Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehrstuhl für Informatik 8
Prof. Dr. Leif Kobbelt

Master Thesis

Spline Surface Fitting

Janis Born
Matricular Number: 297818

July 2015

Primary Reviewer: Prof. Dr. Leif Kobbelt
Secondary Reviewer: Prof. Dr. David Bommes

Hiermit versichere ich, diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht zu haben.

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Aachen, July 6, 2015

Janis Born

Contents

Chapter 1

Introduction

The most significant discernable feature of a three-dimensional object is the shape of its surface. For the digital processing, manipulation and storage of surface data, a variety of representations are used, most notably polygon meshes where the object surface is approximated by a network of a finite number of polygons. Polygon meshes, especially triangle meshes, are ubiquitous in geometry processing and rendering and are a common interchange format between different applications and algorithms.

In the context of 3D modeling and computer aided design (*CAD*), polygon meshes are less popular due to their inability to accurately represent smoothly curved surfaces. Instead, spline representations that construct shapes from piecewise polynomial surfaces are used. Spline representations are desirable for a number of reasons:

- *Abstraction capacity*: Where polygon meshes might require hundreds or thousands of faces to approximate a rounded feature, spline surfaces can accurately represent such features often by a single patch with a handful of control points, hence offering more intuitive editing capabilities to a designer.

- *Built-in smoothness*: Spline surfaces retain their desired smoothness properties when their control points are modified.

- *Compact storage*: Due to the sparse number of control points, spline surfaces typcially achieve a lower memory footprint than polygonal representations.

Spline representations have been used for a variety of applications including industrial, automotive and aerospace design, architecture, entertainment, etc.

Since manual design is often tedious, the automatic conversion of real-world objects into spline surfaces has attracted significant research attention. The long-term research goal is the creation of an automated reconstruction pipeline, allowing to scan an existing input object, extract its surface data, and automatically generate a suitable spline surface representation ready for editing, manufacturing or rendering.

Spline surfaces representing complex objects are typically assembled from individual, often rectangular surface patches that smoothly join together. The configuration of these four-sided patches on the surface, called the *quad layout*, can have a significant impact on the resulting shape and editing flexibility of the surface. Existing automatic spline reconstruction techniques have mostly ignored the importance of creating meaningful quad layouts and have resorted to ad-hoc methods that lead to patch configurations which are mostly unrelated to the shape of the input object. However, in recent years, powerful algorithms for the automatic generation of high-quality geometry-aware quad layouts have emerged.

In this work, we examine how traditional spline reconstruction techniques can benefit from the recent advancements in quad layout generation. Additionally, we propose a novel spline surface model suitable for the approximation using quad layouts with non-uniform patch sizes.

We begin our report with a brief review of the fundamental definitions of polynomial curves and surfaces and their smoothness conditions (Chapter 2), followed by a general overview and discussion of the spline fitting framework (Chapter 4). Then, we present our suggested novel spline model, termed G^1 Bézier splines. We analyze the degrees of freedom of the model and derive explicit construction rules (Chapter 5). Next, we turn to the topic of surface smoothing, where we discuss the discretization and weighting of the commonly used thin plate energy (Chapter 6). Finally, we evaluate the approximation quality of different layout generation methods, spline models, and approximation strategies (Chapter 7).

Chapter 2

Fundamentals

This section gives a brief summary of the definitions of Bézier curves and surfaces and establishes some properties and notational conventions that we will use throughout the rest of this thesis. For a more in-depth introduction to the topic of curve and surface representation, we refer to the literature, e. g. [PBP02; Far02; HL96].

2.1 Affine Space

In all subsequent geometric constructions, we do not really care about the particular representation of points and vectors. Even if these objects are typically represented by elements from $\mathbb{R}^d$, we like to remain oblivious of the origin or the dimensionality of the underlying coordinate system. Therefore, in the following, we just assume that all points are from some *affine space* $\mathbb{A}$ over some *vector space* $\mathbb{V}$.

For any number of points $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{A}$, taking an affine combination with weights $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ produces again a point in $\mathbb{A}$:

$$\sum_{i=1}^{n} \alpha_i = 1 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i \cdot \mathbf{p}_i = \mathbf{q} \in \mathbb{A} \; .$$

The difference of two points $\mathbf{p}, \mathbf{q} \in \mathbb{A}$ produces a vector $\mathbf{p} - \mathbf{q} = \mathbf{v} \in \mathbb{V}$, and, conversely, adding a vector to a point yields another point, i. e. $\mathbf{p} + \mathbf{v} = \mathbf{q} \in \mathbb{A}$.

In some cases, e. g. where we need to compute distances between points, we remember that $\mathbb{A}$ and $\mathbb{V}$ are represented by some $\mathbb{R}^d$ and use the respective operations from the real coordinate space.

2.2 Bézier Representation

The foundation for the definition of Bézier curves and surfaces is the representation of polynomials in Bernstein form. The i-th *Bernstein polynomial* of degree n is a univariate, real-valued polynomial $B_i^n : \mathbb{R} \to \mathbb{R}$ given by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \, .$$

For a given degree n, the $n+1$ Bernstein polynomials $\{B_0^n, \ldots, B_n^n\}$ form a basis of the space $\mathbb{P}_n$ of all polynomials of degree n, called the *Bernstein basis*. Some well-known properties of the Bernstein basis are:

(B1) The basis functions are non-negative, i. e., $B_i^n(t) \geq 0$.

(B2) The basis functions form a partition of unity: $\sum_{i=0}^{n} B_i^n(t) = 1$.

(B3) At $t = 0$ and $t = 1$, only the first and last basis functions attain a value of 1 while all other basis functions are zero, i. e. $B_0^n(0) = 1$ and $B_n^n(1) = 1$.

(B4) $B_i^n(t) = 0$ for $i > n$ or $i < 0$, due to the definition of the binomial coefficient.

Furthermore, Bernstein polynomials have simple closed-form solutions for their derivatives, which reduce to differences of lower-order Bernstein polynomials, i. e.

$$\frac{\mathrm{d}}{\mathrm{d}t} B_i^n(t) = n \left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t) \right) \tag{2.1}$$

and their integrals over $[0, 1]$, which are just constants

$$\int_0^1 B_i^n(t) \, \mathrm{d}t = \frac{1}{n+1} \, .$$

2.2.1 Bézier Curves

A *Bézier curve* of degree n is a univariate polynomial $\mathbf{b} \colon [0, 1] \to \mathbb{A}$ defined in terms of the Bernstein basis, i. e.

$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t) \mathbf{b}_i \, . \tag{2.2}$$

The $n + 1$ coefficients $\mathbf{b}_0, \ldots, \mathbf{b}_n \in \mathbb{A}$ uniquely define the curve $\mathbf{b}(t)$ and are called *Bézier points*.

Due to (B1) and (B2), for any $t \in [0,1]$, $\mathbf{b}(t)$ is a convex combination of the Bézier points $\mathbf{b}_0, \ldots, \mathbf{b}_n$. Furthermore, due to (B3), the start and end points of the curve interpolate the first and last Bézier points, i. e. $\mathbf{b}(0) = \mathbf{b}_0$ and $\mathbf{b}(1) = \mathbf{b}_n$.

The derivative of an n-th degree Bézier curve w. r. t. its parameter t results again in a Bézier curve with degree reduced by one:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{b}(t) &= \sum_{i=0}^{n} \mathbf{b}_i \frac{\mathrm{d}}{\mathrm{d}t} B_i^n(t) \\
&\overset{(2.1)}{=} n \sum_{i=0}^{n} \mathbf{b}_i B_{i-1}^{n-1}(t) - n \sum_{i=0}^{n} \mathbf{b}_i B_i^{n-1}(t) \\
&\overset{(B4)}{=} n \sum_{i=0}^{n-1} \mathbf{b}_{i+1} B_i^{n-1}(t) - n \sum_{i=0}^{n-1} \mathbf{b}_i B_i^{n-1}(t) \\
&= n \sum_{i=0}^{n-1} \Delta\mathbf{b}_i B_i^{n-1}(t) \ .
\end{aligned}$$

Here the $\Delta\mathbf{b}_i$ indicate forward differences of the Bézier points, i. e.

$$\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i \ .$$

Conversely, integrating a Bézier curve raises its degree by one and is computed as

$$\int \mathbf{b}(t)\ \mathrm{d}t = \frac{1}{n+1} \sum_{i=0}^{n+1} \Sigma\mathbf{b}_i B_i^{n+1}(t) + \mathbf{c}$$

where the $\Sigma\mathbf{b}_i$ denote partial sums of the Bézier points, i. e.

$$\Sigma\mathbf{b}_i = \sum_{k=0}^{i-1} \mathbf{b}_k$$

and $\mathbf{c}$ is some integration constant. For the definite definite integral over the domain $[0,1]$, the expression further simplifies to

$$\int_0^1 \mathbf{b}(t)\ \mathrm{d}t = \frac{1}{n+1} \sum_{i=0}^{n} \mathbf{b}_i \ .$$

2.2.2 Scaled Bernstein Basis

While the Bernstein form of a Bézier curve is the generally accepted standard representation, sometimes a transformation to another basis can help to simplify some computations.

