Kernel-based Construction Operators for Boolean Sum and Ruled Geometry

Haitham Fadila[a,*], Q Youn Hong[b], Gershon Elber[a]

[a] *Faculty of Computer Science, Technion-Israel Institute of Technology, Israel*
[b] *Department of Computer Science and Engineering, Hanyang University, Ansan, South Korea*

Abstract

Boolean sum and ruling are two well-known construction operators for both parametric surfaces and trivariates. In many cases, the input freeform curves in $\mathbb{R}^2$ or surfaces in $\mathbb{R}^3$ are complex, and as a result, these construction operators might fail to build the parametric geometry so that it has a positive Jacobian throughout the domain.

In this work, we focus on cases in which those constructors fail to build parametric geometries with a positive Jacobian throughout while the freeform input has a kernel point. We show that in the limit, for high enough degree raising or enough refinement, our construction scheme must succeed if a kernel exists. In practice, our experiments, on quadratic, cubic and quartic Bézier and B-spline curves and surfaces show that for a reasonable degree raising and/or refinement, the vast majority of construction examples are successful.

Keywords: Tensor Product Surfaces, Tensor Product Trivariates, Boolean Sum, Ruled Curves/Surfaces, Half Boolean Sum, Kernel.

1. Introduction

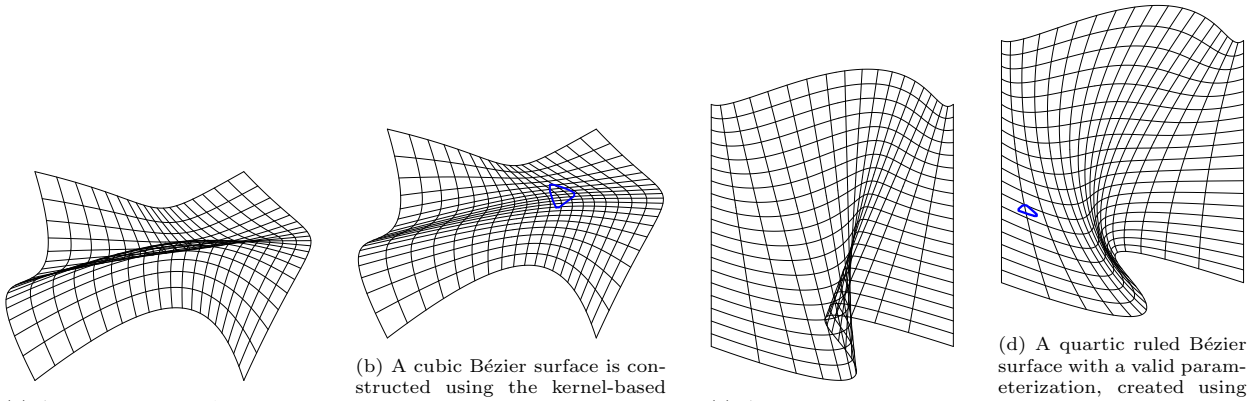
Boolean sum and ruling operators are very common construction operations in geometric modeling (GM) and computer-aided design (CAD). Given freeform parametric curves in $\mathbb{R}^2$ or surfaces in $\mathbb{R}^3$, these operators construct a surface or a trivariate, respectively, which interpolates the input on its boundary. The constructed geometry is considered optimal when the parameterization yields a constant Jacobian throughout the domain, a result that is impossible to achieve, in general. The parameterization is considered valid when the Jacobian is positive over the entire domain, sometimes excluding the boundaries. Then, the existence of an oriented normal field is ensured and robust integration using quadrature (interior) points is secured, among others.

Testing the validity of the parameterization of a planar surface is relatively simple. Given a planar C^1 continuous B-spline surface $S(u, v)$, the (unnormalized) normal field of S is computed as the cross product of B-spline functions Elber (1992), as $\overline{n}(u, v) = \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}$, where $\frac{\partial S}{\partial u}$ and $\frac{\partial S}{\partial v}$ are the two partial derivatives of S. Then, one can examine the coefficient of (the Z component of) $\overline{n}(u, v)$ and ensure they are all positive. By the convex hull property of B-spline functions Cohen et al. (2001), this is sufficient for $\overline{n}(u, v)$ to be positive throughout. Verifying the positivity of the Jacobian of a trivariate can follow a similar set of steps, and in this work, this symbolic approach is employed to verify the values that the Jacobian can assume.

The Boolean sum and ruling are simple and automatic, computationally efficient, operators, and are employed by many GM/CAD systems. Yet, they do not guarantee a valid parameterization. Cases of failing

*Corresponding author
Email addresses: haitam.f@campus.technion.ac.il (Haitham Fadila), qyoun.hong@gmail.com (Q Youn Hong), gershon@cs.technion.ac.il (Gershon Elber)

to produce a valid parameterization are quite common, and solutions are quite complex. This work is about improving the odds of (automatically) yielding a valid parameterization by computing the kernel of the input geometry, and exploiting this special location, if exists. See Figure 1, for examples.



(a) A cubic Bézier surface is constructed by a Boolean sum operator, with an invalid parameterization. See also (b).

(b) A cubic Bézier surface is constructed using the kernel-based Boolean sum operator with a valid parameterization. The surface's kernel is colored blue. Same input as (a).

(c) A quartic ruled Bézier surface with an invalid parameterization. See also (d).

(d) A quartic ruled Bézier surface with a valid parameterization, created using the kernel-based scheme. The surface's kernel is colored blue. Same input as (c).

Figure 1: Examples of invalid planar surfaces which have kernel points, constructed using regular Boolean sum and ruling operators, while the kernel-based variations of these constructors succeed in building planar surfaces with valid parameterizations.

A point P is said to be visible to a point Q if there exists a line segment between P and Q, that is not blocked (intersected) by any other geometry. If, for an object O, there is a point P so that all points on ∂O, the boundary of O, are visible to P, P is considered a kernel point of O. An object O with at least one kernel point is also denoted a star-shaped object. The kernel, if exists, is typically a convex region in O. A kernel point is an *interior kernel point* if it is in the open domain of the kernel.

As part of this work, we consider three geometric constructors: ruling, Boolean sum, and half Boolean sum, a simpler variation of the Boolean sum that was introduced in Masalha et al. (2021). We will first consider the case of planar input curves that construct a planar surface, and then, the spatial case where the input curves and/or surfaces in $\mathbb{R}^3$ synthesize a trivariate.

The rest of this work is organized as follows. Section 2 discusses related previous work, and the constructor of the half Boolean sum is briefly presented in Section 3. In Section 4, the algorithms for the kernel-based constructions are introduced, and in Section 5, some results are portrayed. Section 6 considers some extensions to our method, and finally, Section 7 concludes this effort and discusses possible future work.

2. Previous work

The kernel of a closed domain O is the loci of points in the interior of the domain, that are visible from every point on the boundary of the domain. If a point K belongs to the kernel of O, there always exists a straight line segment that connects K and an arbitrary point on ∂O, the boundary of O, while never intersecting with any part of the boundary geometry. If K is an interior kernel point, then a line from K is tangent to no point in ∂O.

The kernel is an important tool for the visibility analysis of geometry, and has been used in hidden surface removal and collision-free path generation in a locomotion problem Guibas et al. (1995), or locating lights in scene rendering Ghosh (2007). The kernel also provides information on the *star-shapedness* of geometry, which can be useful to determine the proper parameterization of a closed domain Jüttler et al. (2019).

The computation of the kernel has been thoroughly studied for polygons in computational geometry. In the context of kernel computation, a polygon consists of n ordered vertices with n directed edges connecting the consecutive vertices. Then, the kernel of a polygon is the common intersection of n half-planes located

along the edges. Shamos and Hoey (1976) proposed an $O(n \log n)$ algorithm to compute the kernel of simple polygons by intersecting these half-planes, and this intersection-based algorithm was improved by Lee and Preparata (1979) to $O(n)$. Preparata and Shamos (1985) also stated the kernel computation of three-dimensional polyhedra based on half-plane intersections, but in a theoretical sense. Recently, Sorgente et al. (2021, 2022) presented algorithms to compute the kernel of a polyhedron, by cutting the axis-aligned bounding box (AABB) of the polyhedron iteratively with the planes containing facets of the polyhedron.

For the kernel of curved shapes, Dobkin and Souvaine (1990) computed the kernel of a curvilinear polygon by modifying the kernel computation algorithm of a polygon. Elber et al. (2006), on the other hand, proposed a method to compute the kernel of fully smooth and continuous closed freeform curves and surfaces. They proved that the kernel of a closed C^1 curve is bounded by the envelopes of tangent lines of the curve. Additionally, the boundary of the kernel of a closed curve is determined only by tangent lines at inflection points of the curve. Multivariate polynomial equations were then formulated to compute the boundary of the kernel domain of a closed curve. Similar to the kernel of a closed curve, the kernel of a closed C^1 surface is bounded by the envelopes of tangent planes of the surface. The kernel of a closed surface was constructed by intersecting tangent plane patches sampled at parabolic points of the surface. Hong and Elber (2022) formulated inequality constraints that must be satisfied by the kernel of freeform curves and surfaces, based on the observation that a point P must be on the positive side of every tangent line/plane of a curve/surface if the point belongs to the kernel of the curve/surface, with inward normals. The verified regions of the solution of the inequality constraints constitute a conservative approximation of the kernel of the curve/surface, and were detected using a subdivision-based multivariate solver.

