Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Doctoral School of Informatics

Implicit surface patches in 3D geometric modeling

Doctoral Dissertation

Author
Ágoston Sipos

Advisor
Dr. Tamás Várady

September 19, 2023

Contents

Kivonat

A számítógéppel segített geometriai modellezésben a digitálisan tárolt objektumok reprezentációja alapvető kérdés. Görbék, felületek és tömör testek intuitív módon definiálhatók matematikai egyenletek segítségével, ezáltal nagymértékben egyszerűsödik a tervezés, a gyártás, a végeselemes analízis és a grafikus megjelenítés.

A dolgozat fókuszában új felületreprezentációk vizsgálata áll, melyek az implicit felületek közé tartoznak: egy $F(x, y, z) = 0$ alakú egyenlet adja meg őket a 3-dimenziós térben. Az implicit felületeket jellemzően szabályos felületek (gömbök, hengerek, kúpok, tóruszok, stb.) megadására használják; ezzel szemben jelen disszertáció véges, N-oldalú implicit felületdarabokkal foglalkozik, amelyek képesek szabadformájú geometria reprezentálására és a felületek folytonos összeillesztésére; miközben megőrzik az implicit felületreprezentáció lényeges számítási előnyeit.

A legfontosabb eredmények közé tartozik az I-patch reprezentáció új matematikai tulajdonságainak felismerése és alternatív felületegyenletek kidolgozása, amely segítségével lehetőség nyílik N-oldalú határgörbe-hurkok interpolálására. Egy új „racionális" felírás alapján a felületet előjeles távolságok kombinációjaként lehet előállítani. A bevezetett „hűen közelítő" távolságtér jelentősen felülmúlja a korábbi távolságszámítási eljárásokat. Az új corner I-patch reprezentáció sarok-interpolánsokat kombinálva hoz létre felületeket, a korábbi oldal-interpoláns alapúakkal szemben.

Az implicit felületek használatának előnyei két alkalmazási területen kerülnek bemutatásra: szabadformájú modellezés kontrollpoliéderek alapján és háromszöghálók approximációja egymáshoz simán kapcsolódó I-patch-ek segítségével.

Abstract

In Computer Aided Geometric Design, the representation of objects in digital form is a cardinal question. Curves, surfaces, and solid models can intuitively be defined by continuous mathematical equations that provide major benefits for 3D design, manufacture, analysis, and rendering.

The focus of this dissertation is to explore new surface families that belong to the category of implicit surfaces, defined by an equation $F(x, y, z) = 0$ in the 3-dimensional space. While implicit surfaces are mostly used for modeling regular surfaces (spheres, cylinders, cones, torii, etc.), this dissertation focuses on finite, multi-sided implicit patches that are capable to represent freeform geometry, can ensure smooth connections and – at the same time – retain the computational advantages of implicit representation.

The results presented here include newly recognized mathematical properties and alternative formulations for the I-patch representation that can interpolate multi-sided boundary loops. A new rational formulation is suggested by means of which I-patches can be interpreted as a constrained sum of 3D distances. I have proposed a new faithful distance function that outperforms previous formulations with regard to distance estimation. The novel corner I-patch representation combines corner interpolants in contrast to the previously applied side interpolants.

The benefits of implicit surfacing have been demonstrated in two application areas: freeform design based on control polyhedra and approximation of meshes using smoothly connected patchworks of I-patches.

Introduction

Computer-Aided Design and Manufacturing (CAD/CAM) are cutting-edge technologies that significantly contributed to the progress of human civilization in the last decades. In a variety of areas such as architecture, mechanical engineering, vehicle engineering, or medicine, we frequently encounter entities designed and manufactured using CAD/CAM systems. The ease of creating complex 3D models on a computer has also been a primary factor in the development of emerging fields such as virtual or augmented reality and additive manufacturing (3D printing).

3D geometric modeling plays a key role in CAD/CAM systems. It deals with mathematical representations, computational algorithms, and a great variety of applications. The fundamentals of Computer Aided Geometric Design (CAGD) were laid down in the 1970s, since then a huge amount of research has been published presenting new geometric representations and design techniques.

The requirements of various applications are extremely diverse and often contradictory, and the advantages and deficiencies of the methods must be balanced. There are a few basic questions to be asked when a modeling scheme is selected.

Overview of modeling paradigms

Some of the most important factors to be considered are listed here.

Solid vs. surface model A fundamental question is whether we want to model a closed volume of the 3D space or just want to represent a surface shell. In the former case, we focus on operations to "add or remove material", and we need to maintain the geometric and topological consistency of the models. For surfaces, we generally apply a different set of operations to create, edit, and manipulate complex constrained geometries.

Regular vs. freeform geometry When creating surfaces or solids, we have the choice to use only regular surfaces (cylinders, cones, spheres, tori, etc.) as building blocks from which we build up our models with Boolean operations, or to use schemes that are capable of producing a wide variety of shapes based on the modification of simple entities (control

points, or other parameters). The latter is called freeform modeling and includes Bézier- and B-spline surfaces or Coons-patches. Although there is no clear-cut classification, the two paradigms have a dissimilar mathematical basis and support different operations and interrogation techniques.

Discrete vs. continuous representation There is also a choice between storing primitive data (points or scalars), and using linearized structures (e.g. triangle meshes, or voxel-based trilinear interpolation) to reproduce the surface; or using continuous mathematical representations of the 3D entities such as metaballs [Blinn, 1982] or NURBS. Exact mathematical representations usually require less data for representing the same surface with a similar level of detail and can guarantee exact smooth connections between components, also, it is possible to perform exact computations of various geometric quantities accurately, unlike the discrete models where only estimations can be computed.

Parametric vs. implicit equation Surface formulae can fall into two main categories: implicit surfaces are defined as the zero set of an $\mathbb{R}^3 \to \mathbb{R}$ function, while parametric surfaces are defined by a $\mathbb{R}^2 \to \mathbb{R}^3$ mapping of a 2-dimensional region into the 3D space. This choice correlates with the choice between regular and freeform surfaces: regular surfaces are generally represented by implicit equations while freeform surfaces are dominantly in parametric form. We remark, that the wide class of swept and lofted surfaces are often represented as a combination of the two basic surface classes.

General topology surfacing

In freeform modeling, the most frequently used schemes all fit into the category of *tensor product surfaces*, and thus always have two pairs of opposing sides. However, in practical design, in particular for modeling aesthetic objects, the use of multi-sided (i.e. non four-sided) patches is inevitable.

Handling non-four-sided patches has three main avenues. The first is trimming, when the multi-sided patch and its boundary curves are embedded into the inside of a four-sided parametric patch and its domain (see e.g. [Farin, 2002]). The second is splitting the patch into only four-sided components via specifying internal boundary curves and normal fences between them [Peters, 2019]. Both these approaches suffer from the fact that specifying these curves is a non-trivial task.

This dissertation advocates for the third option, which is the use of *genuine* multi-sided patch representations. Most such surfaces in the literature are in parametric form. They include *control point based* surfaces [Loop and DeRose, 1989; Krasauskas, 2002; Várady et al., 2016] which are calculated as a weighted average of an editable point structure; and *transfinite interpolation* surfaces [Charrot and Gregory, 1984; Kato, 1991; Sabin, 1996;

Várady et al., 2011] that are constructed as a blend of ribbon surfaces, defined along individual boundaries.

Parametric surface representations offer 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 in the case of multi-sided patches. It is also important to consider that 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.

There are various modeling tasks, where the use of *implicit multi-sided patches* has many advantages: (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 freeform 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.

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 photorealistic rendering, due to their computational efficiency in ray tracing (see e.g. [Hart, 1996; Seyb et al., 2019]).

The primary purpose of this work is to revisit the classical area of implicit surface representations, and attempt to significantly enhance the I-patch concept of [Várady et al., 2001]. In some sense, I-patches are similar to the transfinite schemes mentioned earlier, as boundary curves and cross-derivatives – in this case given in implicit 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 *ribbon* and a *bounding* surface, both given in implicit form (similarly to functional splines [Li et al., 1990]).

Structure of this thesis

The thesis is divided into two main chapters. Chapter 1 explores the limitations and the challenges of implicit surfacing, and accordingly extends former surface representations and proposes new implicit schemes. In Section 1.3 new formulae and techniques for classical representations are presented, in Section 1.4 we introduce the concept of faithful distance function for multi-sided I-patches, while Section 1.5 describes a new implicit representation, called corner I-patches.

The focus of Chapter 2 is to describe areas of application where I-patches can be utilized. In Section 2.3 two schemes for modeling complex patchworks with polyhedral frames

are presented, while in Section 2.4 a method for approximating triangular meshes with a collection of smoothly connected I-patches is detailed.

Notations

$\mathbf{P}_i, \mathbf{Q}$	2- or 3-dimensional points
$\mathbf{v}, \mathbf{n}$	2- or 3-dimensional direction vectors
$\mathbf{P}_x, \mathbf{n}_y, \mathbf{Q}_z$	Given (x, y, or z) coordinate of a point or a vector
I, R_i, B_i	Implicit functions ($\mathbb{R}^k \to \mathbb{R}$)
$I(x, y, z)$ or $R_i(\mathbf{P})$	Implicit functions evaluated in a certain point
$\widetilde{I}, \widetilde{R_i}, \widetilde{B_i}$	Implicit curves or surfaces (i.e. $\widetilde{I} = \{\mathbf{P} \in \mathbb{R}^k : I(\mathbf{P}) = 0\}$)
$\mathbf{a} \cdot \mathbf{n}, \mathbf{u} \times \mathbf{v}$	Dot and cross product of vectors
c, w_i, ω, λ	Real numbers
d, i, j, k, n	Integers

Chapter 1

New representations in multi-sided implicit surfacing

1.1 Overview

An implicit surface is the zero isosurface of a given scalar-valued function. Formally, given

$$F : \mathbb{R}^3 \to \mathbb{R}, \tag{1.1}$$

we call the set

$$\{\mathbf{P} \in \mathbb{R}^3 \,|\, F(\mathbf{P}) = 0\} \tag{1.2}$$

an implicit surface.

Similarly, in two dimensions, we call the zero set of an implicit function

$$G : \mathbb{R}^2 \to \mathbb{R}, \tag{1.3}$$

i.e.

$$\{\mathbf{P} \in \mathbb{R}^2 \,|\, G(\mathbf{P}) = 0\} \tag{1.4}$$

an implicit curve.

As indicated in the Notations, implicit surfaces and curves will be marked with $\widetilde{\ }$ – in the above cases by $\widetilde{F}$ and $\widetilde{G}$.

1.1.1 Advantages of implicit surfaces

Point classification Interrogation of points – whether they are on an implicit surface – is a simple operation. For an implicit surface $\widetilde{F}$ and a point $\mathbf{P}$, it simply means checking if $F(\mathbf{P}) = 0$.

We can also easily classify points based on which half-space of the surface they are in, as in one of the half-spaces $F(\mathbf{P}) < 0$, in the other $F(\mathbf{P}) > 0$.

Consequently, it is also simple for a closed implicit surface to filter points that are within the enclosed volume. In this thesis, the following convention will be used: inside the volume $F(\mathbf{P}) < 0$ while outside $F(\mathbf{P}) > 0$.

Boolean operations Implicit surfaces are also well-suited for executing Boolean operations on their enclosed volumes. For example

$$\Omega(F) \cup \Omega(G) \equiv \Omega(\min(F, G)) \tag{1.5}$$

$$\Omega(F) \cap \Omega(G) \equiv \Omega(\max(F, G)) \tag{1.6}$$

$$\Omega(F) \setminus \Omega(G) \equiv \Omega(\max(F, -G)), \tag{1.7}$$

where $\Omega(F)$ denotes the volume enclosed by F, i.e. $\{\mathbf{P} \in \mathbb{R}^k : F(\mathbf{P}) \leq 0\}$

Approximation An implicit equation naturally defines a distance metric from the zero isosurface, which is nonzero in other places and continuous. It is thus directly usable in least squares fitting:

$$\arg\min_{F} \sum_{\mathbf{P} \in \mathbf{PS}} (F(\mathbf{P}))^2, \tag{1.8}$$

where **PS** denotes the set of points to be approximated.

We should be careful, however, because

- $F \equiv 0$ is a trivial optimal solution, so normalization methods have to be applied to avoid it,
- the distance metric of an implicit surface can be highly distorted, which can ruin the approximation.

Ray casting Implicit surfaces can also be trivially intersected by rays when doing ray casting or ray tracing. The equation of a ray is $\mathbf{P} + t \cdot \mathbf{d}$, so calculating intersections is equivalent to solving a univariate (generally) nonlinear equation

$$F(\mathbf{P} + t \cdot \mathbf{d}) = 0. \tag{1.9}$$

In some cases, this can be solved directly. In the general case, it may be solved by using ray marching, increasing t in small steps, and searching for the first change of the sign. Then, the root can be further approximated using regula falsi, or similar numerical methods.

1.1.2 Disadvantages of implicit surfaces

Tessellation Tessellating – i.e. generating surface points and creating a mesh from them – is a much more difficult task for implicit surfaces, then for parametric surfaces. On one hand, we can generate points of parametric surfaces by substituting (2D) parametric coordinates of the domain into the equation, and create a mesh by tessellating the domain, and mapping the triangulation onto the surface. On the other hand, in the implicit case, we have to resort to more complex and less stable algorithms.

Marching cubes [Lorensen and Cline, 1987], dual contouring [Ju et al., 2002] and their variants are popular for meshing implicit surfaces. They subdivide the bounding box into small cells, then approximate local parts of the surface by polygons in each cell (MC) or dual to the cell structure (DC). However, their quality is usually worse than tessellations coming from parametric surfaces.

Texture mapping Parametric surfaces with a low distortion parameterization have a natural way to map a texture to the surface, by using the parameterization as texture coordinates.

On implicit surfaces, however, there is no such direct method, and it is difficult to create a local 2D coordinate system on the surface.

Design Control point-based surfaces are very popular in design, due to their intuitiveness to define complex surfaces. This has been made possible by bases that have the *affine invariance*, the *convex hull*, and the *partition of unity* properties; with the control points acting as coefficients on them. These include the Bernstein and B-spline bases.

For implicit surfaces, although some control data based methods have been published, like the *A-patch* [Bajaj et al., 1995], they have never become widely used, due to the lack of intuitive control for design.

1.2 Previous work

Implicit surfaces have extensive literature, related to a wide range of topics, including scattered data interpolation, approximation, blending, ray tracing, etc. For a general introduction, see the classic book on the subject [Bloomenthal et al., 1997].

1.2.1 Global methods

In this context, a global method is a scheme that generates one single surface equation to be evaluated on the whole 3D space. This can be achieved through various techniques, some of which are listed here.

Constructive solid geometry (CSG) is based on the fact that an implicit surface equation can also be a representation of the enclosed volume if interpreted as

$$\{\mathbf{P} \in \mathbb{R}^3 | F(\mathbf{P}) \leq 0\}. \tag{1.10}$$

CSG [Roth, 1982] utilizes simple Boolean operations on these volumes and uses primitives (cuboids, cylinders, prisms, spheres, etc.) as building blocks. CSG builds up a tree, with operations in the nodes and primitives in the leaves. Blends can also be added to certain operations so that the surface looks smooth where multiple primitives connect.

High-degree algebraic polynomials can also be used to represent complex shapes with a single equation. The surface can be fit to interpolate a set of constraints. The feasibility and minimal degree of this construction were explored in [Bajaj and Ihm, 1992a], which gives explicit equations, based on Bézout's theorem, for the interpolation of Hermite boundary conditions with implicit surfaces, resulting in a linear system. However, the eligible solutions often include highly curved and self-intersecting surfaces, which are difficult to filter out and complicate ray tracing, distance estimation, and similar operations. This will later be investigated in 1.3.1.2.

Neural networks are also suitable to represent implicit surfaces. Given basically any kind of input, the neural network – as a universal approximator – can learn to approximate it. They can also be augmented with additional data for training them, a recent development is Neural Radiance Fields (NeRF) [Mildenhall et al., 2021] which is capable of estimating the emitted radiance for photorealistic rendering. Their drawback is that a huge computational capacity is required, but the latest publications show rapid improvements in that regard [Müller et al., 2022].

1.2.2 Local methods

Local methods subdivide the 3D space into (regular or irregular) regions and define implicit surfaces in each of them. If geometric continuity on the region boundaries is to be preserved, they have to adhere to certain constraints. (For the definition of geometric continuity in an implicit context, see [Garrity and Warren, 1991].)

Voxel reconstruction is a computationally effective way to reconstruct an implicit surface. Storing values in each vertex of a regular grid has a high storage demand, but evaluating the surface equation is efficient. Trilinear interpolation even has hardware support on modern GPUs, although it produces only C^0-continuous surfaces.

Smooth surfaces can be acquired by tricubic interpolation [Lekien and Marsden, 2005]. There is also a list of enhanced approaches like using a modified (BCC or FCC) grid structure [Vad et al., 2014] or storing gradient values and approximating the isosurface by Taylor polynomials [Bán and Valasek, 2020].

Blend (or fillet) surfaces between two or more surfaces are the result of a basic operation in CAD systems. Implicit blending surfaces were introduced in the seminal paper of Hoffmann and Hopcroft [Hoffmann and Hopcroft, 1985], and many important problems – like unwanted bulges and discontinuities – were identified and resolved by the displacement blend [Rockwood, 1989]. 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 [Ricci, 1973] with F-rep [Pasko et al., 1995], based on R-functions, or soft blending [Dekkers et al., 2004], which satisfies a Lipschitz condition. Recent developments include gradient-based operators [Gourmel et al., 2013] and better topology control [Zanni et al., 2015].

Functional splines [Li et al., 1990; Hartmann, 1990, 2001; Zhu et al., 2008], 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 [Liming, 1947]. In our setting, these correspond to the products of the primary and bounding surfaces, respectively. The exact patch equation will be described in 1.3.1.3. The bounded blend of [Pasko et al., 2005] also defines boundaries as the intersections of primaries and a single bounding surface, while ensuring that the latter contains the entire blend. I-patches [Várady et al., 2001] are defined as the weighted sum of mixed products between ribbon and bounding surfaces. This scheme will be explained in detail in 1.3.2.

Cell-based methods divide the 3D space into not necessarily regular regions based on geometric rules. [Warren, 1992] proposes local fitting of implicit surfaces in a subdivision of 3D space (e.g. a simplicial mesh), 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 [Sederberg, 1985] and A-patches [Bajaj et al., 1995] are very similar, as they are also defined over simplexes. In 3D, the generated patches are always 3- or 4-sided. It can be proven that 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.

We remark that the use of multi-sided parametric patches to represent isosurfaces extracted on a grid was investigated in [Chávez and Rockwood, 2015]. First, a boundary curve network with Hermite interpolation is created, and then the surface is represented using the multi-sided parametric patches defined in [Várady et al., 2011].

1.3 Constructing smoothly connected implicit patchworks

The focus of this section will be the investigation of implicit surface representations to be used for creating piecewise implicit patchworks. In this context, we will consider a collection of implicit patches defined over space partitions and separated by bounding surfaces, where the patches of the neighboring cells connect along the respective bounding surface with a given geometric continuity.

In 1.3.1, a deeper analysis is given into some of the methods mentioned previously, with examples. In 1.3.2, we will show new results extending the capabilities of the I-patch representation. The contents of 1.3.1 and 1.3.2.1 are merely revising previous work (except the definition of Liming-surfaces in 1.3.1.1), while the other parts contain the details of the novel results summarized in the theses.

1.3.1 Revisiting classical methods

In this section, we investigate the direct algebraic interpolation technique by [Bajaj and Ihm, 1992a], and the functional splines, including Liming's method, in detail.

1.3.1.1 Liming's method

Liming's method for conic (quadratic) curves as outlined in [Liming, 1947] is defined as follows (see also Figure 1.1):

1. Let $L_1, L_2 : \mathbb{R}^2 \to \mathbb{R}$ be two lines in implicit form, that represent two (distinct) tangents to a desired curve
2. Let $C : \mathbb{R}^2 \to \mathbb{R}$ be a secant line to the curve, going through the two points of tangency
3. Then, $\forall \lambda \in (0;1)$: the curve

$$Q = (1-\lambda) \cdot L_1 \cdot L_2 - \lambda \cdot C^2 \tag{1.11}$$

   is a nonsingular quadratic curve, fulfilling the tangential conditions.

There is an important property of Liming's method we will need to utilize later. Although intuitive, no proof was found in the literature, so we include one.

Lemma 1. *Let $\widetilde{Q}$ be a quadratic implicit curve, $\widetilde{L_1}, \widetilde{L_2}$ two distinct tangent lines of it, $\widetilde{C}$ the secant through the points of tangency. Then, $\exists \lambda \in (0;1), \omega \in \mathbb{R}$ s.t.*

$$(1-\lambda) \cdot L_1 \cdot L_2 - \lambda \cdot C^2 \equiv \omega \cdot Q. \tag{1.12}$$

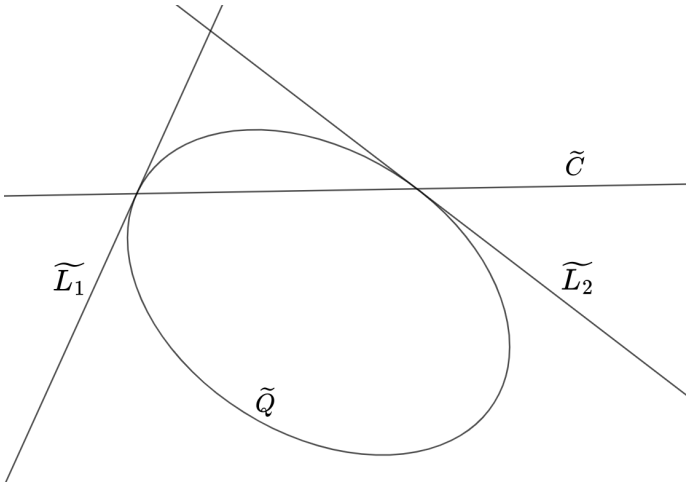


Figure 1.1: *An example of Liming's method.*

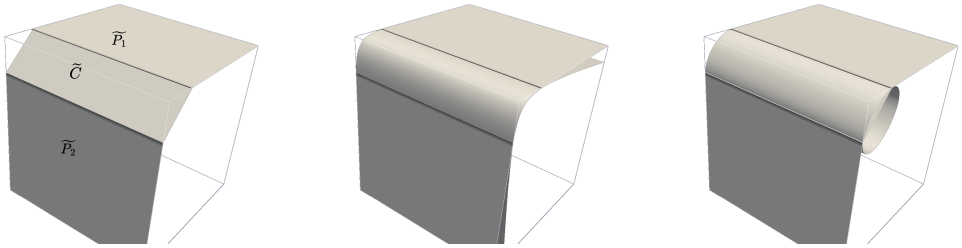


Figure 1.2: *Liming's method with surfaces.*

The proof is included in Appendix A. The meaning of this is that for any quadratic curve and any two tangent lines to it, there exists a λ value for which the original curve is reproduced.

In this dissertation, we extend Liming's method for surfaces. Consider the planes $\widetilde{P_1}$ and $\widetilde{P_2}$ and the cutting plane $\widetilde{C}$, then the surface

$$Q = (1 - \lambda) \cdot P_1 \cdot P_2 - \lambda \cdot C^2 \tag{1.13}$$

is a quadratic surface with G^1 continuity to $\widetilde{P_1}$ and $\widetilde{P_2}$. See examples with various λ values in Figure 1.2.

1.3.1.2 Hermite interpolation

The method described by Bajaj and Ihm in [Bajaj and Ihm, 1992a] deals with constructing a minimal degree algebraic surface, satisfying given constraints. These include points with an associated normal vector, or curves with a normal function along them.

An algebraic surface defined by the function $F : \mathbb{R}^3 \to \mathbb{R}$ is said to interpolate the con-

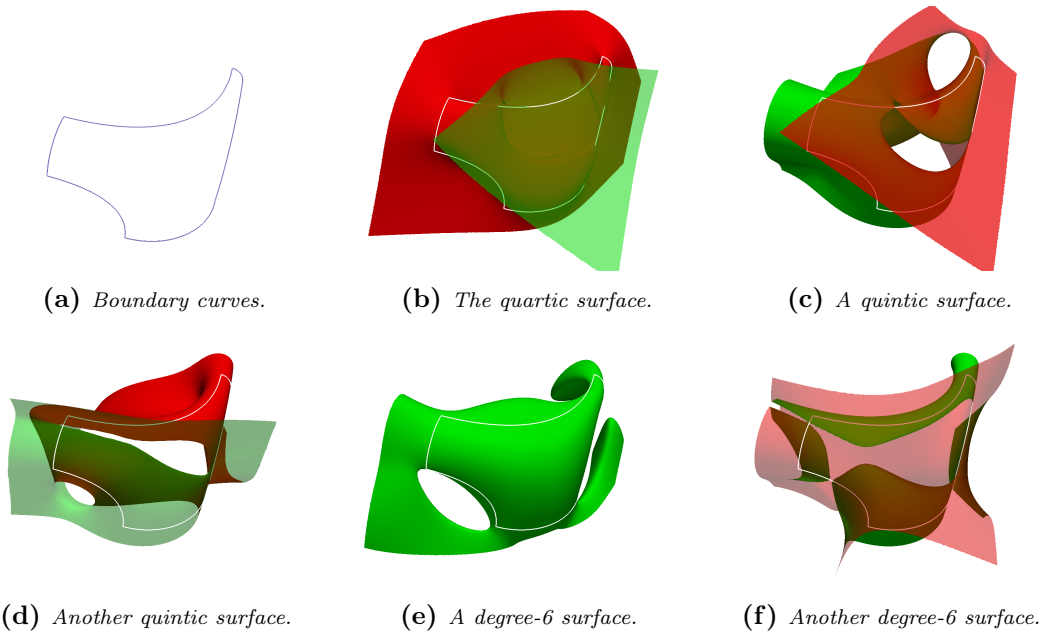
straints with C^1 continuity, if

$$F(\mathbf{P}) = 0, \tag{1.14}$$

$$\nabla F(\mathbf{P}) = \lambda \cdot \mathbf{n}, \tag{1.15}$$

for all $\mathbf{P}$ specified points or all points of specified curves, where $\mathbf{n}$ is the specified normal vector at $\mathbf{P}$ and λ is a nonzero real number.

For creating the equation system to obtain the surface coefficients, Bézout's theorem is used as described in [Bajaj and Ihm, 1992a]: a curve of degree d intersects a surface of degree n at most in $n \cdot d$ points, if the number of intersections is finite; otherwise, a component of the curve lies entirely in the surface. From this, it follows that if we choose $n \cdot d + 1$ points along the curve, then a surface fulfilling the constraints will interpolate the branches of the curve containing those points.



(a) *Boundary curves.* **(b)** *The quartic surface.* **(c)** *A quintic surface.*

(d) *Another quintic surface.* **(e)** *A degree-6 surface.* **(f)** *Another degree-6 surface.*

Figure 1.3: *Algebraic surfaces of various degrees interpolating boundary curves.*

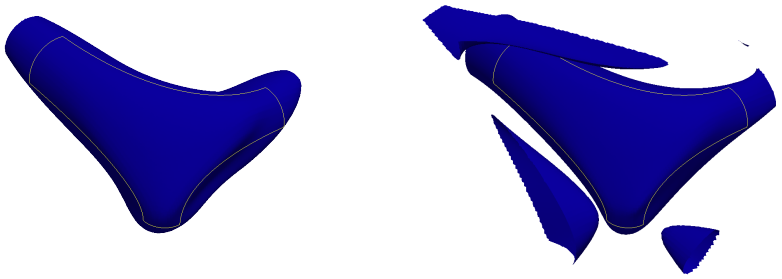
Now having reduced everything to point constraints, for all $(\mathbf{P}, \mathbf{n})$ pair we have the following equations:

$$F(\mathbf{P}) = 0, \tag{1.16}$$

$$\mathbf{n}_x \cdot \partial_y F(\mathbf{P}) - \mathbf{n}_y \cdot \partial_x F(\mathbf{P}) = 0, \tag{1.17}$$

$$\mathbf{n}_x \cdot \partial_z F(\mathbf{P}) - \mathbf{n}_z \cdot \partial_x F(\mathbf{P}) = 0. \tag{1.18}$$

Here we suppose that $\mathbf{n}_x \neq 0$. Otherwise the equation $\mathbf{n}_z \cdot \partial_y f(\mathbf{P}) - \mathbf{n}_y \cdot \partial_z f(\mathbf{P}) = 0$ needs to be used instead of one of the above-mentioned, so that the coordinate included in both tangential equations is nonzero.



