G^1-smooth Branching Surface Construction from Cross Sections

N. C. Gabrielides [a] A. I. Ginnis [a] P. D. Kaklis [a,*] M. I. Karavelas [b,c]

[a]*National Technical University of Athens, School of Naval Architecture and Marine Engineering, 9 Heroon Polytechneiou, GR-157 73 Zografou, Greece*

[b]*University of Crete, Department of Applied Mathematics, GR-714 09 Heraklion, Crete, Greece*

[c]*Foundation for Research and Technology - Hellas, Institute of Applied and Computational Mathematics, P.O. Box 1385, GR-711 10 Heraklion, Crete, Greece*

Abstract

This paper proposes a framework for constructing G^1 surfaces that interpolate data points on parallel cross sections, consisting of simple disjoined and non-nested contours, the number of which may vary from plane to plane. Using appropriately estimated cross tangent vectors at the given points, we split the problem into a sequence of local Hermite problems, each of which can be one of the following three types: *"one-to-one"*, *"one-to-many"* or *"many-to-many"*.

The solution of the *"one-to-many"* branching problem, where one contour on the i-plane is to be connected to $\mathcal{M}$ contours on the $(i+1)$-plane, is based on combining skinning with trimming and hole filling. More specifically, we first construct a C^1 *surrounding curve* of all $\mathcal{M}$ contours on the $(i+1)$-plane. Next, we build the so-called *surrounding surface* that skins the i-plane contour with the $(i+1)$-plane surrounding curve and trim suitably along parts of the surrounding curve that connect contours. The resulting multi-sided hole is covered with quadrilateral Gordon-Coons patches that possess G^1 continuity. For this purpose, we develop a hole-filling technique that employs shape-preserving *guide curves* and is able to preserve data symmetries. The *"many-to-many"* problem is handled by combining the *"one-to-many"* methodology with a zone-separation technique, that achieves splitting the *"many-to-many"* problem into two *"one-to-many"* problems. The methodology, implemented as a C++ Rhino v3.0 plug-in, is illustrated via two synthetic data sets and in the context of two realistic design examples. Finally, the paper concludes with discussing ongoing work towards improving the robustness and the applicability of the method regarding the surrounding curve construction step.

Key words: design, reconstruction, cross sections, branching surfaces, G^1 surfaces, skinning, trimming, hole filling, shape-preserving interpolation

1. Introduction

The request for designing or reconstructing objects from planar cross sections arises in various applications, ranging from CAD [6], to GIS [12, 37], and medical imaging [34, 30].

In the *reconstruction* case the density of input data is likely to be very high, in terms of the densities of captured cross sections and points per contour, thus orienting research toward developing fast algorithms for constructing C^0 planar triangular interpolants of the cross-sectional data. On the contrary, when we *design* a surface from parallel cross sections, the available data information is limited - only a small number of contours is usually available - which imposes the need for data interpolants that possess adequate smoothness (at least G^1) and whose analytic type enables their easy and robust transfer to the surface modeler used by the designer.

The method presented herein fits in the *design* context, though it is also readily applicable for *smooth reconstruction* purposes. Furthermore, we focus on handling the general case of planar cross sections that consist of disjoint smooth Jordan curves, henceforth referred to as contours, whose number can vary from plane to plane. We are thus able to handle, besides the standard *"one-to-one"* case, the *"one-to-many"* and *"many-to-many"* configurations, guaranteeing a G^1 interpolation surface, composed from polynomial patches of principally quadrilateral topology, that branches suitably at/through an intermediate, designer-specified, level/zone, respectively.

There is a plethora of papers dealing with the *"one-to-one"* problem in the design context, employing mainly B-Spline surfaces and NURBS [32, 19, 21, 18, 33]. Part of this literature is concerned with controlling the shape of the outcome surface by, e.g., minimizing its twist [14], or preserving its sectional curvature between shape similar contours [23, 24].

* Corresponding author.
Email addresses: ngabriel@deslab.ntua.gr (N. C. Gabrielides), ginnis@naval.ntua.gr (A. I. Ginnis), kaklis@deslab.ntua.gr (P. D. Kaklis), mkaravel@tem.uoc.gr (M. I. Karavelas).

On the other hand, lots of references deal with developing algorithms for solving the *local "one-to-one"* or *tilling,* according to [29], problem for reconstruction purposes. Such algorithms yield usually C^0 triangular surfaces with the aid of global optimality criteria, such as bounding the maximum volume [27], or minimizing the area of the surface [11, 38, 29]. In these works the sought for triangulation is thought as a path in a graph. Computing this path can also be accomplished locally, node-by-node, by imposing local criteria on the path nodes, such as minimizing the edge length of each triangle that is being generated after adding a new node to the path [7, 4, 17, 35]. Note that, in contrast to their complexity, these local approaches introduce a non-local step, namely the definition of the first node. There also exist hybrid schemes, which use local weights for satisfying global criteria [9, 39]. Finally, the graph model of [27], that works for polygonal contours, is extended in [36] for continuous parametric contour representations, enabling its application to quadrilateral surfaces; see also [8, 14].

The literature devoted to the *"one-to-many"* problem can be classified into four main families. The family of *contour-connection* methods attempt to artificially render an *"one-to-many"* problem into an *"one-to-one"* by connecting the disjoint contours with line [7] or triangular facet bridges [29]. The first choice succeeds only for very simple configurations, while the second constrains unnaturally the saddle points of the branching surface to lie on the plane containing the disjoint contours.

The so-called *intermediate-contour* methods are based on introducing an intermediate contour, thus splitting the original problem into two problems, an *"one-to-one"* and a new *"one-to-many"*. The second problem is further simplified into $\mathcal{M}$ *"one-to-one"* problems by suitably partitioning the intermediate contour into $\mathcal{M}$ parts. This idea was proposed in [36, 9], while [20] is apparently the first work where this idea is transformed into an algorithm.

The family of *partial contour connection and hole filling* methods, outlined in [3, 1, 2], is characterized by matching partially the disjoint contours with the single contour of the neighboring plane, thus leaving a number of holes to be filled in the final step of such a scheme. In [15], where the case *"one-to-two"* is being treated, the single hole is filled by an appropriately chosen hyperboloid.

Finally, the family of *implicit* schemes relies on the assumption that we possess implicit representations of the contours composing the cross sections. Then an implicit interpolant can be obtained by taking a convex combination of the contour representations [5], or employing the notion of distance function [22, 10].

The herein presented method for handling the *"one-to-many"* problem belongs to the class of partial contour connections and hole filling schemes. Its apparent novelties pertain to:

(i) Ensuring G^1 parametric continuity,
(ii) Providing as final outcome a spline polynomial surface, composed from patches of mainly quadrilateral topology, thus enabling portability to contemporary CAD systems,
(iii) Intensive use of shape-preserving interpolation techniques for all 2D/3D curves encountered during the surface construction scheme, in order to control the shape quality of the final outcome,
(iv) Introducing the concepts of *surrounding curve* and *surrounding surface* (§4.1),
(v) Developing a hole-filling technique that, though based on [16], is able to preserve data symmetries (§4.3), and
(vi) Constructing the guide curves as shape-preserving polynomial splines, that can intrinsically incorporate the geometry of the hole boundary (§4.3).

Last but not least, and to the best of the authors knowledge, pertinent literature is lacking in works handling the *"many-to-many"* problem in a G^1 setting, which is achieved by the technique presented in §5 of this paper.

The rest of the paper is structured as follows. Starting with a formulation of the (global) problem (see Problem 2.1), §2 ends with its decomposition to a sequence of local Hermite problems; see Problem 2.2. In §3 we briefly present the technique we adopt for constructing the contour curves and the tangent ribbons along them. §4 deals with the *"one-to-many"* configuration; this section is the kernel of the paper and consists of three subsections. In §4.1 we introduce the notions and describe the construction of *bridges* and *surrounding curve/surface.* §4.2 outlines the trimming process on the surrounding surface, that leaves us with a hole-filling request, which is handled in some detail in §4.3. Section 5 outlines the method for reducing the *"many-to-many"* case to two *"one-to-many"* problems. The tangent-vector estimates, that are necessary for decomposing the global problem into a sequence of local ones, are obtained as described in §6.

The proposed methodology is illustrated in §7 against two synthetic data sets and two realistic examples. The synthetic examples include a symmetric *"one-to-three"* branching configuration with circular contours (see Figs. 18–23) and a highly non-symmetric *"one-to-two"* configuration with non-convex contours; see Figs. 24–27. As for the realistic examples, the first is related with designing a detergent container with handle (see Figs. 28–29), while the second one stems from the area of ship design; see Figs. 30–32. The paper ends with summarizing the basic advantages of the proposed approach (§8.1) and discussing ongoing work towards improving the robustness and the applicability of the method with regard to the *surrounding curve* construction step; see §8.2 and Figs. 33–34.

2. Problem decomposition

As stated in the introduction, we aim to develop a method for modeling or reconstructing an object from its cross sections with a set of parallel planes, say $z = z_i, i = 1, ..., N$. It is permitted that the boundary of each cross section may consist of one or more non-nested disjoint Jordan curves, $C_{ij},\ j = 1, \dots, \mathcal{M}_i,\ i = 1, \dots, N$, referred to as *contours*, with $\mathcal{M}_i$ varying, in general, from section to section. Further, and in conformity with the case that is likely to arise in practice, it is assumed that we possess only discrete approximations of contours C_{ij} in the form of ordered point sets $\mathcal{P}_{ij} \in C_{ij}$. Then, we proceed to formulate the following

Problem 2.1 *Construct a composite G^1 parametric surface* $\mathbf{S}(u, v)$ *that interpolates the contour point sets* $\mathcal{P}_{ij} = \{\mathbf{P}_{ijk} \in \mathbb{E}^3, k = 0, \ldots, m_{ij} - 1\}$, $j = 1, \ldots, \mathcal{M}_i$, $i = 1, \ldots, N$, *and whose isoparametrics* $v = z_i$ *lie on the planes* $z = z_i$, $i = 1, \ldots, N$, *respectively.*

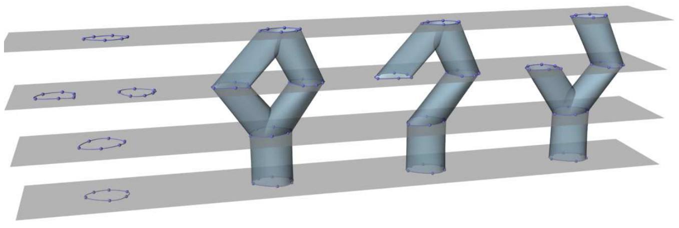


Fig. 1. One data set, three different correspondence choices

Regarding the well-posedness of the above problem, one can readily observe that, if $\mathcal{M}_i > 1$ for some *i*-plane, the *correspondence* question: *which of the contours on the i-plane should be connected to which of the contours on the* $(i + 1)$*-plane,* admits more than one, topologically legitimate, answers; see Fig. 1. Its solution can be represented by a *graph* structure $\mathcal{G}$, the vertices of which are the contours, while edges indicate their correspondences [29]. In what follows, we shall assume that $\mathcal{G}$ is user-defined, which is a definitely reasonable assumption in the design context. In general, the correspondence problem remains an important issue; see [31] and the references therein. Let us now consider the correspondence issue locally, i.e., between two consecutive planes. The connections between the contours of two neighboring planes can be represented by a set of disjoint connected subgraphs $\mathcal{G}_{i\ell}$ of $\mathcal{G}$, $\ell = 1, ..., L_i$, the structure of which can be of the following three types:

I. A list of two elements, consisting of two vertices, each on the *i*- and $(i + 1)$- planes, and an edge connecting them.
II. A two-level tree, consisting of one vertex on the *i*-plane, N_{i+1} vertices on $(i + 1)$-plane and N_{i+1} edges connecting them.
III. $\mathcal{G}_\ell$ has $N_i > 1$ vertices on the *i*-plane and $N_{i+1} > 1$ vertices on the $(i + 1)$-plane.