One example for such an alternative basis is the *scaled Bernstein basis* [FR88], which is obtained by a simple component-wise scaling of the standard Bernstein basis. A Bézier curve of degree n, given in standard Bernstein form (as in (2.2)), is written in scaled Bernstein form as

$$\mathbf{b}(t) = \sum_{i=0}^{n} \tilde{B}_i^n(t)\tilde{\mathbf{b}}_i$$

where

$$\tilde{B}_i^n(t) = t^i(1-t)^{n-i} \quad \text{and} \quad \tilde{\mathbf{b}}_i = \binom{n}{i}\mathbf{b}_i \ .$$

Note how the the binomial coefficients no longer appear in the basis functions $\tilde{B}_0^n, \ldots, \tilde{B}_n^n$ (also called *scaled Bernstein polynomials*), but are moved into the coefficients $\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_n$. Following Peters [Pet94], we employ the shorthand notation using square brackets for polynomials in scaled Bernstein form, i. e.

$$\left[\tilde{\mathbf{b}}_0, \ldots, \tilde{\mathbf{b}}_n\right] = \sum_{i=0}^{n} \tilde{B}_i^n(t)\tilde{\mathbf{b}}_i = \mathbf{b}(t) \ .$$

The benefit of this scaled Bernstein representation is that the multiplication of two polynomials simplifies to a discrete convolution [SR03]: Given two real-valued polynomials $a \in \mathbb{P}_n$, $b \in \mathbb{P}_m$ by their coefficients in scaled Bernstein form

$$a = [a_0, \ldots, a_n] \quad \text{and} \quad b = [b_0, \ldots, b_m] \ ,$$

their product $c = ab$ is a polynomial of degree $n+m$ and is computed by summing up shifted copies of one polygon scaled by the coefficients of the other one, i. e.

$$\begin{aligned} c = a * b &= \sum_{i=0}^{n} a_i \cdot \mathrm{pad}(b, i, n-i) \\ &= b * a = \sum_{j=0}^{m} b_j \cdot \mathrm{pad}(a, j, m-j) \end{aligned}$$

where $\mathrm{pad}(x, p, q)$ constructs a new polynomial by padding the coefficient list of a given polynomial x with p zeroes from the left and q from the right, i. e.

$$\mathrm{pad}([x_0, \ldots, x_k], p, q) = [\underbrace{0, \ldots, 0}_{p}, x_0, \ldots, x_k, \underbrace{0, \ldots, 0}_{q}] \ .$$

As a more concrete example, for $n = 3$ and $m = 2$, this becomes

$$\begin{aligned} c = \quad & [\ c_0,\ c_1,\ c_2,\ c_3,\ c_4,\ c_5\ ] \\ = a_0\ & [\ b_0,\ b_1,\ b_2,\ 0,\ 0,\ 0\ ] + \\ a_1\ & [\ 0,\ b_0,\ b_1,\ b_2,\ 0,\ 0\ ] + \\ a_2\ & [\ 0,\ 0,\ b_0,\ b_1,\ b_2,\ 0\ ] + \\ a_3\ & [\ 0,\ 0,\ 0,\ b_0,\ b_1,\ b_2\ ] \\ = b_0\ & [\ a_0,\ a_1,\ a_2,\ a_3,\ 0,\ 0\ ] + \\ b_1\ & [\ 0,\ a_0,\ a_1,\ a_2,\ a_3,\ 0\ ] + \\ b_2\ & [\ 0,\ 0,\ a_0,\ a_1,\ a_2,\ a_3\ ]\ . \end{aligned}$$

This way of multiplying polynomials is useful both for analytical derivations and for the implementation in software. The multiplication of polynomials in scaled Bernstein form works analogously when one of the two polynomial factors is vector-valued instead of scalar.

We can use this representation to derive a very simple degree elevation scheme for Bézier curves. This is facilitated by the fact that the factor 1 can also be written as a scaled Bernstein polynomial in two coefficients:

$$1 = t + (1 - t) = 1 \cdot t^1(1 - t)^0 + 1 \cdot t^0(1 - t)^1 = [1\ 1]\ .$$

Since $[1\ 1]$ is, in fact, the multiplicative identity, a Bézier curve $\mathbf{b}$ remains unchanged by a multiplication $[1\ 1]\,\mathbf{b} = 1 \cdot \mathbf{b} = \mathbf{b}$. However, if we compute the product $[1\ 1]\,\mathbf{b}$ using the convolution $[1\ 1] * [\mathbf{b}_0, \ldots, \mathbf{b}_n]$, we get

$$\begin{aligned} [1\ 1]\,\mathbf{b} = [\ & \mathbf{b}_0,\ \mathbf{b}_1,\ \mathbf{b}_2,\ \ldots,\ \mathbf{b}_{n-1}\ \ \mathbf{b}_n,\ \ 0\ ] + \\ [\ & 0,\ \ \mathbf{b}_0,\ \mathbf{b}_1,\ \ldots,\ \mathbf{b}_{n-2}\ \mathbf{b}_{n-1},\ \mathbf{b}_n\ ] \end{aligned}$$

which represents the polynomial $\mathbf{b}$ using $n + 2$ coefficients, i. e., w. r. t. a basis of degree $n + 1$. This leaves us with a very simple way to raise the degree of polynomials in scaled Bernstein form: We obtain the new list of coefficients by just summing up pairs of adjacent original coefficients:

$$[1\ 1]\,\mathbf{b} = [\mathbf{b}_0, \mathbf{b}_0 + \mathbf{b}_1, \mathbf{b}_1 + \mathbf{b}_2, \ldots, \mathbf{b}_{n-2} + \mathbf{b}_{n-1}, \mathbf{b}_{n-1} + \mathbf{b}_n, \mathbf{b}_n]\ .$$

2.2.3 Tensor Product Bézier Surfaces

By multiplying the elements of two Bernstein bases in a tensor product fashion, one obtains a bivariate Bézier representation describing surfaces instead of curves.

We define a *tensor product Bézier surface* of degree (n, m) as a bivariate polynomial function $\mathbf{b}\colon [0,1] \times [0,1] \to \mathbb{A}$, given by

$$\begin{aligned}\mathbf{b}(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v)\mathbf{b}_{ij} \\ &= \begin{pmatrix} B_0^n(u) & \cdots & B_n^n(u)\end{pmatrix} \begin{pmatrix} \mathbf{b}_{00} & \cdots & \mathbf{b}_{0m} \\ \vdots & \ddots & \vdots \\ \mathbf{b}_{n0} & \cdots & \mathbf{b}_{nm}\end{pmatrix} \begin{pmatrix} B_0^m(v) \\ \vdots \\ B_m^m(v)\end{pmatrix} .\end{aligned}$$

Since the products of Bernstein polynomials form again a basis, $\mathbf{b}$ is uniquely defined by its $(n+1)(m+1)$ coefficients $\mathbf{b}_{ij}$ (for $i = 0, \ldots, n$, and $j = 0, \ldots, m$) which are also referred to as the Bézier points or the *Bézier grid* of $\mathbf{b}$.

