Isogeometric Segmentation: Construction of cutting surfaces

Michael Haberleitner, Bert Jüttler

michael.haberleitner@jku.at, bert.juettler@jku.at

Altenberger Str. 69, 4040 Linz,
Institute of Applied Geometry, Johannes Kepler Universität Linz, Austria

Abstract

The objective of Isogeometric Segmentation is to generate a decomposition of a solid, given in boundary representation, into a collection of a relatively small number of base solids, which can easily be subdivided into topological hexahedra. This can be achieved by repeatedly splitting the solid. In each splitting step, one chooses a cutting loop, which is a cycle of curves around the boundary of the solid, and constructs a cutting surface that splits the solid into two simpler ones. When only hexahedra or pre-defined base solids are left this process terminates.

The construction of the cutting surface must ensure that two essential properties are fulfilled: the boundary curves of the surface interpolate the previously constructed cutting loop and the surface neither intersects itself nor the boundary of the solid. A novel method for generating the cutting surface is presented in this paper. The method combines two steps: First we generate an implicit guiding surface, which is subsequently approximated by a trimmed spline surface in the second step.

Keywords: Isogeometric analysis, segmentation, trimmed surface fitting, implicit guiding surface, parameterization, collision avoidance

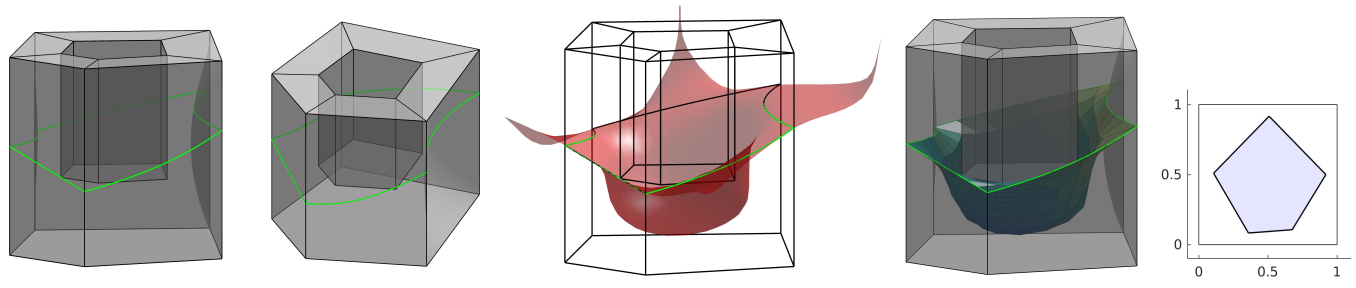


Figure 1: From left to right: Vase-shaped object with non-planar faces seen from two different viewing directions, implicit guiding surface, parameterized surface patch, and its automatically created parameter domain.

1. Introduction

Since its introduction by T.J.R. Hughes et al. in 2005 [1], the framework of isogeometric analysis has attracted rapidly growing attention from the numerical analysis and the geometric modeling communities. The underlying idea, namely to combine finite element analysis with geometric design by reusing the same basis functions, has led to significant improvements of the interaction between the representations used in Computer-Aided Design and in Numerical Simulation, see [2] for more information. The current state-of-the-art in this field is captured by the two recent special issues of influential journals [3, 4].

With the growing interest in isogeometric analysis, it was soon noticed that the realization of its potential advantages requires to address new challenging problems. A prominent example is the need to develop techniques for creating NURBS-based domain parameterizations from boundary-represented CAD data, as the resulting NURBS representations provide the basic description of geometric data for isogeometric analysis, cf. [5].

These parameterizations may be classified into single– and multi-patch representations. The construction of single patch spline models from boundary data has been addressed by numerous publications. We briefly mention some of them: Gravesen et al. address the challenge of creating a regular single-patch domain parameterization from boundary data [6]. A method for volumetric parameterization and trivariate B-spline fitting using harmonic mappings has been described by Martin et al. [7] for objects of cylindrical topology. Zhang et al. [8] describe a construction of a solid T-spline parameterization for genus zero objects from triangulated boundary data. This method has later been extended to solids possessing an arbitrary topological genus [9].

Although it has some benefits, the use of a single patch imposes severe constraints on the topology of the domain. Algorithms for creating multi-patch representations, that provide increased flexibility, are therefore of vital interest. Typically, such algorithms consist of two steps: First the domain is subdivided into a collection of topological hexahedra. Second, one constructs a spline parameterization for each of these blocks. In order to benefit from the potential advantages of isogeometric analysis, one should construct segmentations into relatively few hexahedral patches. This is quite different from the usual approach to hexahedral mesh generation, which has been studied in the context of the classical finite element method, see e.g. [10] and the references cited therein.

Parameterization techniques for multi-patch domains have been studied by Xu et al. [11] using variational methods. A combinatorial approach to planar multi-patch domains, which is based on a complete enumeration of the possible patch layouts, has been described recently in [12]. Suitable spline spaces for multi-patch domains have been analyzed in [13, 14].

The problem of decomposing a domain into a small number of topological hexahedra, which are suitable for spline parameterizations, has been called the isogeometric segmentation problem in [15]. One may distinguish between two approaches:

The first one uses splines on polycube domains, see e.g. [16, 17]. The parameterization algorithm first generates a polycube domain (i.e., a collection of cubes) that resembles the given solid object, and constructs a parameterization by considering a deformation that transforms the domain into the solid. Al Akhras et al. combine polycubes with pants decomposition of the boundary surface to subdivide the given solid into a collection of cuboids [18]. Clearly, the polycube-based approach is quite powerful but has some difficulties when dealing with features (sharp edges) on the boundary. This problem has been addressed recently in [19].

The second approach, which has been established in a series of papers [15, 20, 21], is based on iterated splitting of the initial solid using cutting surfaces. This surface is obtained from a cutting loop, which is a cycle of curves on the solid's boundary surface. While the selection of the loop and the construction of its curve segments is now well understood, the actual construction of the cutting surface has not yet been investigated.

The current paper focuses on this problem, which is an essential ingredient of the isogeometric segmentation pipeline described in [22]. Given a three-dimensional solid in boundary representation as a collection of trimmed NURBS surfaces, and a cutting loop, we generate a representation of the cutting surface as a trimmed NURBS surface patch. Its boundary interpolates the given cutting loop, but its interior must not intersect the boundary of the solid.

Two different techniques will be combined in order to solve this problem. First we construct an *implicit spline surface*, that roughly interpolates the cutting loop and stays away from the other parts of the solid's boundary. This surface is obtained using methods for implicit spline surface fitting. In the second part we use techniques for trimmed *spline surface fitting* to obtain a cutting surface that simultaneously approximates the given cutting loop in the boundary and the implicit guiding surface in the interior.

Implicit curves and surfaces are a well-established tool for geometry reconstruction [23, 24, 25, 26]. More recently, Wang et al. use implicit PHT-splines to reconstruct curves and surfaces [27], while Pan et al. [28] employ a low-rank tensor approximation technique to reduce the complexity of the required computer memory.

Spline surface fitting addresses the problem of geometry reconstruction using parametric curves and surfaces, see the surveys [29, 30] for more information. In particular, techniques for spline surface fitting to implicit surfaces are of interest. Related work includes a paper by Wurm et al. [31], who find a tensor-product spline surface representation of a given algebraic surface by minimizing a non-linear objective function.

The remainder of this article consists of four major parts. First we give a detailed explanation of the cutting problem and introduce the notions that will be used throughout the paper in Section 2. We then formulate a suitable constrained optimization problem in Section 3, which allows us to obtain an implicit guiding surface. As the next step we discuss the construction of a parametric representation of the cutting surface in Section 4. Finally we present several computational results that illustrate our approach in Section 5.