Each of the above local subgraph cases leads to a local interpolation subproblem, which will be referred to as: (I) the *"one-to-one"* subproblem, (II) the *"one-to-many"* or *branching* subproblem and (III) the *"many-to-many"* or *multiple branching* subproblem, respectively.

We can then proceed to decompose Problem 2.1 into a sequence of local subproblems of the above three types, provided we can eventually secure the validity of the G^1 global continuity condition. For this purpose, we shall assume that all data points $\mathbf{P}_{ijk}$ are enhanced with tangent vector estimates $\mathbf{T}_{ijk}$ along the v parametric direction. A way to obtain these estimates, that relies on a C^0 version of the methodology described in the ensuing three sections, is described in §6. The sought for decomposition of Problem 2.1 can then take the following form:

Problem 2.2 *Let be given the contour point sets* $\mathcal{P}_{ij}$, *their corresponding tangent-vector estimates* $\mathcal{T}_{ij} = \{\mathbf{T}_{ijk} \in \mathbb{R}^3, k = 0, \ldots, m_{ij} - 1\}$, $j = 1, \ldots, \mathcal{M}_i$, $i = 1, \ldots, N$, *and the graph* $\mathcal{G}$.

(i) Construct planar parametric G^1 curves $\mathbf{C}_{ij}(u)$ *and tangent-vector ribbons* $\mathbf{T}_{ij}(u)$ *that interpolate* $\mathcal{P}_{ij}$ *and* $\mathcal{T}_{ij}$, *respectively.*

(ii) Construct the solution $\mathbf{S}(u, v)$ *of Problem 2.1 as the union of all composite G^1 parametric patches* $\mathbf{S}_i(u, v)$, $z_i \leq v \leq z_{i+1}$, *that solve the local subproblems defined by the Hermite data prepared in (i) and the local subgraphs* $\mathcal{G}_{i\ell}$ *of* $\mathcal{G}$, $\ell = 1, ..., L_i$, $i = 1, \ldots, N - 1$.

3. Planar contours and their tangent ribbons

In order to construct the contour curves $\mathbf{C}_{ij}$ and the tangent ribbons $\mathbf{T}_{ij}$ referred to in Problem 2.2(i) we first have to supply the data-sets $\mathcal{P}_{ij}$ with lists $\mathcal{U}_{ij}$ of parametric nodes.

We adopt the method proposed in [28], where for convex point-sets $\mathcal{P}_{ij}$ the parametric node of each point is taken equal to its polar angle with respect to the center of the polygon $\mathcal{P}_{ij}$. If $\mathcal{P}_{ij}$ defines a non-convex polygon we employ a technique introduced in [9], which can roughly be described through the example shown in Fig. 2. The first step is to decompose the polygon into elementary convex hulls. This leads to a general planar tree structure. Then, starting from the last level of it, we project the internal vertices of each concavity onto the edge of the above level and continue with the next level until all vertices are projected onto the convex hull.

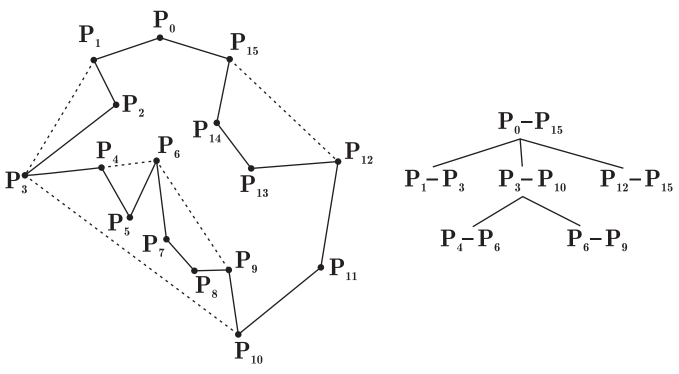


Fig. 2. Decomposing a non-convex polygon into a tree of its elementary convex hulls.

Having defined the knot vectors $\mathcal{U}_{ij}$ and exploiting the shape-preserving curve interpolation scheme in [25], we compute the contour curves $\mathbf{C}_{ij}$ and the tangent-vector distributions $\mathbf{T}_{ij}$ along them. Then, the solution of the *"one-to-one"* problem follows readily by using the standard skinning surface scheme.

4. The *"one-to-many"* Hermite problem

4.1. *Surrounding curve/surface*

Before proceeding with a detailed construction of the surrounding curve, let us denote by H the convex hull of all point sets $\mathcal{P}_{i+1,j}$, $j = 1, \ldots, \mathcal{M}$, on the $(i+1)$-plane, by $\mathcal{H}$ the boundary of H, and by H^c the complement, with respect to H, of the closed domains bounded by the simple polygons formed by the ordered point sets $\mathcal{P}_{i+1,j}$. The complement H^c is the union of some open disjoined domains Q_k, $k = 1, \ldots, R$. Each Q_k may be multiply connected, if it contains at least one of the point sets $\mathcal{P}_{i+1,j}$, or simply connected otherwise.

Definition 4.1 *Let Q_k, $k = 1, \ldots, R$, be the boundary (resp., outer boundary) of a simply (resp., multiply) connected domain Q_k. If Q_k shares common points with only one of the polygons formed by $\mathcal{P}_{i+1,j}$, Q_k will be called outer residual, otherwise it will be referred to as inner residual.*

The above definition is illustrated in Fig. 3.

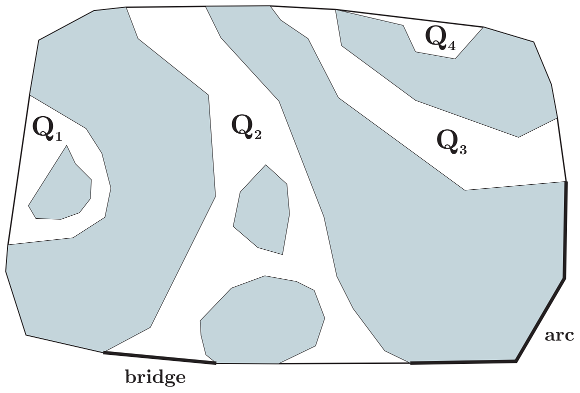


Fig. 3. A set of $\mathcal{M} = 6$ coplanar contours and their convex hull. The complement H^c consists of the multiply connected outer residual Q_1 (the outer boundary of Q_1 shares points with a single polygon), the multiply connected inner residual Q_2, the simply connected inner residual Q_3, and the simply connected outer residual Q_4.

In the sequel we shall restrict ourselves to H^c consisting only of simply connected inner and outer residuals. The general case is discussed in §8.2.

Definition 4.2 *Let H^S be the domain obtained by subtracting from H all the outer residuals. The boundary of H^S is a closed polygonal line, which will be called the surrounding polygon of $\mathcal{P}_{i+1,j}$, $j = 1, \ldots, \mathcal{M}$, and denoted by $\tilde{C}$. The segments of $\tilde{C}$ that do not belong entirely on any of the polygons formed by $\mathcal{P}_{i+1,j}$ will be called* bridges*, while the rest of will be referred to as* arcs*.*

Note that arcs are in general polygonal portions of the polygons formed by $\mathcal{P}_{i+1,j}$, while bridges are always line segments linking different $\mathcal{P}_{i+1,j}$'s; see Fig. 3.

In order to simplify the description of our method for solving the *"one-to-many"* problem and without any loss of generality, we shall henceforth assume that the $\mathcal{M}$ point-sets $\mathcal{P}_{i+1,j}$, $j = 1, \ldots, \mathcal{M}$, form only one inner residual Q. Under this assumption, we order the contours in such a way that the j-th bridge connects the polygons formed by $\mathcal{P}_{i+1,j}$ and $\mathcal{P}_{i+1,j+1}$. This in turn implies that each polygon, defined by $\mathcal{P}_{i+1,j}$, meets two bridges that split it into two parts: one that lies on the boundary of Q and one belonging to the surrounding polygon $\tilde{C}$. Analogously constructed contour curves $\mathbf{C}_{i+1,j}$ and tangent ribbons $\mathbf{T}_{i+1,j}$, are split into two parts: the outer one, corresponding to $\tilde{C}$, and the inner part corresponding to the inner residual.

Next, we provide $\tilde{C}$ with a parameterization $\tilde{\mathcal{U}}$ by employing the method described in §3. We reparameterize the outer part of $\mathbf{C}_{i+1,j}$ according to $\tilde{\mathcal{U}}$ and finally construct a C^1 connection between the outer parts of $\mathbf{C}_{i+1,j}$ and $\mathbf{C}_{i+1,j+1}$. This procedure defines the so called *surrounding curve* $\tilde{\mathbf{C}}$. Working analogously with the tangent ribbons $\mathbf{T}_{i+1,j}$ we define a tangent vector distribution $\tilde{\mathbf{T}}$ on it.

The so-called *surrounding surface*, can now be constructed by employing the standard Hermite skinning technique for interpolating the contour $\mathbf{C}_i$ on the i-plane and the surrounding curve $\tilde{\mathbf{C}}$ on the $(i + 1)$-plane along with their corresponding tangent ribbons $\mathbf{T}_i$ and $\tilde{\mathbf{T}}$, a process illustrated in Figs. 4–6.

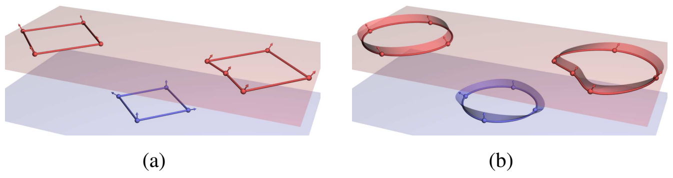


Fig. 4. (a) Data from an *"one-to-two"* example. (b) The initial contours and tangent ribbons on them.

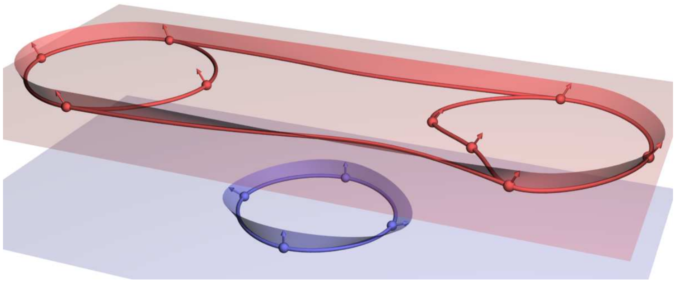


Fig. 5. The surrounding curve $\tilde{\mathbf{C}}$ and the contour $\mathbf{C}_i$ along with their tangent ribbons $\tilde{\mathbf{T}}$ and $\mathbf{T}_i$, respectively.