Note that by fixing one of the two parameters, e. g. $v = v^*$, one obtains a single iso-parametric curve on the surface:

$$\begin{aligned}\mathbf{b}^*(u) = \mathbf{b}(u, v^*) &= \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v^*)\mathbf{b}_{ij} \\ &= \sum_{i=0}^{n} B_i^n(u) \sum_{j=0}^{m} B_j^m(v^*)\mathbf{b}_{ij} \\ &= \sum_{i=0}^{n} B_i^n(u)\mathbf{b}_i^*\end{aligned}$$

Due to the end-point interpolation property of Bézier curves, the coefficients of the boundary curves obtained by fixing u or v to either 0 or 1 can be read off directly from the Bézier grid, i. e.

$$\mathbf{b}(u,0) = \sum_{i=0}^{n} B_i^n(u)\mathbf{b}_{i0} , \qquad \mathbf{b}(u,1) = \sum_{i=0}^{n} B_i^n(u)\mathbf{b}_{im} , \tag{2.3}$$

$$\mathbf{b}(0,v) = \sum_{j=0}^{m} B_j^m(u)\mathbf{b}_{0j} , \qquad \mathbf{b}(1,v) = \sum_{j=0}^{m} B_j^m(u)\mathbf{b}_{nj} . \tag{2.4}$$

Analogously to curves, taking partial derivatives of a Bézier surface produces once again a Bézier surface:

$$\frac{\partial}{\partial u}\mathbf{b}(u,v) = n\sum_{i=0}^{n-1}\sum_{j=0}^{m} B_i^{n-1}(u) B_j^m(v)\Delta^{10}\mathbf{b}_{ij} , \tag{2.5}$$

$$\frac{\partial}{\partial v}\mathbf{b}(u,v) = m\sum_{i=0}^{n}\sum_{j=0}^{m-1} B_i^n(u) B_j^{m-1}(v)\Delta^{01}\mathbf{b}_{ij} , \tag{2.6}$$

where $\Delta^{10}\mathbf{b}_{ij}$ and $\Delta^{01}\mathbf{b}_{ij}$ denote forward differences in the rows and columns of the Bézier grid, respectively, i. e.

$$\Delta^{10}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j} \ , \tag{2.7}$$
$$\Delta^{01}\mathbf{b}_{i,j} = \mathbf{b}_{i,j+1} - \mathbf{b}_{i,j} \ . \tag{2.8}$$

Note how taking derivatives decreases the degree in the respective dimension by one. Repeated applications of Eqs. (2.5) and (2.6) lead to the general formula for higher-order mixed derivatives,

$$\frac{\partial^{p+q}}{\partial u^p \partial v^q}\mathbf{b}(u,v) = \frac{n!}{(n-p)!}\frac{m!}{(m-q)!}\sum_{i=0}^{n-p}\sum_{j=0}^{m-q} B_i^{n-p}(u) B_j^{m-q}(v) \Delta^{pq}\mathbf{b}_{ij} \ , \tag{2.9}$$

where the iterated forward differences $\Delta^{pq}\mathbf{b}_{ij}$ are defined recursively in terms of repeated applications of Eqs. (2.7) and (2.8):

$$\Delta^{p+1,q}\mathbf{b}_{i,j} = \Delta^{10}\Delta^{p,q}\mathbf{b}_{i,j} \ ,$$
$$\Delta^{p,q+1}\mathbf{b}_{i,j} = \Delta^{01}\Delta^{p,q}\mathbf{b}_{i,j} \ .$$

For brevity, we also sometimes write the partial derivatives of Bézier surfaces using subscripts, e. g.

$$\mathbf{b}_{\mathbf{u}} = \frac{\partial}{\partial u}\mathbf{b} \ , \quad \mathbf{b}_{\mathbf{v}} = \frac{\partial}{\partial v}\mathbf{b} \ , \quad \mathbf{b}_{\mathbf{uv}} = \frac{\partial^2}{\partial u \partial v}\mathbf{b} \ , \text{ etc.}$$

2.3 Continuous Surfaces

Since they are polynomials, Bézier surfaces are smooth in their interior. However, when joining multiple surfaces to a piecewise Bézier shape (which is necessary in many modeling scenarios, cf. Section 2.3.3), discontinuities can arise where the individual pieces meet. Such discontinuities can be apparent as gaps, creases, or sudden changes in curvature along the boundaries. Where such features are not desired, special care must be taken during the construction of the surface elements. In this section, we show the derivation of some well-known sufficient conditions to achieve a certain degree of smoothness across the subdomain boundaries of piecewise Bézier surfaces.

For now, we will focus on a single boundary between two adjacent surface parts. In the following, we suppose the two surfaces are given by two tensor product Bézier surfaces $\mathbf{p}(u,v)$ and $\mathbf{q}(u,v)$, both of degree (n,m). We will examine continuity across their common boundary $\mathbf{p}(1,v)$ and $\mathbf{q}(0,v)$. Since we are ultimately interested in the Bézier points near the boundary, we use the symmetric re-labeling $\tilde{\mathbf{p}}_{i,j} = \mathbf{p}_{n-i,j}$ for the Bézier grid of $\mathbf{p}$. This setup is illustrated in Fig. 2.1.

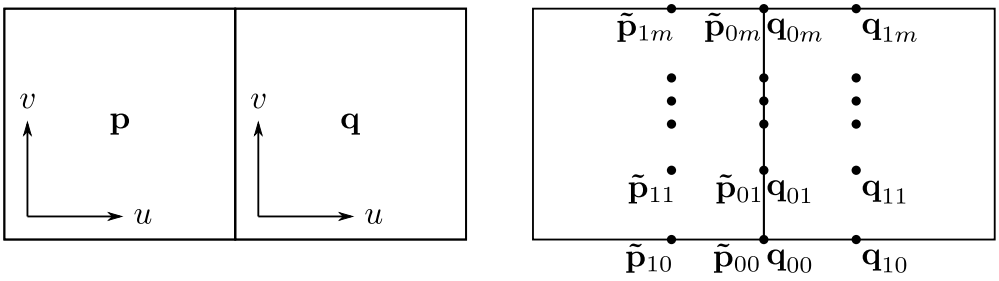


Figure 2.1: Two Bézier surfaces $\mathbf{p}$ and $\mathbf{q}$ are joined smoothly along a common boundary. Left: Parametrization. Right: Labeling of the control points.

2.3.1 C^r Continuity

We say that the surfaces $\mathbf{p}$ and $\mathbf{q}$ join with *parametric continuity* of order r (or C^r continuity) if and only if along the common boundaries $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$, all partial derivatives up to order r are equal:

$$\frac{\partial^{s+t}}{\partial u^s \partial v^t}\mathbf{p}(1, v) = \frac{\partial^{s+t}}{\partial u^s \partial v^t}\mathbf{q}(0, v) \tag{2.10}$$

for all $s, t \in \mathbb{N}_0$ such that $s + t \leq r$ and $v \in [0, 1]$.

For C^0 continuity (i. e., $r = 0$), Eq. (2.10) implies that the two boundary curves must be equal, i. e., $\mathbf{p}(1, v) = \mathbf{q}(0, v)$, which, according to Eq. (2.4), expands to

$$\sum_{j=0}^{m} B_j^m(v)\tilde{\mathbf{p}}_{0j} = \sum_{j=0}^{m} B_j^m(v)\mathbf{q}_{0j} \ . \tag{2.11}$$

Due to the linear independence of the Bernstein polynomials, the two sums in Eq. (2.11) are equal if and only if all coefficients are equal, i. e. $\tilde{\mathbf{p}}_{0j} = \mathbf{q}_{0j}$ for all $i = 0, \ldots, m$. This simple construction rule for the Bézier grids ensures that $\mathbf{p}$ and $\mathbf{q}$ join without gaps to a connected composite surface. Still, visible creases can occur where $\mathbf{p}$ and $\mathbf{q}$ meet.

Note that by fixing the boundary curves $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$ to be equal, all partial derivatives in the parameter direction of this curve are also equal, i. e.

$$\mathbf{p}(1, v) = \mathbf{q}(0, v) \quad \Rightarrow \quad \frac{\partial^k}{\partial v^k}\mathbf{p}(1, v) = \frac{\partial^k}{\partial v^k}\mathbf{q}(0, v) \quad \forall k \in \mathbb{N}_0 \ . \tag{2.12}$$

The creases along the boundary can be remedied by moving on to C^1 continuity (i. e., $r = 1$). In addition to the C^0 conditions, Eq. (2.10) now implies additional conditions for the first-order partial derivatives:

$$\frac{\partial}{\partial u}\mathbf{p}(1, v) = \frac{\partial}{\partial u}\mathbf{q}(0, v) \quad \text{and} \quad \frac{\partial}{\partial v}\mathbf{p}(1, v) = \frac{\partial}{\partial v}\mathbf{q}(0, v) \ . \tag{2.13}$$

Note that by enforcing C^0 continuity, the curves $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$ are equal and thus, due to Eq. (2.12), the second part of condition (2.13) already holds. For the remaining first condition, we take the derivatives of $\mathbf{p}$, $\mathbf{q}$ w. r. t. u, yielding the derivative surfaces $\mathbf{p_u}$, $\mathbf{q_u}$, as defined by Eq. (2.5). Evaluating these derivative surfaces along the common boundary according to (2.3), we obtain

$$\begin{aligned} \mathbf{p_u}(1, v) &= \mathbf{q_u}(0, v) \\ \sum_{j=0}^{m} B_j^m(v) \Delta^{10} \mathbf{p}_{n-1,j} &= \sum_{j=0}^{m} B_j^m(v) \Delta^{10} \mathbf{q}_{0,j} \ , \end{aligned}$$

which, again, due to the independence of the Bernstein basis functions, results in individual constraints on the coefficients for $j = 0, \ldots, m$:

$$\begin{aligned} \Delta^{10} \mathbf{p}_{n-1,j} &= \Delta^{10} \mathbf{q}_{0,j} \\ \mathbf{p}_{n,j} - \mathbf{p}_{n-1,j} &= \mathbf{q}_{1,j} - \mathbf{q}_{0,j} \\ \tilde{\mathbf{p}}_{0,j} - \tilde{\mathbf{p}}_{1,j} &= \mathbf{q}_{1,j} - \mathbf{q}_{0,j} \ . \end{aligned} \tag{2.14}$$

