Heterogeneous Parametric Trivariate Fillets

Ramy Masalha[a], Emiliano Cirillo[a,b], Gershon Elber[a]

[a] *Computer Science Department, Technion Israel Institute of Technology, Haifa, Israel*
[b] *École Polytechnique Fédérale de Lausanne, Institute of Mathematics, Lausanne, Switzerland*

Abstract

Blending and filleting are well established operations in solid modeling and computer-aided geometric design. The creation of a transition surface which smoothly connects the boundary surfaces of two (or more) objects has been extensively investigated. In this work, we introduce several algorithms for the construction of, possibly heterogeneous, trivariate fillets, that support smooth filleting operations between pairs of, possibly heterogeneous, input trivariates. Several construction methods are introduced that employ functional composition algorithms as well as introduce a *half Volumetric Boolean sum* operation. A volumetric fillet, consisting of one or more tensor product trivariate(s), is fitted to the boundary surfaces of the input. The result smoothly blends between the two inputs, both geometrically and material-wise (properties of arbitrary dimension). The application of encoding heterogeneous material information into the constructed fillet is discussed and examples of all proposed algorithms are presented.

Keywords: Parametric Geometry; Blending and filleting; Heterogeneity; Volumetric representations (V-reps); Functional composition

1. Background and Common Definitions

Given two objects, *blending* refers to the creation of a transition surface which smoothly connects boundary surfaces. Blending and filleting are used for rounding sharp edges or corners, and they are of high importance in the field of computer-aided design. Due to mechanical, aesthetic or other reasons, it is typically desired to replace sharp edges and corners by smooth faces. The boundary surfaces being connected are called *primary surfaces* and the boundary curves of the fillet on the primary blending surface are called *rail curves* (See Figure 1). The blending surface is usually required to meet with the primary surfaces with at least G^1 continuity. That is, to share a tangent plane with the primary surfaces along the rail curves.

While some use the terms *blending* and *filleting* interchangeably, in this paper, we use *blending* to refer to the process of constructing a blending surface, and we use *filleting* to refer to the process of constructing a (set of) trivariate(s) that fill the space between the primary surfaces and their blending surface. The (set of) trivariate(s) that fill this space will be called a fillet.

Consider the two trivariates, $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{R}^k, k \geq 3$. $\mathcal{T}_i,\ i \in \{1, 2\}$ can include, in addition to geometric information in $\mathbb{R}^3$, additional (scalar, vector, tensor, etc.) properties for $k > 3$, such as material properties. Let $S_1 \in \mathcal{T}_1$ and $S_2 \in \mathcal{T}_2$ be two boundary surfaces in $\mathbb{R}^k$ of the trivariates that intersect along curve $C = S_1 \cap S_2$. In this work, we seek to create a volumetric fillet between S_1 and S_2 along C. Here again, S_1 and S_2 will also be denoted as the *primary surfaces*. Further, we denote by $c_{\mathcal{I}}^{S_i}$ the intersection curve, C, in the parametric space of $S_i, i \in \{1, 2\}$. I.e. $C = S_i(c_{\mathcal{I}}^{S_i})$. In the ensuing discussion, small letters will denote geometry in a parametric space, and capital letters will be used to refer to geometry in the Euclidean space. In this work, we assume all parametric forms are sufficiently differentiable (typically at least C^1) and regular (their Jacobian never vanish).

Because the algorithms we introduce only make use of the boundary surfaces of the input trivariates (possibly with their heterogeneous properties, in $\mathbb{R}^k, k \geq 3$), hence after, we will consider the input to be S_1 and S_2, rather than $\mathcal{T}_1$ and $\mathcal{T}_2$. The following subsections review background material and introduce functionalities that will be exploited in the proposed filleting algorithms.

*Ramy Masalha
Email addresses: ramymassalha@gmail.com (Ramy Masalha), emiliano.cirillo@epfl.ch (Emiliano Cirillo), gershon@cs.technion.ac.il (Gershon Elber)

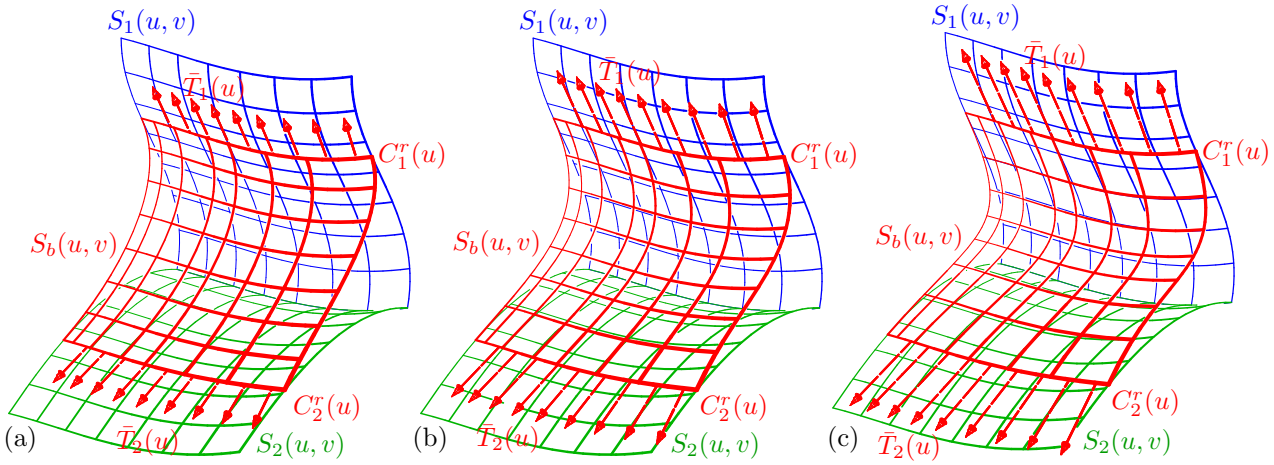


Figure 1: This figure shows three blending surfaces (red) built using the cubic Hermite interpolation scheme, for the two surfaces, $S_1(u,v)$ and $S_2(u,v)$. The interpolated rail curves are $C_1^r(u)$ and $C_2^r(u)$. In each picture, a different magnitude of tangent field is used. The red arrows indicate the tangent fields of $\bar{T}_i(u)$. $\bar{T}_i(u)$ denotes normalized and scaled tangent fields, along the rail curves. See also Algorithm 1

.

1.1. Cubic Hermite Blending Surface

A common approach to construct a blending surface uses the cubic Hermite interpolation scheme Cohen et al. (2001), which offers a C^1 continuity. Given two rail curves, $C_1^r(u)$ and $C_2^r(u)$, and two tangent fields along them, $T_1(u)$ and $T_2(u)$, the following surface interpolates the given rail curves and reconstructs the given tangent fields along them:

$$S_b(u,v) = C_1^r(u)h_{00}(v) + C_2^r(u)h_{01}(v) + T_1(u)h_{10}(v) + T_2(u)h_{11}(v), \tag{1}$$

where $h_{ij}(v)$ are the cubic Hermite basis functions. Figure 1 shows three examples of a cubic Hermite blending surface with different magnitudes of tangent fields, T_i, along the boundaries, while the rail curves and the directions of the tangent fields are all kept the same. This blending scheme will be extended and employed here, toward heterogeneous trivariate fillets.

1.2. Constructing the Rail Curves

While the rail curves (and their associated tangent fields) can also be provided by the end user, herein we offer an additional way to construct them. Let S_1 and S_2 be the two given surfaces, and let C be their intersection curve. The default construction scheme we offer to use as rail curves is using a Euclidean offset approximation of C, lying on S_1 and S_2. More precisely, let C_i^r be a curve on S_i located at Euclidean distance d from C, the intersection curve. Then, C_i^r, which is denoted the Euclidean offset curve of C on S_i, can be used as a rail curve in the construction of the blending surface. Figure 2 shows an example of two resulting Euclidean offset rail curves, when the primary surfaces are a cylinder and a general Bézier surface. See Elber and Kim (2020) for more on this Euclidean offset-approximation computation on surfaces. In the rest of this work, we assume the rail curves, c_i^r, are provided.

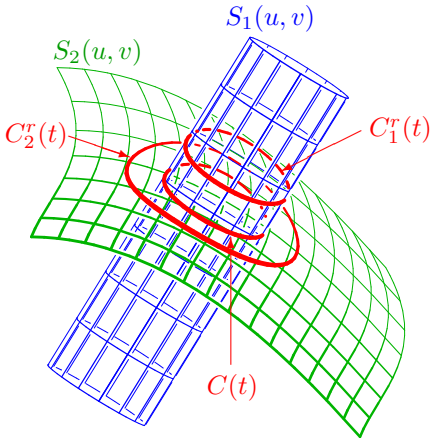


Figure 2: Euclidean offset curves, $C_i^r(t)$, of the intersection curve $C(t)$ of a cylinder $S_1(u,v)$ and a Bézier surface $S_2(u,v)$, on the surfaces S_1 and S_2.

1.3. Constructing the Tangent Fields

The tangent fields are necessary to ensure G^1 continuity. Given the rail curves, we seek univariate tangent fields, along the rail curves, in the tangent fields of the primary surfaces, that are also orthogonal to the direction of the rail curves. These tangent fields are hence computed, for a rail curve $C_i^r(t) = S_i(c_i^r(t))$, as

$$T_i(t) = C_i'^r(t) \times (N_i(c_i^r(t)),$$

where $N_i(u,v)$ is the normal field of $S_i = S_i(u,v)$ as $N_i = \frac{\partial Si}{\partial u} \times \frac{\partial Si}{\partial v}$.

At each point, the resulting tangent field is simultaneously orthogonal to the normal of S_i (which means it lies in the corresponding tangent plane), and to the tangent field of C_i^r, $C_i'^r$. Note we assume regularity of C_i^r and S_i, and hence T_i is well defined.

We also seek to give the user control over the magnitude of the tangent fields. Hence, we approximate the normalized tangent fields, only to scale these fields by a user-specified parameter. Because normalization of a rational field is typically not rational, the normalization of the tangent field, is approximated, following Kim and Elber (1997). Again, Figure 1 shows three examples of a blending surface with different magnitudes of tangent fields along the given rail curves, and where $\bar{T}$ denotes the normalized fields.

1.4. Constructing a Blending Surface

In summary, we construct a blending surface using the cubic Hermite interpolation scheme, with two positional constraints (the rail curves), as explained in Section 1.2, and two tangential constraints, as explained in Section 1.3. Algorithm 1 summarizes the process of creating a blending surface.

Algorithm 1: Computing a blending surface.

Input:
$S_1(u_1, v_1)$, $S_2(u_2, v_2)$ - primary surfaces;
$c_i^r(t)$ - rail curve in the parameter space of S_i;
τ - control over the magnitudes of the tangents;
Output:
B - a cubic Hermite blending surface;

BlendingSrf$(S_1, S_2, c_1^r, c_2^r, \tau)$**:**
  for $j \leftarrow 1$ ***to*** 2 **do**
    $C_i^r(t) = S_i(c_i^r(t))$;
    $N_i(u, v) = \frac{\partial S_i(u,v)}{\partial u} \times \frac{\partial S_i(u,v)}{\partial v}$;
    $T_i(t) = C_i'^r(t) \times N_i(c_i^r(t))$;
    $\bar{T}_i(t) = \frac{T_i(t)}{||T_i(t)||}$; `// An approximation`
  $S_b(u, v) = C_1^r(u)h_{00}(v) + C_2^r(u)h_{01}(v) + \tau\bar{T}_1(u)h_{10}(v) + \tau\bar{T}_2(u)h_{11}(v)$;
  return B;

1.5. Surface-Surface Functional Composition

We have already seen in this work use-cases of curve-surface functional composition. For example, when the normal field of S_i was restricted to the univariate rail curve $c_i^r(t)$, as $N_i(c_i^r(t))$. However, the computation of surface-surface composition places some additional challenges that must be discussed.

Unlike the cases of blends of surfaces, here, for volumetric, trivariate, fillets, we will also be required to represent general, non iso-parametric, sub-region of the primary surfaces. Specifically, *we seek to precisely represent and interpolate the surface sub-region of S_i between the intersection curve and the rail curve*. We seek a tensor product representation for this sub-region, $\hat{S}_i(s, t) \subset S_i(u, v)$, in order to employ it in the trivariate fillet construction. Toward this end, we will mostly exploit surface-surface composition Elber (1992); DeRose et al. (1993), as follows:

$$\hat{S}_i(s, t) = S_i(Rld(c_i^r(t), c_{\mathcal{I}}^{S_i}(t))), \tag{2}$$

where $Rld(\cdot, \cdot)$ denotes a ruled surface between the two given parametric curves:

$$Rld(c_i^r(t), c_{\mathcal{I}}^{S_i}(t))) = (1 - v)c_i^r(t) + vc_{\mathcal{I}}^{S_i}(t), v \in [0, 1]. \tag{3}$$

See Figure 3. Note S_i and hence $\hat{S}_i$ can reside in $\mathbb{R}^k, k \geq 3$ and contain geometric prescriptions but also additional (material) properties.

A difficulty arises in obtaining a representation of the required sub-regions by functional composition when a sub-region of surface $\hat{s}_i$ that participate in the fillet crosses a knot line in the domain of S_i, an arrangement we denote *B-spline surfaces in general position*. The composition, $S_i(\hat{s}_i)$, of a tensor product spline function S_i with another spline function $\hat{s}_i$ is no longer a tensor product spline if $\hat{s}_i$ crosses a knot line in the domain of S_i, in a general (non iso-parametric) way. Hence, and after discussing previous work in Section 2, in Section 3, we assume that the primary surfaces are polynomial/Bézier (as it is clear that the composition of two polynomial surfaces is yet another polynomial surface) and construct fillets consisting of tensor product trivariates. Then, we will (partially) deal with the difficulty of crossing knot lines, in several ways, in Section 4. In Section 5, the exploitation of trimmed trivariates in building volumetric fillets is discussed, as an alternative, and in Section 6, we deal with the problem of filleting over trivariates with C^1 discontinuities. In Section 7, several approaches to blend and encode property values, in $\mathbb{R}^k$, into the fillet, are considered. Then, in Section 8, some results are presented, and finally, in Section 9, we conclude.

2. Related Work

Methods for constructing blending surfaces have been researched intensively in the last decades, and we sample this rich body. Bajaj and Ihm (1992) and Gao and Li (2002) utilize the Hermite interpolation scheme to construct blending surfaces, and Sanglikar et al. (1990) constructs a mathematical model for a blending surface generated by rolling a ball between the two primary surfaces. Varady et al. (1989) discusses methods to solve topological problems, such as vertex crossing, that may be faced when constructing a blending surface and Lin et al. (2014); Cheng (2003) represent a blending surface as a set of curves. See Vida et al. (1994) for a survey of blending techniques.

