Topology-preserving polyhedral design using ribbon-based multi-sided patches

Péter Salvi[a,*], Jázmin Szörfi[b], Márton Vaitkus[a] and Tamás Várady[a]

[a]*Department of Control Engineering and Information Technology, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111, Budapest, Hungary*

[b]*Shapr3D Ltd., Akadémia u. 6., H-1054, Budapest, Hungary*

ARTICLE INFO

Keywords:
polyhedral design
multi-sided surfaces
ribbons
generalized B-spline surfaces
geometric continuity

Abstract

Polyhedral design is a well-established paradigm for creating complex free-form shapes by smoothing control polyhedra. Several approaches exist, including recursive subdivision and direct algorithms. We introduce a direct method where multi-sided and multi-connected, ribbon-based patches are stitched together. While traditional methods are based on control polyhedra with only convex faces, our approach can handle concave angles and multiple hole loops, as well; moreover, the free-form patchwork has *exactly* the same topological structure as the input control polyhedron. The algorithm consists of three basic steps: (i) computing an auxiliary polyhedron, (ii) defining a general topology curve network and determining cross-derivative (ribbon) data along the patch boundaries, (iii) computing Generalized B-spline surfaces. Furthermore, using a shape parameter, a family of models with the same topological structure can be generated. Several test examples are provided to demonstrate the capabilities of this new method.

1. Introduction

Polyhedral design is an important 3D modeling paradigm, widely utilized to create complex free-form shapes in various fields such as engineering, medicine, computer animation and digital art. The basic idea is that nice, intuitive shapes can be generated by smoothing a control polyhedron. As the user relocates vertices (control points) or faces, the corresponding approximating surface model evolves in a natural and predictable manner.

Existing approaches can be distinguished by the algorithms used to transform the polyhedron into smooth free-form surfaces. While *recursive subdivision* methods produce a sequence of refined polyhedra that converges to a limit surface, *direct methods* create patchworks that interpolate curve networks explicitly constructed from the polyhedron. The well-known Doo–Sabin [8] and Catmull–Clark [4] recursive subdivision methods, as well as the majority of the direct approaches, assume that the control polyhedron consists only of convex polygonal faces. If this assumption is not valid, it becomes necessary to subdivide the corresponding faces by artificial edges in a preprocessing step. While this may be easy for simple geometries, in some complex cases – in particular for objects with hole loops – it can be inconvenient and ambiguous, and may produce extraordinary vertices with high valency; see examples later in Section 3.

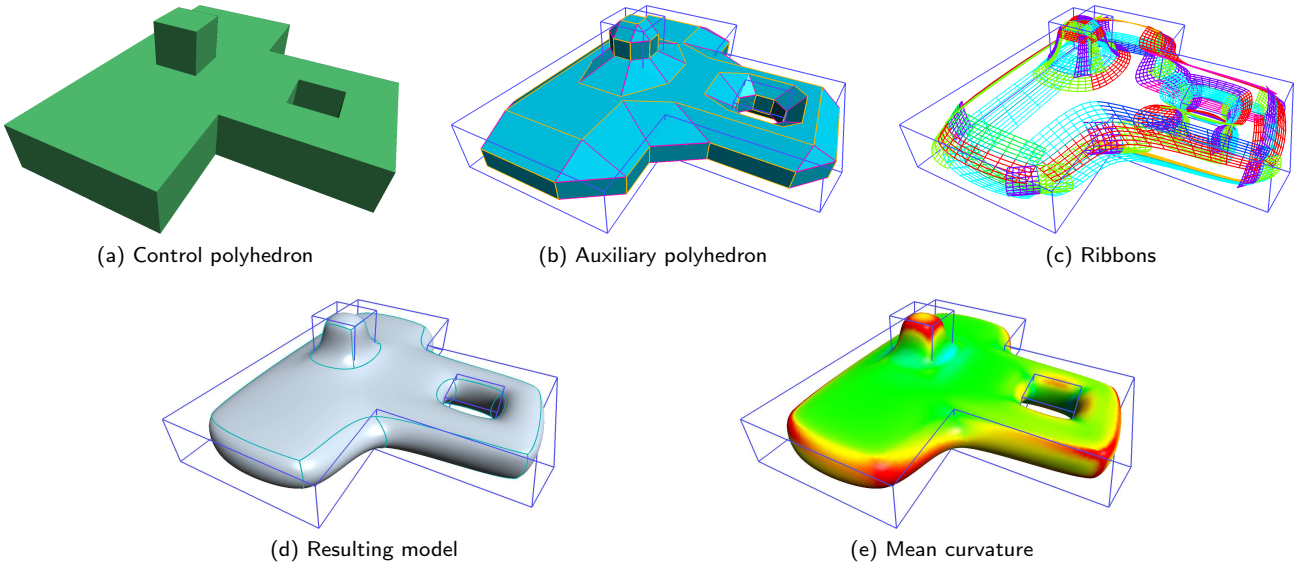
In this paper, we present a direct polyhedral surfacing method that builds on recent advances in multi-sided surface modeling, such as the Generalized Bézier (GB) [46] and the Generalized B-spline (GBS) [43] patches. These patches overcome the aforementioned convexity limitation, as they can handle surfaces with concave boundaries and interior hole loops. Instead of restricted singly connected polygons, the patches are defined over curved, multi-connected domains. The resulting patchwork topologically reproduces the control polyhedron; to our best knowledge, this approach is the first of its kind. Currently the method ensures only G^1 continuity, although visual tests show acceptable numerical curvature continuity (NG^2), as well.

The basic concept is depicted in Figure 1. Figure 1a shows the input control polyhedron, from which an *auxiliary polyhedron* is created by means of chamfering (Fig. 1b). By connecting the centroids of the *vertex facets* (faces of the auxiliary polyhedron associated with an original vertex) a curve network is established, and a set of ribbon surfaces is defined based on the auxiliary polyhedron (Fig. 1c). These ribbons determine cross-derivatives for the curved domain patches (Fig. 1d). A mean curvature map of the model is presented in Figure 1e.

Actually a family of objects is defined, as the chamfering step can produce various auxiliary polyhedra depending on a *fullness* value. This shape parameter allows the user to select an intermediate state between strongly blended (filleted) and strongly smoothed configurations.

Our method is well-suited for defining a wide range of free-form solid objects where aesthetics are of primary concern. Furthermore, we have found this approach particularly useful in curve-network-based

salvi@iit.bme.hu (P. Salvi)
ORCID(s): 0000-0003-2456-2051 (P. Salvi); 0000-0003-2064-763X (M. Vaitkus); 0000-0001-9547-6498 (T. Várady)



(a) Control polyhedron (b) Auxiliary polyhedron (c) Ribbons

(d) Resulting model (e) Mean curvature

Figure 1: Workflow: from control polyhedron to surface model.

design for creating an initial network with a naturally associated set of G^1-compatible ribbons.

A preliminary version of our work appeared in a local conference proceedings [41]. The present paper represents a substantial extension of that early version, including new methods for chamfering, auxiliary polyhedron construction, ribbon generation and the use of Generalized B-spline surfaces [43], as well as a more thorough evaluation and comparison to existing methods.

The paper is structured as follows. After reviewing prior work in Section 2, we analyze the difficulties of the convex division (Sec. 3), then we describe the basic steps of the algorithm (Sec. 4). The method of creating the auxiliary polyhedra (Sec. 5) is followed by computing the curve networks and the ribbons of the patches (Sec. 6). The GBS surface equations are revisited in Section 7. Finally, the capabilities and variations of our method are demonstrated through several examples in Section 8. Suggestions for future work conclude the paper.

2. Previous work

Polyhedron-based surface design has an extensive literature – see e.g. [31] for a comprehensive survey of the topic. Often the term *control cage* is used; this permits non-planar boundary polygons for the perimeter loop of the faces, as well. Almost all existing methods make a variety of assumptions about the geometry or topology of the control polyhedron that our work aims to alleviate.

2.1. Recursive subdivision and hole-filling

The most widely used family of methods are based on recursive subdivision [6, 32], including the well-known Doo–Sabin [8] and Catmull–Clark [4] schemes. Subdivision rules can theoretically be applied to arbitrary polyhedra, but non-convex or multiply connected faces must be first split into convex polygons, as discussed in Section 3. A large variety of alternative subdivision schemes have been proposed [3], including some that allow for semi-sharp creases/blends [6], T-nodes [38], or even non-manifold topology [27]. However, none of these methods relax the limitations of the classical methods regarding non-convex faces. Some subdivision variants can handle *open* polyhedra with concave *boundary corners* [2] —in contrast our goal is to generate smooth surfaces from *closed* polyhedra with general faces.

