Construction of Smooth Branching Surfaces using T-splines

A Ginnis, K Kostas, Panagiotis Kaklis

Construction of Smooth Branching Surfaces using T-splines

A.I. Ginnis[a], K.V. Kostas[b], P.D. Kaklis[c,d,a]

[a]*School of Naval Architecture & Marine Engineering, National Technical University of Athens, Greece*
[b]*School of Engineering, Nazarbayev University, Kazakhstan*
[c]*Department of Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, UK*
[d]*EPICE AROMATH, Inria, Sophia Antipolis Méditerranée*

Abstract

The request for designing or reconstructing objects from planar cross sections arises in various applications, ranging from *CAD* to *GIS* and *Medical Imaging*. The present work focuses on the *"one-to-many"* branching problem, where one of the planes can be populated with many, possibly tortuous and densely packed, contours. The proposed method combines the proximity information offered by the *Euclidean Voronoi diagram* with the concept of *surrounding curve*, introduced in [1], and T-splines technology [2] for securing a flexible and portable representation. Our algorithm delivers a single T-spline that deviates from the given contours less than a user-specified tolerance, measured via the so-called discrete Fréchet distance [3] and is C^2 everywhere except from a finite set of point-neighborhoods. Subject to minor enrichment, the algorithm is also capable to handle the *"many-to-many"* configuration as well as the *global* reconstruction problem involving contours on several planes.

Keywords: Branching surface, T-splines, Voronoi diagram, surrounding curve, discrete Fréchet distance

1. Introduction

Designing or reconstructing objects from planar cross sections arises in various application areas, ranging from *Computer-Aided Design (CAD)* [4, 5] to *Geographic Information Systems (GIS)* [6, 7] and *Medical Imaging* [8, 9, 10]. Developing generic methods for handling the user needs in the aforementioned application areas poses a variety of challenges of topological, geometrical and modeling nature, such as complying with the genus of the underlying surface, securing the required smoothness and accuracy and, finally, relying on as portable as possible surface representations.

State-of-the art algorithms have so far adopted a *divide-and-conquer* strategy within the cells created by the arrangement of the given cross-sections and may be coarsely categorized into a pair of families. The first one adopts *surface extraction through volume reconstruction* via, e.g., volume rendering, in particular marching cubes, pruning the Delaunay triangulation between adjacent cross sections [11], implicit models [12, 13], set-valued functions [14] and multi-level partition-of-unity implicit models [15]. The second family includes *interpolation approaches* based on splines, in most cases low-degree (linear) piecewise interpolants (triangulations), involving global optimality criteria, such as bounding the maximum volume [16], minimizing the area of the surface [17, 18, 19], simplifications of the generalized Voronoi diagram [20], tiling with the aid of immersed medial axis [21], straight-skeletons of the symmetric difference of consecutive slices [22], and combining trimming of contour-surrounding curves with hole filling [1]. In the same category, [23] proposes a method for the iterative refinement of sets, corresponding to cross-sections, producing weighted averages of sets that can be used, through a subdivision scheme, for the representation of the 3D object.

*Corresponding author: A.I. Ginnis, ginnis@naval.ntua.gr

The cases of non-parallel cross sections [24] or non-manifold surface networks that partition the space into regions with multiple labels [25] are also attracting increasing interest in pertinent literature. Finally, one should point to recent contributions which can preserve the known genus of the original object by, e.g., appropriate sampling conditions [26] or enriching the classical divide-and-conquer strategy with global combinatorial optimization for exploring feasible topologies of tiles and computing a score that assesses the likelihood of each topology [8].

The present work develops a method that is able to deliver smooth solutions to the *"one-to-many"* local reconstruction problem for cases where one of the planes (*plane-1*; see Figure 1) is populated with many, possibly tortuous and densely packed, contours. The paper improves drastically the method presented in [1] by strengthening its topological robustness and expanding its representation portability.

The topological structure is built upon the proximity information contained in the planar *Euclidean Voronoi diagram (EVd)* of the contours set on plane-1. More accurately, we firstly improve the flexibility of the convex-hull based concept of the *surrounding curve*, introduced in [1], so that, on the basis of EVd and a user-defined viewing angle ω, it is automatically decided which contours of plane-1 will be connected to parts of the single contour in the other plane (*plane-0*). In the sequel, we generate an EVd-based quadrilateral tiling of a 2D circular domain with holes (see Figure 2), which is homeomorphic to the sought-for surface. This tiling determines the number and the neighborhood of the surface's saddle points; see Figure 3.

Regarding the representation portability, we employ T-splines technology [2], which offers a flexible representation that is NURBS-compatible but free from deficiencies stemming from their non-local tensor-product nature. T-splines technology is among the recently developed NURBS generalizations that allow local refinements, such as THB-splines [27], LR B-splines [28], Hierarchical B-splines and T-splines. In comparison with [1] that delivers a G^1-continuous multi-patch NURBS surface with trimmed patches, the present method guarantees that the output is a single T-spline that is C^2 everywhere except from a finite set of point-neighborhoods, where smoothness degrades to G^1 level. Furthermore, our numerical experience indicates that the fairness of the surfaces resulting from the proposed method is superior in comparison with those delivered from [1].

Based on the *"one-to-many"* reconstruction process, we can handle the more general *"many-to-many"* case, by introducing, similarly to the approach followed in [29], a contour on an intermediate plane and applying the *"one-to-many"* method twice. In our case, the introduced contour can be constructed as the intersection of the intermediate plane with the surface interpolating the *surrounding curves*, which correspond to the contour-sets on the two planes; see §2. Thus, for the case where no correspondence between contours can be established, this approach offers a possible solution taking also into consideration the shape information encoded in the surrounding curves. Furthermore, enriching each contour with cross-plane tangent distributions, one can handle the *global* reconstruction problem involving contours on several parallel planes.

The rest of the paper is structured in three sections. Section 2 starts with a coarse description of our method by presenting its key features, namely the quadrilateral tiling of a disc with holes homeomorphic with the sought for surface, the surrounding curve and the Voronoi vertices enclosed therein and, finally, the minimal control cage and its refinement. The section ends with §2.2, providing the algorithm for constructing the initial minimal control cage and the way it is refined for improving the quality of the surface. In §3, we test the performance of the algorithm for two one-to-many problems: a simple radially symmetric 1-3 configuration with circular contours (see Fig. 7) and a complex 1-18 configuration with contours possessing size and shape ranging over a wide spectrum; see Fig. 9. Finally, two additional examples, Figs. 12, 15, the second one involving real medical data, illustrate that the algorithm can be readily used for handling data lying on several planes. The paper ends by summarizing the basic advantages of the proposed approach and discussing future work (§4). Appendix A outlines the adopted approach for refining the control polygons of the given contours, so that the resulting curves, independently of their parameterization, are within a specified tolerance from the initial contours. This approximation step is required so that the generated T-spline surface approximates, at a user-specified tolerance, the input contours.

2. Method description

2.1. General

As mentioned in the introduction, the paper focuses on developing an automatic algorithm for solving the one-to-many branching problem formulated as below:

- **input**: a set of planar, simple and closed curves represented as B-Spline curves with individual parameterizations: one base contour $\mathbf{C}_{0,1}$ on one plane (plane-0), plus multiple contours $\mathbf{C}_{1,\mu},\ \mu = 1, ..., m$ on a parallel plane (plane-1), and a tolerance value ϵ limiting the allowed deviation from the input curves; see Fig. 1.

- **output**: construct a T-spline surface approximating, at tolerance ϵ, the input contours on the two planes.