As noted above, the C^0 condition forces opposite boundary Bézier points to be equal. Let us refer to them by $\mathbf{c}_j = \tilde{\mathbf{p}}_{0,j} = \mathbf{q}_{0,j}$. Thus, Eq. (2.14) becomes

$$\mathbf{c}_j = \frac{1}{2} \tilde{\mathbf{p}}_{1,j} + \frac{1}{2} \mathbf{q}_{1,j} \ , \tag{2.15}$$

which implies a straightforward geometric interpretation for C^1 continuity: For each row $j = 0, \ldots, m$ of the Bézier grids, the three points $\tilde{\mathbf{p}}_{1,j}$, $\mathbf{c}_j$, and $\mathbf{q}_{1,j}$ must be collinear and $\mathbf{c}_j$ must split the line segment $\tilde{\mathbf{p}}_{1,j} \mathbf{q}_{1,j}$ in a $1 : 1$ ratio. Hence, we also refer to Eq. (2.15) as the *midpoint condition*.

The conditions derived above for C^0 and C^1 continuity are both necessary and sufficient.

2.3.2 G^r Continuity

While C^r continuity ensures that a piecewise surface is sufficiently smooth, it is often not practical for geometric modeling since it depends on a particular parametrization of the individual surface pieces. Since a surface with a given shape can be represented w. r. t. different parametrizations, it is possible that some of these representations are C^r while others are not. This calls for a more general notion of continuity which is independent of a particular parametrization. This notion is expressed by allowing reparametrizations of the adjacent surfaces to achieve a C^r joint and is known as G^r or *geometric continuity* (for an in-depth examination, see [Pet02]).

First, let us clarify what constitutes a *valid reparametrization*. Regarding two patches $\mathbf{p}$ and $\mathbf{q}$ which are to be joined with G^r continuity along their common boundaries $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$, two functions $\mathbf{a}, \mathbf{b} \colon \mathbb{R}^2 \to \mathbb{R}^2$ are valid reparametrizations of $\mathbf{p}$ and $\mathbf{q}$ if the following conditions hold:

- They map boundary points to boundary points, i. e. for any $v \in [0, 1]$,

  $$\mathbf{a}(1, v) = (1, v') \quad \text{and} \quad \mathbf{b}(0, v) = (0, v'')$$

  such that $v', v'' \in [0, 1]$.

- They never map interior points of one domain to the other side of the boundary, i. e. for all $(u, v) \in [0, 1]^2$,

  $$\mathbf{a}(u, v) = (u', v') \quad \text{and} \quad \mathbf{b}(u, v) = (u'', v'')$$

  such that $v' \leq 1$ and $v'' \geq 0$. This condition rules out cusps, i. e. the case of $\mathbf{p}$ and $\mathbf{q}$ folding back on each other along the boundary.

- They are injective and r times differentiable.

Now, the two surfaces $\mathbf{p}$ and $\mathbf{q}$ join with G^r continuity across their common boundaries $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$ if and only if there exist valid reparametrizations $\mathbf{a}$, $\mathbf{b}$ such that the reparametrized surfaces $\mathbf{p} \circ \mathbf{a}$ and $\mathbf{q} \circ \mathbf{b}$ join with C^r continuity:

$$\frac{\partial^{s+t}}{\partial u^s \partial v^t}(\mathbf{p} \circ \mathbf{a})(1, v) = \frac{\partial^{s+t}}{\partial u^s \partial v^t}(\mathbf{q} \circ \mathbf{b})(0, v) \tag{2.16}$$

for all $s, t \in \mathbb{N}_0$ such that $s + t \leq r$ and $v \in [0, 1]$.

Expanding Eq. (2.16) for $r = 1$ yields the three G^1 conditions

$$(\mathbf{p} \circ \mathbf{a})(1, v) = (\mathbf{q} \circ \mathbf{b})(0, v) \ , \tag{2.17}$$

$$\frac{\partial}{\partial u}(\mathbf{p} \circ \mathbf{a})(1, v) = \frac{\partial}{\partial u}(\mathbf{q} \circ \mathbf{b})(0, v) \ , \tag{2.18}$$

$$\frac{\partial}{\partial v}(\mathbf{p} \circ \mathbf{a})(1, v) = \frac{\partial}{\partial v}(\mathbf{q} \circ \mathbf{b})(0, v) \ . \tag{2.19}$$

Just like in the previous section, we immediately see that condition (2.19) always holds: $(\mathbf{p} \circ \mathbf{a})(1, v)$ and $(\mathbf{q} \circ \mathbf{b})(0, v)$ are curves parametrized by v. According to condition (2.17), these two curves are equal. Hence, all derviatives w. r. t. the curve parameter v are also equal and condition (2.19) holds.

Note how the two boundary curves $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$ need not be identical. For G^0 continuity (or higher), it suffices if $\mathbf{p}(1, v)$ and $\mathbf{q}(0, v)$ trace out the same curve but have different parametrizations. Although G^0 continuity hence offers increased freedom in designing the boundary curves, this is almost never

used in practice: It is much simpler to construct identical boundary curves by just choosing common Bézier points, thus restricting to C^0 instead of G^0 continuity. Additionally, this choice greatly simplifies the further derivation of G^1 constraints for the Bézier grids. This combination of C^0 and G^1 continuity is ubiquitous in the literature (although seldom spelled out explicitly) and we will follow suit in the following derivations. In our case, this means restricting the reparametrizations $\mathbf{a}$ and $\mathbf{b}$ to the identity map on the boundary curve, i. e.

$$\mathbf{a}(1,v) = (1,v)\ , \quad \mathbf{b}(0,v) = (0,v) \qquad \text{for } v \in [0,1]\ ,$$

thereby implying equality of $\mathbf{p}(1,v) = \mathbf{q}(0,v)$ due to (2.17).

Thus, just condition (2.18), i. e. the equality of the cross-boundary derivatives, remains to establish G^1 continuity. We obtain the derivative w. r. t. u by applying the chain rule for bivariate functions:

$$\begin{aligned}\frac{\partial}{\partial u}(\mathbf{p}\circ\mathbf{a})(1,v) &= \big((\mathbf{p_u}\circ\mathbf{a})(1,v) \quad (\mathbf{p_v}\circ\mathbf{a})(1,v)\big)\,\mathbf{a_u}(1,v)\\ &= \big(\mathbf{p_u}(1,v) \quad \mathbf{p_v}(1,v)\big)\,\mathbf{a_u}(1,v)\\ &= \mathbf{p_u}(1,v)\cdot\mathbf{a_u}(1,v)^{[1]} + \mathbf{p_v}(1,v)\cdot\mathbf{a_u}(1,v)^{[2]}\end{aligned}$$

where the bracketed superscripts indicate vector indices (i. e., scalar values). By applying analogous transformations to the right-hand side accordingly, condition (2.18) expands to

$$\begin{aligned}&\mathbf{p_u}(1,v)\cdot\mathbf{a_u}(1,v)^{[1]} + \mathbf{p_v}(1,v)\cdot\mathbf{a_u}(1,v)^{[2]}\\ =\ &\mathbf{q_u}(0,v)\cdot\mathbf{b_u}(0,v)^{[1]} + \mathbf{q_v}(0,v)\cdot\mathbf{b_u}(1,v)^{[2]}\ .\end{aligned} \tag{2.20}$$

Remembering that $\mathbf{p_v}(1,v) = \mathbf{q_v}(0,v)$, we can rearrange Eq. (2.20) to

$$\alpha(v)\cdot\mathbf{p_u}(1,v) - \beta(v)\cdot\mathbf{q_u}(0,v) + \gamma(v)\cdot\mathbf{p_v}(1,v) = 0 \tag{2.21}$$

where

$$\begin{aligned}\alpha(v) &= \mathbf{a_u}(1,v)^{[1]}\ ,\\ \beta(v) &= \mathbf{b_u}(0,v)^{[1]}\ ,\\ \gamma(v) &= \mathbf{a_u}(1,v)^{[2]} - \mathbf{b_u}(1,v)^{[2]}\ .\end{aligned}$$

Equation (2.21) expresses the G^1 condition in a very straightforward way: Instead of comparing two reparametrized surfaces, it directly relates the transversal ($\mathbf{p_u}$ and $\mathbf{q_u}$) and versal ($\mathbf{p_v}$) derivatives along the two surface boundaries by three scalar *connection functions* α, β, γ. These functions result from derivatives of the reparametrizations $\mathbf{a}$, $\mathbf{b}$ and hence have the following properties:

- They are $r - 1$ times differentiable.
- $\alpha(v) > 0$ and $\beta(v) > 0$ for $v \in [0, 1]$.

