NFN - Nationales Forschungsnetzwerk

Geometry + Simulation

http://www.gs.jku.at

Isogeometric Segmentation: The case of contractible solids without non-convex edges

B. Jüttler, M. Kapl, Dang-Manh Nguyen, Qing Pan, M. Pauley

G+S Report No. 16

July 2014

Der Wissenschaftsfonds.

JOHANNES KEPLER UNIVERSITY LINZ

Isogeometric Segmentation: The case of contractible solids without non-convex edges

Bert Jüttler[a], Mario Kapl[a,*], Dang-Manh Nguyen[a], Qing Pan[a], Michael Pauley[a]

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

Abstract

We present a novel technique for segmenting a three-dimensional solid with a 3-vertex-connected edge graph 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 a certain class of predefined base solids. The splitting step of the algorithm is based on simple combinatorial and geometric criteria. The segmentation technique described in the paper is part of a process pipeline for solving the isogeometric segmentation problem that we outline in the 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 often relies 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), manh.dang_nguyen@jku.at (Dang-Manh Nguyen), panqing@lsec.cc.ac.cn (Qing Pan), michael.pauley@jku.at (Michael Pauley)

– 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.

– Xu 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 [9–12].

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. [17–19]. 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.

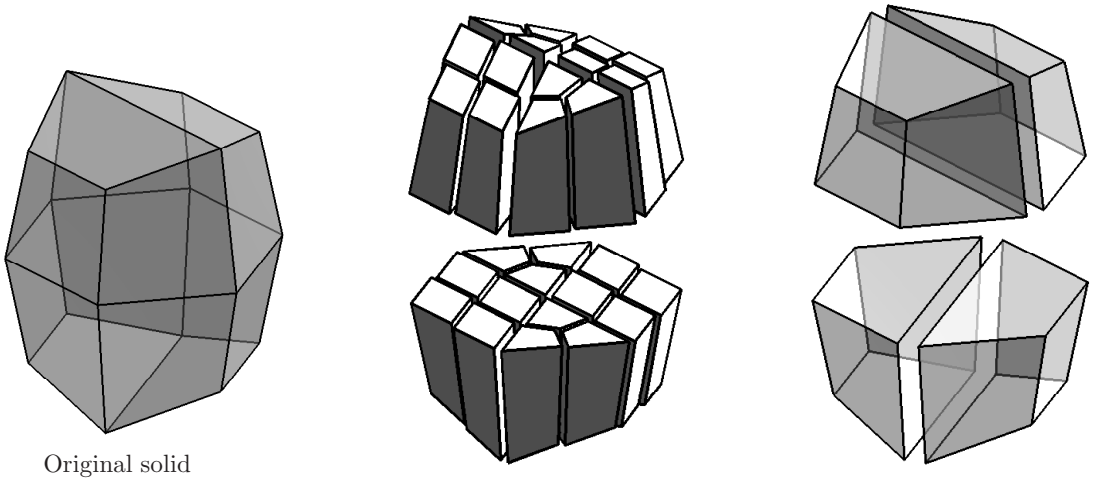
– Another approach to the segmentation of a CAD model is the representation as a hexahedral mesh with many hexahedra of approximately uniform size and shape. This is usually referred to as the problem of hex(ahedral) mesh generation. Due to the importance of hex meshes for numerical simulation, this 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 [24–30].

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 expect that a valid solution to the isogeometric segmentation problem can be obtained using a smaller number of topological hexahedra than the traditional hexahedral meshing. This is because



Original solid

Hexahedral meshing
HEs: **28**

Isogeometric segmentation
HEs: **4**

Figure 1: An example that helps to distinguish our isogeometric segmentation (IGS) problem from the traditional hex meshing (THM) problem. *Left*: An example solid with a 3-vertex-connected edge graph. *Middle*: A natural decomposition of the example solid as a solution to the THM problem; we believe that this is the coarsest decomposition that does not possess T-joints. *Right*: A segmentation of the example solid as a solution to our IGS problem; we note that the segmentation admits T-joints.

– the individual hexahedral elements do not need to meet face-to-face (cf. Figure 1), and

– the shape of NURBS volumes is much more flexible than that of traditional hexahedral elements. For instance, a ball can be represented exactly with only seven tri-quartic NURBS volumes, but its (approximate) representation as a hexahedral mesh may require a large number of elements, depending on the required geometric accuracy of the hexahedral mesh.

It is difficult to quantify the savings in the number of hexahedral elements, since this depends also on the required geometric accuracy of the hexahedral mesh. The reduction of the number of elements, however, is not the sole advantage of using isogeometric segmentation and isogeometric analysis. The resulting collection of trivariate NURBS volumes can be used to obtain a traditional NURBS-based CAD model. (This CAD model could be constructed from those boundary surfaces of the hexahedral NURBS volumes which lie on the boundary of the solid.) This results in a CAD model and a computational model which are consistent with each other. This advantage is at least as important as the reduction in the number of hexahedra, since it supports a higher level of interoperability of the various design and analysis tools, in particular in connection with shape optimization.

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 is discussed in a follow-up paper [32]. Future work will also address decompositions that allow a more direct coupling of the various subdomains. More precisely, the algorithm presented in the paper is part of a process pipeline for solving the isogeometric segmentation problem stated above. An outline of the entire pipeline will be given in the final section of the paper.

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 the possibility to automatically guide the segmentation steps based on simple combinatorial and geometric criteria. Section 6 presents decomposition results for different example solids. An outline of the entire process pipeline for solving the isogeometric segmentation problem is given in Section 7. Finally, we conclude the paper.

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 $\mathbf{n}_1, \mathbf{n}_2$ of these faces. Let $\mathbf{t}_1, \mathbf{t}_2$ be the two tangent vectors of the planar curve segments (oriented such that they point away from the edge). If $\mathbf{n}_1, \mathbf{t}_1$ do not separate $\mathbf{n}_2, \mathbf{t}_2$, 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 future work):

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 2 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 3, 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.

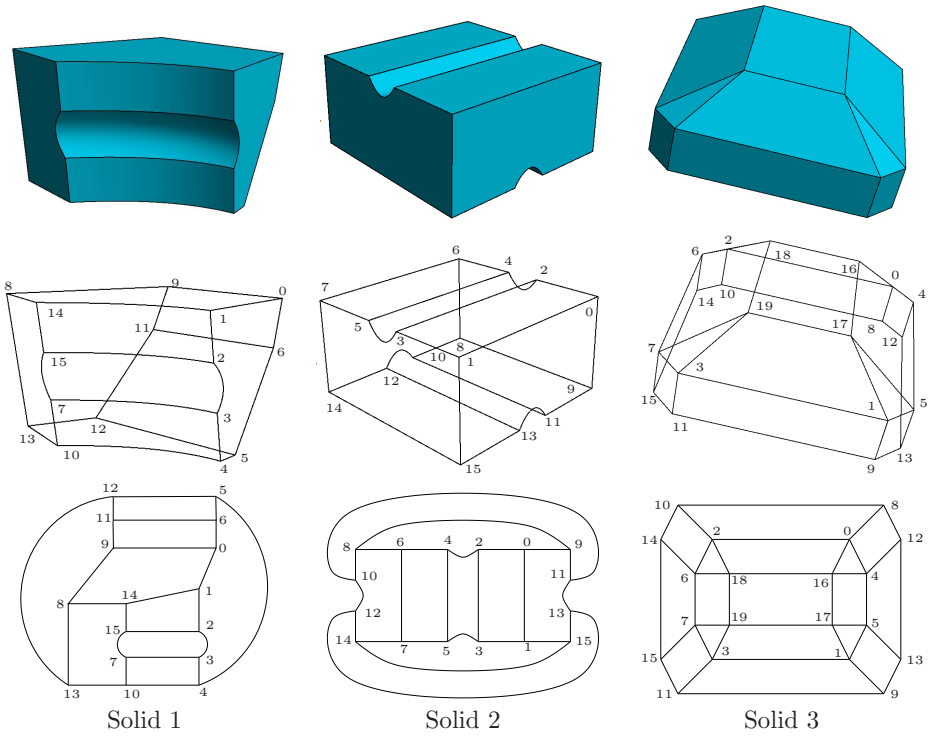


Figure 2: First row: Three solids with sharp edges. Second row: The associated edge graphs with only convex edges. Third row: The corresponding planar embeddings.

