Computers & Graphics (2020)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Multi-sided implicit surfacing with I-patches

Ágoston Sipos*, Tamás Várady, Péter Salvi, Márton Vaitkus

Budapest University of Technology and Economics, Hungary

ARTICLE INFO

Article history:

Implicit surfaces, Multi-sided patches, Setback vertex blends, Polyhedral design

ABSTRACT

I-patches represent a family of implicit multi-sided surfaces. Similarly to functional splines, each boundary curve of the patch is defined as the intersection of a primary and a bounding surface, both given in implicit form, and the patch can connect to the primaries with arbitrary geometric continuity. Following the publication of Várady et al. [1], this paper elaborates the basic formulation in more detail, and introduces several interesting features, including a distance-based surface interpretation, consistent orientation of the primaries, setting shape parameters, and handling various special cases.

Implicit multi-sided patches are primarily used for connecting simple implicit surfaces, such as planes, cylinders, spheres etc., however, I-patches are also capable of modeling complex free-form shapes. We show constructions for producing setback vertex blends with conic boundaries and patchworks defined by control polyhedra. We discuss the benefits and limitations of the representation through several examples.

1. Introduction

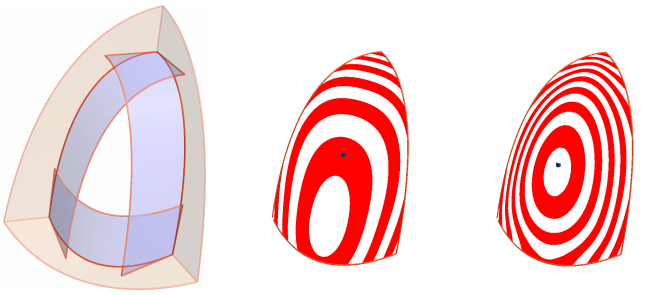
Surface modeling with multi-sided patches is dominated by representations given in parametric form [2, 3, 4]; however, there are various design situations, where the use of *implicit multi-sided patches* is preferable: (i) Creating accurate connections to surfaces given in implicit form, in particular to planes and natural quadrics, e.g. in hole filling, vertex blending and lofting. (ii) Defining a complex free-form object by a control polyhedron and obtaining a collection of smoothly connected patches, where the implicit form is beneficial since regular shapes can also be incorporated. (iii) Approximating complex surfaces defined by vector fields of distances and gradients, e.g. cell-based representations and marching surface techniques.

Parametric surface representations offer a great versatility in shape design and analysis; however, for mapping a planar domain to 3D we need some intrinsic parameterization, and this is a delicate issue. While it is easy to tessellate parametric surfaces, there are several geometric interrogations that proved to be computationally demanding, e.g. intersections and joining trimmed patches.

Among implicit surface representations, algebraic polynomials are mostly used to represent simple regular surfaces. Representing free-form shapes is not easy, as high-degree algebraic polynomials may produce various shape problems, such as singularities, self-intersections, and disconnected parts. Implicit surfaces are more rigid than parametrics, and editing the shape interior is a real challenge, as there are no obvious control structures to do so. They are generally C^∞-continuous, representing half-spaces, so point membership classification is easy. No parameterization is needed for distance computations and approximating data points. Implicit surfaces are favorable in photo-realistic rendering, due to their computational efficiency in ray tracing (see e.g. [5, 6]).

In this paper we revisit the classical arena of implicit representations, and attempt to significantly enhance the I-patch concept of Várady et al. [1], published almost two decades ago. In some sense, I-patches are similar to transfinite schemes, where

*Corresponding author:
e-mail: asipos@edu.bme.hu (Ágoston Sipos)



(a) Conic primary and planar bounding surfaces (b) Contours (c) Contours after changing the interior fullness

Figure 1: A simple 3-sided I-patch.

boundary curves and cross-derivatives – given in parametric form – are blended together in a smooth, somewhat controllable manner. I-patches interpolate a loop of boundary curves, where each segment is defined as the intersection of a *primary* and a *bounding* surface, both given in implicit form (similarly to functional splines [7]). A simple 3-sided example is shown in Figure 1. Three primaries (conic sweeps) and three bounding surfaces (planes) define the boundary constraints for the two patches in Figs. 1b and 1c; this also illustrates that there is some degree of freedom to control the fullness of the patch, i.e., the curvatures in the interior, as will be explained later).

I-patches smoothly connect to the primaries with G^1, G^2 or higher degree geometric continuity (see Section 3). The primary surfaces may be *a priori* determined, or often they are created artificially to satisfy positional and tangential constraints along given boundaries; in this case we prefer to use the term (primary) *ribbon*. Our current interest is to elaborate multi-sided implicit surface representations that play a somewhat similar role as their parametric counterparts, i.e., we wish to (i) constrain and edit the boundaries of the shape, and (ii) possibly modify its interior while avoiding awkward twists and self-intersections. At the same time, we wish to exploit the advantages of having exact, algebraic representations.

The paper is structured as follows. After reviewing prior work in Section 2, we revisit the basic theory of I-patches, supplemented by a new, distance-based interpretation (Section 3). We continue with the construction of ribbons and bounding surfaces in Section 4, and focus on I-patches with conic boundaries. This is followed by two potential application areas – vertex blending and polyhedral design (Section 5). In Section 6 we discuss some special cases and related difficulties that need to be handled, then compare I-patches to alternative formulations and show a few test examples. Suggestions for future research conclude the paper.

2. Previous work

Implicit surfaces have an extensive literature, related to a wide range of topics, including scattered data interpolation, approximation, ray tracing, etc. For a general introduction, see the classic book on the subject [8]. In this paper, we are mainly interested in *blending* and *modeling* with implicit surfaces, as well as in methods of *tessellation*. The rest of this section is organized around these keywords.

2.1. Blending

Creating a smooth blend between two or more surfaces is a basic operation in CAD systems. Implicit blending surfaces were introduced in the seminal paper of Hoffmann and Hopcroft [9], and many important problems – like unwanted bulges and discontinuities – were identified and resolved by the displacement blend [10]. The superelliptic blend used in this method still had gradient discontinuities, which can be avoided by replacing the standard set-theoretic CSG operators of Ricci [11] with F-rep [12], based on R-functions, or soft blending [13], which satisfies a Lipschitz condition. Recent developments include gradient-based operators [14] and better topology control [15].

There is also another approach to blending – the one followed by this work –, where the boundaries of the created surface are explicitly defined (in contrast to just specifying a blending range). As outlined in the Introduction, the boundary curves are determined from the intersection of a primary and a bounding surface. The generated patch interpolates these curves, while smoothly connecting to the primary surfaces. (For the definition of geometric continuity in an implicit context, see [16].)

The feasibility and minimal degree of this construction was explored in [17], which gives explicit equations, based on Bézout's theorem, for the interpolation of Hermite boundary conditions with implicit surfaces, resulting in a linear system. While a direct fit gives important insights, the eligible solutions often include highly curved and self-intersecting surfaces, which are difficult to filter out. (See also Section 6.2.2 for an example.)

Functional splines [7, 18, 19, 20], on the other hand, take a more pragmatic approach, by defining the patch as a blend between a *base* and a *transversal surface*, generalizing Liming's conic formula [21]. In our setting, these correspond to the products of the primary and bounding surfaces, respectively. The exact patch equation can be found in Section 6.2.1, along with an example and comparisons to our method. The bounded blend of Pasko et al. [22] also defines boundaries as the intersections of primaries and a single bounding surface, while ensuring that the latter contains the entire blend.

The present work is based on I-patches [1], which is defined as the weighted sum of mixed products between primary and bounding surfaces – see Section 3 for details. (Note that a similar idea had already appeared in [23].)

2.2. Modeling

Implicit surface modeling is generally done by CSG operations, extended by blending [24]. This also allows sketch-based modeling [25, 26]. However, these methods generate "blobby" objects, suitable mainly for organic models and animations.

Warren [27] proposes local fitting of implicit surfaces in a subdivision of 3D space (e.g. a simplicial mesh), in this way avoiding the problems of higher-order interpolation. The implicit surface is defined as an interpolant of values at given vertices. Continuity constraints at the vertices are computed

from a user-defined collection of planes embedded in the mesh – in other words, a control polyhedron. Depending on the interpolant used, the surface can exactly interpolate the values at the vertices or approximate them with additional smoothness.

Algebraic splines and A-patches [28, 29] are very similar, in that they are also defined in simplexes. In 3D, the generated patches are always 3- or 4-sided. It can be proved that the 3-sided implicit surfaces defined in tetrahedra have at most one intersection with a line going through the apex, so there will be no self-intersections; a comparable assertion is known for 4-sided surfaces, as well. The construction is such that C^1 or C^2 continuity between the patches can be ensured, and the remaining degrees of freedom can be used for (local and/or global) shape adjustment.

A framework for polyhedral design with implicit surfaces was published in [30], generating a C^1-continuous set of surfaces from a triangular mesh, using the interpolation method of [17].

As we will see, I-patches can be used for complex hole filling problems, such as the setback vertex blend (Section 5.1), and for modeling with arbitrary topology polyhedra (Section 5.2).