Thus, instead of finding suitable reparametrizations $\mathbf{a}$, $\mathbf{b}$, we can equivalently find suitable connection functions α, β, γ such that Eq. (2.21) holds in order to establish G^1 continuity of a joint.

Conversely, Eq. (2.21) can be used to derive sufficient G^1 conditions by fixing some connection functions α, β, γ. In particular, by fixing $\beta(v) = 1$, we get

$$\mathbf{q_u}(0, v) = \alpha(v) \cdot \mathbf{p_u}(1, v) + \gamma(v) \cdot \mathbf{p_v}(1, v) \ ,$$

which offers an intuitive geometric interpretation (Fig. 2.2): At every boundary point determined by a parameter v, the local tangent plane of the left patch is spanned by the partial derivatives $\mathbf{p_u}$ and $\mathbf{p_v}$. Now, the cross-boundary derivative of the right patch, $\mathbf{q_u}$, must lie in this tangent plane, i. e. be a linear combination of the tangent vectors.

By setting $\alpha(v) = \beta(v) = 1$ and $\gamma(v) = 0$, the G^1 condition reduces to $\mathbf{p_u}(1, v) = \mathbf{q_u}(0, v)$, i. e. the C^1 condition. Hence, C^1 continuity is a special case of G^1 continuity.

A common technique for deriving explicit G^1 constraints in terms of Bézier points is to prescribe fixed polynomial connection functions α, β, γ. Since the boundary tangent curves $\mathbf{p_u}$, $\mathbf{p_v}$ and $\mathbf{q_u}$ are polynomial curves (with their coefficients given by differences of Bézier points), the resulting G^1 condition is again a sum of polynomials. By raising the resulting polynomials to the same degrees and comparing coefficients, a set of conditions in terms of linear equalities of Bézier points is derived. To simplify these computations, it can be helpful to switch to a scaled Bernstein representation (see Section 2.2.2) to compute the polynomial multiplications and the degree elevation.

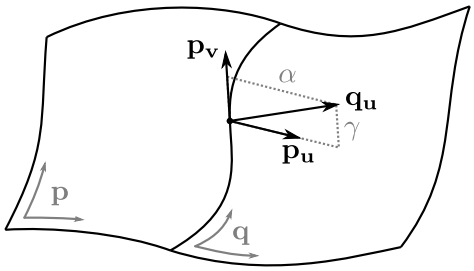
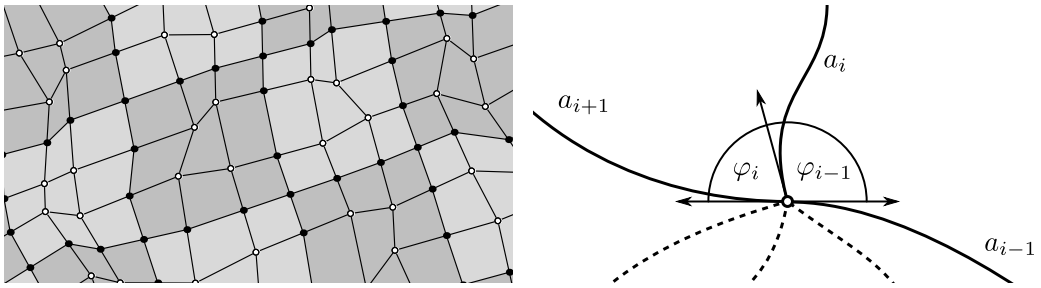


Figure 2.2: Geometric intuition for a G^1 joint: The patches $\mathbf{p}$ and $\mathbf{q}$ join with G^1 continuity if at each point on the boundary, $\mathbf{q_u}$ is a linear combination of $\mathbf{p_u}$ and $\mathbf{v_u}$ (with weights α, γ), i. e. lies in the local tangent plane of $\mathbf{p}$.



(a) **(b)**

Figure 2.3: (a): Bézier grids of a C^1 continuous spline surface. Solid dots are constrained by the midpoint condition. **(b)**: Three neighboring arcs emanating from a layout node. The tangent vectors corresponding to arcs a_{i-1} and a_{i+1} must form a $180°$ angle.

2.3.3 Non-Regular Layouts

In Section 2.3.1, we derived a simple construction rule for joining two adjacent Bézier surfaces with C^1 continuity: Each triplet of adjacent Bézier points that straddles the boundary must be in a collinear and equidistant configuration (according to the midpoint condition, cf. Eq. (2.15)). Using multiple Bézier surfaces, a smooth composite surface or *spline surface* can be assembled by enforcing this rule for all boundaries between individual surface pieces (see Fig. 2.3a). In 3D space, such a spline surface without self-intersections forms a 2-manifold.

A particular configuration of how Bézier surfaces are combined to a spline surface may be described by a *layout graph*, a graph embedded into the manifold surface whose faces (*patches*) correspond to the individual Bézier surfaces and whose edges (*arcs*) correspond to the subdomain boundaries. Accordingly, vertices (*nodes*) of the layout graph correspond to the corners of the Bézier surface domains. Since tensor product Bézier surfaces are four-sided, each patch is bordered by four arcs and the layout graph is also referred to as a *quad layout*.

It turns out that the possible layouts of non-degenerate Bézier spline surfaces with C^1 continuity are restricted to *regular quad layouts* where all interior nodes are *regular nodes*, i. e. have valence 4. This is a direct result of the collinearity condition for C^1 joints: Consider an interior node of valence n with incident arcs $a_0, \ldots, a_{n-1}$ (as shown in Fig. 2.3b). Each arc a_i corresponds to a boundary between two Bézier surfaces. If these two surfaces join C^1-continuous, their cross-boundary tangents must be collinear, which affects the two boundary curves associated with the arcs a_{i-1} and a_{i+1} to either side of a_i (indices taken modulo n). Since the region around the node is locally planar, the angles φ_{i-1} between a_{i-1} and a_i and φ_i between a_i and a_{i+1} must sum to a half circle: $\varphi_{i-1} + \varphi_i = \pi$.

Also due to planarity, all angles φ_i around the node form a full circle:

$$2\pi = \sum_{i=0}^{n-1} \varphi_i = \frac{1}{2} \sum_{i=0}^{n-1} (\varphi_{i-1} + \varphi_i) = \frac{1}{2} n\pi \ ,$$

and therefore $n = 4$.

The fact that C^1 spline surfaces must have regular quad layouts severely restricts their topology: For the layout of a closed surface without holes, the number of nodes v, arcs e, and patches f is related to the genus g of the surface by the Euler characteristic $v - e + f = 2(1 - g)$. Each arc corresponds to two half-arcs, $h = 2e$. Since we have a quad layout, each patch is surrounded by four half-arcs, i. e. $h = 4f$ and due to regularity, each node has four outgoing half-arcs: $h = 4v$. Thus, the left-hand side of the Euler characteristic cancels to 0, implying $g = 1$. This means that closed C^1 spline surfaces can only model objects with genus 1, i. e. torus topology.

To overcome this restriction, C^1 continuity is abandoned in favor of the weaker condition of G^1 continuity. Most importantly, G^1 continuity does not require all cross-boundary derivatives to be collinear and hence does not force all interior layout nodes to have valence 4. Interior nodes where more or fewer than 4 patches meet are referred to as *irregular nodes* (other common names from the literature are *extraordinary vertex* or N*-vertex*), and their incident arcs as *irregular arcs*. Quad layouts which contain irregular nodes are called *non-regular*.

Chapter 3

Related Work

Previous research that is relevant to our work can be roughly categorized into three fields: The mathematical representation of smooth surfaces (Section 3.1), the segmentation of surfaces into suitable parametric domains (Section 3.2), and the automatic approximation of 3D objects by a simplified surface model (Section 3.3).