Another interpretation of standard subdivision methods is that they produce tensor-product B-splines in regular regions, while filling the multi-sided "holes" corresponding to irregular neighborhoods with an *infinite* sequence of patches. A variety of methods have been proposed to fill irregular holes with a *finite* number of patches. Possible solutions include (i) covering the hole with standard 4- or 3-sided patches meeting at a common extraordinary vertex [33, 30, 26, 16, 9], (ii) defining local surface pieces and blending them into their regular neighborhood [21, 1], (iii) using a single *multi-sided* surface [45] to fill the hole [22, 51, 13]. All published methods are currently limited to control cages with singly connected faces.

Yang [48] investigated the use of so-called moving B-spline curves and surfaces to allow the flexible modeling of sharp/rounded corners and straight edges – their

method however suffers from the same topological limitations as standard B-splines.

2.2. Direct methods

Another family of methods for polyhedral design directly generates smoothly connected surface patches from the control polyhedron.

2.2.1. Polyhedral splines

So-called *polyhedral* or *surface splines* [29, 31] devise rules to construct a network of standard patches corresponding to elements of the control polyhedron. Irregular elements are typically converted into quadrilateral or triangular patches organized into a *central split* configuration. Note that many of the previously mentioned hole-filling methods can also be interpreted this way. Currently existing polyhedral splines are not directly applicable to non-convex or multi-connected faces, and – despite recent progress on improving their flexibility [15, 17, 9, 12] – they can only handle a limited set of polyhedron topologies [19].

A central element to our method is the operation of chamfering / corner cutting, which is also commonly employed in polyhedral constructions that generalize bi-quadratic splines [28, 23, 34, 18]. In contrast to these works, we employ the chamfered polyhedron only as an auxiliary object, and preserve the polyhedron topology with our patch network.

2.2.2. Multi-sided schemes

An attractive possibility is to create genuine multi-sided representations [45] from irregular regions, see e.g. [25, 47, 50]. In a pioneering early work, Loop and DeRose [25] constructed polynomial curves and cross-derivatives from the control polyhedron, which are then interpolated with S-patches [24]. They proposed two variants that were analogous to quadratic and cubic splines (corresponding to dual and topology-preserving schemes in our classification). Similar constructions have been applied to other types of multi-sided patches, as well [47, 50]. Although recently various multi-sided representations have been extended to support concave and multi-connected polygonal domains (e.g. S-patches [40], Generalized Bézier [37, 46] and B-spline [43] patches), such enhanced representations – to our best knowledge – have not been employed for polyhedral design.

A direct approach close to our own is the basis of the SuperD system developed by Rockwood et al. [35], where a dual network of curves and cross-derivative vector fields is constructed from the control polyhedron, and the resulting G^1 boundary conditions (ribbons) are interpolated using transfinite patches [44]. For purely quadrilateral control meshes, the resulting network has only 4-valent vertices, which makes G^1 continuity easy to achieve. In contrast to this dual construction, we present a method that preserves the topological structure of the polyhedron, and handles arbitrary non-convex and multiply connected faces, while ensuring G^1 smoothness (with higher order continuity conceptually possible, and left for future work).

2.3. Manifold-based methods

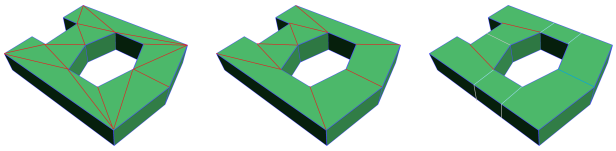
An alternative class of direct approaches create approximant/interpolant surface patches based on the local neighborhood of each mesh element (typically each vertex) of the polyhedron, then blend together the local pieces on their overlapping domains [11, 7]. Such "manifold-based" constructions can achieve a high order of formal smoothness (even C^∞ is possible [49, 36]), but they are currently limited to quadrilateral or triangular meshes, and controlling the final shape tends to be more difficult than for patch-based methods.

3. Difficulties with non-convex faces

As discussed earlier, the majority of traditional methods – including recursive subdivision and dual topology schemes – require the insertion of artificial subdividing edges into the non-convex faces of the control polyhedron. Without these edges, errors would occur when centroids are computed. For instance, the Doo–Sabin method interpolates the centroid of the faces, the first step of the Catmull–Clark scheme relies on subdivision at the centroids, and the central-split constructions typically build on quadrilaterals meeting at a center point. However, for concave or multi-connected faces the centroid may lie near the boundary or even outside the face entirely, which typically leads to shape artifacts or geometrically invalid objects.

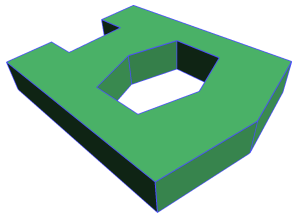
While defining artificial subdividing edges can be a straightforward task for simple faces, difficulties may arise when we deal with concave angles and hole loops. The most common practical approach is to triangulate each face by connecting every vertex to its "visible" neighbors. However, this often produces too many narrow triangles, introduces unwanted asymmetries and increases vertex valency, all of which can further degrade shape quality. Figure 2a illustrates this problem, where the top face of a test model with a six-sided hole is subdivided into 15 triangles.

Alternatively, one can attempt to create general convex faces, for example, by merging adjacent triangles into convex polygons. Such an example is shown in Figure 2b, where the triangle subdivision is successfully transformed into a set of quads. Nevertheless, to determine an optimal convex division is a challenging task. As shown in Figure 2c, dividing faces by dropping perpendicular lines from concave corners to the closest boundary segments may seem intuitive, but it necessitates inserting new division points that propagate further subdivision into the surrounding side faces, as well.

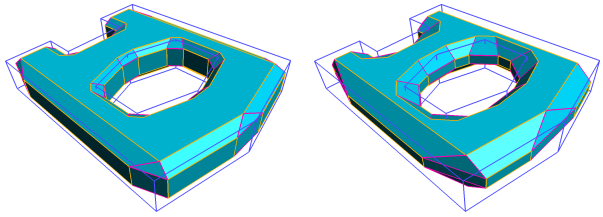


(a) Triangulation (b) Merging triangles (c) By shortest path

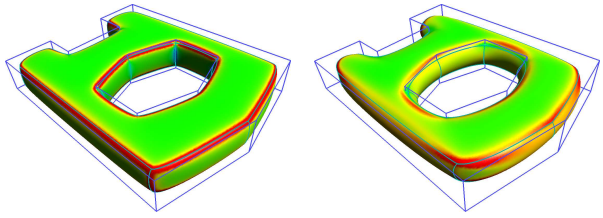
Figure 2: Decomposing a complex face into convex parts.



(a) Input control polyhedron



(b) Auxiliary polyhedra



(c) Surface models with mean curvature

Figure 3: Smoothing/blending using the fullness parameter (left: 0.4, right: 0.7).

In summary, subdividing faces into convex polygons is a complex task involving both geometry and topology. Face configurations are generally ambiguous, consequently these may lead to different shapes depending on the subdivision applied. These limitations motivate our *topology-preserving approach*, which eliminates the need for artificial subdivision and produces a natural structure conforming strictly to the control polyhedron.

4. The workflow of the algorithm

Our input is a general topology control polyhedron and a fullness parameter defined in $(0, 1)$. The algorithm consists of the following phases. First an auxiliary polyhedron is generated by chamfering the edges and the vertices of the control polyhedron (see Sec. 5). The dimensions of the auxiliary polyhedron are determined by the fullness value. Low values indicate narrow chamfers and keep the auxiliary polyhedron relatively close to the control polyhedron, thus objects resembling blended (filleted) models will be created. High values create larger chamfers, shrinking the auxiliary polyhedron into the interior of the control polyhedron, thus the smoothing effect will be stronger. This is illustrated in Figure 3 where two variants are shown: a "blended" object with fullness $= 0.4$ and a "smoothed" object with fullness $= 0.7$. In this sense, fullness defines a continuous transition between blending and smoothing. (Here – and on all curvature maps throughout the paper – green indicates zero curvature, while the transition to red/blue is computed linearly by hue, on a symmetrically clamped range.)