2.3. Tessellation

There is a plethora of polygonization methods for implicit surfaces; see [31] for a general survey. In our context, since the isosurface may have several branches from which we need to display only one, region growing methods, such as [32, 33], are favorable. Alternatively, we propose a simple and efficient method that generates high-quality meshes using an auxiliary parametric surface (see Section 6.4 for details).

3. I-Patches

In this section we revisit I-patches, and show a new interpretation based on distances.

3.1. Basic concept

Multi-sided I-patches are defined by a loop of 3D curves; each curve c_i is the intersection of two surfaces given in implicit form $c_i = \{P_i = 0\} \bigcap \{B_i = 0\}$, where $P_i(x, y, z) = 0$ denotes the *primary surface* to which the patch will smoothly connect, and $B_i(x, y, z) = 0$ denotes the *bounding surface* that defines the intersection. The equation of I-patches in Várady et al. [1] was given in the following *polynomial* form:

$$I(x, y, z) = \sum_{i=1}^{n} w_i P_i \prod_{j \neq i} B_j^2 + w_0 \prod_{j=1}^{n} B_j^2 = 0, \qquad (1)$$

where the w_i-s are scalar weights associated with the individual sides, and w_0 is a central weight that influences the fullness of the patch. It is easy to demonstrate this concept with a 3-sided patch, given as

$$I(x, y, z) = w_1 P_1 B_2^2 B_3^2 + w_2 P_2 B_3^2 B_1^2 + w_3 P_3 B_1^2 B_2^2 + w_0 B_1^2 B_2^2 B_3^2 = 0. \qquad (2)$$

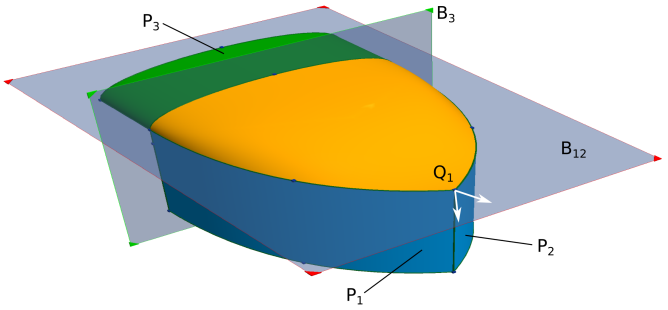


Figure 2: Two primary surfaces intersected by the same bounding plane.

This I-patch will interpolate the boundary curves. For example, take the curve c_1. The first term will be zero due to $P_1 = 0$, the other three terms due to $B_1 = 0$. This also shows that the effect of P_1 will gradually vanish as we get close to side 2 or 3, where the bounding functions B_2 and B_3 become zero.

The I-patch will smoothly connect to the primaries, since its gradient vector will be parallel to them. For example, take again boundary curve c_1, and write the equation as $I = P_1 G + B_1^2 H$. Then

$$\nabla I = \nabla P_1 G + P_1 \nabla G + 2 B_1 \nabla B_1 H + B_1^2 \nabla H = \text{const} \cdot \nabla P_1, \qquad (3)$$

since the second, third and fourth terms equal to zero on c_i. In an analogous way, G^2 or higher degree geometric continuity to the primaries can be achieved, if the exponent of the bounding surfaces is three or higher.

3.2. Remarks

It should be noted that at the corners of the I-patches, where two primaries and two bounding surfaces meet, the gradient vector is always zero. Take, for example, the corner of c_1 and c_2, then the G function in the above expression will be zero due to $B_2 = 0$, so the gradient will vanish, as well. This can be an advantage or a disadvantage. If two primaries at the corner share a common tangent plane, the normal vector of the I-patch will be uniquely determined. However, if not, a singular vertex is created, see for example point Q_1 in Figure 2.

Another special case to be investigated is when two (or more) bounding surfaces coincide. We demonstrate the problem again through a 3-sided patch. Assume that we have three boundaries $P_1 \bigcap B_{12}$, $P_2 \bigcap B_{12}$, and $P_3 \bigcap B_3$. Then we can factor out B_{12}^2 from the equation to obtain

$$I(x, y, z) = B_{12}^2 \left(w_1 P_1 B_3^2 + w_2 P_2 B_3^2 + w_3 P_3 B_{12}^2 + w_0 B_{12}^2 B_3^2 \right) = 0, \qquad (4)$$

which is a branching surface, since neither of the terms interpolates all sides. The solution to this problem is to collect all primary surfaces that belong to the same boundary, and handle them as a single surface in the form of their product. Taking our example,

$$I(x, y, z) = w_{12}(P_1 P_2) B_3^2 + w_3 P_3 B_{12}^2 + w_0 B_{12}^2 B_3^2 = 0 \qquad (5)$$

will yield the correct result. Note that we can assign only a single weight to P_1P_2. An example is shown in Figure 2. This I-patch (orange) connects to three curved primary surfaces; the two blue primaries (vertical sweeps) are intersected by the same planar bounding surface.

It is easy to constrain an I-patch to interpolate an arbitrary 3D point (x_0, y_0, z_0). Assuming that the weights w_i have already been fixed, then the value w_0 can be determined by solving the equation $I(x_0, y_0, z_0) = 0$, where w_0 is the only unknown. Figure 1 shows two variants of a 3-sided patch interpolating two different reference points in the middle.

3.3. Distance-based interpretation

Here we show that I-patches can also be interpreted based on distances. This helps to better understand the formulation and set certain shape parameters. A somewhat similar approach occurs in classical geometry, as well. Take an ellipse and its implicit equation $x^2/a^2 + y^2/b^2 - 1 = 0$, $a \geq b$. It can be interpreted as the locus of points, where the sum of the Euclidean distances from two focal points is constant, i.e.,

$$d_1(x, y) + d_2(x, y) = \|(x, y) - F_1\| + \|(x, y) - F_2\| = 2a, \quad (6)$$

where $F_1 = (-c, 0)$, $F_2 = (c, 0)$ and $c^2 = a^2 - b^2$.

We generalize this for I-patches and combine distances from the primitive surfaces. We may take $d_i = P_i(x, y, z)$ and search for the locus of points where $\sum_{i=1}^{n} d_i = d_0$; however, in order to ensure the interpolation property, we need to involve the bounding surfaces, as well, and apply *weighted algebraic distances* d_i in the form of $d_i = w_iP_i/B_i^2$. These can be derived from the polynomial I-patch equation, if we divide all terms by $\prod_{j=1}^{n} B_j^2$. For example, a 3-sided I-patch in *rational* form is the following:

$$I(x, y, z) = d_1 + d_2 + d_3 - d_0 = \frac{w_1P_1}{B_1^2} + \frac{w_2P_2}{B_2^2} + \frac{w_3P_3}{B_3^2} + w_0 = 0. \quad (7)$$

In fact, it is easy to show that distance-based I-patches can reproduce certain standard implicit surfaces. For example, combining three primaries of orthogonal elliptic cylinders and three related planar bounding surfaces, one can exactly reproduce an ellipsoid:

$$I(x, y, z) = \frac{\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1}{z^2} + \frac{\frac{y^2}{b^2} + \frac{z^2}{c^2} - 1}{x^2} + \frac{\frac{z^2}{c^2} + \frac{x^2}{a^2} - 1}{y^2} + \left(\frac{1}{a^2} + \frac{1}{b^2} + \frac{1}{c^2}\right) \quad (8)$$

$$= \frac{\left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1\right)(x^2y^2 + y^2z^2 + z^2x^2)}{x^2y^2z^2}. \quad (9)$$

(See Section 4.6 on how to set w_0.) This expression contains the equation of the ellipsoid multiplied by a second term, which is an isolated point at the origin. When $a = b = c = 1$, we obtain an octant of a sphere with unit radius. In Figure 3 we show a sequence of isosurfaces $d_1 = w_1P_1/B_1^2 = \text{const}$. The

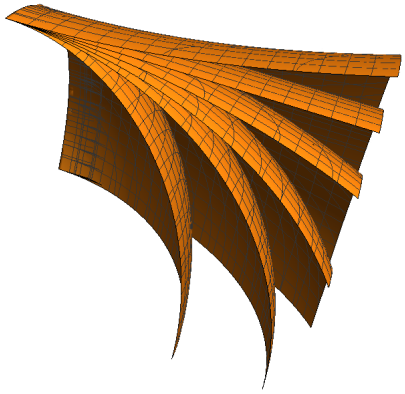


Figure 3: Different isosurfaces of a weighted algebraic distance.

combination of this sort of *algebraic distance fields* produce the final I-patch.

The distance-based formula nicely works for the interior points of the patch; however, it degenerates to $0/0$ expressions in the vicinity of boundary curve c_i. In order to avoid this, we use another formula, multiplying the equation by B_i^2/w_i. For example,

$$I(x, y, z) = P_1 + \left(\sum_{i=2}^{n} d_i + d_0\right) B_1^2/w_1 = 0. \quad (10)$$

At the points of c_1 both terms are zero; in the close vicinity of c_1 the quadratic expression B_1^2/w_1 remains small, thus the I-patch behaves as P_1, satisfying our expectations.