3.1 Surface Representation

Bézier curves [Cas63; Béz67] and their bivariate generalization, tensor product Bézier surfaces [Boo62], provide a simple parametric representation of polynomial curves and surfaces. Joining multiple Bézier surfaces with C^r continuity produces smooth spline surfaces with a regular layout which are generalized by B-Spline surfaces [GR74]. NURBS surfaces [Ver75] further broaden this definition by allowing weighted Bézier points, thus providing a unified representation of both piecewise polynomial surfaces and conic sections. One restriction of NURBS, the strictly grid-like arrangement of knots in the parameter domain, is lifted by T-Splines [SZB+03] which allow partial rows and columns of knots, hence providing a more compact representation for tensor product spline surfaces.

All of the aforementioned representations have enjoyed broad popularity in both modeling and design applications over the past decades. Besides providing an intuitive framework for shape design, their appeal is due to their simple mathematical representations which lend themselves to efficient algorithms for construction, evaluation, and rendering. Furthermore, all these representations can be used to generate surfaces that are guaranteed to have a certain degree of continuity across their interior. Their common drawback, however, is their restriction to a regular, i. e. grid-like domain structure, making it impossible to model general objects of arbitrary topology using a single, closed surface. Hence, it is an ongoing

effort to generalize the concept of tensor product spline surfaces to non-regular domains, enabling the construction of surfaces with arbitrary topology while preserving the built-in continuity guarantees.

The family of representations known as *subdivision surfaces* abandons the idea of an explicit parametrization in favor of an algorithmic approach: Instead of describing the surface of an object by a bivariate function, subdivision surfaces are represented by an initial control mesh and a subdivision rule specifying how to refine the control mesh and how to update its vertex positions. Iterated applications of the subdivision rule produce a sequence of increasingly refined versions of the control mesh. Under certain conditions, this sequence converges towards a continuous limit surface. The crucial advantage of this approach comes from the invention of subdivision schemes which are not limited to regular quadrilateral control meshes, but can instead be used to subdivide arbitrary polygonal meshes. Prominent examples for such schemes are due to Doo and Sabin [DS78] and Catmull and Clark [CC78], generalizing biquadratic and bicubic B-Spline surfaces, respectively. Near irregular vertices of the input mesh, the limit surfaces of these methods were later shown to be G^1 continuous for Doo-Sabin subdivision and C^1 continuous for Catmull-Clark subdivision [Rei95].

A common objection against the use of subdivision surfaces is that it is impossible to explicitly evaluate points on the surface in the vicinity of irregular vertices. While it has been shown that subdivision surfaces *can* in fact be evaluated at arbitrary locations [Sta98], this evaluation scheme turns out to be mathematically involved and computationally expensive. In contrast, the evaluation of parametric surfaces (such as tensor product Bézier patches) is simple and efficient to compute. Therefore, an alternative approach is to imitate the limit surface of a subdivision process by a collection of Bézier surfaces. This was realized by Peters who introduced a family of construction methods known as *Surface Splines* [Pet94; Pet95a; Pet95b] which serve as a parametric approximation to Doo-Sabin subdivision surfaces. For a given polygonal control mesh of arbitrary topology, a few Doo-Sabin subdivision steps are applied, hence producing a mesh where all vertices have valence four. Then, instead of further iterating the subdivision procedure, the mesh surface is converted to a network of tensor product Bézier surfaces such that each vertex of the control mesh (also called *control point*) corresponds to one surface patch. The Bézier points of the individual surfaces are computed from convex combinations of the control point positions in a way that ensures a globally smooth surface. In particular, the individual Bézier patches join with C^1 continuity everywhere except around irregular vertices, where patches join G^1-continuously. Surface Splines maintain a low polynomial degree: Around irregular vertices, bicubic Bézier surfaces (degree $(3, 3)$) are used and in regular regions, they are biquadratic (degree $(2, 2)$) and locally form a uniform tensor product B-Spline surface. For a given control mesh connectivity, Surface Splines

form a basis for a vector space of continuous spline surfaces where each basis element corresponds to the region of influence of one control point, i. e. one vertex of the control mesh.

Unfortunately, Surface Splines must trade off between some desirable design properties. Specifically, there is a trilemma between (*a*) low polynomial degree; (*b*) tangent plane continuity everywhere; and (*c*) free positioning of the control points, of which at most two of these three properties can be fulfilled at the same time. Often, (*c*) is sacrificed in favor of (*a*) and (*b*), which is paid with some positional constraints on the control points around irregularities. Furthermore, it has proven difficult to generalize Surface Splines from their original formulation to higher degrees of continuity. Already for curvature (i. e. C^2 and G^2) continuity, the derivation of Bézier coefficients becomes quite involved and contains free parameters that lack an obvious default choice [Pet96].

3.2 Quad Layout Generation

Approximating the shape of a given object by a spline surface can be broken down into two major tasks: First, the connectivity of a suitable control mesh is determined. Then, a spline surface is generated from the control mesh and its geometry is optimized to capture the shape of the input object. If the resulting spline surface is a piecewise tensor product Bézier surface (as is the case in Surface Splines), the connectivity of the control mesh determines the quadrilateral layout graph of the composite surface. Due to this correspondence, a natural way to construct a control mesh is to inscribe the layout graph directly in the surface of the input object.

Segmenting the surface of an object into a quad layout has numerous applications beside spline surface reconstruction: Quad layouts serve as domain for surface parametrizations which are used e. g. for texture mapping, detail encoding or re-meshing. High-quality quad layouts adapt to the overall geometric structure of an object and tend to have arcs that align with the local curvature of the surface or run along creases or feature edges. At the same time, they tend to use a low number of irregular nodes which are placed in geometrically meaningful locations.

Automatic methods for the generation of quad layouts have been proposed in the context of different applications. For the purpose of spline surface fitting, Eck and Hoppe [EH96] use a decimation-based approach where a polygonal input surface is reduced to a coarse triangle mesh with an even number of faces. Then, pairs of neighboring triangles are merged to form an all-quadrilateral network. Another approach based on manifold harmonics is used for quad meshing [DBG+06]: By connecting adjacent extrema of eigenfunctions of the Laplace-Beltrami operator, a quadrilateral network known as the Morse-Smale complex

arises. A subsequent relaxation procedure optimizes the geometry of nodes and arcs to produce a smooth result.

While the above approaches produce valid quad layouts, their results do not adequately capture the geometry of the input object and tend to place layout nodes in implausible locations. For the generation of more geometry-aware quad layouts, parametrization-based techniques, such as the Periodic Global Parameterization method due to Ray *et al*. [RLL+06], have been used. While enabling the generation of quad layouts aligned with the directions of principal surface curvature, such approaches can not ensure a globally consistent layout connectivity.

The more recent Dual Loops Meshing approach [CBK12] achieves both simple and geometrically faithful quad layouts with guaranteed global consistency: Desired locations for irregularities on the surface are separated from each other by a network of curvature-aligned closed geodesic loops which intersect pairwise and approximately orthogonally. The dual of this loop network is a quad layout with irregular nodes in the desired locations, whose arcs and nodes are finally adapted to the surface shape by an iterative optimization process [CK14b].

Due to the numerous (possibly conflicting) requirements imposed for quad layouts, a fully automatic solution is not always desirable. Hence, there are several semi-automatic approaches that assist a human designer in creating valid, high-quality quad layouts using a set of interactive editing tools [JLW10; TDN+12; CK14a].

Many quad layout generation algorithms additionally produce a parametrization of the interior of each patch, often as a by-product of their inner workings. If this is not the case, a parametrization can be created retroactively, e. g. by a discrete harmonic parametrization of each individual patch [FH02].

3.3 Spline Surface Fitting

The task of turning 3D models in the form of polygonal meshes or point clouds into smooth spline surfaces has attracted steady research interest over the past decades. Spline representations are desirable due to their smoothness, their high-level editing capabilities and their memory efficient storage compared to unstructured data.

Generally, automatic methods for spline surface fitting are differentiated by their input requirements (point cloud vs. surface mesh data), surface domain structure (prescribed vs. automatically generated), and output representation (parametric vs. subdivision surfaces). On the other hand, most methods are based on the same conceptual model for the determination of the surface geometry: Somehow, a correspondence between positions on the input mesh and positions on the spline surface is established, e. g. by matching parametrizations. Then, a fitting energy

functional, measuring the approximation error between corresponding spline surface and input surface points is expressed in terms of the surface model parameters (i. e. control points). Often, the evaluation of this energy is discretely approximated by taking a finite amount of samples at corresponding positions on the input and output surface. The best approximating surface is then obtained by solving the optimization problem of finding the minimal energy surface parameters.

Particular examples for spline surface fitting based on subdivision surfaces include Hoppe *et al.* [HDD+94] who reconstruct a subdivision surface from unstructured point cloud data and Litke *et al.* [LLS01] who use a non-variational approach to fit Catmull-Clark surfaces to manifold objects without solving a global optimization problem.