Several algebraic methods have been proposed to construct blending surfaces. These algebraic methods include, for example, solving a set of ordinary differential equations (ODEs) or partial differential equations (PDEs), as in You et al. (2018, 2014), solving an optimization problem, as in Farouki et al. (2019), and utilizing Grobner basis methods, as in ru Wu and shi Zhou (2000).

The blending surface is typically required to meet with the primary surfaces with G^1 continuity, but some works construct a blending surface that satisfies further requirements or have different features. Belkhatir and Zidna (2009) provides a method for constructing a blending surface which is G^n continuous to the primary surfaces, You et al. (2018) deals with the construction of blending surface with curvature continuity, Farouki et al. (2019) requires the constructed blending surface to optimize different measures, such as maximal deviation and uniformity of parametric speed, and Elber (2005) proposes a method for the construction of shaped blending surfaces for decoration purposes.

The problem of constructing a blending surface has been tackled within various contexts. Zhou and W.-H (2012); Bizzarri et al. (2017); Chen and Tang (2003) deal with constructing a blending surface between multiple surfaces. Li (2004); Hsu (2006) discuss methods for the blending of implicit surfaces. You et al. (2019) deals with the blending of time-dependent parametric surfaces and Hoffmann and Hopcroft (1988) discusses the problem of blending two surfaces in projective space.

All the above work has focused on blending surfaces, and to the extent of our knowledge, no research has been conducted on the construction of trivariate fillets. In this work, we propose several algorithms toward this end, of constructing volumetric trivariate fillets.

We are also interested in previous work on the blending and encodement of heterogeneous property functions. Biswas et al. (2004), for example, propose several distance-based methods for the purpose of material blending, such as convex combination of the materials at different regions, and Samanta and Koc (2004); and Samanta and Koc (2005) introduce methods for the blending of material represented by free-form functions and solve an optimization problem to determine material variation. More recently, Ameta and Witherell (2019) dealt with the encoding of properties into *transition regions* by associating each transition region with a *transition function*, and setting the property value at a given point in a specific transition region to a combination of several transition functions. Additionally, Cirillo and Elber (2020) tackled the encodement of properties over V-rep models by a *feature preserving* blending scheme.

Some other works offer methods to support Boolean operations on heterogeneous objects. Kumar and Dutta (1998) represent heterogeneous material by r_m*-objects*, which are collections of r_m*-sets*, and define Boolean operations over them. Dutta and Shin (2001); Shin (2002) propose material blending approaches of two heterogeneous primitive sets involved in a Boolean operation, based on a convex combination of the two materials, and Kou et al. (2006) utilize non-manifold cellular representation and heterogeneous features tree representation to model complex heterogeneous objects. Finally, see Kou and Tan (2007); Li et al. (2020) for extensive reviews of the literature on heterogeneous object modeling.

Compared to previous works, this work extents the thoroughly investigated problem of the construction of blending surface to the volumetric context of trivariate filleting, and discusses the application of heterogeneity encodement methods over the constructed fillets. The filleting algorithms we introduce support varying sets of parametric surface types, and operate under different assumptions, while the resulting fillets (smoothly) blend the two inputs, both geometrically and material-wise. The practicality of our methods is demonstrated by various examples, such as 3D printing of the constructed heterogeneous fillet, in Section 8.

3. Trivariate Fillet Construction between Two Bézier Surfaces

Following the difficulties in performing the required surface-surface composition operations, as portrayed in Section 1.5, in this section, we assume the input primary surfaces, S_1 and S_2, are polynomial Béziers. Recall we denoted by $Rld(\cdot,\cdot)$ the ruled surface between two curves. Similarly, for any two surfaces $S_a(u,v)$ and $S_b(u,v)$ defined over $[0,1]^2$, we denote by $Rld(S_a,S_b)$ the ruled volume between S_a and S_b:

$$Rld(S_a,S_b)(u,v,w) = (1-w)S_a(u,v) + wS_b(u,v), w \in [0,1]. \tag{4}$$

The following sections introduce our first trivariate fillet construction algorithms between two, possibly heterogeneous, Bézier boundary surfaces.

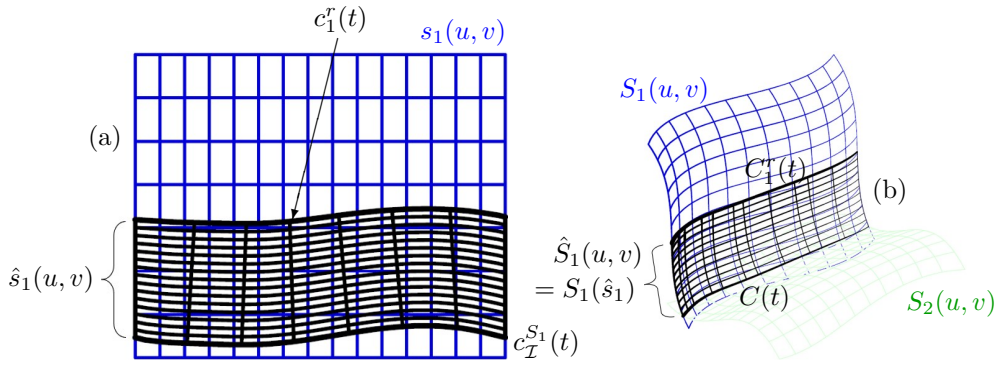


Figure 3: A sub-region, $\hat{S}_1(u,v) \in S_1(u,v)$, to be filleted. $\hat{S}_1$ is bounded by the intersection curve, $C(t) = S_1(c_{\mathcal{I}}^{S_i})$, and the rail curve, $C_1^r(t) = S_1(c_1^r)$. (a) shows the preimage of $\hat{S}_1(u,v)$, in the parametric domain of S_1, as s_1. (b) shows S_1 and $\hat{S}_1$ in the Euclidean space, along with a faded version of S_2, the second surface to be filleted along intersection curve $C(t)$.

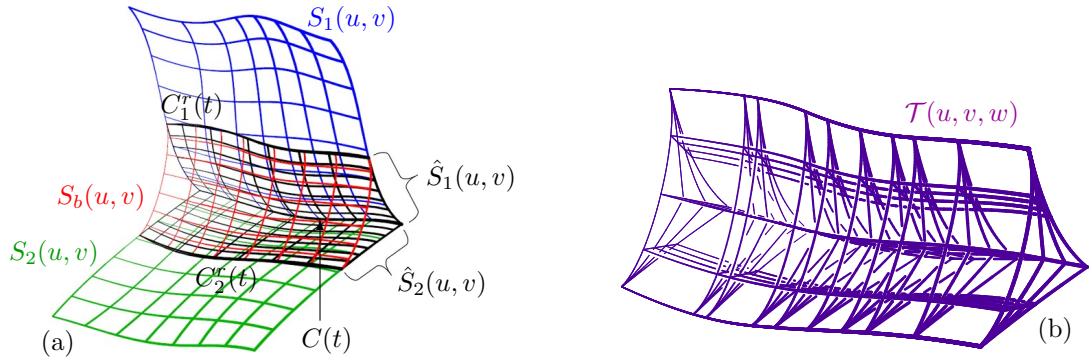


Figure 4: In (a), the two primary Bézier surfaces, $S_1(u,v)$ and $S_2(u,v)$ are shown, along with the sub-regions of S_1 and S_2 that are to be filleted, as $\hat{S}_1(u,v)$ and $\hat{S}_2(u,v)$, between the intersection curve, $C(t)$, and the rail curves, C_i^r $i \in \{1,2\}$. $\hat{S}_i(u,v)$ are computed using surface-surface functional composition. Also shown in (a) is the cubic Hermite blending surface, $S_b(u,v)$, between the rail curves. (b) presents the corresponding computed fillet trivariate, $\mathcal{T}(u,v,w)$, created by Algorithm 2, as a ruled volume between $\hat{S}_{12} = \hat{S}_1 \cup \hat{S}_2$ and S_b.

3.1. Constructing a Trivariate Fillet by a Ruled Volume

Given $\hat{S}_i(u,v),\ u,v \in [0,1],\ i \in \{1,2\}$, as in Equation (2), denote by $\hat{S}_{12} = \hat{S}_1 \cup \hat{S}_2$ the merged along C surface:

$$\hat{S}_{12} = \hat{S}_1 \cup \hat{S}_2 = \begin{cases} \hat{S}_1(u,2v), & 0 \le v \le 0.5, \\ \hat{S}_2(u,2(v-0.5)), & 0.5 < v \le 1. \end{cases} \tag{5}$$

The fillet we aim to create is bounded in between $\hat{S}_{12}$ and the blended surface, $S_b(u,v)$, as in Equation (1) and Algorithm 1. Hence, a ruled volume between $\hat{S}_{12}$ and the blended surface can serve as the desired fillet:

$$\mathcal{T}(u,v,w) = Rld(\hat{S}_{12}(u,v), S_b(u,v)),$$

after matching the domain and the functional spaces of S_{12} and S_b, via degree raising and refinements.

Note both $\hat{S}_{12}(u,v)$ and $S_b(u,v)$ and hence $\mathcal{T}(u,v,w)$ can reside in $\mathbb{R}^k, k \ge 3$ and may contain geometric prescriptions but also additional properties. Algorithm 2 summarizes the process of constructing a fillet trivariate using this formulation, as a ruled volume. Figure 4 shows a resulting trivariate fillet of using Algorithm 2, over two Bézier surfaces.

3.2. Constructing a Trivariate Fillet by a Volumetric Boolean Sum

The fillet construction scheme presented in Section 3.1 has two complete surface faces of the filet trivariate vanishing into curves, along the rail curves. Such singularities must be difficult in some applications and hence herein, we consider an alternative.

We utilize the volumetric Boolean sum operation Elber et al. (2012), which computes a tensor product trivariate representation of a space enclosed by six surfaces sharing boundaries in a cube-like topology. However, the desired fillet space, enclosed by $\hat{S}_1$, $\hat{S}_2$ and S_b, does not present a cube topology. Hence, the fillet construction algorithm introduced in this section, starts by subdividing the fillet space into three sub-spaces, each having a cube-like topology (see Figure 5).

Algorithm 2: computing a filleting trivariate by a ruled volume, for Bézier primary surfaces.

Input:
$S_1(u_1, v_1)$, $S_2(u_2, v_2)$ - the primary Bézier surfaces;
$c_i^r(t)$ - rail curve in the parameter space of S_i;
$c_{\mathcal{I}}^{S_i}(t)$ - intersection curve in the parameter space of S_i;
$C(t)$ - intersection curve in Euclidean space;
τ - control over the magnitudes of the tangents;
Output:
T, a filleting trivariate;

BzrTrivarFillet$(S_1, S_2, c_1^r, c_2^r, c_{\mathcal{I}}^{S_1}, c_{\mathcal{I}}^{S_2}, C, \tau)$**:**
  for $i \leftarrow 1$ ***to*** 2 **do**
    $\hat{S}_i(u, v) = S_i(Rld(c_{\mathcal{I}}^{S_i}, c_i^r))$; `// Figure 3`
  $\hat{S}_{12}(u, v) = \hat{S}_1 \cup \hat{S}_2$; `// Equation (5)`
  $S_b(u, v) = BlendingSrf(S_1, S_2, c_1^r, c_2^r, \tau)$; `// Algorithm 1`
  $\mathcal{T}(u, v, w) = Rld(\hat{S}_{12}(u, v), S_b(u, v))$; `// Ruled volume between matched domains/function spaces`
  return $\mathcal{T}$;

The division method goes as follows. Let $C_c(t)$ be a curve in the interior of the fillet space, and let $C_1(t)$, $C_2(t)$ and $C_b(t)$ be three curves on $\hat{S}_1(u, v)$, $\hat{S}_2(u, v)$ and $S_b(u, v)$, respectively. The three ruled surfaces, $Rld(C_c, C_1)$, $Rld(C_c, C_2)$ and $Rld(C_c, C_b)$, divide the fillet space into three sub-spaces having each a cube topology, and the volumetric Boolean sum operation is employed to construct a separate fillet trivariate for each sub-space.

While almost any curve, $C_c(t)$, inside the fillet space, and almost any three curves $C_1(t) \subset \hat{S}_1(u, v)$, $C_2(t) \subset \hat{S}_2(u, v)$ and $C_b(t) \subset S_b(u, v)$, are suitable to perform the above division process, we seek to choose curves that will (approximately) satisfy the following two requirements:

1. For symmetry purposes, the curve C_c is desired to be (approximately) equidistant from the three surfaces $\hat{S}_1$, $\hat{S}_2$ and S_b.
2. Aiming at stable Jacobian values in all three trivariates, we strive for the three constructed fillet trivariates to be (approximately) equiangular, near $C_c(t)$.

Appendix A details one possible approach to the computation of this arrangement while an accurate computation of the curve C_c, so it is precisely equidistant from $\hat{S}_1$, $\hat{S}_2$ and $\hat{S}_b$, can be sought by defining C_c to be the trisector of the three surfaces $\hat{S}_1$, $\hat{S}_2$ and S_b and letting C_1, C_2 and C_b be the foot supporting curves of trisector C_c on $\hat{S}_1$, $\hat{S}_2$, and $\hat{S}_b$, respectively. The trisector of three surfaces can be computed via six constraints with seven unknowns, as shown in Bartoň et al. (2010). However, herein, and for simplicity of the computation, we follow the approximation approach explained in Appendix A.

Then, and having these supporting curves, the three trivariates are constructed following Algorithm 3, that together form the whole fillet. See also the cross section sketch of the associated surface in Algorithm 3, that are used in each of the three trivariates. Note that while the volumetric Boolean sum operation is typically executed over six boundary surfaces forming a cube topology, Algorithm 3 employs the volumetric Boolean sum over only four surfaces forming a sleeve. Two capping surfaces are automatically generated from the four provided surfaces, from the two open ends of the sleeve - Two sets of four boundary curves of the four provided surfaces. In this work, the generation of the capping surfaces has been done using the Boolean sum method Cohen et al. (2001). Figure 5 (c) shows one result of executing Algorithm 3 on two primary Bézier surfaces.