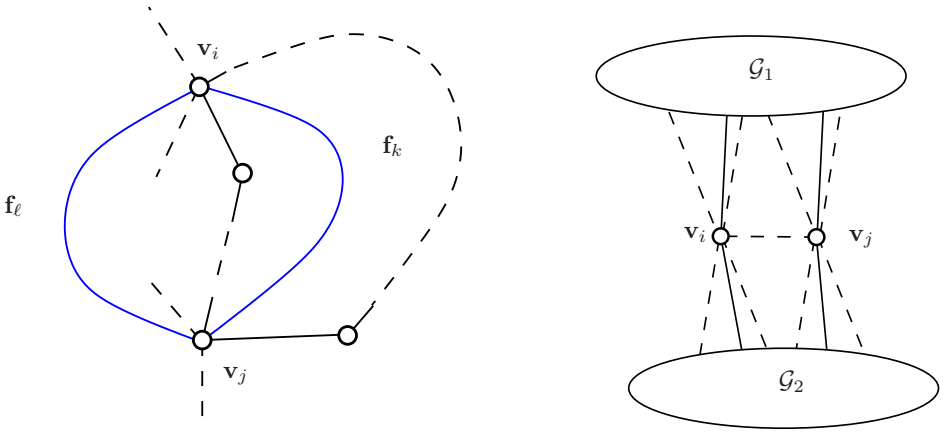


Figure 3: Left: An edge graph that possesses a face with two non-neighboring vertices sharing two faces is not 3-vertex connected. Right: A non-3-vertex-connected graph possess a face with two non-neighboring vertices sharing more than one face.

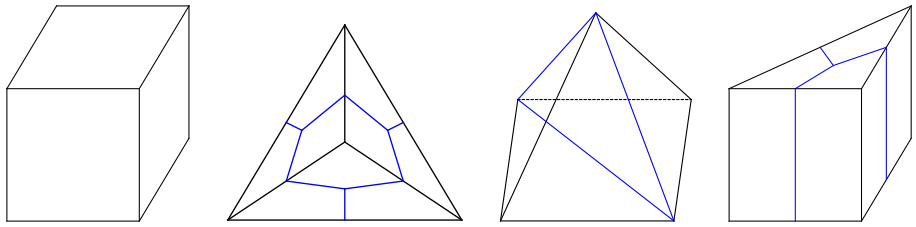


Figure 4: Four types of base solids. Each base solid can be represented by a mesh of topological hexahedra.

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 vertex whose deletion splits the graph would appear in the same face twice. Consider the situation in Figure 3, 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 4. 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, resulting in 24 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

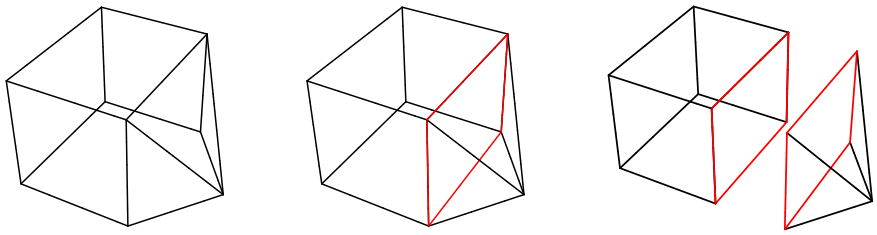


Figure 5: Left: The initial edge graph of a solid. Center: A valid cutting loop with one auxiliary edge and three existing edges (red edges). Right: The resulting two smaller edge graphs with the cutting surface.

non-adjacent vertices on a face. Any two non-adjacent vertices of a face can be joined by an auxiliary edge.

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 5 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 4. 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.[1]

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

[1] An alternative way to cut a prism with n sided base, $n \geq 5$, results in $\lceil n/2 \rceil - 1$ hexahedra, possibly by cutting at midpoints of edges. A second alternative is to continue using SplitSolid until all prisms are reduced to triangular prisms and hexahedra, and this is what we do in the examples in Section 6.

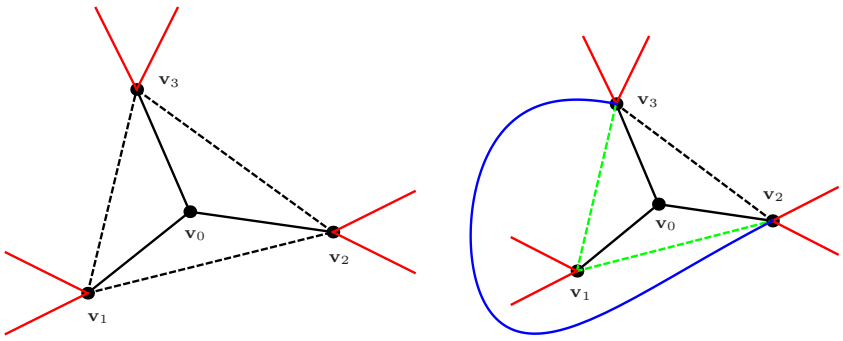


Figure 6: An extended 1-ring neighborhood of an arbitrary vertex $\mathbf{v}_0$ having valency three. For more detail, we refer to the first case in the proof of Theorem 6.

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.

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 6, 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 6, 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

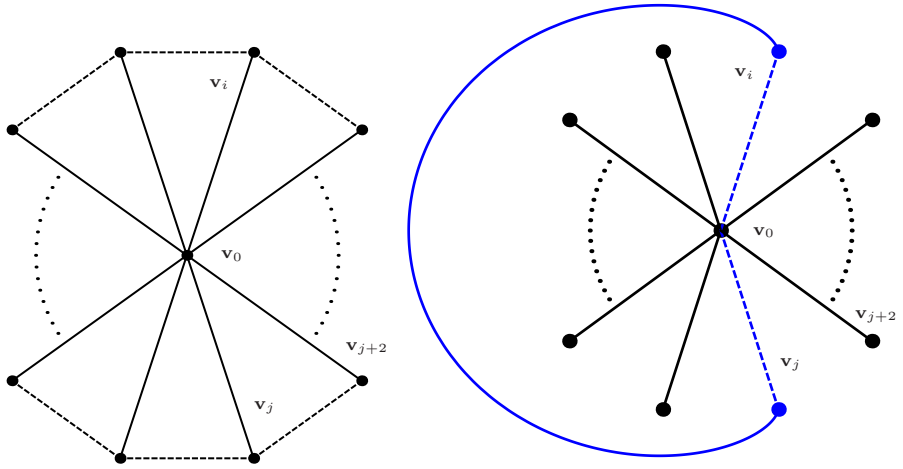


Figure 7: A part of the 1-ring neighborhood of a vertex $\mathbf{v}_0$ with a valency greater than three. For more detail, we refer to the second case in the proof of Theorem 6.

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.

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 7, 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 7, 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.

	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 8, 9 and 10, respectively.

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 8-10. In all examples, the length of the valid cutting loops was restricted to at most $s = 7$.

The selection of the valid cutting loop is performed by algorithm ChooseCuttingLoop. It is useful to run SplitSolid multiple times with different random number seeds, and choose the best outcome. For the examples in Section 6, after several runs, the segmentations with the smallest number of hexahedra were examined and one was selected manually after visually inspecting its suitability for generating NURBS volume parameterizations. For the future we plan to automate this selection process based on parameterization quality measures. Moreover, it will be useful to incorporate an estimate of the quality of the resulting shapes into the value of a cutting loop and this is a subject of our ongoing research.

We briefly discuss an upper bound on the complexity of ChooseCuttingLoop. The loops containing a given edge can be enumerated in order of increasing length using Yen's algorithm [36] in time $\mathcal{O}(\ell^3)$ per loop, where ℓ is the number of vertices. The time required to check that a loop is valid and compute its value is small compared to ℓ^3. Therefore, if K is the number of loops to be examined, the complexity is $\mathcal{O}(\ell^3 K)$. To find an upper bound for K, suppose that there are m edges, the maximum valency of any vertex is $v_{\max}$, the maximum number of vertices on any face is $d_{\max}$ and the maximum length of loops considered is s. Then, beginning with an edge and constructing a loop by adding one new (existing or auxiliary) edge at a time, we see that $K = \mathcal{O}\left(m\left(v_{\max} d_{\max}\right)^{s-1}\right)$. This estimate is very pessimistic, and does not consider that a loop must close up. Furthermore, depending on the choice of the decomposition sequence ω, the enumeration of loops may sometimes be stopped early when it can be guaranteed that no longer loops can have higher scores than the current best.