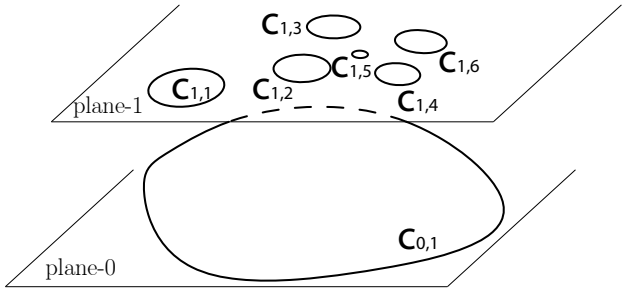


Figure 1: Input data for the one-to-many problem.

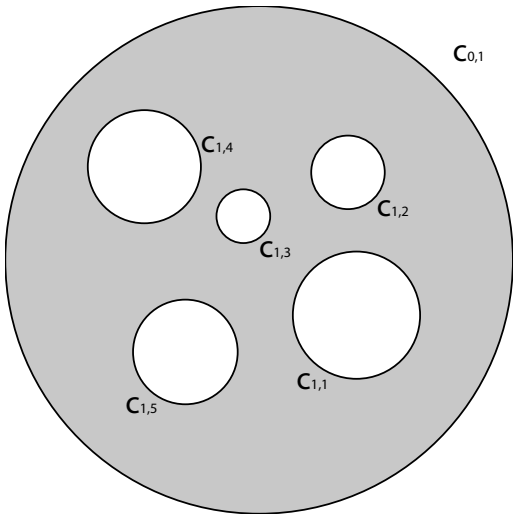


Figure 2: Hollowed disc corresponding to interpolating surface. Surface borders correspond to contours on plane-0 ($\mathbf{C}_{0,1}$) and plane-1 ($\mathbf{C}_{1,\mathrm{i}},\ \mathrm{i} = 1, \ldots, 5$).

The construction of the interpolating/approximating T-spline surface is based on the generation of an appropriate control network, the so-called *control cage*. For this control cage, we need to establish the minimum number of control vertices, required per contour, and their connectivity so that the topological structure of the surface can be established. The algorithmic procedure that computes these minimum numbers and the corresponding connectivity is described in §2.2.

The surface we are willing to construct, for the aforementioned one-to-many case, is topologically equivalent to a disc with holes. The external circular boundary of the disc corresponds to the single contour on plane-0, whereas the hole boundaries correspond to the set of contours on the remaining plane-1; see Figs 2 and 1. Aiming to construct a branching surface comprising a set of quadrilateral surface patches, we would like to generate a surface decomposition of the disc in Fig. 2 into quadrilateral components according to the following rules:

- If a patch corner touches one of the contours, then one of the patch-edges forming this corner should be a part of the touching contour. As a consequence, patch corners on contours are shared by exactly two neighboring patches.

- The number of corners that are common to three or more than four patches should be limited to the extend possible.

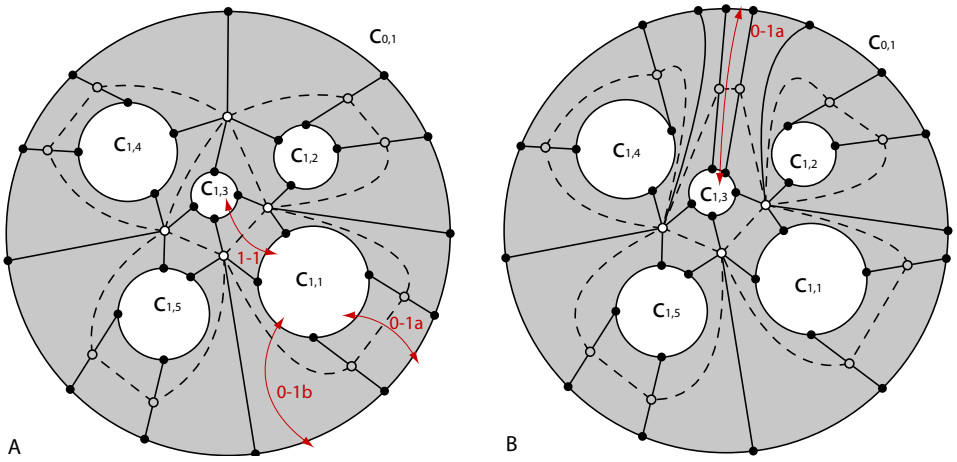


Figure 3: Disc decompositions.

Two disc decompositions that follow the previously mentioned rules are depicted in Fig. 3. In both sketches all bullets correspond to patch corners while curve segments between bullets, both solid and dashed, correspond to patch edges. Specifically, black bullets are corners that lie on contours, gray bullets are corners shared by exactly four patches while the white ones correspond to corners that do not lie on contours and are shared among three or more than four patches. Solid edges have one of their endpoints lying on a contour while dashed ones have no endpoint on a contour and are referred to as *bridge-edges*. We can also view this decomposition as a graph where bullets correspond to vertices and patch-edges to graph-edges. Using the bridge-edges we can establish an association between different contour parts. For this purpose, we classify bridge-edges in three types as follows: a) simple bridges connecting simple vertices, i.e., vertices of degree 4, b) complex edges connecting complex vertices, i.e. vertices of degree greater than 4 and c) intermediate edges where one vertex is simple and the other complex. Hence, two contour parts belonging to two different contours may be either disassociated, i.e., there exists not a single bridge-edge of any kind lying between them or they are associated via:

1. A complex bridge forming the so-called $1-1$ connection, i.e., associated contour parts belong to coplanar contours; see Fig. 3.
2. A simple bridge forming the so-called $0-1a$ connection, i.e., contour parts lie on different planes and are associated in a simple way: if we removed the bridge, a single quadrilateral patch could be formed with those two parts opposite to each other; see Fig. 3.
3. An intermediate edge forming the so-called $0-1b$ connection, i.e., contour parts lie on different planes but the their association is not simple as the removal of the bridge would not form a quadrilateral patch; see Fig. 3.

The difference between the two alternative decompositions A, B, depicted in Fig. 3, is that in decomposition A all associations of the contour parts of $\mathbf{C}_{1,3}$ belong to $1-1$ category, while in decomposition B, $\mathbf{C}_{1,3}$ parts have associations that are of all types. In other words, in the first decomposition $\mathbf{C}_{1,3}$ has no direct association to the contour on plane-0, which is not the case in the second decomposition; see Fig. 3B.

Our method primarily consists of an automatic process that generates the graph in Fig. 3 and uses it as the initial "control cage" for the interpolating surface. The main components in this process deal with the identification of the contour parts from plane-1 that are associated with parts of the single contour on plane-0 and the identification of the white bullets, i.e., the *extraordinary points* in the controlling network, which control the remaining inter-contour associations. Furthermore, a preprocessing step is likely to be required for guaranteeing that the generated surface will deviate from the input contours within a user-specified tolerance; see Appendix A.

Appealing to the concept of the *surrounding curve* we can handle the first issue, i.e., finding the associated parts between the two planes, while the set of vertices of the Voronoi diagram

of the contours in plane-1, clipped by the surrounding curve, play the role of the extraordinary points. Specifically, the surrounding curve is based on the convex hull of the contours in plane-1. In its simplest form, introduced in [1], it coincides with it and in this case the contour parts from plane-1, that connect to plane-0, are those that are on the convex hull. We have generalized this concept (see §8.2 ibid.) so that, given a user-specified angular tolerance, the surrounding curve is permitted to touch contours that are fully in the interior of the convex hull. The extraordinary points can be used for indicating the saddle points in the generated surface and the Voronoi vertices seem a natural selection for their position. As can be seen from Fig. 3, the exact number of such points is affected from the connections to plane-0 which are dictated by the construction of the surrounding curve as described in §2.2. Once we have established the initial control cage, i.e., identified the number of required control points and their connectivity, the next step involves the replacement of contour points with control points of the contours. Finally, the initial control-cage is enhanced by including all contour control-points and introducing additional edges based on the already established connection-correspondence defined by the bridge-edges. This enriched control cage is used for the final, in our implementation cubic, T-spline surface generation, which is C^2 everywhere with the exception of the neighborhood of extraordinary points, where G^1 continuity is achieved.