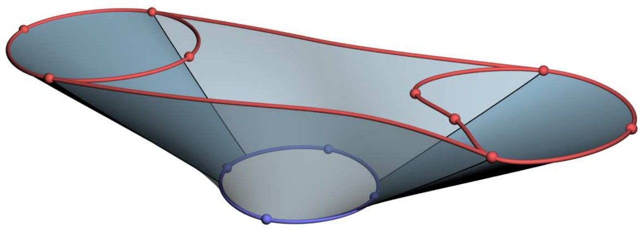


Fig. 6. The surrounding surface interpolating the contour $\mathbf{C}_i$ and the surrounding curve $\tilde{\mathbf{C}}$. The transparent parts correspond to the two bridges.

As one may observe in Fig. 6, in order for the final surface to interpolate the $\mathcal{P}_{i+1,j}$'s, the surrounding surface has to be trimmed near the bridges. The resulting new bounds of the trimmed surrounding surface along with the inner parts of the contour curves, which have not been interpolated yet, form a hole (see Fig. 8), which remains to be filled in order to complete the interpolation process. The following two paragraphs deal with these two problems.

4.2. *Trimming*

Let $\mathbf{S} : [u_S, u_E] \times [0, 1] \longrightarrow \mathbb{E}^3$ be a patch of the surrounding surface, corresponding to a bridge of the surrounding polygon. We aim to trim $\mathbf{S}(u, v)$ so that the trimmed patch will still interpolate the bridge end-points $\mathbf{B}^S = \mathbf{S}(u_S, 1)$ and $\mathbf{B}^E = \mathbf{S}(u_E, 1)$.

The *domain curve* $\mathbf{X}(t)$, $t \in [0, T]$, of the trimming curve to be constructed, is selected to be a C^1 quadratic spline in order to:

(i) satisfy Hermite boundary conditions at bridge end-points,
(ii) reduce the number of hole sides and,
(iii) keep as low as possible the degree of the trimming curve $\mathbf{Y}(t) = \mathbf{S} \circ \mathbf{X}(t)$.

Eventually, $\mathbf{X}(t)$ is set to be symmetric with respect to the middle of the interval $[u_S, u_E]$, consisting of two linear and two quadratic segments as shown in Fig. 7. The extent of the linear segments defines a free parameter $r \in (0, 1)$, which will be used in the hole filling construction. Note that C^1 continuity of $\mathbf{X}(t)$ dictates dependency between T and r, which in our case takes the form $T = 2(r + 1)$.

Along the so-constructed trim curve $\mathbf{Y}(t)$ (see Fig. 7) we need to define a cross-tangent vector distribution that is needed for filling the hole, while ensuring G^1-continuity along its boundary. This distribution, say $\mathbf{Y}^{(1)}(t)$, can be expressed as a linear combination of the first partial derivatives of $\mathbf{S}(u, v)$: $\mathbf{Y}^{(1)}(t) = a_S(t)\frac{\partial \mathbf{S}(u,v)}{\partial u} + b_S(t)\frac{\partial \mathbf{S}(u,v)}{\partial v}$. For the linear part, $t \in [0, 1-r]$, of the domain curve $\mathbf{X}(t)$, we take $\mathbf{Y}^{(1)}(t) = \frac{\partial \mathbf{S}(u,v)}{\partial u}$, whereas for the quadratic part, $t \in [1-r, 1+r]$, we use cubic blending functions $a_S(t)$, $b_S(t)$ fulfilling the Hermite boundary conditions:

$$a_S(1-r) = 1,\ a_S(1+r) = 0, \quad b_S(1-r) = 0,\ b_S(1+r) = 1,$$

$$a'_S(1-r) = a'_S(1+r) = b'_S(1-r) = b'_S(1+r) = 0,$$

where prime denotes differentiation with respect to t. It is worth noticing that it can be easily proved that the so constructed cross-tangent vector distribution is nowhere collinear with the tangent of the trimming curve.

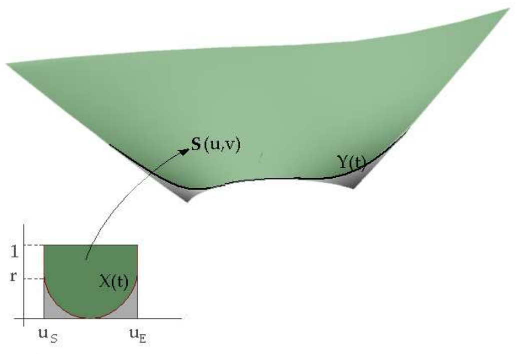


Fig. 7. The domain curve $\mathbf{X}(t)$ and its map $\mathbf{Y}(t)$ on $\mathbf{S}(u, v)$.

4.3. *Hole filling*

The trimming of the surrounding surface provides us with a hole with $2\mathcal{M}$ C^1-continuous sides, which remains to be filled in order to give a complete solution to the *"one-to-many"* problem; see Fig. 8.

Our method adopts the methodology presented in [16] (see Fig. 9). According to [16], given a user-defined center $\mathbf{K}$ and a

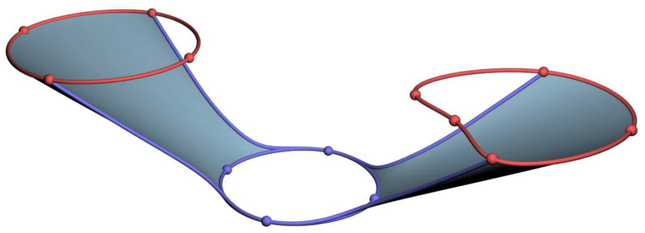


Fig. 8. The hole formed after trimming the surrounding surface.

set of guide curves $\mathbf{G}_\kappa(t)$, $t \in [0, t_\kappa]$, $\kappa = 1, \ldots, 2\mathcal{M}$, that connect $\mathbf{K}$ with the middle parameter point $\mathbf{M}_\kappa$ on each side, along with appropriately defined derivative distributions on them, the hole can be filled by a Gordon-Coons patchwork. In our case we are restricted to G^1 continuity, offering nevertheless the additional advantages of permitting $\mathbf{G}_\kappa$ to be splines and preserve data symmetries in the following sense: *if* $\mathbf{G}_\kappa$ *is planar and the cross tangent vectors at the end-points of* $\mathbf{G}_\kappa$ *are symmetric with respect to this plane, then the tangent ribbons in between should be symmetric as well.*

The hole center $\mathbf{K}$ is user-defined, lying between the i- and $(i+1)$-planes with its xy-projection set initially to be equal to the centroid of the polygon formed by those $\mathbf{M}_\kappa$ lying on the $(i+1)$-plane, in case it is convex, or just averaging them, otherwise. The user-defined z-coordinate K_z of $\mathbf{K}$, $z_i < K_z < z_{i+1}$, is linked with the parameter r, used in §4.2, for controlling the extent of the linear parts of the domain curves $\mathbf{X}(t)$, as follows:

$$r = \frac{K_z - z_i}{z_{i+1} - z_i}. \tag{4.1}$$

A natural choice for the tangent plane through $\mathbf{K}$ is the plane $z = K_z$, on which we define the so-called star vectors $\mathbf{v}_\kappa$ [16], that will be used as boundary tangent vectors of the guide curves $\mathbf{G}_\kappa$ emanating from $\mathbf{K}$. The direction of $\mathbf{v}_\kappa$ is determined by projecting the vector connecting $\mathbf{K}$ with $\mathbf{M}_\kappa$ onto $z = K_z$, while its length is taken as the average of the lengths of the tangent vectors of the neighboring hole sides at $\mathbf{M}_{\kappa-1}$ and $\mathbf{M}_{\kappa+1}$. Finally, we force the surface to exhibit planar behavior in the neighborhood of $\mathbf{K}$ by imposing zero second-order derivatives at $\mathbf{K}$.

As a preprocessing step, implied by the Gordon-Coons scheme in [16], we have to reparameterize pairs of contour segments (segments $\mathbf{M}_{\kappa-1}\mathbf{M}^L_\kappa$, $\mathbf{M}_{\kappa+1}\mathbf{M}^R_\kappa$ in Fig. 10) which bound neighboring patches that are *topologically opposite* to their common edge (segment $\mathbf{KM}_\kappa$ in Fig. 10), so that eventually they are defined over the same parametric interval. Employing linear reparameterization and requesting minimization of the length variation of their end tangent vectors, we are led to a well-posed least-squares problem.

Then, we proceed to construct the family of guide curves $\mathbf{G}_\kappa$, which is divided into two sets, those $\mathbf{G}_\kappa$ that connect $\mathbf{K}$ with contour segments (solid curves in Fig. 10) and those connecting $\mathbf{K}$ with trimming curves (dashed curves in Fig. 10). Guide curves belonging to the first set are constructed as shape-preserving 3D polynomial curves (cf. [13]) that satisfy the boundary conditions implied by the corresponding star vectors

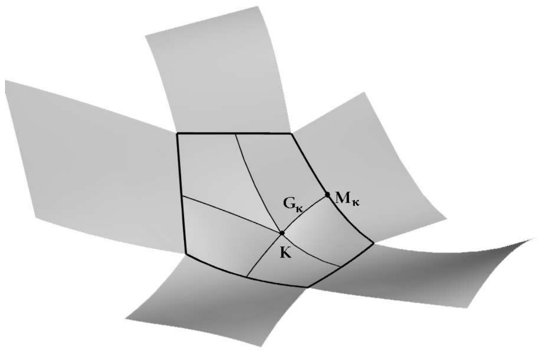


Fig. 9. The Gordon-Coons patches covering a five-sided hole.

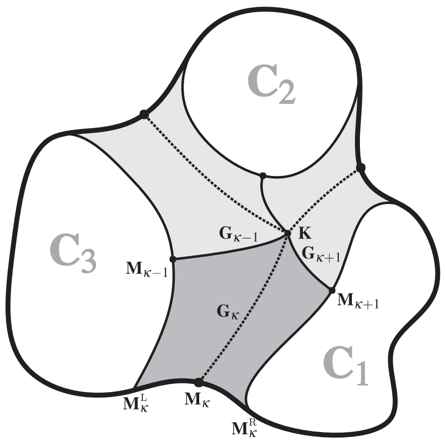


Fig. 10. Top view of three contours, $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$, on the $(i+1)$-plane forming one inner residual and the Gordon-Coons patchwork covering the six-sided hole.

and the requested planar behavior at **K**, as well as positional and tangential information readily available at $\mathbf{M}_\kappa$. For the second set of guide curves, we use shape-preserving interpolation splines; the *xy*-projections of the interpolation points are obtained from the middle isoparametric of the linear skinning surface interpolating the topologically opposite contour segments. This choice serves our aim to embed the shape of the contributing contours to the guide curve. The *z*-coordinate of the interpolation points is calculated with the aid of the cubic Hermite polynomial that connects **K** with $\mathbf{M}_\kappa$; see Fig. 11.