For the alternative case of fitting parametric spline surfaces, the work of Eck and Hoppe [EH96] has been seminal. Their approach converts unstructured laser scan data into a tangent-plane continuous Surface Spline representation consisting of tensor product Bézier surfaces. First, a triangle mesh surface is extracted from the input point cloud which is turned into a quad layout by decimation and pairwise merging of triangles (cf. Section 3.2). The dual of this quad layout is taken as the control mesh for a Surface Spline construction. Then, corresponding samples are distributed on the generated Bézier surfaces and the parametrized patches of the input surface. Optimal positions for the Surface Spline control points are found by minimizing the distances of corresponding samples in a least-squares sense. Further extensions include an iterative parameter update of the Bézier sample points and an adaptive local refinement scheme of the underlying quad layout.

This reconstruction approach has been adapted to other parametric representations such as T-Spline surfaces [LRL06].

A common issue for all above methods are small surface oscillations arising from the spline approximation. This is countered by introducing an additional fairness energy term to the surface optimization problem, penalizing surfaces with strong curvature or higher-order derivatives. A discussion of different fairness functionals is found in [Gre94]. Westgaard and Nowacki [WN01] present a surface design scheme that constructs smooth surfaces subject to user-specified fairness functionals, maintaining additional constraints for interpolation or approximation of certain points.

Chapter 4

Framework

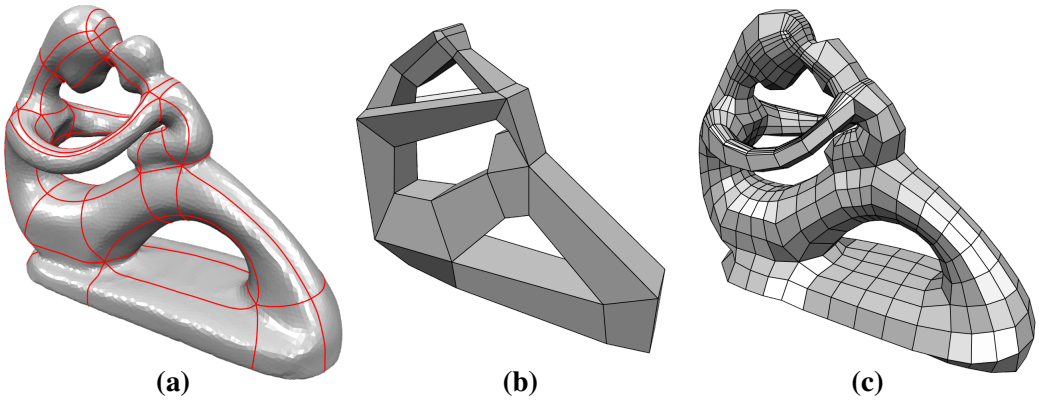
In this section, we give an overview of the general architecture of our Surface Spline fitting framework. In particular, we define the necessary data structures and intermediate meshes that are used to construct the resulting spline surface. Next, we show how to establish correspondences between points on the input mesh and output spline surfaces, which is then used to define error measures suitable to quantify the approximation quality of the generated spline surface. We can then express the problem of fitting the spline surface as an optimization of the approximation error measure by finding suitable Surface Spline control points. Finally, we show how this optimization problem can be reduced to a simple linear least squares problem.

4.1 Mesh Extraction and Refinement

Any triangle mesh with an embedded quad layout can serve as input for our algorithm (in practice, we mostly use layouts created using the Dual Loops meshing and embedding alignment algorithms by Campen *et al.* [CBK12; CK14a]).

We assume the input data is given in form of a 2-manifold triangle mesh called *input mesh* $\mathcal{I}$ (Fig. 4.1a) whose connectivity is given by the graph $(V_\mathcal{I}, E_\mathcal{I})$, implying a set of triangular faces $F_\mathcal{I}$. The quad layout embedded into $\mathcal{I}$ is specified by a set of arc edges $E \subseteq E_\mathcal{I}$ and node vertices $V \subseteq V_\mathcal{I}$. From this embedded quad layout, a separate *layout mesh* $\mathcal{L}$ (Fig. 4.1b) is extracted where each edge $E_\mathcal{L}$ corresponds to one arc, represented in $\mathcal{I}$ by a path of arc edges in E. Vertices $V_\mathcal{L}$ of the layout mesh correspond to the node vertices V in $\mathcal{I}$. If the layout embedded in $\mathcal{I}$ is a valid quad layout, all faces $F_\mathcal{L}$ of the layout mesh $\mathcal{L}$ are four-sided.

We use the following notations to refer to corresponding predecessors for elements of $\mathcal{L}$: For a vertex $v \in V_\mathcal{L}$, $\operatorname{pre} v \in V_\mathcal{I}$ refers to the corresponding node vertex in $\mathcal{I}$. For an edge $e \in E_\mathcal{L}$ corresponding to an arc embedded in $\mathcal{I}$, $\operatorname{pre} e \subseteq E$



(a) **(b)** **(c)**

Figure 4.1: From an input triangle mesh $\mathcal{I}$ **(a)**, the connectivity of the embedded quad layout (red) is extracted to form the layout mesh $\mathcal{L}$ **(b)**, which is subsequently refined into the patch mesh $\mathcal{P}$ **(c)**.

is a set containing all edges that constitute this arc in the input mesh. Similarly, for a layout mesh face $f \in F_{\mathcal{L}}$, the predecessors $\operatorname{pre} f \subseteq F_{\mathcal{I}}$ are the set of input mesh triangles falling into the layout patch corresponding to f.

In addition to the embedding of the quad layout, $\mathcal{I}$ also stores parametrizations for the individual patches. These parametrizations are represented by piecewise linear functions by storing a parameter value $\mathbf{u}_{\mathcal{I}} \in [0,1]^2$ at every triangle corner. The extracted layout mesh $\mathcal{L}$ also stores corresponding parameter values $\mathbf{u}_{\mathcal{L}} \in \{0,1\}^2$ of the face corners. Parametrizations are usually provided along with the embedded layout as input data. If none are provided, we generate simple parametrizations for each patch by fixing the corner and boundary parameter values and computing the values for the interior vertices according to a discrete harmonic parametrization.

The layout mesh $\mathcal{L}$ is a quad mesh and can therefore potentially serve as the quad layout for a tensor product Bézier spline surface. However, $\mathcal{L}$ is typically not directly usable as a surface domain for two reasons:

1. The resolution of $\mathcal{L}$ is often quite coarse and hence the resulting spline surface would contain too few degrees of freedom to achieve an acceptable approximation of the input shape.

2. Surface Splines impose some connectivity requirements on the quad layout. In particular, all pairs of irregular nodes must be separated from each other by at least three arcs. This ensures that the G^1 constructions around irregular nodes and their associated control point constraints do not conflict with one another.

We therefore refine the grid structure of $\mathcal{L}$, which is achieved in two phases: In

the first phase, some edges of $\mathcal{L}$ are annotated with a desired number of subdivisions. We describe this by a map $s\colon E_{\mathcal{L}} \to \mathbb{N}_0$ where $s(e) = n$ indicates that the edge e should be subdivided at least n times (i. e. into at least $n + 1$ pieces). Hence, for edges that require no subdivision, $s(e) = 0$. The values $s(e)$ are computed as the maximum of two intermediate subdivision requirements, $s(e) = \max\{s_{\mathrm{A}}(e), s_{\mathrm{C}}(e)\}$, which serve to counteract the previously mentioned deficiencies:

1. $s_{\mathrm{A}}(e)$ describes the desired subdivisions to improve the approximation accuracy of the spline surface. To achieve a roughly uniform subdivision, we pick a maximum arc length $\alpha_{\max}$ and compute $s_{\mathrm{A}}(e)$ as follows: For each edge $e \in E_{\mathcal{L}}$, we measure the length α_e of the associated arc embedded in $\mathcal{I}$ by summing up all edge lengths of $\operatorname{pre} e$. Then, we set
$$s_{\mathrm{A}}(e) = \left\lfloor \frac{\alpha_e}{\alpha_{\max}} \right\rfloor .$$
2. $s_{\mathrm{C}}(e)$ denotes the minimum required subdivisions of edge e due to Surface Spline connectivity constraints. As stated before, irregular layout nodes need to be isolated from each other by at least three arcs, which we enforce by a simple, sufficient heuristic: For each edge $e \in E_{\mathcal{L}}$, we set $s_{\mathrm{C}}(e)$ to the number of incident irregular vertices. This ensures that edges connecting two irregular vertices are split twice, producing a separation by three edges.

