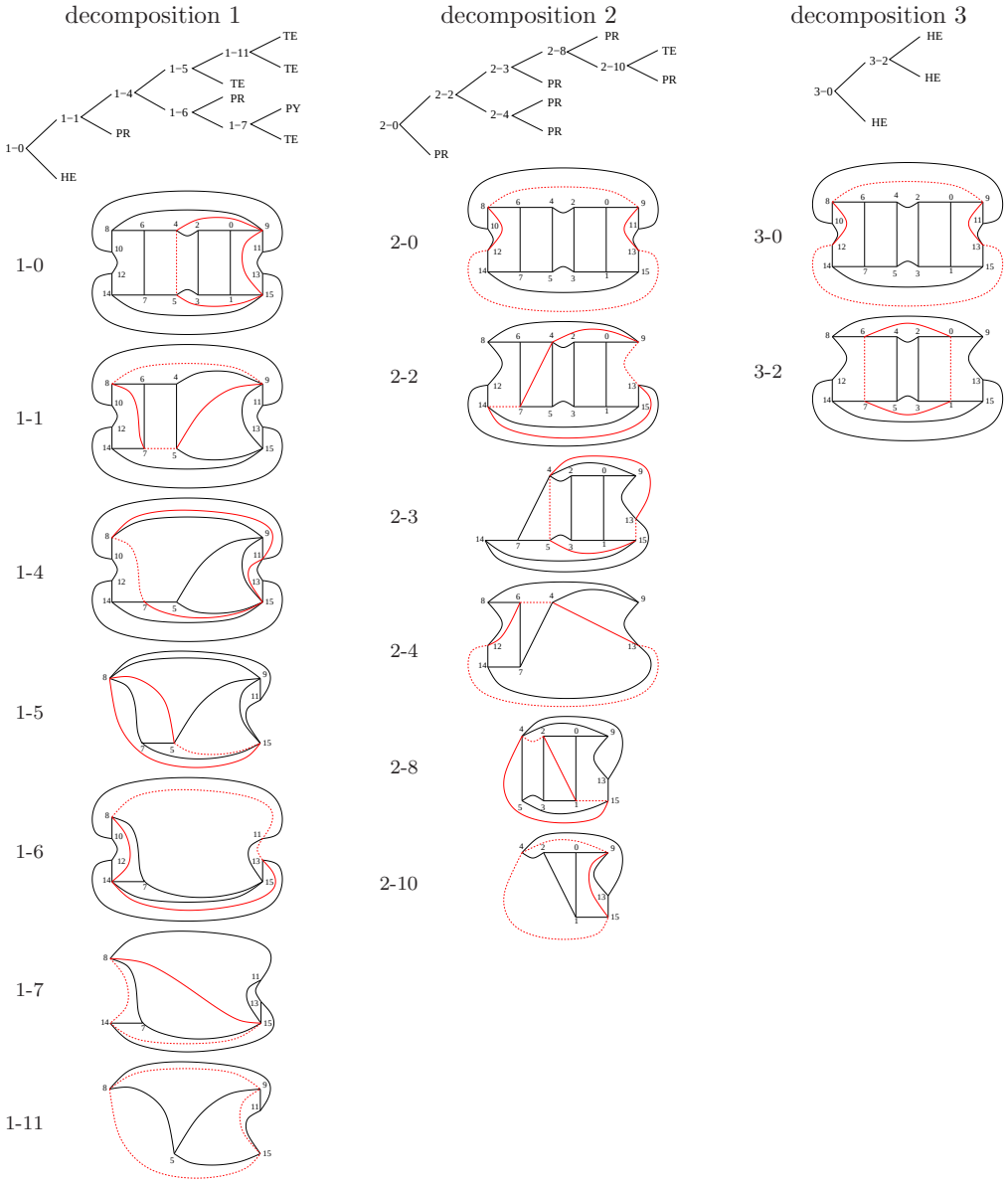

Figure 8: Three different decomposition results (i.e. the planar embeddings of the resulting edge graphs) for Solid 2, see Figure 1, by using the three different decomposition sequences ω given in Table 1. The red loops in each step are the selected cutting loops with the existing (dashed) and the auxiliary edges (non-dashed).