We can also look at the number of (recursive) calls to SplitSolid in terms of s and ℓ. If $\ell = 4$ then the solid is a tetrahedron and only one call to the function is required; suppose now $\ell > 4$. Suppose a cutting loop of length i is used, and there are j vertices on one side of the loop and k vertices on the other side. Then $i + j + k = \ell$, and the two resulting solids after the split have $i + j$ and $i + k$ vertices respectively. Maximizing over i, j, k defines a recursive function which provides an upper bound on the number of calls. This function can be computed exactly; asymptotically it is $\mathcal{O}(2^s \ell)$.

The solids appearing as inputs to SplitSolid at recursive depth D must have at most $\ell - D$ vertices; therefore the maximum recursion depth is $\ell - 4$. The maximum vertex degree $v_{\max,D}$ at depth D satisfies $v_{\max,D} \leq \ell - D - 1$. It also increases by at most 1 per step, so $v_{\max,D} \leq v_{\max} + D$; combining these inequalities gives $v_{\max,D} \leq \frac{1}{2}(v_{\max} + \ell - 1)$. The maximum number of vertices on a face $d_{\max,D}$ is bounded above by $\max(d_{\max}, s)$. Putting together all of the above gives a pessimistic estimate for the total complexity for SplitSolid of $\mathcal{O}\left(\ell^4 m \left((v_{\max} + \ell - 1)\max(d_{\max}, s)\right)^{s-1}\right)$. Note $m \leq \frac{1}{2}\ell(\ell-1), v_{\max} < \ell, d_{\max} < \ell$. Therefore, treating s as a constant, the complexity of SplitSolid is polynomial in ℓ.

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. First, we apply our approach to the three solids shown in Figure 2 to study the effect of different choices for the decomposition sequence. Further examples are then investigated to observe the geometric aspects of the segmentation and the consequences of different preprocessing strategies.

Solids 1–3. (See Figure 8-11). As first examples, we choose the three solids from Figure 2 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 2 depicts the initial geometries, the resulting edge graphs and the corresponding planar embeddings. The resulting decompositions of the edge graphs are shown in Figures 8-10. 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 11, we present the segmentation results for this decomposition sequence for the three solids.

Chair stand. Now we apply our method to the chair stand shown in Figure 12. Our experience from the previous examples led us to conclude that Decomposition Sequence 3 (See Table 1) is a good choice for keeping the number of hexahedra in the final segmentation small. Therefore we use this sequence for the following examples. The chair stand consists of five identical pieces, and for simulation purposes it is preferable that the segmentation algorithm is applied to a single piece, so that the result

	Solid 1	Solid 2	Solid 3
"simple" decomposition	16	16	n/a
decomposition 1	16	31	15
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 8-10 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.

respects the symmetry of the design. The piece has two holes in it, and our method applies only to simply connected geometries, therefore some manual preprocessing is required. We show the effects of our method in combination with several preprocessing strategies. Firstly, we try just adding internal faces as shown in Figure 12. We refer to this as the 1-cut strategy. The result is shown in Figure 13; the solid is segmented into five hexahedra but the shapes are not of a good quality, being difficult to parameterize. Our second preprocessing strategy, the 2-cut strategy, cuts the solid into smaller pieces by making two cuts for each hole. The preprocessed solid and the results are shown in Figure 14. The 2-cut strategy results in a segmentation into 7 hexahedra, with higher quality. Even higher quality is achieved using four cuts per hole (4-cut strategy, Figure 15) which results in $(2 \times 5) + (4 \times 7) + (2 \times 1) = 40$ topological hexahedra but of a higher quality. (In this example, topological pentagonal prisms arise, each of which are cut into a hexahedron and a triangular prism. A smaller number of hexahedra could be achieved by inserting vertices.) Finally we try dividing the piece along its two reflection symmetries (Figure 16) which results in 32 topological hexahedra but the quality of the shapes is not as high as for the 4-cut strategy.

Mechanical part of Zhang and Bajaj. Consider a mechanical part depicted in Figure 17a. The solid is similar to that considered in Zhang and Bajaj [30]. In a first pre-processing step, we manually cut the two cylindrical components off from the solid. As a circular face of each cylindrical component can be subdivided into five topological quads, each of the cylindrical component can be segmented into five topological hexahedra. The remaining component is a holey cube of genus 3 which can be obtained from a cube by cutting away the intersection of the cube with two orthogonal cylinders with the same radius, see Figure 17b. This component consists of eight congruent pieces. As mentioned before, it suffices to only segment one piece. Our approach applied to one piece results in the decomposition depicted in Figure 18a. We note that out of all resulting base solids, the (only) topological hexahedron has a vertex V (see Figure 18a) at which two adjacent faces meet at a zero angle. The study for avoiding such a situation will be carried out in the follow-up paper [32]. Nevertheless, by utilizing the yet another mirror symmetry of one piece, we obtain the decomposition presented in Figure 18b in which the problem related to zero angles

disappears.

Mechanical part of Yamakawa and Shimada. Another mechanical part is shown in Figure 19. This is based on an example from [37]; we have preprocessed it to cut a non-convex edge. The holes can be dealt with using a 1-cut, 2-cut or 4-cut strategy. (Figure 19 shows the part preprocessed according to the 1-cut strategy.) Whichever choice is made, the top piece ends up being a union of topological prisms, so we only consider the base. The result for the 1-cut strategy is shown in Figure 20. Although there are just one triangular prism and two hexahedra, this segmentation is not ideal as it would be difficult to produce volume parameterizations. Figure 21 shows the result for a 2-cut strategy: one of the two (identical) pieces is segmented into two prisms and a hexahedron. Thus the base of the mechanical part is segmented into a total of 14 hexahedra. (Since a topological pentagonal prism arises, by allowing the addition of new vertices, the solid could be segmented into fewer hexahedra). Figure 22 shows the result for the 4-cut strategy. One of the two nontrivial pieces is segmented into two prisms and a hexahedron. Thus the base of the mechanical part is segmented into 18 hexahedra. The results in Figures 20, 21, 22 support our claim that the quality of the final shapes depends on the total turning angle of the curves that are input into our algorithm.

In preprocessing some solids with holes, we examined a 1-cut, 2-cut and 4-cut strategy, and found that the 4-cut strategy leads to base solids with better shape, even if the total number of topological hexahedra is higher. We believe that in the interest of high quality shapes, it is useful to ensure that there are restrictions on the total variation of the surface normal of any surface and the total turning angle of any edge, as well as the maximum ratio of the lengths of any two edges. It is also useful to incorporate geometric criteria into the value of a cutting loop, and work in this direction is ongoing.

7. Towards an isogeometric segmentation pipeline

The work presented in this paper is the first step towards establishing a process pipeline for solving the isogeometric segmentation problem. We use this section to describe the planned workflow of the entire process. In order to illustrate the process we shall use the TERRIFIC part (data courtesy of our project partner Dr. S. Boschert / Siemens) which is one of the benchmark examples in the European project that provides some of the funding for this research, see Figure 23(**a**). The process is divided into the following steps, illustrated in Figure 23.

1. CAD model simplification and preparation. Most existing CAD models cannot be used directly, since they contain too many details or the geometric description of the boundary is too complicated. For instance, single faces of the boundary may be described by several (possibly trimmed) surface patches, and this would cause problems for the isogeometric segmentation. Consequently, it is necessary to establish a clean representation of the geometry before entering the remaining steps of the isogeometric segmentation pipeline.