Polar parameterizations of star shapes utilize line segments as parameter lines to connect the center point with the boundary points. In Jüttler et al. (2019), circular arcs are used as parameter lines and the flexibility of underlying polar parameterization has been increased to be used for a wider class of domains, including star-shaped ones. Recently, Trautner et al. (2021) analyzed another generalization of polar parameterizations, which used parabolic arcs as parameter lines. These methods build regular planar surfaces, except at the center point, where it is singular. Also, the control mesh of these generalizations does not have a tensor-product structure.

Another way to build surfaces with a valid parameterization is by reparameterizing the input curves. In Cohen et al. (1997), a pair of curves are reparameterized by solving a tangent matching problem, such that the ruled surface between them has a valid parameterization, if one exists.

Recently, Atkinson et al. (2021) introduced a scheme to parameterize domains with smooth boundaries by calculating a diffeomorphism between an open unit disk and the domain boundary. First, a homotopy is constructed between the unit disk and the domain boundary. Then, the boundary of the homotopy is extended to the inside of the domain.

3. Half Boolean Sum

Following Masalha et al. (2021), given two adjacent freeform surfaces in $\mathbb{R}^3$, the half Boolean sum operator constructs a trivariate in which two of its boundaries interpolate the two input surfaces. Herein, we also consider a variant of this operator for a surface from two adjacent curves in $\mathbb{R}^2$ or $\mathbb{R}^3$, in Section 3.1, and a trivariate from two/three adjacent surfaces in $\mathbb{R}^3$, in Section 3.2.

3.1. A Surface Construction from Two Adjacent Curves

Given two adjacent freeform curves, $C_1(u)$ and $C_2(v)$, that share an endpoint location P, the half Boolean sum surface is constructed as:

$$S(u, v) = C_1(u) + C_2(v) - P.$$

See Figure 2 for one example.

Lemma 1. *$S(u, v)$ interpolates the input curves $C_1(u)$ and $C_2(v)$ on two of its boundaries.*

Proof. *Let $v = 0$. Then,*

$$S(u, 0) = C_1(u) + C_2(0) - P = C_1(u) + P - P = C_1(u),$$

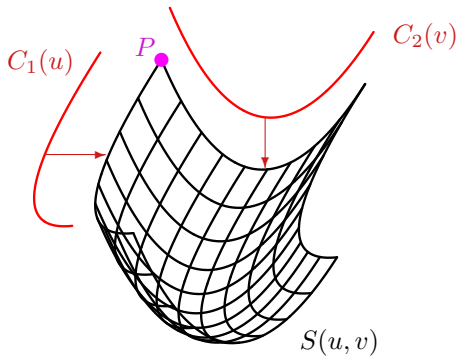


Figure 2: A half Boolean sum surface $S(u,v)$ constructed using two adjacent freeform curves in $\mathbb{R}^3$, as $S(u,v) = C_1(u) + C_2(v) - P$, where P is the curves' common endpoint. Note that the input curves are exploded (in red) a bit, for clarity.

and similarly for $S(0,v) = C_2(v)$.

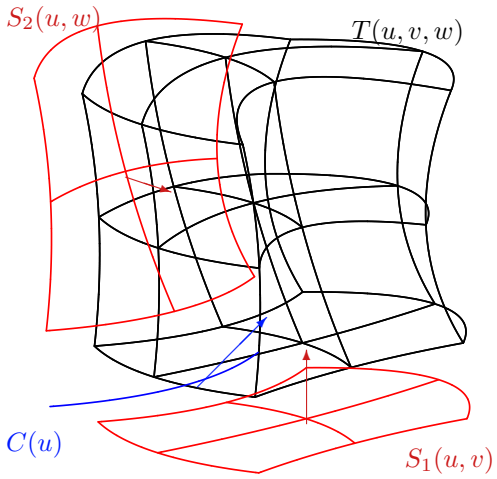
■

3.2. A Trivariate Construction from Two/Three Adjacent Surfaces

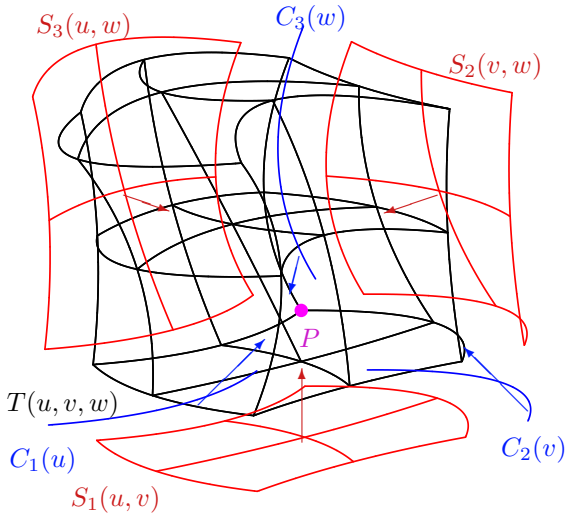
This operator was introduced by Masalha et al. (2021) for two surfaces sharing a boundary curve. Given two freeform surfaces, $S_1(u,v)$ *and* $S_2(u,w)$*, that share a common boundary curve,* $C(u)$*, the half Boolean sum trivariate is constructed as:*

$$T(u,v,w) = S_1(u,v) + S_2(u,w) - C(u).$$

See Figure 3 (a) for an example.



(a) A half Boolean sum trivariate $T(u,v,w)$ is constructed using two adjacent freeform surfaces in $\mathbb{R}^3$, S_1 and S_2. T is computed as $T(u,v,w) = S_1(u,v) + S_2(u,w) - C(u)$, where $C(u)$ is the common boundary curve of S_1 and S_2.



(b) A half Boolean sum trivariate $T(u,v,w)$ is constructed using three adjacent freeform surfaces in $\mathbb{R}^3$, S_1, S_2, and S_3. T is computed as $T(u,v,w) = S_1(u,v) + S_2(v,w) + S_3(u,w) - (C_1(u) + C_2(v) + C_3(w)) + P$, where $C_1(u)$, $C_2(v)$ and $C_3(w)$ are the common boundary curves of the respective surfaces, and P is the common endpoint of these three curves.

Figure 3: Constructions of trivariates from two/three adjacent surfaces. Note that the input surfaces are exploded a bit, for clarity.

Lemma 2. *$T(u,v,w)$ interpolates the input surfaces $S_1(u,v)$ and $S_2(u,w)$ on two of its boundaries.*

Proof. *Let $v=0$. Then,*

$$T(u,0,w) = S_1(u,0) + S_2(u,w) - C(u) = C(u) + S_2(v,w) - C(u) = S_2(v,w),$$

and similarly for $T(u,v,0) = S_1(u,v)$.

■

Given three freeform boundary surfaces $S_1(u,v)$, $S_2(v,w)$, and $S_3(u,w)$ that share boundary curves $C_1(u)$, $C_2(v)$, $C_3(w)$ so that P is the common endpoint of these curves, the half Boolean sum trivariate is constructed as:

$$T(u,v,w) = S_1(u,v) + S_2(v,w) + S_3(u,w) - (C_1(u) + C_2(v) + C_3(w)) + P.$$

See Figure 3 (b) for one example.

Lemma 3. *$T(u,v,w)$ interpolates the input surfaces $S_1(u,v)$, $S_2(v,w)$, and $S_3(u,w)$ on three of its boundaries.*

Proof. *Let $w=0$. Then,*

$$\begin{aligned} T(u,v,0) &= S_1(u,v) + S_2(v,0) + S_3(u,0) - (C_1(u) + C_2(v) + C_3(0)) + P \\ &= S_1(u,v) + C_2(v) + C_1(u) - (C_1(u) + C_2(v) + P) + P \\ &= S_1(u,v). \end{aligned} \tag{1}$$

and similarly for $T(u,0,w) = S_3(u,w)$ and $T(0,v,w) = S_2(v,w)$.

■

4. The Kernel-based Construction Algorithm

We assume the input is valid in the following senses. All parametric freeform inputs are regular and sufficiently differentiable and the topology they form together is valid. That is, intersection and self-intersection free, and the angles between tangents at adjacent locations (corners for planar surfaces) are less than 180 degrees (or otherwise the Jacobian will always be invalid at that corner).