The second step is the definition of a free-form curve network by connecting the centroids of the vertex facets (i.e., faces of the auxiliary polyhedron associated with an original vertex), see Section 6. In the current project cubic Bézier curves are used. At concave corners, the adjacent cubic segments are concatenated into a single B-spline with C^1 continuity. The auxiliary polyhedron determines cross-derivatives in such a way that the ribbons associated with the opposite sides of a boundary segment are G^1-continuous, thus the adjacent patches will connect with G^1 continuity, as well.

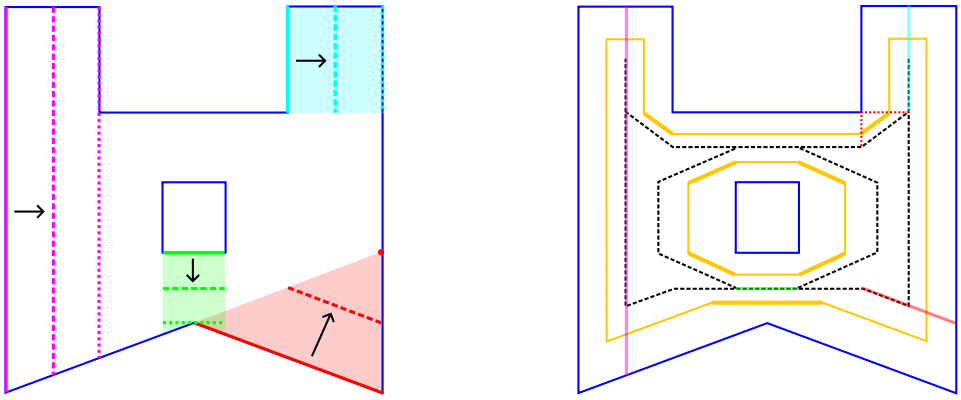
The third step is the actual patch generation (see Sec. 7). Each patch has a perimeter loop and possibly hole loops, and from these a curved domain (i.e., a planar domain with curved boundaries) can be determined. The curved domain is parameterized, and local coordinates are computed for each boundary curve. Then the multi-sided patches are evaluated. In our project we use Generalized B-spline patches, but – as it will be shown in Section 8 – other ribbon-based patches can also be applied.

5. Auxiliary polyhedron

As we have seen in the previous section, the first step is to create an auxiliary polyhedron that will serve as a basis for generating the curve network and the ribbon surfaces. We shrink each face and connect the resulting vertices similarly to a Doo–Sabin step, but instead of convex combinations, we use *edge offsetting* as the basis of our algorithm. In the rest of this section we look at the details of how this is done; Section 5.1 summarizes the shrinking algorithm, which is followed by handling "tilted" facets in Section 5.2.

5.1. Face shrinking by offset lines

The basic idea is very simple: a face is shrunk by parallel offsets of its edges. The offset distance is computed on a per-edge basis from the *maximal offset distance*, which is intuitively the largest offset we can safely make (see exact definition below). The ratio of the offset distance to the maximum is the fullness shape parameter already illustrated in Figure 3 of the previous section.



(a) Maximal offsets (b) Offsets with fullness $= 0.6$

Figure 4: Shrinking a complex face.

The maximal offset distance is defined as 'half of the distance of the closest vertex in the offset direction, *when the polygon is limited by the wedge area enclosed by the adjacent edges*'. The rationale behind this is to ensure that the shrunk face has no self-intersections, even in highly concave cases. We explain this in detail, referring to Figure 4a—a somewhat artificial example showcasing different configurations.

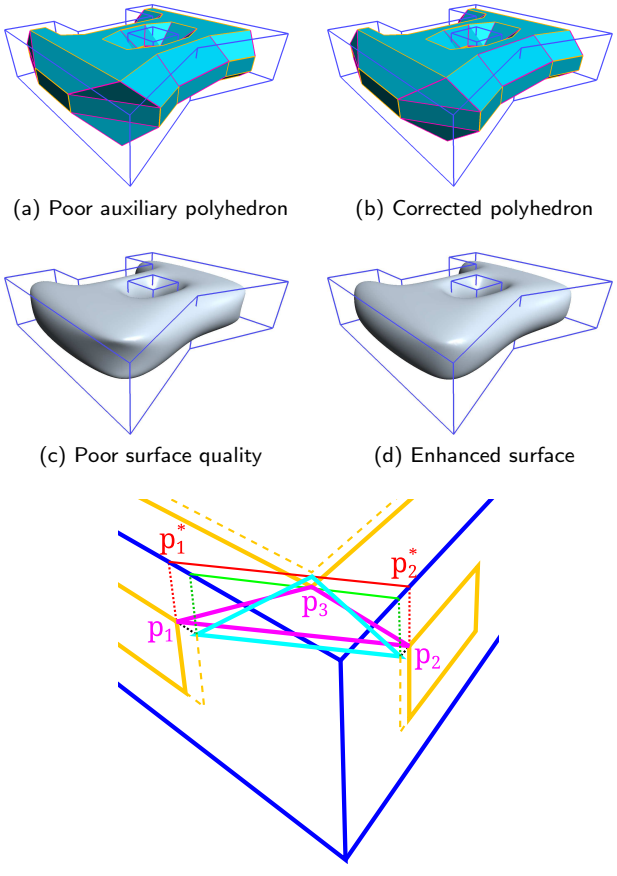
The face consists of a 9-sided polygon with a rectangular hole inside. Note the concave corner at the bottom, and two other ones at the top-left and top-right. First let us take the left side of the polygon, shown in magenta. If we "push" this line to the right, we can go only until the second, dotted line, where it meets the top-left concave vertex. We set the maximal offset (shown by a dashed line) to half of this distance. Since no offset goes beyond the halving line, non-adjacent offsets will never intersect.

Similarly, on the top-right, the cyan line can be pushed to the right only until it meets the top-right corner, thus defining the maximal offset line halfway between the two. Note that in this case we only consider a small part of the polygon, shaded in cyan, the region defined by (the extension of) the adjacent sides.

At the bottom-right the same principle defines the wedge region shaded in red; the polygon limited to this region introduces a new vertex shown by a red dot – the red line cannot be pushed any farther than that, so the maximal offset is the middle, dashed line.

Finally, the bottom line of the central hole can be pushed downwards as far as the concave vertex below. Here, again, only the green region is considered, so vertices outside that would not change the maximal offset distance.

Figure 4b shows all maximal offsets with black, dashed lines. These define a simplified medial axis-like structure, where a concave vertex of the original face is associated with an extra segment (instead of a parabolic arc), whose endpoints are computed by intersecting an offset line with the elongation of the adjacent edge, see the red dotted lines in the top-right corner. A shrunk polygon corresponding to fullness $= 0.6$ is shown in yellow; segments associated with concave vertices are drawn with thick lines.



(a) Poor auxiliary polyhedron (b) Corrected polyhedron

(c) Poor surface quality (d) Enhanced surface

(e) Changing the offsets

Figure 5: Handling tilted facets.

5.2. Facet corrections

The vertex facets of the auxiliary polyhedron are not necessarily planar. In this case, we replace them by projecting their vertices to the best-fit plane.

Triangular facets may not always represent a proper corner cut, when the normal of the face tilts away from the associated corner instead of pointing towards it, see Figure 5a. This should be prevented as such a configuration would lead to incorrect cross-derivatives, and, eventually, to poor surface quality, see Figure 5c.

This problem can be detected in the following way. Take two vertices of the facet, $\mathbf{p}_1$ and $\mathbf{p}_2$, and project them to the face containing the third one ($\mathbf{p}_3$). The projected points are denoted by $\mathbf{p}_1^*$ and $\mathbf{p}_2^*$, see Figure 5e. Then the red line connecting $\mathbf{p}_1^*$ and $\mathbf{p}_2^*$ should separate $\mathbf{p}_3$ from the associated corner of the control polyhedron. When this is not the case, we perform some heuristic modifications (the modified offsets are shown with dashed lines):

- *Increase both offsets on face 3.* Project $\mathbf{p}_3$ to the red line, thereby implicitly changing the two associated offsets. This could, in theory, enlarge the offsets excessively, so we clamp these never to go beyond 90% of the maximal offset distance.
- *Decrease the longitudinal offsets on faces 1 and 2.* Translate the red line to contain $\mathbf{p}_3$ (result shown in green). Apply the same translation to the $\overline{\mathbf{p}_1\mathbf{p}_2}$ line and intersect it with faces 1 and 2 to determine the new offsets.