The distance-based formulation helps to efficiently evaluate the surface in the interior, and set the individual weights w_i (see Section 4.6).

4. Constructing I-patches

In this section we discuss how to construct the components of I-patches, including boundaries, primary ribbons and bounding surfaces. We explain how to orient them, and what are the options for shape editing.

The I-patch has n corner points denoted by $\mathbf{Q}_i$ (Figure 4), and for each there is a well-defined tangent plane π_i with normal vector $\mathbf{N}_i$ (see concrete constructions in Section 5). Each primary ribbon P_i interpolates the i-th boundary curve and satisfies the tangential constraints at the related two corners. We deal with straight boundaries, conic boundaries, I-segments and general boundary curves.

4.1. Straight boundaries

The simplest case is when the tangent planes π_i and π_{i+1} are identical, containing the desired straight segment $\overline{\mathbf{Q}_i\mathbf{Q}_{i+1}}$. Then P_i is directly defined, and B_i will be a plane orthogonal to P_i, containing both $\mathbf{Q}_i$ and $\mathbf{Q}_{i+1}$.

4.2. Conic boundaries

A conic boundary c_i is defined by a control triangle, using Liming's formula [21], as follows. Take the corner points and pick a middle control point (denoted by $\mathbf{E}_i$) on the intersection

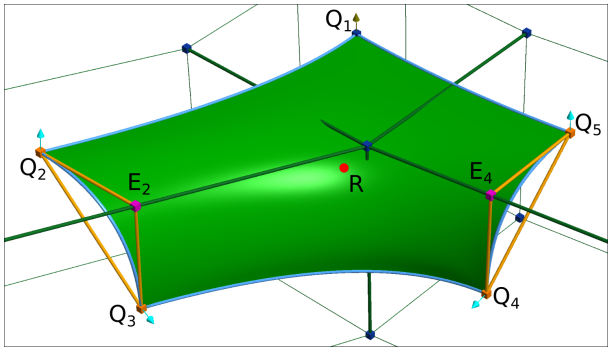


Figure 4: Control triangles. The small arrows show the normal vectors $\mathbf{N}_i$ corresponding to the corner points $\mathbf{Q}_i$, $\mathbf{R}$ is a reference point.

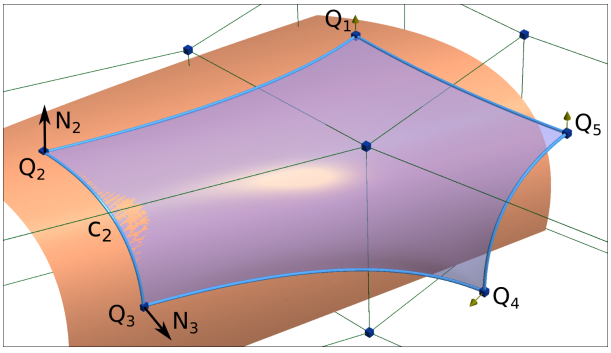


Figure 5: A ribbon surface on the $\mathbf{Q}_2\mathbf{Q}_3$ boundary.

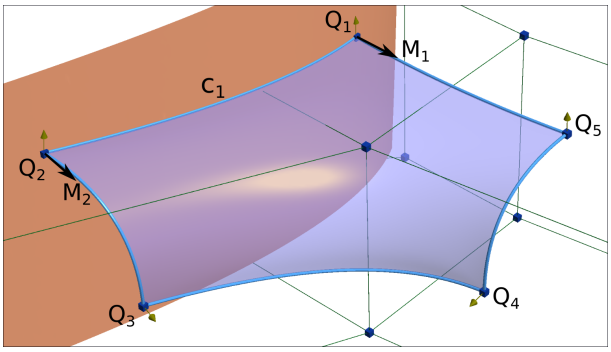


Figure 6: A curved bounding surface on the $\mathbf{Q}_1\mathbf{Q}_2$ boundary.

line of planes π_i and π_{i+1}. Then the segments $\overline{\mathbf{Q}_i\mathbf{E}_i}$, $\overline{\mathbf{E}_i\mathbf{Q}_{i+1}}$, $\overline{\mathbf{Q}_i\mathbf{Q}_{i+1}}$ determine three lines in implicit form, denoted by l_1, l_2 and l_3, respectively. The curve c_i goes smoothly from l_1 to l_2, touching them at their intersection points with l_3. This is a planar implicit equation given in the form

$$c_i(x, y) = (1 - \lambda_i)l_1 l_2 - \lambda_i l_3^2 = 0. \tag{11}$$

Here λ_i is the fullness factor, which also determines the type of conic. For example, in Figure 4 $\mathbf{Q}_2\mathbf{E}_2\mathbf{Q}_3$ represents a control triangle for curve c_2. All conics can also be defined in rational Bézier form $c_i(t)$, using the same three control points and a rational weight – a property we are going to exploit later, see the tessellation algorithm in Section 6.4.

We consider three cases.

Case 1. Let us construct a quadratic primary ribbon P_i that contains c_i. It is a natural idea to use Liming's technique for three planes, then

$$P_i(x, y, z) = (1 - \lambda_i)\pi_i\pi_{i+1} - \lambda_i\widetilde{\pi}_i^2 = 0, \tag{12}$$

where $\widetilde{\pi}_i$ denotes the cutting plane. This must contain the chord $\overline{\mathbf{Q}_i\mathbf{Q}_{i+1}}$, but there is a degree of freedom, called the *sweep direction* s_i, by means of which we actually determine this plane. The normal of $\widetilde{\pi}_i$ will always be the vector product of the chord and s_i.

One special case is when s_i is chosen to be orthogonal to the plane of the control triangle, and simple conical sweeps are generated. If both $\mathbf{N}_i$ and $\mathbf{N}_{i+1}$ are contained in the plane of the control triangle, and the conic represents a circular arc, a cylindrical surface is obtained. In order to set a good sweep direction for arbitrary tangent plane normals, we propose to set it orthogonal to the average of $\mathbf{N}_i$ and $\mathbf{N}_{i+1}$; this method degenerates to the above cylindrical surface. In these cases, a *planar bounding surface* B_i is used, determined by the control triangle, thus c_i is reproduced. An example of a Liming-ribbon, which touches the corners $\mathbf{Q}_2$ and $\mathbf{Q}_3$, can be seen in Figure 5.

Case 2. Another possible way of creating conic boundaries is when the primary ribbon P_i is set as the plane of the control triangle, being incident to π_i and π_{i+1}. Then we define B_i as a quadratic Liming surface using binormals $\mathbf{M}_i$ at the two corners. This construction produces *curved bounding surfaces*. Since B_i also contains the conic, the intersection with P_i will reproduce c_i, as required. Figure 6 shows a curved bounding surface interpolating corners $\mathbf{Q}_1$ and $\mathbf{Q}_2$.

Case 3. We may need to create conic boundaries by combining the previous two cases. When the angle of the sweep direction with the plane of the control triangle is less than a given tolerance, we retain the general quadratic Liming-ribbon P_i, and intersect it with a curved bounding surface B_i.

It is a necessary condition for the existence of a conic boundary with Liming-ribbons that the opposite corner points fall into the consistently oriented positive half spaces of the tangential planes. Formally, $\pi_i(\mathbf{Q}_{i+1}) > 0$ and $\pi_{i+1}(\mathbf{Q}_i) > 0$ must hold. This condition was also pointed out by Bajaj and Ihm [30]. An example is shown in Figure 7; the top curve of an "almost toroid" surface lies in the xy-plane, while the left normal vector $\mathbf{N}_2$ is tilted forward, and the right normal vector $\mathbf{N}_1$ is tilted backwards. Here the above criterion is not satisfied, and no appropriate conic ribbon can be inserted, since the two normals would point into different half-spaces, yielding a nonsense shape. In these cases, we need to apply another advanced construction, described in the next section.

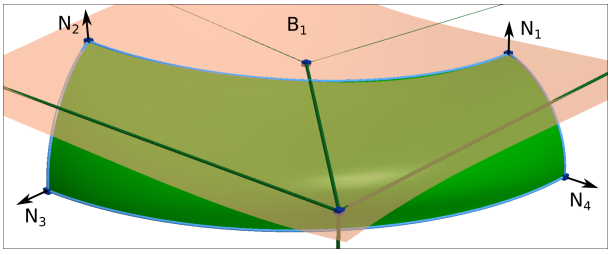


Figure 7: An I-ribbon with twisted normal vectors.

4.3. I-segments and I-ribbons

First we describe I-segments. These are the 2D counterparts of I-patches, where two implicit curves are being blended. I-segments, defined by quartic polynomials, are more general curves than conics. Given two primitive lines l_i and two local bounding lines h_i in the plane of the control triangle, the segment runs from one intersection point to the other (see examples in Figure 8). We use the following polynomial formula:

$$I(x, y) = w_1 l_1 h_2^2 + w_2 l_2 h_1^2 + w_0 h_1^2 h_2^2 = 0. \quad (13)$$

I-ribbons are based on the same logic, combining tangent planes π_i and π_{i+1} at two adjacent corners with local bounding planes μ_i and μ_{i+1}. Then

$$P_i(x, y, z) = w_{i,1}\pi_i\mu_{i+1}^2 + w_{i,2}\pi_{i+1}\mu_i^2 + w_{i,0}\mu_i^2\mu_{i+1}^2 = 0. \quad (14)$$

A natural choice for the local bounding surfaces is to use two parallel planes perpendicular to the chord and interpolating the corner points. Thus we have created an I-ribbon P_i; its corresponding bounding surface B_i can either be a plane or a curved bounding surface, as discussed for Liming-ribbons. In Figure 7 an I-ribbon is shown that resolves the difficult case with twisted normals mentioned above.