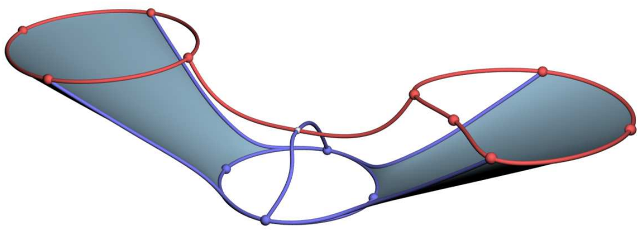


Fig. 11. The constructed guide curves.

It remains to build the tangent ribbons $\mathbf{L}_\kappa$ and $\mathbf{R}_\kappa$ that are to be imposed on every pair of neighboring patches, sharing the same guide curve $\mathbf{G}_\kappa$ as common boundary. These are expressed as

$$\mathbf{L}_\kappa(t) = \lambda_\kappa(t)\,(\mathbf{l}_\kappa(t) - \mathbf{r}_\kappa(t)) + \mu_\kappa(t)\frac{d\mathbf{G}_\kappa(t)}{dt} \tag{4.2}$$

and

$$\mathbf{R}_\kappa(t) = \nu_\kappa(t)\,(\mathbf{r}_\kappa(t) - \mathbf{l}_\kappa(t)) + \xi_\kappa(t)\frac{d\mathbf{G}_\kappa(t)}{dt} \tag{4.3}$$

where $\mathbf{l}_\kappa(t)$ and $\mathbf{r}_\kappa(t)$ denote the tangent ribbons that we would have obtained if step 6 in [16] had been applied to the two neighboring patches. Obviously, $\mathbf{R}_\kappa(t)$ and $\mathbf{L}_\kappa(t)$ are coplanar with $\frac{d\mathbf{G}_\kappa(t)}{dt}$ while $\lambda_\kappa(t)$, $\mu_\kappa(t)$, $\nu_\kappa(t)$ and $\xi_\kappa(t)$ are functions such that $\mathbf{L}_\kappa(t)$ and $\mathbf{R}_\kappa(t)$ share the same boundary conditions with $\mathbf{l}_\kappa(t)$ and $\mathbf{r}_\kappa(t)$, respectively. The computed star vectors $\mathbf{v}_\kappa$, along with the requested planar behavior at **K** and the available geometric information at $\mathbf{M}_\kappa$ provide a set of boundary conditions that uniquely determine $\mathbf{l}_\kappa$ and $\mathbf{r}_\kappa$ as cubic polynomials. As for the functions $\lambda_\kappa(t)$, $\mu_\kappa(t)$, $\nu_\kappa(t)$ and $\xi_\kappa(t)$, it can be proved that they should satisfy the following constraints:

$$\lambda_\kappa(0) = \frac{\mathbf{v}_{\kappa-1} \times \mathbf{v}_\kappa}{(\mathbf{v}_{\kappa-1} - \mathbf{v}_{\kappa+1}) \times \mathbf{v}_\kappa}, \quad \lambda_\kappa(t_\kappa) = \frac{1}{2}, \tag{4.4}$$

$$\mu_\kappa(0) = \frac{\mathbf{v}_{\kappa+1} \times \mathbf{v}_{\kappa-1}}{\mathbf{v}_\kappa \times (\mathbf{v}_{\kappa-1} - \mathbf{v}_{\kappa+1})}, \quad \mu_\kappa(t_\kappa) = 0, \tag{4.5}$$

$$\nu_\kappa(0) = \frac{\mathbf{v}_{\kappa+1} \times \mathbf{v}_\kappa}{(\mathbf{v}_{\kappa+1} - \mathbf{v}_{\kappa-1}) \times \mathbf{v}_\kappa}, \quad \nu_\kappa(t_\kappa) = \frac{1}{2}, \tag{4.6}$$

$$\xi_\kappa(0) = \frac{\mathbf{v}_{\kappa-1} \times \mathbf{v}_{\kappa+1}}{\mathbf{v}_\kappa \times (\mathbf{v}_{\kappa+1} - \mathbf{v}_{\kappa-1})}, \quad \xi_\kappa(t_\kappa) = 0, \tag{4.7}$$

$$\lambda'_\kappa(0) = -\nu'_\kappa(0), \quad \mu'_\kappa(0) = \xi'_\kappa(0), \tag{4.8}$$

$$\lambda'_\kappa(t_\kappa) = \mu'_\kappa(t_\kappa) = \nu'_\kappa(t_\kappa) = \xi'_\kappa(t_\kappa) = 0. \tag{4.9}$$

Restricting $\lambda_\kappa(t)$, $\mu_\kappa(t)$, $\nu_\kappa(t)$ and $\xi_\kappa(t)$ to be cubic polynomials and requiring,

$$\lambda'_\kappa(0) = \mu'_\kappa(0) = \nu'_\kappa(0) = \xi'_\kappa(0) = 0, \tag{4.10}$$

gives us unique representations for these polynomials. Then $\mathbf{L}_\kappa$ and $\mathbf{R}_\kappa$ can be calculated using (4.2) and (4.3).

The filling hole process can now be accomplished (see Fig. 12) by employing the Gordon-Coons scheme, once we can guarantee the validity of the compatibility conditions at the corner points of the hole. This is achieved by scaling the tangent ribbon along each trimming curve with a suitably defined cubic with vanishing end-derivatives.

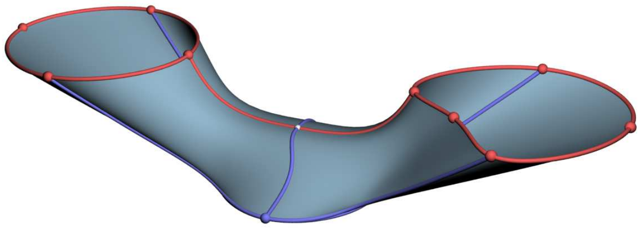


Fig. 12. The final *"one-to-two"* surface.

5. The *"many-to-many"* Hermite problem

The *"many-to-many"* problem is handled via a direct extension of the *"one-to-many"* methodology described in the previous section. In fact the two methodologies are identical up to the construction step of the surrounding surface; see Fig. 13.

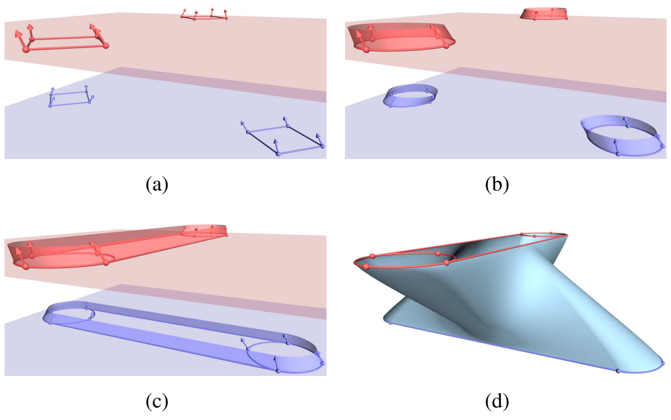


Fig. 13. Handling the *"many-to-many"* problem up to the surrounding-surface step: (a) data sets, (b) contours along with tangent ribbons, (c) surrounding contours with tangent ribbons, (d) surrounding skinning surface.

Next, by taking two v-isoparametric curves on the surrounding surface we create a separating zone (see Fig. 14) between the i- and the $(i+1)$-planes, that reduces the problem into two independent to each other *"one-to-many"* subproblems, for which the necessary surrounding surfaces are already there; see Figs. 15 and 16.

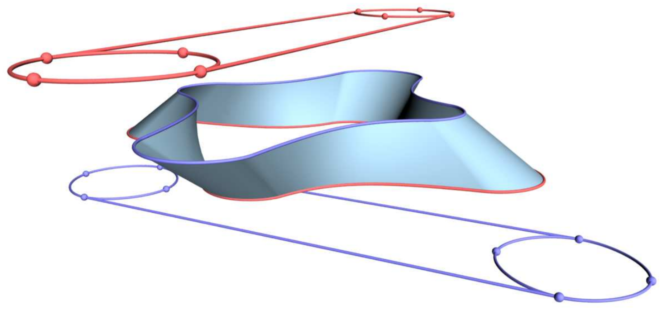


Fig. 14. The separating strip.

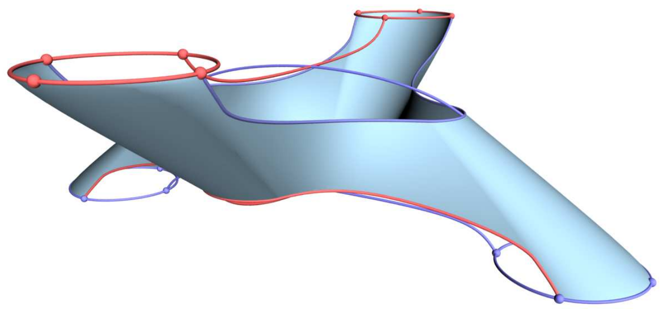


Fig. 15. The surrounding surface after trimming, along with the constructed guide curves.

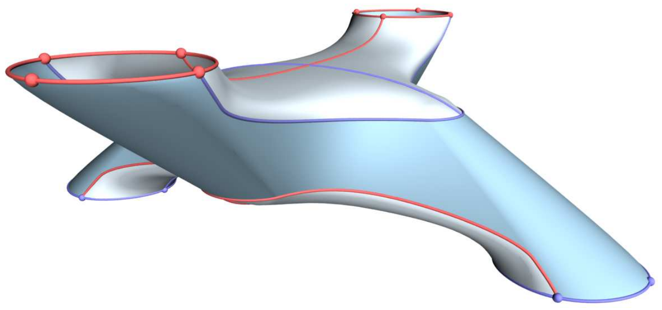


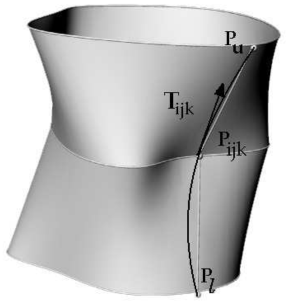
Fig. 16. The surface after hole filling.

6. Tangent-vector estimation

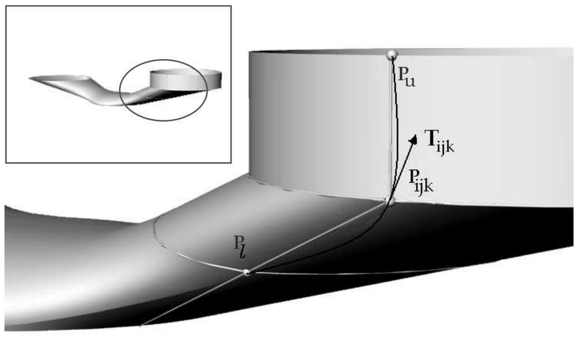
Tangent vectors, especially those on the intermediate contours, are rarely given in real world applications, thus a way for estimating them is needed. For this purpose we use a C^0 version of the so far developed methodology in the following sense: skinning and hole filling does not take into account any tangential information.