Figure 1 visualizes the whole procedure: The left two pictures show a three-dimensional solid and a given cutting loop[1], cf. Section 2. The picture in the center depicts an implicit guiding surface and the last two pictures show the parametrized cutting surface and its automatically generated trimming-loop.

2. Preliminaries

We consider the problem of decomposing a d-dimensional simply connected domain (a solid object), which is given in boundary representation into two smaller simply connected domains for $d = 2, 3$. More precisely we are given a list of n *facets*

$$F_i : \Omega_i \subset [0,1]^{d-1} \to \mathbb{R}^d \quad \text{for} \quad i = 1, \ldots, n,$$

[1]Note that the cutting loop in this example does not induce a meaningful segmentation of the solid. It was artificially chosen in order to illustrate the required properties of cutting surfaces. This comment also applies to the cutting data in Figs. 5, 6 and 12. In contrast, real cutting loops have been used in Figs. 13 and 15.

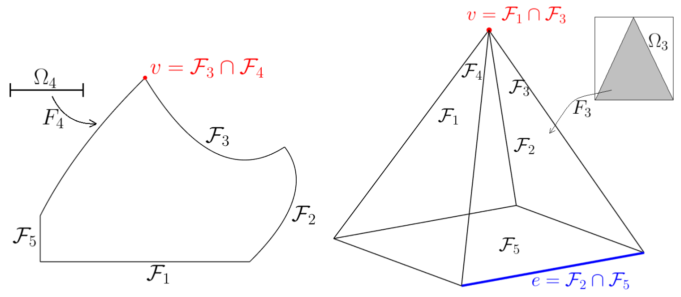


Figure 2: Left: A 2D solid with five facets (curve segments) and one highlighted ridge (vertex). Right: A 3D solid with five facets (trimmed surface patches) and two highlighted ridges (vertex and edge).

such that $\bigcup_i F_i(\Omega_i)$ is the boundary of a solid object S in $\mathbb{R}^d$. We distinguish between a parameterized facet F_i and its geometric locus $\mathcal{F}_i = F_i(\Omega_i)$. For simplicity we use the notion *facet* for both of them.

In the two-dimensional case ($d = 2$), the parameter domains Ω_i are intervals and the associated facets F_i are simply segments of planar curves. The solid S is the planar domain that is bounded by these segments.

A three-dimensional solid ($d = 3$) is a domain in $\mathbb{R}^3$ that is bounded by surface patches F_i. More precisely, one considers trimmed surface patches, as the parameter domains Ω_i can be general solids in the plane, i.e., planar domains which are bounded by a curve polygon (see Fig. 2).

The non-empty intersections $\mathcal{F}_i \cap \mathcal{F}_j \subset \mathbb{R}^d$ for $i \neq j$ will be called *ridges*. In the planar case ($d = 2$), the ridges are the start- and end-points of the boundary curves and therefore vertices. The ridges can be either edges or vertices for dimension $d = 3$, see Fig. 2.

In order to decompose the given solid into two smaller ones, we need to construct two new lists of facets $F_i^\ell(\Omega_i^\ell)$, $\ell = 1, 2$, with the following properties:

- The facets are represented by parameterizations, i.e., $F_i^\ell : \Omega_i^\ell \subset [0,1]^{d-1} \to \mathbb{R}^d$ for $i = 1, \ldots, n^\ell$.
- Each list of facets $\bigcup_i \mathcal{F}_i^\ell$ forms the boundary of a simply connected solid object S^ℓ in $\mathbb{R}^d$.
- The input solid S is the disjoint union of the two newly generated ones, i.e., it satisfies $S = S^1 \cup S^2$ and $(S^1)^\circ \cap (S^2)^\circ = \emptyset$.

Intuitively speaking, we are cutting the solid S into two smaller solids S^1 and S^2.

In addition to the properties listed above, we assume that exactly one facet needs to be generated in order to cut the given solid. More precisely, exactly one facet of each solid S^ℓ is not contained in one of the given facets. The newly generated facet will be called the *cutting facet* C. It is also called cutting curve and cutting surface for dimension $d = 2$ and $d = 3$, respectively.

Without loss of generality, the first facet of the two subsolids is assumed to be the cutting one, i.e., it satisfies

$$\Omega_1^1 = \Omega_1^2 \text{ and } F_1^1 = F_1^2$$

while the remaining facets fulfill

$$\exists k : \quad F_i^\ell(\Omega_i^\ell) \subset F_k(\Omega_k) \text{ for } i > 1.$$

We will construct the *cutting facet* $\mathcal{C} = C(\Omega)$,

$$C : \Omega \to \mathbb{R}^d, \text{ where } \Omega = \Omega_1^1 = \Omega_1^2 \text{ and } C = F_1^1 = F_1^2$$

from given *cutting data* $\mathcal{D} = (L, \vec{t})$, which is a pair consisting of boundary ridges L and associated tangent vectors $\vec{t} : L \to \mathbb{R}^d$, see Figs. 3 and 4.

The *boundary ridges* specify the boundary of the cutting facet:

- In the planar case we use two distinct vertices on the boundary, which can be either existing vertices or newly created ones, see Fig. 3. The latter ones will be called auxiliary vertices.
- A closed loop L of at least three curve segments on the boundary, meeting in vertices, is required for dimension $d = 3$, see Fig. 4. The loop must not intersect itself. Again, we use either existing or newly created (auxiliary) segments and vertices.

The tangent vectors control the tangent vector resp. the tangent plane of the cutting facet along the boundary. For each point $x \in L$, we specify a tangent vector $\vec{t}(x)$ that points to the interior of the solid S. For dimension $d = 3$, it is required that the vectors vary smoothly along the curve segments and define a consistent tangent plane at the vertices.

It should be noted that the cutting data may be required to satisfy additional assumptions, which are needed by the overall segmentation algorithm, see [15, 20]. For instance, the two points of L are not allowed to lie on the same boundary segment for dimension $d = 2$, and the loop L must not visit any facet more than once for dimension $d = 3$. We do not discuss these assumptions in more detail, since they are not relevant for the construction of the cutting facet.

The cutting facet C has to satisfy the following conditions:

- Boundary interpolation: $C(\partial\Omega) = L$,
- Boundary tangent interpolation: $\vec{t}(C(u))$ is contained in the tangent plane of C at $C(u)$ for $u \in \partial\Omega$,
- No collision with the boundary: $C(\Omega^\circ) \subset S^\circ$,
- Regularity: C is injective and regular (in particular, it does not intersect itself).

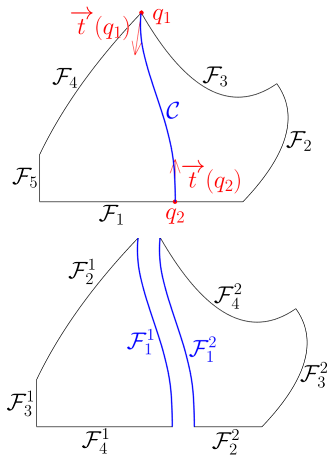


Figure 3: Top: Two-dimensional solid with valid cutting data and a suitable cutting curve. Bottom: Splitting into two sub-solids along the cutting curve.