The resulting triangle (shown in cyan) has good orientation; the corrected auxiliary polyhedron and the enhanced surface are shown in Figures 5b and 5d.

These changes on the offsets are performed independently for each vertex. The modifications are then aggregated in the following way: (i) when an offset was only enlarged, the maximum is used; (ii) when it was only decreased, the minimum is used; (iii) otherwise it is left unchanged.

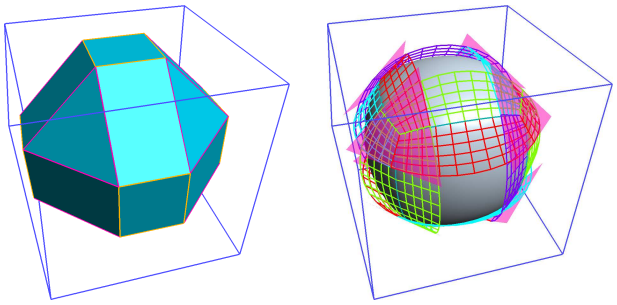
6. Boundaries and ribbons

Having defined the auxiliary polyhedron in the previous section, the next step is to generate a curve network and the associated cross derivatives in the form of ribbons. First we generate Bézier ribbons for each boundary segment (Sec. 6.1), then the smoothly connected Bézier ribbons will be concatenated into B-spline ribbons (Sec. 6.2).

6.1. Bézier ribbons

Figure 6 shows a cube example. The ribbons are generated in two phases: first a *preliminary Bézier ribbon* is created, which is then modified to ensure G^1 continuity between the patches.

The control points of the preliminary ribbons are placed at the vertices, centroids, and edge midpoints



(a) Auxiliary polyhedron (b) Facets and preliminary ribbons

Figure 6: Cube example.

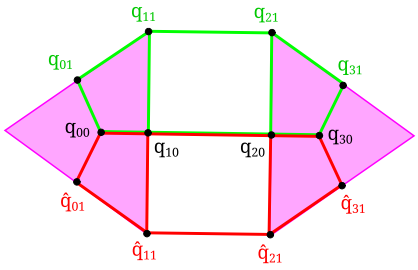


Figure 7: Preliminary ribbon generation.

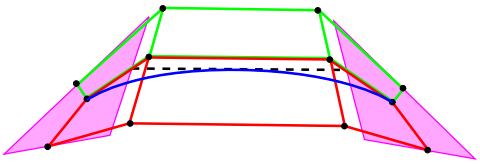


Figure 8: Scaling the tangents of preliminary ribbons.

of the facets created by the chamfering process, see Figure 7. These 4×2 grids define cubic-by-linear Bézier ribbons.

Experience shows that the resulting curve network tends to be a little flat. We modify the above placements by scaling the tangent vectors by $\frac{4}{3}$, see Figure 8, but we retain the *cross-derivative control vectors* $\mathbf{q}_{i1} - \mathbf{q}_{i0}$ of Figure 7. Note, that this modification has the effect that the midpoints of the boundary curves lie on the auxiliary polyhedron (see the dotted line in Fig. 8), so the fullness of these curves is in a sense maximal.

This procedure generalizes to non-three-sided facets, as well, but in this case no scaling is needed, see Figure 9.

There are various ways to ensure G^1-continuity between the opposite preliminary Bézier ribbons, for example using a direction blend, such as the Chiyokura method [5]. Here we apply a simpler technique, where we choose all cross-derivative control vectors to be symmetrical. It is easy to see that a modification is only needed at the outermost pair of vectors $\mathbf{q}_{01}$ and

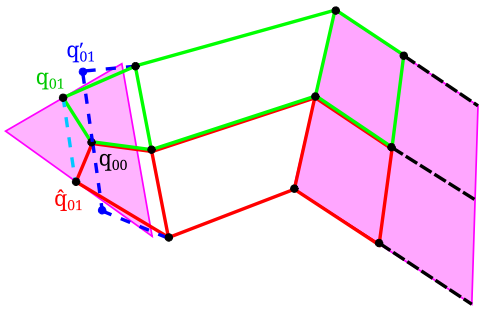


Figure 9: Ribbon construction in the general case; modifications for G^1 continuity are shown with blue lines.

$\mathbf{q}_{31}$; we define

$$\mathbf{q}'_{01} := \mathbf{q}_{00} + \sigma_k \cdot (\mathbf{q}_{01} - \hat{\mathbf{q}}_{01}), \tag{1}$$

where σ_k is a scaling factor depending only on the valence k of the corner. (The definition of $\mathbf{q}'_{31}$ is similar.) The simple mean ($\sigma_k = \frac{1}{2}$) works well in most cases, but we found that patch quality is generally higher if we choose a larger value for triangular facets. In all our examples $\sigma_3 = \frac{4}{5}$ is used, see Figure 9.

In this way, we have produced *modified Bézier ribbons* that will ensure that the adjacent patches join with G^1 continuity. These ribbons meet at a common tangent plane – defined by the facet – at the corners, but they are generally *not* twist-compatible. The incompatibility is resolved using rational correction terms in the surface equation, similar to Gregory's twists [10] (see Sec. 7, Eq. 4).

6.2. B-spline ribbons

At the concave corners the related Bézier ribbons connect smoothly with C^1 continuity (see Fig. 9). In these cases we can concatenate the successive Bézier ribbons into C^1-continuous B-splines, using a knot vector with equidistant internal knots and multiplicity 2. The control points of the B-spline ribbons are created by omitting the common control points of the Bézier ribbons, as shown in Figure 10.

7. Patchwork

The patchwork of our model is a collection of GBS patches. For the sake of completeness, we briefly revisit the patch equation; details can be found in the original paper [43]. As discussed in the previous section, here we combine longitudinally cubic B-spline ribbons, maintaining G^1 continuity in the cross-direction. The perimeter loop of the patch is composed of n open B-spline curves (clamped B-splines), and there are n_L periodic B-spline loops in the interior. The surface is defined over a curved domain in (u, v) obtained by first flattening the boundary curves of the perimeter loop. In a second step, hole loops are mapped onto the domain ensuring an optimal placement; details can be found in Appendix A.

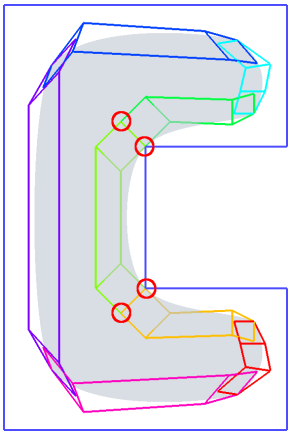


Figure 10: Concatenating successive Bézier ribbons into a C^1 B-spline surface. Control points marked with a red circle are removed.

The ribbons $\mathbf{R}_i$ are represented as biparametric surfaces with local coordinates $(s_i, h_i) \in [0,1]^2$, see Figure 11. The side parameter s_i is constant 0 on the $i-1$st side of the domain, increases according to the knot vector Ξ_i from 0 to 1 on the ith side, and it is constant 1 on the $i+1$st side. The distance parameter h_i is constant 0 on the ith side, increases from 0 to 1 on the adjacent sides, and it is constant 1 on all other sides. For periodic curves h_i is constant 0 on the ith side and constant 1 on all other sides. Harmonic interpolation is used to map the global patch coordinates (u, v) to the local side parameters (s_i, h_i), see details in [43].