2.2. Construction of the Control Cage.

In this subsection we present the construction process of the T-spline surfaces control cage. We begin by describing the procedure for generating the minimal control cage, i.e., the cage that utilizes the minimum number of contour-points required per contour in both planes, and their connectivity so that its topological structure materializes the quadrilateral tiling needed for the generation of the branching surface, which is topologically equivalent to a disc with holes, as outlined in the first part of this section; see also Fig. 3 . It is worth noticing that in the below construction we assume that all constructed Voronoi vertices are of degree 3 and every triad of contours corresponds to at most one Voronoi Vertex. The generation of the minimal control cage is accomplished by following the steps described in sequel:

Step 0 - **discretization:** Create a densely discretized representation of the initial contours belonging on both planes. The discretization is controlled by the value of a discretization parameter.

Step 1 - **convex-hull and contour connectors on plane-1:** Compute the convex hull (*CH*) for the contour point-sets on plane-1 and identify the line segments, henceforth referred to as *CH_ContourConnectors*, that belong to the convex hull and connect consecutive contour point-sets; see dashed line segments in Fig. 4a.

Step 2 - **Voronoi triangles identification:** Compute the Delaunay triangulation of the point-sets on plane-1 and identify the Voronoi triangles, i.e., the triangles connecting points from 3 distinct contour point-sets; see solid thick triangles in Fig. 4b.

Step 3 - **modification of contour connectors:** Replace each *CH_ContourConnector* with the internal sides of the corresponding Voronoi triangle, if the triangle's internal angle ω satisfies $\omega \geq \alpha$, where α is an input angular parameter, and generate the set of *ContourConnectors*; see Figs. 4b-c. This procedure is carried out iteratively until all Voronoi triangles have an internal angle smaller than α; see Figs. 4b-d. The triangles that have been used in this procedure are removed from the construction process.

Step 4 - **connecting Voronoi vertices to contours:** For each remaining Voronoi triangle generate the corresponding Voronoi vertex. Moreover, for each pair of contours that is connected with two *ContourConnectors*, introduce an additional vertex on the corresponding Voronoi edge, which will act as an additional Voronoi vertex. Next, for each Voronoi vertex, generate a connecting line segment between the vertex and the corresponding point on each contour; see Fig. 4e. Finally, if two *ContourConnectors* have a common end-vertex, this vertex is replaced by two new ones on the corresponding contour using an appropriate separation parameter and the final set of *ContourConnectors* is produced; see dashed lines in Fig. 4f.

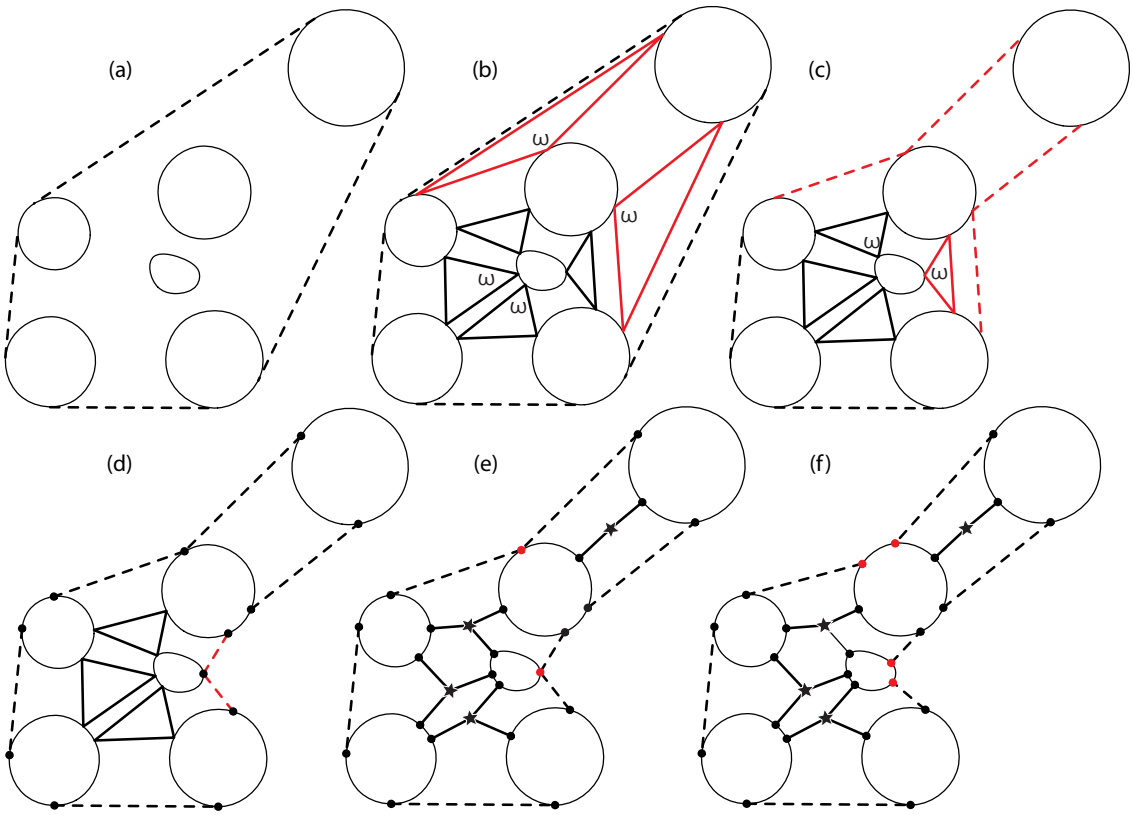


Figure 4: Minimal control cage construction (steps 0-4): working on plane-1.

Step 5 - **surrounding curve & correspondence to plane-0 contour:** Assemble the *SurroundingCurve (SC)* as the polygonal line that comprises the *ContourConnectors*, the segments between the separated vertices of Step 4 and the segments joining the endpoints of the *ContourConnectors* incident at the same contour point-set; see Fig. 5a. Using the convex-hull projection, see [30], generate a polar parameterization of the contour point-set on plane-0 and the SC on plane-1. Next, create line segments connecting the end-points of the *ContourConnectors* with the points of the contour of plane-0 that share the same polar parametrization; see Fig. 5b. Furthermore, identify the points on the contour point-set on the plane-0, that correspond to the mid-points of the *ContourConnectors* on plane-1; see dashed lines in Fig. 5b.