(a) *A minimal degree quartic surface.* **(b)** *A quintic surface on the same boundary conditions.*

Figure 1.4: *Algebraic surfaces fit on the same boundaries.*

The number of variables is the number of free coefficients of a degree n algebraic surface, which is $\binom{n+3}{3} - 1$. In the rare case when the rank of the equation system is equal to that, we have a unique solution. If there are fewer independent equations, the system is under-determined and there are infinitely many solutions. If there are more, the system is over-determined, so there are no exact solutions with that degree. The minimum degree is the smallest degree for which the equation system is not over-determined, and the minimum degree solution(s) is (are) the element(s) of the solution set of those equations.

An example using a closed curve loop as a set of constraints is shown in Figure 1.3. The curves are circular arcs and parabolas. The normal vectors at the corners are uniquely defined by the curves, while the normal fence function is a linear blend.

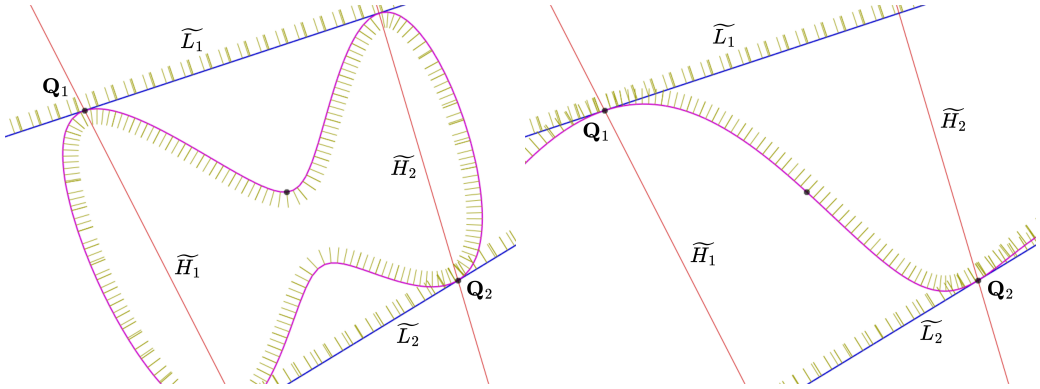
The minimal degree solution is a quartic surface which is unique (Figure 1.3b), but the resulting patch contains two components, none of them touching the whole boundary loop. Using quintic surfaces, the homogenous equation has a 6-dimensional nullspace, from which any element leads to a solution. However, in our investigation, no nice surface inside that space has been found, only surfaces like Figures 1.3c and 1.3d. Finally, a degree-6 surface configuration instantly produced a nice surface; Figure 1.3e shows a patch by taking the coefficient vector belonging to the numerically lowest singular value. It has also been easy to find incorrect degree 6 surfaces, as well (Figure 1.3f).

At the same time, unique minimal degree solutions represent in many other cases suitable surfaces, see Figure 1.4a, which is a minimal (4) degree patch, while a quintic surface is also suitable, but introduces additional branches (Figure 1.4b).

1.3.1.3 Functional spline

The formula of functional splines, as given by [Li et al., 1990] is

$$(1 - \mu) \cdot F - \mu \cdot G^k, \tag{1.19}$$



(a) *Functional spline with internal inflection.* **(b)** *Target curve with internal inflection.*

Figure 1.5: *Handling inflections.*

where F is called *base surface* and G is called *transversal surface*, $0 < \mu < 1$ is a scalar parameter, $2 \leq k$ is an integer parameter. The property of the functional spline surface is that it connects to the base surface with G^{k-1} continuity in the intersection points of the base and the transversal surface.

In a multi-sided setting, if we have a ribbon surface to interpolate for each side, the base surface should be the union of the ribbon surfaces, which is simply achieved by $F = \prod R_i$. The transversal surface is a surface that interpolates all boundary curves of the patch. In special cases, a single surface can be provided, but the general solution is to create one *bounding surface* ($\widetilde{B_i}$) for interpolating each boundary curve and use their product as the transversal surface.

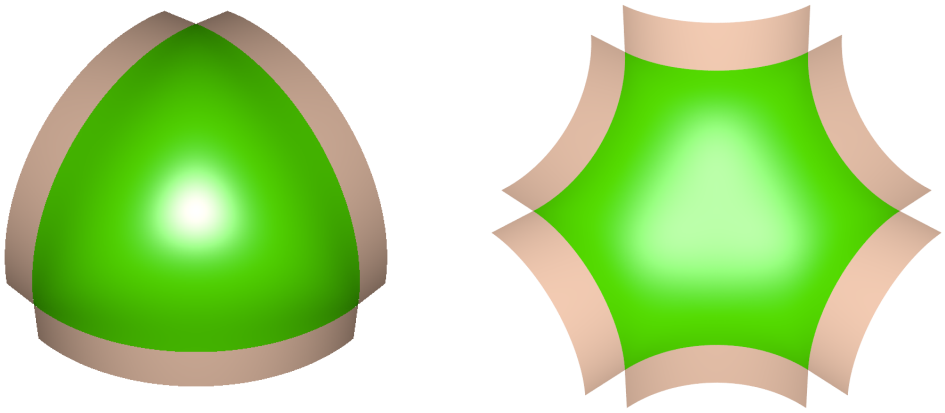
Simple 2D examples show that functional splines can only handle convex configurations, having problems with internal inflections. Take, for example, the absurd curve in Figure 1.5a, connecting $\mathbf{Q}_1$ and $\mathbf{Q}_2$ within the area spanned by $\widetilde{H_1}$ and $\widetilde{H_2}$, given by functional splines, instead of the much nicer curve in Figure 1.5b which the scheme can not reproduce. This limitation has been formalized in [Hartmann and Feng, 1993] where the exact definition of convexity in this regard and the relevant theorems can be found.

Focusing on 3D surfaces, this means that configurations like Figure 1.6a can be solved by functional splines, but ones like Figure 1.6b cannot. This problem motivated the introduction of the *symmetric parabolic functional splines*, as described in [Hartmann, 2001]. The following equation has been proposed:

$$(1 - \lambda) \cdot F_a \cdot G_b^k - \lambda \cdot F_b \cdot G_a^k = 0, \tag{1.20}$$

where $\widetilde{F_a}$ is a surface touching all the locally convex boundaries, $\widetilde{F_b}$ is a surface touching all the locally concave boundaries, $\widetilde{G_a}$ and $\widetilde{G_b}$ intersect $\widetilde{F_a}$ and $\widetilde{F_b}$ respectively at the appropriate boundaries, and $0 < \lambda < 1$ is a scalar parameter.

Using the entities R_i and B_i, let $A \subset \{1, 2, \ldots, n\}$ be an index set denoting the ribbon surfaces with the same convexity. Then $f_a = \prod_{i \in A} R_i$, $f_b = \prod_{i \notin A} R_i$, $g_a = \prod_{i \in A} B_i$,



(a) *Simple 3-sided configuration.* **(b)** *Simple 6-sided configuration.*

Figure 1.6: *Sample inputs and patches for ribbon-based implicit surfaces.*

$g_b = \prod_{i \notin A} B_i$.

More complex boundary constraints can be solved using symmetric functional splines, see Figure 1.7. In this case, the odd-numbered boundaries are in set A and the even-numbered ones in its complement.

1.3.2 Notes on I-patches

1.3.2.1 Preliminaries

Multi-sided I-patches are defined by a loop of 3D curves; each curve $\widetilde{C_i}$ is the intersection of two surfaces given in implicit form $\widetilde{C_i} = \widetilde{R_i} \bigcap \widetilde{B_i}$, where $\widetilde{R_i}$ denotes the *ribbon surface* to which the patch will smoothly connect, and $\widetilde{B_i}$ denotes the *bounding surface* that defines the intersection. The equation of I-patches in [Várady et al., 2001] was given in the following *polynomial* form:

$$I = \sum_{i=1}^{n} w_i R_i \prod_{j \neq i} B_j^2 + w \prod_{j=1}^{n} B_j^2, \tag{1.21}$$

where the w_i-s are scalar weights associated with the individual sides and w 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

$$I = w_1 R_1 B_2^2 B_3^2 + w_2 R_2 B_3^2 B_1^2 + w_3 R_3 B_1^2 B_2^2 + w B_1^2 B_2^2 B_3^2. \tag{1.22}$$

This I-patch will interpolate the boundary curves. For example, take the curve $\widetilde{C_1}$. The first term will be zero due to $R_1 = 0$, and the other three terms due to $B_1 = 0$, so for the

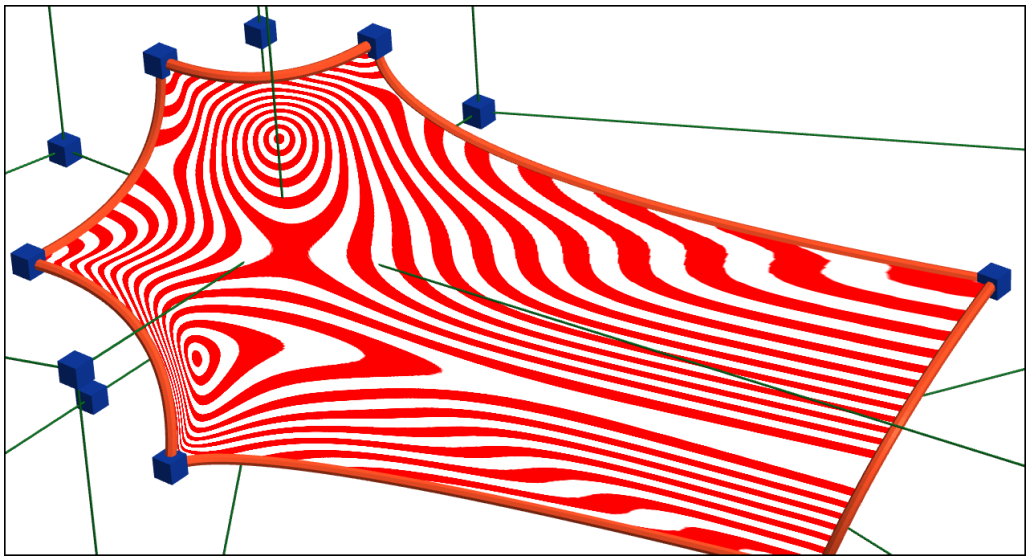


Figure 1.7: *6-sided symmetric functional spline.*

points of the boundary curve $I = 0$. This also shows that the effect of R_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 ribbons, since its gradient vector will be parallel to theirs. For example, take again boundary curve $\widetilde{C_1}$, and write the equation as $I = R_1 G + B_1^2 H$. Then

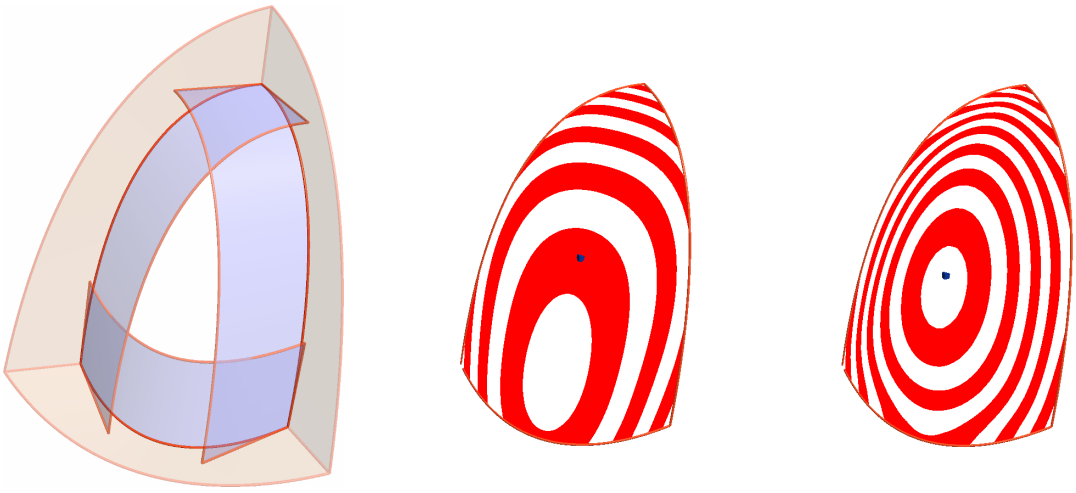
$$\nabla I = \nabla R_1 G + R_1 \nabla G + 2B_1 \nabla B_1 H + B_1^2 \nabla H = \text{const} \cdot \nabla R_1, \tag{1.23}$$

since the second, third, and fourth terms are equal to zero on $\widetilde{C_1}$. In an analogous way, G^2 or higher degree geometric continuity to the ribbons can also be achieved, if the exponent of the bounding surfaces is three or higher.

It is easy to constrain an I-patch to interpolate an arbitrary 3D point $\mathbf{P}_0$. Assuming that the weights w_i have already been fixed, then the value w can be determined by solving the equation $I(\mathbf{P}_0) = 0$, where w is the only unknown. Figure 1.8 shows two variants of a 3-sided patch interpolating two different reference points in the middle.

1.3.2.2 Handling special cases

It should be noted that at the corners of the I-patches, where two ribbons and two bounding surfaces meet, the gradient vector is always zero. Take, for example, the corner of $\widetilde{C_1}$ and $\widetilde{C_2}$, 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 ribbons 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 $\mathbf{Q}_1$ in Figure 1.9. Amongst others, this problem motivated the research into corner I-patches which will be described in 1.5.



(a) *Conic ribbon and planar bounding surfaces*

(b) *Contours*

(c) *Contours after changing the interior fullness*

Figure 1.8: *A simple 3-sided I-patch.*

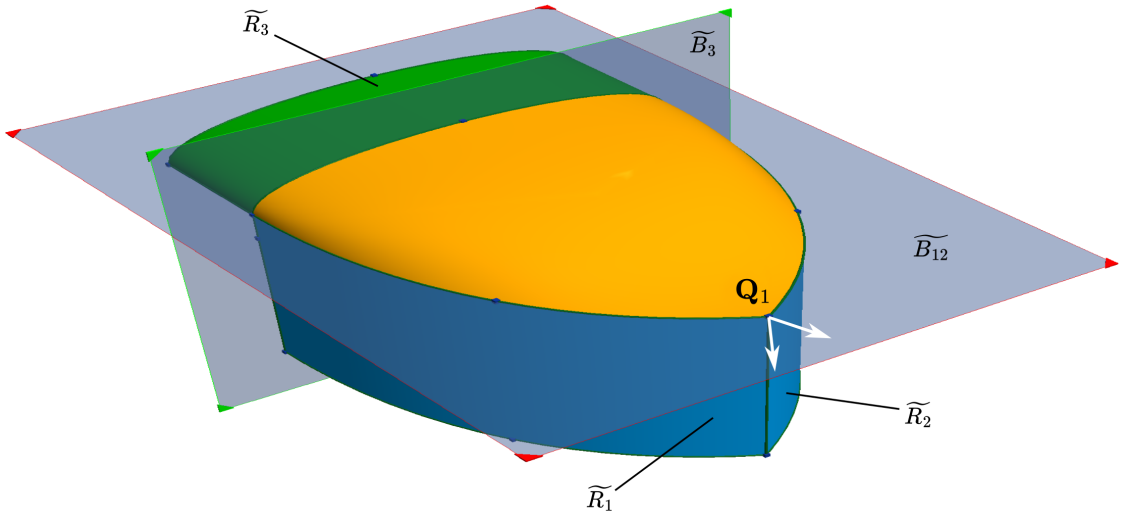


Figure 1.9: *Two ribbon surfaces intersected by the same bounding plane.*

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 boundary curves $\widetilde{R_1} \bigcap \widetilde{B_{12}}$, $\widetilde{R_2} \bigcap \widetilde{B_{12}}$, and $\widetilde{R_3} \bigcap \widetilde{B_3}$. Then we can factor out B_{12}^2 from the equation to obtain

$$I = B_{12}^2 \left(w_1 R_1 B_3^2 + w_2 R_2 B_3^2 + w_3 R_3 B_{12}^2 + w B_{12}^2 B_3^2 \right), \tag{1.24}$$

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

$$I = w_{12}(R_1 R_2) B_3^2 + w_3 R_3 B_{12}^2 + w B_{12}^2 B_3^2 \tag{1.25}$$

will yield the correct result. Note that here we can assign only a single weight to $R_1 R_2$. An example is shown in Figure 1.9. This I-patch (orange) connects to three curved ribbon surfaces; the two blue ribbons (vertical sweeps) are intersected by the same planar bounding surface.

1.3.2.3 I-segments and I-lofts

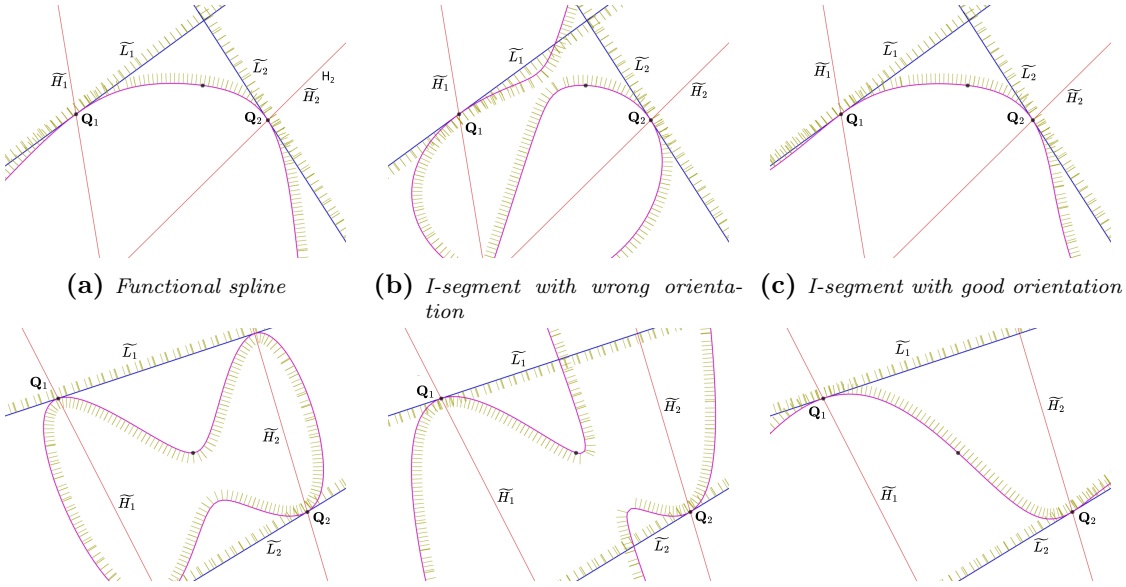
We introduce new entities related to I-patches. I-segments are the 2D counterparts of I-patches, where two implicit lines are being blended. I-lofts are 3D surfaces based on the same logic, combining just two ribbon surfaces.

I-segments, defined by quartic polynomials, are more general curves than conics defined by Liming's method. As a result, they provide more shape control and are capable of handling inflections. Given two tangential lines $\widetilde{L_i}$ and two local bounding lines $\widetilde{H_i}$ in the plane containing the tangential lines, the segment runs from one intersection point to the other. We use the following polynomial formula:

$$I = w_1 L_1 H_2^2 + w_2 L_2 H_1^2 + w H_1^2 H_2^2. \tag{1.26}$$

We can use I-segments to analyze and better understand the properties of the I-patch representation. For example, in Figure 1.10, have a deeper look into the convexity issue previously outlined concerning functional splines in 1.3.1.3.

In Figure 1.10a the functional spline curve matches the orientation of the ribbons. In 1.10b we have an I-segment with inconsistent orientation, yielding a wrong curve, however, 1.10c is consistent, and we obtain a good result. In 1.10d the functional spline curve cannot properly match the given orientations, and thus a nonsense curve is obtained. Fig. 1.10e shows an incorrectly oriented I-segment, but in 1.10f 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 orientation of the tangential lines is set in a consistent way.



(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 1.10: *2D curves represented by functional splines and I-segments.*

This shows that in the case of I-patches, the orientation of the ribbon surfaces is especially important. We will look at this in 1.3.2.6.

Let us move to 3D; I-lofts combine two tangent planes $\widetilde{P_1}$ and $\widetilde{P_2}$ at two adjacent corners with local bounding planes $\widetilde{M_1}$ and $\widetilde{M_2}$. Then

$$I = w_1 P_1 M_2^2 + w_2 P_2 M_1^2 + w M_1^2 M_2^2. \tag{1.27}$$

See an input configuration for I-lofts in Figure 1.11.

I-lofts are particularly useful for creating implicit ribbons that adhere to prescribed normal vectors in the corner points and have a smooth surface. These applications will be discussed in 2.3.1.2 and 2.4.1.

1.3.2.4 Relation between classical methods and I-patches

It is easy to see that a Liming-curve

$$Q = (1-\lambda) \cdot L_1 \cdot L_2 - \lambda \cdot C^2 \tag{1.28}$$

is a "1-sided" I-patch in 2D with $R_1 = L_1 \cdot L_2$, $B_1 = C$ and $\frac{w_1}{w} = \frac{1-\lambda}{\lambda}$, as the equation of a 1-sided I-patch is

$$I = w_1 R_1 + w B_1^2. \tag{1.29}$$

It is also true that with multi-sided 2D I-patches Liming-curves can still be reproduced with a correct setting of coefficients.

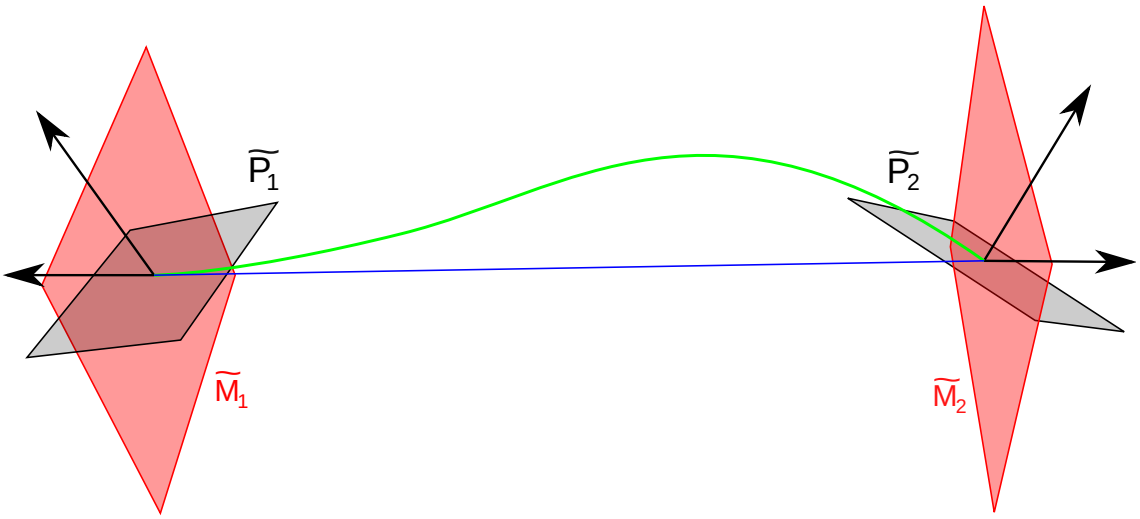


Figure 1.11: *Input configuration for an I-loft.*

Theorem 1. *Let* $\widetilde{L}_i\,(i = 1,2,3,4)$ *be four distinct tangent lines,* $\mathbf{P}_i\,(i = 1,2,3,4)$ *the points of tangency. Let* $\widetilde{C_1}$ *be a line through* $\mathbf{P}_1$ *and* $\mathbf{P}_2$*,* $\widetilde{C_2}$ *through* $\mathbf{P}_3$ *and* $\mathbf{P}_4$*. Then, the family of two-sided I-patches*

$$I_{\mathbf{w}} := w_1 \cdot L_1 \cdot L_2 \cdot C_2^2 + w_2 \cdot L_3 \cdot L_4 \cdot C_1^2 + w \cdot C_1^2 \cdot C_2^2 \tag{1.30}$$

fulfills the tangential conditions. Moreover, if there exists a quadratic curve satisfying the constrained tangents, the I-patch $\widetilde{I}$ *reproduces it with well-chosen coefficients.*

The related proof can be found in Appendix A.

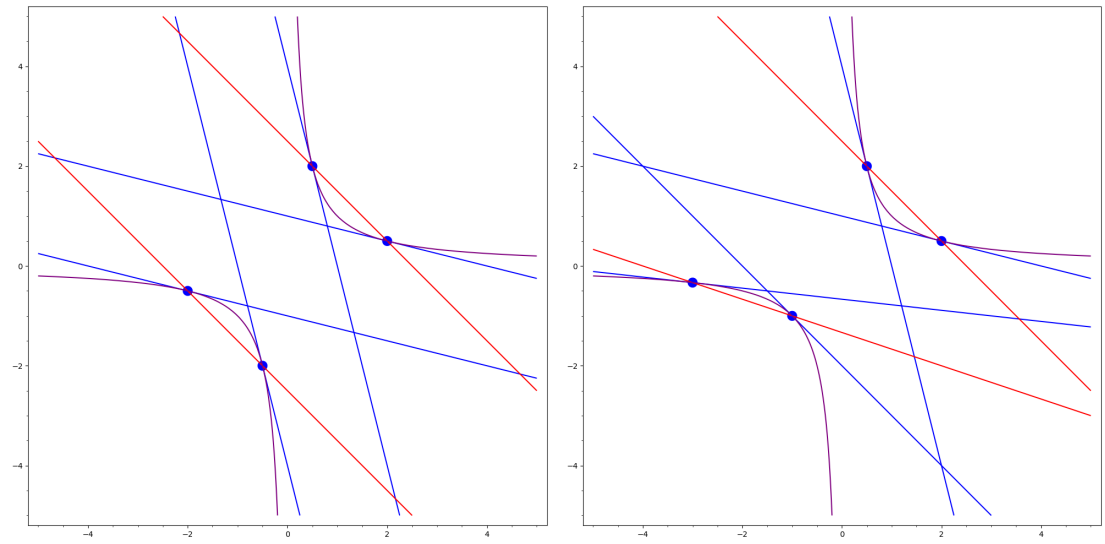
This means that we can reproduce quadratic curves with the I-patch scheme with suitable geometric entities and coefficients supplied to it. If the given tangent lines are such that a quadratic curve tangential to all of them exists, there are coefficients by means of which it can be reproduced.

In Figures 1.12a and 1.12b you can see the reproducibility of a hyperbola by choosing four points and four tangents of it, and setting in the first case $w_1 = w_2 = 4/9$, $w_0 = 32/81$. In the second example, the correct weights are $w_1 = 3/4, w_2 = 4/9$, and $w_0 = 985/1296$.

In Figure 1.13, it can be seen that by changing the weights we can achieve shapes that are similar, but different to the hyperbola.

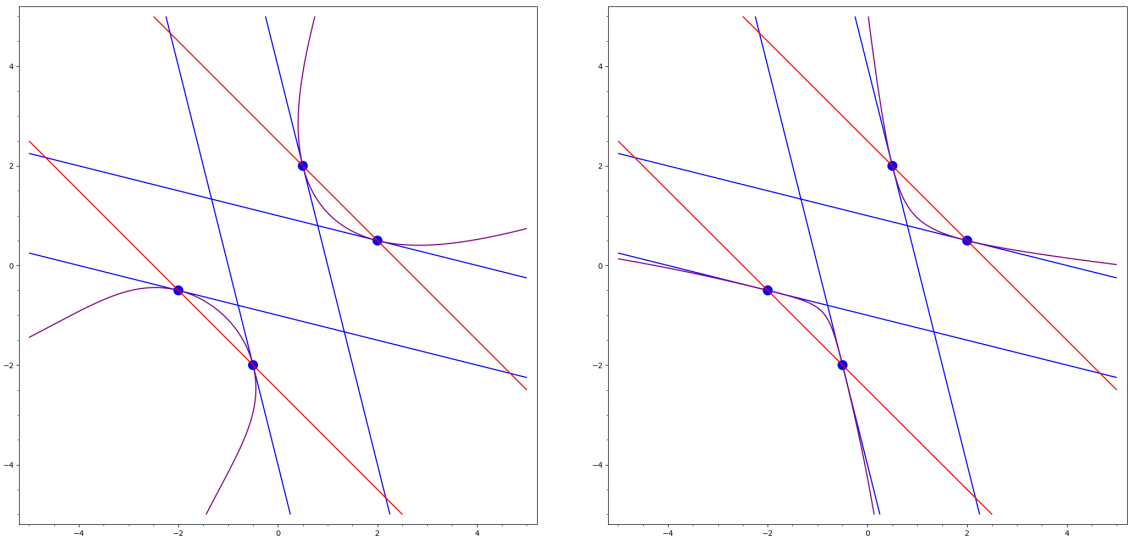
1.3.2.5 The rational formulation

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



(a) *Reproducing a hyperbola.* **(b)** *Reproducing a hyperbola with different tangents.*

Figure 1.12: *Representing a hyperbola with the I-patch scheme*
Points and tangents are blue, bounding lines are red, the resulting curve is purple.