Suppose we want to obtain an estimate of $\mathbf{T}_{ijk}$ at $\mathbf{P}_{ijk}$. We propose to take as $\mathbf{T}_{ijk}$ the tangent of a z-parameterized parabola, that passes through $\mathbf{P}_{ijk}$ and two points, $\mathbf{P}_u$ and $\mathbf{P}_l$, lying on the two u-isoparametrics of the neighboring patches of the C^0 surface, meeting at $\mathbf{P}_{ijk}$.

It is reasonable to expect that the estimation varies contin-



(a)



(b)

Fig. 17. (a) Tangent vector estimation on an intermediate contour between skinning surfaces. (b) Tangent vector estimation of an intermediate contour which bounds at least one Gordon-Coons patch.

uously with respect to u. To meet this constraint we take $\mathbf{P}_u$ and $\mathbf{P}_l$ on the intersection of the u-isoparametric with the furthest, with respect with the i-plane, v-isoparametrics for which continuity with regard to u is preserved. So in the simplest case when the neighboring surfaces are both skinning (see Fig. 17(a)), $\mathbf{P}_u$ and $\mathbf{P}_l$ are taken on the upper and lower contours, respectively. In the more elaborate case of Fig. 17(b), where the upper neighboring surface is skinning and the lower one is a patchwork of Gordon-Coons patches and a trimmed skinning surface, $\mathbf{P}_l$ lies on the v-isoparametric with $v = K_z$, namely the z-coordinate of the hole center $\mathbf{K}$. Such a choice places $\mathbf{P}_l$ on the furthest from the i-plane v-isoparametric that guarantees continuity with respect to u.

7. Examples

The kernel of the presented methodology, which is the solution of the *"one-to-many"* branching problem, is illustrated here through two synthetic and two realistic examples.

The first synthetic example aims to illustrate that the method is able to retain the symmetries of the data. More specifically, the contour point sets lie on two planes, $z = 0$ and $z = 10$, and form an *"one-to-three"* problem with one inner residual, as shown in Fig. 18. The data point sets are taken from two circles of radius 5 and 8.5 units, respectively, and are arranged in such a way that their formation exhibits symmetry with respect to three planes meeting along the z-axis (see Fig. 19). Furthermore, the imposed tangent vectors are such that they do not violate this symmetry, i.e., they are parallel to the z-axis and share the same length. The steps of the method, as described in §4, namely skinning, trimming and hole-filling, are illustrated through Figs. 20, 21 and 22, respectively. Finally, Fig. 23 reveals the capability of the method to preserve data symmetries, by depicting the color map of the Gaussian curvature of the

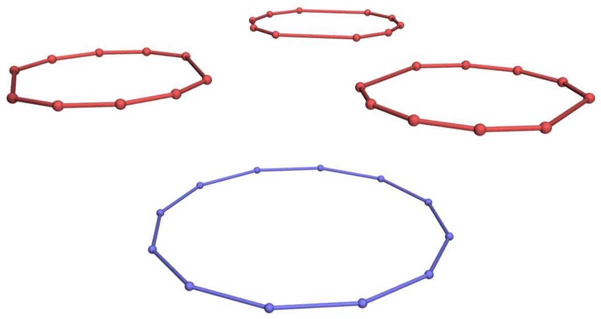


Fig. 18. **A symmetric data set**: The contour point sets.

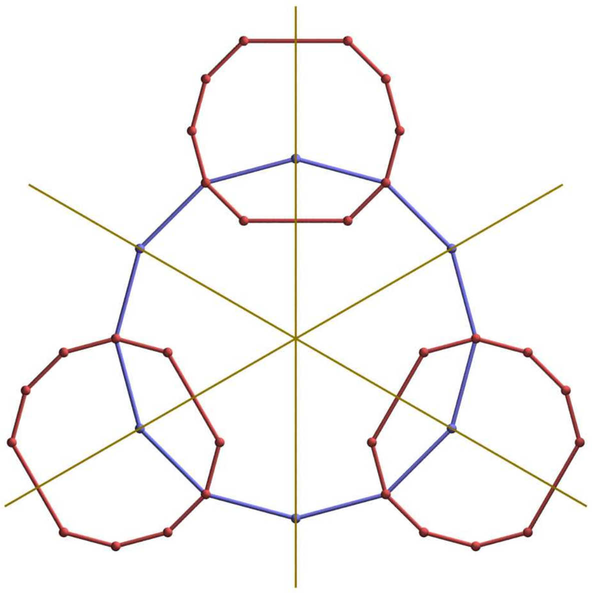


Fig. 19. **A symmetric data set**: Top view of the data points and the traces of the planes of symmetry on the xy-plane.

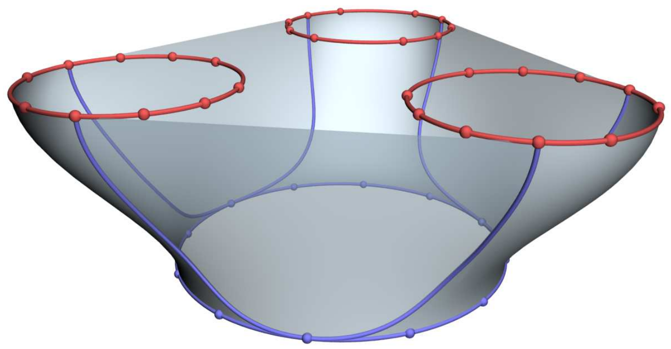


Fig. 20. **A symmetric data set**: The surrounding surface with the trimmed area shown transparently.

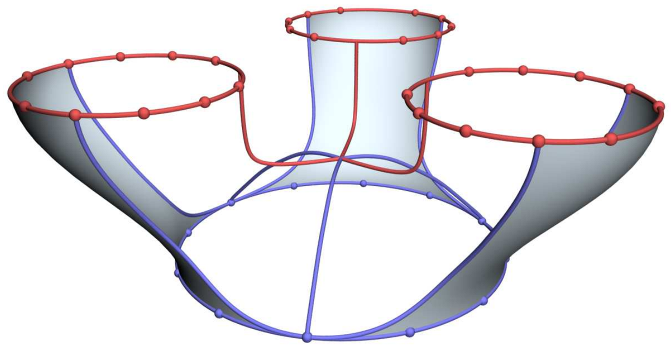


Fig. 21. **A symmetric data set**: The boundary of the six-sided hole along with the constructed guide curves which form six four-sided patches.

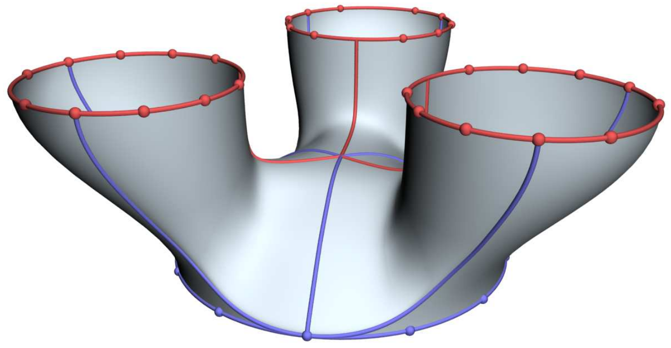


Fig. 22. **A symmetric data set**: The final *"one-to-three"* branching surface after hole filling.

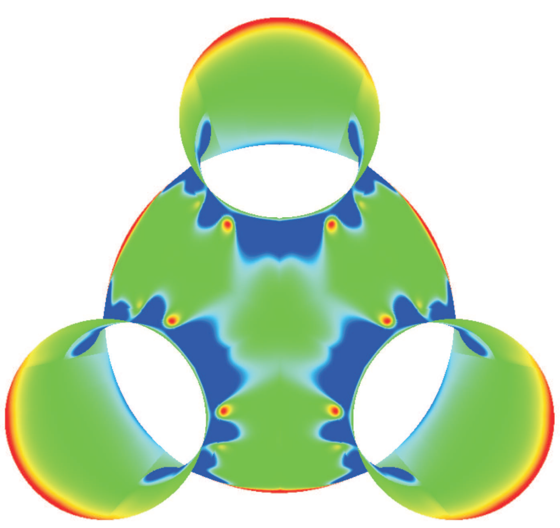


Fig. 23. **A symmetric data set**: Top view of the surface with the color map of its Gaussian curvature, revealing the capability of the method to preserve data symmetries.

final surface.

The second synthetic example is an *"one-to-two"* problem, where the two coplanar contour point-sets (see Fig. 24) are selected to be highly non-symmetric and non-convex, while their convex hull boundaries penetrate each other. The corresponding surrounding surface with its trimmed area is depicted in Fig. 25, while the boundary of the resulting four-sided hole along with the constructed guide curves is illustrated in Fig. 26. The final result, depicted in Fig. 27, reveals the gorge-like nature of the underlying surface.

The first of the two practical examples deals with the design of a detergent container with handle. The contour point-sets lie on eight planes and form six *"one-to-one"* and two *"one-to-two"* branching problems, as illustrated in Fig. 28(a). Data points in this figure are also endowed with longitudinal tangent vectors, devised with the aid of the technique summarized in §6. The final outcome of the method is depicted in the right part of this figure; see Fig. 28(b). The intermediate steps of constructing and trimming the two surrounding surfaces, as well as the constructed guide curves required for filling the two holes, are illustrated in Figs. 29(a) and 29(b), respectively.

The second practical example is related to computer-aided ship design. More specifically, Fig. 30 depicts sections from a hull with bulbous bow. The proposed approach enables the designer to handle uniformly and efficiently the topological changes that occur in ship sections as we move from amidships towards the bulbous-bow and bow-flare areas. As opposed to our assumption for closed point-sets, the data in question are mainly open, which is bypassed by mirroring initial data with respect to a plane parallel and well above the deck plane; see Fig. 31(a). In the branching area we are thus faced with an *"one-to-three"* problem with two inner residuals.

One can readily note from Fig. 31(b) that the bridges selected do not lie on the convex hull of the three coplanar contour point sets, which is a basic choice made in §4.1, but have been placed near the longitudinal symmetry plane of the hull, the so-called centerplane. This choice can be justified on the basis of hydrodynamic performance criteria — design waterline should not be bluff, nevertheless it reveals that the policy of adhering bridges to the convex hull may become restrictive. In this connection, we are currently working towards enriching our method with alternative surrounding curves, that take into account the geometry and relative position of the contours with respect to their convex hull; see §8.2.

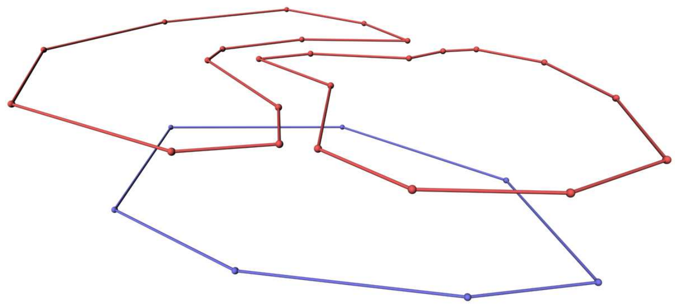


Fig. 24. **A non-symmetric data set**: The contour point-sets.