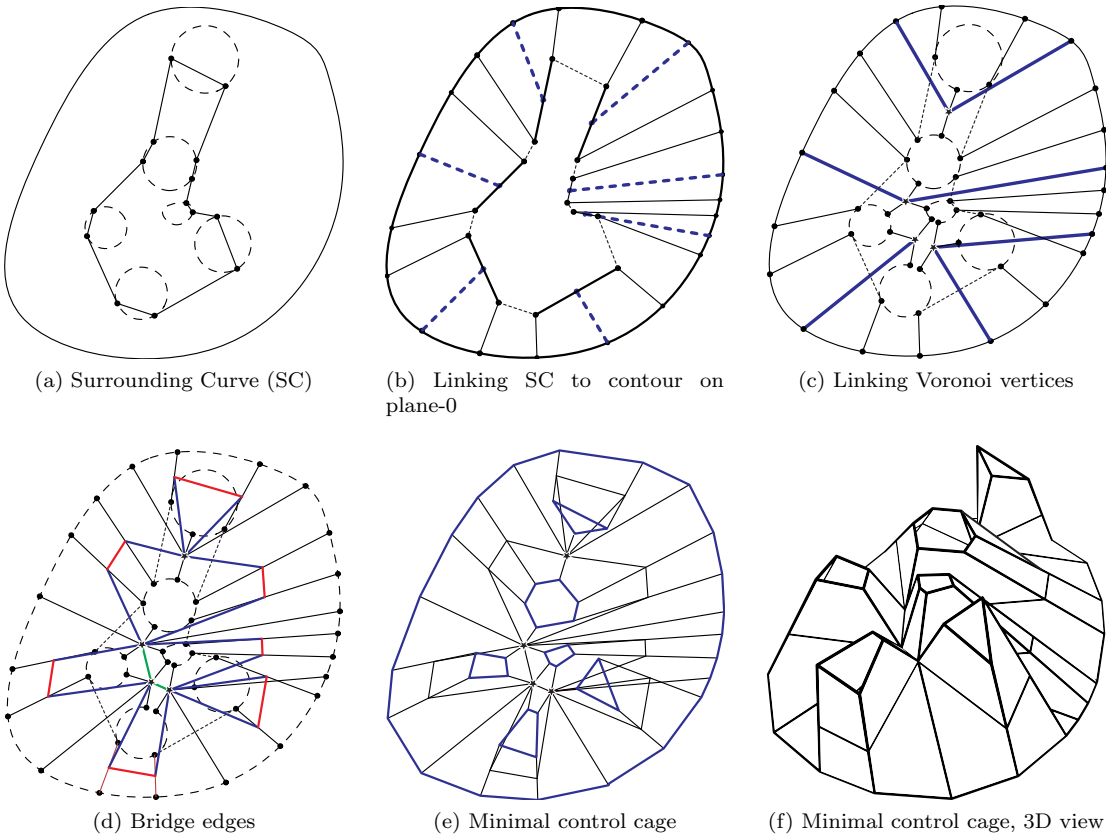
Step 6 - **linking Voronoi vertices to contours on plane-0 and plane-1:** Perform a vertical translation of each Voronoi vertex using the input parameters *v_immersion_i*, thus positioning each Voronoi vertex between the two levels. Next, for each *ContourConnector*, create a line segment connecting the corresponding Voronoi vertex, i.e., the one whose circle touches both connected contours, with the points on the contour of plane-0, identified in Step 5, which correspond to the *ContourConnector* mid-points; see Fig. 5c.

Step 7 - **bridge-edges:** Create bridge-edges as follows:

$1-1$: Generate line segments connecting neighboring Voronoi vertices, defined as those Voronoi vertices that are associated to the same two contours; see thick green lines in Fig. 5d.

$0-1b$: For each *ContourConnector*, create two line segments, connecting the corresponding Voronoi vertex with the segments connecting plane-0 with plane-1 and emanating from the end-points of the *ContourConnector*, at positions defined by the parameter *v_immersion_i*; see thick blue lines in Fig. 5d.

$0-1a$: Create the line segments connecting the end-points of the $0-1b$ edges that lie on lines emanating from the same contour of plane-1; see thick red lines in Fig. 5d.



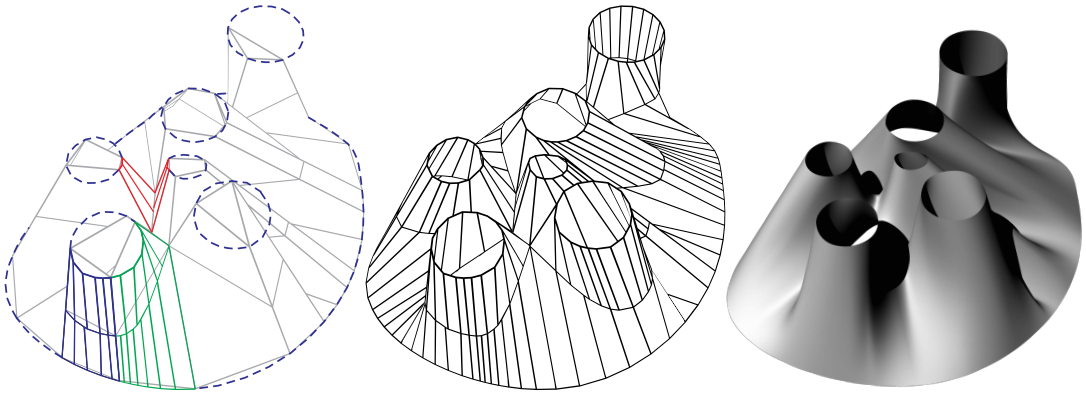
(a) Surrounding Curve (SC) (b) Linking SC to contour on plane-0 (c) Linking Voronoi vertices

(d) Bridge edges (e) Minimal control cage (f) Minimal control cage, 3D view

Figure 5: Minimal control cage construction (steps 5-8).

Step 8 - **minimal control cage:** Connect by linear segments the identified points on each contour (in both planes); see thick solid lines in Fig. 5e. These segments along with the Voronoi vertices and the complete connectivity of points and Voronoi vertices described in the previous steps constitute the minimal control cage; see Figs 5e,5f.

Having created the minimal control cage, we proceed with refining the cage in order to guarantee that the final T-spline surface deviates from the given contours within a tolerance ϵ, regardless of the parameterizations employed in its construction. To this end, we first employ the algorithm described in Appendix A for a given tolerance ϵ and produce control polygons for all given contours; see thick dashed lines in Fig. 6a. If the generated control points of a contour are fewer than implied by the minimal control cage we perform adequate knot-insertions. We then relocate the contour-points of the minimal control cage to their nearest control points of the individual contours, leading to a modified control cage and an explicit correspondence map between the control points of different contours via the bridge-edge concept; see Fig 6a. This correspondence map is used for refining the cage, since the parts requiring refinement can be easily identified. The refining process comprises the construction of the polygonal lines connecting the remaining control points, the modification of the bridge edges in order to encapsulate the shape of the connected polygonal lines and finally, the creation of linear T-edges between corresponding polygonal lines and bridge edges; see red, green and blue thick lines in Fig. 6a corresponding to $1-1$, $1-0a$ and $1-0b$ associations respectively. The final control cage, depicted in Fig. 6b, is used to construct the T-spline branching surface depicted in Fig. 6c.



(a) Refinement of the minimal control cage

(b) Final control cage

(c) Final T-spline surface

Figure 6

3. Examples

Our method, described in §2, has been implemented as a plug-in to Rhinoceros®[31] modeling environment with the aid of Autodesk®'s T-Splines® Plug-In [32] for Rhinoceros®. Our plug-in is coded in C# using the RhinoCommon SDK[1]. All examples presented in this work have been produced using the aforementioned implementation.

Our first example involves a simple data-set with one contour on plane-0 and three contours on plane-1. All contours are circles and the data-set is radially symmetric as can be seen from Figures 7 and 8. The generated minimal (initial) and final control cages are depicted in Fig. 7a and 7b, while Fig. 7d presents a renders view of the final T-Spline surface based on the final control-cage shown in Fig. 7c. Finally, in Fig. 8, the Gaussian curvature for the T-spline surfaces based on the minimal (initial) and final control cages are depicted. For both cases it is easy to observe that the radial symmetry of the data-set is preserved in our construction. In this example, we have a single extraordinary point in the construction and the overall number of vertices for the minimal control cage is 43 while this number rises to 140 when we require that the circular contours are approximated with a tolerance of 1%. The conversion[2] of the latter T-spline surface to a NURBS representation yields a multi-patch surface with 1416 control points.