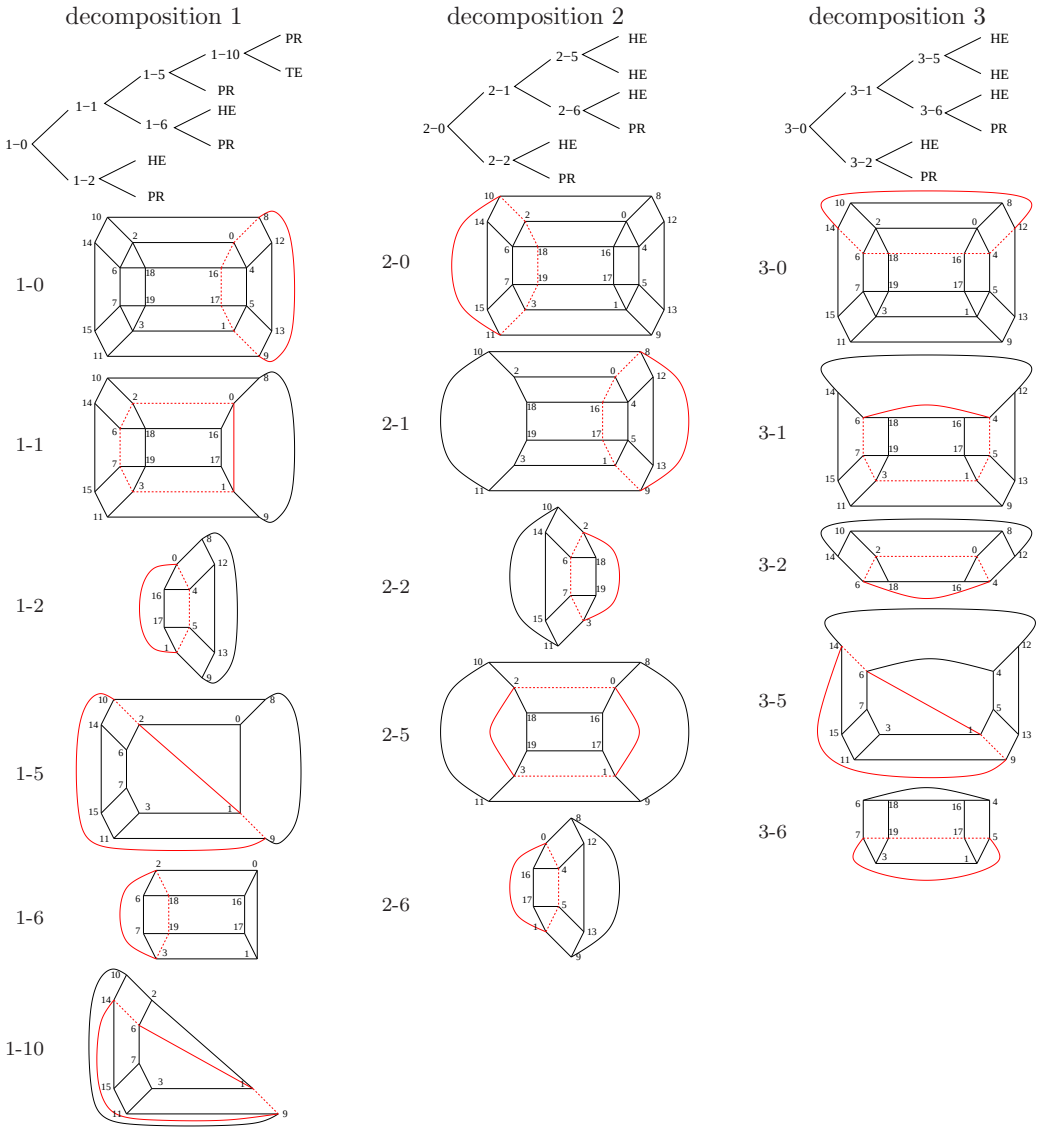

Figure 9: Three different decomposition results (i.e. the planar embeddings of the resulting edge graphs) for Solid 3, see Figure 1, by using the three different decomposition sequences ω given in Table 1. The red loops in each step are the selected cutting loops with the existing (dashed) and the auxiliary edges (non-dashed).