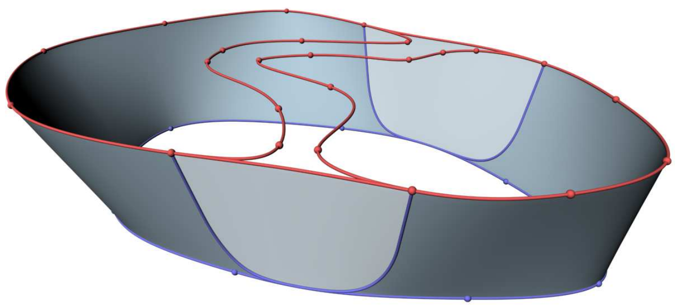


Fig. 25. **A non-symmetric data set**: The surrounding surface with the trimmed area shown transparently.

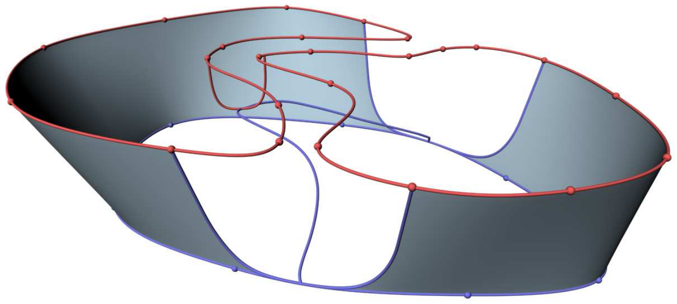


Fig. 26. **A non-symmetric data set**: The boundary of the four-sided hole along with the constructed guide curves which form four quadrilateral patches.

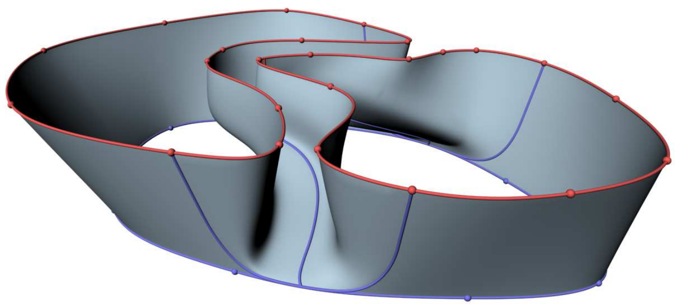
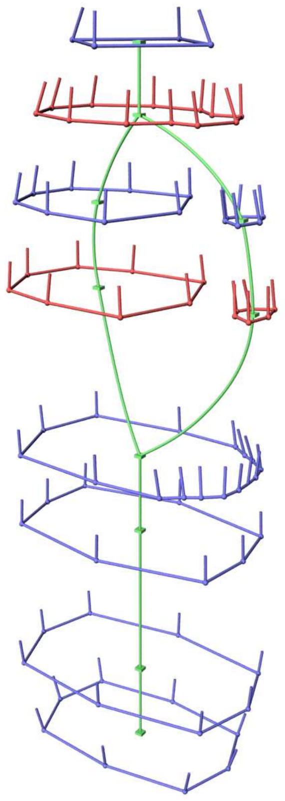
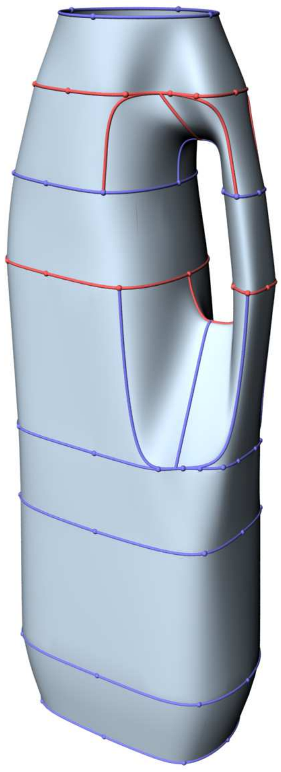
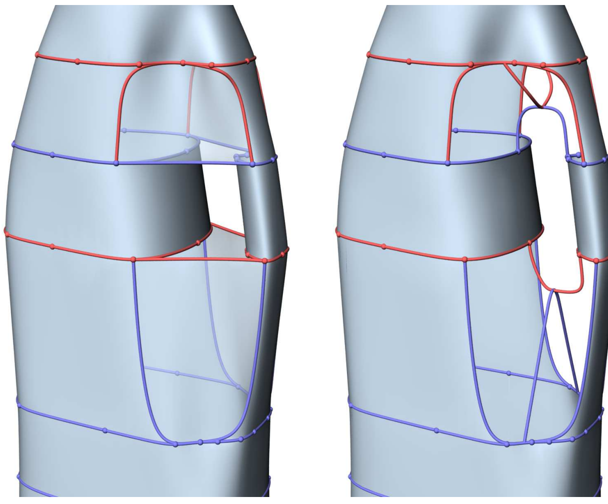


Fig. 27. **A non-symmetric data set**: The final *"one-to-two"* branching surface after hole filling.

(a) (b)

Fig. 28. **The container example**: (a) Contour point sets, tangent vector estimates and the correspondence graph. (b) The final container surface.



(a) (b)

Fig. 29. **The container example**: (a) The two surrounding surfaces with the corresponding trimmed areas shown transparently. (b) The two constructed guide curve families.

Coming back to the graphical output of the example in question, the surrounding surface, the holes with the constructed guide curves and the branching surfaces are shown in Figs. 31(c)–(e), correspondingly. The final outcome of the hull design process is illustrated in Fig. 32.

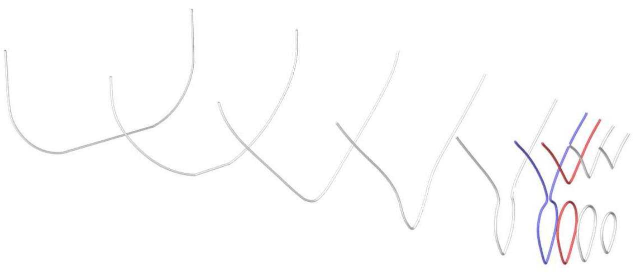
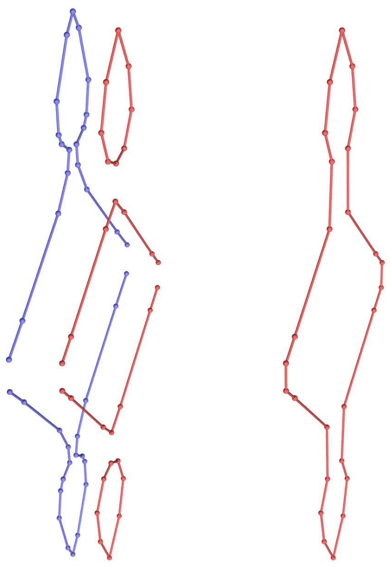
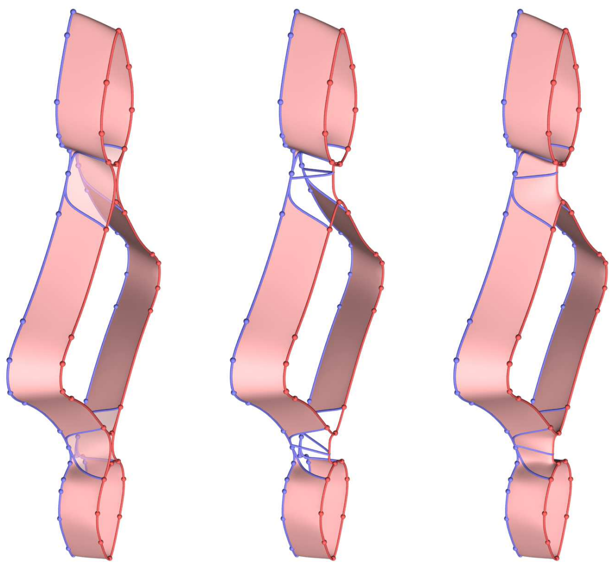


Fig. 30. **The bulbous hull example**: Input cross sections.



(a) (b)



(c) (d) (e)

Fig. 31. **The bulbous hull example**: (a) The contour point sets mirrored. (b) The surrounding polygon. (c) The surrounding surface with the trimmed area shown transparently. (d) The boundary of the holes, along with the constructed guide curves. (e) The final branching surfaces after hole filling.

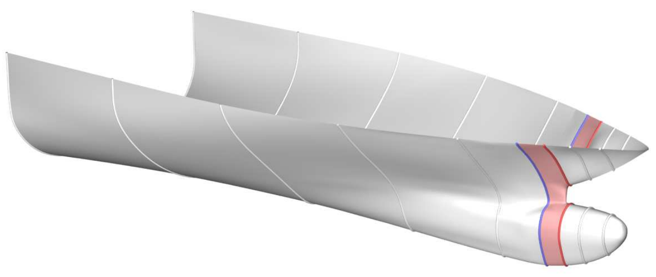


Fig. 32. **The bulbous hull example**: The final hull.

8. Summary and ongoing work

8.1. *Summary*

The algorithm solving the *"one-to-many"* problem is the core of the paper. It seems to have many advantages over other algorithms in the literature, due to the analytic structure of the final surface patchwork, the G^1 smoothness and its ability to preserve the shape and symmetry of the input data. The method was designed to be easily implemented in an ordinary CAD system, thus it uses structures (such as polynomial rectangular surface patches), which constitute the basis of most of these systems. Since the proposed method is aiming to provide a framework for designing branching surfaces, it leaves to the involved designer a number of free parameters up to his alley. The definition of the trimming curve or the construction of the guide curves or the heuristic rules for the information on the hole center, can be freely modified in order to better customize the method for a particular application. In the following paragraphs we provide some specific comments and a discussion on current work towards removing a topological instability of the algorithm and improving its applicability regarding the surrounding curve construction step.

To start with, the methodology adopted for computing the parametrization of the point-sets is fast and stable. Moreover, it provides an acceptable automatic solution to the parameter-matching problem for constructing the necessary skinning surfaces. On the other hand, however, the algorithm used for constructing the contour curves [25], guarantees the construction of convexity-preserving curves independently of the chosen parametrization. Note also that the same algorithm can guarantee that no intersections between different contour curves may occur, provided the polygons formed by $\mathcal{P}_{ij}$ do not intersect one another.

The surrounding curves we construct at each plane can rapidly and robustly handle point-sets $\mathcal{P}_{i+1,j}$ and the tangential information supplied by the already constructed contour curves. On the other hand, we should note an apparent topological instability of the proposed algorithm, with respect to the relative position of the above point-sets to their convex hull. Let us assume, e.g., that the polygons formed by those point-sets all contribute to the convex hull, i.e., there is one (or more) simply connected residual(s). Then, a small perturbation of one of the polygons may result in the formation of a multiply connected inner/outer residual, a situation which is not treated by the algorithm. In Section 8.2 we describe some ideas on how to extend the definition of the surrounding curve so as to connect not only point-sets that touch their convex hull, but also point-sets *near* the convex hull.

The rest of the method involves skinning surfaces and Gordon-Coons interpolation, which are both widely used by the CAGD community. They also enable us to easily extend the algorithm in order to incorporate NURBS curves. Based on these underlying structures, we can increase the order of continuity of the resulting surface, at the cost of enriching the data sets with higher order information.

As a final note on the approach presented in this paper, the solution to the *"one-to-many"* problem enabled us to give an analogously reliable solution to the *"many-to-many"* problem, a case so far handled by very few algorithms.