The ribbons are ruled surfaces represented by a grid of control points:

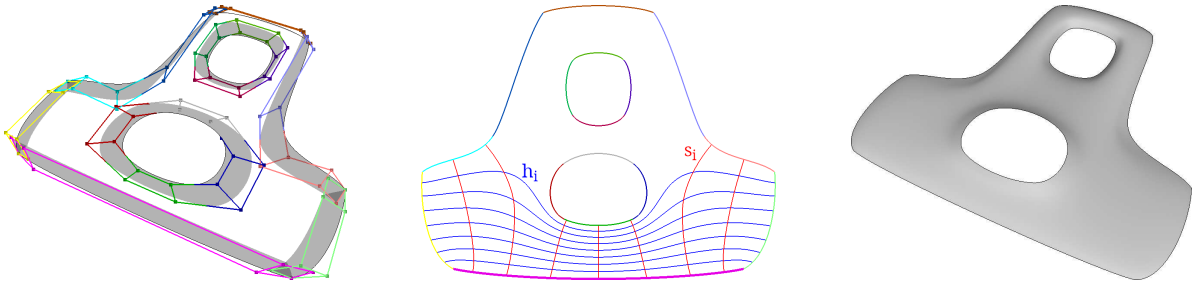
$$\mathbf{R}_i(s_i, h_i) = \sum_{j=0}^{d_i} \sum_{k=0}^{1} \mathbf{C}_{ijk} \cdot N_j^{3,\Xi_i}(s_i) B_k^1(h_i), \tag{2}$$

where $\mathbf{C}_{ijk}$ is a control point in the kth row and jth column of ribbon i; $N_j^{3,\Xi_i}(s_i)$ is the jth cubic B-spline basis function with knot vector Ξ_i; $d_i + 1$ is the number of control points longitudinally, and $B_k^1(h_i)$ denote the linear Bernstein polynomials. The GBS patch reproduces the ribbon constraints and is constructed via the same control points as the ribbons, but with different blends:

$$\hat{\mathbf{S}}(u, v) = \sum_{i=1}^{n+n_L} \sum_{j=0}^{d_i} \sum_{k=0}^{3} \mathbf{C}_{ijk} \cdot \mu_j^i(h_i) \cdot N_j^{3,\Xi_i}(s_i) B_k^3(h_i), \tag{3}$$

where the rational scalar function μ is defined as

$$\mu_j^i(h_i) = \begin{cases} h_{i-1}^2/(h_{i-1}^2 + h_i^2), & i \le n, j \in \{0, 1\}, \\ h_{i+1}^2/(h_{i+1}^2 + h_i^2), & i \le n, j \in \{d_i - 1, d_i\}, \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$



(a) Input ribbons (b) Local parameters of the bottom side (c) Evaluated surface

Figure 11: Generalized B-spline surface evaluation.

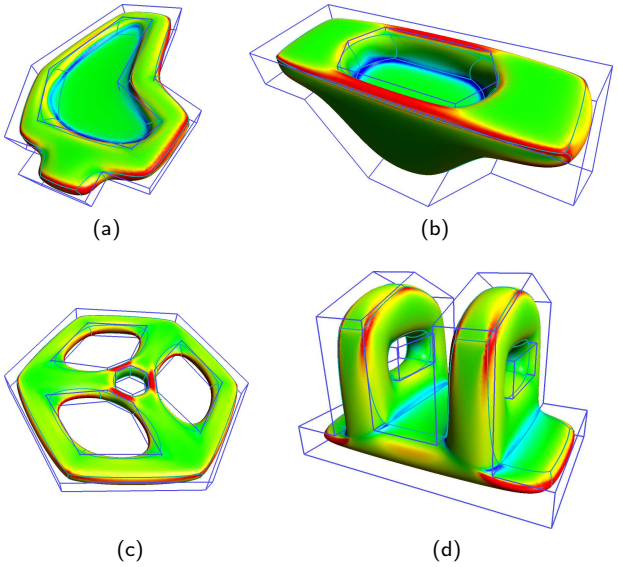
Here cubic Bernstein functions $B_k^3(h_i)$ are used for ensuring the G^1 interpolation property; these guarantee that on all "distant" boundaries, where h_i is equal to 1, the contribution of the ith ribbon will vanish. The rational terms $\mu_j^i(h_i)$ are needed only for open ribbons, these eliminate the effect of ribbon i on the immediately adjacent sides $i-1$ and $i+1$. Finally, we need to normalize the sum of the ribbons by $\Gamma(u,v)$, the sum of the blending functions, in order to provide affine invariance, so the GBS patch equation is

$$\mathbf{S}(u,v) = \frac{\hat{\mathbf{S}}(u,v)}{\Gamma(u,v)}. \tag{5}$$

8. Discussion

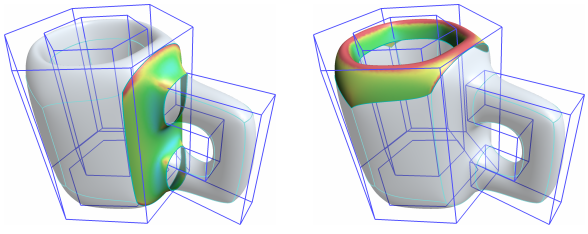
Figure 12 illustrates a few interesting test models where the patchwork seamlessly adapts to the geometry of the control polyhedron. These models represent relatively simple engineering objects with not too many faces; our method is not primarily intended for modeling highly complex organic or artistic shapes.

Experiments indicate that our surface models are dominated by relatively flat patches; these exhibit highly curved parts along the boundaries and at the corners, and relatively low curvature in their interior. Nevertheless, there may be several patches that possess high interior curvature. This is illustrated by a test model shown in Figure 13, where 21 out of the 23 surfaces are relatively flat and only two show high curvature.



(a) (b) (c) (d)

Figure 12: Various example models with mean curvature.



(a) Side surface (b) Top surface

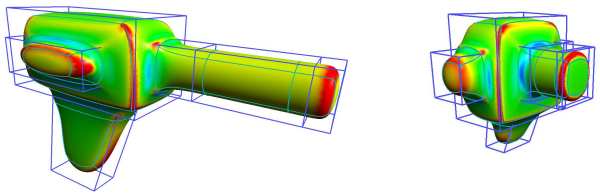
Figure 13: Highly curved surfaces of the 'mug' model.

8.1. Editing the control polyhedron

The input for our method is a control polyhedron, which can be constructed and manipulated using standard polyhedral design techniques. This process allows for the combination of parametric solid primitives via Boolean operations, as well as the application of local operations to modify specific faces or extend the existing geometry. Such local operations include extrusion, tweaking, tilting, and drafting. For example, Figure 14a illustrates an object composed of parametric solid blocks; by simply adjusting the dimensions of these primitives, the user can generate a distinct geometry while maintaining the same underlying topology, as shown in Figure 14b.

Figure 15 demonstrates a sequence of local operations:

(a) A polygonal contour is drawn on the top face of a simple block.
(b) The contour is selected and extruded.
(c) An additional edge is inserted to subdivide the top-right face.



(a) Before (b) After

Figure 14: Editing a model

Fullness	μ	σ
0.60	0.978344	0.0306607
0.65	0.946315	0.0187254
0.70	0.914290	0.0064731
0.75	0.882015	0.0064879
0.80	0.849639	0.0192531

Table 1
Sphere approximation by a cube of edge length 2, centered at $(0, 0, 0)$. The table summarizes, for various fullness values, the mean and standard deviation of sample point distances from the origin.

(d) This new edge is shifted, resulting in a five-sided top face.
(e) The middle concave face is 'tweaked' (tilted) to modify its geometry.
(f) The final object is generated through a Boolean subtraction.

8.2. Fullness

As described earlier, our approach produces a series of surface models based on a fullness parameter which determines the width of the chamfers, and thereby the full auxiliary polyhedron. In principle, it can take any value in the $(0, 1)$ interval depending on the user's choice, however we recommend values between 0.2 and 0.8 to prevent extreme blending artifacts or oversmoothing.

The default fullness value is set to 0.7. This specific value was determined by optimizing the approximation of a sphere using a cubical control polyhedron, see Figure 16. Taking a cube between corners $(-1, -1, -1)$ and $(1, 1, 1)$ this setting yielded a mean radius of 0.91 with a standard deviation of $6 \cdot 10^{-3}$. Results for other fullness values are summarized in Table 1.

8.3. Alternative chamfering methods

In Section 5 we presented a global method for producing an auxiliary polyhedron, though clearly many possible alternatives exist for setting the offset lines that bound the chamfers. For instance, the user may want to set a uniform offset distance for all edges (Fig. 17). Another approach is to enforce symmetric chamfering, where the offset distances would be identical on the opposite sides of each edge, for example, by always taking the smaller offset distance. An example is shown in Figure 18. If we use our default chamfering, then lifting the right protrusion of the model will drag the middle closed loop upwards (Fig. 18a). If we apply the above-mentioned symmetric chamfering, the closed loop on the right side will retain its height (Fig. 18b).

In general, a dedicated graphical user interface would make it possible to locally adjust the individual offset distances where the default offsetting is not satisfactory.

8.4. Comparison with recursive subdivision

It is instructive to compare our method with recursive subdivision. We need to insert artificial subdivision edges to obtain corresponding objects that have only convex faces. Different construction rules yield different objects. Figure 19 shows Doo–Sabin, Catmull–Clark and GBS versions of the same models. A very subjective qualitative assessment is given here, as follows.