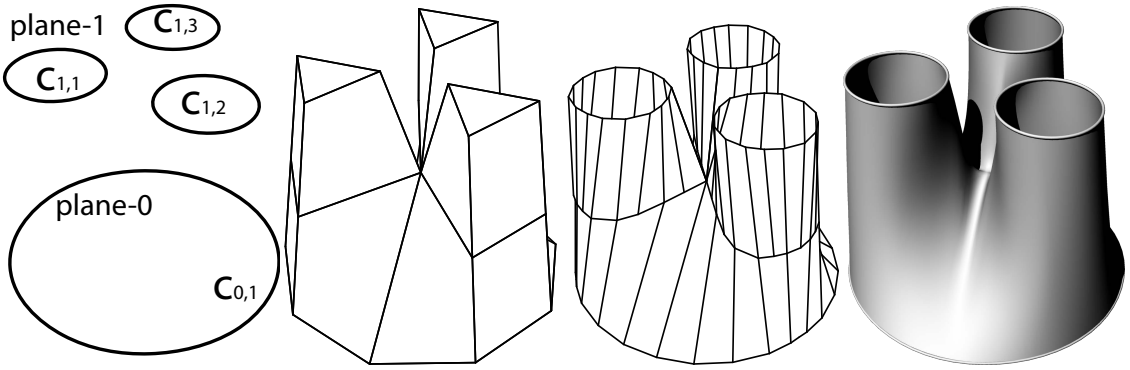
In our second example, we are demonstrating the output of our approach for the case of a more complex data-set with 18 contours on plane-1 of varying size and shape; see Fig. 9. As can be easily seen from this figure, 4 contours ($\mathbf{C}_{1,5}$, $\mathbf{C}_{1,8}$, $\mathbf{C}_{1,9}$ and $\mathbf{C}_{1,17}$) are non-convex with $\mathbf{C}_{1,8}$ exhibiting a very abrupt shape change.

Figure 10 depicts the generated T-spline surface using the initial / minimal control cage. Its enrichment with the total number of control points used in the representation of contours leads to the final control cage which generates the T-spline surface depicted in Fig. 11. This final surface guarantees that distance of the boundary edges of the T-spline surface from the corresponding input contours are below a specified tolerance, which, in this case, is below 0.05% of contour's length. The number of control points in this final representation is 741 and the conversion to a NURBS yields 126 surface patches with an overall number of control points rising to 28942. Finally, the number of extraordinary points for this example is 24.

In our final examples, we will demonstrate the usage of the developed approach for the generation of T-spline surfaces interpolating / approximating contours in multiple planes. The first of these examples involves 3 planes as can be seen in Fig. 12. The overall number of contours

[1]RhinoCommon is a cross-platform .NET plug-in Software Development Kit.

[2]The conversion is performed using the corresponding functionality of Autodesk®'s T-Splines® Plug-In.



(a) Data-set (b) Minimal Cage (c) Final Cage (d) Rendered surface

Figure 7: Control cages of a radial-symmetric data-set with 4 input contours: 1 on plane-0 and 3 on plane-1.

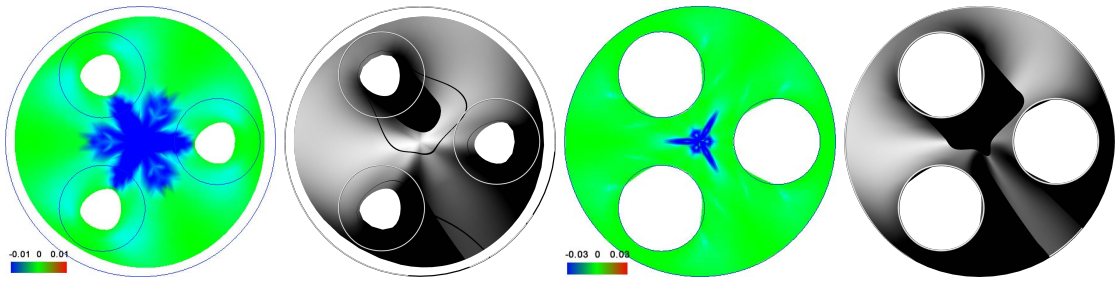


Figure 8: Gaussian curvature graph for the initial (left) and final (right) surface using cages shown in 7b and 7c, respectively.

is 8 with 1 contour lying at each boundary plane (plane-1 and plane-1*) and the remaining 6 contours at the intermediate plane-0. We apply our construction process twice, generating one T-spline surface, T_1, between the contours on planes 1 and 0 and one surface, T_{1*}, between planes 0 and 1*. The control cages for those two surfaces are depicted in Fig. 14a. Employing the same approximating tolerance, the boundary edges of those T-spline surfaces on plane-0 will coincide. As a second step, in our construction for this particular example, we modify the second row of control points in both surfaces so that, for each boundary control point $\mathbf{b}_i$ on plane-0, the vectors $\mathbf{b}_{i,1} - \mathbf{b}_i$ and $\mathbf{b}_{i,1*} - \mathbf{b}_i$ are collinear, where $\mathbf{b}_{i,1}$, $\mathbf{b}_{i,1*}$ denote the previous and next to $\mathbf{b}_i$ control points that belong to T_1 and T_{1*}, respectively; see Fig. 14b. Employing this modification for all the boundary control points on plane-0, first-order geometric continuity is secured between the two surfaces, as can been seen in Fig. 13. The extraordinary points in this final construction are 8 overall; 4 in each T-spline surface.

As our last example, we use the 44 contours shown in Fig. 15a coming from a mesh model of the eighth tooth retrieved from the AIM@SHAPE shape repository [33]. The original model is a surface mesh in VRML format, which is depicted in Fig. 15d. The contours were acquired by initially intersecting the initial mesh with parallel planes and subsequently approximating them with cubic B-Splines. The one-to-many cases exhibited in this example are three, denoted with think solid lines in Fig. 15a. Their resulting T-spline control cages are shown in Fig. 15b, while the reconstructed surface for the whole tooth is depicted in Fig. 15c. Obviously, the surface approximation quality is dependent on the selection and approximation of the contours set, however, with this rather rough set we obtained a maximum deviation, between the initial and reconstructed surface, of less than 1% of tooth's width. The required number of control points for all three branching areas is around 1000, while more than 8000 would be required for acquiring the same approximation level using a multi-patch NURBS surface approach.

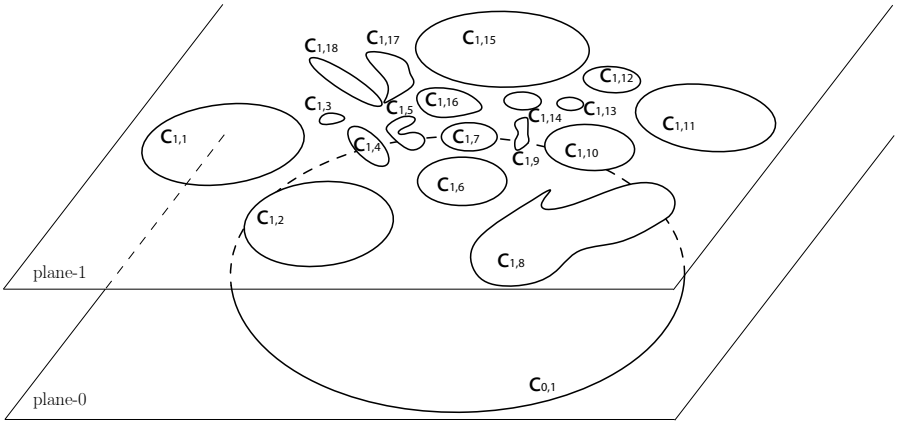


Figure 9: Data-set with 19 input contours: 1 on plane-0 and 18 on plane-1.