4. Trivariate Fillet Construction between Two B-spline Surfaces

The filleting Algorithms introduced in the previous chapter assume the two primary surfaces, S_1 and S_2, are Bézier surfaces. In this section, we discuss how it is possible to extend these algorithms to support B-spline based geometry under certain assumptions (that are explained in Section 4.2). The algorithms described in this section divide the input B-spline surfaces, S_1 and S_2, into pairs of Bézier sub-surfaces. Then, for each such pair of Bézier surfaces, we compute their trivariate fillet, so that their union provides the whole fillet of the input B-spline geometry (see Figure 6). Section 4.1 describes how the subdivision locations of S_1 and S_2 are computed, and Section 4.2 introduces the subdivision and fillet construction algorithm, and explains the assumptions we make on S_1 and S_2 to ensure that the algorithm can be applied. As not all cases of volumetric fillets for input tensor product B-spline geometry can be

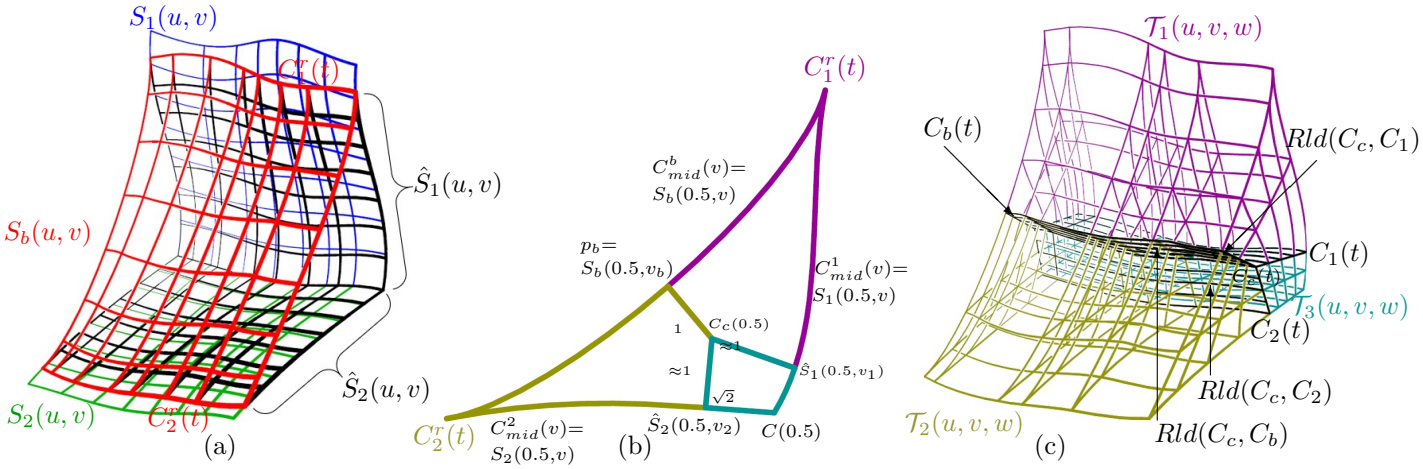


Figure 5: In (a), the two primary Bézier surfaces, $S_1(u,v)$ and $S_2(u,v)$ along with their blending surface, $S_b(u,v)$ are shown. In (b), The division of the fillet space into three cube-like topologies is presented in a cross section. Finally, (c) presents the three reconstructed trivariates, $\mathcal{T}_i$, in this arrangement, forming together the whole fillet. See also Algorithm 3.

Algorithm 3: Computing filleting trivariates via volumetric Boolean sum operations.

Input:

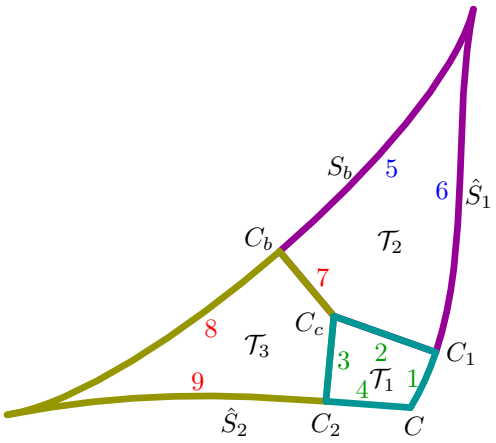
$S_1(u_1,v_1)$, $S_2(u_2,v_2)$ - the primary Bézier surfaces;
$c_i^r(t)$ - rail curve in the parametric space of S_i;
C_c - the interior curve to the fillet space;
$C_i = S_i(c_i)$ - The projection curve of C_c on S_i $i \in \{1,2\}$;
$C_b = S_b(c_b)$ - The projection curve of C_c on S_b;
$c_{\mathcal{I}}^{S_i}(t)$ - Intersection curve in the parametric space of S_i;

Output:

$\mathcal{T}_1, \mathcal{T}_2$ and $\mathcal{T}_3$ - Three trivariates forming the whole fillet;

BzrTrivarFilletBoolSum$(S_1, S_2, c_1^r, c_2^r, C_1, C_2, C_b, c_{\mathcal{I}}^{S_1}, c_{\mathcal{I}}^{S_2})$**:**

$\mathcal{T}_1(u,v,w) = VBS($ // Volumetric Boolean sum
  $S_1(Rld(c_{\mathcal{I}}^{S_1}, c_1))$, // Cross section,curve 1
  $Rld(C_1, C_c)$, // Cross section,curve 2
  $Rld(C_c, C_2)$, // Cross section,curve 3
  $S_2(Rld(c_2, c_{\mathcal{I}}^{S_2}))$; // Cross section,curve 4
$\mathcal{T}_2(u,v,w) = VBS($ // Volumetric Boolean sum
  $S_1(Rld(c_1, c_1^r))$, // Cross section,curve 5
  $S_b(Rld(c_1^r, c_b))$, // Cross section,curve 6
  $Rld(C_b, C_c)$, // Cross section,curve 7
  $Rld(C_c, C_1)$; // Cross section,curve 2
$\mathcal{T}_3(u,v,w) = VBS($ // Volumetric Boolean sum
  $S_b(Rld(c_b, c_2^r))$, // Cross section,curve 8
  $S_2(Rld(c_2^r, c_2))$, // Cross section,curve 9
  $Rld(C_2, C_c)$, // Cross section,curve 3
  $Rld(C_c, C_b)$; // Cross section,curve 7
return $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$;



reduced to a fillet trivariate between a set of tensor product Bézier pairs, in Section 4.3, an alternative to the fillet construction that circumvents the surface-surface composition altogether is considered, at the cost of being only an approximation.

4.1. Computing the Subdivision locations in the B-spline surfaces

The values along which S_1 and S_2 will be subdivided are computed as follows. As a first step, S_1 and S_2 are subdivided along all of their interior knots, leaving only Bézier patches. Let S_i be one of the input surfaces $\{S_1, S_2\}$ and let S_j be the other surface. For each u-knot, u_{i0}, and each v-knot, v_{i0}, of S_i, let $\mathcal{C}_i^{u_0}(v) = S_i(u_{i0}, v)$ and

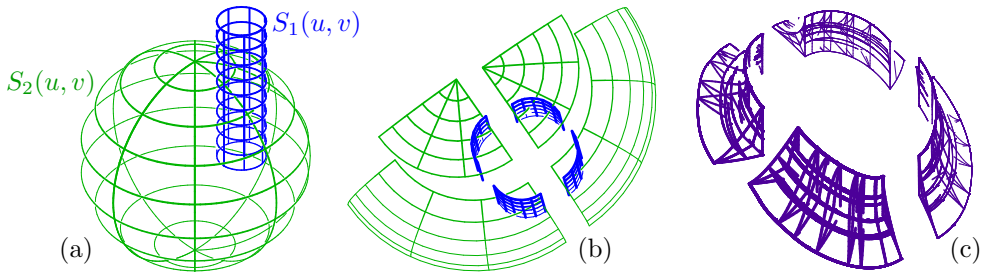


Figure 6: (a) A cylinder, $S_1(u,v)$, intersects with a sphere, $S_2(u,v)$. (b) The sphere and the cylinder are subdivided into pairs of matched Bézier sub-surfaces. (c) A fillet trivariate is then created between each pair of Bézier sub-surfaces, as shown in (b). The union of the four fillet trivariates between sub-surfaces form the whole desired fillet of the original cylinder and sphere.

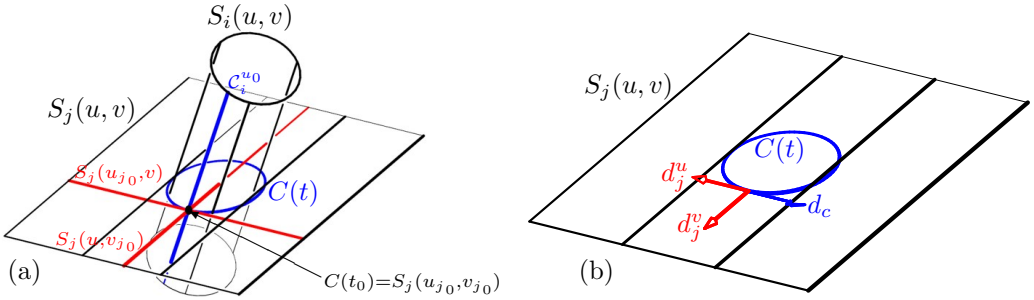


Figure 7: To create a trivariate volumetric fillet between a planar surface, S_j, and a B-spline cylinder, S_i, the cylinder (and the plane) are subdivided into pairs of matched Bézier patches. In (a), the cylinder, $S_i(u,v)$, is subdivided along the knot line, $\mathcal{C}_i^{u_0}(v) = S_i(u_0,v)$. Then, we find the intersection points, $\mathcal{I} = \{S_j(u_{j_0},v_{j_0})\}$, between $\mathcal{C}_i^{u_0}(v)$ and the intersection curve, $C(t)$, only to subdivide S_j at each such intersections, along one of the iso-parametric curves: either $S_j(u_{j_0},v)$ or $S_j(u,v_{j_0})$. To decide between these two subdivision options, in (b), we compute the unit u-tangent, d_j^u, and unit v-tangent, d_j^v, of S_j, at $S_j(u_{j_0},v_{j_0})$, and choose to subdivide along the direction that is the most orthogonal to the unit tangent, d_c, of $C(t)$, at $C(t_0)$. Herein, S_j will be divided along the v direction.

$\mathcal{C}_i^{v_0}(u) = S_i(u,v_{i0})$ be the corresponding iso-parametric curves of S_i. Now consider the intersection locations, $\mathcal{I}$, of S_j and curves $\mathcal{C}_i^{u_0}(v)$ or $\mathcal{C}_i^{v_0}(u)$, if any. S_j will be further subdivided at all $\mathcal{I}$, in either u or v. We now describe how to compute the subdivision values of S_j corresponding to the set $\mathcal{I}$ for $S_j \cap \mathcal{C}_i^{u_0}$, while the subdivision values corresponding to the set of intersection points $S_j \cap \mathcal{C}_i^{v_0}$ are similarly computed. See Figure 7.

Let $c_{\mathcal{I}}^{S_i}(t) = (u^{S_i}(t), v^{S_i}(t))$ be the intersection curve in S_i's parametric space and let $C(t) = S_i(u^{S_i}(t), v^{S_i}(t))$ be its Euclidean image. Given u-knot u_{i0}, using a univariate solver (e. g. Machchhar and Elber (2018)), we solve the following constraint in t:

$$u^{S_i}(t) = u_{i0}, \tag{6}$$

to find all the value(s), $\{t_0\}$, for which the u-coordinate of the intersection curve, $c_{\mathcal{I}}^{S_i}(t_0)$, is equal to u_{i0}. Note that t_0 is a solution of Equation (6) if and only if the intersection curve, $C(t) = S_i(u^{S_i}(t), v^{S_i}(t))$, intersects with the iso-parametric curve, $\mathcal{C}_i^{u_0}(v) = S_i(u_{i0},v)$, at the point $C(t_0)$. That is, $\mathcal{C}_i^{u_0}$ intersects with the other surface, S_j, at $C(t_0)$. For each such t_0, let $(u_{j_0},v_{j_0}) = c_{\mathcal{I}}^{S_j}(t_0)$ be the intersection point in S_j's parametric space.

Having an intersection point identified, we either subdivide S_j at the u-value, u_{j_0}, or at the v-value, v_{j_0}. Ideally, we aim to subdivide S_j in a direction orthogonal to the intersection curve, at the intersection point $C(t_0)$, because such orthogonality would make the fillet more fair-looking and the computation more numerically stable (better Jacobians). Let d_j^u and d_j^v be the unit u-tangent and unit v-tangent of S_j at the point (u_{j_0},v_{j_0}) (see Figure 7(b)), respectively, and let d_c be the unit tangent of the intersection curve, C, at $C(t_0) = S(u_{j_0},v_{j_0})$. If $|\langle d_j^u, d_c\rangle| < |\langle d_j^v, d_c\rangle|$, the u-value, u_{j_0}, is added to the subdivision values of S_j. Otherwise, the v-value, v_{j_0}, is added. Algorithm 4 summarizes the computation process of these subdivision values.

4.2. The Fillet Construction

Recall that the fillet we seek to create is bounded by sub-regions $\hat{S}_i = S_i(Rld(c_{\mathcal{I}}^{S_i}, c_i^r))$ (recall Figure 3). The subdivision of S_1 and S_2 along the parametric values computed by Algorithm 4, also imposes a similar subdivision on $\hat{S}_1$ and $\hat{S}_2$, the sub-regions between the rail curves and the intersection curve. The subdivided patches of $\hat{S}_1$ and $\hat{S}_2$ will be denoted $\hat{S}_{1,1},\ldots,\hat{S}_{1,n}$ and $\hat{S}_{2,1},\ldots,\hat{S}_{2,m}$, respectively. The fillet construction algorithm we introduce in this section operates under the assumption that $n = m$, and it constructs a separate trivariate fillet between each ($\hat{S}_{1,i}$, $\hat{S}_{2,i}$) pair. However, in general, we don't have a tensor product representation of $\hat{S}_1$ and $\hat{S}_2$, to ensure that $n = m$ so that $\{\hat{S}_{1,i}\}$ and $\{\hat{S}_{2,i}\}$ can be properly matched. Hence, the following condition is verified: on each subdivision step, and for each iso-parametric curve, $\mathcal{C}$, corresponding to one of S_i's subdivision values ($i \in \{1,2\}$), the number

Algorithm 4: Computing subdivision values for B-spline primary surfaces.