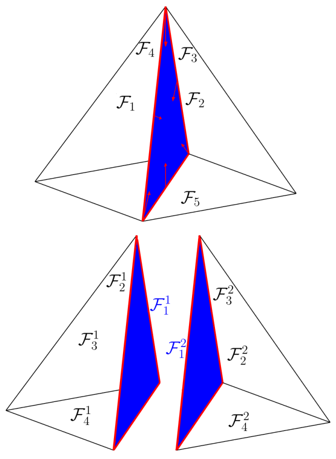


Figure 4: Top: Three-dimensional solid with valid cutting data and a suitable cutting surface. Bottom: Splitting into two sub-solids along the cutting surface.

In the case $d = 2$, the cutting facet C is a curve segment, that connects two given points on $\bigcup_i \mathcal{F}_i$, while staying inside the domain bounded by the facets $\mathcal{F}_i$, see Figure 3 (top). For dimension $d = 3$ the cutting facet is a surface patch whose boundary coincides with L. Again C stays inside the domain bounded by the facets. In every point x on ∂p, the tangent plane of C contains the given vector $\vec{t}(x)$, see Figure 4 (top).

Given a d-dimensional solid given by its boundary facets $F_i(\Omega_i)$, and cutting data $\mathcal{D}$, we present a novel method that generates a cutting facet C. For the planar case, this problem has already been dealt with in [21]. In this paper, the authors proposed a method based on the minimization of certain penalty functionals, that led to satisfying results. In principle, it would be possible to adapt this method to the three-dimensional case. However, the computational effort is rather high. Therefore we introduce an alternative algorithm that is simultaneously suitable for dimensions $d = 2$ and $d = 3$. We summarize it in the following

```
Algorithm: Cutting surface generation
Input: Solid $S$ and cutting data $\mathcal{D}$.
1. Find an implicit guiding curve / surface.
2. Parameterize the guiding curve / surface.
Output: parameterized cutting curve / surface
```

Depending on the dimension d, we are looking either for a curve or for a surface. For simplicity, we will always refer to surfaces, independently of the dimension d. Similarly, we will always refer to L as the cutting loop, even if this is not accurate for $d = 2$.

3. Implicit guiding surface

The cutting loop L separates the boundary ∂S into two simply connected, disjoint pieces $\overline{b}$ and $\underline{b}$, i.e., $\partial S \setminus L = \overline{b} \cup \underline{b}$. (We do not consider solids whose boundary is not split into two components by a cutting loop, such as the horned sphere. These solids are not likely to occur in CAD applications.) We obtain the guiding surface with the help of a level set function, that takes positive values on $\overline{b}$, negative values on $\underline{b}$ and vanishes on the loop L.

We need to transform the given cutting data into suitable boundary data for the level set function. To do so, we compute a normal vector $\vec{n}(x)$ for each point $x \in L$.

In the planar case, we simply rotate the vector $\vec{t}(x)$ by $\pi/2$ towards $\overline{b}$, see Fig. 5, left. In the case $d = 3$, we compute the tangent plane at all points $x \in L$ and pick the normal vector that points towards $\overline{b}$, see Fig. 5, right. Note that we assumed a consistent distribution of the tangent vectors, thereby ensuring the existence of a tangent plane at all points of the cutting loop (including vertices).

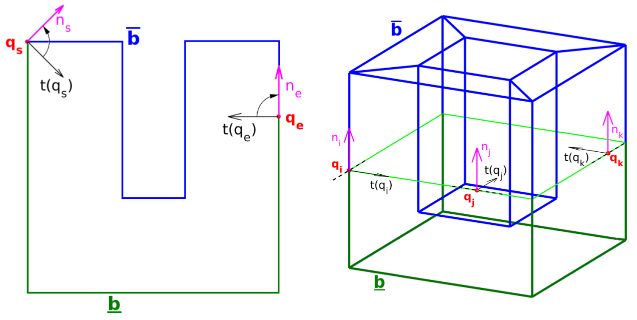


Figure 5: Boundary data for the construction of the guiding surface for $d = 2$ (left) and $d = 3$ (right).

The guiding surface is defined by a level set function that solves the *cutting problem*: Find $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ such that

$$\begin{cases} f(x) = 0, & \forall x \in L, \\ \nabla f(x) = \vec{n}(x), & \forall x \in L, \\ f(x) > 0, & \forall x \in \overline{b}, \\ f(x) < 0, & \forall x \in \underline{b}. \end{cases} \tag{CP}$$

In order to make this accessible to a numerical solution, we perform a discretization. We choose d-variate tensor-product B-splines $(\phi_i)_{i\in\mathcal{I}}$, where $\mathcal{I}$ is some index set, which are defined in the bounding box of the solid S. More precisely, we use cubic splines with open knot vectors and uniform inner knots, the number of which is specified by the user (typically either 3 or 7 knots per direction in our examples). The discretized level set function takes the form

$$f^*(x) = \sum_{i\in\mathcal{I}} \phi_i(x) c_i \tag{1}$$

with unknown scalar coefficients c_i. They are found by solving the *discretized cutting problem*,

$$\begin{cases} f^*(x) \approx 0 & \forall x \in L^* \\ \nabla f^*(x) \approx \vec{n}(x) & \forall x \in L^* \\ f^*(x) \geq \delta & \forall x \in \overline{b}^* \\ f^*(x) \leq -\delta & \forall x \in \underline{b}^*. \end{cases} \tag{DCP}$$

The influence of the user-specified discretization parameter $\delta > 0$ will be studied later in Section 5.

The discretized sets L^*, $\overline{b}^*$ and $\underline{b}^*$ are created by sampling points on the curves and surfaces, respectively. We exclude a small strip around the loop L when generating $\overline{b}^*$ and $\underline{b}^*$.

Neither (CP) nor (DCP) have unique solutions. Therefore we introduce an optimization problem, that allows us to identify the optimal solution with respect to a combination of certain quality measures,

$$F(\mathbf{c}) = \lambda_{\text{int}} F_{\text{int}}(\mathbf{c}) + \lambda_{\tan} F_{\tan}(\mathbf{c}) + \lambda_{\text{reg}} F_{\text{reg}}(\mathbf{c}) \to \min \tag{2}$$

where $\mathbf{c} = (c_i)_i$ is the vector of coefficients, see (1), and the parameters λ_{int}, $\lambda_{\tan}$, λ_{reg} are positive weights. They control the influence of the different contributions, cf. Section 5.

The first quality measure

$$F_{\text{int}}(\mathbf{c}) = \sum_{x\in L^*} f^*(x)^2$$

represents the approximate interpolation constraint for the cutting loop, and the second one

$$F_{\tan}(\mathbf{c}) = \sum_{x\in L^*} \|\nabla f^*(x) - \vec{n}(x)\|^2$$

is used to constrain the normal vectors (and hence the tangent planes of the level set surface) along the cutting loop. Additionally we regularize the solution by considering the functional

$$F_{\text{reg}}(\mathbf{c}) = \int_{[0,1]^d} \sum_{i,j} \left(\frac{\partial^2}{\partial x_i \partial x_j} f^*(x) \right)^2 \mathrm{d}x, \tag{3}$$

that is related to the curvature of the level set surface.

We obtain the unknown coefficients by solving the *Regularized + Discretized Cutting Problem*

$$\begin{cases} F(\mathbf{c}) \to \min \\ \text{subject to } f^*(x) \geq \delta & \forall x \in \overline{b}^* \\ \text{and } f^*(x) \leq -\delta & \forall x \in \underline{b}^* \end{cases} \tag{RDCP}$$

Lemma 1. *The regularized + discretized cutting problem RDCP has a unique solution if the set of feasible points (which is defined by the inequality constraints) is non-empty.*

Proof. The objective function is convex if the weights λ_{int}, $\lambda_{\tan}$, λ_{reg} are all positive, [see 26, Proposition 1]. Thus we obtain a convex optimization problem, since the inequality constraints are linear. □

Minimization of the functional (2) requires the evaluation of the integral (3) multiple times. For a fast computation we exploit the tensor-product structure of the spline function f^* in order to precompute univariate integrals of B-spline basis functions. The evaluation of (3) can then be performed by a fast matrix-vector multiplication.