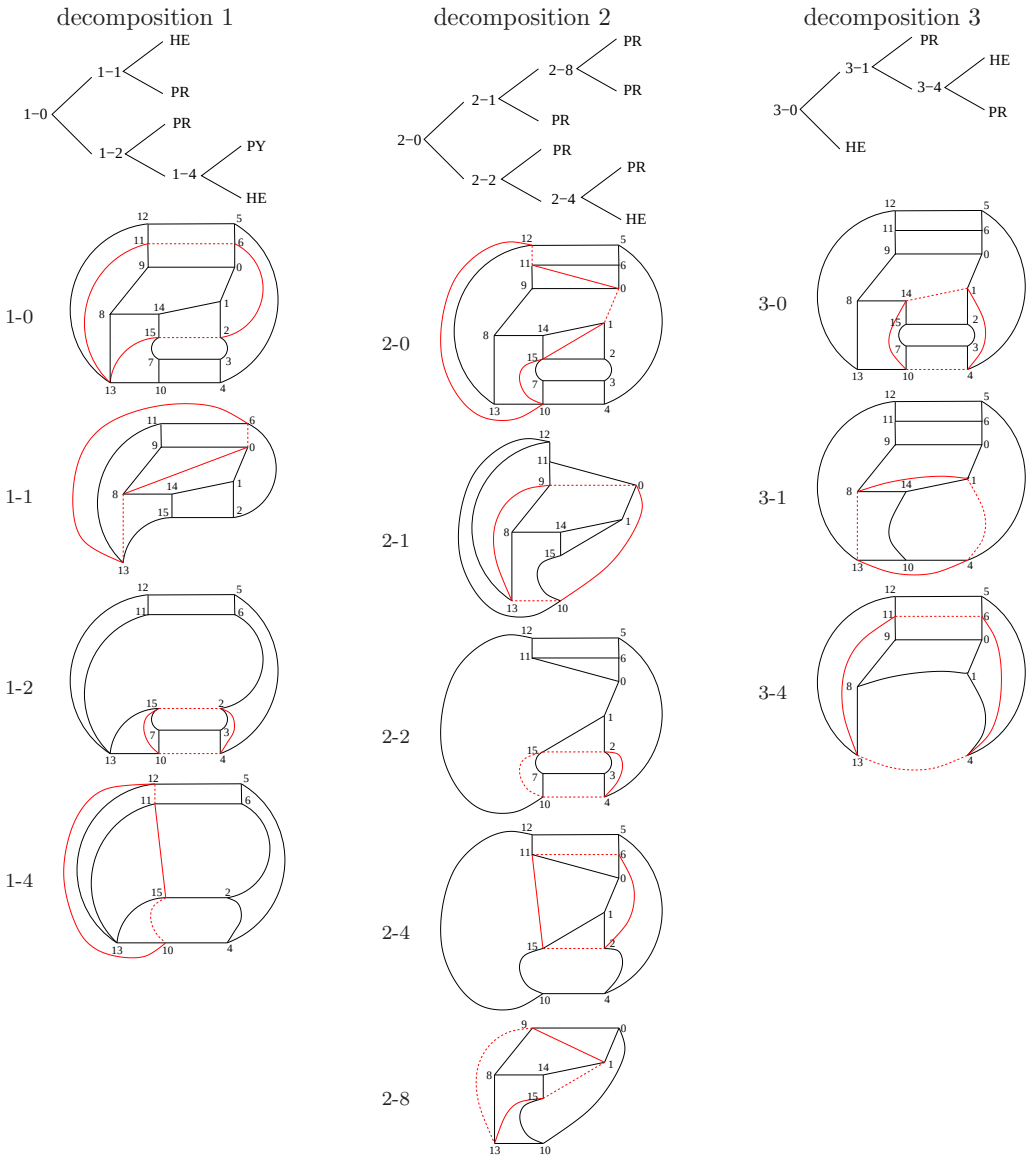


Figure 8: Three different decomposition results (i.e. the planar embeddings of the resulting edge graphs) for Solid 1, see Figure 2, by using the three different decomposition sequences ω given in Table 1. The red loops in each step are the selected cutting loops with the existing (dashed) and the auxiliary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

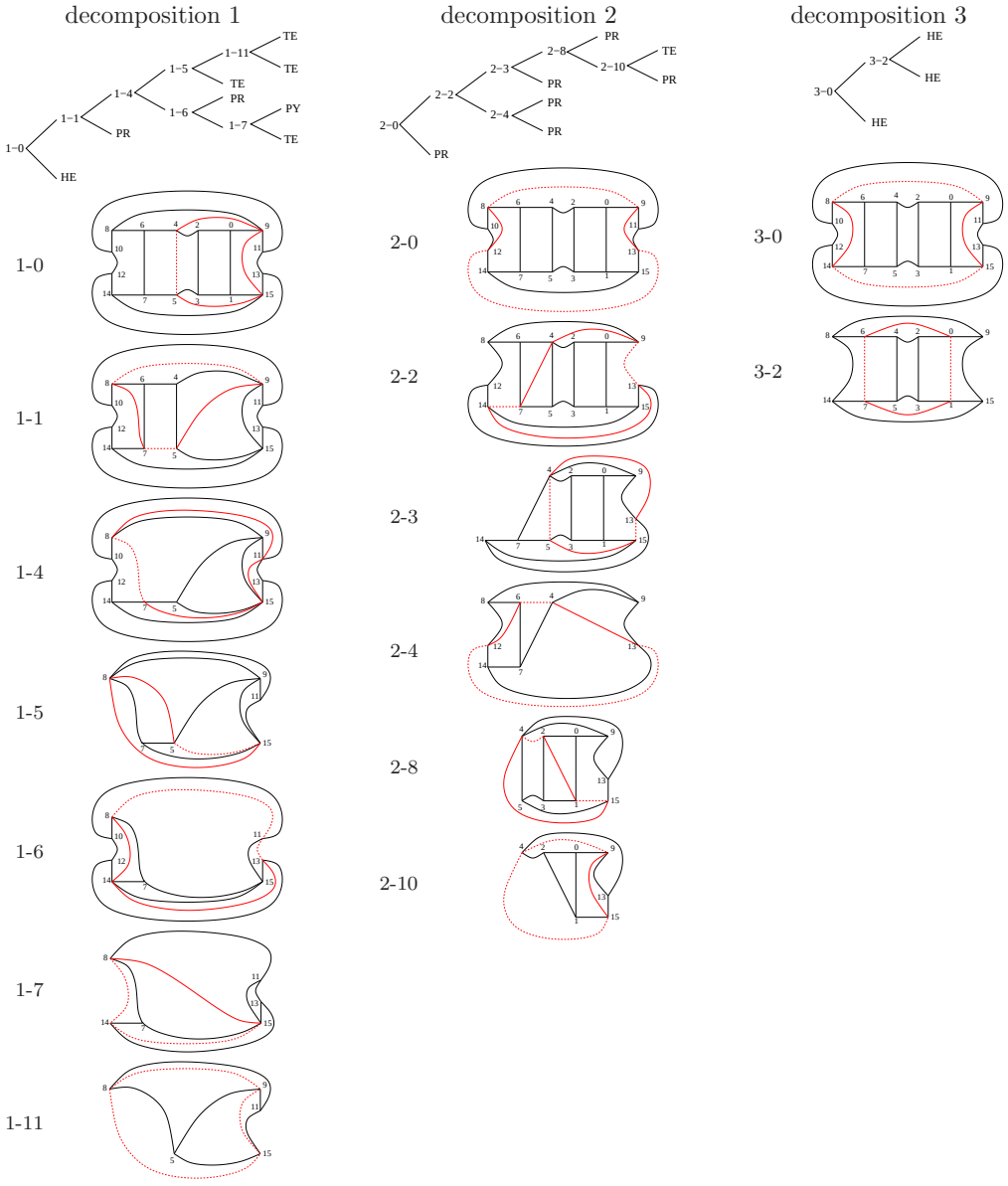


Figure 9: Three different decomposition results (i.e. the planar embeddings of the resulting edge graphs) for Solid 2, see Figure 2, by using the three different decomposition sequences ω given in Table 1. The red loops in each step are the selected cutting loops with the existing (dashed) and the auxiliary edges (non-dashed).