(a) $w_1 = 0.5, w_2 = 1.0$ **(b)** $w_1 = 0.8, w_2 = 2.0$

Figure 1.13: *Modifying the weights and changing the shape of the curve.*

of the Euclidean distances from two focal points is constant, i.e.,

$$D_1(x,y) + D_2(x,y) = \|(x,y) - \mathbf{F}_1\| + \|(x,y) - \mathbf{F}_2\| = 2a, \tag{1.31}$$

where $\mathbf{F}_1 = (-c, 0)$, $\mathbf{F}_2 = (c, 0)$ and $c^2 = a^2 - b^2$.

We generalize this for I-patches and combine distances from the primitive surfaces. We take $d_i = R_i(\mathbf{P})$ 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 $d_i = w_i R_i(\mathbf{P})/B_i^2(\mathbf{P})$. 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:

$$I = D_1 + D_2 + D_3 - D_0 = \frac{w_1 R_1}{B_1^2} + \frac{w_2 R_2}{B_2^2} + \frac{w_3 R_3}{B_3^2} + w_0. \tag{1.32}$$

In fact, it is easy to show that rational-form I-patches can reproduce certain standard implicit surfaces. For example, combining three orthogonal elliptic cylinders as ribbons and the three related planar bounding surfaces, one can exactly reproduce an ellipsoid:

$$\begin{aligned} I(x,y,z) &= \frac{\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1}{z^2} + \frac{\frac{y^2}{b^2} + \frac{z^2}{c^2} - 1}{x^2} + \frac{\frac{z^2}{c^2} + \frac{x^2}{a^2} - 1}{y^2} + \left(\frac{1}{a^2} + \frac{1}{b^2} + \frac{1}{c^2}\right) \\ &= \frac{\left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1\right)(x^2y^2 + y^2z^2 + z^2x^2)}{x^2y^2z^2}. \end{aligned} \tag{1.33}$$

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 1.14 a sequence of isosurfaces $d_1 = w_1 R_1/B_1^2 = const$ is shown. The combination of this sort of *algebraic distance fields* produces the final I-patch.

An advantage of the rational formulation is that it requires fewer operations to be evaluated. While evaluating an n-sided I-patch in the original form takes $(n+1) \cdot n$ multiplication, evaluating it in the rational form needs only n multiplication and n division. This is an asymptotical difference, thus more important when the number of sides is higher.

The rational formula nicely works for the interior points of the patch; however, it is singular, i.e. degenerates to 0/0 expressions in the vicinity of the boundary curves. In order to avoid this, there we use another formula; close to $\widetilde{C_i}$ – i.e. where the divisor B_1^2 becomes so small that it hurts the precision with the used floating point representation – multiply the equation by B_i^2/w_i. For example, take curve $\widetilde{C_1}$ and use

$$I = R_1 + \left(\sum_{i=2}^{n} d_i + d_0\right) B_1^2/w_1. \tag{1.34}$$

In the points of $\widetilde{C_1}$ both terms are zero; in the close vicinity of $\widetilde{C_1}$ the quadratic expression

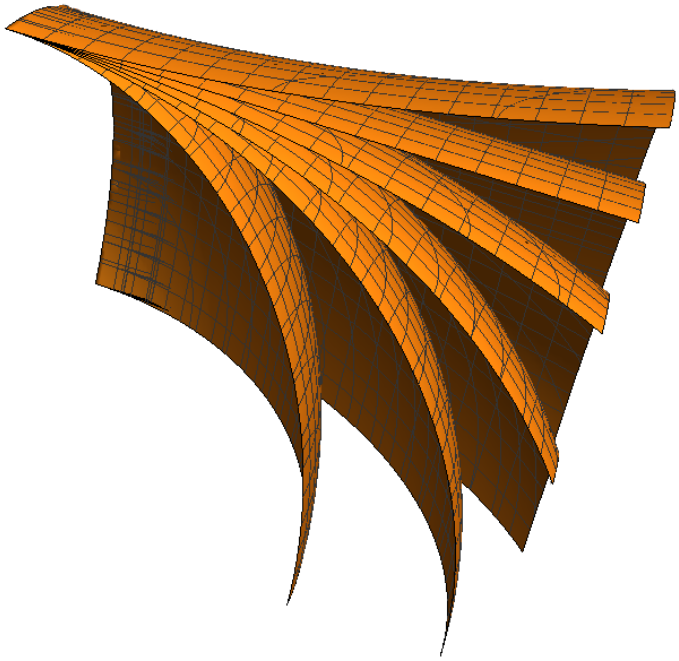


Figure 1.14: *Different isosurfaces of a weighted algebraic distance.*

B_1^2/w_1 remains small, thus the I-patch behaves as $\widetilde{R_1}$, satisfying our expectations.

As in the vicinity of the corner points, two of the terms degenerate to 0/0, assuming e.g. that we are close to both $\widetilde{B_1}$ and $\widetilde{B_2}$, the formula

$$I = R_1 B_2^2 + R_2 B_1^2 + \left(\sum_{i=3}^{n} d_i + d_0 \right) B_1^2 B_2^2 / (w_1 w_2) \tag{1.35}$$

can be used. See Figure 1.15 for an example.

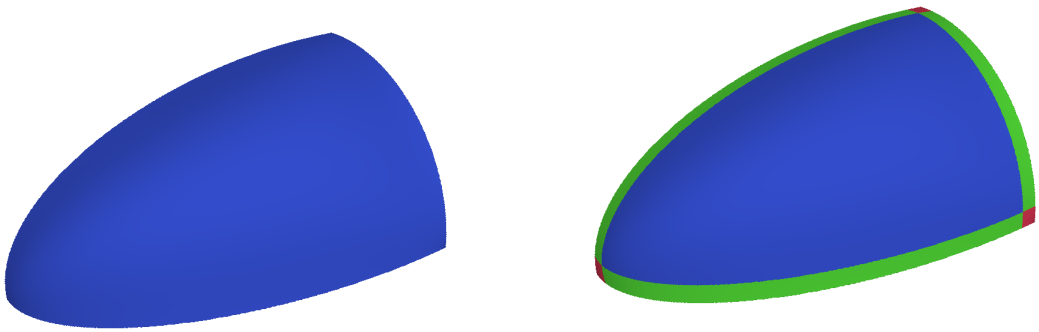
As the parts evaluated with the different formulae are connecting with G^∞ (they being a scalar function multiple of each other), the surface remains aesthetic, however, C^1 continuity is not fulfilled, the magnitude of the gradient changes in the interior.

The rational formulation helps to efficiently evaluate the surface in the interior, and set the individual weights w_i (see e.g. 2.3.1.3).

1.3.2.6 Consistent orientation

In contrast to functional splines, the orientation of the ribbons is crucial for I-patches, as it was shown during the investigations with I-segments earlier. 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.

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



(a) *Entire patch in the same color*

(b) *Patch colored based on evaluation method; blue: Eq. 1.32, green: Eq. 1.34, red: Eq. 1.35*

Figure 1.15: *Raycasted I-patch evaluated using the rational form in the interior and the modified rational forms along the boundaries.*

$\widetilde{B_i}$-s. (It needs to be checked whether this condition is satisfied, and modify some of the bounding surfaces, when needed.)

We also assume that a consistently oriented, piecewise normal vector fence exists for the boundary loop, as shown in Figure 1.16. Matching this will define the correct orientation (sign) of the ribbon surfaces. This means we have to set the orientation of all ribbons such that their gradients point in the same (or opposite) direction as the prescribed normal fence. Naturally, after the ribbons have been oriented, we can only set positive terms as the corresponding coefficients.

1.3.3 Summary

I have analyzed the properties of implicit curves and surfaces defined by boundary constraints – with an emphasis on I-patches – and have described criteria to acquire a connected, non-self-intersecting surface. For an I-patch interpolating a closed curve loop of boundary curves with a consistent normal fence, a formula is proposed to set initial coefficients automatically. Methods for handling special cases – coinciding or intersecting ribbon or bounding surfaces – were also introduced. It has been shown that the I-patch has an alternative evaluation called the rational form which computationally is more efficient in the interior, and that it can be viewed as a generalization of classical distance-based implicit blending methods.

These results were published in [Sipos et al., 2020], [Sipos et al., 2022b], and [Sipos, 2022b].

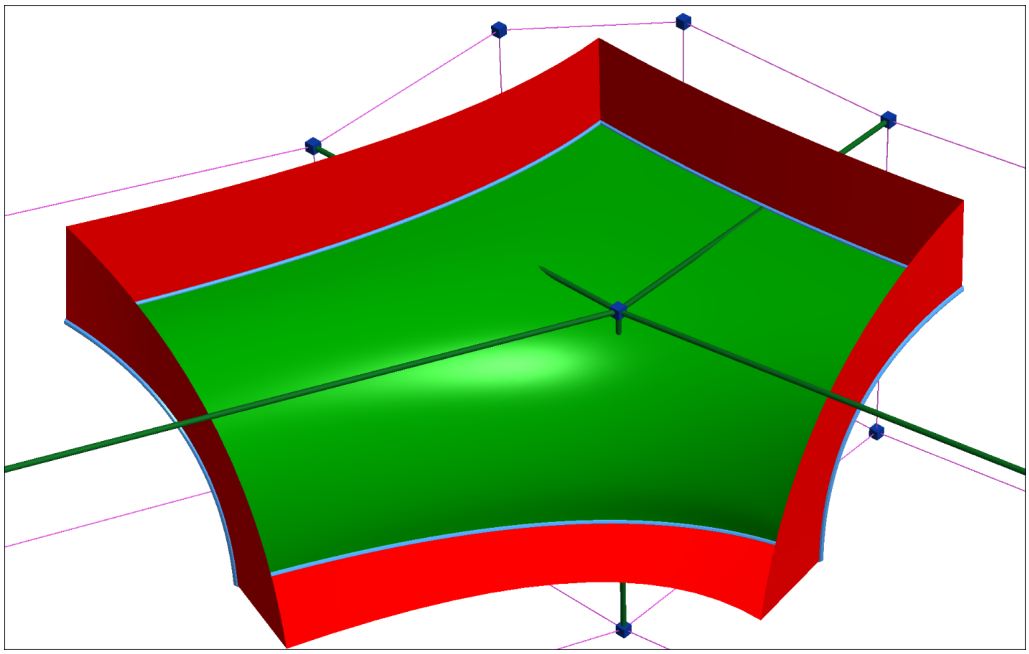


Figure 1.16: *Patch with a consistent normal fence; all ribbons must have the same (e.g. positive) half-space in the direction of the fence.*

1.4 The faithful distance field of I-patches

1.4.1 Motivation

The computation of distances from a given surface, or the creation of an offset surface by a given distance, are heavily used operations in several applications, such as design, geometric intersections and interrogations, and data approximation. One advantage of implicit surfaces is that they possess a natural distance field from their zero isosurfaces. Unfortunately, algebraic distances, even when multiplied by a carefully chosen scalar factor, do not yield Euclidean distances, except for planes.

In the case of I-patches, neither the original algebraic nor the rational form yields a suitably close approximation of the geometric distance from the surface, both have high variation.

A frequently applied, practical method for enhancing distance fields is to normalize the implicit equation by the norm of its gradient [Taubin, 1994]. This is a good solution to obtain approximate Euclidean distances in the vicinity of the zero isosurface, but these expressions contain square roots and may exhibit singularities. They also deviate from the correct distance when we move farther away from the implicit surface.

1.4.2 Formulation

We propose a normalization for I-patches that produces a good approximation of the Euclidean distance field, not only in the vicinity of the surface, but in a much wider range.

The literature often uses the terminology "faithful" for such distances [Lukács et al., 1998], and we also retain this adjective.

Using the notation $A_i = w_i/B_i^2$, we can formulate the rational equation of the I-patch as a weighted sum of ribbon surfaces $\widetilde{R_i}$, multiplied by blending functions:

$$I = \sum_{i=1}^{n} R_i \cdot A_i - w_0. \tag{1.36}$$

We derive a faithful normalization by dividing with the sum of the blending functions:

$$\hat{I} = \frac{\sum_{i=1}^{n} R_i \cdot A_i - w_0}{\sum_{i=1}^{n} A_i}. \tag{1.37}$$

Examining an algebraic offset of the normalized I-patch, it was recognized that

$$\hat{I} - \delta = \frac{\sum_{i=1}^{n} (R_i - \delta) \cdot A_i - w_0}{\sum_{i=1}^{n} A_i}. \tag{1.38}$$

In other words, the offset of the normalized I-patch is also a normalized I-patch, created from ribbon offsets and blended by the same A_i functions (i.e., the same bounding surfaces).

Note that the same principle also works with the polynomial form of the I-patch:

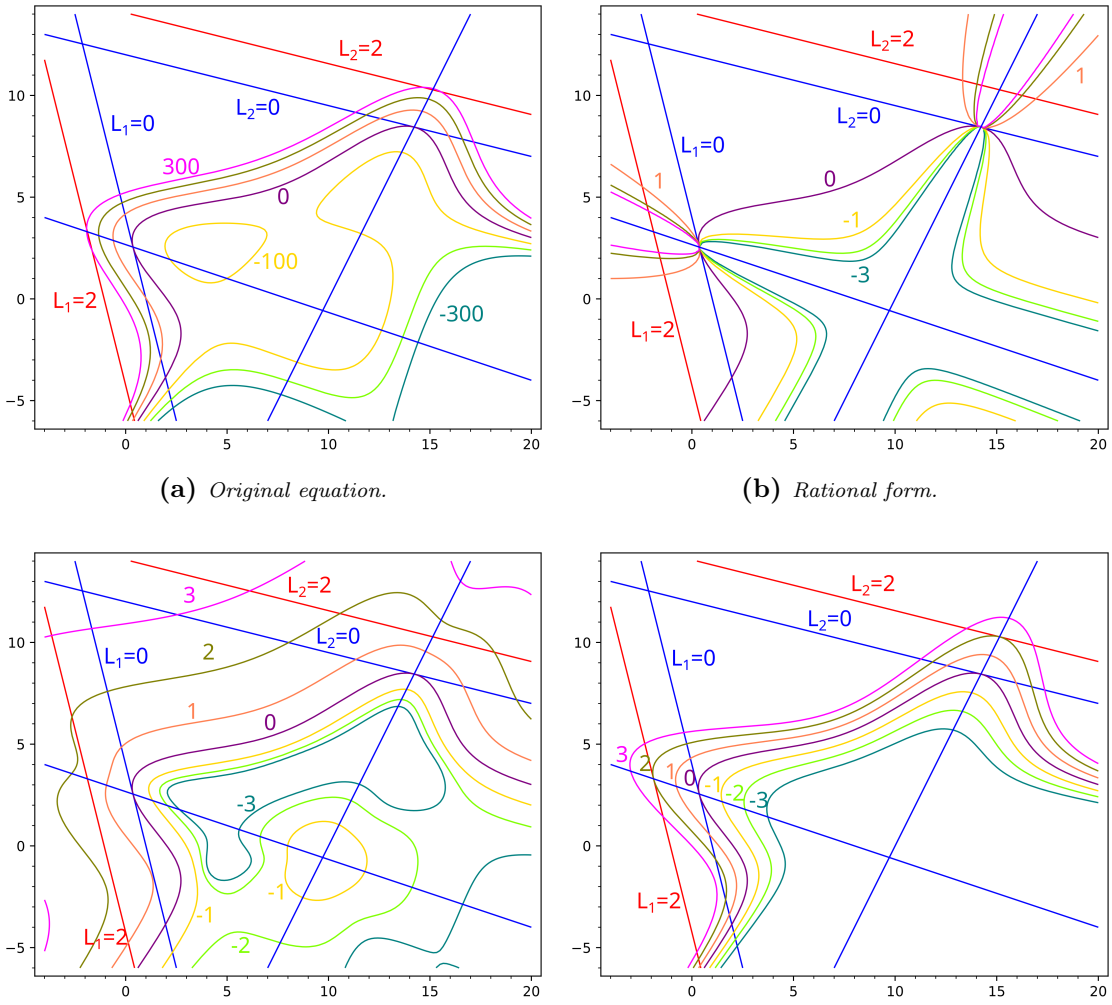
$$\hat{I} = \frac{\sum_{i=1}^{n} w_i R_i \prod_{j \neq i} B_j^2 - w_0 \prod_{i=1}^{n} B_i^2}{\sum_{i=1}^{n} w_i \prod_{j \neq i} B_j^2}. \tag{1.39}$$

This is a more complex, but equivalent equation, which can be evaluated on the boundary, as well.

1.4.3 Analysis

First, let us demonstrate this concept in 2D using an I-segment $\widetilde{S}$, which is a planar implicit curve, that blends together two lines $\widetilde{L_1}$ and $\widetilde{L_2}$ (Fig. 1.17). We wish to compute an offset of the I-segment that smoothly connects the accurately displaced offset lines $\widetilde{L_1 - d}$ and $\widetilde{L_2 - d}$, and expect to obtain a good distance field between them. Figure 1.17a shows the distribution of the original (algebraic) distances, Figure 1.17b presents the offsets defined by the rational form, and Figure 1.17c shows the distance field after gradient normalization, in all of which one can observe the uneven and unproportional distribution of the offset curves. Our proposed normalization in Figure 1.17d shows a faithful distance field.

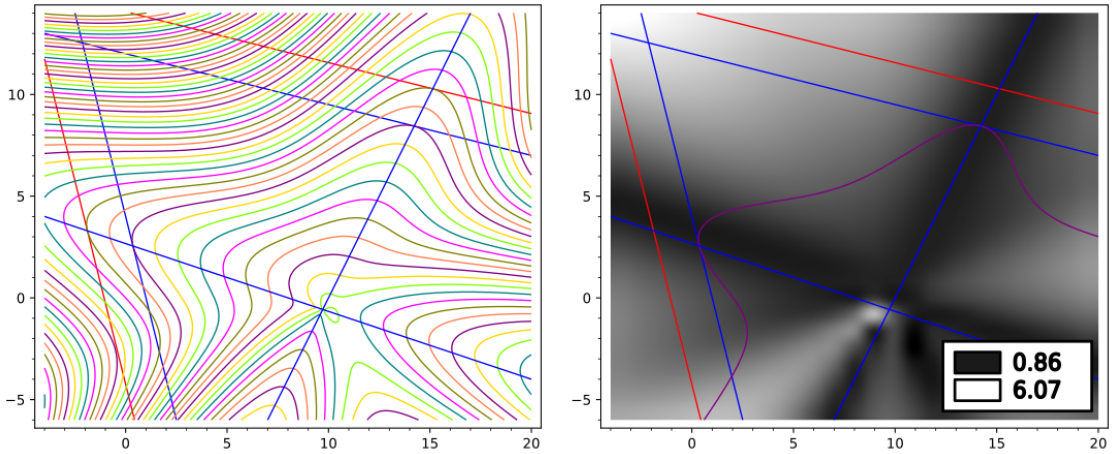
The faithful distance is only usable in the positive half-spaces of the boundings. In other parts of the space, branches of the I-segment or I-patch may exist, thus the distance function can give nonsense results. The previously analyzed I-segment and many of its offsets can be seen in Figure 1.18a. In Figure 1.18b, a quantitative analysis of the gradient lengths of the faithful distance function can be seen.



(a) *Original equation.* **(b)** *Rational form.*

(c) *Gradient normalization.* **(d)** *Faithful normalization.*

Figure 1.17: *Offsets of an I-segment using different forms of equation.*



(a) *Numerous faithful offsets of an I-segment.* **(b)** *Gradient magnitude of the faithful distance field.*

Figure 1.18: *Analysis of the faithful distance field of an I-segment.*

The faithfulness of the formulation in 3D depends on the distance field of the ribbons. For this reason, it is recommended to use ribbons that either have an evaluable Euclidean

distance field, or to which the faithful approach can be used – like I-lofts. In the latter case, the equation of the faithful I-loft ribbon is

$$\hat{R}_i = \frac{w_{i,1}P_{i,1}M_{i,2}^2 + w_{i,2}P_{i,2}M_{i,1}^2 - w_{i,0}M_{i,1}^2M_{i,2}^2}{w_{i,1}M_{i,2}^2 + w_{i,2}M_{i,1}^2}, \tag{1.40}$$

whereas the equation of the patch is

$$\hat{I} = \frac{\sum_{i=1}^n w_i\hat{R}_i/B_i^2 - w_0}{\sum_{i=1}^n w_i/B_i^2}, \tag{1.41}$$

and substituting results in

$$\hat{I} = \sum_{i=1}^n \frac{w_i/B_i^2 \cdot (w_{i,1}P_{i,1}M_{i,2}^2 + w_{i,2}P_{i,2}M_{i,1}^2 - w_{i,0}M_{i,1}^2M_{i,2}^2)}{(w_{i,1}M_{i,2}^2 + w_{i,2}M_{i,1}^2)\cdot\sum_{i=1}^n w_i/B_i^2} - \frac{w_0}{\sum_{i=1}^n w_i/B_i^2}. \tag{1.42}$$

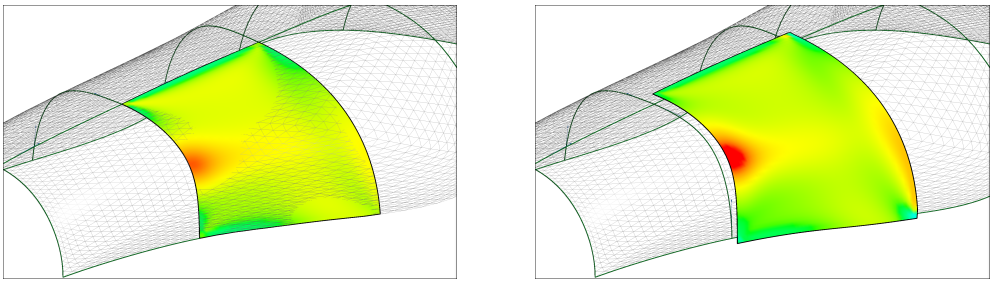
With regard to the $P_{i,j}$-s, we can observe that the same property that was outlined in Eq. 1.38 holds, i.e.

$$\begin{aligned}\sum_{i=1}^n \frac{w_i/B_i^2 \cdot (w_{i,1}(P_{i,1}-\delta)M_{i,2}^2 + w_{i,2}(P_{i,2}-\delta)M_{i,1}^2 - w_{i,0}M_{i,1}^2M_{i,2}^2)}{(w_{i,1}M_{i,2}^2 + w_{i,2}M_{i,1}^2)\cdot\sum_{i=1}^n w_i/B_i^2} - \frac{w_0}{\sum_{i=1}^n w_i/B_i^2} = \\ = \sum_{i=1}^n \frac{w_i/B_i^2 \cdot (w_{i,1}P_{i,1}M_{i,2}^2 + w_{i,2}P_{i,2}M_{i,1}^2 - w_{i,0}M_{i,1}^2M_{i,2}^2)}{(w_{i,1}M_{i,2}^2 + w_{i,2}M_{i,1}^2)\cdot\sum_{i=1}^n w_i/B_i^2} - \frac{w_0}{\sum_{i=1}^n w_i/B_i^2} - \\ - \sum_{i=1}^n \frac{w_i/B_i^2 \cdot (w_{i,1}\delta M_{i,2}^2 + w_{i,2}\delta M_{i,1}^2)}{(w_{i,1}M_{i,2}^2 + w_{i,2}M_{i,1}^2)\cdot\sum_{i=1}^n w_i/B_i^2} = \hat{I} - \delta\end{aligned} \tag{1.43}$$

As the $\widetilde{P_{i,j}}$-s are planes (thus an Euclidean distance function is available), the I-patch still behaves in a faithful manner with respect to them, although the weight functions are more complex.

In Figure 1.19 an I-patch is shown with its offset, using mean curvature maps. Figures 1.19a and 1.19b show the patch and its offset isosurface, and Figure 1.20 shows four superimposed offset patches with $d = -5$, 0, 5 and 10.

Using the faithful distance metric, offsets of I-patches or I-segments can be directly computed yielding reasonably good approximations of Euclidean distances. Approximating data points using I-patches or I-lofts becomes easy and computationally efficient since we only need to substitute into the above normalized equations, and optimize accordingly.



(a) *Original surface.* **(b)** *Offset surface.*

Figure 1.19: *Offsetting an I-patch.*

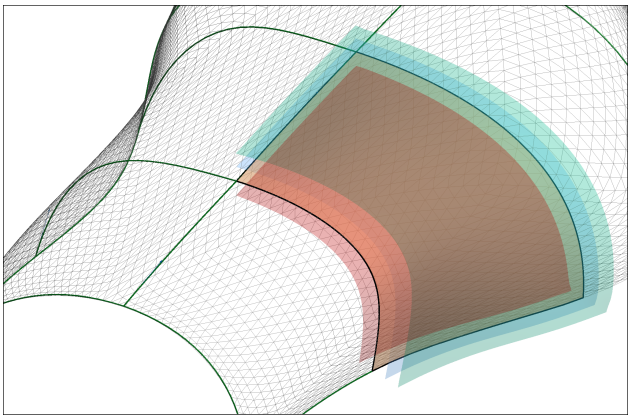
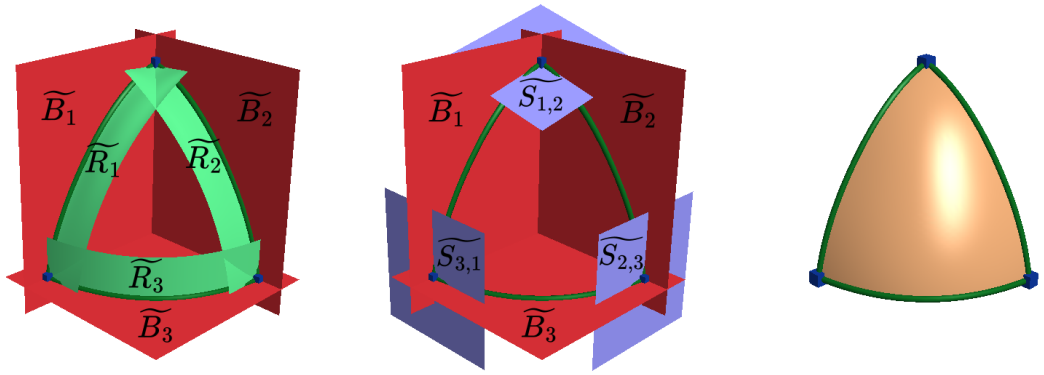


Figure 1.20: *Multiple offsets of an I-patch.*

1.4.4 Summary

I have proposed a "faithful" distance metric for I-patches, which inherits the distance from the ribbon surfaces that define the patch along the boundaries. The faithful distance metric is a better approximation of the Euclidean distance, than the algebraic function and in a much greater vicinity of the surface than the widely used gradient normalized formulations. This result was published in [Sipos et al., 2022a].



(a) *Input (ribbons and bounding surfaces) for an* ***I-patch****.*

(b) *Input (corners and bounding surfaces) for a* ***corner I-patch****.*

(c) *A resulting patch.*

Figure 1.21: *I-patch and corner I-patch.*

1.5 The corner I-patch representation

We have seen some limitations of I-patches earlier: their singularity in their corner points and the difficulty of creating proper ribbon and bounding surfaces in all cases. This motivated my research to create implicit patches that 1) have a fixed bounding surface structure, enclosing finite volume spaces, 2) use simpler geometric entities that the often complex ribbon surfaces, and 3) are non-singular along the surface.