Figure 10: Rendered view of the initial T-spline surface along with the input contour-set.

Figure 11: Rendered view of the final T-spline surface along with the input contour-set.

4. Conclusions & Future work

In this work, we have presented an automatic construction process for the generation of a smooth T-spline surface that interpolates / approximates simple planar curves (contours) that lie on parallel planes being C^2 everywhere except from the neighborhood of a finite set of points, where its smoothness degrades to G^1 level. Specifically, we have demonstrated the automatic generation of the T-spline control cage (T-mesh) that defines the T-spline surface for the special case of the "one-to-many" problem between two planes, where multiple contours lie on one plane and only one on the other. In our construction we have assumed that Voronoi vertices are of degree 3 and that for every triplet of contours there's a maximum of one Voronoi vertex. These assumptions are used for simplifying the construction of complex and intermediate bridge-edges. However, if we incorporate the actual Voronoi edges in our construction, the requirement for these limiting assumptions can be eliminated.

The proposed construction can be also utilized for the case of the more general "many-to-many" problem, where multiple contours lie on both planes. This can be accomplished by the introduction of a single contour on an intermediate plane. The introduced contour is not an arbitrary contour but is generated by intersecting the surface interpolating the surrounding curves from the two planes with the intermediate plane. Hence, the introduced contour captures the shape information from the two planes and allows us to split the "many-to-many" problem in two "one-to-many" problems. Obviously, this solution is limited by the assumption that all contours converge to a single one lying on the introduced plane, which however is a reasonable assumption when no contours' correspondence can be established. A more elaborate procedure must be followed if we want to avoid this convergence to a single contour which could be a topic for future research.

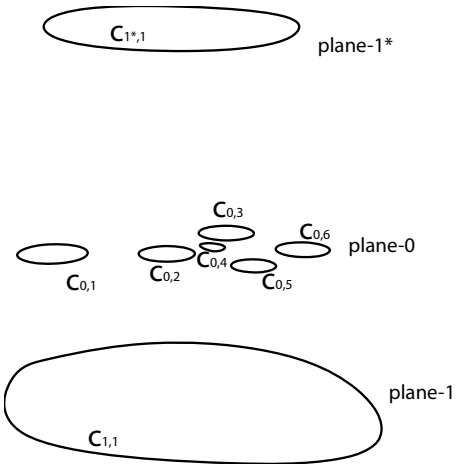


Figure 12: Data-set with contours on 3 parallel planes: plane-1, plane-0 and plane-1*

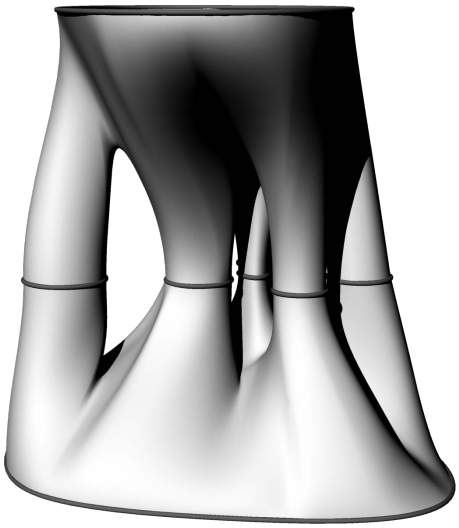
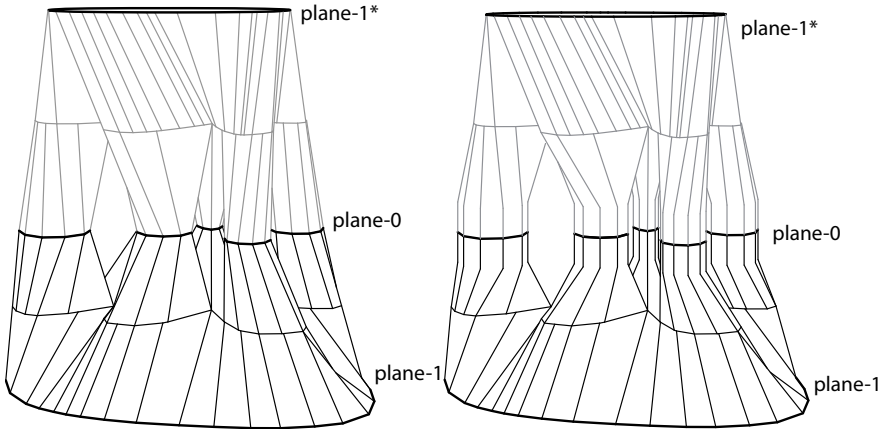


Figure 13: Rendered view of the final T-spline surface. Contours are depicted in dark-gray color.

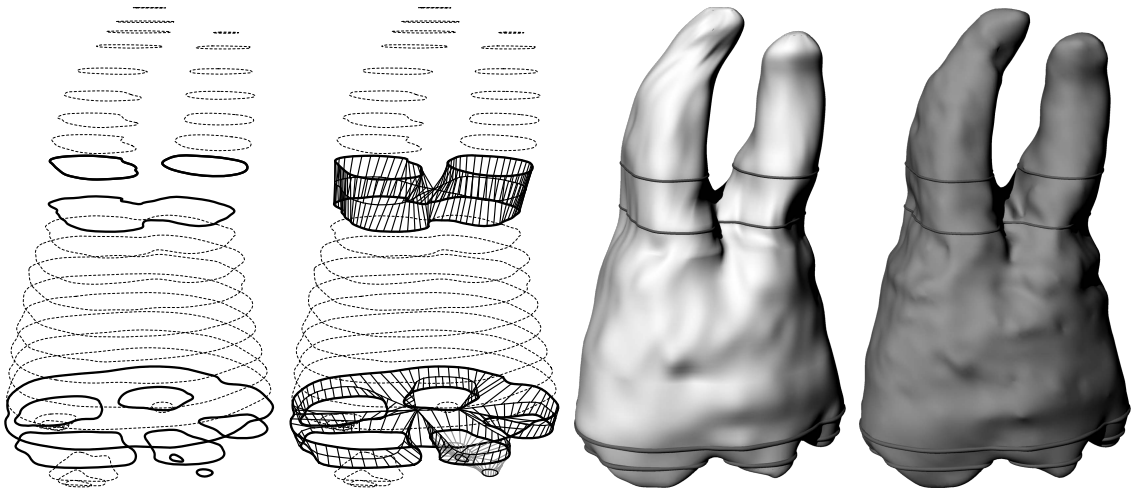


(a) Initial control cage. (b) Modified control cage.

Figure 14: The control cage of the T-spline surface interpolating contours on 0 and 1 planes is black, while the one corresponding to planes 0 and 1* is gray. Tangency handles are modified locally about plane-0 so that we can guarantee tangential continuity.

Furthermore, we have demonstrated the use our construction algorithm for the case of 3 parallel planes (see example 3 in §3) where the introduction of cross-plane tangent distributions on along the contours of the intermediate planes permits the extension of our approach to more than 2 planes. Finally, our approach is applied for reconstructing a tooth surface based on a rather small number of 44 contours. The resulting reconstructed surface is considerably close to the initial tooth-mesh as already discussed in §3.