Input:
$S_1(u_1, v_1)$, $S_2(u_2, v_2)$ - B-spline primary surfaces;
C - intersection curve in Euclidean space;
$c_{\mathcal{I}}^{S_1}$ - intersection curve in the parametric space of S_1;
$c_{\mathcal{I}}^{S_2}$ - intersection curve in the parametric space of S_2;
Output:
L_i^u - subdivision values of S_i in the u parameter direction;
L_i^v - subdivision values of S_i in the v parameter direction;

GetSubdivVals(S_1, S_2, C)**:**
  $L_1^u,\ L_1^v$ = interior knots of S_1 in u and v;
  $L_2^u,\ L_2^v$ = interior knots of S_2 in u and v;
  for $i \leftarrow 1$ ***to*** 2 **do**
    $j = (i == 1)?\ 2 : 1$;
    foreach *u-knot,* u_{i0}*, of* S_i **do**
      $\mathcal{C}_i^{u_0}(v) = S_i(u_{i0}, v)$;
      foreach $C(t_0) \in \mathcal{C}_{\mathcal{I}}^{u_0} \cap C$ **do**
        AddSubdivVal$(S_j, c_{\mathcal{I}}^{S_j}, t_0, L_j^u, L_j^v)$;
    foreach *v-knot,* v_{i0}*, of* S_i **do**
      $\mathcal{C}_i^{v_0}(u) = S_i(u, v_{i0})$;
      foreach $C(t_0) \in \mathcal{C}_i^{v_0} \cap C$ **do**
        AddSubdivVal$(S_j, c_{\mathcal{I}}^{S_j}, t_0, L_j^u, L_j^v)$;
    return ${L_1}^u, {L_1}^v, {L_2}^u, {L_2}^v$;

`// Auxiliary function`

Function `AddSubdivVal`(S, c, t_0, L^u, L^v)**:**
  $(u, v) = c(t_0)$;
  $C = S(c)$;
  $d_u = \frac{\partial S(u,v)}{\partial u} / \| \frac{\partial S(u,v)}{\partial u} \|$;
  $d_v = \frac{\partial S(u,v)}{\partial v} / \| \frac{\partial S(u,v)}{\partial v} \|$;
  $d_c = \frac{dC(t)}{dt} / \| \frac{dC(t)}{dt} \|$;
  if $|\langle d_j^u, d_c \rangle| < |\langle d_j^v, d_c \rangle|$ **then**
    $L^u = L^u \cup \{u\}$;
  else
    $L^v = L^v \cup \{v\}$;

of times $\mathcal{C}$ crosses the intersection curve, C, must be equal to the number of times $\mathcal{C}$ crosses the rail curve, C_i^r (see Figure 8). The following lemma shows that if this condition is satisfied, $n = m$ and we have a proper matching:

Lemma 1. *Let* ${L_1}^u, {L_1}^v, {L_2}^u, {L_2}^v$ *be the subdivision values obtained from Algorithm 4. If for each u-subdivision step,* $u_0 \in {L_i}^u$ $(i \in \{1, 2\})$*, the iso-parametric curve,* $\mathcal{C}_i(v) = S_i(u_0, v)$*, intersects with the intersection curve,* $C(t)$*, the same number of times it intersects with the rail curve,* $C_i^r(t)$ *(i.e.* $|\mathcal{C}_i(v) \cap C(t)| = |\mathcal{C}_i(v) \cap C_i^r(t)|$*, where* $|\cdot|$ *denotes the cardinality of the set), and a similar condition holds for each v-subdivision step,* $v_0 \in {L_i}^v$*, then after subdividing* S_1 *and* S_2 *along all of the subdivision values in* ${L_1}^u, {L_1}^v, {L_2}^u$ *and* ${L_2}^v$*,* $\hat{S}_1$ *and* $\hat{S}_2$ *will be split into the same number of sub-surfaces.*

Proof: Let $i \in \{1, 2\}$ and let j be the index corresponding to the other surface. Because each isoparametric curve we subdivide along, crosses both $C(t)$ and $C_i^r(t)$ the same number of times, the number of sub-surfaces of $\hat{S}_i$ is uniquely determined by the number of intersection points between $C(t)$ and the isoparametric curves we subdivide S_i along. Since for each subdivision isoparametric curve, $\mathcal{C}_i$, of S_i, and for each intersection point, $C(t_0) \in \mathcal{C}_i \cap C$, Algorithm 4 matches a subdivision isoparametric curve, $\mathcal{C}_j$ of S_j, that also crosses C at $C(t_0)$, we deduce that the number of intersection points between C and the subdivision isoparametric curve of S_i is equal to the number of intersection points between C and the subdivision isoparametric curves of S_j. Thus, the lemma holds.
■

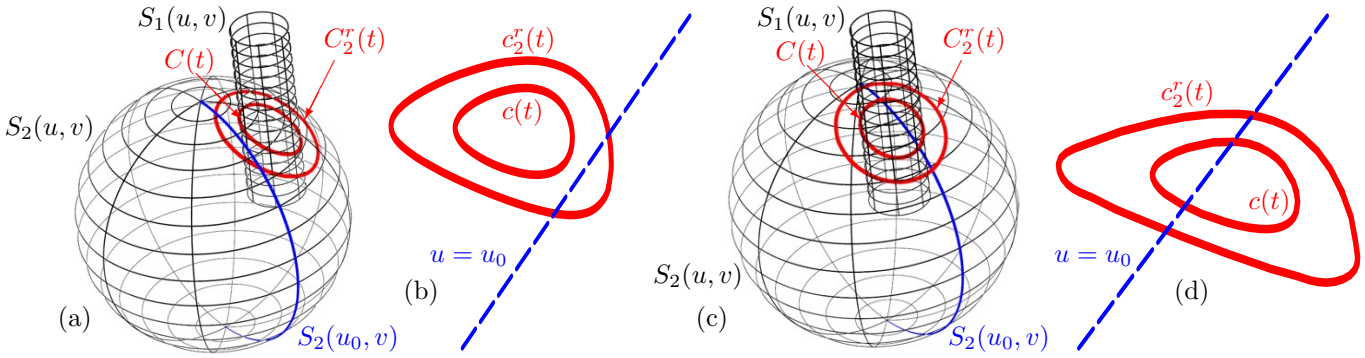


Figure 8: During the fillet construction process between the cylinder, $S_1(u,v)$, and the sphere $S_2(u,v)$, S_2 is subdivided along an iso-parametric curve, $\mathcal{C}(v) = S_2(u_0,v)$. If the number of times $\mathcal{C}(v)$ crosses the intersection curve, $C(t)$, is equal to the number of times $\mathcal{C}(v)$ crosses the rail curve, $C_2^r(t)$, then the fillet construction continues successfully. Otherwise, the fillet construction fails and is aborted. In (a), the fillet construction fails, because $\mathcal{C}(v)$ crosses $C_2^r(t)$ twice while it does not cross $C(t)$. (b) presents (part of) the parametric space of the sphere, S_2, in (a). In (c) and (d) the cylinder from (a) and (b) is translated a bit so $\mathcal{C}(v)$ crosses both $C(t)$ and $C_2^r(t)$ twice, and the fillet construction can be completed successfully.

Before introducing the fillet construction algorithm for B-spline geometry, we define the following data structure:

Definition 4.1. *A* FilletBoundary *is a data structure holding the following fields:*

- i *- either 1 or 2. Specifies whether this boundary corresponds to a sub-surface region of* S_1 *or* S_2
- $\tilde{S}$ *- a sub-surface region of* S_i*.*
- $\tilde{c}$ *- a sub-curve of the intersection curve,* $c_{\mathcal{I}}^{S_i}$*, in the parametric space of* S_i*.*
- $\tilde{c}_r$ *- a sub-curve of the rail curve,* c_i^r*, in the parametric space of* S_i*.*

Further, we assume each instance, $\mathcal{FB}$, of the FilletBoundary data structure supports the following functionalities:

- $\mathcal{FB}$.Split($\mathcal{C}$): Receives an iso-parametric curve, $\mathcal{C}$, of S_i as an input. If $\mathcal{C}$ is also an isoparametric curve of $\tilde{S}$, the function splits $\tilde{S}$ along $\mathcal{C}$, splits $\tilde{c}$ and $\tilde{c}_r$ on each of their intersection points with $\mathcal{C}$ and returns a list of FilletBoundaries, corresponding to the sub-surfaces and sub-curves resulting from splitting the original FilletBoundary. If $\mathcal{C}$ is not an isoparametric curve of $\tilde{S}$, the function does nothing and returns a list containing one item - the original FilletBoundary.
- $\mathcal{FB}$.Verify($\mathcal{C}$): Receives an iso-parametric curve $\mathcal{C}$, of S_i, returns true if the number of times $\mathcal{C}$ crosses $\tilde{C}_r = S_i(\tilde{c}_r)$ is equal to the number of times $\mathcal{C}$ crosses $\tilde{C} = S_i(\tilde{c})$, and returns false otherwise.

Algorithm 5 utilizes the above data structure to construct a list of trivariates, that together form the whole fillet between the B-spline primary surfaces, if possible. The algorithm splits the two primary surfaces into Bézier sub-surfaces and constructs a separate fillet between each pair of Bézier sub-surfaces that share the same sub-curve, $\tilde{C}$, of the intersection curve, C. Note that we assume the input primary surfaces, S_i, the intersection curve, C, and the rail curves, C_i^r are all continuous. Hence, after calling $\mathcal{FB}$.Split($\mathcal{C}$) (Line 16 of Algorithm 5) to split the input geometry, the resulting pairs of sub-curves and sub-surfaces will connect continuously on $\mathcal{C}$. This also ensures that each two adjacent fillet trivariates constructed by Algorithm 5 will connect continuously. As mentioned above, the algorithm operates under the assumption that each iso-parametric curve corresponding to one of the subdivision values computed by Algorithm 4 crosses the intersection curve, C, the same number of times it crosses the rail curve, C_i^r, throughout the subdivision process. This condition is checked in Line 14. Again, recall Figure 6 for an example of a fillet construction between two B-spline surfaces.

4.3. Volumetric Filleting Construction by Surface Fitting

Recall that the filleting algorithms introduced so far depend on surface-surface composition to compute a tensor product representation of the fillet boundary surfaces, $\hat{S}_i \subset S_i$, under the assumption that the two composed surface regions are Bézier surfaces. In this section, we propose an alternative to the computation of a tensor product representation of (an approximation of) the required sub-surfaces. The (approximated) fillet construction method introduced in this section supports both Bézier and B-spline surfaces in general positions (i.e. possible crossing of knot lines).

Algorithm 5: Fillet construction for B-spline primary surfaces.

Input:
$S_1(u_1, v_1)$, $S_2(u_2, v_2)$ - the primary Bézier surfaces;
$c_{\mathcal{I}}^{S_i}(t)$ - intersection curve in parametric space of S_i;
$c_i^r(t)$ - rail curve in the parametric space of S_i;
$C(t)$ - intersection curve in Euclidean space;
Output:
L_f - a list of trivariates, forming the whole fillet;

BspTrivarFillets$(S_1, S_2, c_{\mathcal{I}}^{S_1}, c_{\mathcal{I}}^{S_2}, c_1^r, c_2^r, C)$**:**
  FilletBoundary
    $B1 = (1, S_1, c_{\mathcal{I}}^{S_1}, c_1^r)$,
    $B2 = (2, S_2, c_{\mathcal{I}}^{S_2}, c_2^r)$; `// Definition 4.1.`
  ${L_1}^u, {L_1}^v, {L_2}^u, {L_2}^v = GetSubdivVals(S_1, S_2, C)$; `// Algorithm 4.`
  $L_{B1} = \{B1\}$; `// A list containing B1`
  $L_{B2} = \{B2\}$; `// A list containing B2`
  $L_f = \emptyset$;
  `// Subdivide at all values computed by Algorithm 4.`
  for $i \leftarrow 1$ ***to*** 2 **do**
    foreach *SubdivisionVal*, $s \in {L_i}^u \cup {L_i}^v$ **do**
      $L(t)$ = iso-parametric curve on S_i corresponding to s;
      foreach $B \in L_{Bi}$ **do**
        $L_{Bi}.pop(B)$;
        if $!B.Verify(L)$ **then**
          `// Cannot fit a fillet!`
          return $\emptyset$;
        $L_{Bi} = L_{Bi} \cup B.Split(L)$;
  foreach $B_1, B_2 \in L_{B1}, L_{B2}$ *sharing the same sub curve,* $C' = B_i.S'(B_i.c')$ **do**
    $L_f = L_f \cup BzrFillet(B_1.S', B_2.S', B_1.c_r', B_2.c_r', C', \tau)$; `// E.g. Algorithm 2`
  return L_f;

Let $c_1, c_2 \in s_i$ be two curves in the domain of S_i, $i \in \{1, 2\}$, and recall

$$s_r(u, v) = Rld(c_1, c_2)(u, v) = (1 - v)c_1(u) + vc_2(u)$$

is a ruled surface between c_1 and c_2. To compute an approximation of the sub-surface $\hat{S}_i = S_i(s_r)$, we uniformly sample a set of points, $\{(u_j, v_j)\} \subset s_r$, in S_r's domain, only to compute a B-spline surface that least-square-fits (Cohen et al. (2001)) the set of their Euclidean images, $\{S_i(s_r(u_j, v_j))\}$. The control points of the boundaries of the resulting open-end B-spline surface can be fixed to ensure an accurate interpolation of the boundary curves, and with sampling that ensures the Schoenberg Whitney interpolation conditions Cohen et al. (2001). For example, the surface-surface composition in Line 3 of Algorithm 2 can be replaced by this least-square-fitting method to compute an approximation of $\hat{S}_i$, thus extending previous Algorithm 2 to also support B-spline surfaces in general position. See Figures 9 and 15 for examples of a fillet created following the above least-square-fitting method.

5. Volumetric Filleting using Trimmed Trivariates

In this section, we utilize the V-rep framework, introduced in Massarwi and Elber (2016), to construct a volumetric fillet. In the V-rep framework, a volumetric space can be represented as the sub-domain of one or more B-spline tensor product trivariates, that are delineated by trimming (and possibly trimmed) surfaces in Euclidean space.