In our examples, we solved RDCP using the fmincon function of Matlab$^{\text{TM}}$. Typical problems, where the number of spline coefficients varies between 100 (for curves) to 1000 (for surfaces), can be solved within a few seconds or minutes on standard hardware.

An empty set of feasible points results if the number of knots, which is used to define the tensor-product splines $(\phi_i)_{i\in\mathcal{I}}$, is too small. We increase the number of knots by dyadic refinement until RDCP possesses solutions. Additionally we increase the number of knots if the errors

$$f^*(x) \text{ and } ||\nabla f^*(x) - \vec{n}(x)||, \quad x \in L^*,$$

exceed a user-defined tolerance. Recall that the values $f^*(x)/||\nabla f^*(x)||$ estimate the distance of the point x to the level set surface $f^* = 0$. We may omit the denominator, since the gradients of f^* approximate unit normal vectors along the loop.

Figure 6 shows several instances of guiding curves and surfaces. All curves and surfaces are represented as zero level sets of cubic spline functions.

The zero level set of the function f^* guides the construction of the cutting surface. The latter surface is represented by a trimmed NURBS patch, the construction of which will be addressed in the next section. It should be noted that the guiding surface may have extraneous components as in Fig. 7, which are not considered when creating the cutting surface. It is also possible to place auxiliary obstacles that influence the shape of the guiding surface.

4. Parameterization of the guiding surface

The parameterization step uses different approaches for dimensions two and three.

First we consider the curve case, i.e., dimension $d = 2$. A variety of parameterization techniques for implicitly defined curve segments exists. We implemented a marching algorithm, which uses a predictor-corrector method to generate a polygon, and combined it with least squares fitting, in order to construct a parametric spline curve, that approximates the resulting polygon. The cutting data (two

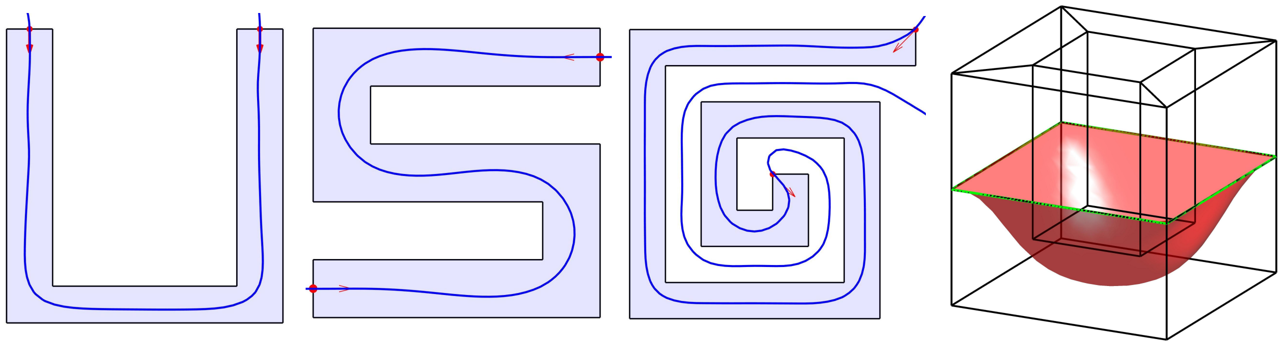


Figure 6: Guiding curves and surfaces, represented as zero level sets of cubic spline functions.

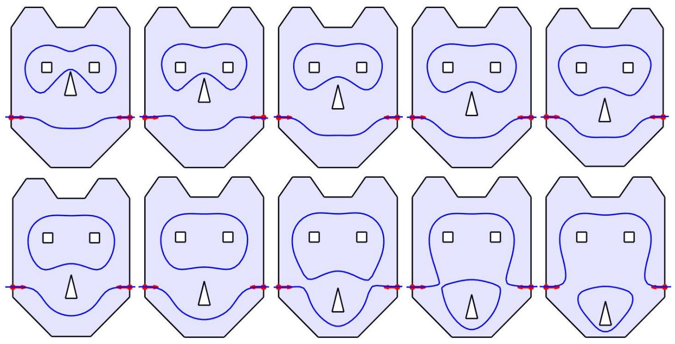


Figure 7: Modification of the shape of the guiding curve by using auxiliary obstacles.

end points and associated tangents) are used to define end points and boundary tangents for the curve obtained by the least-squares fitting. The accuracy of the final result can be controlled by adjusting the number of control points for the approximating curve and by choosing an appropriate sampling density (depending on the stepsize of the marching algorithm).

The second case, where we need to find an approximate parameterization of the guiding surface, is far more challenging to deal with. In fact, it requires the solution of two non-trivial problems.

We construct an approximate parameterization of the guiding surface by a trimmed spline patch

$$\begin{aligned} p : \Omega \subset [0,1]^2 &\to \mathbb{R}^3, \\ (u,v) &\mapsto \sum_i \psi_i(u,v) d_i. \end{aligned} \tag{4}$$

The parameterization is defined by tensor-product B-splines ψ_i, control points $d_i \in \mathbb{R}^3$ and a trimmed parameter domain Ω. The spline patch has to satisfy two conditions:

1. Its boundary $p(\partial\Omega)$ interpolates the cutting loop L and the tangent planes along the boundary contain the associated tangent vectors. In addition, the tangent vectors point to the interior of the surface patch.
2. The interior of the surface is an approximate parameterization of the guiding surface. More precisely, the composition $f^* \circ p$ is close to zero on Ω.

The patch is constructed in two steps.

Step 1 - Construction of the trimmed parameter domain. We choose the domain Ω as a planar polygon, contained within the unit square, that mimics the shape of the cutting loop. This similarity will lead to better results in the second (parameterization) step.

We simply use the unit square $[0,1]^2$ for four-sided cutting loops whenever this is appropriate, thereby avoiding trimming in this case. A more general polygon is used to define the domain if the square is not suitable.

More precisely, we choose this polygon such that the ratios of adjacent edge lengths are approximately equal to the ratios of the lengths of the corresponding curve segments, and the oriented angles between incoming and outgoing tangents at vertices are approximately the same and preserve the sign. Several techniques are used to find this polygon [32]:

- The first one simply projects the vertices of the cutting loop into a plane and connects them by straight lines. This simple method already leads to good results in most cases, since the segmentation algorithm described in [15, 20, 22] generally uses relatively simple cutting loops (in particular near-planar ones).

- Some solids however do not allow for (near-) planar cutting loops, and therefore a projection into a plane would not result in a valid polygon. In this case, a second technique is used which relies on quadratic optimization:

$$\arg\min_x \,(Ax-b)^T(Ax-b) \quad \text{subject to} \quad Cx = d \tag{5}$$

where

$$A = \operatorname{diag}(\frac{1}{\ell_i}), \quad b = (1,\dots,1)^T, \quad d = (0,0)^T, \quad \text{and}$$

$$C = \begin{pmatrix} \cos(\alpha_1) & \cos(\alpha_1+\alpha_2) & \cdots & \cos(\sum_{i=1}^n \alpha_i) \\ \sin(\alpha_1) & \sin(\alpha_1+\alpha_2) & \cdots & \sin(\sum_{i=1}^n \alpha_i) \end{pmatrix}.$$

The values ℓ_i are the lengths of the original cutting loop in 3d, while one obtains α_i by scaling the tangent turning angles of the cutting loop, such that

$\sum_{i=1}^{n} \alpha_i = 2\pi$. As result of the optimization (5) one obtains a set of edge-lengths x_i, that together with the angles α_i form a closed planar polygon, which mimics the shape of the cutting loop.