Further, we assume that the input has open-end (clamped) end conditions, and only end control points affect the boundaries. In other words, the input curves can be either Bézier or open-end B-spline freeform curves or surfaces. An initial Bézier/B-spline surface S_0 or trivariate T_0 is constructed using the respective regular operator: a (half) Boolean sum or a ruling. If the parameterization of S_0/T_0 has a negative Jacobian in some regions in the constructed geometry, the approach we propose in this work is employed as follows:

- *The kernel of S_0/T_0 is computed.*
- *Assume at least one kernel point, K, is detected. If a whole kernel region is detected, the centroid of that region is computed as an interior kernel point K.*
- *An* interior control point *is a control point that does not affect the boundaries, under the open-end conditions' assumption. In other words, all control points that are not in the first nor in the last row/column{/depth} of the surface{/trivariate} of the control mesh. All internal control points of the control mesh of S_0/T_0 are moved to K, creating S_1/T_1.*
- *The parameterization of S_1/T_1 is examined for validity (positive Jacobian throughout) and if valid, quit.*

- *If the parameterization of S_1/T_1 is not valid, the result will be incrementally refined/degree raised, as S_i/T_i, before moving all internal control points toward K.*

Intuitively, consider the limiting case in which S_i has more interior knots than its degree, in each of its parametric axes. By moving all interior control points to K we end up with two types of polynomial patches in S_i:

1. *Surface patches where all control points are identically equal to K. Then, the Jacobian of the geometry in this interval is zero throughout.*
2. *Surface patches where all control points are identically equal to K except the first/last row/column of control points (which will all be on the boundary). Then, this patch is a ruling between K and that boundary region. By the fact the K is an interior kernel point, this ruling patch will almost always assume a valid parameterization. Toward the end of Section 5, we will show why it will not always be a valid parameterization geometry and how additional refinements successfully resolve that.*

With these steps and the added intuition for the knot insertion in the limit case, we now present the following result:

Theorem 4. *Consider a parametric surface S or trivariate T that was constructed using (half) Boolean sum/ruling of parametric input curves or surfaces, respectively, with at least one interior kernel point K, while some region in the domain has a negative Jacobian. In the limit, by raising the degrees of and/or (uniformly) refining S or T, and then moving all interior control points to K, a non-negative Jacobian throughout the domain must result. At K, the Jacobian might vanish.*

Proof. *See Appendix.* ■

We reiterate that by moving all interior control points to K, the Jacobian, J, might indeed vanish at K for B-spline based geometry, with more interior knots than the orders. For now, we will allow J to vanish at K and will propose a possible remedy in Section 6. Algorithm 1 presents the approach in detail, for kernel-based planar surface constructors.

5. Results

To estimate the success rate of the proposed kernel-based surface construction operators, we tested these constructors on a variety of randomized inputs. For each operator, we built three random datasets: cubic and quartic Bézier curves, and uniform quadratic open-end B-spline curves with five control points. We have three kernel-based operators, i.e., half Boolean sum, Boolean sum, and ruling. In total, we have nine different random datasets.

All the freeform samples in the different datasets were verified to have a valid kernel domain, and the constructed surfaces via the regular operators were found to have an invalid parameterization. Clearly, these datasets are not unique, and they only exemplify the performance of the presented kernel-based construction approach.

In Section 5.1, we discuss the generation scheme of the random datasets of the kernel-based planar surface constructor. Then, in Sections 5.2 to 5.4, we present, for each operator, its success rate in constructing a surface with a valid parameterization on these datasets, while introducing additional degrees of freedom to the input curves, either by degree raising or by refinement.

5.1. The Random Datasets for the Kernel-based Planar Surface Constructors

Each random dataset of the kernel-based planar surfaces' constructor is composed of one thousand samples of random freeform curves in $\mathbb{R}^2$ with a valid kernel domain, such that the regular operator fails to build a surface with a valid parameterization. In addition, the samples are validated to be (self-)intersection free, and adjacent geometry is angularly validated (e.g., at corners).

Algorithm 1 Planar kernel-based Surface Construction Scheme.

Input:

$\mathcal{C}$ - The input freeform curves in $\mathbb{R}^2$;
$\mathcal{T}$ - The surface constructor: half Boolean sum, Boolean sum, or ruled surface;
$AddDOF$ - The method to add degrees of freedom: either *Degree Raising* or *Refinement*;
$MaxDOF$ - Bounds on added degrees of freedom;

Output:

S_i - Constructed surface using constructor $\mathcal{T}$, with a valid parameterization, or *None* if failed;

Algorithm:

$S_0 := \mathcal{T}(\mathcal{C})$;
if S_0 has a valid parameterization **then**
  return S_0;
end if
$\mathcal{K}$:= a kernel of S_0, if any;
if $\mathcal{K} = \emptyset$ **then**
  return None;
end if
K := Interior kernel point as centroid of $\mathcal{K}$;
for $i = 0$; $i < MaxDOF$; $i++$ **do**
  if $AddDOF$ == *Refinement* **then**
    $\overline{S_i} := S_0$ refined with i uniform knots in each interval;
  else
    $\overline{S_i} := S_0$ degree raised i times;
  end if
  $S_i :=$ moving all inner control points of $\overline{S_i}$ to K;
  if S_i has a valid parameterization **then**
    return S_i;
  end if
end for
return *None*;

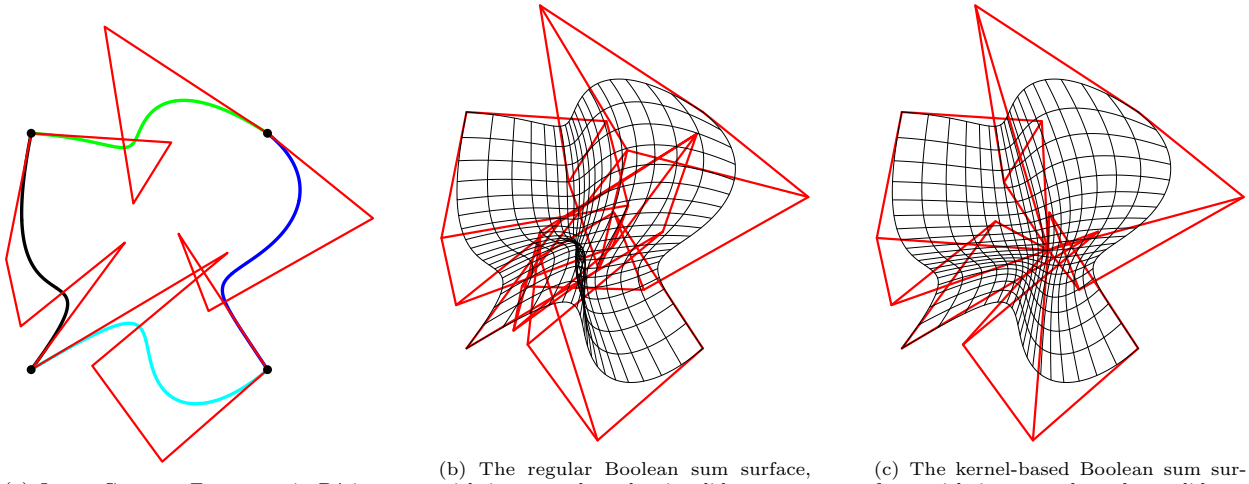
For each sample in the datasets of the Boolean sum operator, we generate four random curves: left, right, bottom and top. The control points of the curves were generated around $[-1, 1]^2$. *The first and the last control points of the left curve are* $(-1, -1)$ *and* $(-1, 1)$, *and its inner control points are generated randomly in* x *between* $(-2, 0)$ *and in* y *between* $(-2, 2)$. *The other three curves were randomly generated in a similar way.*

Similarly, we generate two random curves, bottom and top, for each sample in the datasets of the ruling operator. The ruled surface is formed between these two curves, while the ruling direction is initiated with three control points and degree two to provide initial degrees of freedom for the kernel-based constructor. Left and top curves are similarly generated, for each sample in the datasets of the half Boolean sum.

The kernel for each sample in a kernel-based planar surfaces' constructor is computed using the conservative approach of Hong and Elber (2022), and the boundary of the regular surface constructor (S_0 *in Algorithm 1) is given as an input to the kernel computation.*

5.2. Boolean Sum Surfaces

Figures 4 and 5 show examples of the construction of a kernel-based Boolean sum surface with a valid parameterization, for quartic Bézier and quadratic B-spline input curves. Figures 6 and 7 present the success rate of the kernel-based Boolean sum constructor, on the operator's three random datasets (cubic and quartic Bézier and quadratic B-spline), using the two different methods to add degrees of freedom to the input curves (degree raising and refinements). In Figures 6 and 7 and hence after and in all graphs, the x-axis

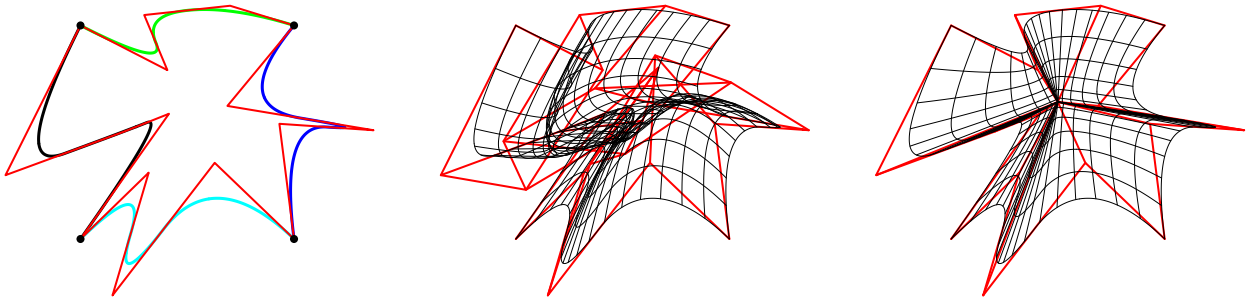


(a) Input Curves. Four quartic Bézier curves.