To construct a V-rep fillet between S_1 and S_2, we initially compute a tensor product trivariate containing the filleted space, only to trim it by the fillet's boundary surfaces, $\hat{S}_1$, $\hat{S}_2$ and S_b (recall Algorithm 1) obtaining the desired V-rep fillet. The following operation, which we denote *half Volumetric Boolean Sum*, is introduced to compute a tensor product trivariate containing the fillet space. Following the notations of the previous sections, and as before let

$$\hat{S}_i(u, v) = S_i(Rld(c_{\mathcal{I}}^{S_i}, c_i^r)) = S_i((1 - v)c_{\mathcal{I}}^{S_i}(u) + vc_i^r(u)), \quad i \in \{1, 2\},$$

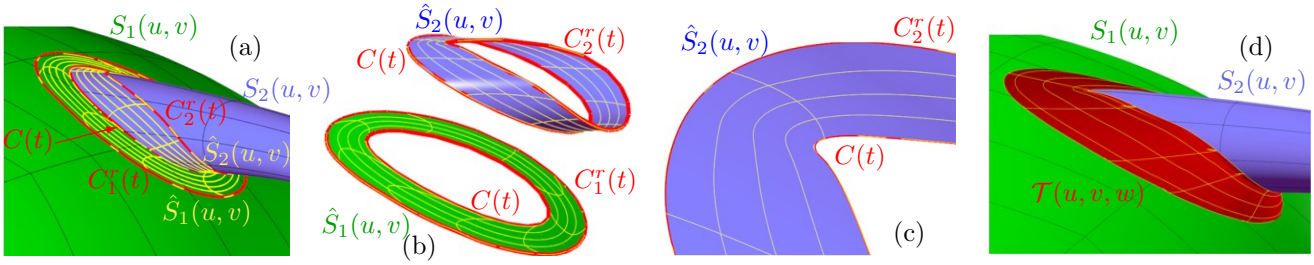


Figure 9: In (a), two surfaces, $S_1(u,v)$ and $S_2(u,v)$ are shown, along with the subregions of S_1 and S_2 that are to be filleted, $\hat{S}_1$ and $\hat{S}_2$, between the intersection curve, $C(t)$, and the rail curves, $C_i^r(t)$. (b) presents (an approximation of) the two subregions, $\hat{S}_1$ and $\hat{S}_2$, between the intersection curve, $C(t)$, and the rail curves, $C_i^r(t)$, computed by the least-square fitting method described in Section 4.3. (c) is a zoom in view of $\hat{S}_1(u,v)$. (d) presents the computed fillet trivariate, $\mathcal{T}(u,v,w)$, resulting from replacing Line 3 in Algorithm 2 by a least-square-fitting of $\hat{S}_i$. $\hat{S}_i$ were sampled at (12×105) locations in the domain, for the least-squares fit, and have (7×35), for $\hat{S}_1$, and (3×35), for $\hat{S}_2$, control points. See also Figure 15 and Table 1.

be the sub-regions on S_i that serve as the boundary surfaces of the fillet, and let $C(u) = S_i(c_{\mathcal{I}}^{S_i}(t)) = \hat{S}_i(u,0)$ be the intersection curve of S_1 and S_2 in Euclidean space.

Definition 5.1. *The* half Volumetric Boolean Sum *of $\hat{S}_1$ and $\hat{S}_2$, is the trivariate:*

$$\mathcal{T}_{hbs}(u,v,w) = \hat{S}_1(u,v) + \hat{S}_2(u,w) - C(u).$$

The name *half Volumetric Boolean Sum* stems from the fact that this constructor is a subset of the regular Volumetric Boolean Sum Elber et al. (2012). Figures 10 (b) and 11 (b) show two examples of a half Boolean sum trivariate, $\mathcal{T}_{hbs}(u,v,w)$. The half Volumetric Boolean sum operation can also be viewed as an extension of the Volumetric Boolean sum operation, for the case in which only two adjacent surfaces are employed.

Lemma 2. *The boundaries of the half Volumetric Boolean sum, $\mathcal{T}_{hbs}$, interpolate the two input surfaces: $\mathcal{T}_{hbs}(u,v,0) = \hat{S}_1(u,v)$, and $\mathcal{T}_{hbs}(u,0,w) = \hat{S}_2(u,w)$.*

Proof: We prove that $\mathcal{T}_{hbs}(u,v,0) = \hat{S}_1(u,v)$, while the other case can be similarly proven:

$$\begin{aligned} \mathcal{T}_{hbs}(u,v,0) &= \hat{S}_1(u,v) + \hat{S}_2(u,0) - C(u) \\ &= \hat{S}_1(u,v) + C(u) - C(u) \\ &= \hat{S}_1(u,v). \end{aligned} \tag{7}$$

■

The trimming surfaces of the half Volumetric Boolean sum trivariate, $\mathcal{T}_{hbs}$, that delineate the fillet zone are the boundary surfaces, $\hat{S}_1$ and $\hat{S}_2$, of $\mathcal{T}_{hbs}$ as well as the blending surface S_b. S_b is computed by Algorithm 1, while $\hat{S}_1$ and $\hat{S}_2$ can be derived via either surface-surface composition or least-squares fitting.

If the intersection curve, $C(t)$, is closed, the fillet forms a loop and is closed as well, and $\hat{S}_1$, $\hat{S}_2$ and S_b constitute the whole boundary of the fillet. However, if $C(t)$ is open, two additional capping, triangular like, surfaces, S_i^{cap}, are derived (see Figure 10), from the open boundary curves of surfaces $\hat{S}_1, \hat{S}_2$, and S_b. Algorithm 6 summarizes the above process of constructing a V-rep based fillet using a trimmed trivariate. Regardless of $C(t)$ being closed or open, the set of trimming surfaces, $\mathcal{L}$, is forming a closed 2-manifold to properly trim $\mathcal{T}_{hbs}$. Figures 10 and 11 show two examples of the output of Algorithm 6. In Figure 10, the intersection curve, $C(t)$, is open, and therefore two caps should be computed, while in Figure 11, the intersection curve, $C(t)$, is closed, and no caps are constructed.

6. Trivariate Filleting over Multiple Surfaces, Possibly over C^1 Discontinuities

Consider the problem of constructing a fillet trivariate between two trivariates, $\mathcal{T}_1(u,v,w)$ and $\mathcal{T}_2(u,v,w)$, that have C^1 discontinuities along the filleting zone - along their intersection curve, C. See Figure 12. Let $S_{1,i}$ and $S_{2,i}$, $i \in \{0,\ldots,5\}$, be the six boundary surfaces of tensor product trivariates $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Let $\partial\mathcal{T}_k = \bigcup_{i=0}^{5} S_{k,i}$, $k \in \{1,2\}$, and $C = \partial\mathcal{T}_1 \cap \partial\mathcal{T}_2$, and for each $0 \le i \le j \le 5$, denote $C_{i,j} = S_{1,i} \cap S_{2,j}$. That is, intersection curve $C = \bigcup_{0 \le i \le j \le 5} C_{i,j}$. Hence, we now assume that $C(t)$ is closed but only piecewise-C^1. Further, let

$$\mathcal{C}[\cdot] = \{C_{i_0,j_0},\ C_{i_1,j_1},\ C_{i_2,j_2},\ldots\}$$

Algorithm 6: Computing filleting trivariate by V-rep method.

Input:
$\hat{S}_1(u,v)$, $\hat{S}_2(u,w)$ - fillet boundary sub-regions on S_1 and S_2;
$S_b(u,v)$ - blending surface;
$C(u)$ - intersection curve in Euclidean space;
Output:
V, a V-rep model representing the fillet;

VrepFillet(S_1, S_2, S_b, C)**:**
  $\mathcal{T}_{hbs}(u,v,w) = \hat{S}_1(u,v) + \hat{S}_2(u,w) - C(u)$;
  $\mathcal{L} = \{\hat{S}_1, \hat{S}_2, S_b\}$;
  if $C(t)$ *is open* **then**
    $S_c^1 = \mathcal{T}_{hbs}(0,v,w)$;
    $S_c^2 = \mathcal{T}_{hbs}(1,v,w)$;
    for $i \leftarrow 1$ ***to*** 2 **do**
      $C^i_{S_b}(t) = S_c^i \cap S_b$;
      $C^i_u(t) = S_c^i(t,0)$;
      $C^i_v(t) = S_c^i(0,t)$;
      S_i^{Cap} = A trimming surface to $\mathcal{T}_{hbs}$ from surface S_c^i, using closed trimming loop $\{C_t^i, C_u^i, C_v^i\}$;
    $\mathcal{L} = \mathcal{L} \cup \{S_1^{Cap}, S_2^{Cap}\}$
  V = V-rep model resulting from trimming $\mathcal{T}_{hbs}$ by set of surfaces $\mathcal{L}$;

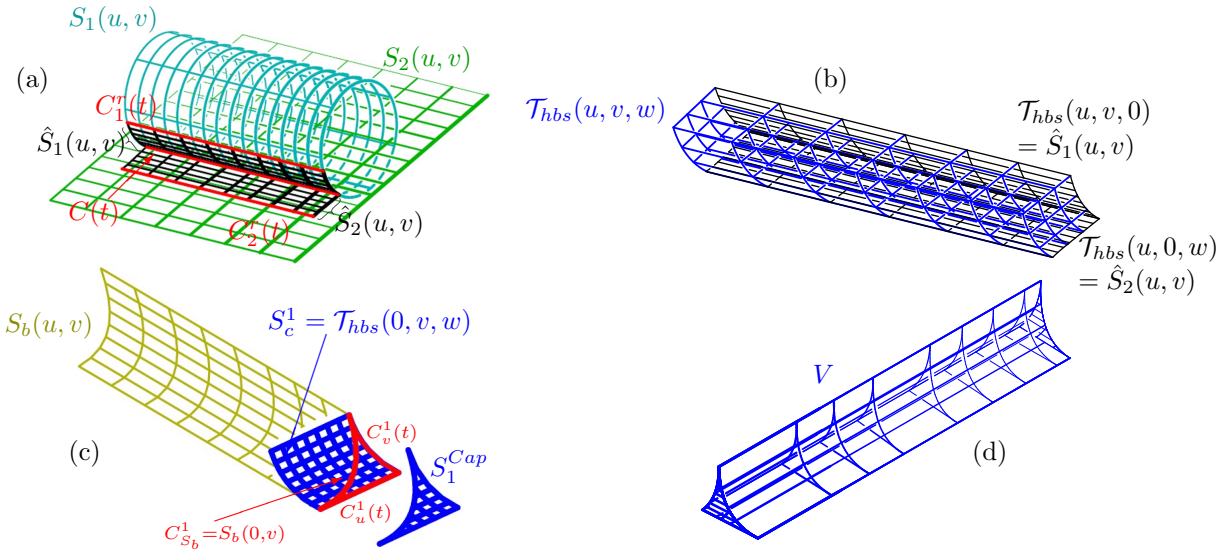


Figure 10: (a) A cylinder, S_1, intersects with a planar surface, S_2, along the intersection curve, C. C_i^r $i \in \{1,2\}$, are the rail curves on S_i, respectively, and the sub-surface regions, $\hat{S}_i = S_i(Rld(c_{\mathcal{I}}^{S_i}, c_i^r))$, are the boundaries of the fillet on S_i. (b) presents the half Volumetric Boolean sum trivariate, $\mathcal{T}_{hbs}$, between surfaces, $\hat{S}_1$ and $\hat{S}_2$ (see Definition 5.1). Herein, the intersection curve, C, is not closed, and hence in (c), two capping surfaces are constructed. S_1^{Cap} is computed by trimming the boundary surface $S_c^1 = \mathcal{T}_{hbs}(0,v,w)$ by curves $C_{S_b} = S_b(0,v)$, C_u^1 and C_v^1. The final V-rep model fillet, created by Algorithm 6, is shown in (d), from a different viewing direction.

be a vector containing the M intersection curves, $C_{i,j}$, sorted in a clockwise order (looking from outside). The filleting process we introduce in this section consists of two main steps. Initially for each $0 \leq i \leq j \leq 5$ such that $C_{i,j} \neq \emptyset$, we construct a separate fillet trivariate between $S_{1,i}$ and $S_{2,j}$, using algorithms discussed in the previous sections. This set of fillets might not form a closed fillet object, due to C^1 discontinuities in the intersection curves. Therefore, in Section 6.1, a second, new, step of this filleting process is discussed, constructing additional fillets that bridge all gaps between the trivariates constructed in this first step.

One should note that the approach we describe in Section 6.1 is best fitted to handle convex C^1 discontinuities of the intersection curve, C. While convex C^1 discontinuities result with the existence of gaps between adjacent rail curves, concave C^1 discontinuities would result with the adjacent rail curves crossing each other. Such concave discontinuities could be similarly handled, as we describe here, while it would result with C^1 discontinuities on the fillet surface, along the trimmed location. In the rest of this section, we focus on creating smooth convex fillets, while

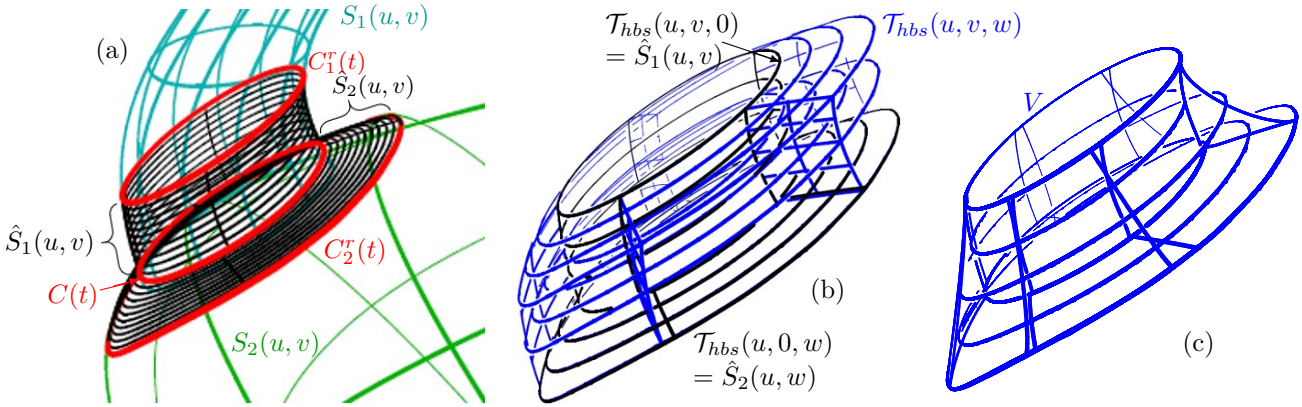


Figure 11: In (a), the spout, S_1, of the Utah teapot, is shown intersecting with the body, S_2, of the Utah teapot, along the intersection curve, C. C_i^r, $i \in \{1,2\}$, are the two rail curves on S_i, and the sub-surfaces, $\hat{S}_i = S_i(Rld(c_{\mathcal{I}}^{S_i}, c_i^r))$, are the boundaries of the fillet on S_i. In (b), the half Volumetric Boolean sum trivariate, $\mathcal{T}_{hbs}$, of $\hat{S}_1$ and $\hat{S}_2$ is presented (see Definition 5.1). (c) shows the V-rep model fillet created by Algorithm 6. See also Figure 17.

we leave the handling of concave C^1 discontinuities for future work.