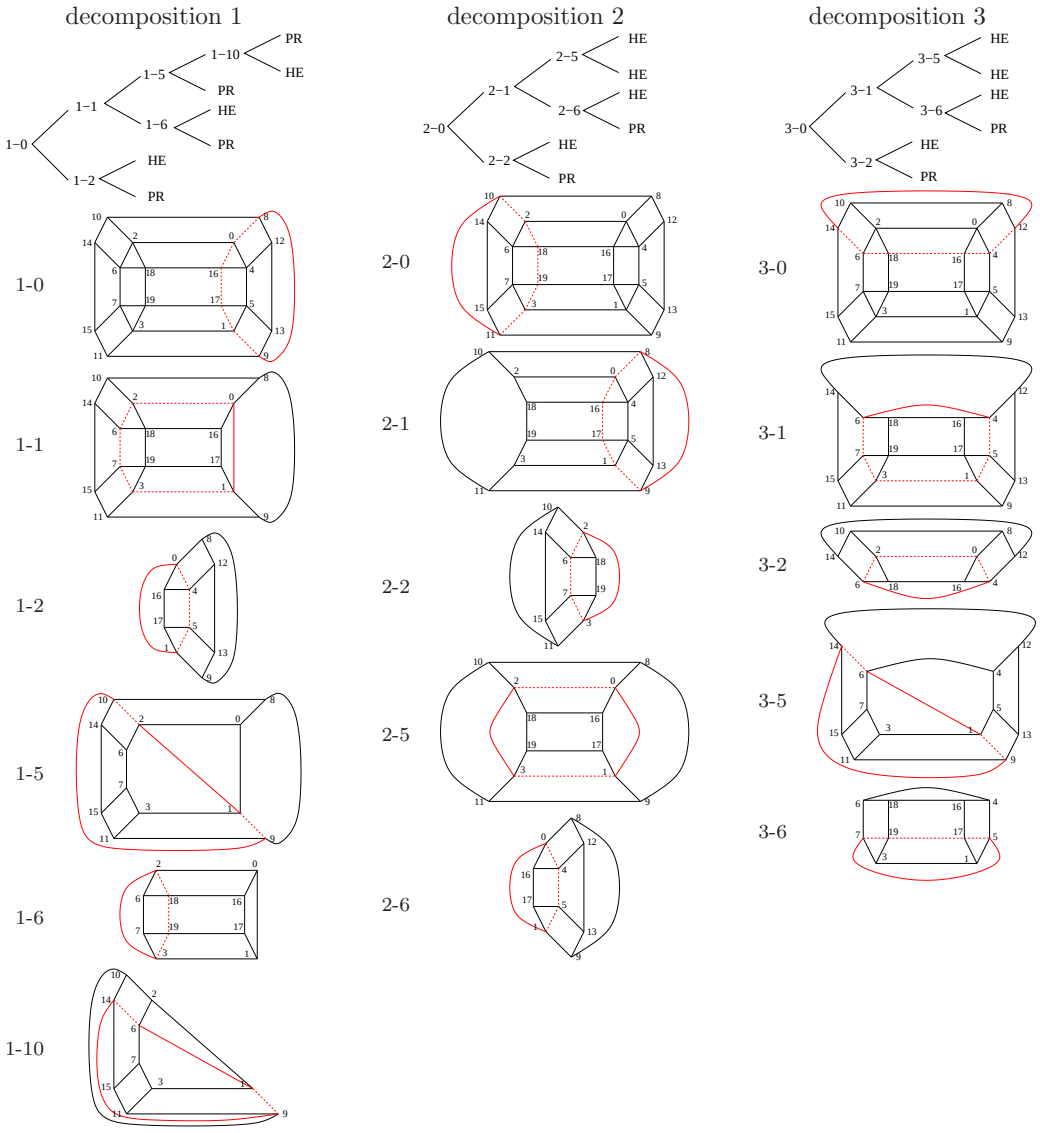


Figure 10: Three different decomposition results (i.e. the planar embeddings of the resulting edge graphs) for Solid 3, see Figure 2, by using the three different decomposition sequences ω given in Table 1. The red loops in each step are the selected cutting loops with the existing (dashed) and the auxiliary edges (non-dashed).

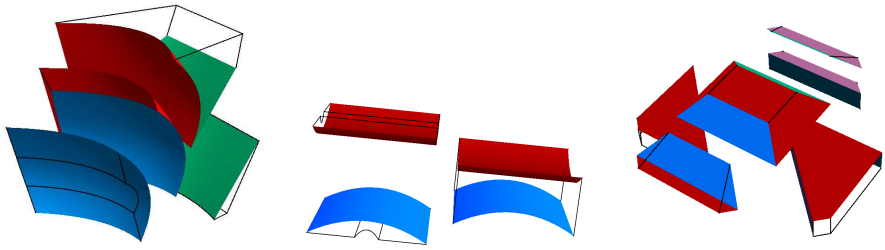


Figure 11: Segmentation results for the three solids from Figure 2 for the third decomposition sequence (decomposition 3) given in Table 1. The associated cutting surfaces for a model are drawn with the same color.

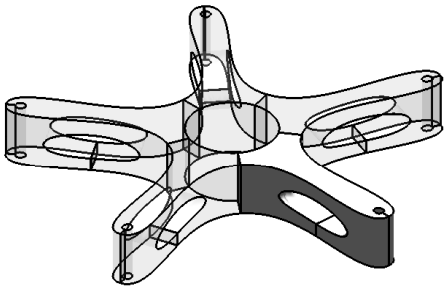


Figure 12: A chair stand, defined by a boundary representation. The stand is comprised of five identical pieces. One fifth of the chair stand is extracted for isogeometric segmentation. The piece has been preprocessed by adding interior faces to make it simply connected.

Our approach relies entirely on *triangulated CAD models*, as this data format provides a very versatile representation of shapes. After a manual simplification, we triangulate the CAD model and use the triangulation to identify the faces of the CAD model by an automatic segmentation procedure. Each of the faces is then approximated by a single trimmed NURBS patch using well-established methods for geometry reconstruction. We obtain a representation of the object's boundary as a collection of trimmed NURBS surface, where each face is described by exactly one surface.

2. Adding auxiliary edges to obtain contractible solids with 3-vertex-connectivity. After creating and analyzing the edge graph, we cut the solid by auxiliary cutting surfaces in order to obtain contractible solids. Moreover, we add auxiliary edges (which have to be considered as non-convex edges) in order to make the edge graph 3-vertex-connected. This step is not yet automated. Some experimental results for different strategies to choose the auxiliary cutting surfaces have been described in the previous section.

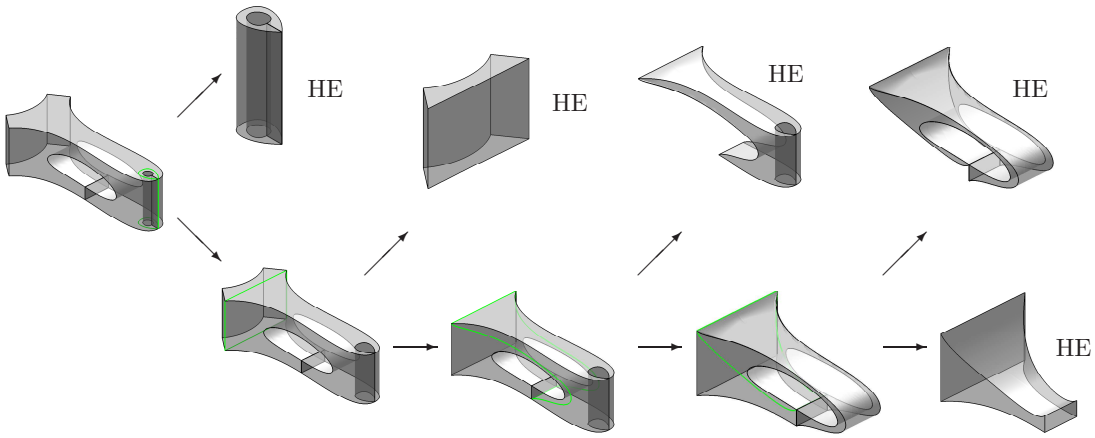


Figure 13: A segmentation of the stand piece, beginning with the solid preprocessed by the 1-cut strategy (Figure 12). In this case, the solid is segmented into 5 topological hexahedra.