(b) The regular Boolean sum surface, with its control mesh - invalid parameterization.

(c) The kernel-based Boolean sum surface, with its control mesh - valid parameterization.

Figure 4: Quartic Bézier (kernel-based) Boolean sum surface.



(a) Input Curves. Four open-end quadratic B-spline curves, each with five control points and a uniform knot vector.

(b) The regular Boolean sum surface, with its control mesh - invalid parameterization.

(c) The kernel-based Boolean sum surface, with its control mesh - valid parameterization.

Figure 5: Quadratic B-spline (kernel-based) Boolean sum surface.

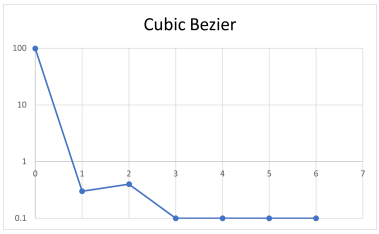
represents the iteration number of adding i degrees of freedom (via degree raising or refinement), and the y-axis represents the success rate in the i iteration (of cases that failed until the i iteration), in percentages, and exploiting a log scale. Figure 6 utilizes degree raising, and Figure 7 utilizes refinement.

5.3. Half Boolean Sum

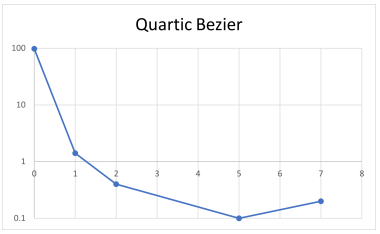
Figure 8 shows an example of constructing a kernel-based half Boolean sum surface with a valid parameterization for quadratic B-spline curves. Figures 9 and 10 display the success rate of the kernel-based half Boolean sum constructor, on the operator's three random datasets, and using the two ways to add degrees of freedom to the input curves. Figure 9 utilizes degree raising, and Figure 10 utilizes refinement.

5.4. Ruled Surfaces

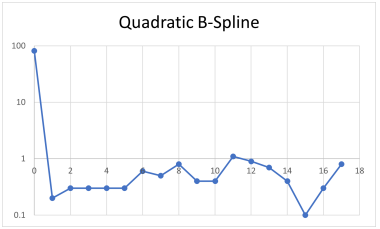
Figure 11 shows an example of a kernel-based ruled surface with a valid parameterization for quadratic B-spline curves, after adding a degree of freedom to the input curves by utilizing the refinement operator for one iteration. Figure 12 shows another example of a kernel-based ruled surface with a valid parameterization for quartic Bézier curves, after adding degrees of freedom to the input curves either by utilizing the refinement operator for one iteration or by utilizing the degree-raising operator for two iterations.



(a) Cubic Bézier curves - success rate per iteration. Success without degree raising (1^{st} iteration): 98.8%, accumulated (all iterations): 99.9%.

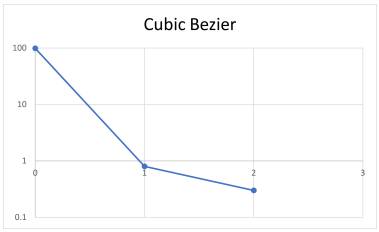


(b) Quartic Bézier curves - success rate per iteration. Success without degree raising (1^{st} iteration): 97.3%, accumulated (all iterations): 99.4%.

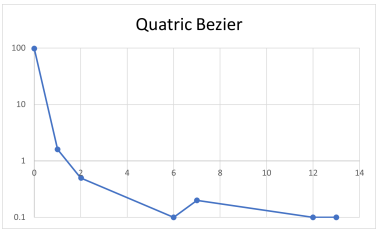


(c) Quadratic B-spline curves with five control points and a uniform knot vector - success rate per iteration. Success without degree raising (1^{st} iteration): 81.7%, accumulated (all iterations): 90.1%.

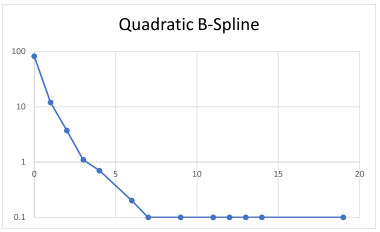
Figure 6: These graphs present the kernel-based Boolean sum operator's success rates in constructing surfaces with a valid parameterization, while the regular Boolean sum operator failed to build a surface with a valid parameterization, and the input has at least one kernel point. The degree-raising operator has been used to add more degrees of freedom to the input curves. Maximal degree: 17.



(a) Cubic Bézier curves - success rate per iteration. Success without knot insertion (1^{st} iteration): 98.8%, accumulated (all iterations): 99.9%.



(b) Quartic Bézier curves - success rate per iteration. Success without knot insertion (1^{st} iteration): 97.3%, accumulated (all iterations): 99.9%.



(c) Quadratic B-spline curves with five control points and a uniform knot vector - success rate per iteration. Success without knot insertion (1^{st} iteration): 81.7%, accumulated (all iterations): 100%.

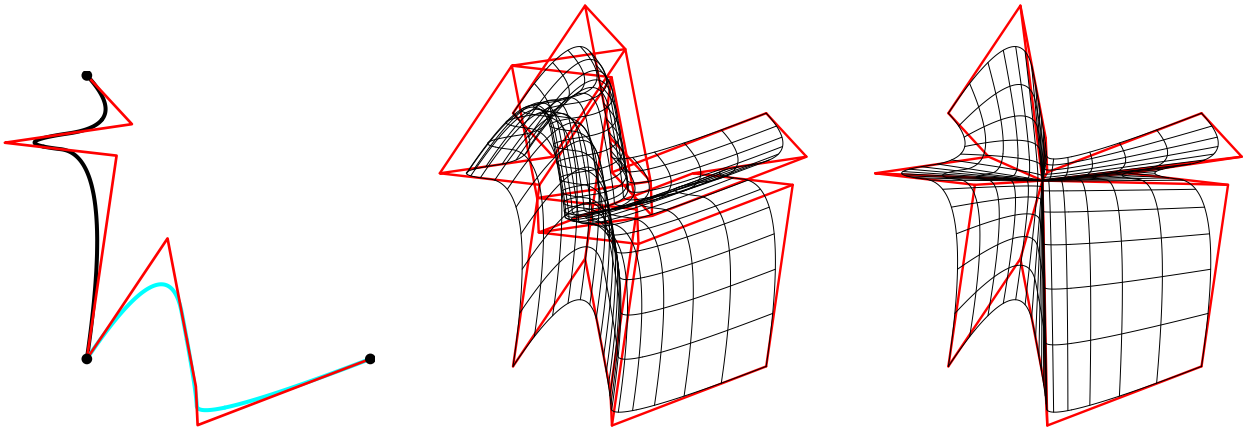
Figure 7: These graphs present the success rates of the kernel-based Boolean sum operator in constructing surfaces with valid parameterization, while the regular Boolean sum operator failed to build a surface with a valid parameterization, and the input has at least one kernel point. The refinement operator has been used to add more degrees of freedom to the input curves. Maximal refinement: 20 knots per knot interval.

Figures 13 and 14 display the success rate of the kernel-based ruling construction, on the operator's three random datasets, employing the two methods to add degrees of freedom to the input curves. Figure 13 utilizes degree raising, and Figure 14 utilizes refinement.

Interestingly, the success rate of constructing a valid quadratic B-spline ruled surface, without adding degrees of freedom to the input curves is quite small, as can be seen in Figures 13 (c) and 14 (c). While the success rate clearly depends on the randomization of the datasets, and because a kernel point exists, after enough degree raisings (herein up to degree 40!), the success rate for this case is still over 80%.

Our initial intuition to Theorem 4 stated that once we have more than degree knots, we will have only two types of patches, those with all control points at K, or those with one boundary row/column of control points and the rest of the control points at located at K. While true, under some conditions that we now portray, these resulting ruling patches can still possess negative Jacobians. This explains the need to introduce more than degree knots, under rare conditions, as some of the graphs in this chapter show.

Consider the construction using Boolean sum in Figure 15. The input consists of quadratic B-spline curves so after the insertion of two knots (or more), all patches are of the above mentioned two types. However, a (corner) patch, especially if K is very close to the boundary/corner, might be created with one of its corners forming more than 180 degrees, as seen in Figure 15 (a). Additional refinements, more than the degree, are required to reduce the span of the boundary/corner patch, until no such >180 degrees corner can be formed in this patch. See Figure 15 (b).

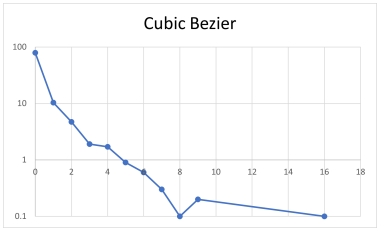


(a) Input Curves. Two open-end quadratic B-spline curves with five control points and a uniform knot vector.