4.4. General boundary curves

In addition to these special constructions, arbitrary primary and bounding surfaces can also be used, producing general 3D boundary curves.

4.5. Orienting ribbons and bounding surfaces

In contrast to functional splines, the orientation of the ribbons is crucial for I-patches. These ribbons occur in separate terms in the equation (not as a single product), and for each ribbon we require that the local gradient must have the same direction as that of the final patch.

Let us see some simple 2D examples in Figure 8. In 8a the functional spline curve matches the orientation of the primaries. In 8b we have an I-segment with inconsistent orientation, yielding a wrong curve, however, 8c is consistent, and we obtain a good result. In 8d the functional spline curve cannot properly match the given orientations, and a nonsense curve is obtained. Fig. 8e shows an incorrectly oriented I-segment, but in 8f the expected result is obtained. It can be seen that the functional spline has problems with creating inflections, while the I-segment can solve these if the primary orientation is consistent.

It is a basic assumption that we wish to keep the I-patch within the union of the positive half-spaces of the bounding surfaces, i.e., we envision the shape on the same side of the B_i-s. (We need to check whether this condition is satisfied, and modify some of the boundary surfaces, when needed, see Section 6.1.) We also assume that a consistently oriented, piecewise normal vector fence exists for the boundary loop, as shown in Figure 9. Matching this will define the correct orientation (sign) of the primary surfaces.

4.6. Setting the shape parameters

If we construct Liming-ribbons, we may assign a fullness value λ_i for each boundary; circular arcs are often chosen, to connect to cylinders. In principle, the sweep direction of these ribbons may also be a parameter to be tweaked.

Once the primaries or the primary ribbons are defined, we need to set the w_i weights. We are not aware of any single best method to do so, however, we apply a heuristic setting that yields good default values in the majority of cases. All w_i must be positive, since the orientation of primaries – as set in the previous subsection – must be retained. Let us pick a reference point $\mathbf{R}$ that is typically, but not necessarily, a point to be interpolated by the I-patch. We set the weighting of the components by satisfying $w_i P_i(\mathbf{R})/B_i^2(\mathbf{R}) = \pm 1$, using the algebraic distances from Section 3.3. The sign depends on the location of the reference point with respect to the ribbon. For example, in Figure 4, the reference point is under the $\mathbf{Q}_1\mathbf{Q}_2$ ribbon, but above the $\mathbf{Q}_2\mathbf{Q}_3$ ribbon. Thus we combine the surfaces in a way that they will have the same contribution at the reference point.

Keeping the above initial w_i values, we have freedom to choose an arbitrary w_0 weight in Eq. (1), that will globally control the fullness of the patch. The geometric meaning is that this value corresponds to the sum of the distances, so accordingly tighter or looser patch interiors can be produced. We set the coefficients such that in $\mathbf{R}$ the sum of the absolute values of all weighted algebraic ribbons is one; however, their sign depends on the sign of the P_i-s. We want the patch to interpolate $\mathbf{R}$, so w_0 needs to be set to $\sum \mathrm{sgn}(P_i(\mathbf{R}))$, which ensures that the patch interpolates $\mathbf{R}$.

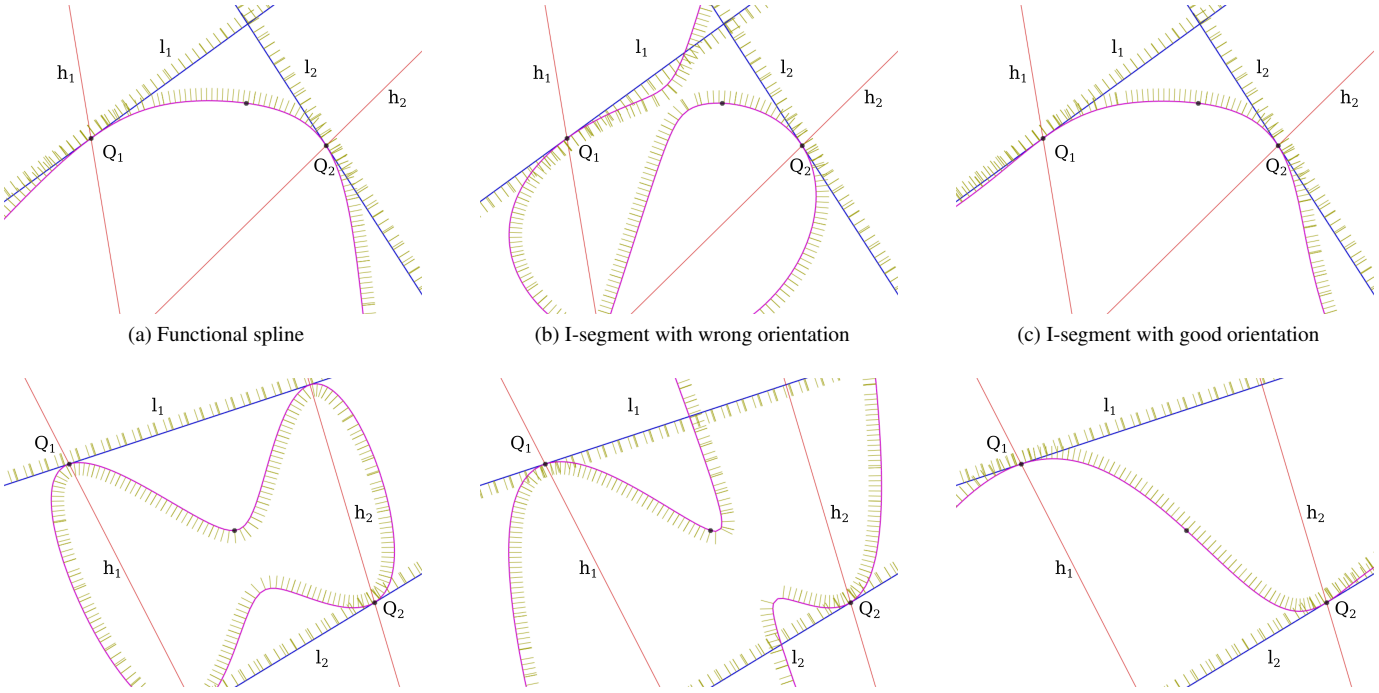
Of course, it may make sense to modify the default weights w_i. In our experience, these are appropriate for simple models, but for complex, twisted configurations some tuning may be necessary. It is possible to manually edit the weights, but this requires experience. Instead we propose to run a numerical optimization: first we create an initial triangulated mesh (as described in Section 6.4) using the defaults, then these values are updated by minimizing the polyhedral energy functional proposed in [34]. Generally a good shape is obtained.

5. Surface modeling with I-patches

In this section we show two application areas where I-patches are useful. The first is *setback vertex blending*, where a multi-sided I-patch is formed to connect primary surfaces and edge blends. The second application is *polyhedral design*, where a complex free-form shape is defined by means of a control polyhedron, yielding a collection of smoothly connected I-patches. The common boundaries and the ribbons are naturally derived.

5.1. Setback vertex blending

Vertex blends play an important role in surface and solid modeling. While *edge blends* (fillets) replace sharp intersection curves, *vertex blends* smoothly connect converging edge blends. Vertex blending is a difficult modeling task, as the edges to be blended may represent very complex configurations; we



(a) Functional spline (b) I-segment with wrong orientation (c) I-segment with good orientation

(d) Functional spline (e) I-segment with wrong orientation (f) I-segment with good orientation

Figure 8: 2D curves with functional splines and I-segments.

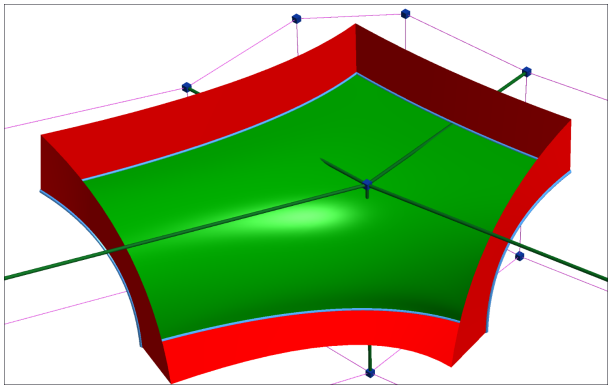


Figure 9: Patch with a consistent normal fence; all primaries must have the same (e.g. positive) half-space in the direction of the fence.