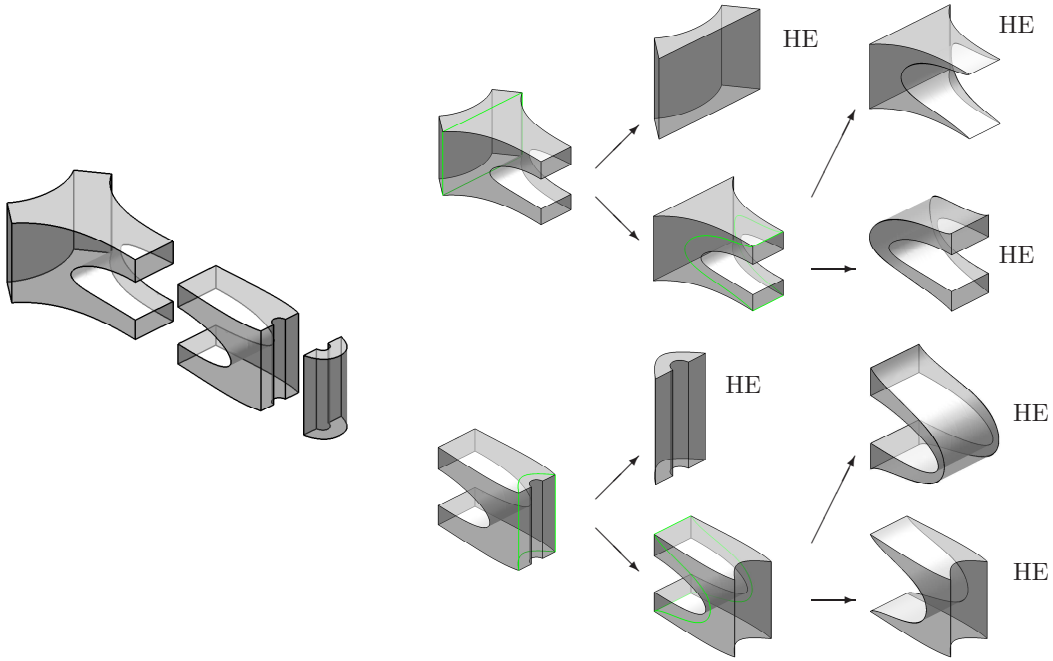


Figure 14: Left: manual preprocessing of the stand piece using the 2-cut strategy. Right: segmentation of the two non-hexahedral pieces.

3. Dealing with non-convex edges. In the next step we eliminate the non-convex edges by cutting the solid using suitable cutting surfaces. This step is discussed in more detail in a follow-up paper [32].
4. Subdividing a contractible solid with only convex edges. At this stage, the original

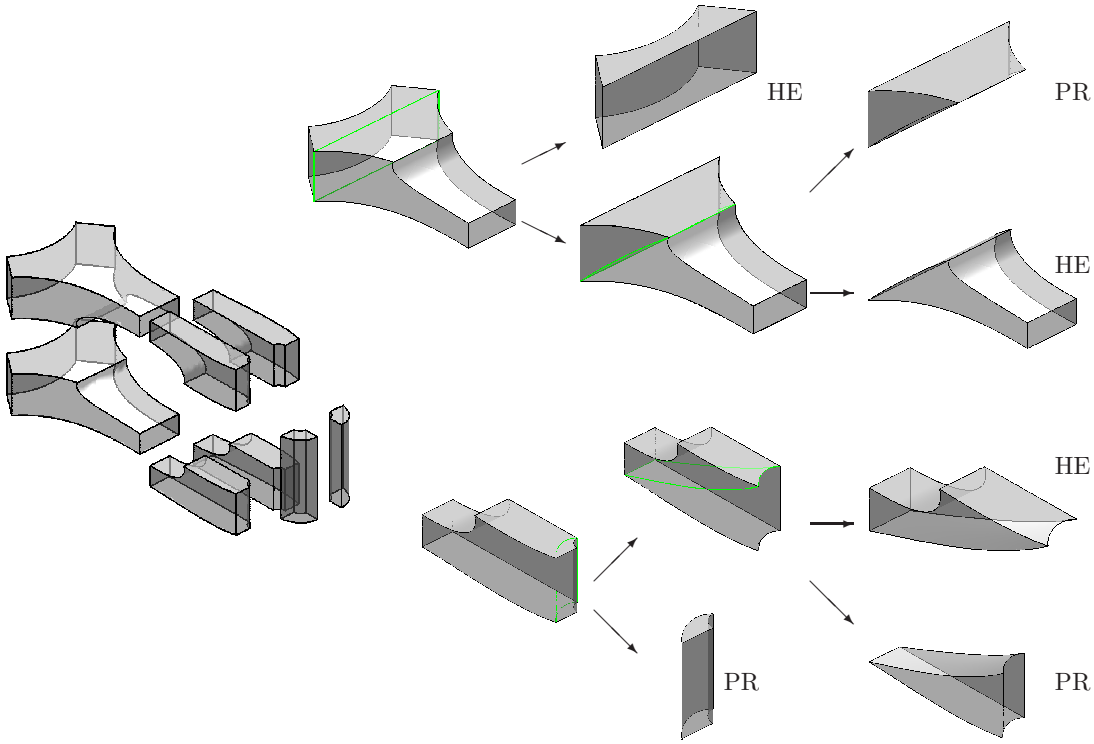


Figure 15: Left: manual preprocessing of the stand piece using the 4-cut strategy. Right: segmentation of the two classes of non-hexahedral pieces.

solid has been segmented into a number of contractible solids with only non-convex edges. We use the techniques developed in the present paper to subdivide them automatically into base solids. Each base solid is then further subdivided to topological hexahedra.

5. Creating trivariate NURBS patches. Finally we create NURBS volume parameterizations for each of the resulting topological hexahedra. Currently we use a simple method which is based on trivariate Gordon-Coons parameterizations and NURBS fitting. More sophisticated techniques have been studied in the literature, see [38] and the references therein.

There are several geometric challenges in the process pipeline which will benefit from a more detailed investigation. These include

- improved techniques for segmenting the faces of the given solid, in particular if blending surfaces are present,
- advanced selection strategies for cutting loops, which take the shape of the resulting sub-solids into account,

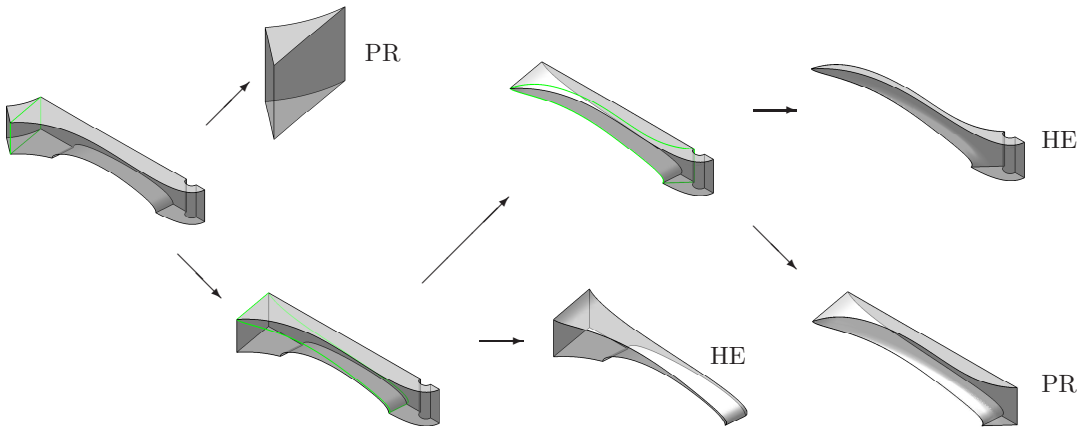


Figure 16: A segmentation of the stand piece, preprocessed by cutting along symmetries. This segmentation yields two triangular prisms and two hexahedra. Thus, each quarter of the stand is decomposed into 8 hexahedra.

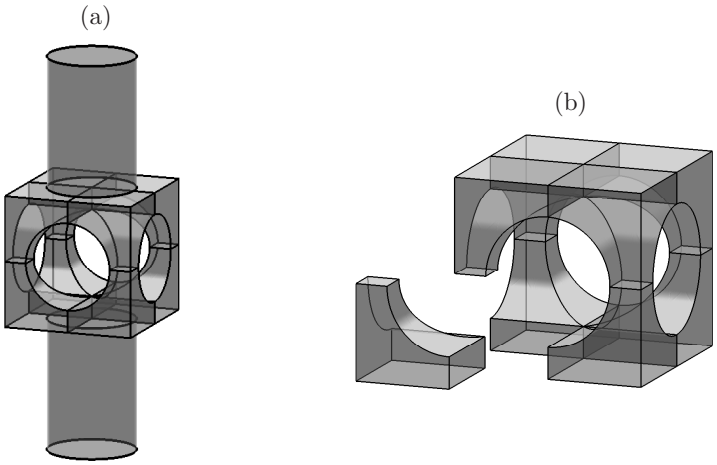


Figure 17: (a): The mechanical part of Zhang and Bajaj. (b): a holey cube of genus 3, the remaining component of the mechanical part after removing the two cylindrical components. The holey cube can be subdivided into eight identical pieces up to mirror symmetries. Only one of the pieces is considered for our segmentation.

- automatic and reliable methods for creating auxiliary edges on existing faces and cutting surfaces from cutting loops, and
- techniques for avoiding T-joints.