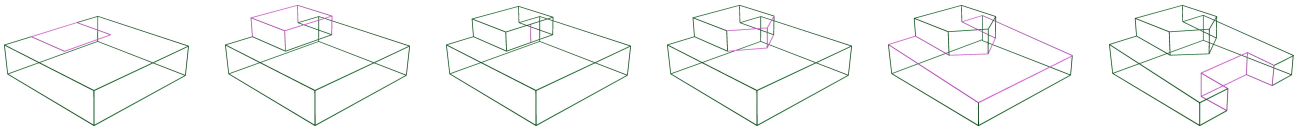
- Doo–Sabin: this scheme produces G^1 objects. The boundaries of the surfaces interpolate the centroids of the input faces at the limit, and these are often visible on the curvature maps.
- Catmull–Clark: undoubtedly this method produces the aesthetically most pleasing models, having G^2 continuity for the whole patchwork (excluding extraordinary vertices). These models shrink inwards, and the smoothing effect is strong.
- Our GBS models lie somewhere in-between, maintaining a stronger correspondence with the control polyhedron. Moreover, this correspondence can be further controlled by the fullness parameter. While our models currently ensure only G^1 continuity, visual analysis indicates reasonably good transitions of numerical curvature NG^2. For instance, in Figure 20 one can observe isophote strips smoothly crossing the patch boundaries.

It is also important to emphasize that different constructions work well with different kinds of cages; one cannot expect that models created for subdivision should automatically generate good-quality objects with this method, or *vice versa*.

8.5. Alternative surface representations

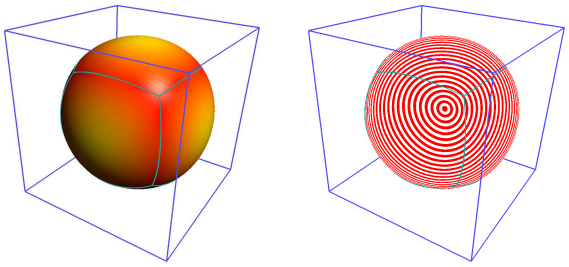
The key element of our patch representation is the use of a curved domain with side-based parameterizations. This makes it possible to explore alternative ribbon-based surface formulations, as well [45]. Specifically, we propose the use of biharmonic patches defined in [14, 42].

Biharmonic patches minimize a discrete Laplacian or thin-plate energy functional over the curved domain, with G^1 boundary conditions prescribed by the ribbons:



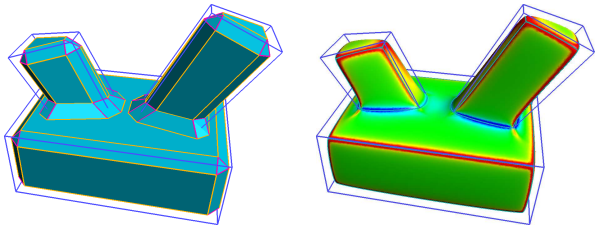
(a) Add contour (b) Extrude (c) Subdivide edge (d) Shift edge (e) Tweak face (f) Subtract

Figure 15: Editing with local (a--e) and Boolean (f) operations.



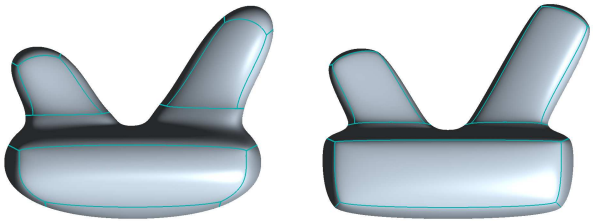
(a) Gaussian curvature (b) Isophote lines

Figure 16: Cube model approximating a sphere.



(a) Auxiliary polyhedron (b) Mean curvature

Figure 17: Model generation with a fix offset.



(a) Default (b) Symmetric

Figure 18: Hole loops in the case of default vs. symmetric chamfering.

$$\begin{aligned} \underset{\mathbf{S}}{\text{minimize}} \quad & \int_\Omega \Delta\mathbf{S}(u,v)^2 du\, dv \\ \text{subject to} \quad & \mathbf{S}(u,v) = \mathbf{P}(u,v), \ (u,v) \in \partial\Omega, \\ & \frac{\partial \mathbf{S}(u,v)}{\partial \mathbf{n}} = \mathbf{T}(u,v), \ (u,v) \in \partial\Omega, \end{aligned} \tag{6}$$

where $\mathbf{P}(u,v)$ and $\mathbf{T}(u,v)$ represent the positional and cross-derivative constraints, respectively.

We incorporate the influence of the side-based (s_i, h_i) parameterization by imposing normal derivative constraints as

$$\begin{aligned} \frac{\partial \mathbf{S}(u,v)}{\partial \mathbf{n}}\bigg|_{\text{side } i} &= \frac{\partial \mathbf{R}_i(s_i(u,v), h_i(u,v))}{\partial \mathbf{n}} \\ &= \frac{\partial \mathbf{R}_i}{\partial s_i}\frac{\partial s_i}{\partial \mathbf{n}} + \frac{\partial \mathbf{R}_i}{\partial h_i}\frac{\partial h_i}{\partial \mathbf{n}}. \end{aligned} \tag{7}$$

When compared with GBS patches, we have found that these patches are somewhat smoother, though in many cases the difference is hardly recognizable, see Figure 21.

We remark that Kato's surface [20], once generalized to curved domains, may serve as a possible surface representation that – at least in principle – could be integrated into our general polyhedral framework.

8.6. Computational cost

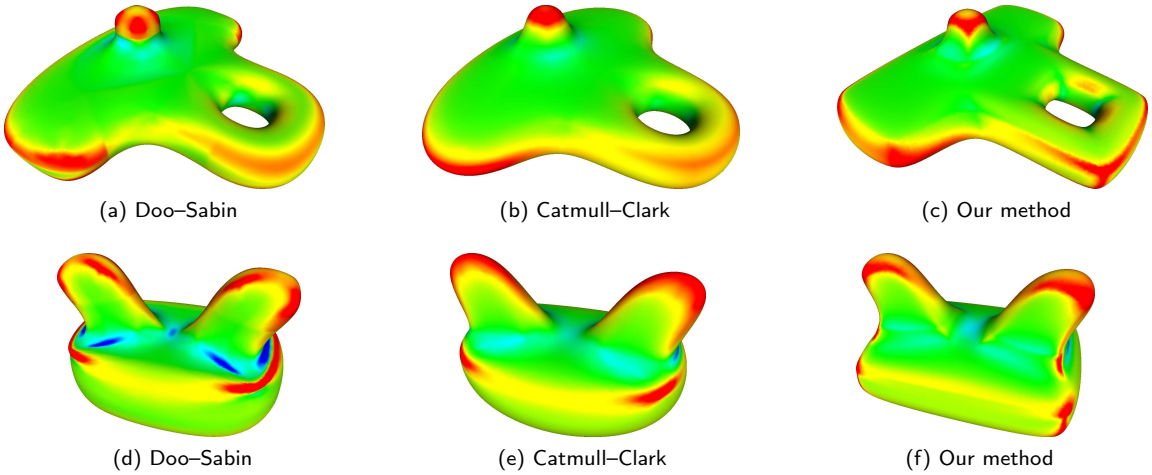
The model is tessellated for visualization by evaluating each patch independently and subsequently stitching the resulting meshes together. To ensure consistency along patch boundaries, we employ a fixed sampling of the boundary curves that is uniform with respect to arc length. The resolution parameter R determines the sampling density by setting the distance between successive sample points to $1/R$ times the diagonal of the bounding box. The domain triangulation is performed using Shewchuk's Triangle library [39].

Table 2 reports the running times for evaluating the models shown in Figure 12. All measurements were obtained from a single-threaded implementation running on an Intel Core i5 processor at 1.60 GHz. The default resolution is set to $R = 100$, but to provide real-time feedback during interactive editing, lower-resolution models ($R = 50$) are displayed while modifications are in progress.

As it can be seen from the data, even this modest configuration allows for (near-)real-time editing. Stitching time is negligible, and computing the parameterization accounts for over 80% of the evaluation time.

Conclusion and future work

We have presented a direct method for generating surface models from a control polyhedron. The resulting models are composed of smoothly connected free-form patches, capable of handling concave corners and hole loops. The topology of the control polyhedron is



(a) Doo–Sabin (b) Catmull–Clark (c) Our method

(d) Doo–Sabin (e) Catmull–Clark (f) Our method

Figure 19: Comparison with recursive subdivision, showing mean curvature.