To achieve this, we propose combining corner interpolants instead of side interpolants, this concept is also used in case of some parametric surface representations in the literature.

1.5.1 Basic equation

A corner I-patch is composed of corner interpolants $\widetilde{S_{1,2}}, \widetilde{S_{2,3}}, \ldots, \widetilde{S_{n,1}}$ and bounding surfaces $\widetilde{B_1}, \widetilde{B_2}, \ldots, \widetilde{B_n}$ (neither of them coincides with another one), such that $\widetilde{S_{i,i+1}}$ denotes the corner interpolant between the ith and the $(i+1)$st boundaries.

Then, the equation of the corner I-patch is

$$I = \sum_{i=1}^{n} \left(w_{i,i+1} \cdot S_{i,i+1} \cdot \prod_{\substack{j=1 \\ j \neq i, j \neq i+1}}^{n} B_j^2 \right) + \sum_{i=1}^{n} \left(w_i \cdot \prod_{\substack{j=1 \\ j \neq i}}^{n} B_j^2 \right) + w \prod_{i=1}^{n} B_i^2, \tag{1.44}$$

where the $w_{i,i+1}$ scalars can be merged into $S_{i,i+1}$, as multiplying with a nonzero number does not change the implicit isosurface, only its distance metric. See an example of corner interpolants and bounding surfaces in Figure 1.21b, in comparison with the input used for an I-patch in Figure 1.21a.

Some important properties of this representation are:

- In each corner, the patch connects with G^1 continuity to the corner interpolants. (This means that the gradient vectors of the surface have the same direction as the gradients of the interpolants there.)
- Along the ith boundary, the shape of the surface does not depend on w and w_j for $j \neq i$.

The $w_{i,i+1}$ coefficients will be called corner coefficients, w_i-s are the side coefficients and w is the central coefficient.

The w_i and w parameters can be set in a process similar to I-patches forcing the patch to interpolate one point on each boundary and one point in the interior of the patch. As the shape of the surface on the ith boundary depends only on w_i, each of those can be set separately, and finally, w can be set to interpolate an interior point, i.e.:

$$w_i := -\frac{S_{i-1,i}(\mathbf{P}_i) \cdot B^2_{i+1}(\mathbf{P}_i) + S_{i,i+1}(\mathbf{P}_i) \cdot B^2_{i-1}(\mathbf{P}_i)}{B^2_{i-1}(\mathbf{P}_i) \cdot B^2_{i+1}(\mathbf{P}_i)}, \tag{1.45}$$

and

$$w := -\frac{\sum_{i=1}^{n} \left(S_{i,i+1}(\mathbf{P}_0) \cdot \prod_{\substack{j=1 \\ j\neq i, j\neq i+1}}^{n} B^2_j(\mathbf{P}_0) \right) + \sum_{i=1}^{n} \left(w_i \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n} B^2_j(\mathbf{P}_0) \right)}{\prod_{i=1}^{n} B^2_i(\mathbf{P}_0)}, \tag{1.46}$$

where $\mathbf{P}_i$ is a point on the ith side, $\mathbf{P}_0$ is a point in the interior to interpolate. The parameters can be computed in this order, i.e. first all w_i, then w.

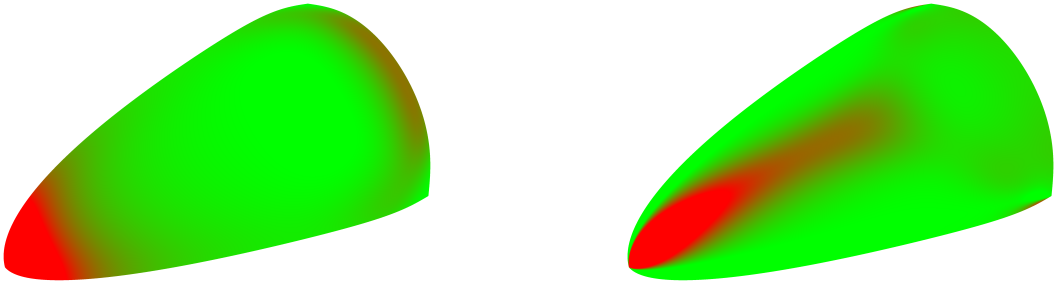
1.5.2 Comparison to I-patches

A disadvantage of I-patches, discussed earlier, is that their gradient is a zero vector in the corner points. This may lead to poor surface quality and generally should be avoided. However, the gradient of corner I-patches is the gradient of the corner interpolant times a nonzero number.

The corner I-patch along the ith boundary connects smoothly to the implicit surface represented by the equation

$$S_{i-1,i} \cdot B^2_{i+1} + S_{i,i+1} \cdot B^2_{i-1} + w_i \cdot B^2_{i-1} \cdot B^2_{i+1}. \tag{1.47}$$

This represents an I-loft (1.3.2.3). Corner I-patches are thus similar (but not equivalent) to I-patches defined by ribbons that are I-lofts. This is because the equation of the I-patch



(a) *Corner I-patch* **(b)** *I-patch*

Figure 1.22: *Corner- and regular I-patch approximating an ellipsoid octant with semi-axes 2, 1 and 1; colored using Gaussian curvature; red: +3, green: 0, blue: −3*

defined by the ribbons in Equation 1.47 would be

$$I = \sum_{i=1}^{n} \left(S_{i,i+1}(B_{i+1}^2 B_{i-1}^2 + B_i^2 B_{i-2}^2) \prod_{\substack{j=1 \\ j\neq i, j\neq i+1}}^{n} B_j^2 \right) + \\ + \sum_{i=1}^{n} \left(w_i B_{i-1}^2 B_{i+1}^2 \prod_{\substack{j=1 \\ j\neq i}}^{n} B_j^2 \right) + w \prod_{i=1}^{n} B_i^2, \tag{1.48}$$

which is not equivalent to Equation 1.44. Indeed, the factor $(B_{i+1}^2 B_{i-1}^2 + B_i^2 B_{i-2}^2)$ causes the gradient of the I-patch to be zero at the corner points. Accordingly, corner I-patches have a lower degree of $2n$, as opposed to the $2n + 2$ for this kind of I-patches.

A comparison of a corner I-patch and an I-patch with the same boundary curves can be seen in Figure 1.22. The patches interpolate the corner points of an ellipsoid octant with semi-axes 2, 1, and 1, but their boundary curves are – as discussed above – I-segments. Colored based on Gaussian curvature, the artifacts near the corners of the I-patch can be visually analyzed.

1.5.3 Faithful distance

A faithful distance function can be defined for the corner I-patch. As seen in e.g. Eq. 1.37, the divisor is the sum of the factors multiplied with the tangential (there ribbon, here corner) surfaces:

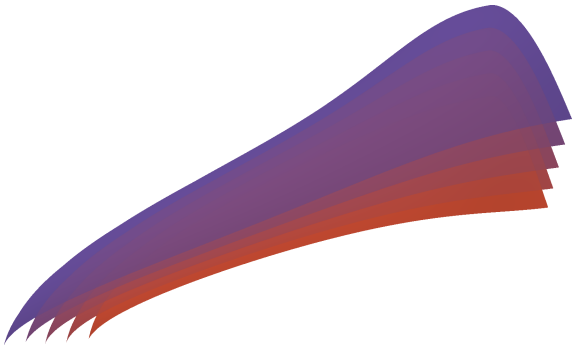


Figure 1.23: *Offsets of a faithful corner I-patch (same as in Figure 1.22a, offset values:* $-0.2, -0.1, 0, 0.1, 0.2$*)*

$$\hat{I} = \frac{\displaystyle\sum_{i=1}^{n} \left(w_{i,i+1} \cdot S_{i,i+1} \cdot \prod_{\substack{j=1 \\ j\neq i, j\neq i+1}}^{n} B_j^2 \right) + \sum_{i=1}^{n} \left(w_i \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n} B_j^2 \right) + w \prod_{i=1}^{n} B_i^2}{\displaystyle\sum_{i=1}^{n} \left(w_{i,i+1} \cdot \prod_{\substack{j=1 \\ j\neq i, j\neq i+1}}^{n} B_j^2 \right)}. \tag{1.49}$$

It is straightforward to verify that the $\widetilde{(\hat{I} - \delta)}$ isosurface is equivalent to the faithful corner I-patch defined by the corners $\widetilde{(S_{i,i+1} - \delta)}$.

In Figure 1.23, five superimposed offset patches of the corner I-patch in Figure 1.22a can be seen.

1.5.4 Corner I-patch with multiple loops

1.5.4.1 Motivation and equation

In some cases, an isosurface must be represented by several disjoint surface elements. In Marching Cubes [Lorensen and Cline, 1987] for example, 7 of the 15 basic configurations result in a surface represented by more than one polygon. These surface elements could be represented separately, but in an implicit representation, it is advantageous to have the same implicit function on a well-defined 3D volume, otherwise, the piecewise implicit function would likely have discontinuities.

Fortunately, corner I-patches are capable of achieving this with a little modification. Consider m separate boundary loops where the lth of those is n_l-sided and for each of them the previously defined corner interpolants $\widetilde{S^l_{1,2}}, \widetilde{S^l_{2,3}}, \ldots, \widetilde{S^l_{n_l,1}}$ and the bounding surfaces

(a) *Two 3-sided loops.* **(b)** *A 3 and a 4-sided loops.*

Figure 1.24: *Multiloop patches.*

$\widetilde{B_1^l}, \widetilde{B_2^l}, \ldots, \widetilde{B_{n_l}^l}$. Then the new equation is

$$I = \sum_{l=1}^{m} \sum_{i=1}^{n_l} \left(w_{l,i,i+1} \cdot S_{i,i+1}^l \cdot \prod_{k=1}^{m} \prod_{\substack{j=1 \\ k \neq l \vee (j \neq i \wedge j \neq i+1)}}^{n_k} (B_j^k)^2 \right) + \\ + \sum_{l=1}^{m} \sum_{i=1}^{n_l} \left(w_{l,i} \cdot \prod_{k=1}^{m} \prod_{\substack{j=1 \\ k \neq l \vee j \neq i}}^{n_k} (B_j^k)^2 \right) + w \prod_{l=1}^{m} \prod_{i=1}^{n_l} (B_i^l)^2. \tag{1.50}$$

The meaning of this is that each corner interpolant is multiplied by all the bounding surfaces, except the two ones beside it, corresponding to the next and previous boundaries of the patch. A simple multiloop surface can be seen in Figure 1.24a, with two loops each composed of three corners.

1.5.4.2 Handling coinciding bounding surfaces

In a multiloop setting, especially if working in a grid of cubes, some bounding surfaces will likely coincide. Consider, for example, the configuration in Figure 1.24b (a configuration of Marching Cubes), where one of the boundings for the 3-sided loop coincides with one of those of the 4-sided loop. The problem is that when two bounding surfaces coincide, they will be in all the members of the weighted sum and thus can be factored out from the equation.

With regard to I-patches, a modified equation for this difficult case has been proposed in 1.3.2.2 and in [Sipos et al., 2020], however, it takes advantage of the 1-to-1 relation between ribbons and bounding surfaces which is not applicable to corner patches.

The proposed solution is as follows. When computing the product of the bounding surfaces not neighboring the respective corner, also omit those that coincide with one of the neighboring ones.

The related equation is:

$$I = \sum_{l=1}^{m} \sum_{i=1}^{n_l} \left(w_{l,i,i+1} \cdot S_{i,i+1}^{l} \cdot \prod_{k=1}^{m} \prod_{\substack{j=1 \\ B_j^k \neq B_i^l, B_j^k \neq B_{i-1}^l}}^{n_k} (B_j^k)^2 \right) + \\ + \sum_{l=1}^{m} \sum_{i=1}^{n_l} \left(w_{l,i} \cdot \prod_{k=1}^{m} \prod_{\substack{j=1 \\ B_j^k \neq B_i^l}}^{n_k} (B_j^k)^2 \right) + w \prod_{l=1}^{m} \prod_{i=1}^{n_l} (B_i^l)^2. \tag{1.51}$$

This means that each corner is multiplied with the product of all bounding surfaces (regardless of which loop they are in) unless they coincide with one of its neighbors. Side components for a given side are the product of all bounding surfaces, other than those coinciding with the bounding surface representing that side. This formulation keeps the properties highlighted regarding the original equation (G^1 continuity to corner interpolants, and sides being only affected by the corresponding coefficient).

A practical implementation for evaluating this is to store all bounding surfaces in an array, and only store indices for them in the loops. When computing the products, we only have to check for the non-equality of the indices.

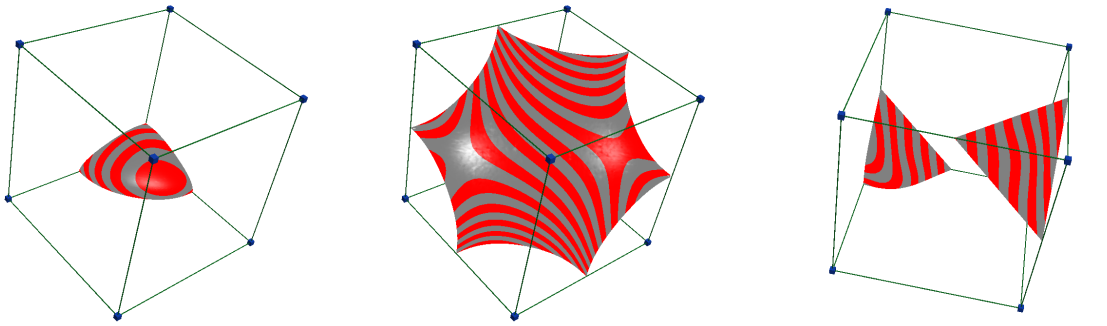
1.5.5 Discussion

1.5.5.1 Limitations

When connecting neighboring patches with geometric continuity, we need them to coincide at their common boundary in both a positional and a differential sense. As the corner I-patch along $\widetilde{B_i}$ connects to the surface defined by Equation 1.47, the patch on the other side of $\widetilde{B_i}$ also has to connect to it. This, however, only happens if $\widetilde{B_{i-1}}$ and $\widetilde{B_{i+1}}$ are identical to the corresponding bounding surfaces for the other patch.

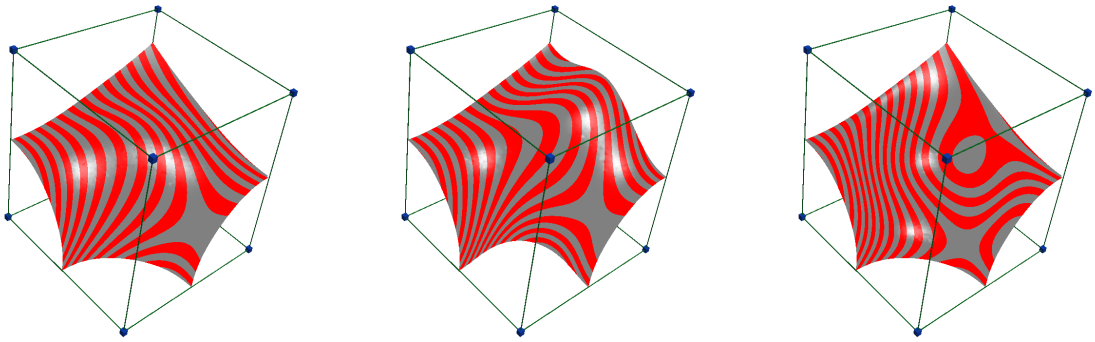
This is not easily fulfilled when creating a general topology patchwork, but it is straightforward if the space is subdivided by planes, creating finite-volume cells. All edges of the cell structure are the intersection line of two planes, corner points are points on these edges. Any such cell structure works with corner I-patches. The simplest of those is of course a regular grid of cubes.

When used in regular cell structures, the $\widetilde{S_{i,j}}$ and $\widetilde{B_i}$ surfaces are all planes. Thus, the patch itself is a polynomial surface, with a degree of twice the number of sides (see Equation 1.44).



(a) *A 3-sided patch.* **(b)** *A 6-sided patch.* **(c)** *A patch consisting of two disjoint components.*

Figure 1.25: *Corner I-patches inside the unit cube.*



(a) *Base patch.* **(b)** *Changing boundaries.* **(c)** *Changing the interior.*

Figure 1.26: *Corner I-patches with the same corner interpolants and different coefficients.*

1.5.5.2 Examples

In Figures 1.25 and 1.26 corner patches are defined inside the unit cube. Figure 1.25a is a 3-sided surface near a corner of the cube. Figure 1.25b is a 6-sided patch that intersects all faces of the cube.

In Figure 1.26, three patches with the same corners but different coefficients can be seen. They are set using the algorithm presented in 1.5.1, i.e. on each side and in the interior one point is fixed, and respective coefficients are calculated from them. Between Figure 1.26a and 1.26b, two side points; between Figure 1.26a and 1.26c, the interior point is changed. The numerical data for these patches can be found in Table 1.1.

A corner I-patch with a non-planar corner interpolant can be examined in Figure 1.27. On the left, a patch with three planar interpolants (same as in Figure 1.22a) can be seen, while on the right, one of the interpolants is changed to a sphere. This somewhat modifies the shape, but the patch overall remains stable. It is worth noting that here, the patch does not interpolate the curvature of the sphere, as the scheme used is only G^1.

In the next examples, a sparse (9×9×9) voxel array was created. For each cell, based on the eight isovalues in the corner, polyline loops were generated from the estimated edge intersections. Ambiguities (when on a face all four edges have an intersection) were resolved

Corner points				
$[0, 0.5, 1]$	$[0, 0.8, 0]$	$[1, 0.5, 0]$	$[1, 0, 0.5]$	$[0.5, 0, 1]$
Patch #1				
Side points				
$[0, 0.6, 0.5]$	$[0.5, 0.6, 0]$,	$[1, 0.35, 0.35]$	$[0.65, 0, 0.65]$	$[0.35, 0.35, 1]$
Side coefficients				
0.6	0.6	-2.34	2.34	-2.34
Interior point:	$[0.5, 0.5, 0.5]$	Central coefficient:		-7.77
Patch #2				
Side points				
$[0, 0.6, 0.5]$	$[0.5, \mathbf{0.8}, 0]$	$[1, 0.35, 0.35]$	$[\mathbf{0.55}, 0, \mathbf{0.55}]$	$[0.35, 0.35, 1]$
Side coefficients				
0.6	**2.2**	-2.34	**0.45**	-2.34
Interior point:	$[0.5, 0.5, 0.5]$	Central coefficient:		$\mathbf{-8.94}$
Patch #3				
Side points				
$[0, 0.6, 0.5]$	$[0.5, 0.6, 0]$,	$[1, 0.35, 0.35]$	$[0.65, 0, 0.65]$	$[0.35, 0.35, 1]$
Side coefficients				
0.6	0.6	-2.34	2.34	-2.34
Interior point:	$[0.5, \mathbf{0.2}, 0.5]$	Central coefficient:		**55.83**

Table 1.1: *Points and coefficients for the patches in Figure 1.26. All patches are in the* $[0; 1]^3$ *cube. Differences from Patch #1 are* ***bold****.*

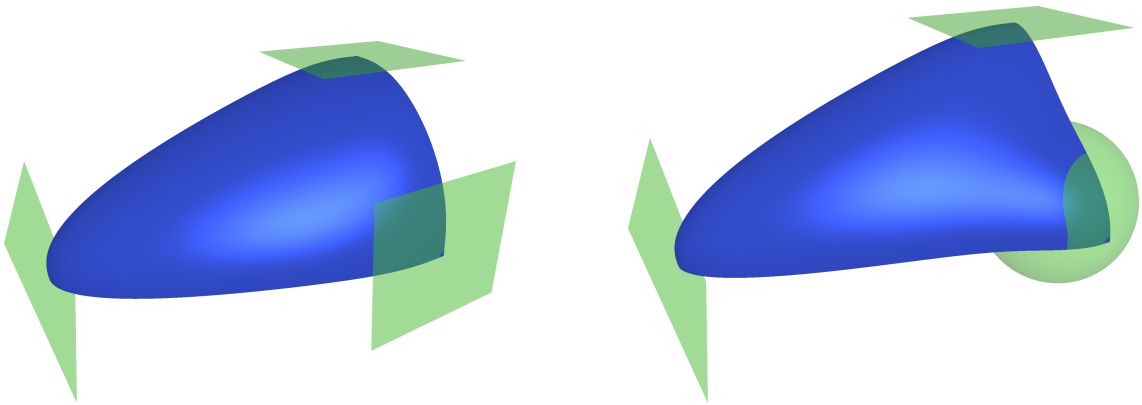
by minimizing the sum of distances between the two pairs. This resulted in one or more loops of closed polylines.

Then, in each of the isovertices, a plane was introduced, with its normal pointing towards the positive cell corner. From those, and the faces of the cube as bounding surfaces, corner I-patches were generated. Coefficients for the corner components were set to 1, and side coefficients were calculated with the triangle rule or the tetragon rule (see Appendix B). The central coefficient was set to 0.

In the first example (Figure 1.28) two sphere-like objects can be seen which are close to each other. Notice that the brown surfaces at their closest points are represented with the same corner I-patch. In the second example (Figure 1.29) a voxel value was modified, extending the volume of the bigger object. In Figure 1.30 a hole was drilled into the object by modifying isovalues in a line.

In the next examples, the scheme was modified so that when introducing the corner planes, an exact implicit function is used for both exactly calculating the isovertex and using an exact gradient. This can be useful in cases where evaluating the original functions would be very costly but approximating them with piecewise polynomial surfaces could bring a reduction in both storage and computation costs.

In Figures 1.31 and 1.32 an ellipsoid can be seen with a lower and a higher resolution cell structure. It can be observed that although the boundary curves approximate the original



(a) *Three plane interpolants.* **(b)** *Two plane and one sphere interpolants.*

Figure 1.27: *Changing a corner interpolant to a sphere.*

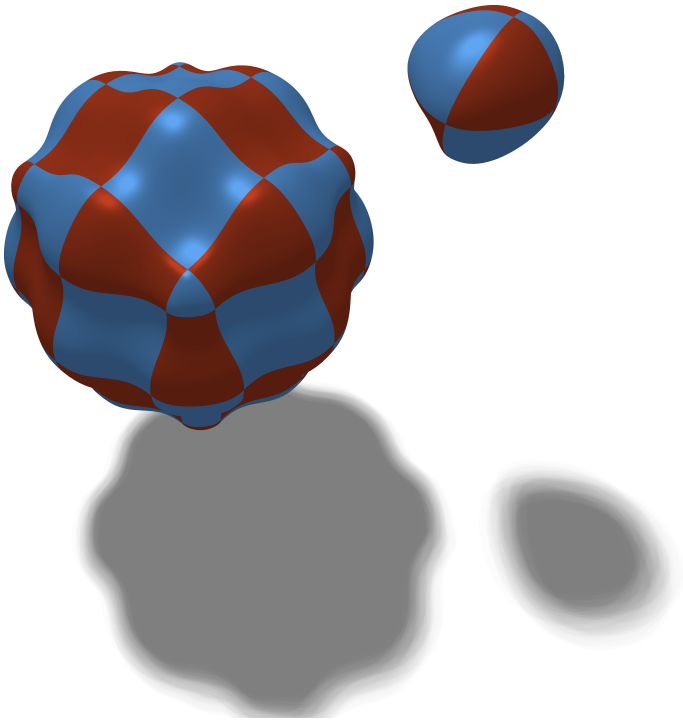


Figure 1.28: *Disjoint surfaces generated from voxel data.*

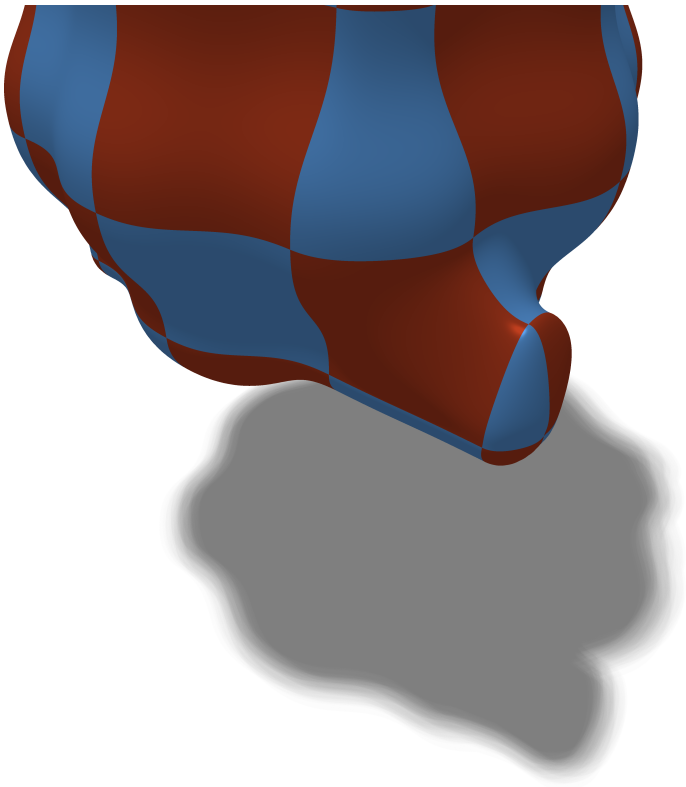


Figure 1.29: *Enlarging the object by modifying a voxel value.*

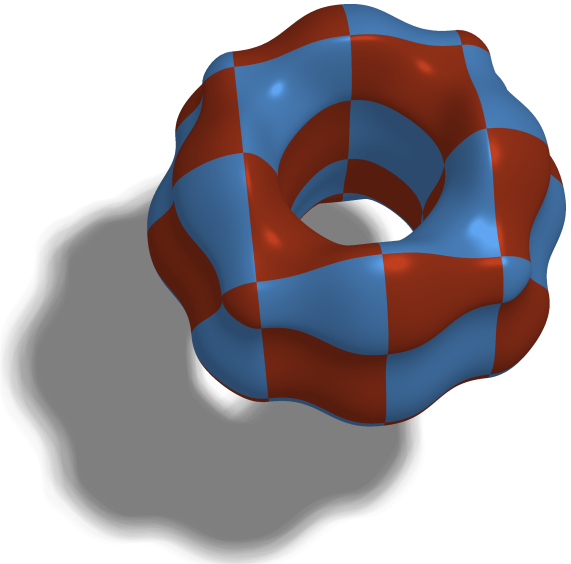


Figure 1.30: *Drilling a hole into the object by modifying voxel values.*

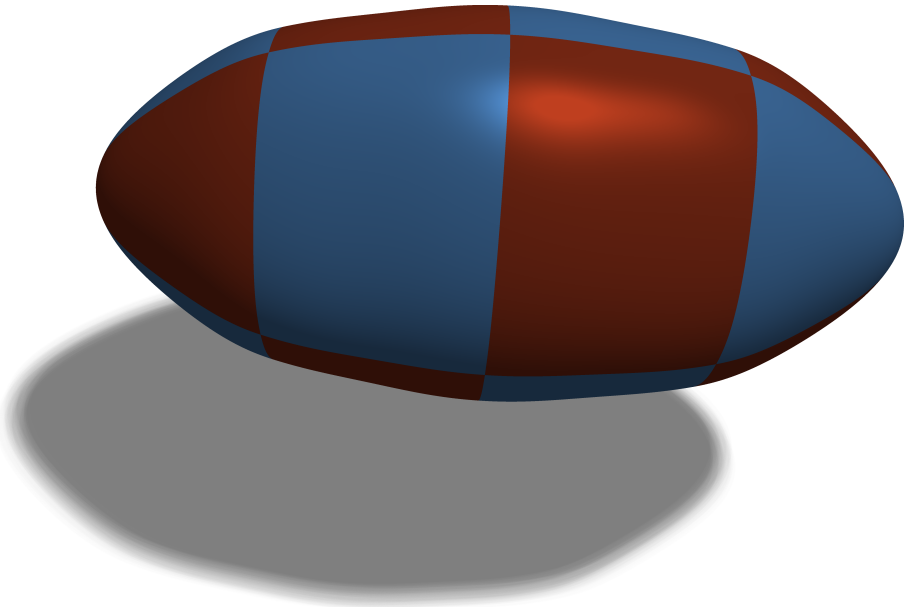


Figure 1.31: *Ellipsoid with lower resolution.*

surface well, the interior of some surfaces is less smooth. Optimization of the coefficients is therefore an important area of future research.