It has been the aim of the paper to establish the isogeometric segmentation problem and to initiate the research which is required to address it. We emphasize that although

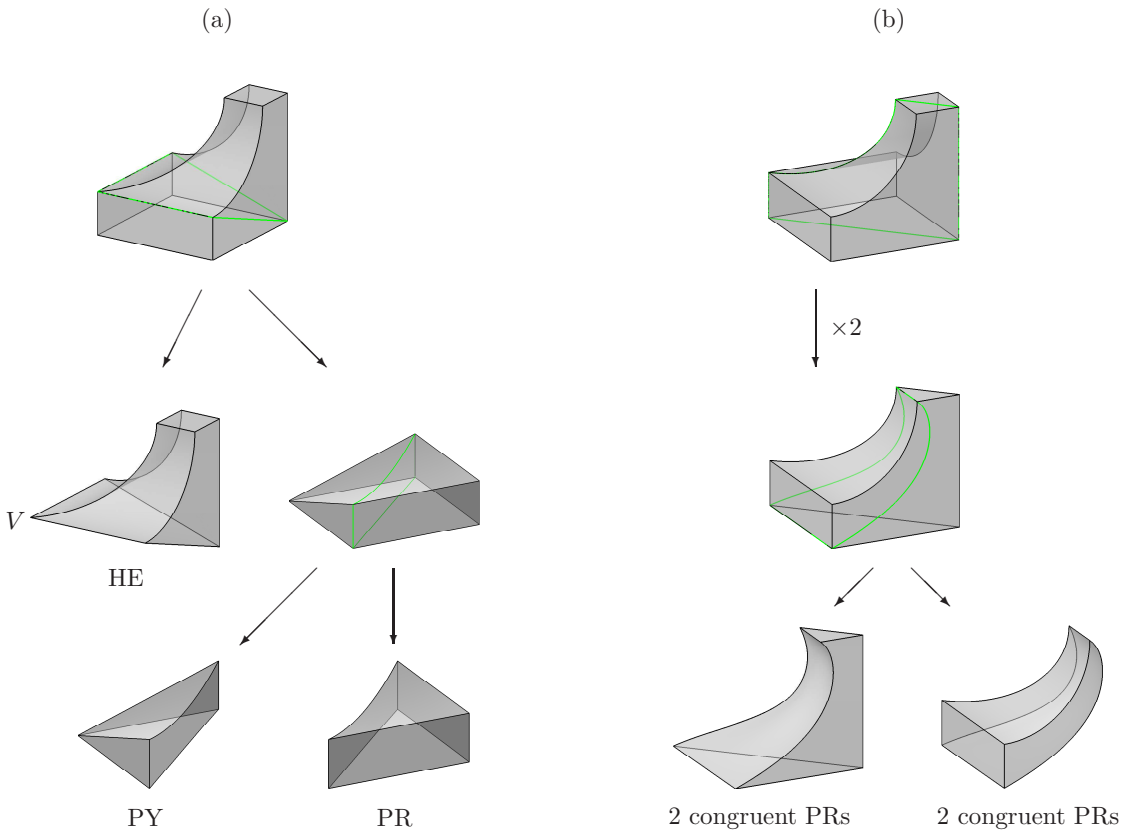


Figure 18: Two segmentations of the solid that is extracted from a mechanical part described in Figure 17b. Each of the two segmentations results in 12 topological hexahedra. In (a), we note that at the vertex V of the (only) HE, each of the side faces (in the present view) meets the bottom face at a zero angle.

this paper only considers solids without non-convex edges, it provides some fundamental results for the entire process which are still valid for solids with non-convex edges, e.g, the criterion for a cutting loop to be (topologically) valid presented in Proposition 5. Reliable and efficient solutions for this problem will be needed if isogeometric analysis is to provide a valuable alternative to standard approaches for numerical simulation. There are challenging problems to solve, but we believe that the possible benefits of a unified geometric representation for analysis and design make it worthwhile to invest time and effort into this project.

8. Conclusion

After formulating the isogeometric segmentation problem for general solids, we presented an edge graph-based volume segmentation algorithm for solving this problem in the case of solids without non-convex edges. This algorithm forms an essential step of the process pipeline addressing the full problem. It repeatedly decomposes a solid into a collection

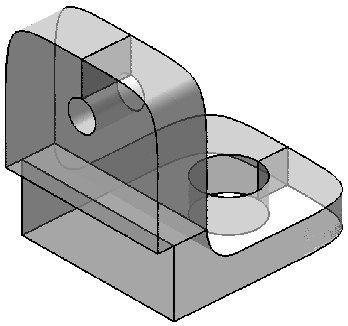


Figure 19: Mechanical part of Yamakawa and Shimada, with smooth surfaces, and preprocessed to deal with non-contractibility and non-convex edges

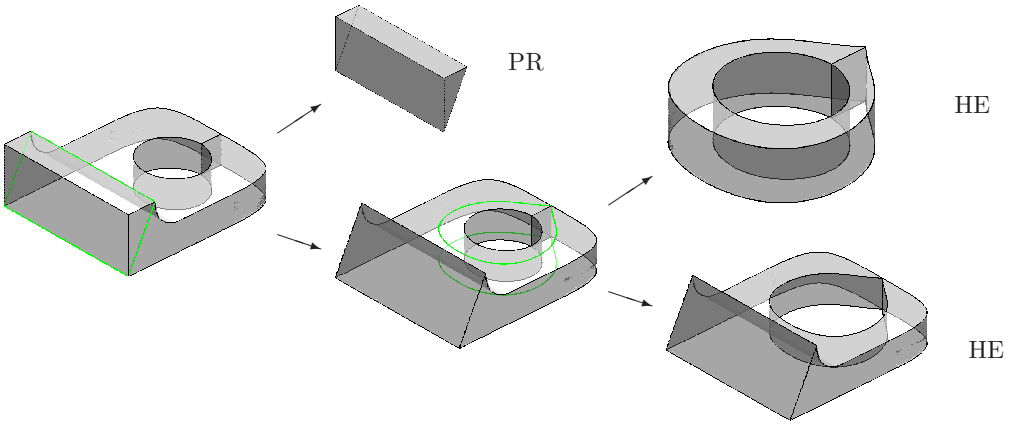


Figure 20: Segmentation of the base of the mechanical part of Yamakawa and Shimada, preprocessed using the 1-cut strategy.

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. We experimented with several examples to choose a decomposition sequence which resulted in a small number of hexahedra.

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. We are currently investigating more sophisticated criteria which make selections more compatible with geometric intuition.

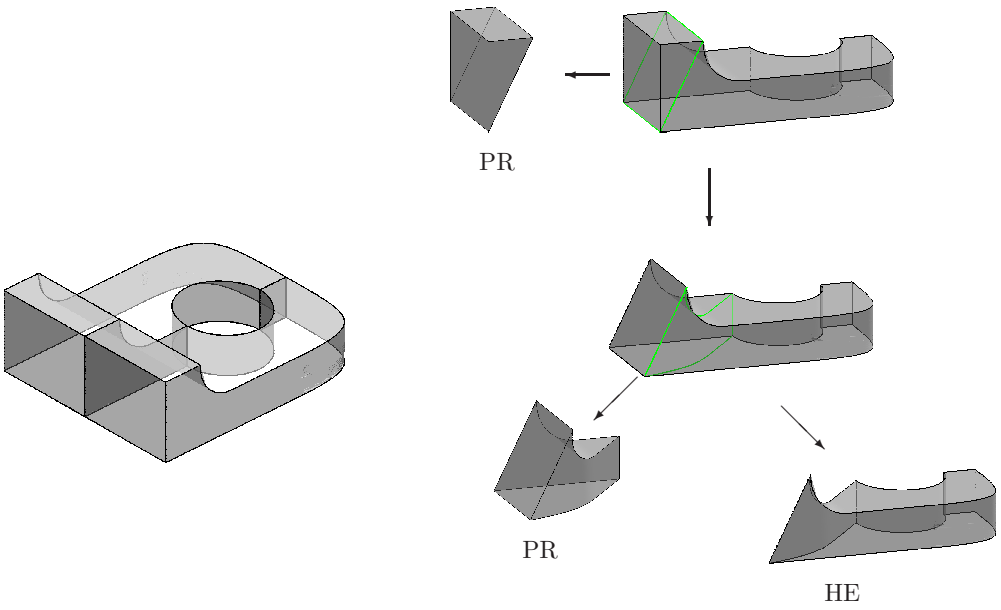


Figure 21: Left: The base of the mechanical part of Yamakawa and Shimada, preprocessed according to the 2-cut stragety. Right: resulting segmentation of one of the two identical pieces.