6.1. Bridging the Gaps in the Fillets, near C^1 discontinuities

The two trivariates to be filleted, $\mathcal{T}_1$ and $\mathcal{T}_2$, might contain C^1 discontinuities along C. Hence, after offsetting all M intersection curves, the resulting set of M rail curves,

$$\{R^k_{i_m,j_m}\}_{m=0}^{M-1}, \quad 0 \le i \le j \le 5, \quad k \in \{1,2\},$$

might not form a C^0 closed continuous rail curve. That is, gaps will exist between adjacent rail curves in the set, again, due to C^1 discontinuities in the intersection curves (see Figures 12).

Let $C_{i_m,j_m} = \mathcal{C}[m]$ and $C_{i_{m+1},j_{m+1}} = \mathcal{C}[m+1]$, be two adjacent intersection curves with a C^1 discontinuity. This discontinuity can happen due to $\mathcal{T}_1$ being discontinuous, $\mathcal{T}_2$ being discontinuous, or both. We assume that the event where both $\mathcal{T}_1$ and $\mathcal{T}_2$ are simultaneously discontinuous at the same location is not likely and ignore it. Hence, and without loss of generality, we assume that there is a gap between $R^1_{i_m,j_m}$ and $R^1_{i_{m+1},j_{m+1}}$, while $R^2_{i_m,j_m}$ and $R^2_{i_{m+1},j_{m+1}}$ are connected with a C^0 continuity (The other case is symmetric). Recall that all intersection curves form together a close, C^0 continuous, curve. Thus, C_{i_m,j_m} and $C_{i_{m+1},j_{m+1}}$ share an endpoint. Let o be the shared endpoint of C_{i_m,j_m} and $C_{i_{m+1},j_{m+1}}$, let p be the last point of $R^1_{i_m,j_m}$ and let q be the first point of $R^1_{i_{m+1},j_{m+1}}$ (See Figure 12 (e)).

Because the rail curves are computed as Euclidean offsets of the intersection curves, it holds that $d(C_{i,j}, R^1_{i_m,j_m}) = d(C_{i_{m+1},j_{m+1}}, R^1_{i_{m+1},j_{m+1}})$, and hence, $d(o,p) = d(o,q) = d$. Let B_d be the locus of points on $S_{1,m}$ and $S_{1,m+1}$ that have a Euclidean offset distance d from p. The subset of B_d between p and q is used to bridge the rail curves, $R^1 i_m, j_m$ and $R^1_{i_{m+1},j_{m+1}}$. Then, a trivariate fillet is constructed between o and the subset of B_d between p and q, just computed, to fill the gap. Note this new trivariate fillet that bridges between the last face of the fillet of intersection curve C_{i_m,j_m} and the first face of the fillet of intersection curve $C_{i_{m+1},j_{m+1}}$ is singular along the C^0 discontinuity of input filleted trivariate, $\mathcal{T}_i$. Again, Figures 12 illustrate this computation process of adding these singular trivariate fillets.

7. Blending Materials and Properties

Applications of our filleting algorithms, such as 3D printing, require the encoding of heterogeneous properties into the constructed fillet, in a way it continuously connects with the two primary surfaces. Thus, we now consider the question of blending the given heterogeneous property functions of the two input surfaces, S_1 and S_2, restricted to the filleted area, $\hat{S}_1$ and $\hat{S}_2$. In other words, we seek to encode blended properties into the trivariates(s) of the fillet that were constructed in previous sections, so as to ensure that the property values are continuous in-between $\hat{S}_1$ and $\hat{S}_2$.

Let $\mathcal{T}$ be a fillet trivariate and let $\hat{S}_1 \subset S_1$ and $\hat{S}_2 \subset S_2$ be the two boundary surfaces of the fillet. We assume that each surface, $\hat{S}_i$, is associated with a unique property function, $\mathbf{g_i}$, encoding properties of $\hat{S}_i$ (such as material). Each property function, $\mathbf{g_i}$, is a multi-variate B-spline that shares the same parametric domain with $\hat{S}_i$. Without loss of generality, we assume $\mathbf{g_i}$ to be a scalar function with sufficient continuity and we denote it as g_i. All the

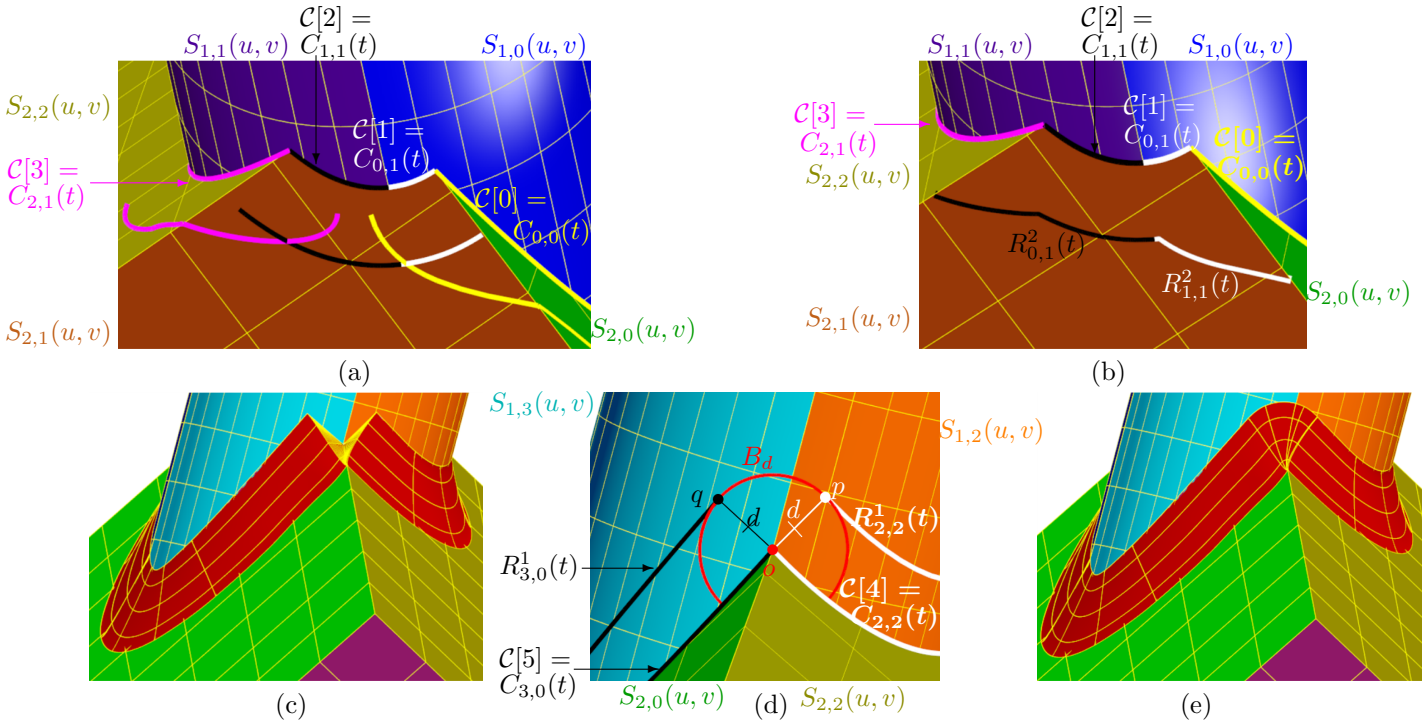


Figure 12: Filleting over C^1 discontinuities. (a) shows all the intersection curves, C_{i_m,j_m}, of the cylinder surfaces, $S_{1,i}$, and the surfaces of the box, $S_{2,j}$, as well as the corresponding rail curves. In (b), the Euclidean offset curves from (a) are clipped and stitched together, forming the rail curve(s). Filleting the corresponding pieces, gaps forms at C^1 discontinuities, as shown in (c). In (d), the intersection curves, $C_{2,2} = S_{1,2} \cap S_{2,2}$ and $C_{3,0} = S_{1,3} \cap S_{2,0}$, are offseted on the boundary surfaces of the cylinder, $S_{1,2}$ and $S_{1,3}$, forming two disjoint pieces of the rail curve on the cylinder, $R^1_{2,2}$ and $R^1_{3,0}$, that end at p and q. A singular fillet is constructed between p and q, around o, to bridge the gap. Finally, this singular fillet, along with other fillet trivariates form the whole fillet between the box and the cylinder, are shown in (e).

following discussion can be easily adapted to the general multivariate setting (vector, tensor etc.) by applying the same material encodement method to each coordinate.

We propose two methods to encode a heterogeneous property function into fillet $\mathcal{T}$. In Section 7.1, we describe a method to directly encode the required property into the control points of the fillet trivariate, as a convex combination of the control points of g_i, hence exploiting *parametric distances*. In contrast, in Section 7.2, we describe a blending scheme based on the *Euclidean distances* of sampled points from $\hat{S}_i$.

7.1. Direct (Parametric) Property Blending

Assume $\mathcal{T}$ is a fillet trivariate constructed as a ruled volume between the merged two primary surfaces, $\hat{S}_{12} = \hat{S}_1 \cup \hat{S}_2$ and the blending surface S_b (recall Algorithm 2). Further, assume that $\hat{S}_{12}$ and S_b are two surfaces that share the same function space. If not, one can always bring both surfaces into a common function space via degree raising and refinements.

Denote by n_u, n_v the number of coefficients of $\hat{S}_{12}$ and S_b and consider

$$g_i(u,v) = \sum_{m=0}^{n_u-1} \sum_{r=0}^{n_v-1} B^{(i)}_{m,r}(u,v) p^{(i)}_{m,r},$$

where $B^{(i)}_{m,r}$ are the basis functions of g_i. Bringing g_i, $i = 1, 2$, into a common function space, we will have $g_i(u,v) = \sum_{m=0}^{n_u-1} \sum_{r=0}^{n_v-1} B_{m,r}(u,v) p^{(i)}_{m,r}$. To continuously encode the blended properties into $\mathcal{T}$, we first associate S_b with the following property function:

$$g_b(u,v) = \sum_{m=0}^{n_u-1} \sum_{r=0}^{n_v-1} (1-\alpha_r) B_{m,r}(u,v) p^{(1)}_{m,r} + \alpha_r B_{m,r}(u,v) p^{(2)}_{m,r} = \sum_{m=0}^{n_u-1} \sum_{r=0}^{n_v-1} ((1-\alpha_r) p^{(1)}_{m,r} + \alpha_r p^{(2)}_{m,r}) B_{m,r}(u,v),$$

where $\alpha_r = \frac{r}{n_v-1}$. Note that $g_b(u,v) = g_1(u,v)$ for $\alpha_r = 0$ and $g_b(u,v) = g_2(u,v)$ for $\alpha_r = 1$.

Then, a process similar to Algorithm 2 can be utilized to merge the two property functions, g_1 and g_2, as

$$g_{12}(u,v) = \begin{cases} g_1(u,2v), & 0 \le v \le 0.5, \\ g_2(u,2(v-0.5)), & 0.5 < v \le 1. \end{cases} \tag{8}$$

only to serve as the property function associated with the merged surface $\hat{S}_{12}$ (recall Equation (5)).

Then, the property function, $g_{\mathcal{T}}(u, v, w)$, encodes the ruled trivariate fillet $\mathcal{T}$, as:

$$g_{\mathcal{T}}(u, v, w) = (1 - w)g_{12}(u, v) + wg_b(u, v),$$

after bringing g_{12} and g_b, once more, to a common function space.

One should note that if g_1 and g_2 have different property values along the intersection curve, $C = \hat{S}_1 \cap \hat{S}_2$, then g_{12} and hence $g_{\mathcal{T}}$ will be discontinuous there. Otherwise, this simple interpolatory scheme will yield C^0 continuity along $\hat{S}_1$ and $\hat{S}_2$.

7.2. Euclidean Distance based Property Blending

For each point $p \in \mathcal{T}$, denote $\pi_i(p) = \arg\min ||\hat{S}_i(q) - p||$. I.e. $\pi_i(p)$ is the point in $\hat{S}_i$'s parametric domain that corresponds to the minimal distance from p to $\hat{S}_i$. Further, following Cirillo and Elber (2020), given two weight functions, $\alpha_1(p), \alpha_2(p)$, $p \in \mathcal{T}$, the function

$$f(p) = \alpha_1(p)g_1(\pi_1(p)) + \alpha_2(p)g_2(\pi_2(p)),$$

is said to be a *local feature preserving* blending scheme, if α_1 and α_2 satisfy the following conditions:

$$\textit{Feature preservation}: \alpha_i(p) = 1, p \in \hat{S}_i, i = 0, 1, \tag{9a}$$

$$\textit{Partition of unity}: \alpha_1(p) + \alpha_2(p) = 1, p \in \mathcal{T}, \tag{9b}$$

$$\textit{Positivity}: \alpha_i(p) \geq 0, p \in \mathcal{T}. \tag{9c}$$

Equations (9b) and (9c) ensure that the blending scheme f is a convex combination of the original property functions g_0 and g_1 at each point. In addition, Equation (9a) makes sure that the value of the property in a point p reproduces the input material distributions as p approaches one of $\hat{S}_1$ and $\hat{S}_2$.

For the purpose of material encodement into the fillet, $\mathcal{T}$, we associate $\mathcal{T}$ with a feature preserving blending scheme, f, that blends the specifications of the materials, g_1 and g_2 over the two surfaces, $\hat{S}_1$ and $\hat{S}_2$, with a desired continuity. The blending scheme we use herein has been suggested in Biswas et al. (2004), and it arises from the well known Shepard interpolant (Shepard (1968)). This blending scheme defines α_1 and α_2 as follows:

$$\alpha_i(p) = \frac{d_i(p)^{-q}}{d_1(p)^{-q} + d_2(p)^{-q}}, \quad i = 0, 1,$$

where $d_i(p) = ||\hat{S}_i(\pi_i(p)) - p||$ represents the minimal distance of p from the boundary of $\hat{S}_i$ and q is a fixed positive integer that guarantees the blending functions to be C^q continuous. The encoding of a material property into $\mathcal{T}$ is done by evaluating f in a user-controlled number of sampled points in the domain of $\mathcal{T}$, and solving a least-square-fitting problem, as shown in Cirillo and Elber (2020). To ensure the interpolation of the boundary surfaces, one should fix and ignore the points at the boundaries during the least-square-fitting, only to substitute them for the control points of the property functions, g_1 and g_2, after the least-square-fitting process. As in the case of direct blending (recall Section 7.1), if S_i ($i \in \{1, 2\}$) is a boundary surface of a trivariate, $\mathcal{T}_i$, then the encoded property function of $\mathcal{T}$ joins original trivariates $\mathcal{T}_i$, along S_i, with C^0 continuity. C^q continuity is guaranteed in the interior of the fillet trivariate, $\mathcal{T}$.