1.5.6 Summary

I have developed a new implicit representation, that combines corner interpolants, called corner I-patches, and published in [Sipos, 2022a] and the journal paper [Sipos, 2023]. Using corner interpolants that are typically planar surfaces, relatively complex patches can be represented. Corner I-patches are capable of representing surfaces consisting of multiple sheets, each defined by a separate loop of corners, which makes them especially useful in Marching Cubes-like cell-based surface reconstruction.

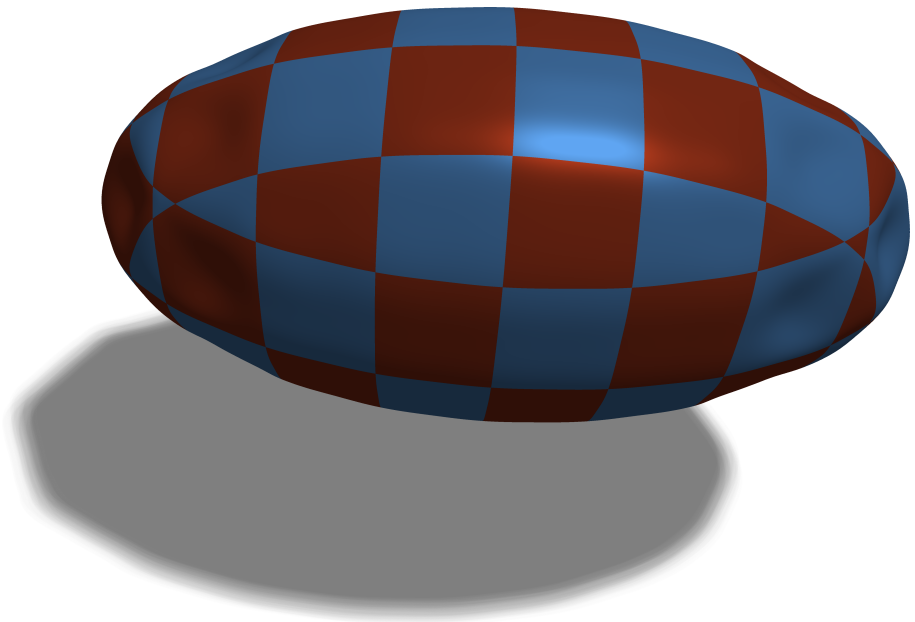


Figure 1.32: *Ellipsoid with higher resolution.*

Chapter 2

Applications of I-patches in freeform modeling and approximation

2.1 Overview

One of the major problems in Computer-Aided Design is the creation of mathematical representations for freeform surfaces. These typically describe complex geometries that cannot be defined by regular shapes such as planes, cylinders, or simple sweeps. Freeform modeling schemes have the following useful features:

- they are capable of approximating any smooth target surface, with a sufficient order; and
- a small change in the control data results in a small and local change on the surface.

It has already been mentioned in the Introduction that most surface elements used in freeform design are four-sided. The reason for this is the convenience of tensor-product parametric surfaces. Their general equation is

$$\mathbf{P}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{X}_{i,j} \cdot f_i(u) \cdot g_j(v), \tag{2.1}$$

with $u, v \in [0,1]$, where $\mathbf{X}_{i,j}$ can be called control vertices. The four boundary curves of the patch are naturally reproduced by fixing $u = 0$, $u = 1$, $v = 0$, and $v = 1$.

In various applications, the use of multi-sided (i.e. ≥ 4) surfaces is required. These include vertex blending and hole filling. Also in certain design situations, the quality of surfaces produced by the composition of multi-sided patches is better than that of only four-sided ones. It should be mentioned that implicit surfaces, representing half-space boundaries are indifferent to this distinction: modeling a four- and a non-four-sided patch in implicit form generally has the same difficulty.

Approximating structured or unorganized point data with freeform surfaces is also an important operation, especially in reverse engineering [Várady et al., 1997]. Least-squares fitting is the most widely used technique, however, most surface representations do not have a directly evaluable Euclidean distance metric, thus in most cases, iterative minimization approaches are utilized.

In the case of fitting B-spline surfaces, generally successive approximation is performed. This means that starting from an initial estimation of parameter values for data points, in alternating steps, parameter values are updated, and distance from those points is minimized. This is a stable, but slow algorithm. In contrast, if our representation of choice is an implicit *signed distance function* (SDF), the solution would be directly computable by solving the normal equation. Implicit faithful distance fields are in between; although not SDFs, they provide a reasonably good approximation for them.

2.2 Previous work

In this chapter, we deal with three important application areas of I-patches, namely polyhedral design, vertex blending, and surface approximation. These areas have been heavily investigated in the CAGD literature, and undoubtedly parametric surface models are predominantly used. Here we collect a few important publications to overview the state-of-the-art and show how implicit modeling can be fitted into the global image.

2.2.1 Polyhedral design

Modeling complex surfaces based on control polyhedra is one of the main areas in CAGD, where a freeform shape is constructed indirectly by means of a complex control polyhedron. As we want to define a composite surface that mimics the shape described by the polyhedron, it is required to support multi-sided configurations and vertices with arbitrary valencies.

The primary method with a vast literature that is used for polyhedral design is recursive subdivision. It includes the popular Catmull-Clark [Catmull and Clark, 1978] and Doo-Sabin [Doo and Sabin, 1978] algorithms, which produce a sequence of polyhedra converging to a smooth limit surface. It is a well-known result that these limit surfaces are smoothly connected B-spline surfaces, but in extraordinary points, they have visible shape artifacts.

Direct polyhedral methods emerged in the 1990s, see [Peters, 1991], using quadrilateral patches, where the main difficulty was splitting the multi-sided regions, while retaining the surface smoothness.

Later, various multi-sided parametric patches were proposed to avoid the problem of internal discontinuities. These include S-patches [Loop and DeRose, 1990], SuperD patches [Rockwood and Gao, 2018] and generalized Bézier patches [Szörfi and Várady, 2022].

The use of implicit surfaces for polyhedral design has been less notable. The application of rational A-patches [Xu et al., 2001] to create C^1-continuous surfaces from triangle meshes, interpolating its vertices could be considered a similar method, although the resulting surface is very different from what a subdivision method would produce.

2.2.2 Vertex blending

A crucial issue in geometric modeling is to create smooth transitions between adjacent primary surfaces. There is a rich literature of blending (filleting) methods, nevertheless our interest here is to produce vertex blends, i.e. small connecting pieces that connect converging edge blends as well as faces at a given vertex.

A general solution to this problem is *setback vertex blending*, see the early papers [Braid, 1997] and [Várady and Rockwood, 1997]. The terminations of the edge blends are pushed

away from the vertex by various setback values to ensure the necessary space for a smooth transition. It has been emphasized in [Várady and Hoffmann, 1998] 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 $2n$.

Again, parametric solutions are predominantly used, see e.g. [Zhou and Qian, 2010] for a method using S-patches, however, in [Hartmann, 2001] and [Várady et al., 2001], (symmetric) functional splines and I-patches, respectively, were applied for various kinds of vertex blends (suitcase, house, box, setback).

In this thesis, we focus on the setback principle and investigate how I-patches can be used for this modeling task.

2.2.3 Surface approximation

Surface approximation is an extremely important area within CAGD. Early publications described methods where a given point cloud (or a related triangular mesh) was approximated by a single surface, however, with the emergence of digital shape reconstruction lots of papers were published that dealt with modeling complex segmented datasets by a collection of patches [Várady et al., 1997].

Surface approximation is also dominated by fitting parametric surfaces over topologically irregular data sets, fundamental questions include how to trim these surfaces, how to fair the shape, how to smoothly connect and constrain the adjacent pieces [Weiss et al., 2002].

Implicit surfaces have been used for the previously mentioned strategy of approximating the underlying data by a single surface, like Poisson reconstruction [Kazhdan et al., 2006], however, according to our best knowledge our proposed approach to use piecewise implicit surfaces is presumably the first of its kind.

2.3 Modeling patchworks with polyhedral frames

In this section, we present two algorithms. The first is *polyhedral design*, where a complex freeform patchwork is defined by means of a control polyhedron, yielding a collection of smoothly connected I-patches. The second is *setback vertex blending*, where a multi-sided I-patch is created to connect primary surfaces and edge blends.

These constructions are based on the following steps: (i) create a boundary curve network derived from the polyhedron, (ii) create ribbons and bounding surfaces based on the frame, (iii) set shape parameters, and (iv) recognize and resolve special cases.

The main contribution of this section is giving a comprehensive framework for the use of implicit patches in these applications. Although the basic methods for creating the control data for these patchworks were discussed in previous work (see especially [Loop and DeRose, 1990] for polyhedral design and [Várady and Rockwood, 1997] for setback vertex blending), we will need to elaborate on the choice of geometric entities and algorithms best suited for implicit surfacing.

2.3.1 Polyhedral design

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

2.3.1.1 Curve networks

The initial step is to create a freeform curve network. Its topological structure corresponds to the dual graph of the polyhedron, see Figure 2.1. 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 centroids we determine local tangent planes; 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 the least-squares sense. Then, ribbons are uniquely defined (see next section) by two centroids and two local tangent planes. Note that these ribbons are being shared by the adjacent patches along the common boundary, thus the construction ensures smooth continuity between the patches.

A simple example is shown in Figure 2.1. 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 uniquely defined for both the left 3-sided and the 5-sided patch, ensuring smooth connection.

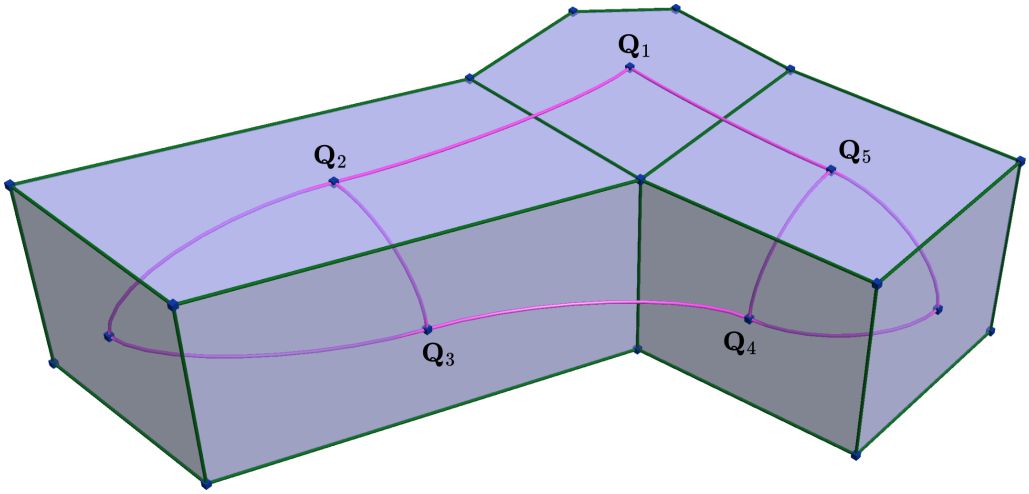


Figure 2.1: *Control frame for polyhedral design.*

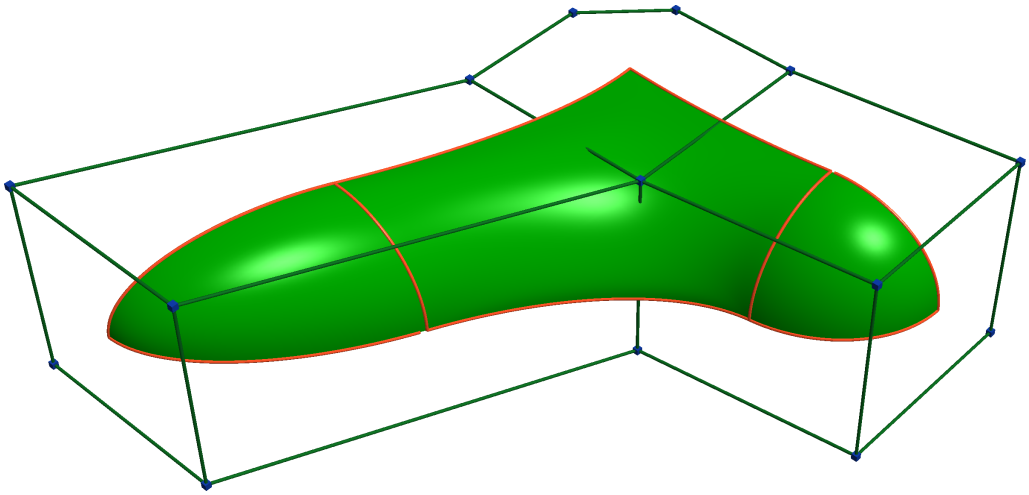


Figure 2.2: *Curve network and related surfaces.*

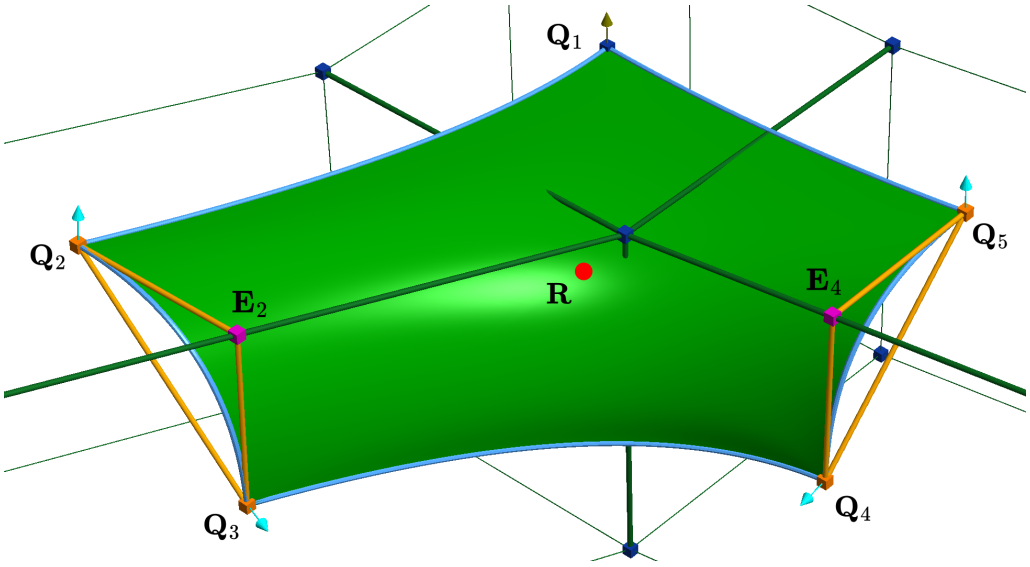


Figure 2.3: *Control triangles. Small arrows show the normal vectors* $\mathbf{n}_i$ *corresponding to the corner points* $\mathbf{Q}_i$*;* $\mathbf{R}$ *is a reference point.*

2.3.1.2 Ribbons and bounding surfaces

In this proposed framework for polyhedral design, we deal with a variety of ribbons and bounding surfaces that are either planes, Liming-surfaces (1.3.1.1) or I-lofts (1.3.2.3). Any ribbon-bounding surface pair can occur under certain circumstances for a given side of a patch.

Planes Planar ribbons are only used when the neighboring faces of the control polyhedron are identical. In this case, that plane can be used as the ribbon and will fulfill the continuity constraints.

Planar bounding surfaces, however, are often used due to their simplicity and the fact that their use possibly makes the polynomial degree of the resulting patch lower. We introduce a control triangle for each side that is made of the two face midpoints and the edge midpoint. The planar bounding surface is defined to be the plane of this triangle. See Figure 2.3.

Liming-surfaces Liming's method is often a straightforward way of creating ribbons. Let P_i be the equation of the plane containing $\mathbf{Q}_i$ and having normal vector $\mathbf{n}_i$, then the equation of the ribbon is

$$R_i = (1 - \lambda_i) P_i P_{i+1} - \lambda_i C_{i,i+1}^2 = 0, \tag{2.2}$$

where λ_i is a scalar parameter, and $C_{i,i+1}$ denotes the equation of 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* $\mathbf{s}_{i,i+1}$, by means of which this plane is determined. The normal of $\widetilde{C_{i,i+1}}$ will be set to the vector product of the chord and $\mathbf{s}_{i,i+1}$.

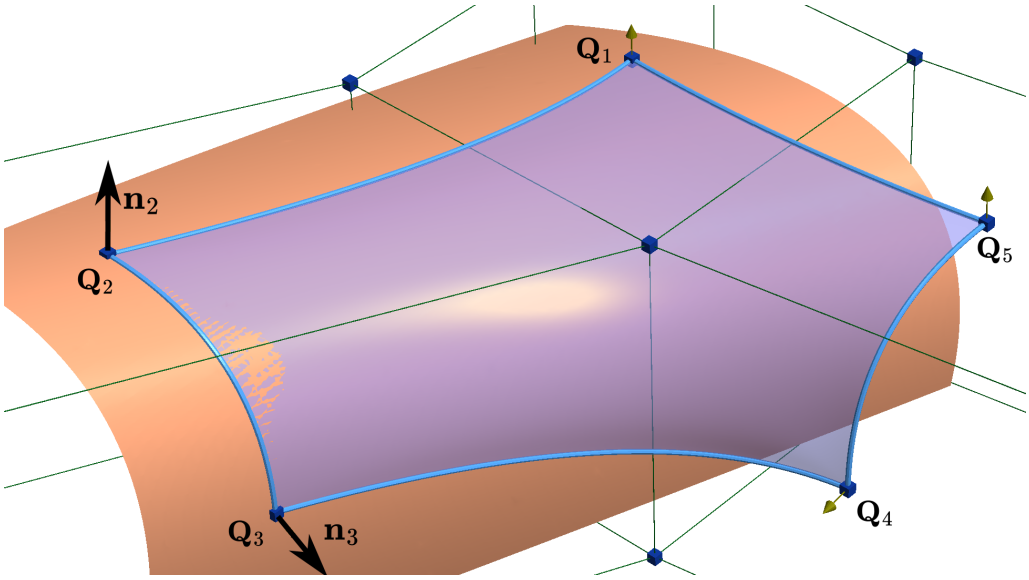


Figure 2.4: *A ribbon surface on the* $\mathbf{Q}_2\mathbf{Q}_3$ *boundary.*

One special case is when $\mathbf{s}_{i,i+1}$ 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 obtain 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. An example of a Liming-ribbon, which touches the corners $\mathbf{Q}_2$ and $\mathbf{Q}_3$, can be seen in Figure 2.4.

Liming-surfaces can also be used as bounding surfaces. Some of the possible reasons for this will be discussed later, in 2.3.3.2. In this case, we define B_i as a quadratic Liming-surface using binormals $\mathbf{m}_i$ at the two corners. This construction produces *curved bounding surfaces*. Figure 2.5 shows a curved bounding surface interpolating corners $\mathbf{Q}_1$ and $\mathbf{Q}_2$.

In cases when both the ribbon and the bounding surface were created this way, the intersection curve (the boundary curve of the patch) will also be quadratic. This is because they both contain the rational quadratic parametric curve defined by the three vertices of the control triangle and rational coefficients which can be directly calculated from λ. Thus it is easy to evaluate points on the boundary curves, which can be useful for some applications.

However, there are cases when a Liming-ribbon or bounding surface cannot be created. It is a necessary condition for the existence of a conic boundary with Liming-ribbons that the opposite corner points fall into consistently oriented positive half-spaces of the tangential planes. Formally,

$$P_i(\mathbf{Q}_{i+1}) > 0, \text{ and } P_{i+1}(\mathbf{Q}_i) > 0 \tag{2.3}$$

must hold. This condition was also pointed out by [Bajaj and Ihm, 1992b]. An example is shown in Figure 2.6; 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

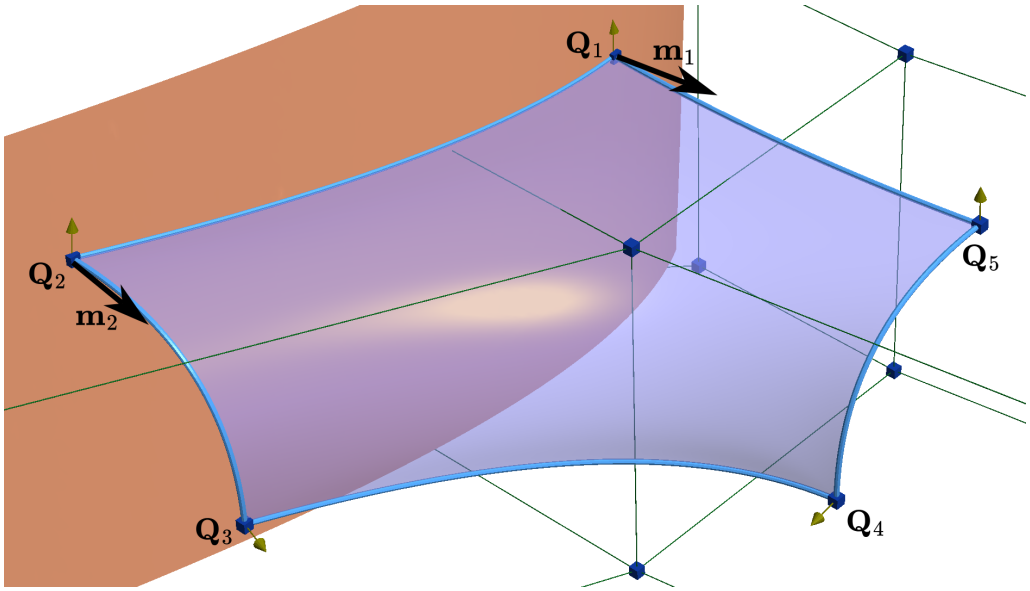


Figure 2.5: *A curved bounding surface on the* $\mathbf{Q}_1\mathbf{Q}_2$ *boundary.*

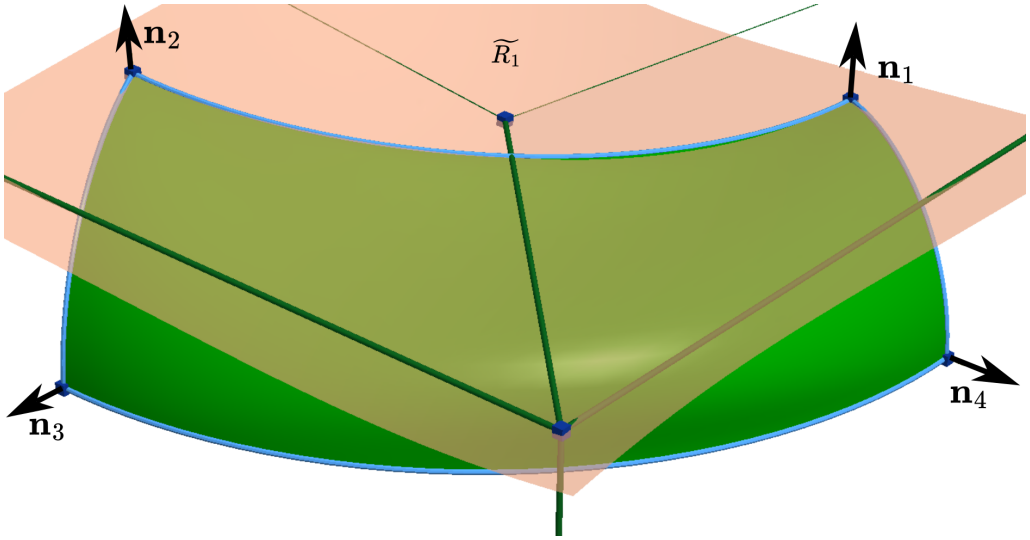


Figure 2.6: *An I-loft with twisted normal vectors.*

backward. 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. This motivates the application of the I-loft surfaces.

I-lofts I-loft ribbons are constructed based on the corner planes $\widetilde{P_i}$ and $\widetilde{P_{i+1}}$ and the local bounding planes $\widetilde{L_{i,1}}$ and $\widetilde{L_{i,2}}$:

$$R_i = w_1 \cdot P_i \cdot L_{i,2}^2 + w_2 \cdot P_{i+1} \cdot L_{i,1}^2 + w \cdot L_{i,1}^2 \cdot L_{i,2}^2. \tag{2.4}$$

A natural choice for the local bounding surfaces is to use two parallel planes perpendicular to the chord and interpolating the corner points. In Figure 2.6 an I-ribbon is shown that resolves the difficult case with twisted normals mentioned above.

We can also use I-lofts as bounding surfaces, especially when the binormal planes $\widetilde{M_i}$ and $\widetilde{M_{i+1}}$ do not fulfill Equation 2.3. In that case the I-loft

$$B_i = w_1 \cdot M_i \cdot L_{i,2}^2 + w_2 \cdot M_{i+1} \cdot L_{i,1}^2 + w \cdot L_{i,1}^2 \cdot L_{i,2}^2 \tag{2.5}$$

can be used instead of Liming-surfaces. Note however, that I-lofts have a polynomial degree of 4, and they are on the second power in the equation of a G^1 I-patch, thus overusing them can make its degree very high.

2.3.1.3 Shape parameters

Once the ribbons and bounding surfaces are defined, the next step is to set the w_i weights. The orientation of all ribbons must ensure a consistent positive direction normal fence, as discussed in 1.3.2.6. All w_i must be positive since the orientation of the ribbons must be retained. Let us pick a reference point $\mathbf{S}$ that is typically, but not necessarily, a point to be interpolated by the I-patch. The individual weights are set by satisfying $w_i R_i(\mathbf{S})/B_i^2(\mathbf{S}) = \pm 1$, using the algebraic distances from 1.3.2.5. The sign depends on the location of the reference point with respect to the ribbon. For example, in Figure 2.3, the reference point lies under the $\mathbf{Q}_1\mathbf{Q}_2$ ribbon, but above the $\mathbf{Q}_2\mathbf{Q}_3$ ribbon. Thus the surfaces are combined in a way that they will have the same contribution at the reference point.

Keeping the above initial w_i values, there is freedom to choose an arbitrary w weight in Equation (1.21), which 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. Previously we set the coefficients such that in $\mathbf{S}$ the sum of the absolute values of all weighted algebraic ribbons is one; however, their sign depends on the sign of the R_i-s. The patch should interpolate $\mathbf{S}$, so w needs to be set to $\sum \mathrm{sgn}(P_i(\mathbf{R}))$, which ensures that the patch interpolates $\mathbf{S}$.

Of course, it may make sense to modify the default weights w_i. Generally, the initial values 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 from the I-patch with the default coefficients, the method for which can be found in Appendix D, using the defaults, then these values are updated by minimizing the polyhedral energy functional proposed in [Pellis et al., 2019]. The optimization runs until there is no significant improvement in the energy, or the maximum number of iterations has been reached.

2.3.1.4 Examples

Polyhedral design is demonstrated using a simple test object (Figure 2.7). When performing simple operations, such as dragging the right face and repositioning the top vertices, the

shape is modified in a natural manner. On the left side, the left face is extruded, creating a new knob-like shape.

2.3.2 Setback vertex blending

Another interesting area where I have investigated the use of I-patches is vertex blending. They 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 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.