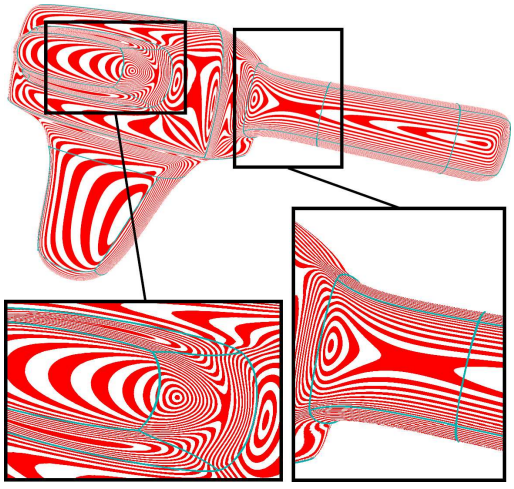


Figure 20: Isophotes on the model shown in Figure 14.

Model	Patches	Vertices	Evaluation	Stitching
(a)	21	12022	1162ms	44ms
		2911	270ms	1ms
(b)	17	18779	1049ms	65ms
		4519	254ms	7ms
(c)	29	10757	2228ms	51ms
		2560	568ms	2ms
(d)	24	24778	1525ms	112ms
		5899	377ms	21ms

Table 2
Running times for evaluating the models in Figure 12. For each model there are two versions, one with default resolution ($R = 100$), and one with low resolution ($R = 50$).

strictly preserved. An important feature that distinguishes our method from traditional subdivision-based approaches is that, depending on a fullness parameter, a family of surfaces can be computed, controlling a balance between blending and smoothing. The method

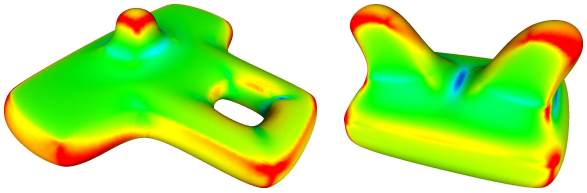


Figure 21: The models in Figure 19 with biharmonic patches.

uses an auxiliary polyhedron that is created through a chamfering procedure. This determines a general topology free-form curve network that is interpolated – along with associated ribbons – by Generalized B-spline patches.

Several avenues remain open for future work. Alternative constructions for the fundamental chamfering step could be explored, including open control meshes, chamfering with offsets that are not parallel to a given edge, and auxiliary constructions for non-planar polygonal faces. The curve network could be generalized using quartic or quintic boundary segments, potentially providing a more even curvature distribution. Furthermore, instead of G^1 ribbons, one may be able to deduce G^2 ribbons from the control polyhedron to achieve higher order smoothness. Alternative patch representations and optimized domain mapping are subjects of ongoing research.

Acknowledgments

This project has been supported by the Hungarian Scientific Research Fund (OTKA, No. 145970).

References

[1] Antonelli, M., Beccari, C.V., Casciola, G., Ciarloni, R., Morigi, S., 2013. Subdivision surfaces integrated in a CAD system. Computer-Aided Design 45, 1294–1305. doi:10.1016/j.cad.2013.06.007.

[2] Biermann, H., Levin, A., Zorin, D., 2000. Piecewise smooth subdivision surfaces with normal control, in: SIGGRAPH '00, ACM. pp. 113–120. doi:10.1145/344779.344856.

[3] Cashman, T.J., 2012. Beyond Catmull–Clark? A survey of advances in subdivision surface methods. Computer Graphics Forum 31, 42–61. doi:10.1111/j.1467-8659.2011.02083.x.

[4] Catmull, E., Clark, J., 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design 10, 350–355. doi:10.1016/0010-4485(78)90110-0.

[5] Chiyokura, H., Kimura, F., 1983. Design of solids with free-form surfaces. ACM SIGGRAPH Computer Graphics 17, 289–298. doi:10.1145/964967.801160.

[6] DeRose, T.D., Kass, M., Truong, T., 1998. Subdivision surfaces in character animation, in: SIGGRAPH '98, ACM. pp. 85–94. doi:10.1145/280814.280819.

[7] Djuren, T., Finnendahl, U., Kohlbrenner, M., Worchel, M., Alexa, M., 2025. Interpolating splines over triangulated surfaces by blending vertex-centric local geometries. Computers & Graphics , 104316doi:10.1016/j.cag.2025.104316.

[8] Doo, D., Sabin, M., 1978. Behaviour of recursive division surfaces near extraordinary points. Computer-Aided Design 10, 356–360. doi:10.1145/280811.280991.

[9] Feng, Y.F., Shen, L.Y., Li, X., Yuan, C.M., Jiang, X., 2023. Patching non-uniform extraordinary points. IEEE Transactions on Visualization and Computer Graphics 30, 4683–4693. doi:10.1109/TVCG.2023.3242131.

[10] Gregory, J.A., 1974. Smooth interpolation without twist constraints, in: Computer Aided Geometric Design, University of Utah. pp. 71–87. doi:10.1016/B978-0-12-079050-0.50009-6.

[11] Grimm, C.M., Zorin, D., 2006. Surface modeling and parameterization with manifolds, in: SIGGRAPH '06, ACM. pp. 1–81. doi:10.1145/1185657.1185728.

[12] Gunpinar, E., Karčiauskas, K., Peters, J., 2024. Splines for fast-contracting polyhedral control nets. Computer-Aided Design 173, 103727. doi:10.1016/j.cad.2024.103727.

[13] Hettinga, G.J., Kosinka, J., 2020. A multisided C^2 B-spline patch over extraordinary vertices in quadrilateral meshes. Computer-Aided Design 127, 102855. doi:10.1016/j.cad.2020.102855.

[14] Jacobson, A., Tosun, E., Sorkine, O., Zorin, D., 2010. Mixed finite elements for variational surface modeling. Computer Graphics Forum 29, 1565–1574. doi:10.1111/j.1467-8659.2010.01765.x.

[15] Karčiauskas, K., Panozzo, D., Peters, J., 2017. T-junctions in spline surfaces. ACM Transactions on Graphics (TOG) 36, 170. doi:10.1145/3136954.

[16] Karčiauskas, K., Peters, J., 2015. Improved shape for multi-surface blends. Graphical Models 82, 87–98. doi:10.1016/j.gmod.2015.06.006.

[17] Karčiauskas, K., Peters, J., 2020. Low degree splines for locally quad-dominant meshes. Computer Aided Geometric Design 83, 101934. doi:10.1016/j.cagd.2020.101934.

[18] Karčiauskas, K., Peters, J., 2021. Multi-sided completion of C^2 bi-3 and C^1 bi-2 splines: a unifying approach. Computer Aided Geometric Design 86, 101978. doi:10.1016/j.cagd.2021.101978.

[19] Karčiauskas, K., Peters, J., 2022. Localized remeshing for polyhedral splines. Computers & Graphics 106, 58–65. doi:10.1016/j.cag.2022.05.019.

[20] Kato, K., 2000. N-sided surface generation from arbitrary boundary edges, in: Curve and Surface Design, AFA. pp. 173–182.

[21] Levin, A., 2006. Modified subdivision surfaces with continuous curvature. ACM Transactions on Graphics (TOG) 25, 1035–1040. doi:10.1145/1179352.1141990.

[22] Lodha, S., 1993. Filling n-sided holes, in: Modeling in Computer Graphics, IAC-CNR. pp. 319–345. doi:10.1007/978-3-642-78114-8_20.

[23] Loop, C., 1994. Smooth spline surfaces over irregular meshes, in: SIGGRAPH '94, ACM. pp. 303–310. doi:10.1145/192161.192238.

[24] Loop, C.T., DeRose, T.D., 1989. A multisided generalization of Bézier surfaces. ACM Transactions on Graphics (TOG) 8, 204–234. doi:10.1145/77055.77059.

[25] Loop, C.T., DeRose, T.D., 1990. Generalized B-spline surfaces of arbitrary topology, in: SIGGRAPH '90, ACM. pp. 347–356. doi:10.1145/97879.97917.

[26] Loop, C.T., Schaefer, S., Ni, T., Castaño, I., 2009. Approximating subdivision surfaces with Gregory patches for hardware tessellation. ACM Transactions on Graphics (TOG) 28. doi:10.1145/1618452.1618497.