- Additionally we also consider a third technique as fall-back strategy, that generates a curved polygon with vertices placed on a circle. This technique is to be used if the optimization fails or returns a self-intersecting polygon.

Step 2 - Finding the control points. The surface patch p is a trimmed bicubic tensor-product spline patch with uniform knots. Initially we choose 3 inner knots per direction and refine if the accuracy of the result is not sufficient.

We generate the control points by solving a constrained optimization problem. The objective function is a combination of three terms:

- The first term

$$\int_{\Omega} \omega(t)[(f^* \circ p)(t)]^2 \, \mathrm{d}t \tag{6}$$

  measures the closeness to the implicit guiding surface. Note that the composition $(f^* \circ p)$ of the two functions f^* and p vanishes at the parameter t if the implicit curve defined by $f^* = 0$ interpolates the point $p(t)$. The weighting function ω is procedurally defined such that it is zero at the domain boundary and increases towards the center of the domain, finally reaching the value 1 in the center of the domain. Using the weighting function improves the accuracy of the boundary interpolation, since the cutting surface approximates the exact cutting data but not the implicit guiding surface near the boundary.

- The second term ensures the approximation of the cutting data. It takes the form

$$\int_{\partial\Omega} \|p(t) - L(t)\|^2 + \lambda \|\nabla p(t) \cdot \vec{n}\,(t)\|^2 \mathrm{d}t \tag{7}$$

  where $L(.)$ and $\vec{n}\,(.)$ are parameterizations of the boundary data over the domain boundary $\partial\Omega$ and λ is a positive weight. The first part ensures positional accuracy, while the second one controls the tangent planes along the boundary. The positive weight λ controls the relative influence of both terms.

- The third term is the approximate thin-plate energy, which is a standard fairness measure for surface fitting in geometry reconstruction [30]. It is used to regularize the fitting result.

The three terms are combined using non-negative weights, whose influence is discussed in the next section. In our experiments, we always scaled the geometry such that the bounding box fits into the unit cube and used standard weights $(10, 1, 1)$ for the three terms.

In addition to the objective function, the optimization also considers equality and inequality constraints. The first ones ensure the interpolation of the cutting loop at the vertices, while the second ones enforce the correct orientation of the boundary tangents, by making sure that the prescribed tangents along the cutting loop point towards the interior of the trimmed surface patch.

For the minimization of the non-linear objective function we need an initial solution, which we find by solving a simplified problem. When the first weight is set to zero, the resulting objective function becomes quadratic and can be solved rather easily. Note that the returned surface will be a regular trimmed surface patch, that interpolates the cutting loop on its boundary. In most situations (characterized by a relatively simple form of the solid and reasonable choice of the cutting loop) the interior of this surface patch will not penetrate the solid's boundary faces, thus the initial solution can be used as a valid cutting surface.

For more complicated cases, however, the initial surface may intersect the boundary of the solid. We then obtain a valid result by minimizing the objective function with a positive first weight, using an iterative method that starts with the initial solution. More precisely, we then apply the fmincon function of Matlab$^{\mathrm{TM}}$ to compute the control points by solving the resulting optimization problem. Examples are presented in the next section.

5. Results

This section consists of two parts. First we focus on the two-dimensional case, i.e., on the problem of finding a curve inside a given planar domain, connecting two points on the domain's boundary. We consider the same test cases as used by Nguyen et al. [21] and show that the new method solves these problems as well while significantly accelerating the computation. In the second part we present results for three–dimensional solids, which are subdivided into topological hexahedra using the algorithm of the isogeometric segmentation pipeline [15, 20]. In the considered examples, the original construction of the cutting surface fails, since the generated surfaces intersect the solid's boundary. We show that the method introduced in the present paper, which uses implicit guiding surfaces, resolves these problems.

5.1. Cutting curves

The first examples are shown in Figure 8. Three relatively simple domains are given, along with endpoints and desired tangent vectors of the cutting curve.

Clearly, our method produces results that are different from the curves obtained by the method in [21]. This is hardly surprising, as the methods rely on different approaches. Nevertheless, the resulting curves are also valid solutions as they fulfill all necessary criteria. Examining the runtimes of both algorithms (see Table 1) shows a significant advantage of the new method: Depending on the

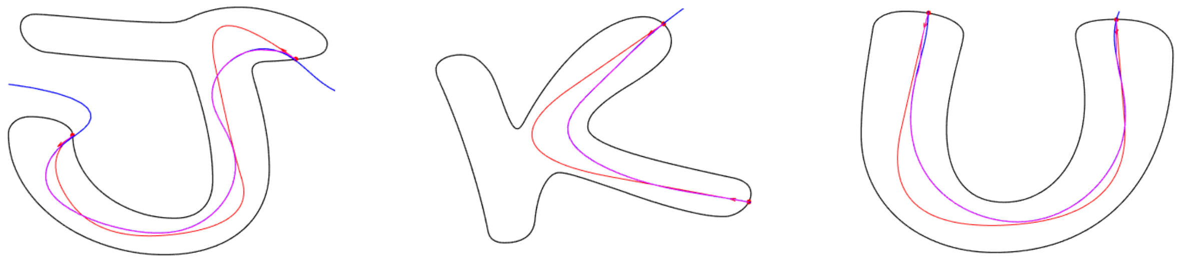


Figure 8: Guiding curves (blue) and cutting curves (purple) for the three character-shaped domains considered in [21]. The red curves have been generated by the method described in [21].

domain's complexity, we accelerate the method by about one or two orders of magnitude.

Example	U	J	K	Maze	Snake
method from [21]	337s	272s	3,309s	410s	7,657s
new method	9s	10s	11s	15s	27s
speedup	37.4	27.2	300.8	27.3	283.6

Table 1: Computing times for the examples shown in Figs. 8 and 9.

There are several reasons for this speedup:

1. Computing the implicit guiding curve involves evaluating the objective function (2) multiple times. The bottleneck of this computation is the regularity term (3), which consists of a double integral over a bivariate function. As mentioned earlier, one can exploit the tensor-product structure of the spline function f^* and rewrite the regularity term in the form $F_{\text{reg}}(\mathbf{c}) = \mathbf{c}^T \cdot M(u,v) \cdot \mathbf{c}$, where each entry of the matrix M is the product of univariate integrals over products of derivatives of B-spline basis functions. The matrix M can be easily computed in advance and every further evaluation of the objective function can be done by a matrix-vector multiplication.
2. In addition to the efficient computation of the objective function of (RDCP), there are only linear inequality constraints, in contrast to the more expensive computation of the penalty functions in [21].
3. For curves, the parameterization step is less complicated than for surfaces. There is no need to solve a constrained optimization problem as there are faster methods available. We use a marching algorithm to create an approximating polygon which is subsequently used to fit a parameterized spline curve.
4. We use standard settings for the degrees and knot vectors of the implicit guiding curve, as well as for the final parameterized curve and recieved valid results for all examples. Only for the snake example we got an improvement by manually chosing a different knot vector for the implicit guiding curve. However the vast speedup would allow to run several different settings and simply using the best result, while still reducing the runtime.