8.2. *Ongoing work*

As mentioned above, the way the surrounding curves are constructed in each plane exhibit a topological instability: point-sets $\mathcal{P}_{i+1,j}$ that touch the convex hull and thus participate in the construction of the surrounding curve, may, under a small perturbation, seize to touch the convex hull and thus fail to participate in the construction of the surrounding curve. Moreover, point-sets near, but not on, the convex hull cannot participate in the surrounding curve and thus cannot be taken into account by our algorithm when constructing the surrounding surface and eventually the final interpolatory surface. One way to, at least partially, remedy this deficiency is to extend the surrounding curve definition by including point-sets that are near the convex hull. There seems to be no unique notion of "nearness", and in what follows we shall describe two ways of measuring proximity. Our discussion below is limited to contour curves $\mathbf{C}_{i+1,j}$ that are convex, although we believe that the same approach can be extended to the case of non-convex contours.

The key idea is to use the planar *Euclidean Voronoi diagram* of the set of contours (viewed as solid compact objects with convex boundary). Before diving into the details, let us give some definitions and discuss some properties of such Voronoi diagrams (see also [26]). Given a set $\mathcal{S}$ of n convex objects in the plane $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, that are pairwise disjoint, define the Euclidean distance $\delta(p, S_k)$ of a point p in the plane, that belongs to the complement of the union of the (interiors of the) S_i's, from some $S_k \in \mathcal{S}$ as $\delta(p, S_k) = \min_{q \in S_k} d(p, q)$, where $d(\cdot, \cdot)$ stands for the Euclidean distance between two points in $\mathbb{E}^2$. Define the *Voronoi region* V_k of S_k to be the set $V_k = \{p \in \mathbb{E}^2 \,|\, \delta(p, S_k) \leq \delta(p, S_\ell),\ \ell \neq k\}$. The Voronoi region of S_k is simply the set of points in the plane that are closer, or at equal distance, to S_k than to any other object in $\mathcal{S}$. The locus of points in $\mathbb{E}^2$ that belong to exactly two Voronoi regions is called a *Voronoi edge*, whereas points in the plane that belong to at least three Voronoi regions are called *Voronoi vertices*. The collection of Voronoi regions (or faces), edges and vertices is called the Voronoi diagram $\mathcal{V}(\mathcal{S})$ of $\mathcal{S}$. For a set of pairwise disjoint convex objects, such as $\mathcal{S}$, the Voronoi diagram $\mathcal{V}(\mathcal{S})$ is a planar graph of size $O(n)$. Its faces are simply connected, and there is one such face per object in $\mathcal{S}$. In fact, the Voronoi region V_k of S_k fully contains S_k in its interior, and, moreover, V_k is *star-shaped* with respect to the medial axis of S_k (see [26] for more details).

The dual graph of $\mathcal{V}(\mathcal{S})$ is called the Delaunay graph $\mathcal{D}(\mathcal{S})$ of $\mathcal{S}$. The vertices of $\mathcal{D}(\mathcal{S})$ are in 1–1 correspondence with the objects in $\mathcal{S}$, whereas the faces of $\mathcal{D}(\mathcal{S})$ are in 1–1 correspondence with the Voronoi vertices in $\mathcal{V}(\mathcal{S})$. The Delaunay graph is typically understood via its compactified version: the Delaunay graph contains a fictitious vertex at infinity, connected with all objects that appear on the convex hull $H(\mathcal{S})$ of $\mathcal{S}$ (or more accurately to all the vertices on the boundary of the infinite face of $\mathcal{D}(\mathcal{S})$ — note that an object in $\mathcal{S}$ may appear multiple times on $H(\mathcal{S})$). Under the non-degeneracy assumptions that:

(i) no three objects in $\mathcal{S}$ share a common tangent line (that leaves them in the same halfplane) and,

(ii) no four objects in $\mathcal{S}$ have a common tangent circle (that fully lies in the complement of the union of the objects in $\mathcal{S}$),

the Delaunay graph $\mathcal{D}(\mathcal{S})$ consists of only triangular faces, i.e., faces with three edges. These "triangles" may not be embeddable with straight line segments, since it is possible to have two triangles sharing two edges. In the compactified version of $\mathcal{D}(\mathcal{S})$, the objects of $\mathcal{S}$ along $H(\mathcal{S})$ are connected via an edge. We call such an edge a *convex hull edge*. At least one of the neighboring triangles of a convex hull edge is an *infinite* triangle, i.e., a triangle one vertex of which is the fictitious vertex at infinity — the other neighboring triangle of a convex hull edge may also be an infinite triangle. Although the discussion below applies to non-degenerate sets of objects, the ideas readily extend to possibly degenerate configurations.

In the surrounding curve context, the objects in the set $\mathcal{S}$ are the convex objects bounded by the contour curves $\mathbf{C}_{i+1,j}$. What is interesting, in this context, is that the Voronoi diagram $\mathcal{V}(\mathcal{S})$, or equivalently its dual Delaunay graph $\mathcal{D}(\mathcal{S})$, captures a lot of proximity information, which we exploit in order to extend the definition of surrounding curve given in §4.2. Let us denote by $\mathcal{S}_0$ the set of objects in $\mathcal{S}$ that contribute to $H(\mathcal{S})$ (see Fig. 33). Now, let $\mathcal{S}_1$ be the set of objects in $\mathcal{S} \setminus \mathcal{S}_0$ that belong to a triangle in $\mathcal{D}(\mathcal{S})$, the other two vertices of which belong to $\mathcal{S}_0$. In other words, we focus on non-convex-hull objects than belong to triangles adjacent to convex hull edges in $\mathcal{D}(\mathcal{S})$. If the objects in $\mathcal{S}_0$ can be thought of as the 0-level objects when approaching $\mathcal{S}$ from infinity, $\mathcal{S}_1$ are the next level (1-level) objects that we encounter after the objects in $\mathcal{S}_0$.

The objects in $\mathcal{S}_1$ are our candidate objects for inclusion in the surrounding curve. Let S_λ be some object in $\mathcal{S}_1$ and let T be a triangle in $\mathcal{D}(\mathcal{S})$ that connects S_λ with two objects S_μ and S_ν in $\mathcal{S}_0$; notice that T may not be unique, i.e., S_λ may be the non-convex-hull vertex of many triangles in $\mathcal{D}(\mathcal{S})$ that are adjacent to convex hull edges (see Fig. 33(b)). Let us denote by C_T the Voronoi circle corresponding to T, and let Σ_λ, Σ_μ and Σ_ν be the points of tangency of C_T with S_λ, S_μ and S_ν, respectively. Let $\Delta_{\lambda\mu\nu}$ denote the triangle defined by the three points Σ_λ, Σ_μ and Σ_ν, and let $\alpha_{\lambda,\mu\nu}$ be the angle of $\Delta_{\lambda\mu\nu}$ at Σ_λ, and $K_{\lambda,\mu\nu}$ and corresponding cone, the apex of which is Σ_λ. Moreover, let $R_{\lambda\mu}$ and $R_{\lambda\nu}$ be the rays emanating from Σ_λ, belonging to $K_{\lambda,\mu\nu}$, that are tangent to S_μ and S_ν (see Figs. 33(a) and 33(c)). We shall denote by $\tilde{K}_{\lambda,\mu\nu}$ the cone defined by $R_{\lambda\mu}$ and $R_{\lambda\nu}$, and by $\tilde{\alpha}_{\lambda,\mu\nu}$ and angle of $\tilde{K}_{\lambda,\mu\nu}$ at its apex Σ_λ. Note that $\tilde{K}_{\lambda,\mu\nu}$ is essentially the region of space visible from Σ_λ through S_μ and S_ν.

The two angles $\alpha_{\lambda,\mu\nu}$ and $\tilde{\alpha}_{\lambda,\mu\nu}$ can be used to measure the proximity of S_λ to $H(\mathcal{S})$. As S_λ approaches $H(\mathcal{S})$, both $\alpha_{\lambda,\mu\nu}$ and $\tilde{\alpha}_{\lambda,\mu\nu}$ tend to π. In view of this property we use these angles in order to decide whether or not an object in $\mathcal{S}_1$ is to be added to the set of objects defining the surrounding curve. The decision can be made via threshold values, which should be considered as design parameters. The designer can choose a value in the interval $[0, \pi]$ as the minimum acceptable value for either $\alpha_{\lambda,\mu\nu}$ or $\tilde{\alpha}_{\lambda,\mu\nu}$. Setting this threshold to zero implies that all objects in $\mathcal{S}_1$ are to participate in the surrounding curve, whereas choosing the threshold to be equal to π essentially reduces to constructing the surrounding curve as in §4.2. What is the most interesting feature of the approach described above, is that it not only gives us a way to augment the set of objects used to construct the surrounding curve, but also defines the circular order with which the surrounding curve is to touch these objects. To be more specific, the surrounding curve, as discussed on §4.2, touches the objects in $H(\mathcal{S})$ in the order they appear as we move around $H(\mathcal{S})$. When an object $S_\lambda \in \mathcal{S}_1$ is chosen, according to one of the two afore-mentioned criteria, to participate in the construction of the surrounding curve, we place it in between S_μ and S_ν in the circular list of objects in $H(\mathcal{S})$; here S_μ and S_ν are two objects in $H(\mathcal{S})$ connected with S_λ via a triangle $T \in \mathcal{D}(\mathcal{S})$, and either the angle $\alpha_{\lambda,\mu\nu}$ or $\tilde{\alpha}_{\lambda,\mu\nu}$ (depending on which criterion we use) is above the threshold value set.

The approach, described in the above paragraphs, for extending the set of contours participating in the construction of the surrounding curve is illustrated in Figs. 33 and 34. Fig. 33 shows the resulting surrounding curves (in fact C^0-versions of them) for two data sets and for the two criteria mentioned above. Fig. 34 illustrates the effect of the threshold value chosen for the second criterion (namely, the one based on the angles $\tilde{\alpha}_{\lambda,\mu\nu}$). We depict the output surrounding curve for six threshold values ranging from 0°, in which case all objects in $\mathcal{S}_1$ participate in the surrounding curve — one of them is actually touched three times, to 150°, in which case the surrounding curve coincides with the convex hull $H(\mathcal{S})$.

References

[1] C. L. Bajaj, E. J. Coyle, and K.-N. Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Models & Image Proc.*, 58:524–543, 1996.

[2] G. Barequet, D. Shapiro, and A. Tal. Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices. *Visual Computer*, 16:116–133, 2000.

[3] G. Barequet and M. Sharir. Piecewise-linear interpolation between polygonal slices. *Comp. Vision & Image Underst.*, 63:251–272, 1996.