[27] Moulaeifard, M., Wellmann, F., Bernard, S., de la Varga, M., Bommes, D., 2023. Subdivide and conquer: adapting non-manifold subdivision surfaces to surface-based representation and reconstruction of complex geological structures. Mathematical Geosciences 55, 81–111. doi:10.1007/s11004-022-10017-x.

[28] Peters, J., 1994. Constructing C^1 surfaces of arbitrary topology using biquadratic and bicubic splines, in: Designing Fair Curves and Surfaces, pp. 277–293. doi:10.1137/1.9781611971521.ch11.

[29] Peters, J., 1995. C^1-surface splines. SIAM Journal on Numerical Analysis 32, 645–666. doi:10.1137/0732029.

[30] Peters, J., 2000. Patching Catmull-Clark meshes, in: SIGGRAPH '00, ACM. pp. 255–258. doi:10.1145/344779.344908.

[31] Peters, J., 2019. Splines for meshes with irregularities. The SMAI Journal of computational mathematics 5, 161–183. doi:10.5802/smai-jcm.57.

[32] Peters, J., Reif, U., 2008. Subdivision Surfaces. volume 3 of *Geometry and Computing*. Springer. doi:10.1007/978-3-540-76406-9.

[33] Prautzsch, H., 1997. Freeform splines. Computer Aided Geometric Design 14, 201–206. doi:10.1016/S0167-8396(96)00029-5.

[34] Reif, U., 1995. Biquadratic G-spline surfaces. Computer Aided Geometric Design 12, 193–205. doi:10.1016/0167-8396(94)00009-H.

[35] Rockwood, A., Gao, K., Farrell, M., 2016. SuperD: SubD modeling without subdivision, in: SIGGRAPH '16, ACM. p. 8. doi:10.1145/2897839.2927431.

[36] Salvi, P., 2024. Polyhedral design with blended n-sided interpolants, in: Proceedings of the Eleventh Hungarian Conference on Computer Graphics and Geometry, NJSZT. pp. 46–51. arXiv:2601.19322.

[37] Salvi, P., Várady, T., 2018. Multi-sided Bézier surfaces over concave polygonal domains. Computers & Graphics 74, 56–65. doi:10.1016/j.cag.2018.05.006.

[38] Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A., 2003. T-splines and T-NURCCs, in: SIGGRAPH '03, ACM. pp. 477–484. doi:10.1145/1201775.882364.

[39] Shewchuk, J.R., 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, in: Lin, M.C., Manocha, D. (Eds.), Applied Computational Geometry: Towards Geometric Engineering. volume 1148 of *Lecture Notes in Computer Science*, pp. 203–222. https://www.cs.cmu.edu/ quake/triangle.html.

[40] Smith, J., Schaefer, S., 2015. Selective degree elevation for multi-sided Bézier patches, in: Computer Graphics Forum, Eurographics Association. pp. 609–615. doi:10.1111/cgf.12588.

[41] Szörfi, J., Várady, T., 2022. Polyhedral design with concave and multi-connected faces, in: Proceedings of the Tenth

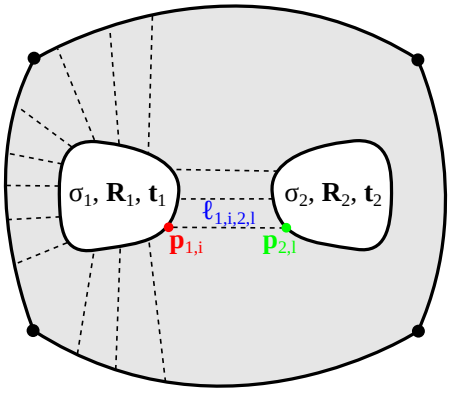


Figure 22: Illustration of constraints on hole placement.

Hungarian Conference on Computer Graphics and Geometry, NJSZT. pp. 28–35.

[42] Vaitkus, M., 2023. A note on ribbon-based biharmonic surface patches. arXiv:2311.12822.

[43] Vaitkus, M., Várady, T., Salvi, P., Sipos, Á., 2021. Multi-sided B-spline surfaces over curved, multi-connected domains. Computer Aided Geometric Design 89, 102019. doi:10.1016/j.cagd.2021.102019.

[44] Várady, T., Rockwood, A., Salvi, P., 2011. Transfinite surface interpolation over irregular n-sided domains. Computer-Aided Design 43, 1330–1340. doi:10.1016/j.cad.2011.08.028.

[45] Várady, T., Salvi, P., Vaitkus, M., 2024. Genuine multi-sided parametric surface patches – a survey. Computer Aided Geometric Design 110, 102286. doi:10.1016/j.cagd.2024.102286.

[46] Várady, T., Salvi, P., Vaitkus, M., Sipos, Á., 2020. Multi-sided Bézier surfaces over curved, multi-connected domains. Computer Aided Geometric Design 78, 101828. doi:10.1016/j.cagd.2020.101828.

[47] Wang, X., Cheng, F., Barsky, B.A., 1999. Blending, smoothing and interpolation of irregular meshes using N-sided Várady patches, in: SMA '99, ACM. pp. 212–222. doi:10.1145/304012.304034.

[48] Yang, X., 2023. Curve and surface construction with moving B-splines arXiv:2307.14677.

[49] Ying, L., Zorin, D., 2004. A simple manifold-based construction of surfaces of arbitrary smoothness. ACM Transactions on Graphics (TOG) 23, 271–275. doi:10.1145/1015706.1015714.

[50] Zheng, J., Zhang, J., Zhou, H., Shen, L., 2005a. Smooth spline surface generation over meshes of irregular topology. The Visual Computer 21, 858–864. doi:10.1007/s00371-005-0345-8.

[51] Zheng, J.J., Zhang, J.J., Zhou, H.J., Shen, L., 2005b. C^2 continuous spline surfaces over Catmull-Clark meshes, in: International Conference on Computational Science and Its Applications, Springer. pp. 1003–1012. doi:10.1007/11424857_108.

A. Mapping hole loops

The input to our method is a set of three-dimensional B-spline ribbons evaluated into boundary polylines denoted by $\mathcal{C}_i^{3D} = \{\mathbf{p}_{i,j}^{3D} \in \mathbb{R}^3\}$, and cross-derivative vectors $\mathcal{D}_i = \{\mathbf{d}_{i,j} \in \mathbb{R}^3\}$. First, for each point $\mathbf{p}_{i,j}^{3D} \in \mathcal{C}_i^{3D}$ of each inner loop, we look for the closest point $\mathbf{p}_{k,l}^{3D} \in \mathcal{C}_k^{3D}$, $k \neq i$ lying on any of the other loops. Let us denote the set of closest point pairs as $\mathcal{P} = \{(i,j,k,l)\}$. Each set of two points are connected with a cubic Hermite spline, with tangent vectors equal to $\mathbf{d}_{i,j}$, $\mathbf{d}_{k,l}$. The set of arc lengths $\mathcal{L} = \{\ell_P\}$, $P \in \mathcal{P}$ of these curves will serve as distance constraints which we wish to enforce between the corresponding 2D loops as well. We next flatten the boundary curves into the plane, while aiming to preserve their length and geodesic curvature using the heuristic method described in [43]. This results in 2D polylines $\widetilde{\mathcal{C}}_i^{2D} = \{\widetilde{\mathbf{p}}_{i,j}^{2D} \in \mathbb{R}^2\}$ that individually mimic the shape of their 3D counterparts, but their placement has little to no relation to their 3D configuration. To determine the optimal placement and scaling of the 2D loops, we first fix the position and orientation of the perimeter loop, and optimize a set of 2D similarity transformations (i.e., scaling factors σ_i, rotations $\mathbf{R}_i$ and translations $\mathbf{t}_i$, determining a transformation $T_i(\mathbf{p}) = \sigma_i(\mathbf{R}_i\mathbf{p}+\mathbf{t}_I)$) for the remaining loops. We do so by minimizing the squared deviation of the 2D distances from the corresponding 3D Hermite arc lengths – see also Figure 22 for an illustration:

$$\sum_{(i,j,k,l)\in\mathcal{P}} \left(\|T_i(\widetilde{\mathbf{p}}_j^{2D}) - T_k(\widetilde{\mathbf{p}}_l^{2D})\| - \ell_P\right)^2 \to \min\,. \quad (8)$$

The resulting nonlinear least-squares optimization problem is solved for the per-loop transformation parameters using the Levenberg–Marquardt method.