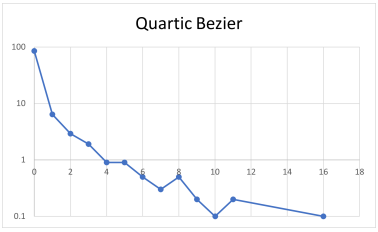
(b) The regular half Boolean sum surface, with its control mesh - invalid parameterization.

(c) The kernel-based half Boolean sum surface, with its control mesh - valid parameterization.

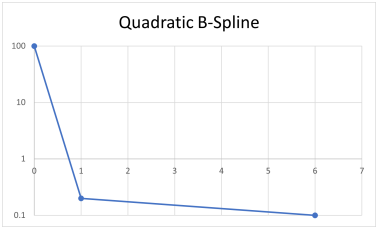
Figure 8: Quadratic B-spline (kernel-based) half Boolean sum surface.



(a) Cubic Bézier curves - success rate per iteration. Success without degree raising (1^{st} iteration): 79.2%, accumulated (all iterations): 100%.



(b) Quartic Bézier curves - success rate per iteration. Success without degree raising (1^{st} iteration): 85%, accumulated (all iterations): 99.9%.



(c) Quadratic B-spline curves with five control points and a uniform knot vector - success rate per iteration. Success without degree raising (1^{st} iteration): 99.7%, accumulated (all iterations): 100%.

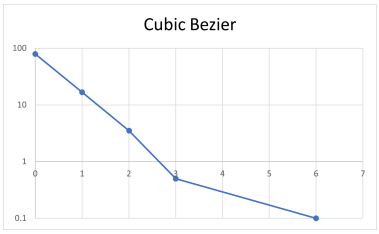
Figure 9: These graphs present the kernel-based half Boolean sum operator's success rates in constructing surfaces with a valid parameterization, while the regular half Boolean sum operator failed to build a surface with a valid parameterization, and the regular half Boolean sum surface has at least one kernel point. The degree-raising operator has been used to add more degrees of freedom to the input curves. Maximal degree: 17.

Hinting on the expected computation costs, all the presented examples in this work were computed in not more than a few seconds on a Windows 10 modern workstation. The computation of the kernel took, on average, from 10 milliseconds for surfaces to about 10 seconds for trivariates. Then, for example, the presented kernel-based Boolean sum operator for surface ranged up to a few seconds in the first ten iterations of adding degree of freedom by either degree raising or refinement.

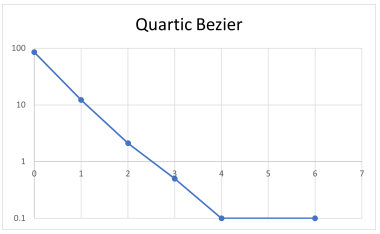
6. Further extending the Kernel-based Surface Construction Algorithm

So far, in all presented results, the Jacobian might vanish at K, especially in the kernel-based B-spline construction cases. As stated, one can clearly have n consecutive interior control points all located at K, where n is larger than the degree of the spline, and as a result, that interval will present zero velocity. To overcome this limitation, we propose to organize the interior control points in a uniform grid, then scale it down and translate it so it is centered at K. See Figure 16, for one example.

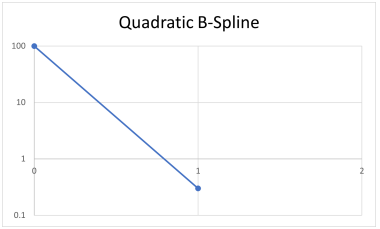
By utilizing this approach, no adjacent control points will be identical and as a result, partial derivatives will not vanish. The size of this scaled-down grid immediately affects the magnitude of the Jacobian there. Yet, a grid too large might result in an invalid parameterization.



(a) Cubic Bézier curves - success rate per iteration. Success without knot insertion (1^{st} iteration): 79.2%, accumulated (all iterations): 100%.

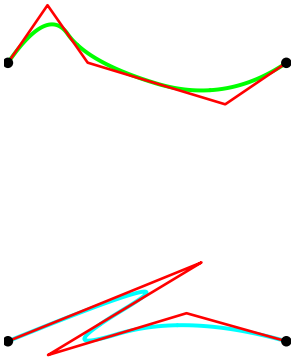


(b) Quartic Bézier curves - success rate per iteration. Success without knot insertion (1^{st} iteration): 85%, accumulated (all iterations): 100%.

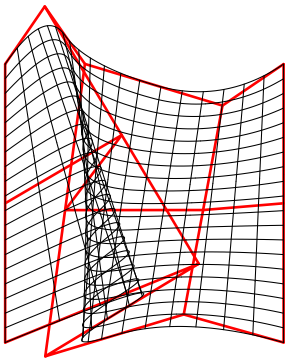


(c) Quadratic B-spline curves with five control points and a uniform knot vector - success rate per iteration. Success without knot insertion (1^{st} iteration): 99.7%, accumulated (all iterations): 100%.

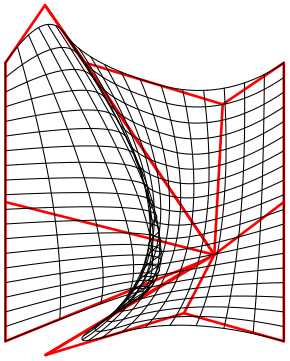
Figure 10: These graphs present the success rates of the kernel-based half Boolean sum operator in constructing surfaces with valid parameterization, while the regular half Boolean sum operator failed to build a surface with a valid parameterization, and the regular half Boolean sum surface has at least one kernel point. The refinement operator has been used to add more degrees of freedom to the input curves. Maximal refinement: 20 knots per knot interval.



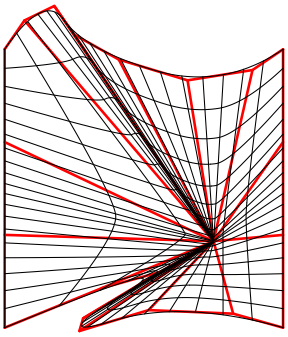
(a) Input Curves. Two open-end quadratic B-spline curves with five control points and a uniform knot vector.



(b) The regular ruled surface, with its control mesh - invalid parameterization.



(c) The kernel-based ruled surface, with its control mesh (and no refinement/degree raising) - invalid parameterization.



(d) The kernel-based ruled surface after one iteration of refinement, with its control mesh - valid parameterization.

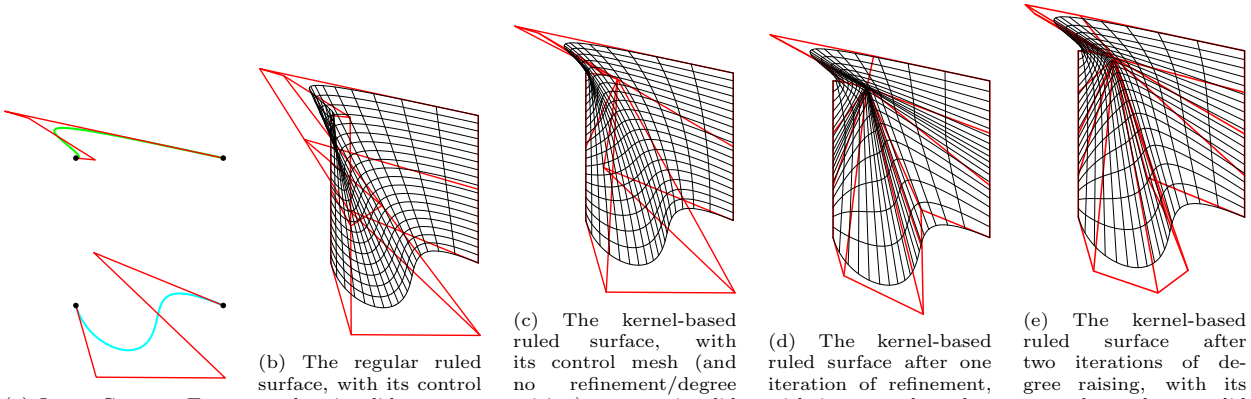
Figure 11: Quadratic B-spline (kernel-based) ruled surface.

The extension of Algorithm 1 to constructors of trivariates in $\mathbb{R}^3$, is immediate - replace every 'surface' in Algorithm 1 with a 'trivariate' and every 'curve' with a 'surface'. Some examples with valid parameterizations of kernel-based constructed trivariates, where the regular constructors yielded invalid parameterizations, are presented in Figures 17 to 19.

7. Conclusion and future work

In this work, we aimed to enhance the validity performance of common freeform construction operators. We handled cases where the regular constructors failed to build a valid surface or trivariate, and a kernel can be found in the domain. While the interior parameterization of Boolean sum can be general, ruled surfaces are expected to have linear ruling lines. Clearly, we are no longer preserving the linear ruling parameterization in this work while we aim to preserve the boundaries of the ruled surfaces, while constructing a valid parameterization. Further, we can potentially utilize our kernel-based construction scheme for other geometric construction operators.

As shown in the experimental results, the presented approach has an improved success rate, as more degrees of freedom are being added to the input. Not surprisingly, the refinement operator typically converges faster than the degree-raising scheme, most likely because refinement induces more locality. For instance,



(a) Input Curves. Two quartic Bézier curves.