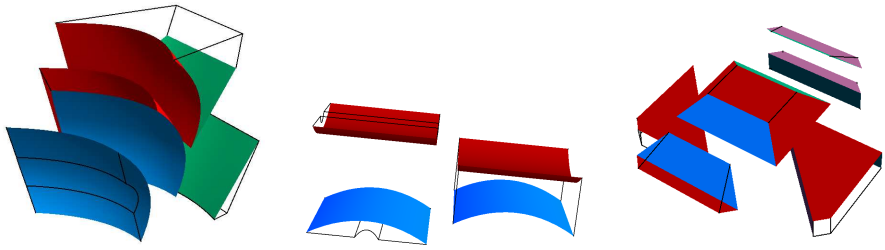

Figure 10: Segmentation results for the three solids from Figure 1 for the third decomposition sequence (decomposition 3) given in Table 1. The associated cutting surfaces for a model are drawn with the same color.

7. Conclusion

We presented an edge graph-based volume segmentation algorithm for solving the IGA segmentation problem. We repeatedly decompose a solid into a collection of predefined base solids and finally into a collection of topological hexahedra. By specifying a decomposition sequence ω in advance, the cutting loops in the single segmentation steps are selected automatically and provide a splitting tree of the edge graph with the base solids as leaves.

It is clear that our splitting algorithm does not always lead to the "best possible" segmentation result. But it is a first approach to automatize the single steps in the isogeometric segmentation process. Theoretically we could even generate all possible decompositions on the planar embedding level of the edge graph and choose the best one.

Since this paper is restricted to edge graphs of solids with only convex edges and with 3-vertex-connected edge graphs, we are currently working to generalize this approach for edge graphs of solids with also non-convex edges and with lower connectivity [32]. Additional topics for future research include the formulation of more advanced geometric criteria for the selection of a cutting loop. Moreover, it will be worthwhile to study automatic segmentation techniques that avoid T-joints in the final result. Last, but not least, techniques for suppressing features and for simplifying CAD models (cf. [36]; e.g., by removing blends between faces) will be of great help for solving the isogeometric segmentation problem in practice.

Acknowledgment. This research was supported by the European Union through the 7th Framework program, project "Towards Enhanced Integration of Design and Production in the Factory of the Future through Isogeometric Technologies" (TERRIFIC, grant agreement no. 284981). Qing Pan was also supported by National Natural Science Foundation of China (Grant #11171103).

References

[1] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (39-41) (2005) 4135–4195.

[2] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, Chichester, England, 2009.

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

[4] T. Martin, E. Cohen, Volumetric parameterization of complex objects by respecting multiple materials, Computers & Graphics 34 (3) (2010) 187 – 197.

[5] M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, A.-V. Vuong, Swept volume parameterization for isogeometric analysis, in: E. Hancock, R. Martin (Eds.), The Mathematics of Surfaces XIII, Vol. 5654 of Lecture Notes in Computer Science, Springer, 2009, pp. 19–44.

[6] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications, Comput. Aided Des. 45 (2) (2013) 395–404.

[7] T. Nguyen, B. Jüttler, Parameterization of contractible domains using sequences of harmonic maps, in: J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, L. Schumaker (Eds.), Curves and Surfaces, Vol. 6920 of Lecture Notes in Computer Science, Springer, 2012, pp. 501–514.

[8] Y. Zhang, W. Wang, T. Hughes, Conformal solid T-spline construction from boundary T-spline representations, Computational Mechanics 51 (6) (2013) 1051–1059.

[9] E. Cohen, T. Martin, R. M. Kirby, T. Lyche, R. F. Riesenfeld, Analysis-aware modeling: understanding quality considerations in modeling for isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 199 (5-8) (2010) 334–356.

[10] X. Li, X. Guo, H. Wang, Y. He, X. Gu, H. Qin, Harmonic volumetric mapping for solid modeling applications, in: Proc. ACM Symp. in Solid and Physical Modeling, ACM, New York, 2007, pp. 109–120.

[11] J. Xia, Y. He, S. Han, C.-W. Fu, F. Luo, X. Gu, Parameterization of star-shaped volumes using green's functions, in: B. Mourrain, S. Schaefer, G. Xu (Eds.), Advances in Geometric Modeling and Processing, Vol. 6130 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 219–235.

[12] J. Xia, Y. He, X. Yin, S. Han, X. Gu, Direct-product volumetric parameterization of handlebodies via harmonic fields, in: Shape Modeling International Conference (SMI), 2010, 2010, pp. 3–12.