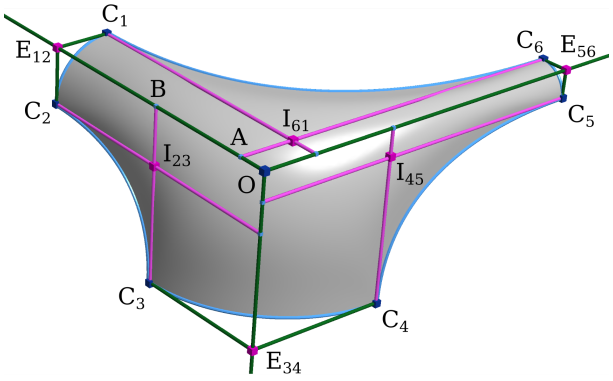


Figure 10: Construction of a setback vertex blend.

may have convex, concave or smooth edges with uneven angles and radii, and must handle many special cases, including tangential, cuspate and degenerate edge pairs.

A general solution to this problem is *setback vertex blending,* see the early papers by Braid [35] and Várady et al. [36, 37]. The termination of the edge blends are pushed away from the vertex by various setback values to ensure the necessary space for a smooth transition. Later further multi-sided setback vertex blends were published using variants of functional splines [19] and S-patches [38]. It has been emphasized in [37] that for connecting n edge blends the most genuine surface model is not an n-sided, but a $2n$-sided patch, although it is possible that the setback vertex blends degenerate, having an arbitrary number of sides between n and $2n$.

The basic components of a setback vertex blend are shown in Figure 10. Here we describe vertex blends for polyhedra, but the concept nicely generalizes for more complex ribbons and bounding surfaces. The i-th edge blend, generally a cylinder, is bounded by two *rail curves* (colored pink), that run on the "previous" and "next" primary surfaces. The edge blends terminate at *profile curves* (blue)*;* for example, one connects two corner points $\mathbf{C}_1$ and $\mathbf{C}_2$. Together with the edge point $\mathbf{E}_{12}$, a planar control triangle is formed, a placeholder for a circular (or conic) profile.

On each primary surface there are two corners that need to be connected by a *spring curve* (blue)*;* for example, the one that connects $\mathbf{C}_2$ and $\mathbf{C}_3$. The rail curves that pass through these points intersect at point $\mathbf{I}_{23}$, defining another triangle for a conic segment. Altogether, the vertex blend is bounded by an alternating sequence of profile curves and spring curves. A formula to

compute the extent of the setbacks can be found in Appendix A; for more details see [37].

The actual construction is fairly straightforward: first an alternating sequence of cylindrical edge blends and planar primary surfaces are computed, then these are intersected by an alternating sequence of bounding planes and curved bounding surfaces. We define a reference point by adding to the central vertex $\mathbf{O}$ a displacement vector based on the average deviation between the edgepoints E_{ij} and the midpoints of the related circular profiles. Then we set the weights w_i and the fullness of the patch, and apply the I-patch equation. We will discuss potential ribbon problems and show interesting examples in Section 6.

5.2. Polyhedral design

Modeling with control polyhedra has many manifestations. The most well-known is recursive subdivision (see e.g. [39]), but quadrilateral parametric surfaces can also be stitched together in various ways (see e.g. [40]). In most of these approaches, handling irregular vertices is still a crucial research issue. Quadratic and cubic S-patches can also be used for polyhedral design [41], but the topological structures in these constructions are somewhat limited.

Here we propose a polyhedron-based representation that produces smoothly connected I-patches. The control polyhedron has a general topology with faces of arbitrary number of sides and vertices of arbitrary valency. The faces are not necessarily planar, and the control structure may be open or closed, but T-nodes are not permitted.

The surface model is determined by a free-form curve network of conic arcs. Its topological structure corresponds to the dual graph of the polyhedron, see Figure 11. For each face of the polyhedron we compute a centroid, $\mathbf{Q}_i$. For each straight segment of the polyhedron there is a corresponding crossing curved edge that connects the centroids of the neighboring faces. For each vertex of the polyhedron with valency k we compute a k-sided surface patch, bounded by a loop of k curved edges.

At the centroid we determine a local tangent plane; for planar control faces this is obvious, for non-planar faces we compute a best-fit plane that contains the centroid and approximates the midpoints of the edges in least-squares sense. The Liming-ribbons (or the I-ribbons) are uniquely defined by two centroids and two local tangent planes, being shared by the adjacent patches along the common boundary.

A simple example is shown in Figure 11. The control polyhedron has one 5-sided and six 4-sided faces. It has internally two 3-valent vertices and one 5-valent vertex, which will yield two 3-sided patches and one 5-sided patch. The $\mathbf{Q}_2\mathbf{Q}_3$ ribbon, for example, is then uniquely defined for both the left 3-sided and the 5-sided patch, ensuring smooth connection.

6. Discussion

We would like to emphasize that modeling with implicit surfaces is a difficult area. We have discussed the mathematical background of I-patches; however, we have to keep in mind that these schemes will produce good shapes only if the ribbons

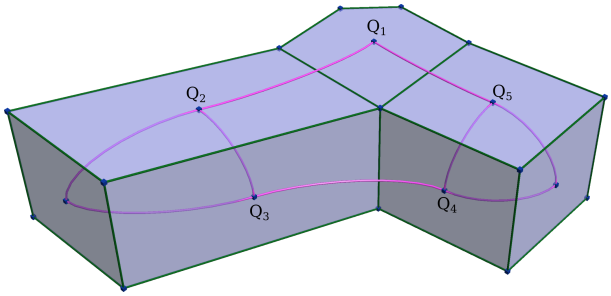


Figure 11: Control cage for polyhedral design.

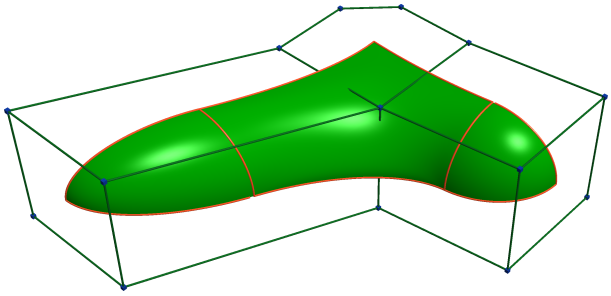


Figure 12: The generated curve network and surfaces.

and the bounding surfaces are "suitable" for patch generation. Certain problems can be detected, and the components can be fixed, as explained below. In the second half of this Section we will compare I-patches with alternative schemes, then show a few test examples, and discuss our tessellation algorithm.

6.1. Handling ribbon problems

In Section 4.3 we have already discussed special cases, where it was impossible to create a Liming-ribbon, and we had to apply a more complex construction, the I-ribbon. Here we deal with two further problems.

6.1.1. Poor ribbons

The I-patch scheme may produce dubious shapes, if the distance fields produced by the primary ribbons abruptly change their sign within the space where the I-patch is going to be created. This may be due to multiple surface branches in the vicinity of the boundaries, or high curvatures, when the ribbon "turns under" itself; see the transparent surfaces in Figures 13 and 14. Self-intersections within the ribbon may also lead to visible shape problems.

In these cases we have different options to repair the ribbons. We can change the fullness λ_i of the boundary conics, which strongly affects the shape of the Liming-ribbons, as well. As the curvature becomes smaller, the artifacts disappear. Such an example can be seen in Figure 13, where the original ribbon of the I-patch was replaced by a "broader" one that locally prevents interference with the interior of the patch.

Another option is to change the representation of the primary ribbon, and exclude the highly curved portion or the undesired branch. We can modify Liming's method to obtain a *piecewise*

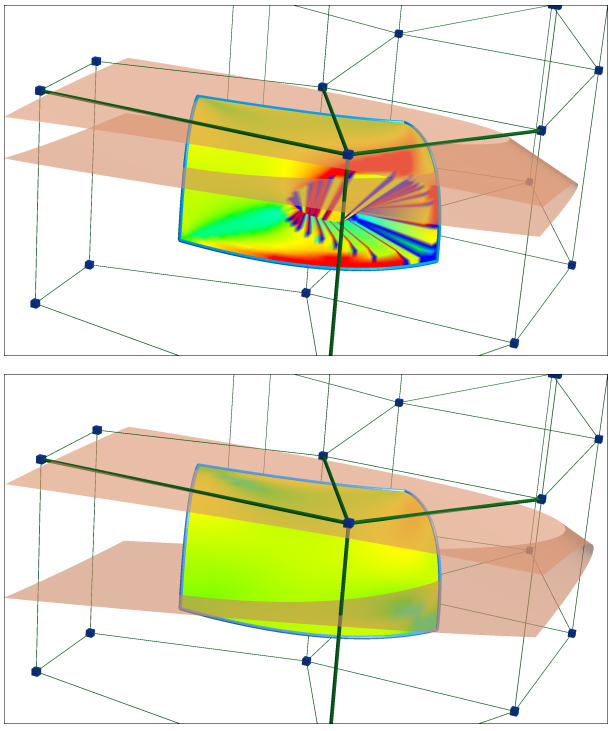


Figure 13: Fixing a shape artifact by tuning the fullness of the ribbons.