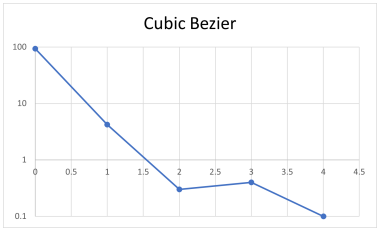
(b) The regular ruled surface, with its control mesh - invalid parameterization.

(c) The kernel-based ruled surface, with its control mesh (and no refinement/degree raising) - invalid parameterization.

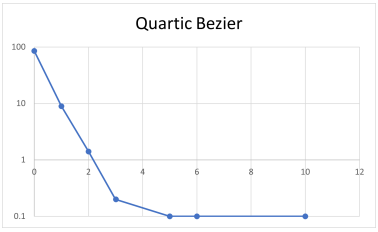
(d) The kernel-based ruled surface after one iteration of refinement, with its control mesh - valid parameterization.

(e) The kernel-based ruled surface after two iterations of degree raising, with its control mesh - valid parameterization.

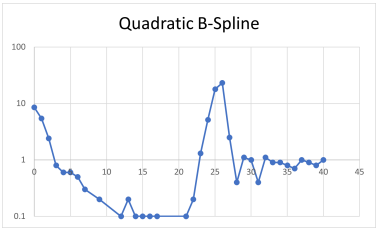
Figure 12: Quartic Bézier (Kernel-based) ruled surface.



(a) Cubic Bézier curves - success rate per iteration. Success without degree raising (1^{st} iteration): 93%, accumulated (all iterations): 98%.



(b) Quartic Bézier curves - success rate per iteration. Success without degree raising (1^{st} iteration): 84.8%, accumulated (all iterations): 95.6%.



(c) Quadratic B-spline curves with five control points and a uniform knot vector - success rate per iteration. Success without degree raising (1^{st} iteration): 8.5%, accumulated (all iterations): 81.3%.

Figure 13: These graphs present the kernel-based ruling operator's success rates in constructing surfaces with a valid parameterization, while the regular ruling operator failed to build a surface with a valid parameterization, and the regular ruled surface has at least one kernel point. The degree-raising operator has been used to add more degrees of freedom to the input curves. Cubic and quartic Bézier maximal degree: 17, and quadratic B-spline maximal degree: 40.

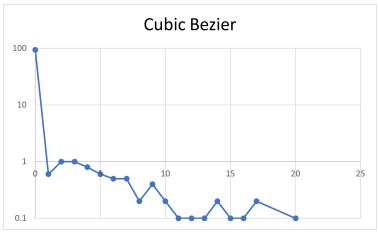
this behavior is evident in Figure 19 (c) where the knot insertion resulted with linear iso-parametric curves that move from the boundaries to the kernel location.

The distribution of the interior control points in the vicinity of the kernel, as discussed in Section 6 (Figure 16), deserves additional attention. Similarly, one can examine the operators' performance with a more careful selection of the interior kernel location, a degree of freedom we did not exercise in this work.

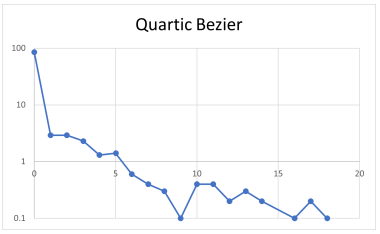
Work has been previously done, using local numeric optimization of relocating interior control points, to reduce the variance of the Jacobian as well as aiming at ensuring its validity. It is typically simpler to optimize a problem starting from a valid solution (whole positive Jacobian) than starting from an invalid answer. Hence, we expect that an intertwined solution of a kernel-based initial constructor, followed by an optimization postprocess, such as Xu et al. (2013), could yield a superior result.

Acknowledgements

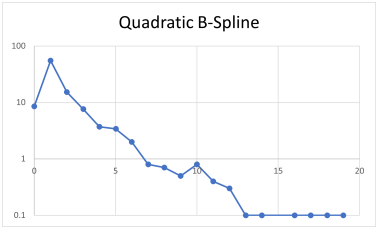
We would like to thank the anonymous reviewers for their comments and suggestions. This project has received funding in part by from the European Union Horizon 2020 research and innovation programme, under Grant Agreement No. 862025.



(a) Cubic Bézier curves - success rate per iteration. Success without knot insertion (1^{st} iteration): 93%, accumulated (all iterations): 99.8%.

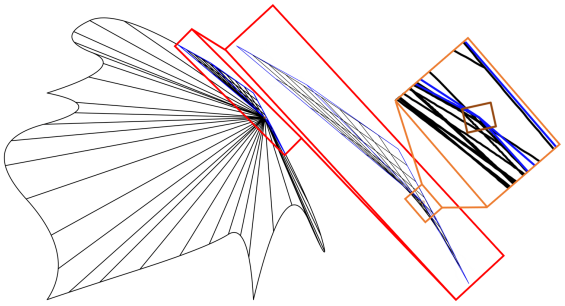


(b) Quartic Bézier curves - success rate per iteration. Success without knot insertion (1^{st} iteration): 84.8%, accumulated (all iterations): 98.9%.

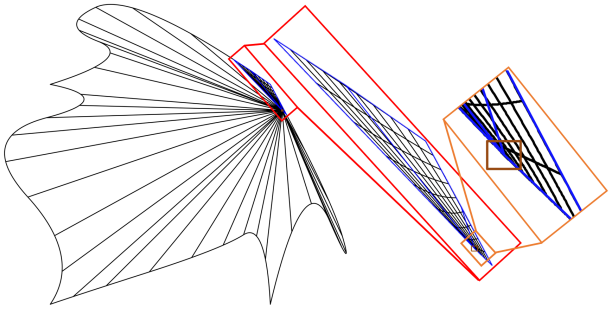


(c) Quadratic B-spline curves with five control points and a uniform knot vector - success rate per iteration. Success without knot insertion (1^{st} iteration): 8.5%, accumulated (all iterations): 99.4%.

Figure 14: These graphs present the success rates of the kernel-based ruling operator in constructing surfaces with valid parameterization, while the regular ruling operator failed to build a surface with a valid parameterization, and the ruled surface has at least one kernel point. The refinement operator has been used to add more degrees of freedom to the input curves. Maximal refinement: 20 knots per knot interval.



(a) Four boundary quadratic B-spline curves yielded this result (after two refinement iterations) with negative Jacobian at the top right patch (in blue), as it has one corner that spans more than 180 degrees.



(b) the same input from (a) yielded this result (after six refinement iterations) with positive Jacobian throughout the surface. The corner patch from (a) now spans less boundary and that corner is just below 180 degrees.

Figure 15: A kernel-based Boolean sum quadratic surface with invalid parameterization, even after two knots that were inserted (a). More knot insertion iterations are required to reach a whole positive Jacobian (b).

References

Atkinson, K., Chien, D., Hansen, O., 2021. Constructing diffeomorphisms between simply connected plane domains. Electronic Transactions on Numerical Analysis 55, 671–686.

Cohen, E., Riesenfeld, R.F., Elber, G., 2001. Geometric modeling with splines. A. K. Peters, New York.

Cohen, S., Elber, G., Bar-Yehuda, R., 1997. Matching of freeform curves. Computer-Aided Design 29, 369–378.

Dobkin, D.P., Souvaine, D.L., 1990. Computational geometry in a curved world. Algorithmica 5, 421–457.

Elber, G., 1992. Free form surface analysis using a hybrid of symbolic and numeric computation. Ph.D. thesis. University of Utah.

Elber, G., Johnstone, J.K., Kim, M.S., Seong, J.K., 2006. The kernel of a freeform surface and its duality with the convex hull of its tangential surface. International Journal of Shape Modeling 12, 129–142.

Ghosh, S.K., 2007. Visibility algorithms in the Plane. Cambridge University Press.

Guibas, L., Motwani, R., Raghavan, P., 1995. The robot localization problem. SIAM Journal on Computing 26, 1120–1138.

Hong, Q.Y., Elber, G., 2022. Detection and computation of conservative kernels of models consisting of freeform curves and surfaces, using inequality constraints. Computer Aided Geometric Design 94, 102075.

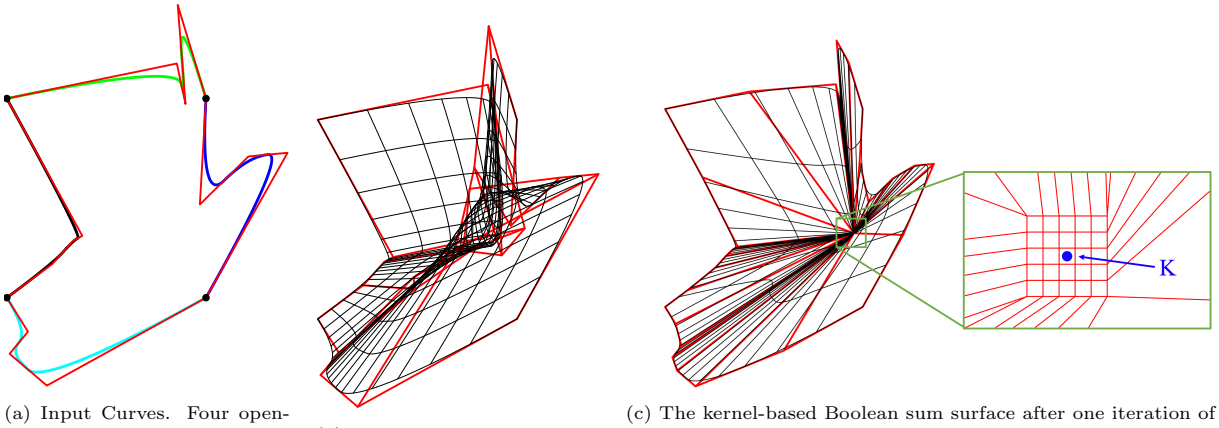
Jüttler, B., Maroscheck, S., Kim, M.S., Hong, Q.Y., 2019. Arc fibrations of planar domains. Computer Aided Geometric Design 71, 105–118.