2.3.2.1 Curve networks

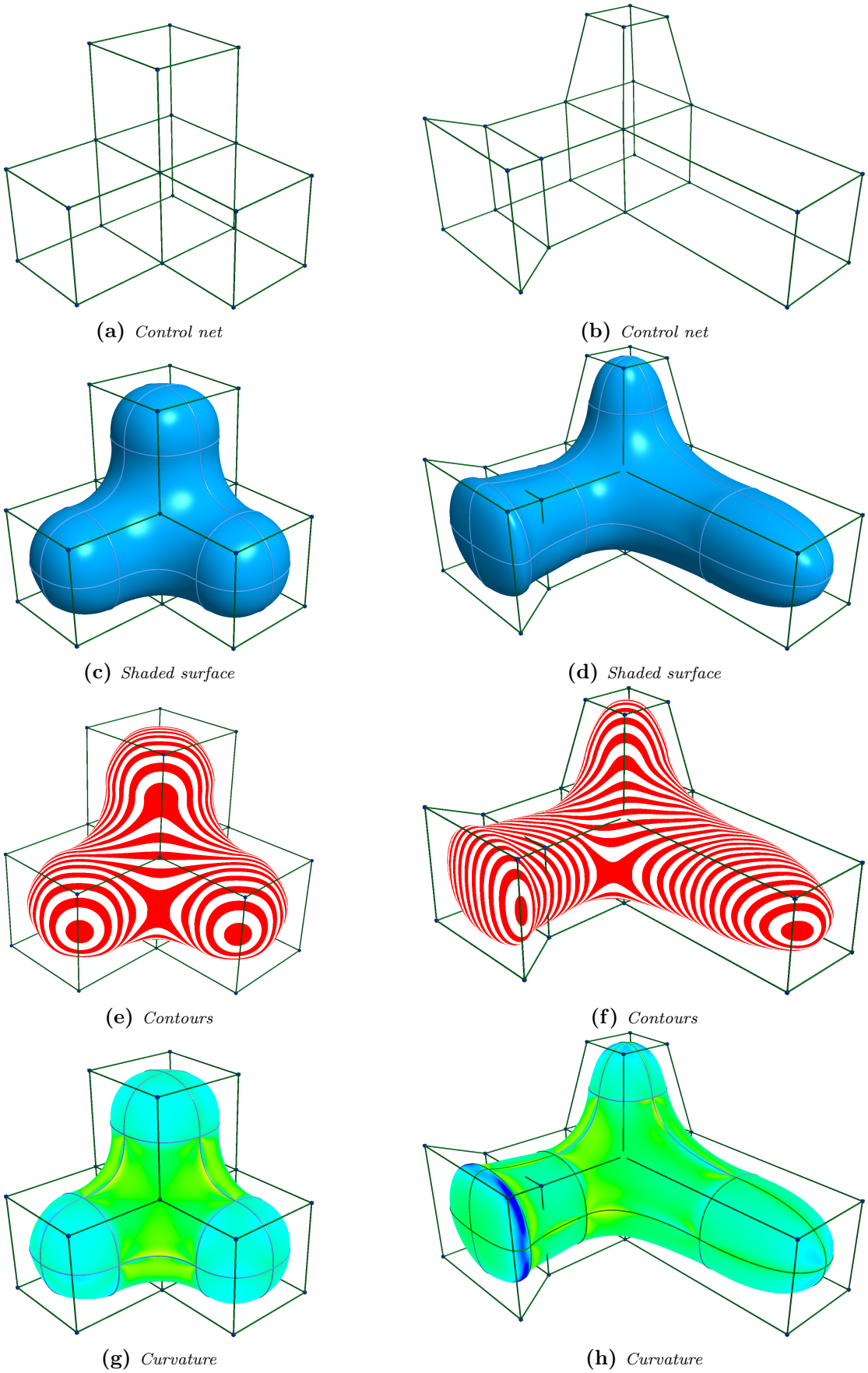
The basic components of a setback vertex blend are shown in Figure 2.8. Here we describe vertex blends for polyhedra, but the concept nicely generalizes for more complex ribbons and bounding surfaces. The ith edge blend, generally a cylinder, is bounded by two *rail curves* (colored pink), that run on the "previous" and "next" ribbons. 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 profile curve.

On each polyhedron face, 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 curve 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 C; for more details see [Várady and Hoffmann, 1998].

2.3.2.2 Ribbons and bounding surfaces

Constructing ribbons and bounding surfaces for setback vertex blending is fairly straightforward: ribbons are an alternating sequence of cylindrical edge blends and planar polyhedron faces, while bounding surfaces are an alternating sequence of bounding planes perpendicular to the blended edges and curved bounding surfaces along the spring curves.

In this proposed scheme, the profile curves of the edge blends are circles, while the spring curves are parabolas. The reason for this is that profile curves can be defined to be symmetric, while spring curves may have an uneven shape, which parabolas were observed to handle better.



(a) *Control net* **(b)** *Control net*

(c) *Shaded surface* **(d)** *Shaded surface*

(e) *Contours* **(f)** *Contours*

(g) *Curvature* **(h)** *Curvature*

Figure 2.7: *Polyhedral design examples.*

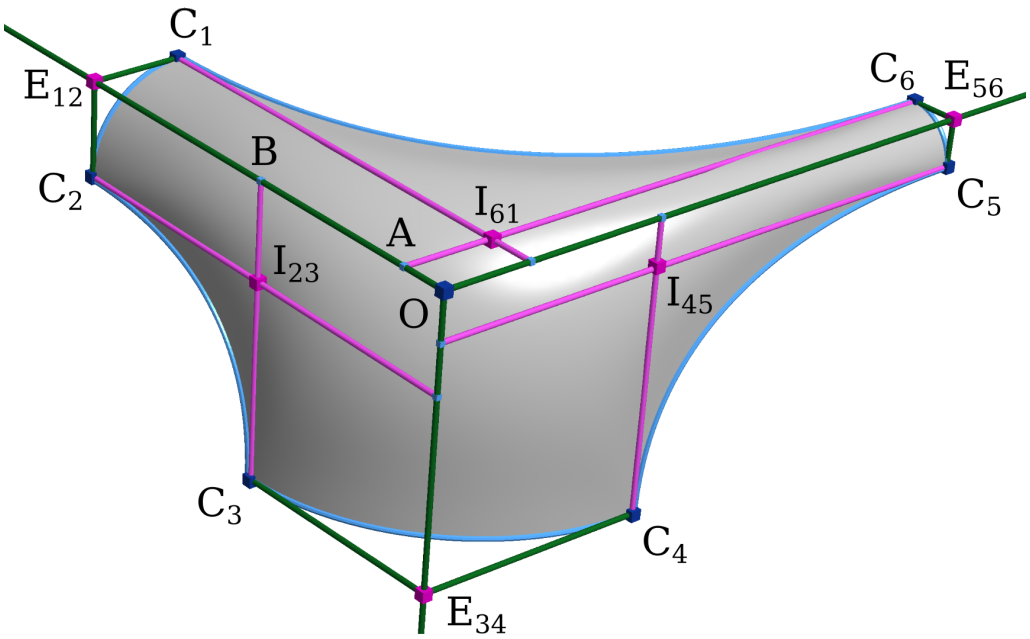


Figure 2.8: *Construction of a setback vertex blend.*

2.3.2.3 Shape parameters

A reference point is defined then by adding a displacement vector to the central vertex **O** based on the average deviation between the edge points $\mathbf{E}_{ij}$ and the midpoints of the related circular profiles. Then we set the weights in the same way as in the case of polyhedral design (2.3.1.3) and apply the I-patch equation.

2.3.2.4 Examples

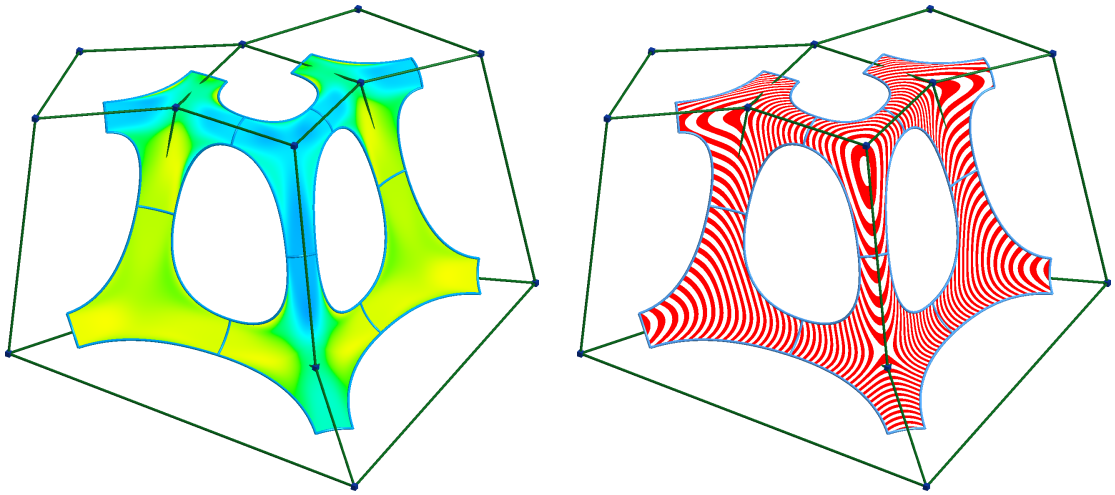
The test case presented is a combination of connected setback vertex blends, applied on a polyhedron. In Figure 2.9, three 8-sided and three 6-sided patches smoothly connect to planar faces (not shown). The patches are displayed with curvature maps and contours.

2.3.3 Special cases

In 2.3.1.2 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-loft. Here we deal with two further problems.

2.3.3.1 Handling ribbon problems

The I-patch scheme may produce dubious shapes if the distance fields produced by the 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,



(a) *Curvature map* **(b)** *Contours*

Figure 2.9: *Connected vertex blends.*

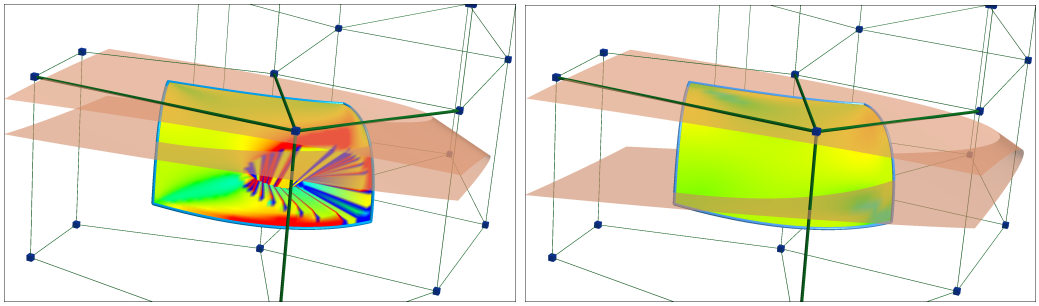


Figure 2.10: *Fixing a shape artifact by tuning the fullness of the ribbons.*

or high curvatures when the ribbon "turns under" itself; see the transparent surfaces in Figures 2.10 and 2.11. 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 2.10, 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 ribbon, and exclude the highly curved portion or the undesired branch. We can modify Liming's method to obtain a *piecewise Liming-ribbon*:

$$R_i = \begin{cases} (1-\lambda)P_iP_{i+1} - \lambda C_{i,i+1}^2 & C_{i,i+1} \geq 0, \\ P_iP_{i+1} & C_{i,i+1} < 0. \end{cases} \tag{2.6}$$

An example is shown in Figure 2.11, where a cylinder with a small radius was replaced by a piecewise ribbon, yielding a good setback vertex blend. The drawback of this method is that having only G^1 continuity along $\widetilde{C_{i,i+1}}$ may affect internal continuity within the patch, although in many cases the curvature maps or the isophote lines do not indicate

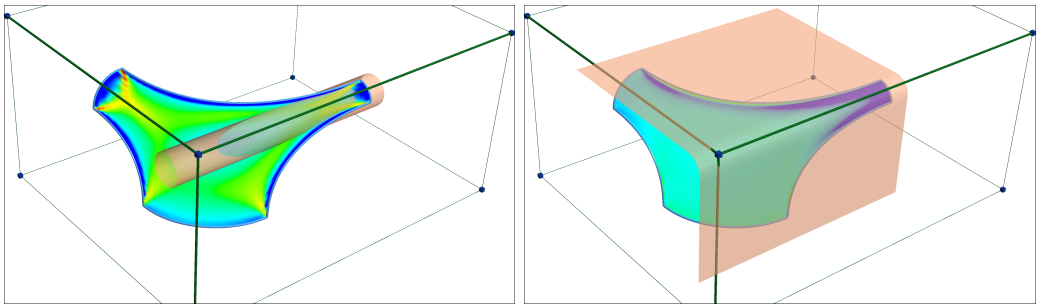


Figure 2.11: *Fixing a shape artifact by using piecewise ribbons.*

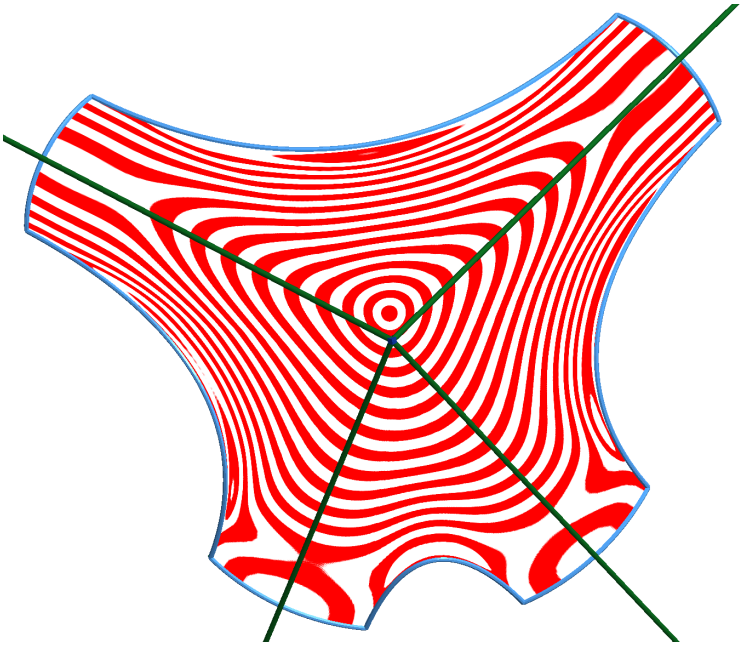


Figure 2.12: *Isophote lines on an 8-sided setback vertex blend represented by an I-patch with piecewise ribbons.*

this phenomenon at all, see for example Figure 2.12.

2.3.3.2 Handling bounding problems

It is a necessary assumption 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 $\widetilde{B_j}$ intersects another boundary curve, where $P_i = B_i = 0$, and thus a disconnected I-patch is 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 2.13, where the top right bounding surface intersects the bottom left boundary. The problem is fixed, if we use a curved bounding surface.

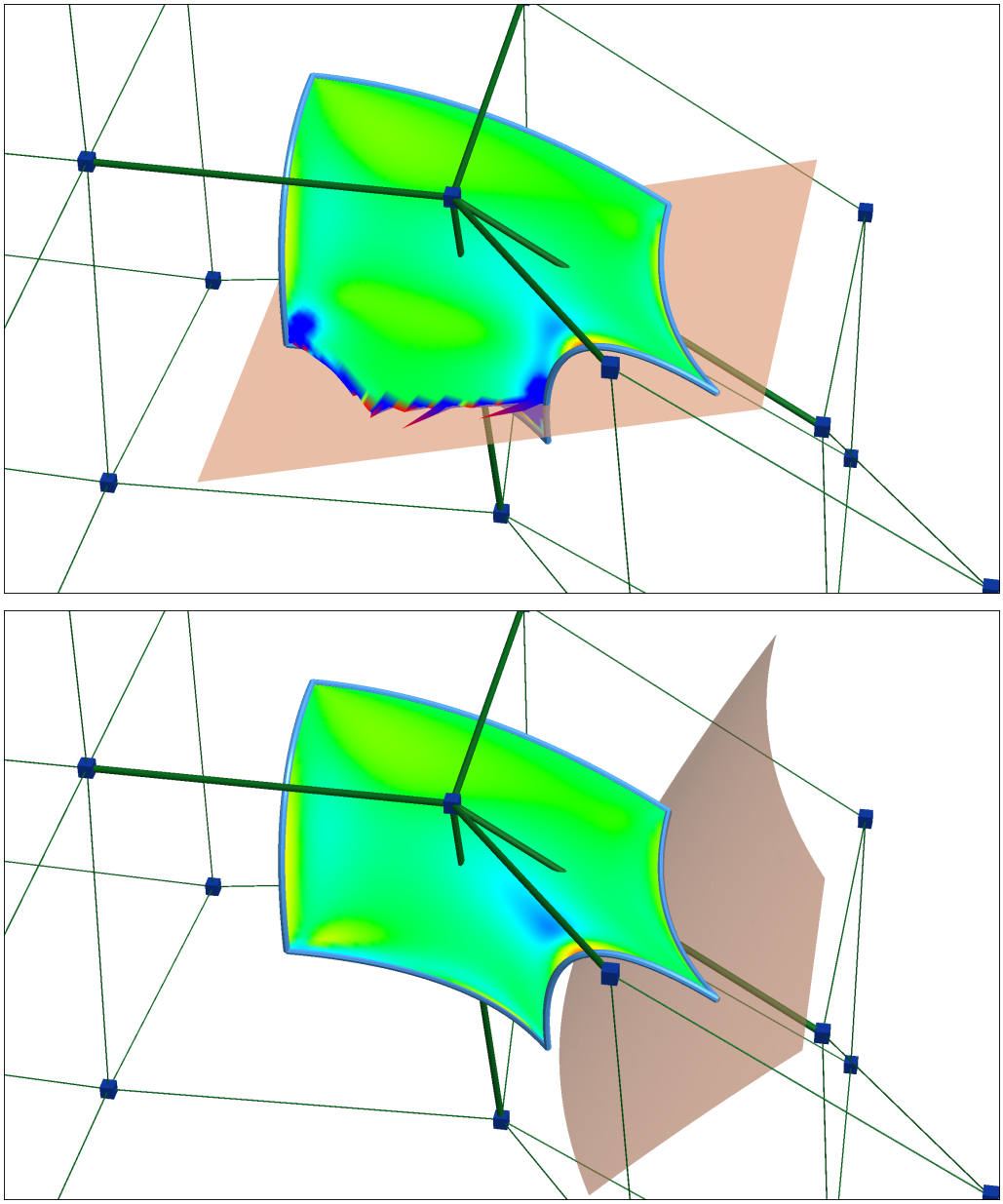


Figure 2.13: *Removing a shape artifact by using a curved bounding surface.*

Algorithm 1: I-patchwork from control polyhedron.

Input: polyhedron (open or closed)
for *all faces of the polyhedron* **do**
  create a midpoint and corresponding normal vector
end
for *all edges of the polyhedron* **do**
  take the midpoints of the two neighboring faces
  if *normals are compatible (Eq. 2.3)* **then**
    create a Liming-ribbon
  else
    create an I-loft ribbon
  end
  create planar bounding as the plane of edge midpoint and two face midpoints
  if *planar bounding intersects other boundaries of the patch* **then**
    create curved bounding using binormal vectors
  end
end
for *all vertices of the polyhedron (corresponding to resulting patches)* **do**
  select reference point **S**
  for *all sides of patch* **do**
    set w_i coefficient that balances contribution in **S**
  end
  set w coefficient so that patch interpolates **S**
end
(optionally) run numerical optimization based on smoothness to tune coefficients
Result: G^1-connected patchwork of I-patches

2.3.4 Summary

I have devised two schemes for modeling complex implicit patchworks via control polyhedra. In polyhedral design each vertex of the polyhedron with valency n will be replaced by an n-sided I-patch, smoothly connecting to the neighboring patches. In setback vertex blending each vertex with n converging edge blends will be represented by a $2n$-sided I-patch, connecting to the polyhedron faces. I have elaborated algorithms for these applications to create general topology curve networks, and various types of ribbon and bounding surfaces, that match the geometric constraints coming from the polyhedral models and ensure G^1 continuity. I have identified special cases that need to be detected and resolved for obtaining valid models. I have also proposed a tessellation method to convert I-patches to good quality triangle meshes.

The outline of the algorithm can be found in Algorithm 1 for polyhedral design and Algorithm 2 for setback vertex blending. The tessellation method is detailed in Appendix D. These results have been published in [Sipos et al., 2020].

Algorithm 2: I-patches for setback vertex blending.

```
Input: polyhedron with planar faces
for all edges of the polyhedron do
  | create edge blends
end
for all vertices of the polyhedron do
  | for all edges connecting to vertex do
  |   | create a planar bounding perpendicular to the edge extract patch corner
  |   |  points from its intersection with the edge blend
  | end
  | for all faces connecting to vertex do
  |   | create a spring curve connecting the corner points on the face create
  |   |  bounding surfaces by extruding it use the planar face as ribbon surface
  | end
  | select reference point S
  | for all sides of patch do
  |   | set w_i coefficient that balances contribution in S
  | end
  | set w coefficient so that patch interpolates S
end
Result: I-patches G^1-connected to faces and edge blends
```

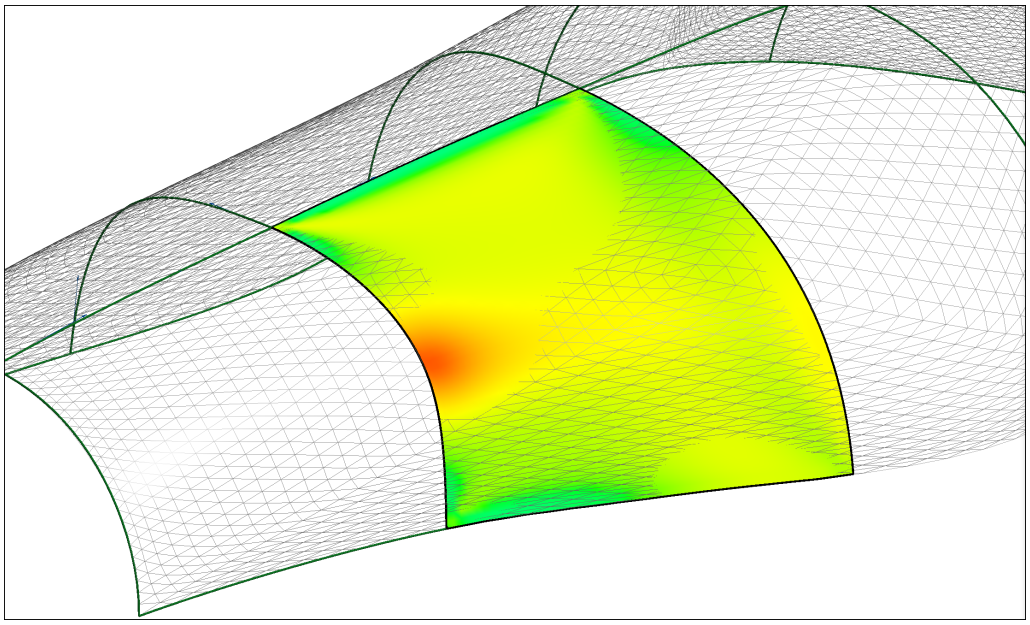


Figure 2.14: *Approximating a selected area on a mesh using an I-patch.*

2.4 Approximating meshes with smoothly connected implicit surfaces

In this section, a new algorithm to approximate triangular meshes using I-patches will be presented. While there is rich literature about doing the same, applying parametric patches, according to our best knowledge, a smoothly connected patchwork of implicit patches has not been applied so far.

The input is a mesh, and an additional structure is supplied: a user-defined vertex graph. These allow us to create a model by connecting relatively low-degree, high-quality single-patch surfaces composed of geometrically intelligible parts.

For example, a mesh, a vertex graph defined on it, and an approximating patch can be seen in Figure 2.14.

2.4.1 Ribbons and bounding surfaces

This section will introduce how to construct the ribbons and bounding surfaces for creating the patchwork of I-patches. As will be explained below, these surfaces are always defined as *weighted combinations* of planes given at the corner points and auxiliary planes derived from the corner data.

Let us take two adjacent corner points $\mathbf{P}_1$, $\mathbf{P}_2$, and incident *corner planes* $\widetilde{P_1}$, $\widetilde{P_2}$ with normal vectors $\mathbf{n}_1$, $\mathbf{n}_2$ (approximated from the mesh with the standard methods given by [Max, 1999] or [Cazals and Pouget, 2005]). The corresponding ribbon interpolates both points and normal vectors. Similarly, the bounding surface interpolates both points, but it is transversal to the ribbon surface. The ribbons and bounding surfaces are defined –

similarly to the constructions described earlier (2.3.1.2) – by three kinds of equations: (i) planes, (ii) *Liming-surfaces*, and (iii) *I-lofts*.

In the case of Liming-surfaces, the choice of the cutting plane C does affect its shape. Nevertheless, the simplest setting of its normal vector $\mathbf{n}_{12} = (\mathbf{n}_1 + \mathbf{n}_2)/2$ proved to be satisfactory. The parameter λ is going to be defined by the optimal approximation of the underlying polyline boundary, see the next section.

With I-lofts, there is freedom to select the bounding planes. We propose to constrain them to be parallel as it improves their stability, because in the direction in which the bounding surfaces are getting closer, fluctuation may be observed on I-lofts. It also seemed advantageous to set them perpendicular to the boundary, thus their normal was set to point in the direction of $\pm(\mathbf{P}_1 - \mathbf{P}_2)$. The weights w_1 and w_2 are shape parameters to set the fullness of I-lofts, and thus ensure the best approximation of the underlying polyline boundary, see the next section.

In this context, we propose using *planar bounding surfaces* to subdivide the interior of the mesh, since these lead to planar boundary curves and the equation of the I-patch becomes simpler. Intersections with Liming-ribbons yield conic curves, while we get planar I-segments with I-lofts. One simple formula to set the normal of this bounding plane comes from forcing it to include the average of the corner normals

$$\mathbf{n}_B = \mathbf{n}_{12} \times (\mathbf{P}_2 - \mathbf{P}_1). \tag{2.7}$$

Curved bounding surfaces may be used in the interior of the mesh to approximate non-planar subdividing curves, but they must be used for the approximation of curved external boundaries. In this case, the local direction of the mesh boundary at the corner points is calculated; the tangent planes of the bounding surface are defined by this vector, and the corresponding mesh normal. When a curved bounding surface is intersected with a curved ribbon, a general 3D curve is obtained.

It should be noted that the use of Liming-surfaces is not always possible. If the boundary to be approximated has a point of inflection, or the corner normals are twisted, only I-lofts can be used as already explained in 2.3.1.2.

It is also important to observe that the ribbons and the bounding surfaces depend only on the corner points, the corner normals, and the mesh, so if two adjacent I-patches share the same $\mathbf{P}_1\mathbf{P}_2$ boundary, the identical surface components will guarantee a smooth connection.

2.4.2 Approximation with I-patches

The purpose of this section is to describe how to construct I-patches that approximate a given mesh. At this point, we assume that the bounding surfaces have already been determined. We intersect these with the mesh and trace polylines on the mesh between

the adjacent corner points. We do not need tracing along the external edges of the mesh, since there the polylines are directly available. First, we create ribbons that approximate these polylines, then an I-patch that approximates the interior points of the mesh.

Ribbons must match the given corner points and the corresponding normals. Liming-ribbons are constructed by optimizing their free λ parameter. This sets the shape of the ribbon and determines the best approximating conic boundary curve. I-lofts are somewhat more flexible, and we have two independent scalar weights for the optimization. The error function to be minimized is the squared sum of the faithful algebraic distances (see 1.4). Euclidean distances could also be used, but there was no substantial difference in the resulting patches, while it would be computationally more costly. As mentioned earlier, it may be possible that a boundary can be approximated by both a Liming-surface and an I-loft. In this case, we choose the one with the smaller error.

Figure 2.15 shows a Liming and an I-loft ribbon, with corner planes and sampled points on the mesh. The deviation map indicates that the ribbons are close to the mesh (green) in the vicinity of the related boundary.

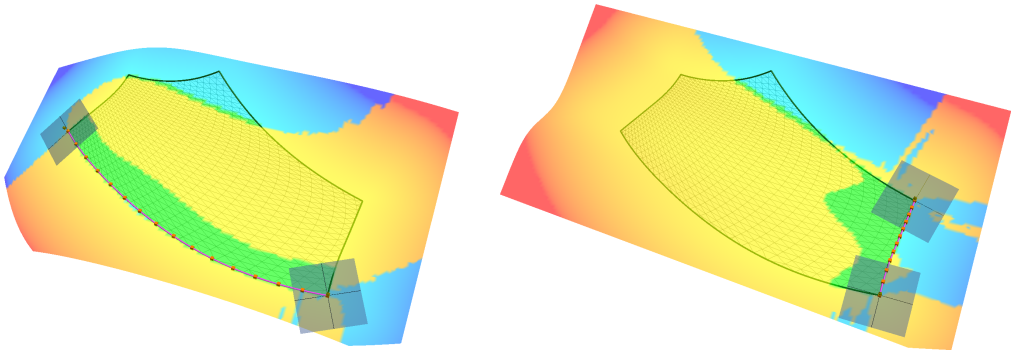


Figure 2.15: *A Liming (left) and an I-loft ribbon (right), showing deviation from the mesh.*

As the next step, mesh points are filtered so only those remain that are enclosed by the bounding surfaces. Then, the mass center of the polyline boundaries of the patch is computed and projected onto the mesh. This initial center point $\mathbf{C}$ is used to determine the initial weights w_i for the ribbons by enforcing that all terms $\frac{w_i R_i(\mathbf{C})}{B_i^2(\mathbf{C})}, i = 1, \ldots, n$ are equal. In other words, the initial patch is composed in a way that the individual ribbons contribute to the patch in an even manner at the center point; the patch is also constrained to interpolate point $\mathbf{C}$.