The next two test cases, shown in Figure 9, involve more complicated domains, which are again dealt with by both methods. Again we obtain valid results and a speed-up of the computing time. The new algorithm is about 27 times faster for the example of the maze and provides a tremendous improvement for the snake domain. While the original algorithm takes slightly over two hours, the new method returns the result after just 27 seconds.

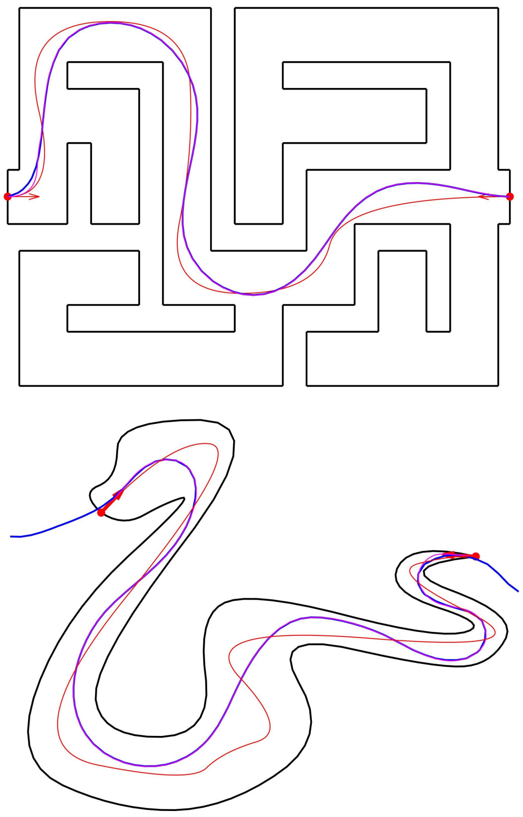


Figure 9: Guiding curves (blue) and cutting curves (purple) obtained for two more complicated domains: Maze (top) and snake (bottom). The red curves have been generated by the method described in [21].

We conclude this section by analyzing the influence of the optimization weights, which are used when computing the guiding curve. We consider an L-shaped domain and compute several implicit curves by considering various settings of the parameter values, see Fig. 10. More precisely, we minimize the objective function of RDCP with fixed values of λ_{int} and λ_{tan} and investigate the influence of the values of λ_{reg} and δ. For every curve, we calculate the minimum distance to the boundary of the domain and the maximum curvature of the resulting curve. It is clearly visible, that an increase of λ_{reg} results in a decrease of the curvature, while a higher value of δ increases the distance between the curve and the domain boundary.

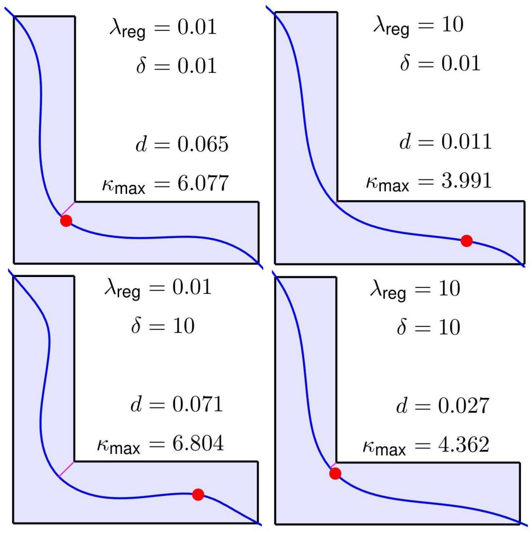


Figure 10: Experimental results showing the influence of the weights used for computing the guiding curve. The value d is the length of the purple line (distance between corner and curve) and $\kappa_{\max}$ is the maximum curvature of the curve segment. The red dot identifies the location of the curvature maximum.

5.2. *Cutting surfaces*

The first example (visualized in Fig. 11) shows a portion of an “iron maiden”: two parallel walls with spikes pointing towards each other. The height of the spikes exceeds half the distance between the two walls, so a planar surface would be penetrated by those spikes. The bottom picture shows the resulting wave-shaped guiding surface, which is fully contained in the empty space left between these spikes.

The second example shows the influence of the regularity weight in the parameterization step. We consider a cube shaped vase with a planar cutting loop, visualized in side view in Fig. 12. The computed implicit guiding surface is shown in red, and we obtain three different parameterized patches, depending on the weight for the fairness measure. As the weight λ_{reg} (weight of the third term) increases, the resulting surface becomes less curved and does not approximate the guiding surface anymore: Note that this may lead to an invalid cutting surface (see Fig. 12 right). In all our examples, we reached satisfying results when setting the weight to 1.

A more complicated version of the vase-shaped object, with non-planar faces and a non-planar cutting loop is shown in Fig. 1. The constructed implicit guiding surface is depicted in the center picture, next to the finally obtained parameterized trimmed surface patch. Note that

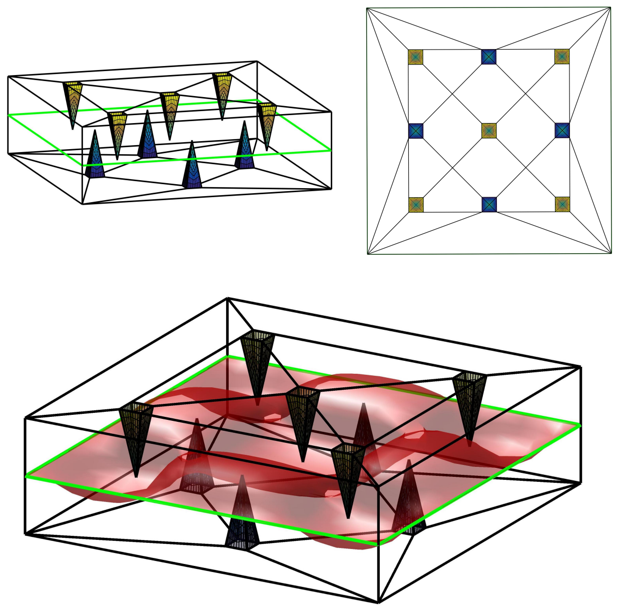
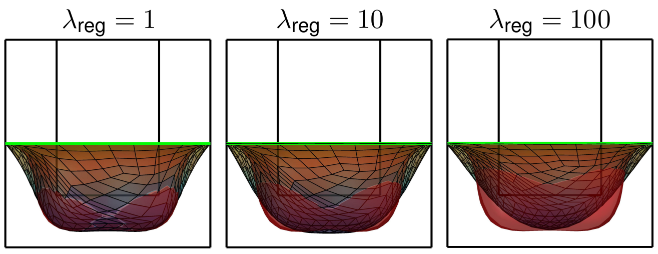


Figure 11: “Iron maiden” domain (top) and resulting guiding surface (bottom).



$\lambda_{\text{reg}} = 1$
$\lambda_{\text{reg}} = 10$
$\lambda_{\text{reg}} = 100$

Figure 12: Influence of the regularity weight in the parameterization step.

the parameter domain (right) is a five-sided planar polygon, which has been automatically generated.

The cutting loops considered in the previous examples were manually generated and did not introduce a meaningful segmentation. The intention of these loops was to show the properties of the cutting surface construction. In contrast to this, the remaining two examples use real cutting loops generated by the segmentation algorithm [15, 20].

The first example is a cube with two slots, which requires a non-trivial segmentation, shown in Fig. 13. We observed that this shape occurs frequently when applying the segmentation method [15, 20] to engineering objects with holes.

The original construction of the cutting surface fails, since two opposite faces of one of the solids in the left part of Fig. 13 intersect each other. This problem can now be resolved by using the novel cutting surface construction. The pictures on the right hand side show the same example, but with cutting surfaces obtained by using implicit guiding surfaces. We obtain a valid segmentation into three topological hexahedra.