Lee, D.T., Preparata, F.P., 1979. An optimal algorithm for finding the kernel of a polygon. J. ACM 26, 415–421.

Masalha, R., Cirillo, E., Elber, G., 2021. Heterogeneous parametric trivariate fillets. Computer Aided Geometric Design 86, 101970.

Preparata, F.P., Shamos, M.I., 1985. Computational geometry: an introduction. Springer-Verlag, Berlin, Heidelberg.

Shamos, M.I., Hoey, D., 1976. Geometric intersection problems, in: 17th Annual Symposium on Foundations of Computer Science (sfcs 1976), pp. 208–215.

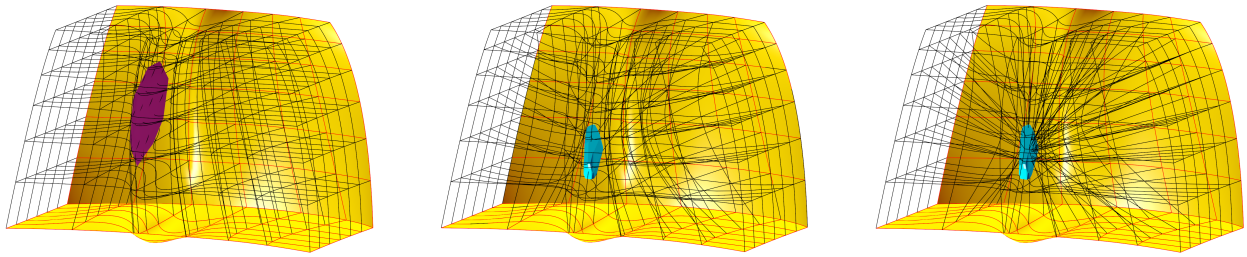


(a) Input Curves. Four open-end quadratic B-spline curves, each with five control points and a uniform knot vector.

(b) The regular Boolean sum surface, with its control mesh - invalid parameterization.

(c) The kernel-based Boolean sum surface after one iteration of refinement is shown on the left, with its control mesh colored in red - positive Jacobian throughout the domain. On the right, a zoom in near the surface's kernel point K is shown, which is colored in blue.

Figure 16: A Quadratic B-spline (kernel-based) Boolean sum surface with a positive Jacobian throughout, including at K.



(a) The regular B-spline half Boolean sum trivariate, of orders $(3, 4, 3)$ and control mesh lengths $(3, 9, 3)$ results in an invalid parameterization. The iso-curves are colored black, and the zero level-set of the Jacobian is shown in purple.

(b) The kernel-based B-spline half Boolean sum trivariate. One iteration of degree raising, now with orders $(4, 5, 4)$ and control mesh lengths $(4, 13, 4)$, yields a valid parameterization. The iso-curves are colored black, and the kernel is shown in cyan.

(c) The kernel-based B-spline half Boolean sum trivariate. One iteration of refinement, now with orders $(3, 4, 3)$ and control mesh lengths $(4, 13, 4)$, yields a valid parameterization. The iso-curves are colored black, and the kernel is shown in cyan.

Figure 17: A B-spline trivariate created using a (kernel-based) half Boolean sum of two surfaces (back and bottom faces) that are shown in yellow.

Sorgente, T., Biasotti, S., Spagnuolo, M., 2021. A geometric approach for computing the kernel of a polyhedron, in: Frosini, P., Giorgi, D., Melzi, S., Rodolà, E. (Eds.), Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference, The Eurographics Association.

Sorgente, T., Biasotti, S., Spagnuolo, M., 2022. Polyhedron kernel computation using a geometric approach. Computers & Graphics 105, 94–104.

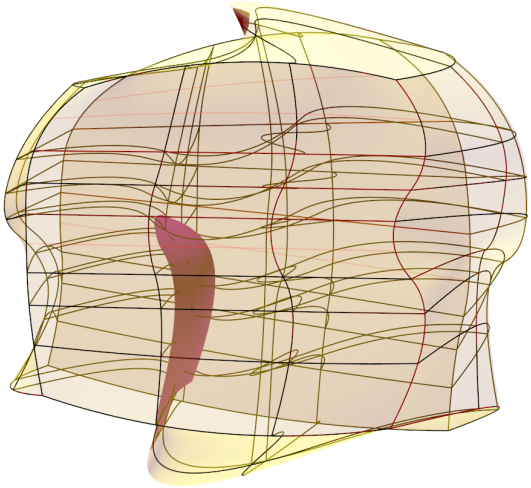
Trautner, S., Jüttler, B., Kim, M.S., 2021. Representing planar domains by polar parameterizations with parabolic parameter lines. Computer Aided Geometric Design 85, 101966.

Xu, G., Mourrain, B., Duvigneau, R., Galligo, A., 2013. Optimal analysis-aware parameterization of computational domain in 3d isogeometric analysis. Computer-Aided Design 45, 812–821.

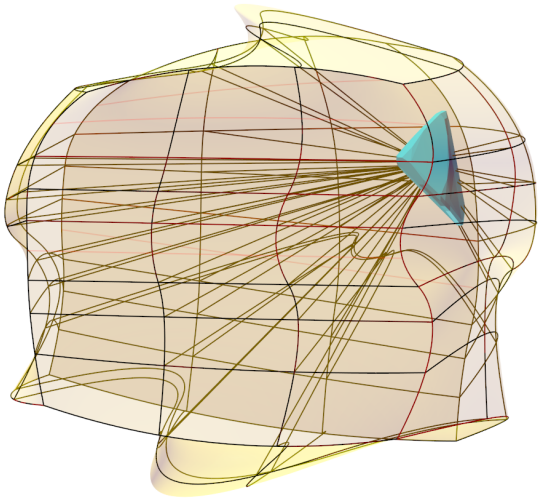
Appendix A. Proof of Theorem 4

We prove Theorem 4 for the Boolean sum operator which is the most complex case, and input Bézier curves under degree raising, the less obvious case between degree raising and refinement. The other construction cases follow similar lines.

Let $S(u, v) = \sum_{j=0}^{n} \sum_{i=0}^{n} P_{ij} \theta_{i,n}(u) \theta_{j,n}(v)$ *be a Bézier surface, which is constructed using the kernel-based Boolean sum operator on four regular* C^1 *continuous Bézier curves that do not intersect and form a topological rectangle. Denote its four regular top, bottom, left and right boundary curves as* $B(v) = \sum_{j=0}^{n} P_{0,j} \theta_{j,n}(v)$,

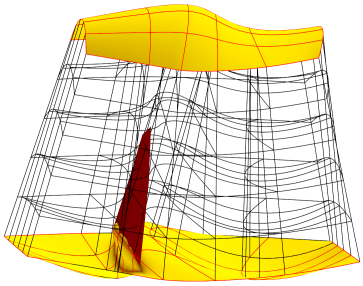


(a) The regular B-spline Boolean sum trivariate, with orders $(3, 4, 3)$ and control mesh lengths $(5, 17, 5)$ yields an invalid parameterization. The iso-curves are colored black, and the zero level-set of the Jacobian is shown in purple (note the two disjoint regions of the zero level-set).

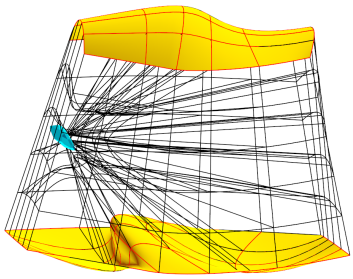


(b) The kernel-based B-spline Boolean sum trivariate. With no refinement/degree raising, so its orders and mesh lengths are as in (a), the kernel-based constructor yields a valid parameterization. The iso-curves are colored black, and the kernel is shown in cyan.

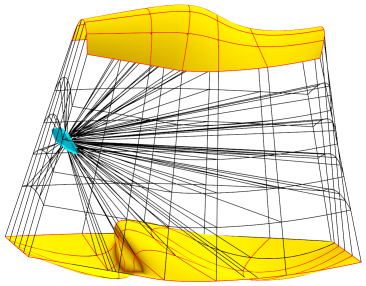
Figure 18: A B-spline trivariate created using (kernel-based) Boolean Sum of six surfaces shown transparent in yellow.