Then, the mesh points are filtered so only those remain that are enclosed by the bounding surfaces. The initial setting of the weights can iteratively be improved to obtain the best approximation of the underlying data. The squared sum of the faithful distances (1.4) is minimized, and the optimal weights are set accordingly. For both the optimization of the ribbon and the patch, derivative-free methods can be applied; in the examples, the Nelder–Mead algorithm was used [Kochenderfer and Wheeler, 2019].

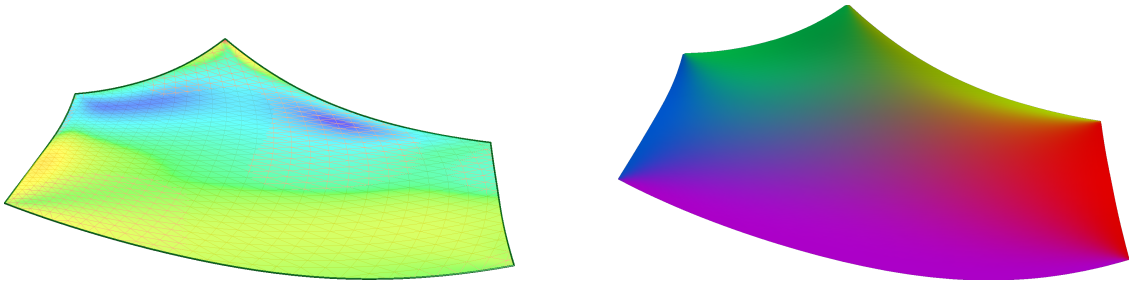
It was observed that it is a good idea to limit the ratios of the optimized weights. Otherwise, one ribbon may dominate the patch equation, and corrupt the surface quality along the

other borders. Accordingly, the maximum relative change of how much the weights of the optimized ribbons can deviate from their initial values was constrained. This maximum ratio is a user parameter σ. (In our examples, σ was set to 5.)

The formalized optimization problem is:

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{N} \hat{I}^2_{\mathbf{w}}(\mathbf{Q}_i), \text{ s.t. } \max_{i,j} \left(\frac{w_i R_i(\mathbf{C})}{B_i^2(\mathbf{C})} \Big/ \frac{w_j R_j(\mathbf{C})}{B_j^2(\mathbf{C})} \right) \leq \sigma, \tag{2.8}$$

where $\mathbf{w} = \{w_i\}_1^n$ is the vector of weights, and $\mathbf{Q}_i$ $(i = 1, \ldots, N)$ denotes the data points to be approximated. Figure 2.16a shows the curvature map of the optimized I-patch. In Figure 2.16b, colors are assigned to the individual sides and show the strength of the normalized blending functions.



(a) *Mean curvature map.* **(b)** *Contributions of each side.*

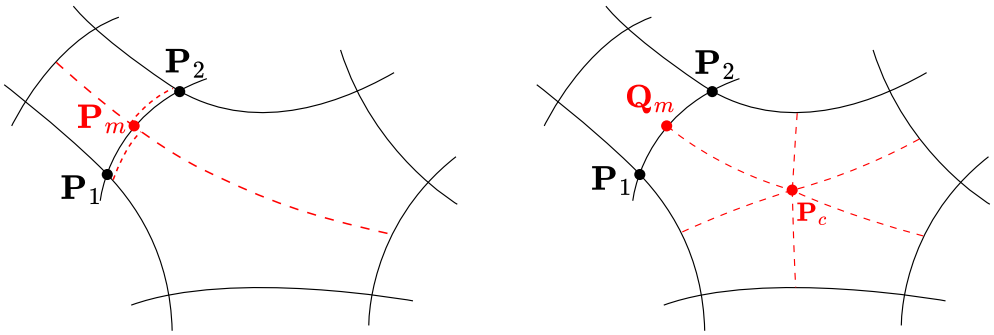
Figure 2.16: *Optimization of a 5-sided patch.*

2.4.3 Adaptive refinement of patchworks

The next problem to be solved is to refine the initial patchwork if it does not approximate the data points within a prescribed tolerance. Below we are going to introduce a method of adaptive refinement that follows simple heuristic rules. If a boundary is out of tolerance, it will be subdivided halfway between its endpoints. If a patch is out of tolerance, central splitting will be applied and a new mesh point will be inserted in its interior. The new subdivision points are connected with new boundary curves to the existing ones and new artificial points in order to create a consistent structure. This leads to a new graph of edges with new ribbons and bounding surfaces, given in the same form as before, i.e., segments defined by two endpoints with normals, and an underlying polyline to be approximated. The above procedure automatically iterates until the requested accuracy is achieved. Naturally, the initial network greatly affects the structure of the final patchwork.

It is crucial, of course, to prevent the propagation of local refinements over the full patchwork, and subdivide only where it is necessary. Fortunately, I-patches are well-suited for producing T-nodes, as will be explained below.

Figure 2.17a shows a boundary connecting $\mathbf{P}_1$ and $\mathbf{P}_2$ that needs to be subdivided; a new mesh point $\mathbf{P}_m$ is inserted, and (at least) four new boundaries are created. This is an X-node, where we compute a new position, and a new normal vector is *taken from the mesh.*



(a) *X-node.* **(b)** *T-node.*

Figure 2.17: *Cases of adaptive refinement.*

Figure 2.17b shows another configuration, where a 6-sided patch needs to be centrally split at $\mathbf{P}_c$, which is the formerly described center point of the original patch. Here we wish to preserve the left ribbon connecting $\mathbf{P}_1$ and $\mathbf{P}_2$, and let the two adjacent subpatches of the subdivided patch *inherit its midpoint* $\mathbf{Q}_m$ and the corresponding normal vector. In other words, the original ribbon and bounding surface are transferred, and both new surfaces along the $\mathbf{P}_1\mathbf{P}_2$ boundary will use exactly the same ribbon surface. Thus, G^1 continuity is naturally maintained. A related example will be shown in the next section.

2.4.4 Examples

Our first example in Figure 2.18 illustrates that I-patches naturally extend beyond the subspace determined by the bounding surfaces. Here we rendered the curvature map of the patch from Figure 1.19a using marching cubes.

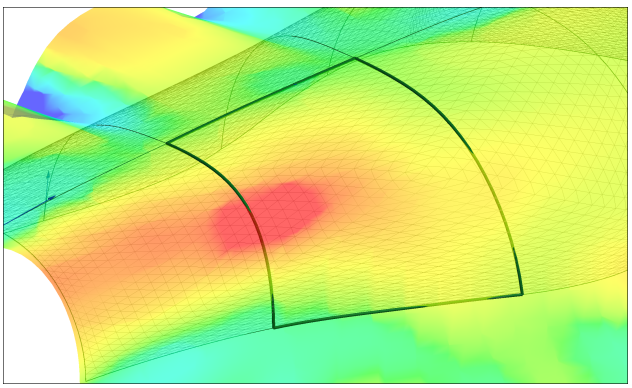


Figure 2.18: *Natural extension of an I-patch.*

The second example in Figure 2.19 shows the deviation map of a *shoe last* model. The network was created by a uniform cell structure yielding two 3-sided, six 4-sided, and two 5-sided I-patches. The accuracy of the approximation – here and at the forthcoming pictures – will be measured in percentages relative to the diagonal of the bounding box. The average (Euclidean) deviation from the mesh is 0.069%, while the maximum is 0.35%.

The third example (Fig. 2.20) is a sheet metal part composed of six 4-sided and two 5-sided patches, illustrating the effect of optimization and refinement. Figures 2.20a and 2.20b show

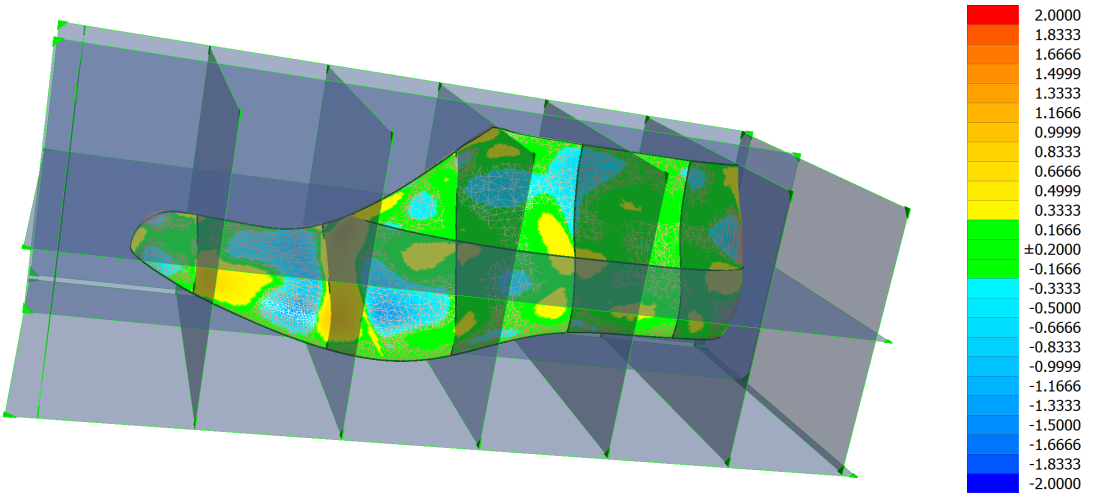
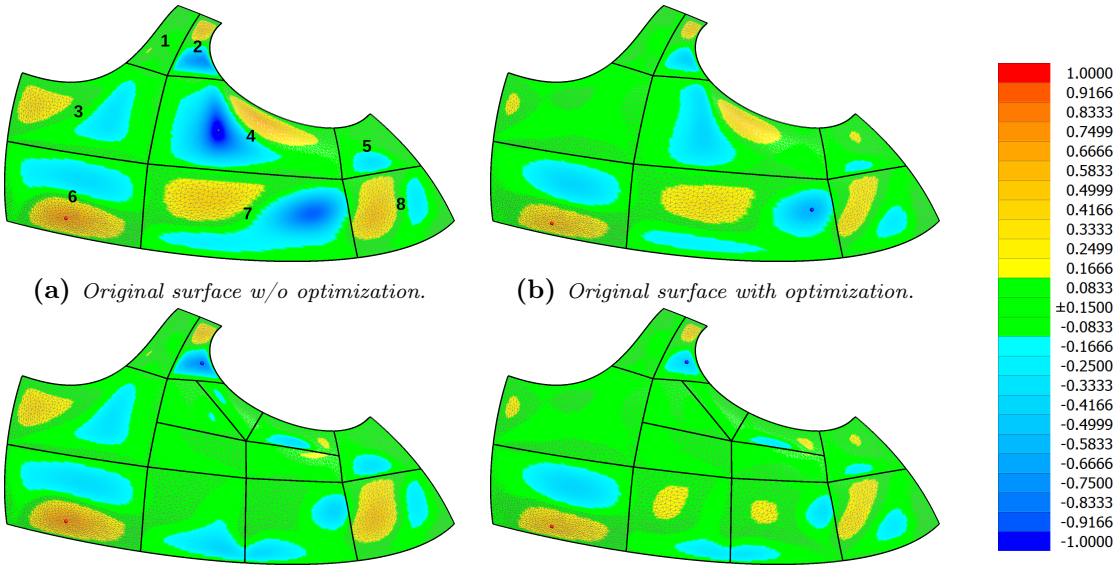


Figure 2.19: *Cell-based subdivision on a model of a shoe last. The color legend shows absolute deviation values; the bounding box diagonal is 410 units long.*

two deviation maps: the initial patch, and the optimized one. Accuracies improve, as shown in Table 2.1.



(a) *Original surface w/o optimization.* **(b)** *Original surface with optimization.*

(c) *Subdivided surface w/o optimization.* **(d)** *Subdivided surface with optimization.*

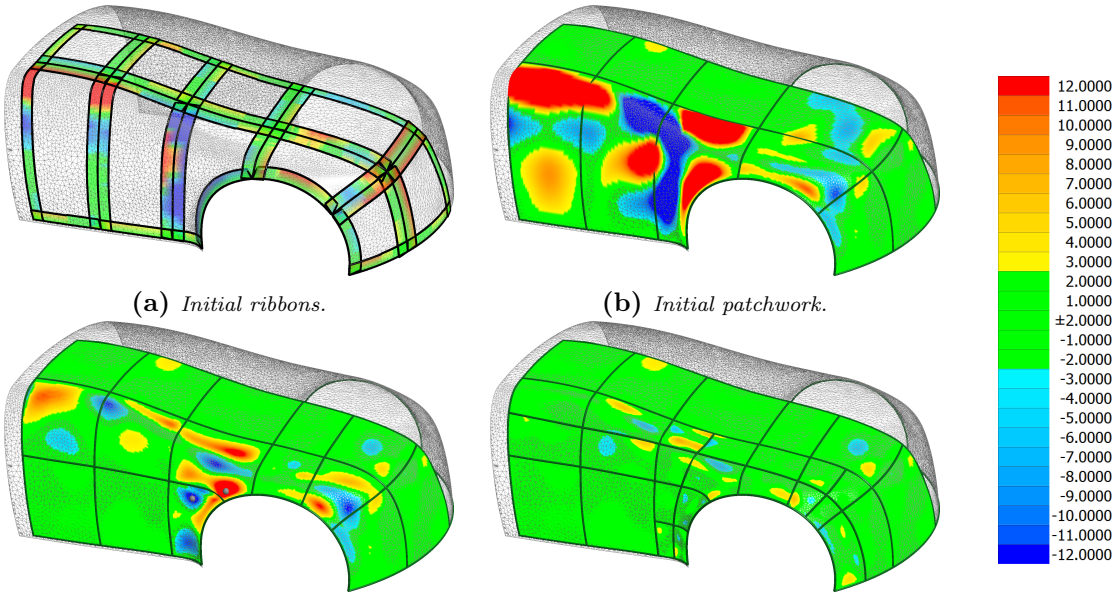
Figure 2.20: *Subdivision of a sheet metal part, with T-nodes and a split. The color legend shows absolute deviation values; the bounding box diagonal is 270 units long.*

Figures 2.20c and 2.20d show the deviation map of a refined structure – patch #4 and patch #7 have been subdivided. As both of these patches are inaccurate, their boundary is also subdivided, and a new X-node is inserted. Boundaries 2–4, 3–4, and 4–5 are sufficiently accurate, so we retain and inherit the ribbons from patches #2, #3, and #5, and create T-nodes accordingly, when patch #4 is subdivided by a central split. For simplicity's sake, planar bounding surfaces were generated throughout this refinement. The refined structure is more accurate, and it can be further improved by optimization (see Table 2.1).

Finally, Figure 2.21 shows the front part of a concept car. The first image depicts the initial ribbons of a sparse network that have been refined in two steps. The deviations decrease as the number of I-patches increases, see Table 2.2.

Model	Average deviation	Maximum deviation
Original w/o optimization (Fig. 2.20a)	0.055%	0.399%
Original with optimization (Fig. 2.20b)	0.035%	0.226%
Split w/o optimization (Fig. 2.20c)	0.041%	0.283%
Split with optimization (Fig. 2.20d)	0.029%	0.175%

Table 2.1: *Quantitative results corresponding to Figure 2.20.*



(a) *Initial ribbons.* **(b)** *Initial patchwork.*

(c) *First refinement step.* **(d)** *Second refinement step.*

Figure 2.21: *Steps of adaptive refinement on a concept car model. The color legend shows absolute deviation values; the bounding box diagonal is 4400 units long.*

Model	# of patches	Average deviation	Maximum deviation
Original (Fig. 2.21b)	10	0.140%	3.000%
After 1 step (Fig. 2.21c)	15	0.027%	0.300%
After 2 steps (Fig. 2.21d)	28	0.016%	0.120%

Table 2.2: *Quantitative results corresponding to Figure 2.21.*

2.4.5 Summary

I have elaborated a method to approximate discrete meshes with patchworks of implicit surfaces. Boundary curves of patches are assigned automatically or manually, the scheme supports a general topology curvenet, with multi-sided patches and permitting T-nodes. For approximating the input points, the least squares method based on the faithful distance metric is used. A refinement method is given, in which the network can be locally refined, only having to subdivide particular patches; the scheme is shown to prevent the loss of

smooth connections anywhere.

The outline of the algorithm can be found in Algorithm 3. These results have been published in [Sipos et al., 2022a].

Algorithm 3: I-patchwork approximating mesh.

```
Input: mesh; low resolution vertex graph
for all vertices of the graph do
    extract normal vector from mesh
end
for all edges of the graph do
    create a bounding surface using the average of the corner normals
    intersect the bounding surface with the mesh, and extract a polyline
    if corner normals are compatible (Eq. 2.3) then
        create a Liming-ribbon, approximating the polyline
    else
        create an I-loft ribbon, approximating the polyline
    end
    if error is above tolerance then
        mark boundary for subdivision
    end
end
for all regions of the graph do
    compute center point
    compute initial weights based on equal contribution
    run constrained optimization on weights
    if error is above tolerance or there are boundaries marked for subdivision then
        for boundaries marked for subdivision do
            insert a midpoint and extract normal from the mesh
            create two new ribbon surfaces
        end
        for other boundaries do
            insert a T-node, retaining the ribbon surface
        end
        subdivide the patch with central split
    end
end
Result: I-patches G^1-connected and approximating the input mesh
```

Conclusion

In this dissertation, I have presented new results related to implicit surfacing in Computer Aided Geometric Design. My primary interest was to explore finite, multi-sided implicit patches that are capable of defining not only regular but freeform shapes, as well. My results extend our knowledge about I-patches and demonstrate interesting applications with several case studies.

In the first part of the thesis, I have proposed new implicit schemes, including I-lofts and corner I-patches which can be used in 3D geometric modeling and built upon in further research. I have also described new evaluation methods for I-patches including the rational form that brought up a new distance-based interpretation, and the faithful distance function that proved to provide much better distance estimation than the former solutions.

In the second part, I have presented three practical algorithms using I-patches. Polyhedral design and setback vertex blending can be used in Computer-Aided Design in cases, when smooth surfaces in implicit representation defined by control polyhedra are required. My mesh approximation technique is useful when a discrete mesh needs to be represented by a collection of smoothly connected implicit patches.

The main results of this dissertation have already been published in the journal papers [Sipos et al., 2020], [Sipos et al., 2022a], and [Sipos, 2023]. These have also been presented at international ([Sipos et al., 2020], [Sipos et al., 2021]) and domestic ([Sipos, 2022a; Sipos et al., 2022b; Sipos and Salvi, 2021; Sipos, 2022b]) conferences. My work contributed to a research project that has been supported by the Hungarian Scientific Research Fund: OTKA, No. 124727, Modeling general topology free-form surfaces in 3D.

Acknowledgements

I would like to first and foremost thank my supervisor, Prof. Tamás Várady for accompanying me through the journey of this research, often going above and beyond the call of duty to ensure that I remain on track. I am also extremely grateful to my colleagues and paper co-authors Péter Salvi, who also helped me a lot in proper academic writing, software prototype implementation and figure creation; and Márton Vatikus, with whom the discussions about reviewing the literature were also invaluable. Special thanks to my MSc thesis advisor Gábor Valasek, who first introduced me into the world of CAGD and has driven me to pursue a PhD in this field.

I thank the Department of Control Engineering and Information Technology for providing a positive work environment throughout my years there. My work on this research has been supported by the Hungarian Scientific Research Fund: OTKA, No. 124727, "Modeling general topology free-form surfaces in 3D". Many images in this thesis have been generated by the Sketches prototype system (ShapEx Ltd, Budapest), I am thankful for the help of György Karikó with the software development.

I would like to also mention further individuals, discussions with whom shaped this research, including visiting professor Alyn Rockwood, and my (former) fellow students Csaba Bálint, Róbert Bán, Stefánia Hartner and István Kovács. I am also thanking my current employer Formlabs for the flexibility provided during the finalization of this thesis.

Bibliography

C. L. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. *ACM Transactions on Graphics (TOG)*, 11(1):61–91, 1992a. doi:10.1145/102377.120081.

C. L. Bajaj and I. Ihm. Smoothing polyhedra using implicit algebraic splines. *ACM SIGGRAPH Computer Graphics*, 26(2):79–88, 1992b. doi:10.1145/142920.134014.

C. L. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics (TOG)*, 14(2):103–133, 1995. doi:10.1145/221659.221662.

R. Bán and G. Valasek. First order signed distance fields. In *Eurographics (Short Papers)*, pages 33–36, 2020. doi:10.2312/egs.20201011.

J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982. doi:10.1145/357306.357310.

J. Bloomenthal, C. L. Bajaj, J. Blinn, B. Wyvill, M.-P. Cani, A. P. Rockwood, and G. Wyvill. *Introduction to implicit surfaces*. Morgan Kaufmann, 1997. ISBN 1558602335.

I. C. Braid. Non-local blending of boundary models. *Computer-Aided Design*, 29(2):89–100, 1997. ISSN 0010-4485. doi:10.1016/S0010-4485(96)00038-3.

E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided design*, 10(6):350–355, 1978. doi:10.1016/0010-4485(78)90110-0.

F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. doi:10.1016/j.cagd.2004.09.004.

P. Charrot and J. A. Gregory. A pentagonal surface patch for computer aided geometric design. *Computer Aided Geometric Design*, 1(1):87–94, 1984. ISSN 0167-8396. doi:10.1016/0167-8396(84)90006-2.

G. Chávez and A. P. Rockwood. Marching surfaces: Isosurface approximation using G^1 multi-sided surfaces. *arXiv preprint arXiv:1502.02139*, 2015. doi:10.48550/arXiv.1502.02139.

D. Dekkers, K. Van Overveld, and R. Golsteijn. Combining CSG modeling with soft blending using Lipschitz-based implicit surfaces. *The Visual Computer*, 20(6):380–391, 2004. doi:10.1007/s00371-002-0198-3.

D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978. doi:10.1016/0010-4485(78)90111-2.

G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann Publishers Inc., 5th edition, 2002. ISBN 1-55860-737-4.

M. S. Floater. Generalized barycentric coordinates and applications. *Acta Numerica*, 24: 161–214, 2015. doi:10.1017/S0962492914000129.

T. Garrity and J. Warren. Geometric continuity. *Computer Aided Geometric Design*, 8(1): 51–65, 1991. doi:10.1016/0167-8396(91)90049-H.

O. Gourmel, L. Barthe, M.-P. Cani, B. Wyvill, A. Bernhardt, M. Paulin, and H. Grasberger. A gradient-based implicit blend. *ACM Transactions on Graphics (TOG)*, 32(2):1–12, 2013. doi:10.1145/2451236.2451238.

J. C. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996. doi:10.1007/s003710050084.

E. Hartmann. Blending of implicit surfaces with functional splines. *Computer-Aided Design*, 22(8):500–506, 1990. doi:10.1016/0010-4485(90)90066-L.

E. Hartmann. Implicit G^n-blending of vertices. *Computer Aided Geometric Design*, 18(3): 267–285, 2001. doi:10.1016/S0167-8396(01)00030-9.

E. Hartmann and Y. Y. Feng. On the convexity of functional splines. *Computer Aided Geometric Design*, 10(2):127–142, 1993. doi:10.1016/0167-8396(93)90016-V.

C. Hoffmann and J. Hopcroft. Automatic surface generation in computer aided design. *The Visual Computer*, 1(2):92–100, 1985. doi:10.1007/BF01898351.

T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of Hermite data. *ACM Transactions on Graphics (TOG)*, 21(3):339–346, 2002. doi:10.1145/566654.566586.

K. Kato. Generation of N-sided surface patches with holes. *Computer-Aided Design*, 23 (10):676–683, 1991. ISSN 0010-4485. doi:10.1016/0010-4485(91)90020-W.

M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, pages 61–70, 2006. doi:10.2312/SGP/SGP06/061-070.

M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. MIT Press, 2019. ISBN 978-0-26-203942-0.

R. Krasauskas. Toric surface patches. *Advances in Computational Mathematics*, 17(1): 89–113, 2002. ISSN 1572-9044. doi:10.1023/A:1015289823859.

F. Lekien and J. Marsden. Tricubic interpolation in three dimensions. *International Journal for Numerical Methods in Engineering*, 63(3):455–471, 2005. doi:10.1002/nme.1296.

J. Li, J. Hoschek, and E. Hartmann. G^{n-1}-functional splines for interpolation and approximation of curves, surfaces and solids. *Computer Aided Geometric Design*, 7(1-4): 209–220, 1990. doi:10.1016/0167-8396(90)90032-M.

R. A. Liming. Conic lofting of streamline bodies. *Aircraft Engineering and Aerospace Technology*, 19(7):222–228, 1947. doi:10.1108/eb031528.

C. T. Loop and T. D. DeRose. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics*, 8(3):204–234, 1989. ISSN 0730-0301. doi:10.1145/77055.77059.

C. T. Loop and T. D. DeRose. Generalized B-spline surfaces of arbitrary topology. In *17th Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pages 347–356. ACM, 1990. ISBN 0-89791-344-2. doi:10.1145/97879.97917.

W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. doi:10.1145/37402.37422.

G. Lukács, R. Martin, and D. Marshall. Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. In *European Conference on Computer Vision*, pages 671–686, 1998. doi:10.1007/BFb0055697.

N. Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999. doi:10.1080/10867651.1999.10487501.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. doi:10.1145/3503250.

T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*, 41(4):102:1–102:15, 2022. doi:10.1145/3528223.3530127.

A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995. doi:10.1007/bf02464333.

G. I. Pasko, A. A. Pasko, and T. L. Kunii. Bounded blending for function-based shape modeling. *IEEE Computer Graphics and Applications*, 25(2):36–45, 2005. doi:10.1109/MCG.2005.37.

D. Pellis, M. Kilian, F. Dellinger, J. Wallner, and H. Pottmann. Visual smoothness of polyhedral surfaces. *ACM Transactions on Graphics (TOG)*, 38(4):1–11, 2019. doi:10.1145/3306346.3322975.

J. Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7: 221–246, 1991. doi:10.1007/BF01888155.

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

A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2): 157–160, 1973. doi:10.1093/comjnl/16.2.157.

A. Rockwood and K. Gao. SuperD: conceptual 3D modeling on mobiles. In *ACM SIGGRAPH 2018 Appy Hour*, pages 1–2. 2018. doi:10.1145/3213779.3213787.

A. P. Rockwood. The displacement method for implicit blending surfaces in solid models. *ACM Transactions on Graphics (TOG)*, 8(4):279–297, 1989. doi:10.1145/77269.77271.

S. D. Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18 (2):109–144, 1982. doi:10.1016/0146-664X(82)90169-1.

M. Sabin. Transfinite surface interpolation. In *Mathematics of Surfaces VI*, pages 517–534. Clarendon Press, 1996. ISBN 0-19-851198-1.

P. Salvi. A multi-sided generalization of the C^0 Coons patch. In *Proceedings of the Workshop on the Advances of Information Technology*, pages 110–111, 2020. doi:10.48550/arXiv.2002.11347.

T. W. Sederberg. Piecewise algebraic surface patches. *Computer Aided Geometric Design*, 2(1-3):53–59, 1985. doi:10.1016/0167-8396(85)90007-X.

D. Seyb, A. Jacobson, D. Nowrouzezahrai, and W. Jarosz. Non-linear sphere tracing for rendering deformed signed distance fields. *ACM Transactions on Graphics (TOG)*, 38 (6):1–12, 2019. doi:10.1145/3355089.3356502.

Á. Sipos. Corner-based implicit patches. In *Proceedings of the 13th Conference of PhD Students in Computer Science*, pages 12–16. SZTE, 2022a. doi:10.48550/arXiv.2207.00806.

Á. Sipos. A generalization of tangent-based implicit curves. In *Proceedings of the Workshop on the Advances of Information Technology*, pages 79–82. BME, 2022b. doi:10.48550/arXiv.2304.04577.

Á. Sipos. Corner-based implicit patches. *Acta Cybernetica*, 2023. doi:10.14232/actacyb.299598.

Á. Sipos and P. Salvi. Creating good quality meshes from smooth implicit surfaces. In *Proceedings of the Workshop on the Advances of Information Technology*, pages 47–51. BME, 2021. doi:10.48550/arXiv.2204.06396.