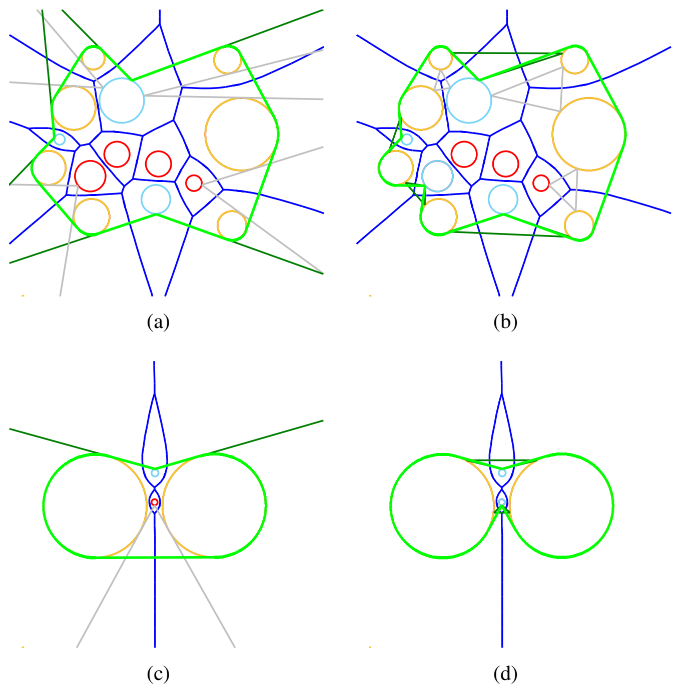


Fig. 33. The two criteria, applied to two data sets, for choosing the sites from $\mathcal{S}_1$ that are to participate in the construction of the surrounding curve. The Voronoi diagram is shown in blue. A C^0-version of the surrounding curve is shown in light green. The objects in $\mathcal{S}_0$ are shown in gold. The objects from $\mathcal{S}_1$ chosen to participate in the construction of the surrounding curve are shown in light blue. The objects in red are either in $\mathcal{S} \setminus (\mathcal{S}_0 \cup \mathcal{S}_1)$ or were rejected. The cones $\tilde{K}_{\lambda,\mu\nu}$ or triangles $\Delta_{\lambda\mu\nu}$ are shown in dark green and gray (corresponding to chosen and rejected objects, respectively). In all cases the threshold value is 90°. (a) The first data set, using the angles $\tilde{\alpha}_{\lambda,\mu\nu}$. (b) The first data set, using the angles $\alpha_{\lambda,\mu\nu}$. (c) The second data set, using the angles $\tilde{\alpha}_{\lambda,\mu\nu}$. (d) The second data set, using the angles $\alpha_{\lambda,\mu\nu}$.

[4] S. Batnitzki, H. I. Price, P. N. Cook, L. T. Cook, and S. J. Dwyer-III. Three-dimensional computer reconstruction from surface contours for head CT examinations. *J. of Computer Assist. Tomogr.*, 5:60–67, 1981.

[5] S. Bedi. Surface design using functional blending. *CAD*, 24:505–511, 1992.

[6] A. Bentamy, F. Guibault, and J.-Y. Trépanier. Cross-sectional design with curvature constraints. *CAD*, 37:1499–1508, 2005.

[7] H. N. Christiansen and T. W. Sederberg. Conversion of complex contour lines into polygonal element mosaics. *Computer Graphics (SIGGRAPH '78 Proceedings)*, 12:187–192, 1978.

[8] S. Cohen, G. Elber, and R. Bar-Yehuda. Matching of freeform curves. *CAD*, 29:369–378, 1997.

[9] A. B. Ekoule, F. C. Peyrin, and C. L. Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Trans. on Graph.*, 10:182–199, 1991.

[10] M. S. Floater and G. Westgaard. Smooth surface reconstruction from cross-sections using implicit surfaces. Report STF42, Sintef, 1996.

[11] H. Fuchs, Z. M. Kedem, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Commun. ACM*, 20:693–702, 1977.

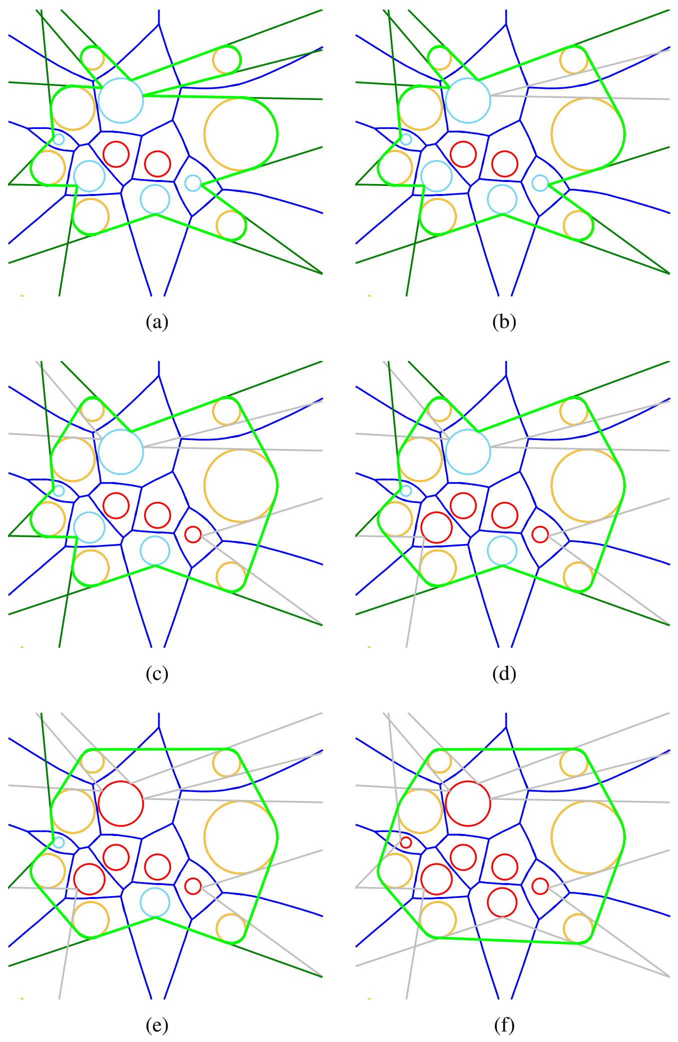


Fig. 34. The second criterion (i.e., the one based on the angles $\tilde{\alpha}_{\lambda,\mu\nu}$) applied to the first data set in Fig. 33 for different threshold values. The colors are as in Fig. 33. The threshold values are: (a) 0°, (b) 30°, (c) 60°, (d) 90°, (e) 120°, (f) 150°.

[12] K. Fujimura and E. Kuo. Shape Reconstruction from Contours Using Isotopic Deformation. *Graphical Models & Image Proc.*, 61:127–147, 1999.

[13] N. C. Gabrielides and P. D. Kaklis. C^4 interpolatory shape-preserving polynomial splines of variable degree. *Computing Suppl.*, 14:119–134, 2001.

[14] T. N. T. Goodman and G. T. D. Greig. Matching and choice of parameter in sectional interpolation. *International Journal of Shape Modeling*, 4:197–208, 1998.

[15] T. N. T. Goodman, B. H. Ong, and K. Unsworth. Reconstruction of C^1 closed surfaces with branching. In G. Farin, H. Hagen, and H. Noltemeier, editors, *Geometric Modelling*, pages 101–115. Springer, Vienna, 1993.

[16] J. Hahn. Filling polygonal holes with rectangular patches. In W. Strasser and H.-P. Seidel, editors, *Theory and practice of geometric modeling*, pages 81–91. Springer, Berlin, 1989.

[17] M. J. Herbert, C. B. Jones, and D. S. Tudhope. Three-dimesional reconstruction of geoscientific objects from

serial sections. *Visual Computer*, 11:343–359, 1995.

[18] M. Hohmeyer and B. A. Barsky. Skinning rational B-spline curves to construct an interpolatory surface. *Graphical Models & Image Proc.*, 53:511–521, 1991.

[19] F. Jaillet, B. Shariat, and D. Vandorpe. Periodic B-spline surface skinning of anatomic shapes. In *9th Canadian Conference in Computational Geometry*, pages 199–210, Kingston, Canada, 1997.

[20] J. Jeong, K. Kim, H. Park, H. Cho, and M. Jung. B-Spline surface approximation to cross sections using distance maps. *Adv. Manufact. Techn.*, 15:876–885, 1999.

[21] J. K. Johnstone and K. R. Sloan. A philosophy for smooth contour reconstruction. *Comp. Suppl.*, 13:153–163, 1998.

[22] M. W. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics*, 13:75–84, 1994.

[23] B. Jüttler. Sectional curvature-preserving interpolation of contour lines. In A. Le Méhauté, C. Rabut, and L. L. Schumaker, editors, *Curves and Surfaces with Applications in CAGD*, pages 203–210. Vanderbilt University Press, Nashville TN, 1997.

[24] P. D. Kaklis and A. I. Ginnis. Sectional-curvature preserving skinning surfaces. *CAGD*, 13:601–619, 1996.

[25] P. D. Kaklis and M. I. Karavelas. Shape-preserving interpolation in $\mathbb{R}^3$. *IMA J. Numer. Anal.*, 17:373–419, 1997.

[26] M. I. Karavelas and M. Yvinec. The Voronoi Diagram of Planar Convex Objects. In *11th Annual European Symposium on Algorithms (ESA '03)*, volume 2832 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2003.

[27] E. Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM J. Res. Devel.*, 19:2–11, 1975.

[28] A. L. Marsan and D. Dutta. Computational techniques for automatically tiling and skinning branched objects. *Computers & Graphics*, 23:111–126, 1999.

[29] D. Meyers, S. Skinner, and K. Sloan. Surfaces from contours. *ACM Trans. on Graph.*, 11:228–258, 1992.

[30] E. Moschino, Y. Maurin, and P. Andrey. Joint registration and averaging of multiple 3D anatomical surface models. *Comp. Vision & Image Underst.*, 101:16–30, 2006.

[31] L. G. Nonato, A. J. Guardos-Vargas, R. Minghim, and M. C. F. De Oloveira. Beta-connection: Generating a family of models from planar cross sections. *ACM Trans. on Graph.*, 24:1239–1258, 2005.

[32] H. Park and K. Kim. Smooth surface approximation to serial cross-sections. *CAD*, 28:995–1005, 1996.

[33] L. Piegl and W. Tiller. Algorithm for approximate NURBS skinning. *CAD*, 28:699–706, 1996.

[34] S. Schmitt, J. F. Evers, G. Duch, M. Scholz, and K. Obermeyer. New methods for the computer-assisted 3D reconstruction of neurons form confocal image stacks. *NeuroImage*, 23:1283–1298, 2004.

[35] T. W. Sederberg, K. S. Klimaszewski, M. Hong, and K. Kaneda. Triangulation of branching contours using area minimization. *Intern. J. of Comp. Geom. & Appl.*, 8:389–406, 1998.

[36] Y. Shinagawa and T. L. Kunii. The homotopy model: A generalized model for smooth surface generation from cross sectional data. *Visual Computer*, 7:72–86, 1991.

[37] N. M. Sirakov and F. H. Muge. A system for reconstructing and visualising three-dimensional objects. *Computers & Geosciences*, 27:59–69, 2001.

[38] K. R. Sloan and J. Painter. Pessimal Guesses May Be Optimal: A Counterintuitive Search Result. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10:949–955, 1988.

[39] E. Welzl and B. Wolfers. Surface reconstruction between simple polygons. In *1st Annual European Symposium on Algorithms (ESA '93)*, volume 726 of *Lecture Notes in Computer Science*, pages 397–408, Berlin/New York, 1993. Springer-Verlag.