(a) The regular B-spline ruled trivariate, of orders $(4, 3, 3)$ and control mesh lengths $(9, 3, 3)$, yields an invalid parameterization. The iso-curves are colored black, and the zero level-set of the Jacobian in purple.



(b) The kernel-based B-spline ruled trivariate after five iterations of degree raising, so its orders are $(9, 8, 8)$ and control mesh lengths $(29, 8, 8)$, yields a valid parameterization. The iso-curves are colored in black, and the kernel is shown in cyan.



(c) The kernel-based B-spline ruled trivariate after three iterations of refinements, with the same orders as in (a) and control mesh lengths $(30, 6, 6)$, yields a valid parameterization. The iso-curves are colored black, and the kernel is shown in cyan.

Figure 19: A B-spline trivariate created using (kernel-based) ruling between two surfaces (bottom and bottom faces) that are shown in yellow.

$T(v) = \sum_{j=0}^{n} P_{n,j}\theta_{j,n}(v)$, $L(u) = \sum_{i=0}^{n} P_{i,0}\theta_{i,n}(u)$ *and* $R(u) = \sum_{i=0}^{n} P_{i,n}\theta_{i,n}(u)$, *and let* K *be an interior kernel point for* B, T, L, R.

The surface u *partial derivative:*

$$\begin{aligned}\frac{\partial S(u,v)}{\partial u} &= \frac{\partial}{\partial u}\left(\sum_{j=0}^{n}\sum_{i=0}^{n} P_{i,j}\theta_{i,n}(u)\theta_{j,n}(v)\right) \\ &= \sum_{j=0}^{n}\frac{\partial}{\partial u}\left(\sum_{i=0}^{n} P_{i,j}\theta_{i,n}(u)\right)\theta_{j,n}(v)\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j=1}^{n-1} \frac{\partial}{\partial u}\left(\sum_{i=0}^{n} P_{i,j}\theta_{i,n}(u)\right)\theta_{j,n}(v) + \\
&\qquad \frac{\partial}{\partial u}\left(\sum_{i=0}^{n} P_{i,0}\theta_{i,n}(u)\right)\theta_{0,n}(v) + \frac{\partial}{\partial u}\left(\sum_{i=0}^{n} P_{i,n}\theta_{i,n}(u)\right)\theta_{n,n}(v) \\
&= \sum_{j=1}^{n-1}\left(\sum_{i=0}^{n-1} n(P_{i+1,j}-P_{i,j})\theta_{i,n-1}(u)\right)\theta_{j,n}(v) + \frac{\partial L(u)}{\partial u}\theta_{0,n}(v) + \frac{\partial R(u)}{\partial u}\theta_{n,n}(v) \\
&= \sum_{j=1}^{n-1} n\Big((K-P_{0,j})\theta_{0,n-1}(u) + (P_{n,j}-K)\theta_{n-1,n-1}(u)\Big)\theta_{j,n}(v) + \frac{\partial L(u)}{\partial u}\theta_{0,n}(v) + \frac{\partial R(u)}{\partial u}\theta_{n,n}(v),
\end{aligned}
\tag{A.1}
$$

as the differences between the interior control points vanish, being all equal to K. $\frac{\partial S(u,v)}{\partial v}$ can be computed in a similar way:

$$
\frac{\partial S(u,v)}{\partial v} = \sum_{i=1}^{n-1} n\Big((K-P_{i,0})\theta_{0,n-1}(v) + (P_{i,n}-K)\theta_{n-1,n-1}(v)\Big)\theta_{i,n}(u) + \frac{\partial B(v)}{\partial v}\theta_{0,n}(u) + \frac{\partial T(v)}{\partial v}\theta_{n,n}(u).
$$

We first show that the normal does not vanish on the boundary of the surface. Without loss of generality, we assume boundary $v=0$ and let $u \in (0,1)$, excluding the cases of $u=0$ or $u=1$, as we assume the input is valid (at the corners):

$$
\begin{aligned}
\frac{\partial S(u,0)}{\partial u} &= \frac{\partial L(u)}{\partial u}, \\
\frac{\partial S(u,0)}{\partial v} &= \sum_{i=1}^{n-1} n(K-P_{i,0})\theta_{i,n}(u) + \frac{\partial B(0)}{\partial v}\theta_{0,n}(u) + \frac{\partial T(0)}{\partial v}\theta_{n,n}(u) \\
&= \sum_{i=1}^{n-1} n(K-P_{i,0})\theta_{i,n}(u) \\
&= \sum_{i=0}^{n} nK\theta_{i,n}(u) - \sum_{i=0}^{n} nP_{i,0}\theta_{i,n}(u) \\
&= n(K-L(u)),
\end{aligned}
\tag{A.2}
$$

as in the limit for $n \to \infty$, $\theta_{0,n}(u) = \theta_{n,n}(u) \to 0$ for $u = (0,1)$.

The (unnormalized) surface normal equals $\overline{n}(u,v) = \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}$, or in the limit, again,

$$
\begin{aligned}
\overline{n}(u,0) &= \frac{\partial S(u,0)}{\partial u} \times \frac{\partial S(u,0)}{\partial v} \\
&= n\frac{\partial L(u)}{\partial u} \times \Big(K - L(u)\Big),
\end{aligned}
\tag{A.3}
$$

Now, because K is an interior kernel location, the vector $(K-L(u))$ is never tangent to the boundary, and specifically never tangent to $L(u)$, or $(K-L(u))$ and $\frac{\partial L(u)}{\partial u}$ are never parallel. In other words, $\overline{n}(u,v)$ cannot vanish on that boundary. The proof for the other three boundaries is similar.

Now, assume $u, v \in (0,1)$. Then:

$$
\frac{\partial S(u,v)}{\partial u} = \sum_{j=1}^{n-1} n\Big((K-P_{0,j})\theta_{0,n-1}(u) + (P_{n,j}-K)\theta_{n-1,n-1}(u)\Big)\theta_{j,n}(v) + \frac{\partial L(u)}{\partial u}\theta_{0,n}(v) + \frac{\partial R(u)}{\partial u}\theta_{n,n}(v)
$$

$$
\begin{aligned}
&= n\theta_{0,n-1}(u)\sum_{j=1}^{n-1}(K-P_{0,j})\theta_{j,n}(v)+n\theta_{n-1,n-1}(u)\sum_{j=1}^{n-1}(P_{n,j}-K)\theta_{j,n}(v))\\
&= n\theta_{0,n-1}(u)\left(\sum_{j=0}^{n}K\theta_{j,n}(v)-\sum_{j=0}^{n}P_{0,j}\theta_{j,n}(v)\right)+\\
&\quad n\theta_{n-1,n-1}(u)\left(\sum_{j=0}^{n}P_{n,j}\theta_{j,n}(v)-\sum_{j=0}^{n}K\theta_{j,n}(v)\right)\\
&= n\theta_{0,n-1}(u)\left(K-B(v)\right)+n\theta_{n-1,n-1}(u)\left(T(v)-K\right), \qquad \text{(A.4)}
\end{aligned}
$$

approaching the limit for $n\to\infty$, $\theta_{0,n}(v)=\theta_{n,n}(v)\to 0$ for $v=(0,1)$. We note here that $\theta_{0,n-1}(u)$ and $\theta_{n-1,n-1}(u)$ also approach zero, albeit slower being of lower degree.

$\frac{\partial S(u,v)}{\partial v}$ can be computed in a similar way:

$$\frac{\partial S(u,v)}{\partial v}=n\theta_{0,n-1}(v)\left(K-L(u)\right)+n\theta_{n-1,n-1}(v)\left(R(u)-K\right).$$

Approaching the limit again, the (unnormalized) interior surface normal equals,

$$
\begin{aligned}
\overline{n}(u,v) &= \frac{\partial S(u,v)}{\partial u}\times\frac{\partial S(u,v)}{\partial v}\\
&= n^2\left[\theta_{0,n-1}(u)\left(K-B(v)\right)+\theta_{n-1,n-1}(u)\left(T(v)-K\right)\right]\times\\
&\quad \left[\theta_{0,n-1}(v)\left(K-L(u)\right)+\theta_{n-1,n-1}(v)\left(R(u)-K\right)\right]\\
&= n^2\Big(\theta_{0,n-1}(u)\theta_{0,n-1}(v)\left[(K-B(v))\times(K-L(u))\right]+\\
&\quad \theta_{0,n-1}(u)\theta_{n-1,n-1}(v)\left[(K-B(v))\times(R(u)-K)\right]+\\
&\quad \theta_{n-1,n-1}(u)\theta_{0,n-1}(v)\left[(T(v)-K)\times(K-L(u))\right]+\\
&\quad \theta_{n-1,n-1}(u)\theta_{n-1,n-1}(v)\left[(T(v)-K)\times(R(u)-K)\right]\Big). \qquad \text{(A.5)}
\end{aligned}
$$

A zero cross-product of non-vanishing vectors means the two vectors are co-linear. However, no such colinearity is possible because K is an interior Kernel location and $(u,v)\in(0,1)$. Further, an inspection, using the right-hand rule, of the four cross products in the last expression reveals that all four have the same Z sign. Then, the summed result is a non-zero normal.

Finally, and while the differentiation of B-spline input is a bit more complex, it will follow similar steps.

■