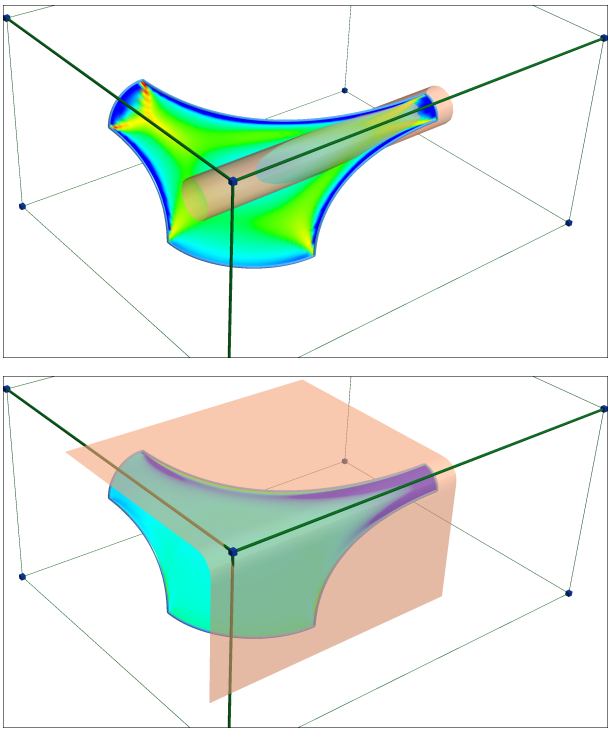


Figure 14: Fixing a shape artifact by using piecewise ribbons.

Liming-ribbon:

$$P_i(x,y,z) = \begin{cases} (1-\lambda)\pi_i\pi_{i+1} - \lambda\widetilde{\pi}_i^2 & \widetilde{\pi}_i \geq 0, \\ \pi_i\pi_{i+1} & \widetilde{\pi}_i < 0. \end{cases} \tag{15}$$

An example is shown in Figure 14, where a cylinder with small radius was replaced by a piecewise ribbon, yielding a good setback vertex blend. The drawback of this method is that having G^1 continuity along $\widetilde{\pi}_i$ may affect internal continuity within the patch, although in many cases the curvature maps or the isophote lines do not indicate this phenomenon at all, see for example Figure 15.

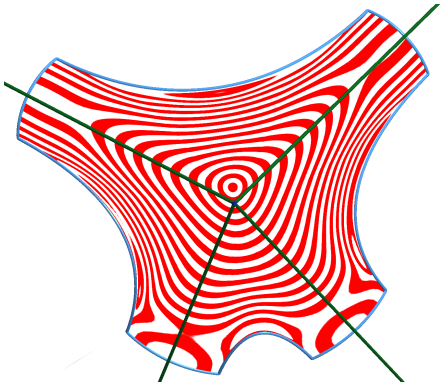


Figure 15: Isophote lines on an 8-sided I-patch with piecewise ribbons.

6.1.2. *Intersecting bounding surfaces*

We assume that the union of the bounding surfaces forms a well-defined, connected space for the patch to be created, and accordingly all bounding surfaces are supposed to have a constant sign inside the patch. This is violated when a bounding surface B_j intersects another boundary curve, where $P_i = B_i = 0$, and thus I-patches of unacceptable quality are obtained. We can detect this problem by checking whether the bounding surface intersects the boundary loop on the distant boundaries. An example is shown in Figure 16, where the top right bounding surface intersects the bottom left boundary. The problem is fixed, if we use a curved bounding surface.

6.2. *Comparisons with other schemes*

In the following we will show some comparisons with functional splines [7, 19] and algebraic Hermite interpolation [17].

6.2.1. *Functional splines*

Functional splines [7] are defined by the equation

$$F(x,y,z) = (1-\lambda)f - \lambda g^{k+1} = 0, \tag{16}$$

a generalization of Liming's formula, cf. Equation (11). Here f is the *base surface*, and g is the *transversal surface*. The resulting isosurface interpolates the intersection of f and g, joining to the former with G^k continuity. While in special cases it can be simplified, generally this means that $f = \prod_i P_i$ and $g = \prod_i B_i$.

It has been known [42] that a large class of low-degree functional splines are convex, and thus cannot be used for specific boundary configurations. Take for example the 6-sided hole loop with alternating convex and concave boundary constraints in Figure 17. In [7], this model was generated as a collection of six smoothly connected functional splines. As discussed in Section 4.5, with I-patches we can set the sign of each primary so that a consistent normal orientation can be achieved.

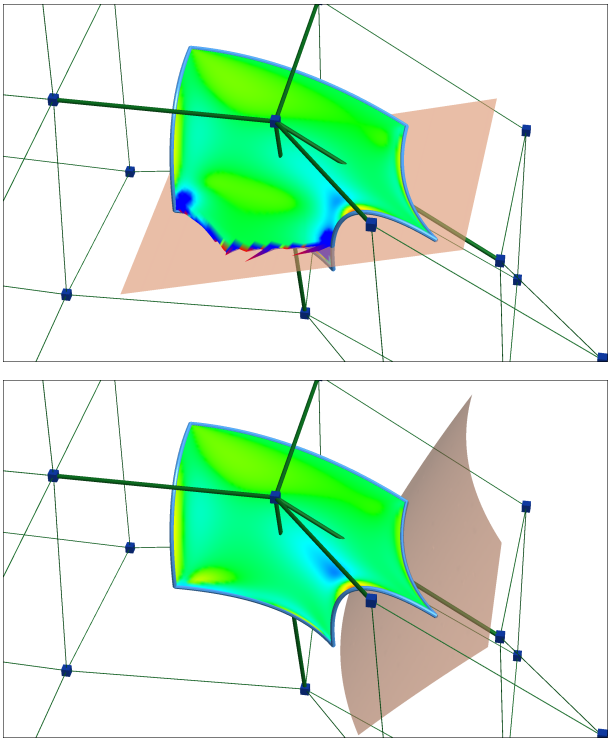


Figure 16: Fixing a shape artifact by using a curved bounding surface.

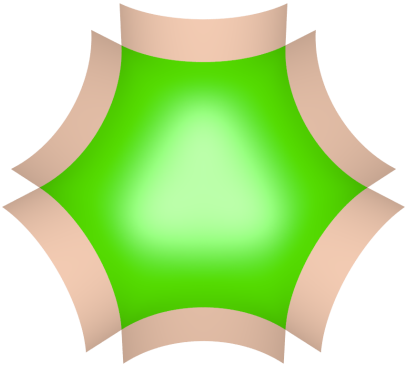


Figure 17: A six-sided hole loop with alternating convexity of curves.

A variation of the above definition is the *symmetric functional spline* [19]

$$F(x, y, z) = (1 - \lambda) f_a g_b^{k+1} - \lambda f_b g_a^{k+1} = 0, \tag{17}$$

which – similarly to I-patches – combines the primary and bounding surfaces. This modified formula allows the creation of non-convex surfaces, such as *house corner blends*, but some ingenuity is required in choosing the correct base and transversal surfaces (see its application for vertex blends in [19]). In contrast, I-patches, having separate components, have more potential to be tuned as they have more degrees of freedom. This is illustrated in Figure 18, where isophote lines are compared.

6.2.2. *Algebraic Hermite interpolation*

The fitting framework described in [17] gives a direct solution for the hole-filling problem, leading to a homogeneous system of linear equations. The algorithm ensures finding the surface of least degree satisfying the boundary constraints. Finding an acceptable solution, however, is not trivial, as we will demonstrate in the following example.

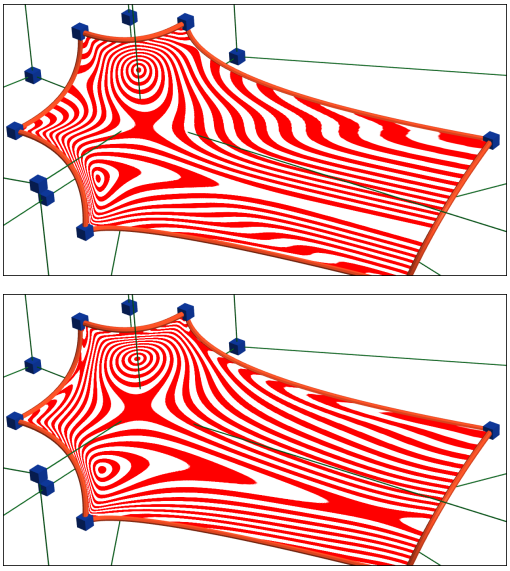


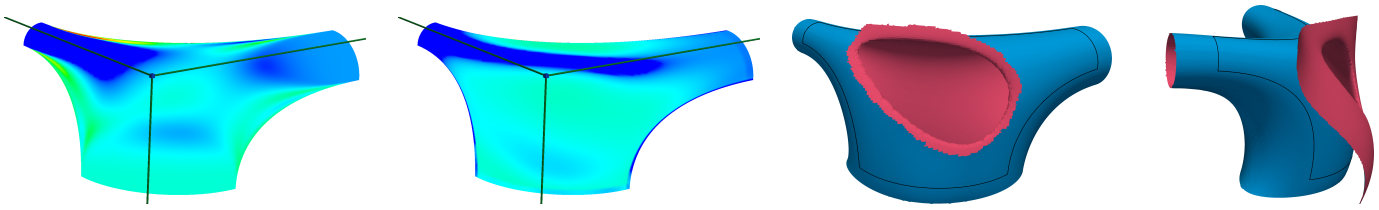
Figure 18: Isophote lines on a six-sided patch, with symmetric functional spline (top) and I-patch (bottom).