8. Results and Examples

We presents several examples of heterogeneous fillets computed for and in-between different pairs of trivariates with heterogeneous properties, herein represented as rgb colors. These rgb colors can represent, for example, material densities, stiffness tensors or heat conductivity factors, etc. The computation, unless otherwise stated, was conducted on a Windows machine with Intel i7 core 2.6GHz, 64 bit and 16GBs RAM. All the examples were created using an implementation based on the IRIT (**?**) solid modeling environment, developed at the Technion, and IRIT's volumetric moduling support (Massarwi and Elber (2016)) was used to create the VModels shown in the figures.

Figures 13 to 14 show fillets that were computed based on Algorithm 2 and *rgb* property encodement that were computed as described in Section 7.1, using the direct (parametric) blending of properties (Section 7.1). It took less than a minute to compute each of these fillets. Figure 15 shows two approximated fillets that were computed as described in Section 4.3 and *rgb* property encodement that was computed as described in Section 7.2, exploiting Euclidean distance based blending. It took a few seconds to least square fit and compute each of these two fillets and about six minutes to compute the cross section image shown in Figure 15 (b), of size (1516×2524).

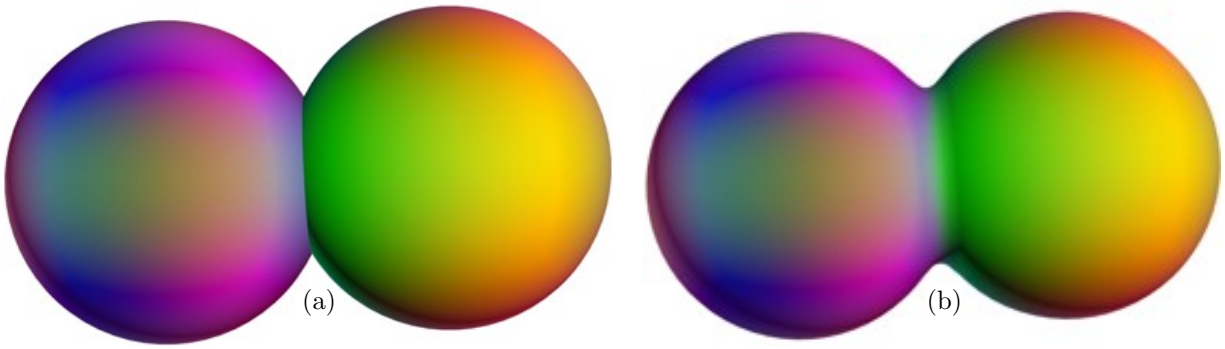


Figure 13: A heterogeneous fillet computed between two tri-cubic spheres. In (a), the two spheres are shown, after heterogeneous data in $E6$ has been independently encoded into them, as (x, y, z, r, g, b). (b) shows the computed final fillet, following Algorithm 2, while the blending of the rgb properties was computed as described in Section 7.1.

Figure 15 (b) shows a slice of the model along with its fillet as a colored image. Numerous such images are used to create the necessary data-set to 3D print such models using the VoxelPrint Stratasys (2020) technology of Stratasys. Figures 16 and 17 present such a 3D printed example. A heterogeneous Utah teapot, in Figures 16 (a) and 17 (a), is being filleted using the described algorithms, in Figures 16 (b) and 17 (b), and then the geometry and materials were simultaneously sliced over three thousand times to yield the necessary data that created the 3D printed model, in Figures 16 (c) and 17 (c). It took a few hours to compute all the slices while the 3D printing process itself consumed almost a day, on a Stratasys machine.

Table 1 summarizes the number of control points and orders for in the input geometry and the resulting fillets, in these displayed results. Note that some entries present the values of multiple trivariates and hence are shown as ranges.

Table 1: Control points and orders of data for Figures 13 - 17

	S_1		S_2		$\hat{S}_1$	
	orders	#ctlpts	orders	#ctlpts	orders	#ctlpts
Fig. 13	$(4,4)$	$(4,4)$	$(4,4)$	$(4,4)$	$(7,19)$	$(7,199)$
Fig. 14	$(4,2)$	$(13,2)$	$(4,4)$	$(13,6)$	$(5,13)$	$(5,16-25)$
Fig. 15 (Base&Rod)	$(4,3)$	$(13,8)$	$(4,3)$	$(13,9)$	$(6,3)$	$(6,35)$
Fig. 15 (Rod&Lamp)	$(4,4)$	$(4,7)$	$(4,3)$	$(13,9)$	$(7,3)$	$(7,35)$
Fig. 16	$(4,4)$	$(13,7)$	$(4,4)$	$(12,13)$	$(7,13)$	$(7,49-133)$
Fig. 17	$(4,4)$	$(7,7)$	$(4,4)$	$(12,13)$	$(7,19)$	$(7,81-145)$
	$\hat{S}_2$		S_b		T	
Fig. 13	$(7,19)$	$(7,189)$	$(4,4)$	$(4,20)$	$(7,19,2)$	$(13,275,2)$
Fig. 14	$(7,19)$	$(7,19-79)$	$(4,4)$	$(4,20)$	$(7,19,2)$	$(13,294-348,2)$
Fig. 15 (Base&Rod)	$(3,3)$	$(3,35)$	$(4,3)$	$(4,35)$	$(6,3,2)$	$(11,35,2)$
Fig. 15 (Rod&Lamp)	$(3,3)$	$(3,35)$	$(4,3)$	$(4,35)$	$(7,3,2)$	$(13,35,2)$
Fig. 16	$(7,13)$	$(7,49-133)$	$(4,3)$	$(4,20)$	$(7,13,2)$	$(13,284-440,2)$
Fig. 17	$(7,19)$	$(7,107-156)$	$(4,4)$	$(4,20)$	$(7,19,2)$	$(13,399-517,2)$

9. Conclusion and future work

This work presented several methods for the construction of trivariate fillets, and discussed two possible methods to encode heterogeneous material into the computed fillet. The trivariate fillet construction algorithms introduced herein support varying sets of surfaces, and each algorithm works under different assumptions. While Section 4.3 presents a method to create an approximated fillet between two general intersecting B-spline surfaces, the problem of constructing a precise trivariate fillet for two B-spline surfaces in general position (crossing knot line) is still open, and is an area for future work.

While the approaches we introduced are shaped to approximately achieve symmetric or qualitative measures (e.g. improved Jacobian values and preserving continuity), these were not the main focus of this work. Methods for qualitatively formalizing and optimizing fillets should be investigated. This, with the understanding that by definition, at the rail curves, the Jacobian of all presented approaches vanish, simply because S_b is tangent to S_i.

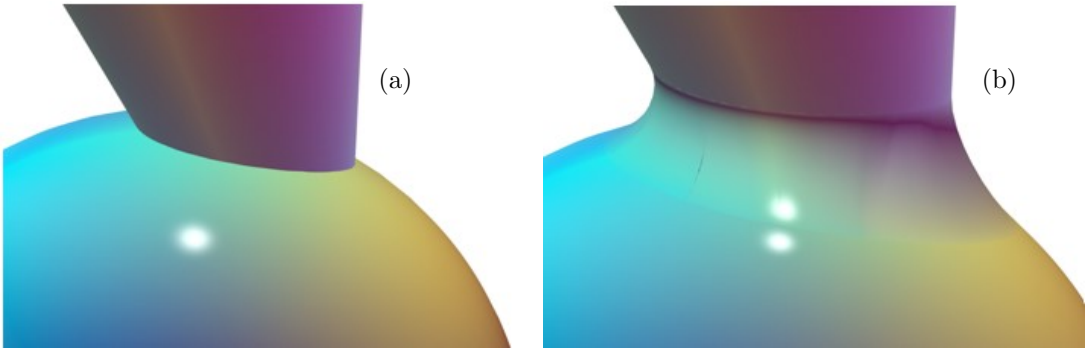


Figure 14: A heterogeneous fillet computed between a spheres and a cone. In (a), intersecting sphere and cone are shown, after heterogeneous data in $E6$ has been independently encoded into both, as (x, y, z, r, g, b). (b) shows the computed fillet, following Algorithm 2, while the blending of the rgb properties was computed as described in Section 7.1.

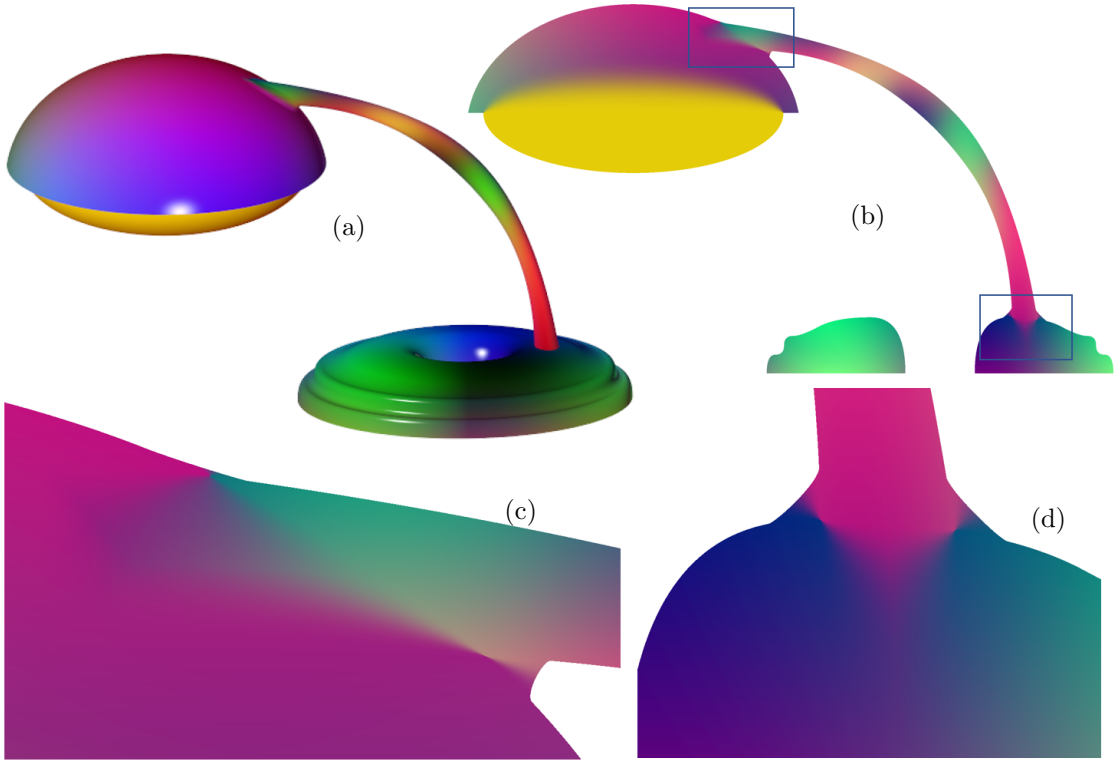


Figure 15: Two heterogeneous fillets computed for a VModel of a lamp, between the base and the connecting-rod and between the connecting-rod and the lamp-head. In (a), the input lamp is shown, after heterogeneous data in $E6$ has been encoded into all primitives independently, as (x, y, z, r, g, b). The fillets were computed by the least-square-fitting method described in Section 4.3. (b) shows a cross section of the filleted model after applying the distance-based blending scheme described in Section 7.2. (c) and (d) are two zoom views of the two blue rectangles in (b), focusing on the fillet zones. See also Figure 9.

The computed fillets join the boundary surfaces of the filleted trivariates with geometrically C^1 continuity. Yet, the property functions adhere to only C^0 continuity over the $\hat{S}_i$. Higher geometric continuity can be achieved using known methods such as quintic Hermite (for C^2 geometric continuity). That said, future work should strive to extend the presented filleting and heterogeneity encodement methods to achieve higher order continuity of properties.

One should remember that if two different geometries present different properties at the intersection curve C, the geometry will present a discontinuity in the property. In fact, in all presented examples in this work, we intentionally encoded independently different property values in the different input trivariates (and their boundary surfaces). Such a difficulty better be managed before the fillet is applied, striving to ensure the properties are smooth or at least continuous along C, if possible.

The context of this work was limited to the filleting of two intersecting trivariates. Therefore, a possible continuation of this work may be extending the introduced filleting algorithms to support the filleting of more than two intersecting objects. Moreover, while the two input trivariates were allowed to have C^1 discontinuities along

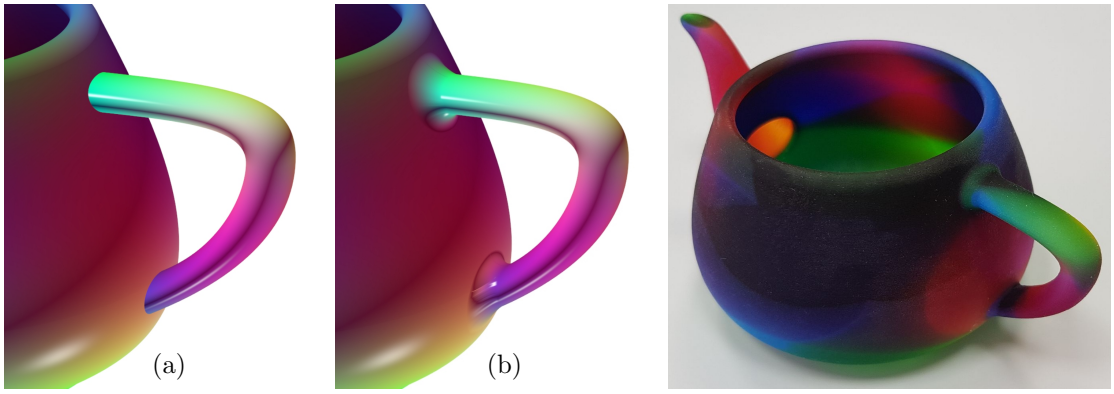


Figure 16: A heterogeneous fillet computed for the Utah teapot, between the body and the handle. In (a), the regular teapot is shown, after heterogeneous data in $E6$ has been encoded into, as (x, y, z, r, g, b). (b) shows the computed fillet, following Algorithm 2 while the rgb blending was computed as described in Section 7.1. In (c), a 3D printed model is shown, using a Stratasys 3D printer. See also Figure 17.