Currently, due to certain simplifying assumptions, the implementation of our algorithm is restricted to contours that lie on parallel planes. This limitation could be overcome by the introduction of a different way for separating contour sets and a more general way for performing the Voronoi vertices' immersion in step 6 of our construction process. Furthermore, in our current implementation we have not taken any precautions for guaranteeing that the generated T-spline



(a) Tooth contours. Thick solid contours belong to planes where branching occurs.

(b) Control cages for the 3 branching cases

(c) Final reconstructed tooth surface

(d) Initial tooth mesh

Figure 15: Model of tooth No.8, retrieved from the AIM@SHAPE shape repository

surface is analysis suitable, however, if required, we may easily accomplish this by appropriate refinement in the neighborhood of extraordinary points.

Some of the simplifying assumptions in our current implementation are due to the employed software package, i.e., Autodesk®'s T-Splines® Plug-In, and the restrictions set by its use. Hence, it would be beneficial for the enhancement of the method and its implementation, if we could access the API of the specific software package or alternatively use a T-spline software library that allows access to internals of the parameterization.

Finally, it is among our aims to pursue the further development of this construction method for trivariate T-splines that would enable volumetric meshing and open a new window for possible applications. A possible way towards this aim may come out of the combination of the T-spline surface representation and the medial axis transform.

Appendix A. Controlling the deviation of the T-spline surface from the contour curves

Let us assume that all of our input contours are represented by closed cubic B-Spline curves and the generated surface is a bicubic T-spline surface with control polygons of boundary edges that coincide with those of the input contours. In this case, if we could impose an appropriate parameterization for each boundary T-spline surface edge, the generated surface would interpolate the input contours. Unfortunately, imposing the appropriate parameterization for all edges would explode the complexity of both the control-cage generating process and the resulting T-spline surface. In this work, we employ a simple uniform parameterization defined by the T-spline construction process; see §2.2. Hence, although the control polygons of the contours coincide with the control polygons of the T-spline boundary edges, the parameterizations of the boundary edges are generally not the ones used in the input contours and as a result, the T-spline surface does not interpolate them.

Two B-spline curves with different parameterizations, which however share the same degree and control polygon are confined to lie within the same area defined by the union of the local convex-hulls generated by corresponding control points; see Fig. A.16. For a simple planar closed curve represented as a B-Spline, we assume that the union of the local convex-hulls, U, occupies a planar region which is defined by two polygonal curves: the interior boundary, B_i, and the exterior boundary B_e. Although the boundary ∂U of U is in general composed of one or more polygonal

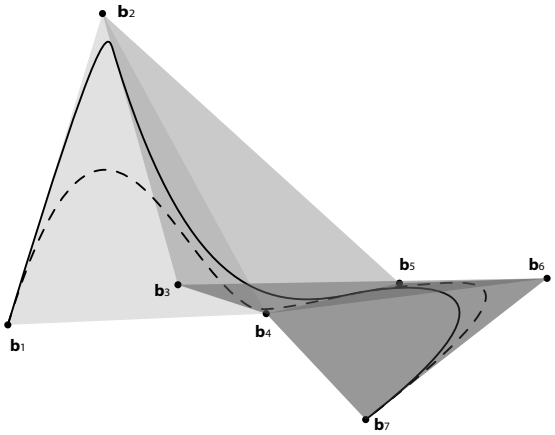


Figure A.16: Two cubic B-Spline curves lying in the union of their local convex hulls: $(\boldsymbol{b}_1\boldsymbol{b}_4\boldsymbol{b}_2) \cup (\boldsymbol{b}_4\boldsymbol{b}_5\boldsymbol{b}_2\boldsymbol{b}_3) \cup (\boldsymbol{b}_4\boldsymbol{b}_6\boldsymbol{b}_3) \cup (\boldsymbol{b}_4\boldsymbol{b}_7\boldsymbol{b}_6\boldsymbol{b}_5)$. Dashed curve's knot vector is $(\mathbf{0}, 0.872, 0.962, 0.983, \mathbf{1})$ while the solid one uses: $(\mathbf{0}, 0.046, 0.656, 0.702, \mathbf{1})$.

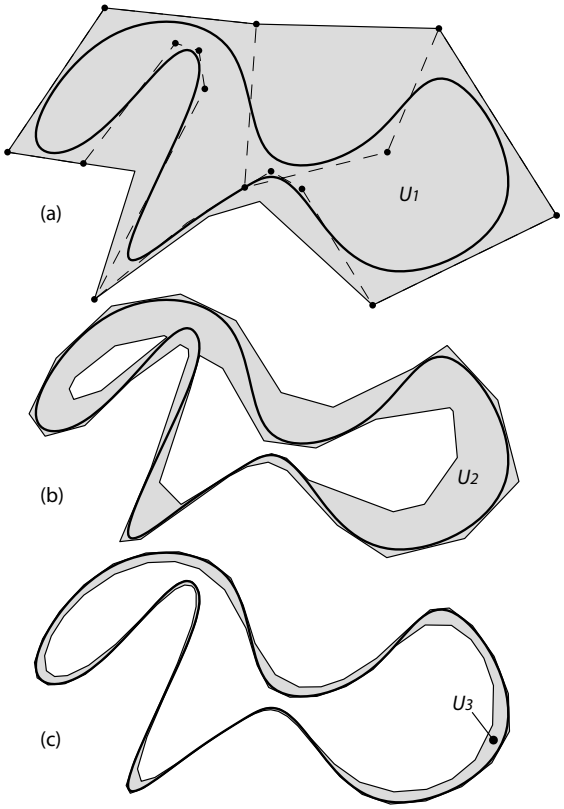


Figure A.17: (a) A cubic B-Spline along with its initial control polygon and local-convex-hull union U_1, colored gray. (b) The same B-Spline curve after uniform knot insertion along with the corresponding local-convex-hull union U_2. (c) After an additional uniform refinement, ∂U_3 is defined by two polygonal boundary curves.

curves, as shown in Fig. A.17, performing a finite number of knot-insertions to the initial curve will always lead to an area U whose boundary ∂U is defined by exactly two separate, polygonal boundaries; see Fig. A.17(c).

Based on this observation we can devise a simple process which can generate a control polygon that when shared by two curves, with arbitrary parameterizations, will confine them to be within a given ϵ distance from each other. Let's assume that two cubic B-Spline curves, $c_1(t_1)$, $c_2(t_2)$, $t_1 \in [a, b]$ and $t_2 \in [c, d]$ with different knot vectors use the same control polygon. The distance between them, measured by the Fréchet metric, is defined to be:

$$\delta_F(c_1(t_1), c_2(t_2)) = \inf_{\alpha,\beta \in C[0,1]} \max_{t \in [0,1]} d\left(c_1\left(\alpha(t)\right), c_2\left(\beta(t)\right)\right),$$

where d is a 2D point metric (e.g., Euclidean distance) and $t_1 = \alpha(t)$, $t_2 = \beta(t)$ are continuous, non-decreasing functions mapping $[0, 1]$ to $[a, b]$ and $[c, d]$, respectively. Let us also assume that the union U of their local convex-hulls is defined by the polygonal curves B_i and B_e. As both curves lie in U, it is obvious that $\delta_F(c_1, c_2) \leq \delta_F(B_i, B_e)$. Furthermore, for every possible re-parameterizations t_i and t_j of c_1 and c_2 the above inequality still holds. Thus, if $\delta_F(B_i, B_e) \leq \epsilon$ the distance of any pair of cubic B-Spline curves with the same control polygon but arbitrary parameterizations will be less than ϵ, i.e., $\delta_F\left(c_1(t_i), c_2(t_j)\right) \leq \epsilon$.