[13] N. J. Lennes, Theorems on the Simple Finite Polygon and Polyhedron, Amer. J. Math. 33 (1911) 37–62.

[14] H. Tverberg, How to cut a convex polytope into simplices, Geometriae Dedicata 3 (1974) 239–240.

[15] C. W. Lee, Subdivisions and triangulations of polytopes, in: Handbook of discrete and computational geometry, CRC Press Ser. Discrete Math. Appl., CRC, Boca Raton, FL, 1997, pp. 271–290.

[16] M. M. Bayer, Barycentric subdivisions, Pacific J. Math. 135 (1) (1988) 1–16.

[17] B. Chazelle, Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm., SIAM J. Comput. 13 (1984) 488–507.

[18] B. Chazelle, L. Palios, Triangulating a nonconvex polytope, Discrete and Computational Geometry 5 (1) (1990) 505–526.

[19] C. L. Bajaj, T. K. Dey, Convex decomposition of polyhedra and robustness, SIAM J. Comput. 21 (2) (1992) 339–364.

[20] C. Chan, S. T. Tan, Volume decomposition of CAD models for rapid prototyping technology, Rapid Prototyping Journal 11 (4) (2005) 221–234.

[21] H. Sakurai, Volume decomposition and feature recognition: part 1 - polyhedral objects, Computer-Aided Design 27 (11) (1995) 833–843.

[22] H. Sakurai, P. Dave, Volume decomposition and feature recognition, part II: curved objects, Computer-Aided Design 28 (6-7) (1996) 519–537.

[23] Y. Lu, R. Gadh, T. J. Tautges, Feature based hex meshing methodology: feature recognition and volume decomposition, Computer-Aided Design 33 (3) (2001) 221–232.

[24] M. Nieser, U. Reitebuch, K. Polthier, CubeCover – parameterization of 3D volumes, Comput. Graph. Forum 30 (5) (2011) 1397–1406.

[25] T. J. Tautges, T. Blacker, S. A. Mitchell, The whisker weaving algorithm: a connectivity-based method for constructing all-hexahedral finite element meshes, Int. J. Numer. Meth. Engrg. 39 (19) (1996) 3327–3349.

[26] D. White, L. Mingwu, S. E. Benzley, G. D. Sjaardema, Automated hexahedral mesh generation by virtual decomposition, in: Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, 1995, pp. 165–176.

[27] B.-Y. Shih, H. Sakurai, Automated hexahedral mesh generation by swept volume decomposition and recomposition, in: 5th International Meshing Roundtable, 1996, pp. 273–280.

[28] S. Han, J. Xia, Y. He, Constructing hexahedral shell meshes via volumetric polycube maps, Computer-Aided Design 43 (10) (2011) 1222–1233.

[29] A. Sheffer, M. Etzion, M. Bercovier, Hexahedral mesh generation using the embedded Voronoi graph, in: In Proceedings of the 7th International Meshing Roundtable, 1999, pp. 347–364.

[30] Y. Zhang, C. Bajaj, Adaptive and quality quadrilateral/hexahedral meshing from volumetric data, in: Computer Methods in Applied Mechanics and Engineering, 2006.

[31] U. Langer, Discontinuous Galerkin domain decomposition methods in IGA, project 3 of NFN S117 "Geometry + Simulation", see www.gs.jku.at (since 2012).

[32] M. Nguyen, M. Pauley, B. Jüttler, Isogeometric segmentation. Part II: Solids with nonconvex edges, in preparation.

[33] J. Matoušek, J. Nešetřil, Invitation to discrete mathematics, 2nd Edition, Oxford University Press, Oxford, 2009.

[34] B. Grünbaum, Convex polytopes, With the cooperation of Victor Klee, M. A. Perles and G. C. Shephard. Pure and Applied Mathematics, Vol. 16, Interscience Publishers John Wiley & Sons, Inc., New York, 1967.

[35] M. Henk, J. Richter-Gebert, G. M. Ziegler, Basic properties of convex polytopes, in: Handbook of discrete and computational geometry, CRC Press Ser. Discrete Math. Appl., CRC, Boca Raton, FL, 1997, pp. 243–270.

[36] A. Thakur, A. G. Banerjee, S. K. Gupta, A survey of CAD model simplification techniques for physics-based simulation applications, Computer-Aided Design 41 (2) (2009) 65 – 80.