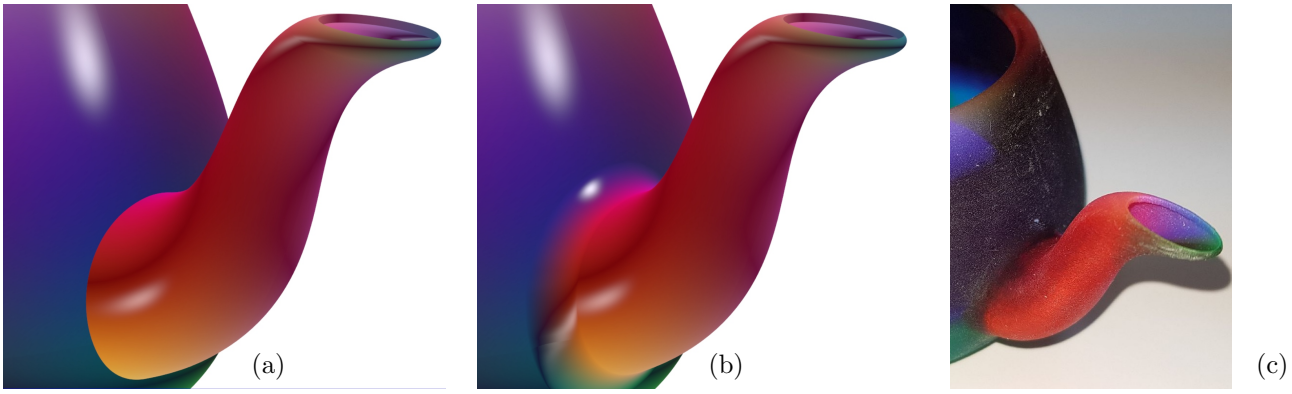


Figure 17: A heterogeneous fillet computed for the Utah teapot, between the body and the spout. In (a), the regular teapot is shown, after heterogeneous data in $E6$ has been encoded into, as (x, y, z, r, g, b). (b) shows the computed fillet, following Algorithm 2 while the rgb blending was computed as described in Section 7.1. In (c), a 3D printed model is shown, using a Stratasys 3D printer. See also Figure 17.

their intersection curve, the proposed handling of concave C^1 discontinuities results with the rail curves being C^1 discontinuous. Other methods for the handling of concave discontinuities should be investigated. Further, while in Section 3.2, the capping surfaces of the fillet were generated by Boolean Sum, one should generally ensure the capping of adjacent fillet trivariates is done in a continuous manner. This capping construction process is yet another degree of freedom to explore and an area for future work.

All of the presented filleting Algorithms depend on the Hermite interpolation scheme to construct the blending surface, while, clearly, other surface blending schemes may be combined with the presented trivariate Algorithms. Additionally, the filleting methods we introduced strongly depended on surface-surface compositions, which makes the amount of the control points in the constructed fillet grow rapidly. Methods for minimizing the number of control points in the resulting fillet should be investigated as well.

Finally, and while we also showed that we are able to 3D print the synthesized geometry with independent encodement of properties, the analysis of such fillets pose a challenge. To begin with, the geometry of the constructed fillet is not compatible with the boundary surfaces, S_i, and hence mortar methods or similar might be required to 'glue' the geometries. Then, and while the Jacobian is positive in the interior of the fillet, the Jacobian of the fillet vanishes at the rail curves. The behavior of such fillets in analysis and in iso-geometric analysis in specific, is yet to be examined.

Acknowledgment

This research was supported in part by the ISRAEL SCIENCE FOUNDATION (grant No. 597/18), in part by the European Union Horizon 2020 research and innovation programme, under grant agreement No. 862025, and in part by the Swiss National Science Foundation, through the project "Design-through-Analysis (of PDEs): the litmus test - BRIDGE Discovery 2019" (Project No. 40B2-0 187094).

Appendix A. Computing the Division of the Fillet Space toward Volumetric Boolean Sum

Recall Figure 5 for this appendix. Let $p = C(0.5)$ be the mid-point of the intersection curve, $C(t)$. Orthogonally project p on the iso-parametric curve, $C^b_{mid}(v) = S_b(0.5, v)$, by solving the following constraint for v:

$$\left\langle p - C^b_{mid}(v), \frac{dC^b_{mid}(v)}{dv} \right\rangle = 0. \tag{A.1}$$

Let $p_b = C^b_{mid}(v_b) = S_b(0.5, v_b)$ be the point that satisfy the above constraints and whose Euclidean image on S_b is the point closest to P. Define C_b to be the u iso-parametric curve of $S_b(u, v)$ that passes through $p_b = (0.5, v_b)$. I.e. $C_b(u) = S_b(u, v_b)$.

To compute $C_c(t)$, the centrally interior curve in the fillet space, we define

$$S_r(u, v) = Rld(C, C_b) = (1 - v)C + vC_b,$$

to be the ruled surface between the intersection curve, C, and the curve C_b, just computed. Then,

$$C_c(t) = S_r\left(t, \frac{\sqrt{2}}{1+\sqrt{2}}\right) = C(t) + \frac{\sqrt{2}}{1+\sqrt{2}}(C_b(t) - C(t)), \tag{A.2}$$

divides the ruled surface, $S_r(u, v)$, by ratio $\sqrt{2} : 1$. This ratio strives for a 'fair' division of the fillet space so that in the case where $\hat{S}_1$ and $\hat{S}_2$ are orthogonal (and S_b is forming a slope of $\pi/4$ at p_b), then the point $C_c(0.5)$ will be equidistant from the three surfaces, S_b, $\hat{S}_1$ and $\hat{S}_2$ (See Figure 5(b)).

To compute the foot curves, C_i, on $\hat{S}_i(u, v)$, we follow a similar process. Let $p_i = \hat{S}_i(0.5, v_i)$, $i \in \{1, 2\}$, be the orthogonal projections of $C_c(0.5)$ on the iso parametric curve, $C^i_{mid}(v) = \hat{S}_i(0.5, v)$ (computed as in Equation (A.1)). Then, foot curve $C_i(u)$ is defined as the v iso-parametric curve on $\hat{S}_i$ passing through p_i. I.e. $C_i(u) = \hat{S}_i(u, v_i)$. Again, Figure 5(b) illustrates this computation process of building C_b, C_c, C_1 and C_2.

Algorithm 7 summarizes this division process as described in this appendix.

Algorithm 7: Computing supporting curves for filleting by volumetric Boolean sum operations.

Input:
$S_1(u_1, v_1)$, $S_2(u_2, v_2)$ - the primary surfaces;
$c^r_i(t)$ - rail curve in the parametric space of S_i;
$C(t)$ - intersection curve in Euclidean space;
τ - control over the magnitudes of the tangent fields of the blending surface;
Output:
$C_c(t), C_i(t), C_b(t)$ - interior curve to the fillet space and three supporting curves on S_u and S_b;

BzrTrivarFilletBoolSumCrvs$(S_1, S_2, c^r_1, c^r_2, C, \tau)$**:**
- $S_b(u, v) = BlendingSrf(S_1, S_2, c^r_1, c^r_2, \tau)$; // Algorithm 1
- $P_b = S_b(u_b, v_b) =$ orthogonal projection of $C(0.5)$ on $C^b_{mid}(v) = S_b(0.5, v)$; // See Equation (A.1)
- $c_b(u) = (u, v_b)$ // $c_b \subset S_b$'s parametric space
- $C_b(u) = S_b(c_b(u))$; // See Figure 5c
- $C_c(t) = C(t) + \frac{\sqrt{2}}{1+\sqrt{2}}(C_b(t) - C(t))$; // See Equation (A.2)
- $P_i = S_i(u_i, v_i) =$ orthogonal projection of $C_c(0.5)$ on $C^i_{mid}(v) = S_i(0.5, v)$; // $i \in \{1, 2\}$
- $c_i(u) = (u, v_i)$; // $c_i \subset S_i$'s parametric space
- $C_i(u) = S_i(c_i(u))$; // See Figure 5c
- return $\{C_c(t), C_i(t), C_b(t)\}$;

References

Ameta, G., Witherell, P., 2019. Representation of graded materials and structures to support tolerance specification for additive manufacturing application. J. Comput. Inf. Sci. Eng. 19.

Bajaj, C.L., Ihm, I., 1992. Algebraic surface design with hermite interpolation. ACM Trans. Graph. 11, 61–91.

Bartoň, M., Elber, G., Hanniel, I., 2010. Topologically guaranteed univariate solutions of underconstrained polynomial systems via no-loop and single-component tests, in: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, Association for Computing Machinery, New York, NY, USA. p. 207–212.

Belkhatir, B., Zidna, A., 2009. Construction of flexible blending parametric surfaces via curves. Mathematics and Computers in Simulation 79, 3599 – 3608. The International Conference on Approximation Methods and numerical Modeling in Environment and Natural Resources.

Biswas, A., Shapiro, V., Tsukanov, I., 2004. Heterogeneous material modeling with distance fields. Computer Aided Geometric Design 21, 215 – 242.

Bizzarri, M., Lvika, M., Kosinka, J., 2017. Skinning and blending with rational envelope surfaces. Comput. Aided Des. 87, 41–51.

Chen, F., Tang, X., 2003. G2 blending of corners with piecewise algebraic surfaces. Proceedings - Pacific Conference on Computer Graphics and Applications 2003, 93–101.

Cheng, J., 2003. Blending quadric surfaces via base curve method. Computer Mathematics , 77–86.

Cirillo, E., Elber, G., 2020. Handling heterogeneous structures and materials using blending schemes in v-reps. Computer Aided Geometric Design 83, 101942.

Cohen, E., Riesenfeld, R., Elber, G., 2001. Geometric Modeling with Splines: An Introduction.

DeRose, T.D., Goldman, R.N., Hagen, H., Mann, S., 1993. Functional composition algorithms via blossoming. ACM Trans. Graph. 12, 113–135.

Dutta, D., Shin, K., 2001. Constructive representation of heterogeneous objects. J. Comput. Inf. Sci. Eng. 1, 205–217.

Elber, G., 1992. Free Form Surface Analysis Using A Hybrid of Symbolic and Numerical Computation. Ph.D. thesis. University of Utah.

Elber, G., 2005. Generalized filleting and blending operations toward functional and decorative applications. Graph. Models 67, 189–203.

Elber, G., Kim, M.S., 2020. Euclidean offset and bisector approximations of curves over freeform surfaces. Computer Aided Geometric Design 80, 101850.

Elber, G., Kim, Y.J., Kim, M.S., 2012. Volumetric boolean sum. Computer Aided Geometric Design 29, 532 – 540. Geometric Modeling and Processing 2012.

Farouki, R., Pelosi, F., Sampoli, M.L., 2019. Optimization of corner blending curves. Computer-Aided Design 117, 102739.

Gao, X.S., Li, M., 2002. Construct piecewise hermite interpolation surface with blending methods. Geometric Modeling and Processing. Theory and Applications. GMP 2002. Proceedings , 53 – 59.

Hoffmann, C., Hopcroft, J., 1988. The geometry of projective blending surfaces. Artificial Intelligence 37, 357 – 376.

Hsu, P.C., 2006. Implicit blends with an individual blending range control on every primitive's subsequent blend, in: Proceedings - Computer Graphics, Imaging and Visualisation: Techniques and Applications, CGIV'06, pp. 534 – 541.

Kim, K., Elber, G., 1997. A symbolic approach to freeform parametric surface blends. THE J. OF VISUALIZATION AND COMPUTER ANIMATION .

Kou, X., Tan, S.T., 2007. Heterogenous object modeling: a review. Computer-Aided Design 39, 284–301.

Kou, X., Tan, S.T., Sze, W.S., 2006. Modeling complex heterogeneous objects with non-manifold heterogeneous cells. Computer-Aided Design 38, 457–474.

Kumar, V., Dutta, D., 1998. An approach to modeling & representation of heterogeneous objects. Journal of Mechanical Design 120, 659–667.

Li, B., Fu, J., Feng, J., Shang, C., Lin, Z., 2020. Review of heterogeneous material objects modeling in additive manufacturing. Vis. Comput. Ind. Biomed. Art 3.

Li, Q., 2004. Blend implicit shapes using smooth unit step functions. WSCG , 297–304.

Lin, H., Xiong, Y., Liao, H., 2014. Semi-structured B-spline for blending two B-spline surfaces. Computers & Mathematics with Applications 68, 706 – 718.

Machchhar, J., Elber, G., 2018. A note on zeros of univariate scalar bernstein polynomials. Computer Aided Geometric Design 66, 75 – 79.

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

Samanta, K., Koc, B., 2004. Heterogenous object design with material feature blending. Comput. Aided Des. Appl. 1, 429–437.

Samanta, K., Koc, B., 2005. Feature-based design and material blending for free-form heterogeneous object modeling. Computer-Aided Design 37, 287–305.

Sanglikar, M., Koparkar, P., Joshi, V., 1990. Modelling rolling ball blends for computer aided geometric design. Computer Aided Geometric Design 7, 399 – 414.

Shepard, D., 1968. A two-dimensional interpolation function for irregularly-spaced data, in: Proceedings of the 1968 23rd ACM National Conference, Association for Computing Machinery, New York, NY, USA. p. 517–524.

Shin, K., 2002. Representation and process planning for layered manufacturing of heterogeneous objects. Ph.D. thesis. University of Michigan.

Stratasys, 2020. Voxelprint technology. URL: https://help.grabcad.com/article/230-guide-to-voxel-printing?locale=en.

Varady, T., Martin, R.R., Vida, J., 1989. Topological considerations in blending boundary representation solid models. Theory and Practice of Geometric Modeling , 205–220.

Vida, J., Martin, R.R., Varady, T., 1994. A survey of blending methods that use parametric surfaces. Computer-Aided Design 26, 341 – 365.

ru Wu, T., shi Zhou, Y., 2000. On blending of several quadratic algebraic surfaces. Computer Aided Geometric Design 17, 759 – 766.

You, L., Ugail, H., Tang, B., Jin, X., You, X., Zhang, J., 2014. Blending using ODE swept surfaces with shape control and C^1 continuity. The Visual Computer 30, 625–636.

You, X., Tian, F., Wen, T., 2019. C^2 continuous blending of time-dependent parametric surfaces. Journal of Computing and Information Science in Engineering 19, 1.

You, X.Y., Tian, F., Tang, W., 2018. A unified approach to blending of constant and varying parametric surfaces with curvature

continuity, in: Proceedings of Computer Graphics International 2018, Association for Computing Machinery, New York, NY, USA. p. 51–56.

Zhou, P., W.-H, Q., 2012. G^n-blending of multiple parametric normal ringed surfaces by adding implicit closings gn-continuous with the surfaces. Computers and Graphics 36, 297–304.