While the paper is restricted to edge graphs of solids with only convex edges and with 3-vertex-connected edge graphs, the extension to solids with non-convex edges is studied in the follow-up paper [32]. The geometric realization of cutting loops, i.e. the construction of surfaces representing the new faces, is a difficult problem and will be the subject of future work.

The segmentation of the piece of the stand in Figure 14 and 15, obtained after a preprocessed decomposition into more-or-less cube-shaped solids, resulted in better shaped pieces than the segmentations in Figures 13 and 16. This suggests that our procedure produces the best shapes when there are limits on the total curvature of any edge or face as well as the ratio of lengths of any two edges. A preprocessing step satisfying such conditions could help to get better results with respect to the shape of the resulting topological hexahedra, and could be an interesting subject for further investigation.

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. [39]; e.g., by removing blends between faces) will be of great help for solving the isogeometric segmentation problem in practice.

Acknowledgment. The authors wish to thank the anonymous reviewers for their comments that helped to improve the paper. We owe special thanks to Dr. M. Schifko (ECS Magna Powertrain). Dr. S. Boschert (Siemens CT) and D. Mayer for their support with

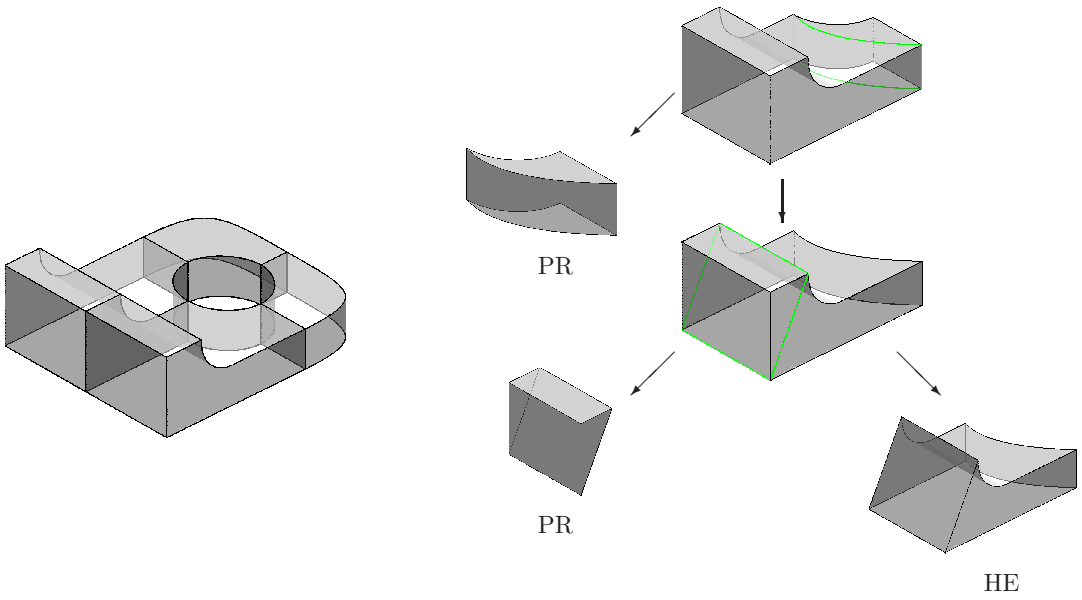


Figure 22: Left: The base of the mechanical part of Yamakawa and Shimada, preprocessed according to the 4-cut stragety. Right: resulting segmentation of one of the two identical pieces that are not topological hexahedra.

the processing of the TERRIFIC part. 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] D.-M. Nguyen, M. Pauley, B. Jüttler, Isogeometric segmentation. Part II: On the segmentability of contractible solids with nonconvex edges, to appear in Graphical Models.

[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.

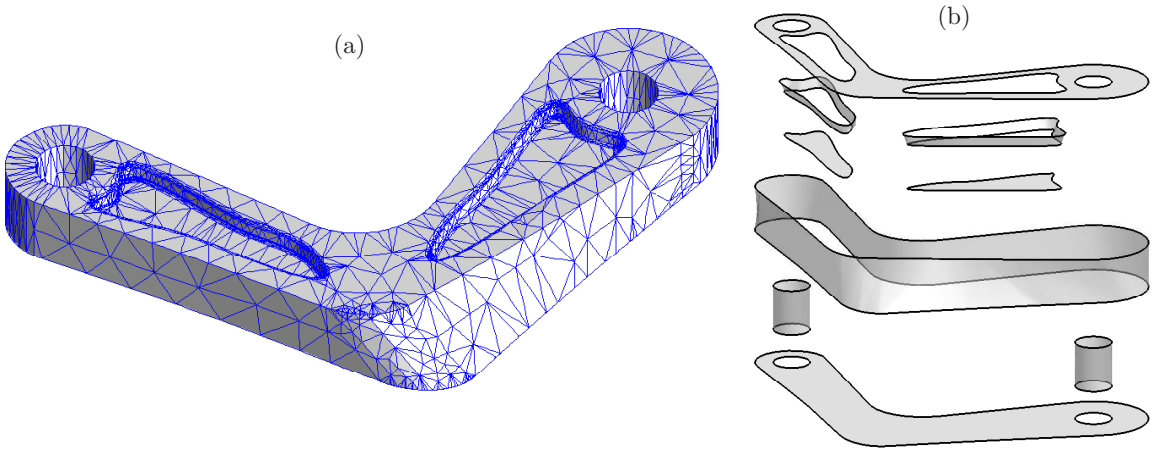
[36] J. Y. Yen, Finding the K shortest loopless paths in a network, Management Sci. 17 (1970/71) 712–716.

[37] S. Yamakawa, K. Shimada, HEXHOOP: Modular templates for converting a hex-dominant mesh to an all-hex mesh, Engineering with Computers 18 (3) (2002) 211–228.

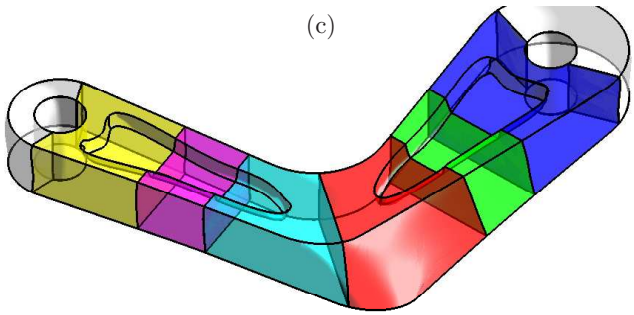
[38] T. Nguyen, B. Jüttler, Parameterization of contractible domains using sequences of harmonic maps, in: J.-D. Boissonnat, et al. (Eds.), Curves and Surfaces, Vol. 6920 of Lecture Notes in Computer Science, Springer, 2012, pp. 501–514.

[39] 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.

Step 1: CAD model simplification and preparation



Step 2: Adding auxiliary edges to obtain contractible solids with 3-vertex-connectivity



Step 3: Dealing with non-convex edges, and Step 4: Subdividing a contractible solid with only convex edges

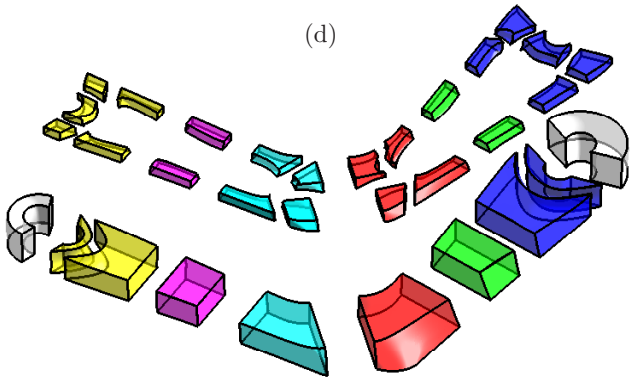


Figure 23: Illustration of the isogeometric segmentation pipeline presented in Section 7, using the TERRIFIC part as an example. In (a), we show a triangulation of the input solid; the faces of this triangulation are fit by trimmed NURBS surfaces shown in (b). The contractible solids depicted in (c) are segmented into the 32 topological hexahedra in (d). The topological hexahedra in (d) are obtained from the solids of the same color in (c). The final step of the pipeline is to create trivariate NURBS patches (not shown).