Á. Sipos, T. Várady, P. Salvi, and M. Vaitkus. Multi-sided implicit surfacing with I-patches. *Computers & Graphics*, 90:29–42, 2020. doi:10.1016/j.cag.2020.05.009.

Á. Sipos, T. Várady, and P. Salvi. Approximating triangular meshes by implicit, multi-sided surfaces. In *Proceedings of the CAD Conference*, pages 236–240. CAD Solutions, 2021. doi:10.14733/cadconfP.2021.236-240.

Á. Sipos, T. Várady, and P. Salvi. Approximating triangular meshes by implicit, multi-sided surfaces. *Computer-Aided Design and Applications*, 19(5):1015–1028, 2022a. doi:10.14733/cadaps.2022.1015-1028.

Á. Sipos, T. Várady, and P. Salvi. Implicit representations for multi-sided surface patches. In *Proceedings of the Tenth Hungarian Conference on Computer Graphics and Geometry*, pages 1–8. NJSZT, 2022b.

J. Szörfi and T. Várady. Polyhedral design with concave and multi-connected faces. In *Proceedings of the Tenth Hungarian Conference on Computer Graphics and Geometry*, pages 28–35. NJSZT, 2022.

G. Taubin. Distance approximations for rasterizing implicit curves. *ACM Transactions on Graphics (TOG)*, 13(1):3–42, 1994. doi:10.1145/174462.174531.

V. Vad, B. Csébfalvi, P. Rautek, and E. Gröller. Towards an unbiased comparison of CC, BCC, and FCC lattices in terms of prealiasing. *Computer Graphics Forum*, 33(3):81–90, 2014. doi:10.1111/cgf.12364.

T. Várady and C. M. Hoffmann. Vertex blending: problems and solutions. In *Conference on Mathematical Methods for Curves and Surfaces*, pages 501–527. Vanderbilt University, 1998. ISBN 0-8265-1315-8.

T. Várady and A. P. Rockwood. Geometric construction for setback vertex blending. *Computer-Aided Design*, 29(6):413–425, 1997. doi:10.1016/S0010-4485(96)00070-X.

T. Várady, R. R. Martin, and J. Cox. Reverse engineering of geometric models – an introduction. *Computer-Aided Design*, 29(4):255–268, 1997. ISSN 0010-4485. doi:10.1016/S0010-4485(96)00054-1.

T. Várady, P. Benkő, G. Kós, and A. P. Rockwood. Implicit surfaces revisited – I-patches. In *Geometric Modelling*, pages 323–335. Springer, 2001. doi:10.1007/978-3-7091-6270-5_19.

T. Várady, A. P. Rockwood, and P. Salvi. Transfinite surface interpolation over irregular n-sided domains. *Computer Aided Design*, 43(11):1330–1340, 2011. ISSN 0010-4485. doi:10.1016/j.cad.2011.08.028.

T. Várady, P. Salvi, and G. Karikó. A multi-sided Bézier patch with a simple control structure. *Computer Graphics Forum*, 35(2):307–317, 2016. ISSN 1467-8659. doi:10.1111/cgf.12833.

J. Warren. Free-form blending: A technique for creating piecewise implicit surfaces. In *Topics in surface modeling*, pages 3–21. SIAM, 1992. doi:10.1137/1.9781611971644.ch1.

V. Weiss, L. Andor, G. Renner, and T. Várady. Advanced surface fitting techniques. *Computer Aided Geometric Design*, 19(1):19–42, 2002. ISSN 0167-8396. doi:10.1016/S0167-8396(01)00086-3.

G. Xu, H. Huang, and C. L. Bajaj. C^1 modeling with A-patches from rational trivariate functions. *Computer Aided Geometric Design*, 18(3):221–243, 2001. doi:10.1016/S0167-8396(01)00018-8.

C. Zanni, M. Gleicher, and M.-P. Cani. N-ary implicit blends with topology control. *Computers & Graphics*, 46:1–13, 2015. doi:10.1016/j.cag.2014.09.012.

P. Zhou and W.-H. Qian. Polyhedral vertex blending with setbacks using rational S-patches. *Computer Aided Geometric Design*, 27(3):233–244, 2010. doi:10.1016/j.cagd.2010.01.001.

C.-G. Zhu, R.-H. Wang, X. Shi, and F. Liu. Functional splines with different degrees of smoothness and their applications. *Computer-Aided Design*, 40(5):616–624, 2008. doi:10.1016/j.cad.2008.02.006.

Appendix A

Proof of Liming's method properties

Lemma 1. *Let $\widetilde{Q}$ be a quadratic implicit curve, $\widetilde{L_1}, \widetilde{L_2}$ two distinct tangent lines of it, $\widetilde{C}$ the secant through the points of tangency. Then, $\exists \lambda \in (0;1), \omega \in \mathbb{R}$ s.t.*

$$(1-\lambda) \cdot L_1 \cdot L_2 - \lambda \cdot C^2 \equiv \omega \cdot Q. \tag{A.1}$$

Proof. Let (x, y) be a point s.t. $Q(x, y) = 0$, $L_1(x, y) \neq 0$, $L_2(x, y) \neq 0$. Let

$$\lambda := \frac{L_1(x, y) \cdot L_2(x, y)}{L_1(x, y) \cdot L_2(x, y) + C^2(x, y)} \tag{A.2}$$

meaning that this point is on our curve as well.

A general quadratic implicit curve is of the form

$$ax^2 + bxy + cy^2 + dx + ey + f, \tag{A.3}$$

having 6 coefficients, although multiplying them with the same scalar does not change the curve.

Fitting a curve on the two tangential conditions, and the additional requirement of interpolating (x, y) means a homogeneous linear system of 5 equations, as according to [Bajaj and Ihm, 1992a] a tangential constraint in 2D can be described with the following equations ((x, y) is the point to interpolate, (dx, dy) is the prescribed gradient).

$$F(x, y) = 0 \tag{A.4}$$

$$\mathbf{d}_x \cdot \partial_y F(x, y) - \mathbf{d}_y \cdot \partial_x F(x, y) = 0 \tag{A.5}$$

Now, apply this to the unknown quadratic curve $F(x, y) = a \cdot x^2 + b \cdot xy + c \cdot y^2 + d \cdot x + e \cdot y + f$, which should interpolate point (x_1, y_1) with gradient (m_1, n_1), point (x_2, y_2) with

gradient (m_2, n_2) and point (x_3, y_3) (with no specified gradient). (Points (x_1, y_1), (x_2, y_2), and (x_3, y_3) are all distinct.)

The partial derivatives are

$$\partial_x F(x,y) = 2a \cdot x + b \cdot y + d \tag{A.6}$$
$$\partial_y F(x,y) = 2c \cdot y + b \cdot x + e \tag{A.7}$$

Thus, the equation system will be

$$ax_1^2 + bx_1y_1 + cy_1^2 + dx_1 + ey_1 + f = 0 \tag{A.8}$$
$$ax_2^2 + bx_2y_2 + cy_2^2 + dx_2 + ey_2 + f = 0 \tag{A.9}$$
$$ax_3^2 + bx_3y_3 + cy_3^2 + dx_3 + ey_3 + f = 0 \tag{A.10}$$
$$m_1 \cdot (2cy_1 + bx_1 + e) - n_1 \cdot (2ax_1 + by_1 + d) = 0 \tag{A.11}$$
$$m_2 \cdot (2cy_2 + bx_2 + e) - n_1 \cdot (2ax_2 + by_2 + d) = 0 \tag{A.12}$$

These are 5 independent equations for (a, b, c, d, e, f), so the solution space is $6 - 5 = 1$ dimensional.

This means that any two solutions are each other multiplied by a scalar.

□

Theorem 1. *Let $\widetilde{L_i}\,(i = 1, 2, 3, 4)$ be four distinct tangent lines, $\mathbf{P}_i\,(i = 1, 2, 3, 4)$ the points of tangency. Let $\widetilde{C_1}$ be a line through $\mathbf{P}_1$ and $\mathbf{P}_2$, $\widetilde{C_2}$ through $\mathbf{P}_3$ and $\mathbf{P}_4$. Then, the family of two-sided I-patches*

$$I_{\mathbf{w}} := w_1 \cdot L_1 \cdot L_2 \cdot C_2^2 + w_2 \cdot L_3 \cdot L_4 \cdot C_1^2 + w \cdot C_1^2 \cdot C_2^2 \tag{A.13}$$

fulfills the tangential conditions. Moreover, if there exists a quadratic curve satisfying the constrained tangents, the I-patch $\widetilde{I}$ reproduces it with well-chosen coefficients.

Proof. The first statement trivially follows from the I-patch properties: for example in $\mathbf{P}_1$, using that $L_1(\mathbf{P}_1) = 0$ and $C_1(\mathbf{P}_1) = 0$;

$$I_{\mathbf{w}}(\mathbf{P}_1) = 0 + 0 + 0 = 0 \tag{A.14}$$

and

$$\begin{aligned}\nabla I_{\mathbf{w}}(\mathbf{P}_1) = c_1(\mathbf{P}_1) \cdot \nabla L_1(\mathbf{P}_1) + \mathbf{v}_1(\mathbf{P}_1) \cdot L_1(\mathbf{P}_1) + \\ + c_2(\mathbf{P}_1) \cdot 2 \cdot C_1(\mathbf{P}_1) \cdot \nabla C_1(\mathbf{P}_1) + \mathbf{v}_2(\mathbf{P}_1) \cdot C_1^2(\mathbf{P}_1),\end{aligned} \tag{A.15}$$

where c_1, c_2 are real-valued, $\mathbf{v}_1, \mathbf{v}_2$ are vector-valued functions. Using the constraints, we get

$$\nabla I_{\mathbf{w}}(\mathbf{P}_1) = c_1(\mathbf{P}_1) \cdot \nabla L_1(\mathbf{P}_1) + 0 + 0 + 0, \tag{A.16}$$

meaning that the point is on the curve, and the direction of the gradient is the same as that of the gradient of the line, so $\widetilde{L_1}$ is indeed a tangential line. □

For the second statement, let Q be the equation of the target quadratic curve,

$$R_1 := L_1 L_2 \tag{A.17}$$

$$R_2 := L_3 L_4 \tag{A.18}$$

$$S_{i,\omega,\lambda} := \omega((1-\lambda)R_i + \lambda C_i^2) \tag{A.19}$$

Due to Lemma 1, $\exists \omega_1, \lambda_1 : S_{1,\omega_1,\lambda_1} \equiv Q$, similarly $\exists \omega_2, \lambda_2 : S_{2,\omega_2,\lambda_2} \equiv Q$

Now, on the one hand

$$\begin{aligned} S_{1,\omega_1,\lambda_1}C_2^2 + S_{2,\omega_2,\lambda_2}C_1^2 = \omega_1(1-\lambda_1)R_1C_2^2 + \omega_1\lambda_1 C_1^2 C_2^2 + \\ + \omega_2(1-\lambda_2)R_2C_1^2 + \omega_2\lambda_2 C_1^2 C_2^2 \end{aligned} \tag{A.20}$$

On the other hand

$$S_{1,\omega_1,\lambda_1}C_2^2 + S_{2,\omega_2,\lambda_2}C_1^2 = Q \cdot (C_1^2 + C_2^2) \tag{A.21}$$

This means that with

$$w_1 = \omega_1(1-\lambda_1) \tag{A.22}$$

$$w_2 = \omega_2(1-\lambda_2) \tag{A.23}$$

$$w_0 = \omega_1\lambda_1 + \omega_2\lambda_2; \tag{A.24}$$

$$I_{[w_1,w_2,w_0]} \equiv Q \cdot (C_1^2 + C_2^2), \tag{A.25}$$

where $(C_1^2 + C_2^2)$ is only zero in the intersection point of the lines, thus $\widetilde{I}$ and $\widetilde{Q}$ have to be the same curve.

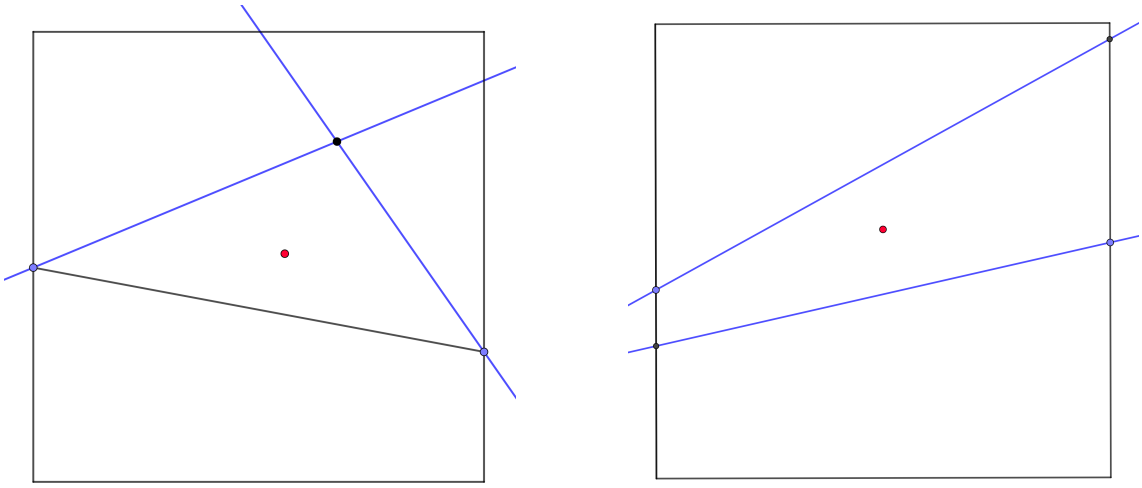
□

Appendix B

Auxiliary point calculation

Side coefficients in the examples are calculated such that the boundary curve interpolates a given point. That point is computed with either the triangle rule or the tetragon rule.

There are two cases: the two corner planes and the bounding face either have an intersection point inside the face, or it does not. If that point exists, we have a triangle (Figure B.1a) and we take its centroid as the point to interpolate. Otherwise, we have a tetragon (Figure B.1b), by intersecting each corner plane with the opposite corner's cube edge. We then take the centroid of this polygon.



(a) *Triangle rule.* **(b)** *Tetragon rule.*

Figure B.1: *Rules for computing interpolated points. Blue points and lines represent corner points and planes. The red point is the centroid to be interpolated.*

The side coefficient can then easily be calculated by evaluating Equation 1.45 for each w_i, as the other side coefficients do not affect the current boundary.

Appendix C

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 C.1. 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:

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

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.

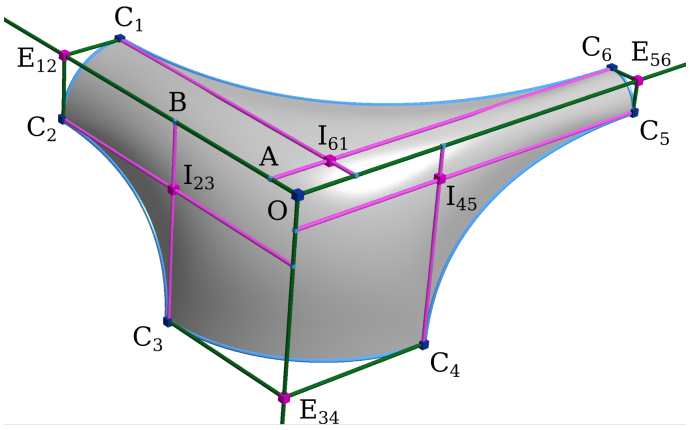


Figure C.1: *Construction of a setback vertex blend.*

Appendix D

Tessellating I-patches

I-Patches with known corner points can be triangulated the following way.

First, a base mesh is created, using an auxiliary parametric surface, which interpolates the corner points and the boundaries. The simplest method is to use a generalized barycentric [Floater, 2015] combination of the corner points over a regular n-sided domain:

$$p(u, v) = \sum_{i=1}^{n} \lambda_i \mathbf{P}_i, \tag{D.1}$$

where $\sum_{i=1}^{n} \lambda_i = 1$, and $[u, v] = \sum_{i=1}^{n} \lambda_i \cdot [u_i, v_i]$. Here, $[u_i, v_i]$ are the domain coordinates of the corners of the regular n-gon; $\mathbf{P}_i$ are the corner points of the surface.

Alternatively, as the implicit surface can be highly curved, a simple transfinite parametric surface can be used, like the multi-sided C^0 Coons patch [Salvi, 2020] which is defined by its boundary curves. In the case of using Liming-surfaces and/or planes as ribbons and bounding surfaces, this is easy to solve since in those cases the boundary curve always has a quadratic rational parametric representation. However, if they do not have a simple parametric representation, for example in the case of I-segments, they can be given as dense polylines evaluated via tessellating each implicit boundary curve. As the mesh vertices will be projected onto the surface, the slight error arising from the inexact curves will not meaningfully affect the final result.

These surfaces can then be elegantly tessellated by dividing the n-sided polygonal domain into triangles like in Figure D.1.

When projecting the points of the mesh onto the isosurface, we must ensure continuous connection to neighboring patches. For that, we need the points on the edge of the patch to remain on the face of the cell after projection. We also want triangles to change their size relative to each other as few as possible.

We achieve this by defining a *projection direction* (a 3-dimensional vector) attribute for each vertex which will prescribe the line, along which the point will be projected. The

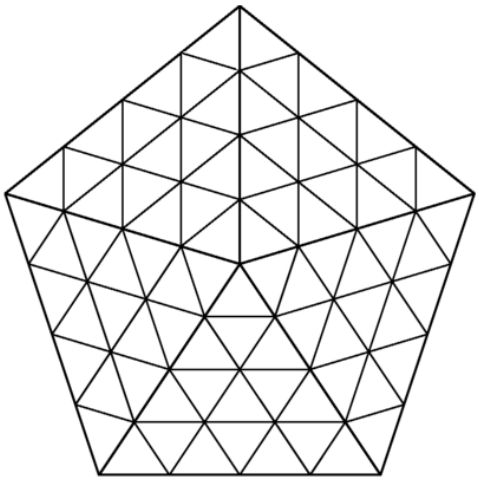


Figure D.1: *Triangulation of a pentagon.*

projection direction is prescribed in the corners of the patch and is then interpolated along the mesh using barycentric coordinates. In the corners, they need to point in the same direction from the isosurface (e.g. into the positive half-space). See the 2-dimensional example in Figure D.2.

If the boundary points have to be moved (i.e. we are not using a base patch with exact boundary points), then at corner points the direction shall be set as the direction of the edge of the cell there. This ensures that in edge points the direction is inside the bounding plane, as it is the weighted sum of two edges in that plane (all other barycentric coordinates are zeros there). Thus, the projected points will also remain there.

See Figure D.3 for a visualization of these direction vectors.

Now we can use ray marching to find the isosurface point for each vertex. By checking the sign of the implicit function in the starting point, we know in which direction the isosurface lays, relative to the interpolated direction. In the end, we replace the mesh positions with the projected positions, getting a mesh representing the piece of the original isosurface inside the cell. Corner points are not moved, and if the boundary curves were already approximated, boundary points stay in place as well.

The above process unfortunately does not guarantee that correct results are obtained. If the isosurface has high curvature variation or large shape artifacts, the resulting mesh might contain abrupt jumps or overlapping triangles. We prevent this by checking if the angle between the ray and the gradient of the surface remains under a prescribed threshold. If not, the resulting mesh is rejected, and we can try to solve the problem by subdividing the cell. This typically indicates, however, that the quality of the implicit surface would not be acceptable.

The process however generally produces better-quality meshes for surface analysis than volume-based methods like Marching Cubes, see Figures D.4 and D.5 for comparison.

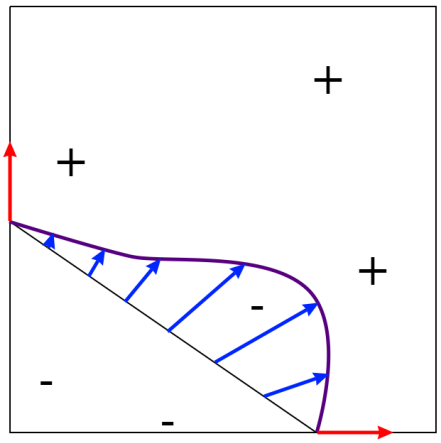


Figure D.2: *2D example of projection direction interpolation. Purple: implicit surface, red: prescribed directions, blue: interpolated directions. Signs denote the sign of the implicit function in that region.*

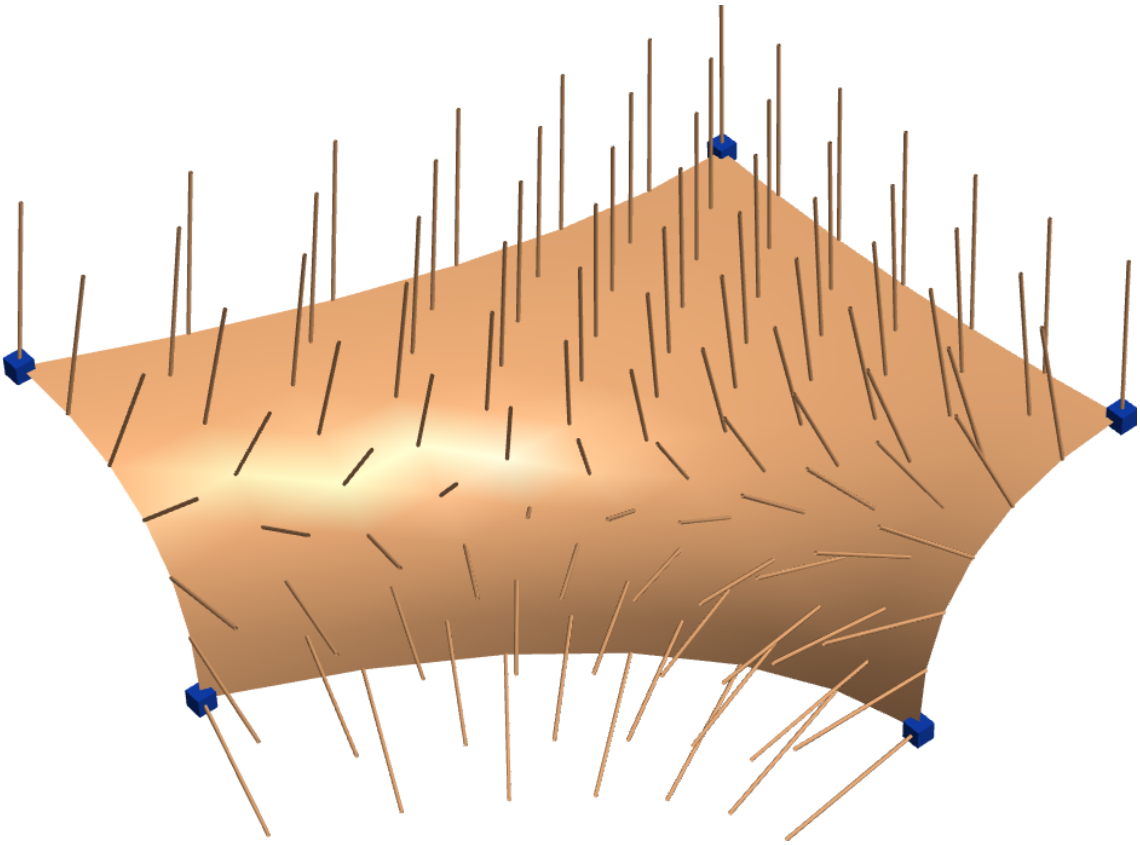


Figure D.3: *Projection directions visualized on the surface of a C^0 Coons patch.*

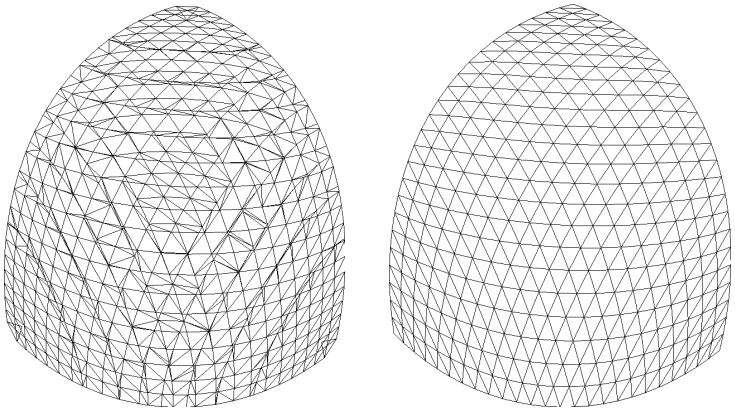


Figure D.4: *Wireframe visualization of a Marching Cubes mesh (left) and a projected mesh (right).*

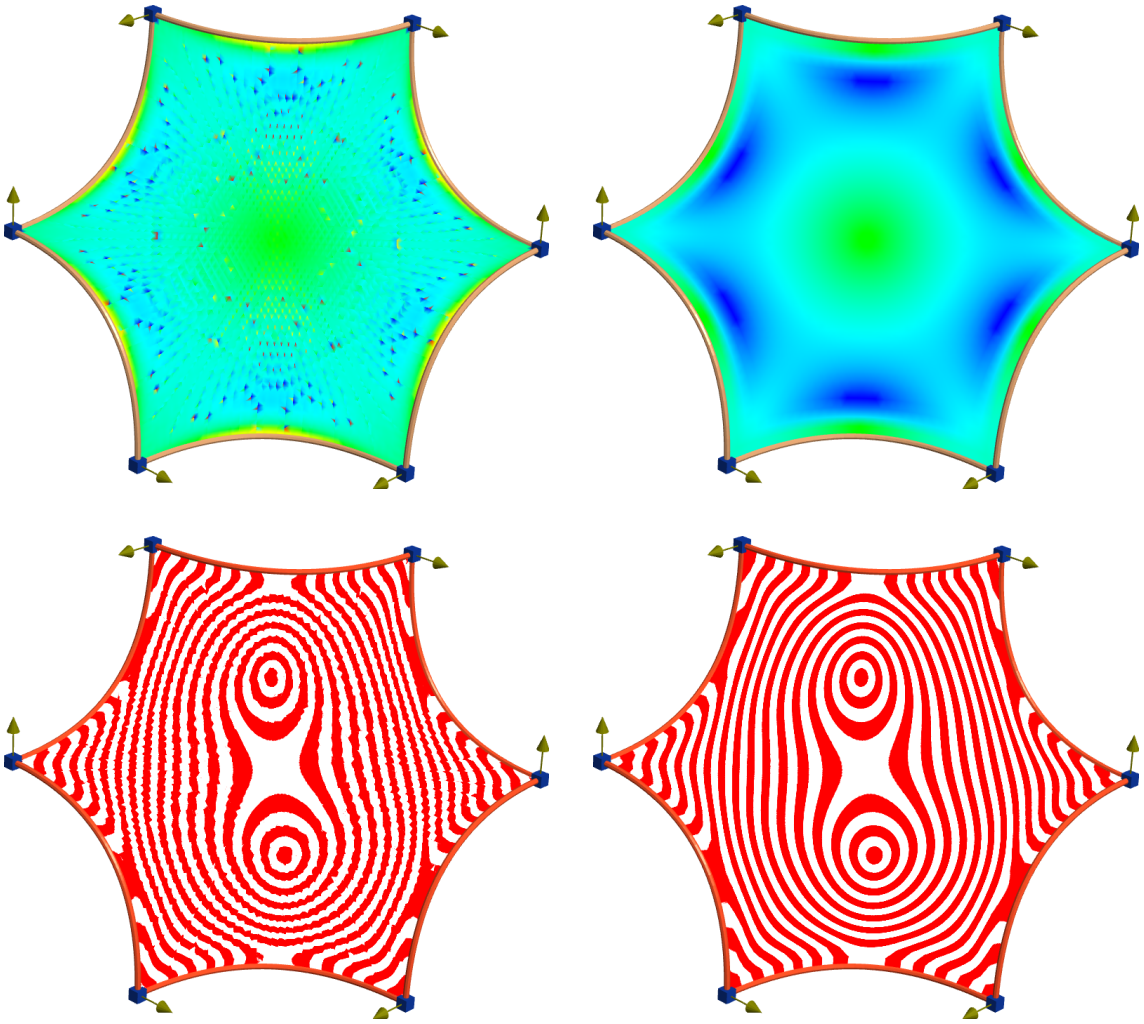


Figure D.5: *Comparing a Marching Cubes mesh to our approach (6-sided patch). Top row: mean curvature, bottom row: isophote lines; both approximated from the meshes.*