Take the setback vertex blend configuration shown in Figure 19a. The profile curves are circular arcs, while the spring curves are parabolas. The normal vectors at the vertices are defined by the end tangent vectors of the adjacent curves; the normal fence along the curves is a linear blend between the end normals, as in [30].

The least degree for which the system is solvable is four, but the (unique) quartic patch contains several singularities (normal flips) on its boundary. Going one degree higher, we find the nice surface shown in Figure 19b. The problem is that this is but one of the infinitely many solutions, as the kernel is 6-dimensional, and any linear combination of its basis vectors corresponds to a valid solution. Most of these, however, contain self-intersections or have large bumps in the interior. Figure 19c, for example, shows another quintic surface that nicely interpolates the boundary constraints, but has a gaping hole inside. (Here the full isosurface is shown to appreciate its form; black lines show the setback vertex blend boundaries.)

6.3. *Test cases*

1. We have shown a 6-sided vertex blend earlier in Figure 10. This connected three cylindrical edge blends with radii 20, 50 and 10 and setbacks 100, 60 and 125. These parameters can easily be edited.
2. An interesting 10-sided patch is shown in Figure 20, connecting three cylinders of almost 90 degree arcs with two very flat cylinders. The I-patch is sliced with parallel planes, and the curvature map is also shown. Piecewise Liming-ribbons were used.
3. The next test case is a combination of connected setback vertex blends, applied on a polyhedron. In Figure 21, three 8-sided and three 6-sided patches smoothly connect to pla-



(a) I-patch with mean curvature (b) Hermite fit with mean curvature (c) Another Hermite fit (full isosurface)

Figure 19: Setback vertex blend with uneven radii.

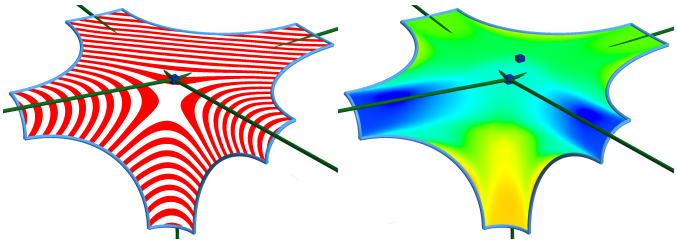
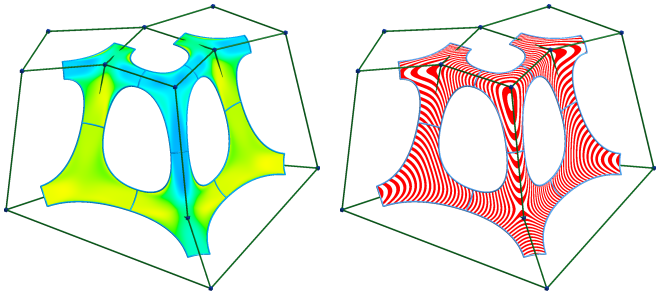


Figure 20: A ten-sided patch with contours and mean curvature.



(a) Curvature map (b) Contours

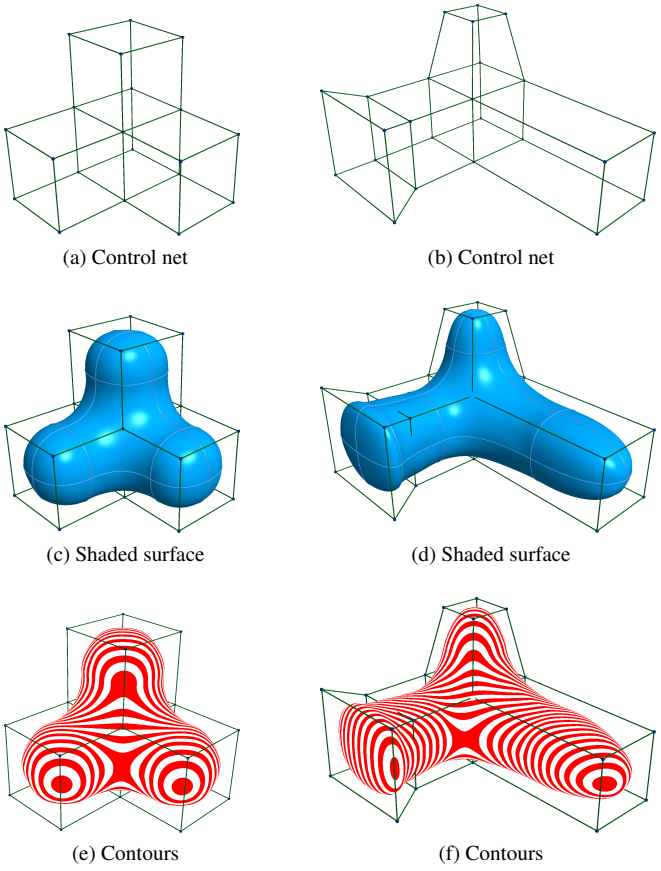
Figure 21: Connected vertex blends.

nar faces (not shown). The patches are displayed with curvature map and contours.

4. We have shown earlier a collection of three smoothly connected I-patches defined by a control polyhedron, see Figure 12.
5. Finally, we mimic polyhedral design using a very simple test object (Figure 22). We performed simple operations, such as dragging the right face and repositioning the top vertices, the shape was modified in a natural manner. On the left side, we extruded the left face and created a new knob-like shape.

6.4. *Tessellation*

I-Patches with parametrizable boundary curves can be tessellated in the following way. First create an auxiliary multi-sided parametric patch (in this paper the C^0 multi-sided Coons patch is used, see Appendix B) with the same boundary curves as the I-patch. Then compute normal vectors at the corner points from the end tangents of the boundaries. After tessellating the auxiliary patch, at each mesh vertex define a ray by taking a convex combination of the corner normals using generalized barycentric coordinates [43]. Along the boundaries this will be a linear



(a) Control net (b) Control net

(c) Shaded surface (d) Shaded surface

(e) Contours (f) Contours

Figure 22: Polyhedral design examples.

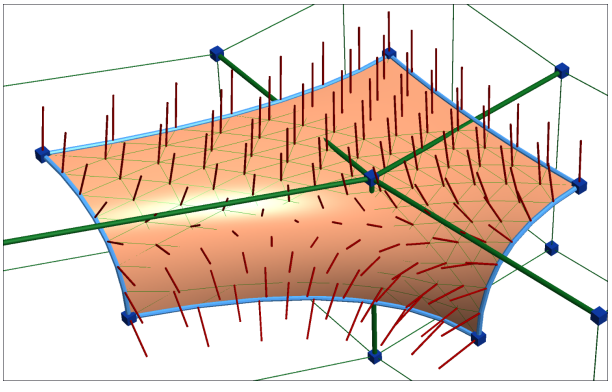


Figure 23: Tessellation by rays emanating from a parametric surface.

sweep between the corner normals; it is assumed that in the interior the rays remain within the union of the positive half-spaces of the bounding surfaces. The rays intersect the I-patch and produce a tessellation (see Figure 23); the angles between the rays and the gradients of the surface should remain under a prescribed threshold. When this is violated, the I-patch is categorized as "poor", possibly having abrupt curvature changes, internal holes or branching. In this case the I-patch needs to be repaired (see Section 6.1). Experience shows that this algorithm produces a nice mesh for valid I-patches and is suitable to recognize ill-conditioned cases.

Conclusion

In this paper we revisited classical surfacing techniques using implicit patches, and attempted to widen the limits of former representations. We believe that the use of implicit multi-sided patches is justified when we want to accurately connect to other, mainly regular, implicit surfaces, and/or want to exploit the well-known benefits of algebraic representations. I-patches represent an interesting approach, where the primary surfaces can be individually controlled by weighted bounding surfaces. We have discussed the basic construction with useful geometric observations, including a distance-based interpretation, consistent orientation of the primaries, and curved bounding surfaces. In our I-patch applications of setback vertex blending and polyhedral design, mainly Liming-ribbons with conic boundaries were combined; however, various cases with more general surface components have also been studied.

Concerning future work, there are interesting issues to enhance the current scheme, including the generation of the most suitable bounding surfaces and the automatic calculation of the ribbon weights that could replace the current heuristic method. The GPU implementation of I-patches is also an important computational topic.

Acknowledgements

This project has been supported by the Hungarian Scientific Research Fund (OTKA, No. 124727), and the National Research, Development and Innovation Fund (TUDFO/51757/2019-ITM, Thematic Excellence Program). The majority of the images were generated by the Sketches prototype system (ShapEx Ltd, Budapest); the authors acknowledge the essential programming contribution of György Karikó.