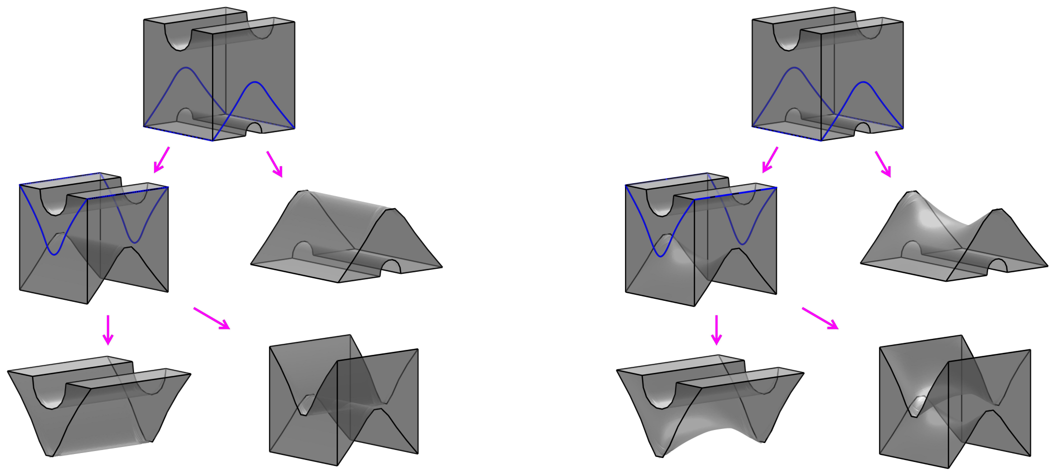


Figure 13: Segmentation of the bi-slotted cube with invalid (left) and valid (right) cutting surfaces.

The final example is a slightly simplified coupling piece of a garden hose, see Fig. 14. Due to the symmetry, it suffices to consider only a quarter of this object.

The result of the segmentation algorithm is presented in Fig. 15. Most segmentation steps are rather simple, as they simply cut slices off the solid, using planar cutting surfaces. There is, however, a more challenging step, which is marked in red. Here, the original construction of the cutting surface fails, due to the particular shape of the solid. The cutting surface penetrates the solid's boundary. This fact is visualized in Fig. 16, left. Using the implicit guiding surface leads to a different result, that resolves this problem, see Fig. 16, right. Summing up, we obtain a valid segmentation into eight topological hexahedra and one triangular prism.

6. Conclusion and outlook

An important part of the isogeometric segmentation pipeline [22] is the robust and reasonably fast construction of cutting surfaces. Such a cutting surface needs to fulfill all required properties, i.e. its boundary interpolates the associated cutting loop and its interior keeps a reasonable distance to the solid's boundary. We proposed a novel method, that combines techniques of implicit surface fitting and trimmed surface fitting in order to achieve this goal. It can also be used to solve the two-dimensional version of this problem, namely constructing a curve inside a planar domain, with specified endpoints and tangent vectors on the domain's boundary. Since the two-dimensional problem was already addressed in [21], we compared the results of both methods and observed a substantial improvement in the computational times. We showed that our method is capable of handling rather difficult domains and complicated cutting loops, as well as dealing with real segmentation problems. Future work may be devoted to a theoretical analysis of the proposed framework for cutting surface construction, concerning both the computational complexity and the existence and uniqueness of solutions.

A disadvantage of the isogeometric segmentation pipeline is the restriction to using a single cutting surface at each step. A midpoint subdivision technique could be used in order to segment a suitable solid into topological hexahedra, by constructing multiple surfaces, that meet in the solid's center. Often this would lead to a better segmentation since these surfaces would cut near-orthogonally through the solid's boundary surfaces, leading to a better distribution of angles. The surfaces used in midpoint subdivision need to fulfill similar properties as the cutting surface we constructed in this paper, e.g. interpolating given boundary curves without intersecting the solid's boundary in the interior. The method proposed in this article should therefore be generalized to this situation.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF), project NFN S117 "Geometry + Simulation", and by the European Commission, project "Multi-objective design optimization of fluid energy machines" (MOTOR), grant agreement no. 678727.

[1] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135–4195.

[2] J. A. Cottrell, T. J. Hughes, Y. Bazilevs, Isogeometric analysis: toward integration of CAD and FEA, John Wiley & Sons, 2009.

[3] T. Hughes, J. Oden, M. Papadrakakis, Isogeometric analysis: Progress and challenges, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1, editorial of a special issue.

[4] B. Jüttler, X. Qian, M. A. Scott, Isogeometric design and analysis, Computer-Aided Design 82 (2017) 1, editorial of a special issue.

[5] F. Massarwi, G. Elber, A B-spline based framework for volumetric object modeling, Computer-Aided Design 78 (2016) 36–47.

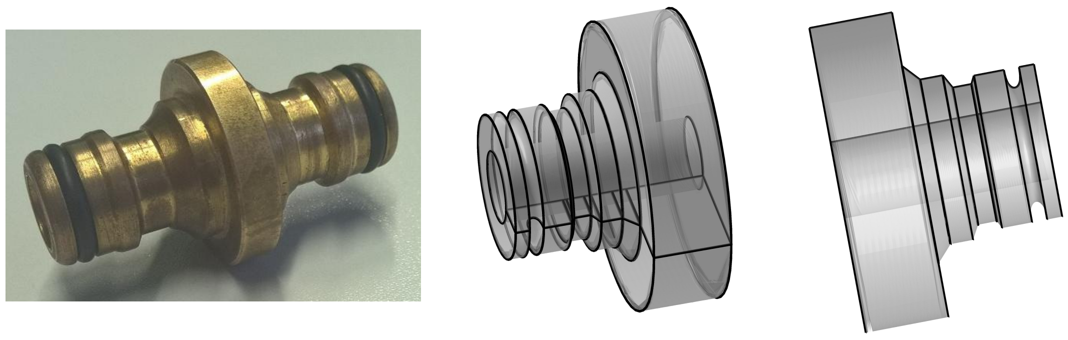


Figure 14: Solid object (center and right) representing a simplified coupling of a garden hose (left).