As the computation of the exact Fréchet distance is a fairly complicated procedure, we use the discrete Fréchet distance (δ_{dF}) and the algorithmic procedure for its computation suggested by Eiter and Mannila [3]. Since, for all polygonal curves P and Q: $\delta_F(P, Q) \leq \delta_{dF}(P, Q)$, we may

write that:

$$\delta_F\left(c_1(t_i), c_2(t_j)\right) \leq \delta_F(B_i, B_e) \leq \delta_{dF}(B_i, B_e).$$

Obviously, if $\delta_{dF}(B_i, B_e) \leq \epsilon \Rightarrow \delta_F\left(c_1(t_i), c_2(t_j)\right) \leq \epsilon$. Hence, the procedure for assuring that the input contour c_j and the corresponding T-spline boundary-edge will lie within an ϵ distance has as follows:

1. Set $k = 0$
2. Compute the union of the local convex hulls (U_k) for the control polygon of c_{jk}.
3. If U is not defined by exactly two separate polygonal lines, perform uniform knot insertion to the contour until U has the required property.
4. Assuming U is defined by polygonal curves B_{ik} and B_{ek}, compute $\delta_{dF}(B_{ik}, B_{ek})$.
5. While $\delta_{dF}(B_{ik}, B_{ek}) \geq \epsilon$
   - identify the area of c_{jk} where $\delta_{dF}(B_{ik}, B_{ke})$ attains its value. Perform knot insertion in the identified segment of c_{jk} and get the new $c_{i,k+1}$ contour
   - Set k=k+1
   - Compute the new U_k and the corresponding B_{ik}, B_{ek}
6. Return the final c_{jk}.

References

[1] N. Gabrielides, A.-I. Ginnis, P.-D. Kaklis, M.-I. Karavelas, G^1-smooth branching surface construction from cross sections., CAD 39 (2007) 639–651.

[2] T. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and TNURCCs, ACM Transactions on Graphics 22 (2003) 477–484.

[3] T. Eiter, H. Mannila, Computing discrete Fréchet distance, Tech. Rep. CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria (1994).

[4] Z. Pei, Q. Wen-Han, Constructive G^1 connection of multiple freeform pipes in arbitrary poses, CAGD 30 (2013) 462–475.

[5] A. Bentamy, F. Guibault, J.-Y. Trepanier, Cross-sectional design with curvature constraints, Computer Aided Design 2005 3 (7) (2005) 1499–508.

[6] K. Fujimura, E. Kuo, Shape reconstruction from contours using isotopic deformation, Graph Model Image Process 61 (1999) 127–47.

[7] N. Sirakov, F. Muge, A system for reconstructing and visualising three-dimensional objects, Computers & Geosciences 27 (2001) 59–69.

[8] M. Zou, M. Holloway, N. Carr, T. Ju, Topology-constrained surface reconstruction from cross-sections, ACM Trans. Graph. 34 (4) (2015) 128:1–128:10.

[9] S. Schmitt, J. Evers, G. Duch, M. Scholz, K. Obermeyer, New methods for the computer-assisted 3D reconstruction of neurons form confocal image stacks, NeuroImage 23 (2004) 1283–98.

[10] E. Moschino, Y. Maurin, P. Andrey, Joint registration and averaging of multiple 3D anatomical surface models, Computer Vision and Image Understanding 101 (2006) 16–30.

[11] J.-D. Boissonnat, Shape reconstruction from planar cross sections, Comput. Vision Graph. Image Process. 44 (1988) 1–29.

[12] V. V. Savchenko, A. A. Pasko, O. G. Okunev, T. L. Kunii, Function representation of solids reconstructed from scattered surface points and contours, Computer Graphics Forum 14 (4) (1995) 181–188. doi:10.1111/1467-8659.1440181.

[13] G. Turk, J. O'Brien, Shape transformation using variational implicit forms, in: ACM SIGGRAPH, 1999, pp. 335–342.

[14] D. Cohen-Or, D. Levin, Guided multi-dimensional reconstruction from cross-sections, Ser. Approx. Decompos. 8 (1996) 47–56.

[15] I. Braude, J. Marker, K. Museth, J. Nissanov, D. Breen, Contour-based surface reconstruction using MPU implicit models, Graphical Models 69 (2) (2007) 139 – 157. doi:http://dx.doi.org/10.1016/j.gmod.2006.09.007.

[16] E. Keppel, Approximating complex surfaces by triangulation of contour lines, IBM J Res Devel 19 (1975) 2–11.

[17] H. Fuchs, Z. Kedem, U. SP, Optimal surface reconstruction from planar contours, Commun ACMl 20 (1977) 693–702.

[18] D. Meyers, S. Skinner, K. Sloan, Surfaces from contours, ACM Trans Graph 11 (1992) 228–258.

[19] K. Sloan, J. Painter, Pessimal guesses may be optimal: A counterintuitive search result, IEEE Trans. Pattern Anal. Mach. Intell. 10 (1988) 949–955.

[20] J.-M. Oliva, M. Perrin, S. Coquillart, 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram, Computer Graphics Forum 15 (1988) 397–408.

[21] C. Bajaj, E. Coyle, L. K-N, Arbitrary topology shape reconstruction from planar cross sections., Graph. Models Image Process. 58 (1996) 524–543.

[22] G. Barequet, A. Vaxman, Nonlinear interpolation between slices, in: Proceedings of the 2007 ACM symposium on Solid and physical modeling, 2007, pp. 97–107.

[23] S. Kels, N. Dyn, Reconstruction of 3d objects from 2d cross-sections with the 4-point subdivision scheme adapted to sets, Computers & Graphics 35 (3) (2011) 741 – 746, shape Modeling International (SMI) Conference 2011. doi:http://dx.doi.org/10.1016/j.cag.2011.03.007.
URL `http://www.sciencedirect.com/science/article/pii/S009784931100046X`

[24] J.-D. Boissonnat, P. Memari, Shape reconstruction from unorganized cross-sections, in: Proceedings of the 5th Eurographics Symposium on Geometry Processing (SGP'07), 2007, pp. 89–98.

[25] G. Barequet, A. Vaxman, Reconstruction of multi-label domains from partial planar cross-sections., Comput. Graph. Forum 28 (2009) 1327–1337.

[26] O. Amini, J.-D. Boissonnat, P. Memari, Geometric Tomography with Topological Guarantees., Discrete Comput. Geom. 50 (2013) 821–856.

[27] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: The truncated basis for hierarchical splines., CAGD 29 (2012) 485–498.

[28] T. Dokken, T. Lyche, K. Pettersen, Polynomial splines over locally refined box-partitions., CAGD 30 (2013) 331–356.

[29] H. Park, K. Kim, 3-D shape reconstruction from 2-D cross-sections, Journal of Design and Manufacturing 5 (1995) 171 – 185.

[30] A.-B. Ekoule, F.-C. Peyrin, C.-L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours., ACM Transactions on Graphics 10 (1991) 182–199.

[31] Robert McNeel & Associates, Rhinoceros, design, model, present, analyze, realize... (2014).
URL http://www.rhino3d.com/

[32] Autodesk Inc, Autodesk T-Splines Plug-in for Rhino (2014).
URL http://www.tsplines.com/

[33] AIM@SHAPE, Aim@Shape Repository, http://visionair.ge.imati.cnr.it/, [Online; accessed May-2016] (2016).