Appendix A. Computing setbacks

It is crucial to set appropriate setbacks, as they define the distances of the profile planes from the vertex. This is explained using Figure 10. Roughly speaking, the formula below takes into consideration the previous and the next rail curve constraints, denoted by $\text{range}_{\text{prev}}$ and $\text{range}_{\text{next}}$, and adds a setback offset to ensure sufficient turning space for the spring curves:

$$\text{setback} = \max(\text{range}_{\text{prev}}, \text{range}_{\text{next}}, 0) + \text{offset}. \tag{A.1}$$

Take, for example, the setback that belongs to the profile curve running from $\mathbf{C}_1$ to $\mathbf{C}_2$. Here $\text{range}_{\text{prev}} = |\mathbf{AO}|$, $\text{range}_{\text{next}} = |\mathbf{BO}|$, and $\text{setback}_{12} = |\mathbf{E}_{12}\mathbf{O}|$. Note that the maximum function must always yield a positive value, as we need to avoid negative ranges that occur at concave edge blends.

Appendix B. Multi-sided C^0 Coons patch

The C^0 Coons patch is a classic surface scheme that interpolates four boundary curves. It can be generalized to any number of sides in the following way [44]:

1. For each side i, a *Coons ribbon* R_i is defined as a C^0 Coons patch based on three consecutive boundaries $\{i-1, i, i+1\}$. The fourth curve is created as a cubic Bézier curve interpolating the position and first derivative at the start- and endpoint of the $(i+2)$nd and $(i-2)$nd boundaries, respectively.
2. The domain of the n-sided patch is a regular n-sided polygon Ω_n. For any domain point, generalized barycentric coordinates $\{\lambda_j\}$ can be computed. Then local parameters are assigned to each side:
$$d_i = 1 - \lambda_{i-1} - \lambda_i, \qquad s_i = \lambda_i/(\lambda_{i-1} + \lambda_i). \tag{B.1}$$
3. Finally, the patch is defined to be
$$S(p) = \sum_{i=1}^{n} R_i(s_i, d_i)\frac{1-d_i}{2}, \tag{B.2}$$
where (s_i, d_i) are the local parameters of the point $p \in \Omega_n$.

For further details refer to the original paper.

References

[1] Várady, T, Benkő, P, Kós, G, Rockwood, A. Implicit surfaces revisited – I-patches. In: Geometric Modelling. Springer; 2001, p. 323–335.
[2] Charrot, P, Gregory, JA. A pentagonal surface patch for computer aided geometric design. Computer Aided Geometric Design 1984;1(1):87–94.
[3] Loop, CT, DeRose, TD. A multisided generalization of Bézier surfaces. ACM Transactions on Graphics (TOG) 1989;8(3):204–234.
[4] Várady, T, Rockwood, A, Salvi, P. Transfinite surface interpolation over irregular n-sided domains. Computer-Aided Design 2011;43(11):1330–1340.
[5] Hart, JC. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer 1996;12(10):527–545.
[6] Seyb, D, Jacobson, A, Nowrouzezahrai, D, Jarosz, W. Non-linear sphere tracing for rendering deformed signed distance fields. ACM Transactions on Graphics (TOG) 2019;38(6):1–12.
[7] Li, J, Hoschek, J, Hartmann, E. G^{n-1}-functional splines for interpolation and approximation of curves, surfaces and solids. Computer Aided Geometric Design 1990;7(1-4):209–220.
[8] Bloomenthal, J, Bajaj, C, Blinn, J, Wyvill, B, Cani, MP, Rockwood, A, et al. Introduction to implicit surfaces. Morgan Kaufmann; 1997.
[9] Hoffmann, C, Hopcroft, J. Automatic surface generation in computer aided design. The Visual Computer 1985;1(2):92–100.
[10] Rockwood, AP. The displacement method for implicit blending surfaces in solid models. ACM Transactions on Graphics (TOG) 1989;8(4):279–297.
[11] Ricci, A. A constructive geometry for computer graphics. The Computer Journal 1973;16(2):157–160.
[12] Pasko, A, Adzhiev, V, Sourin, A, Savchenko, V. Function representation in geometric modeling: concepts, implementation and applications. The Visual Computer 1995;11(8):429–446.

[13] Dekkers, D, Van Overveld, K, Golsteijn, R. Combining CSG modeling with soft blending using Lipschitz-based implicit surfaces. The Visual Computer 2004;20(6):380–391.
[14] Gourmel, O, Barthe, L, Cani, MP, Wyvill, B, Bernhardt, A, Paulin, M, et al. A gradient-based implicit blend. ACM Transactions on Graphics (TOG) 2013;32(2):1–12.
[15] Zanni, C, Gleicher, M, Cani, MP. N-ary implicit blends with topology control. Computers & Graphics 2015;46:1–13.
[16] Garrity, T, Warren, J. Geometric continuity. Computer Aided Geometric Design 1991;8(1):51–65.
[17] Bajaj, CL, Ihm, I. Algebraic surface design with Hermite interpolation. ACM Transactions on Graphics (TOG) 1992;11(1):61–91.
[18] Hartmann, E. Blending of implicit surfaces with functional splines. Computer-Aided Design 1990;22(8):500–506.
[19] Hartmann, E. Implicit G^n-blending of vertices. Computer Aided Geometric Design 2001;18(3):267–285.
[20] Zhu, CG, Wang, RH, Shi, X, Liu, F. Functional splines with different degrees of smoothness and their applications. Computer-Aided Design 2008;40(5):616–624.
[21] Liming, RA. Practical analytic geometry with applications to aircraft. Macmillan; 1944.
[22] Pasko, GI, Pasko, AA, Kunii, TL. Bounded blending for function-based shape modeling. IEEE Computer Graphics and Applications 2005;25(2):36–45.
[23] Warren, J. Blending algebraic surfaces. ACM Transactions on Graphics (TOG) 1989;8(4):263–278.
[24] Wyvill, B, Guy, A, Galin, E. Extending the CSG tree. Warping, blending and boolean operations in an implicit surface modeling system. Computer Graphics Forum 1999;18(2):149–158.
[25] Wyvill, B, Foster, K, Jepp, P, Schmidt, R, Sousa, MC, Jorge, JA. Sketch based construction and rendering of implicit models. In: Computational Aesthetics. A K Peters/CRC Press; 2005, p. 67–74.
[26] Karpenko, O, Hughes, JF, Raskar, R. Free-form sketching with variational implicit surfaces. Computer Graphics Forum 2002;21(3):585–594.
[27] Warren, J. Free-form blending: A technique for creating piecewise implicit surfaces. In: Topics in surface modeling. SIAM; 1992, p. 3–21.
[28] Sederberg, TW. Piecewise algebraic surface patches. Computer Aided Geometric Design 1985;2(1-3):53–59.
[29] Bajaj, CL, Chen, J, Xu, G. Modeling with cubic A-patches. ACM Transactions on Graphics (TOG) 1995;14(2):103–133.
[30] Bajaj, CL, Ihm, I. Smoothing polyhedra using implicit algebraic splines. ACM SIGGRAPH Computer Graphics 1992;26(2):79–88.
[31] De Araújo, BR, Lopes, DS, Jepp, P, Jorge, JA, Wyvill, B. A survey on implicit surface polygonization. ACM Computing Surveys (CSUR) 2015;47(4):1–39.
[32] Hartmann, E. A marching method for the triangulation of surfaces. The Visual Computer 1998;14(3):95–108.
[33] Karkanis, T, Stewart, AJ. Curvature-dependent triangulation of implicit surfaces. IEEE Computer Graphics and Applications 2001;21(2):60–69.
[34] Pellis, D, Kilian, M, Dellinger, F, Wallner, J, Pottmann, H. Visual smoothness of polyhedral surfaces. ACM Transactions on Graphics (TOG) 2019;38(4):1–11.
[35] Braid, I. Non-local blending of boundary models. Computer-Aided Design 1997;29(2):89–100.
[36] Várady, T, Rockwood, A. Geometric construction for setback vertex blending. Computer-Aided Design 1997;29(6):413–425.
[37] Várady, T, Hoffmann, CM. Vertex blending: problems and solutions. In: Proceedings of the international conference on Mathematical methods for curves and surfaces II. Vanderbilt University Press; 1998, p. 501–527.
[38] Zhou, P, Qian, WH. Polyhedral vertex blending with setbacks using rational S-patches. Computer Aided Geometric Design 2010;27(3):233–244.
[39] Warren, J, Weimer, H. Subdivision methods for geometric design: A constructive approach. Elsevier; 2001.
[40] Peters, J. C^1-surface splines. SIAM Journal on Numerical Analysis 1995;32(2):645–666.
[41] Loop, C, DeRose, TD. Generalized B-spline surfaces of arbitrary topology. In: Proceedings of the 17th annual conference on Computer graphics and interactive techniques. ACM; 1990, p. 347–356.
[42] Hartmann, E, Feng, YY. On the convexity of functional splines. Computer Aided Geometric Design 1993;10(2):127–142.
[43] Hormann, K, Sukumar, N, editors. Generalized barycentric coordinates in computer graphics and computational mechanics. CRC Press; 2017.
[44] Salvi, P. A multi-sided generalization of the C^0 Coons patch. In: Proceedings of the Workshop on the Advances of Information Technology. 2020, p. 110–111. arXiv:2002.11347.