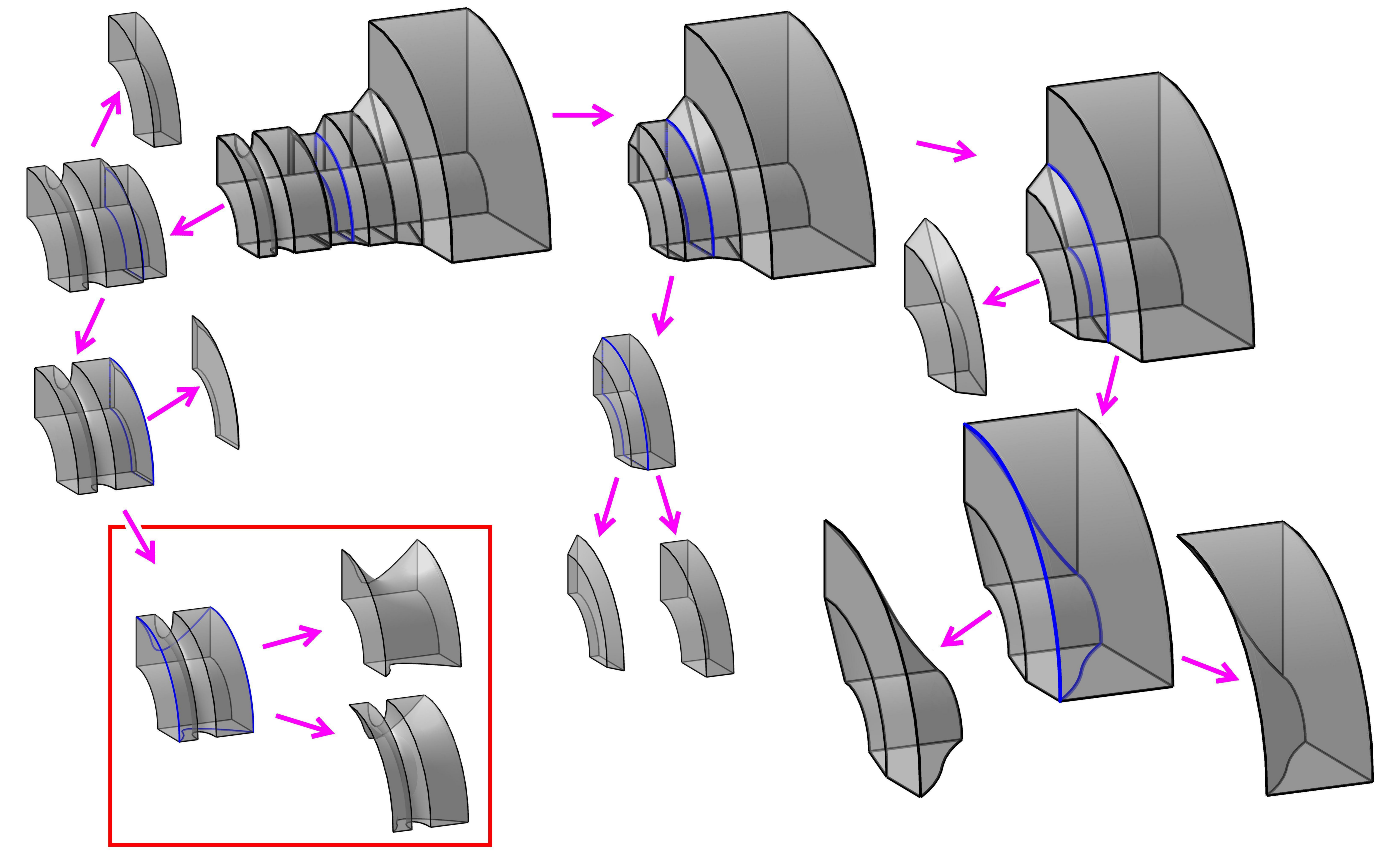


Figure 15: Automatic segmentation of a quarter of the hose coupling, using the algorithm described in [15, 20]. The blue lines represent the selected cutting loops. The three solids. which are marked in the lower-left part, lead to a problem with the construction of the cutting surface.

Figure 16: Failing (left) and valid (right) segmentation of a piece of the hose coupling example. Bottom row: The side view (with a viewing direction parallel to the chords of the circular arcs) shows the intersection (marked by the red arrows) between the opposite boundary faces obtained when using the initial version of the cutting surface (left), which is no longer there when applying the new algorithm (right).

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

[7] T. Martin, E. Cohen, M. Kirby, Volumetric parameterization and trivariate b-spline fitting using harmonic functions, in: Proceedings of the 2008 ACM symposium on Solid and physical modeling, ACM, 2008, pp. 269–280.

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

[9] W. Wang, Y. Zhang, L. Liu, T. J. Hughes, Trivariate solid t-spline construction from boundary triangulations with arbitrary genus topology, Computer-Aided Design 45 (2) (2013) 351 – 360, solid and Physical Modeling 2012.

[10] Y. Zhang, C. Bajaj, Adaptive and quality quadrilateral/hexahedral meshing from volumetric data., Comput Methods Appl Mech Engrg 195 (9) (2006) 942–60.

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

[12] F. Buchegger, B. Jüttler, Planar multi-patch domain parameterization via patch adjacency graphs, Computer-Aided Design 82 (2017) 2–12.

[13] F. Buchegger, B. Jüttler, A. Mantzaflaris, THB–splines: The truncated basis for hierarchical splines, Applied Mathematics and Computation 272 (1) (2015) 159–172.

[14] M. A. Scott, D. C. Thomas, E. J. Evans, Isogeometric spline forests, Computer Methods in Applied Mechanics and Engineering 269 (2014) 222 – 264.

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

[16] K. Wang, X. Li, B. Li, H. Xu, H. Qin, Restricted trivariate polycube splines for volumetric data modeling, IEEE Trans. on Visualization and Computer Graphics 18 (5) (2012) 703–716.

[17] L. Liu, Y. Zhang, T. J. R. Hughes, M. A. Scott, T. W. Sederberg, Volumetric T-spline construction using boolean operations, in: J. Sarrate, M. Staten (Eds.), Proceedings of the 22nd International Meshing Roundtable, Springer International Publishing, 2014, pp. 405–424.

[18] H. A. Akhras, T. Elguedj, A. Gravouil, M. Rochette, Isogeometric analysis-suitable trivariate NURBS models from standard B-Rep models, Computer Methods in Applied Mechanics and Engineering 307 (2016) 256 – 274.

[19] H. Al Akhras, T. Elguedj, A. Gravouil, M. Rochette, Towards an automatic isogeometric analysis suitable trivariate models generation-application to geometric parametric analysis, Computer Methods in Applied Mechanics and Engineering 316 (2017) 623–645.

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

[21] D.-M. Nguyen, M. Pauley, B. Jüttler, Isogeometric segmentation: Construction of auxiliary curves., Computer-Aided Design 70 (2016) 89 – 99.

[22] M. Pauley, et al., The isogeometric segmentation pipeline, in: B. Jüttler, B. Simeon (Eds.), Isogeometric Analysis and Applications 2014, Springer, 2014, pp. 51–72.

[23] C. Bajaj, I. Ihm, J. Warren, Higher-order interpolation and least-squares approximation using implicit algebraic surfaces, ACM Transactions on Graphics (TOG) 12 (4) (1993) 327–347.

[24] V. Pratt, Direct least-squares fitting of algebraic surfaces, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 145–152.

[25] G. Taubin, Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (11) (1991) 1115–1138.

[26] B. Jüttler, A. Felis, Least–squares fitting of algebraic spline surfaces, Advances in Computational Mathematics 17 (2002) 135–152.

[27] J. Wang, Z. Yang, L. Jin, J. Deng, F. Chen, Parallel and adaptive surface reconstruction based on implicit pht-splines, Computer Aided Geometric Design 28 (8) (2011) 463–474.

[28] M. Pan, W. Tong, F. Chen, Compact implicit surface reconstruction via low-rank tensor approximation, CAD Computer Aided Design 78 (2016) 158–167.

[29] T. Varady, R. Martin, Reverse engineering, in: G. Farin, J. Hoschek, M.-S. Kim (Eds.), Handbook of Computer Aided Geometric Design, Elsevier, 2002, Ch. 26, pp. 651–682.

[30] V. Weiss, L. Andor, G. Renner, T. Várady, Advanced surface fitting techniques, Computer Aided Geometric Design 19 (1) (2002) 19–42.

[31] E. Wurm, B. Jüttler, M.-S. Kim, Approximate rational parameterization of implicitly defined surfaces, in: R. Martin, H. Bez, M. Sabin (Eds.), Mathematics of Surfaces XI, Vol. 3604 of Lecture Notes in Computer Science, Springer, 2005, pp. 434–447.

[32] D.-M. Nguyen, M. Pauley, private communication (2013).