Computing $s(e)$ accordingly prescribes required refinements for all edges $e \in E_{\mathcal{L}}$. In order to maintain quad mesh connectivity, the interior of faces $f \in F_{\mathcal{L}}$ needs to be refined as well which is complicated by the fact that two opposite edges of a face f can have different required refinement values s. In such a configuration, a refinement of the edges around f would lead to a situation where it is impossible to tile the interior of f with quadrangular faces without introducing new irregular vertices. We therefore propagate required subdivision values across the mesh such that each pair of opposite edges has the same value, enabling a simple grid-like refinement of the interior of all faces. Refining faces in this fashion produces regular grids of continuous Bézier patches in the resulting surface layout which can be combined into B-Spline surfaces for compact representation.

Propagating refinement values to opposite edges can globally affect the mesh. More precisely, the refinement value $s(e)$ of an edge $e \in E_{\mathcal{L}}$ can potentially propagate to any other edge in its associated *edge ring*, which we define as follows: The edge ring $\operatorname{er} e \subseteq E_{\mathcal{L}}$ is the smallest set such that

- $e \in \operatorname{er} e$; and
- $e' \in \operatorname{er} e$ if e' has an opposite edge o with $o \in \operatorname{er} e$.

$\operatorname{er} e$ forms a chain of parallel edges across the layout mesh which either ends in two boundary edges or forms a closed loop.

By this definition, we can compute the propagated refinement values $S(e)$ for all edges $e \in E_{\mathcal{L}}$ by

$$S(e) = \max \{s(e') \mid e' \in \operatorname{er} e\} \ .$$

Now, the refined *patch mesh* $\mathcal{P}$ (Fig. 4.1c) is constructed from $\mathcal{L}$ as follows: For each edge $e \in E_{\mathcal{L}}$ with a refinement value $S(e) = n$, $n+1$ corresponding edges $e_0, \ldots, e_n \in E_{\mathcal{P}}$ are created in the patch mesh. For each face $f \in F_{\mathcal{L}}$ with two incident neighboring edges e_a, e_b with $S(e_a) = n$, $S(e_b) = m$, a grid of $(n+1) \times (m+1)$ faces $f_{ij} \in F_{\mathcal{P}}$ (with $i = 0, \ldots, n$ and $j = 0, \ldots, m$) is created. Each vertex $v \in V_{\mathcal{L}}$ corresponds directly to a patch mesh vertex $v' \in V_{\mathcal{P}}$. Additionally, $V_{\mathcal{P}}$ contains vertices that arise from the edge and face refinements. We also establish predecessor mappings between elements of $\mathcal{P}$ and $\mathcal{L}$: Patch mesh edges $e \in E_{\mathcal{P}}$ and faces $f \in F_{\mathcal{P}}$ all have unique predecessors $\operatorname{pre} e \in E_{\mathcal{L}}$, $\operatorname{pre} f \in F_{\mathcal{L}}$, corresponding to the original elements from which they were refined.

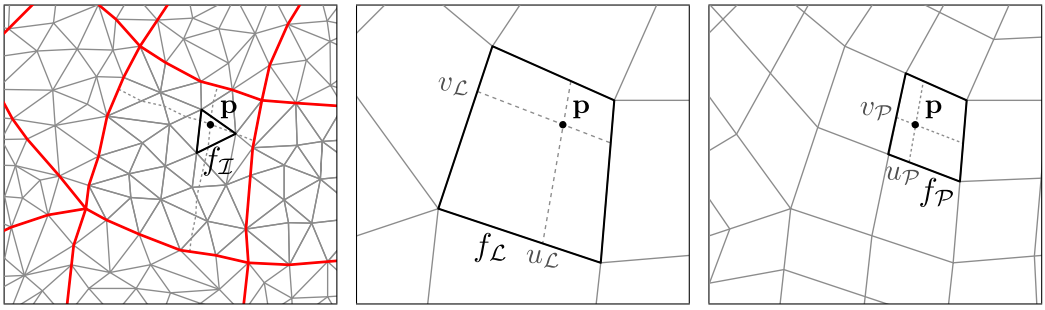
The connectivity of $\mathcal{P}$ can now serve as a quad layout for a Surface Spline construction in the sense that each face of $F_{\mathcal{P}}$ corresponds to one tensor product Bézier surface. Since the control mesh for Surface Splines is dual to the quad layout, each face also corresponds to one Surface Spline control point. However, we do not need to explicitly construct this dual mesh.

4.2 Surface Coordinates

With faces of $\mathcal{P}$ directly corresponding to Bézier surfaces, it is natural to represent a point on a face $f_{\mathcal{P}} \in F_{\mathcal{P}}$ by a local parameter $\mathbf{u}_{\mathcal{P}} \in [0,1]^2$ (Fig. 4.2c). In contrast, points on a face $f_{\mathcal{L}} \in F_{\mathcal{L}}$ of the layout mesh are more naturally represented by a global parameter $\mathbf{u}_{\mathcal{L}} \in [0,1]^2$ corresponding to the parametrization of the input mesh (Figs. 4.2a and 4.2b). Due to the refinement of $\mathcal{L}$ into $\mathcal{P}$, the same surface point can hence be described with respect to either mesh by $(f_{\mathcal{L}}, \mathbf{u}_{\mathcal{L}})$ or $(f_{\mathcal{P}}, \mathbf{u}_{\mathcal{P}})$, i. e. by different coordinates in different faces. To convert between these two representations, we set up conversion functions

$$\begin{aligned} \operatorname{to}_{\mathcal{P}} &\colon F_{\mathcal{L}} \times [0,1]^2 \to F_{\mathcal{P}} \times [0,1]^2 \ , \\ \operatorname{to}_{\mathcal{L}} &\colon F_{\mathcal{P}} \times [0,1]^2 \to F_{\mathcal{L}} \times [0,1]^2 \ . \end{aligned}$$

Since we assume a uniform refinement of $\mathcal{L}$, we can give an explicit expression for $\operatorname{to}_{\mathcal{P}}$: Consider a face $f_{\mathcal{L}} \in F_{\mathcal{L}}$ and a point on $f_{\mathcal{L}}$ represented by coordinates $\mathbf{u}_{\mathcal{L}} = (u_{\mathcal{L}}, v_{\mathcal{L}}) \in [0,1]^2$. Assuming that $f_{\mathcal{L}}$ is subdivided into $n \times m$ patch mesh



(a) **(b)** **(c)**

Figure 4.2: Representation of a point $\mathbf{p}$ on the surface: **(a)**: Input mesh with embedded layout (red). Layout coordinates are stored piecewise linear in the containing faces $f_{\mathcal{I}}$. **(b)**: $\mathbf{p}$ represented by coordinates $(u_{\mathcal{L}}, v_{\mathcal{L}})$ in the extracted layout mesh face $f_{\mathcal{L}}$. **(c)**: $\mathbf{p}$ represented by coordinates $(u_{\mathcal{P}}, v_{\mathcal{P}})$ in the refined patch mesh face $f_{\mathcal{P}}$

faces $f_{\mathcal{P}}^{ij} \in F_{\mathcal{P}}$ (for $i \in \{0, \ldots, n-1\}$ and $j \in \{0, \ldots, m-1\}$), the same point falls into face $f_{\mathcal{P}}^{ij}$ of the patch mesh with

$$i = \left\lfloor \frac{u_{\mathcal{L}}}{n} \right\rfloor \quad \text{and} \quad j = \left\lfloor \frac{v_{\mathcal{L}}}{m} \right\rfloor \ . \tag{4.1}$$

In this face, it is represented by the local coordinates $\mathbf{u}_{\mathcal{P}} = (u_{\mathcal{P}}, v_{\mathcal{P}}) \in [0,1]^2$ given by

$$u_{\mathcal{P}} = n \cdot u_{\mathcal{L}} - i \ , \quad \text{and} \quad v_{\mathcal{P}} = m \cdot v_{\mathcal{L}} - j \ . \tag{4.2}$$

Hence, we can define the forward conversion function

$$\mathrm{to}_{\mathcal{P}}(f_{\mathcal{L}}, \mathbf{u}_{\mathcal{L}}) = (f_{\mathcal{P}}^{ij}, \mathbf{u}_{\mathcal{P}}) \ . \tag{4.3}$$

Since $\mathrm{to}_{\mathcal{P}}$ is both injective and surjective, its inverse exists and we can define the backward conversion function simply by $\mathrm{to}_{\mathcal{L}} = \mathrm{to}_{\mathcal{P}}^{-1}$.

4.3 Correspondences

With the connectivity of the quad layout now fixed by $\mathcal{P}$, the next step is to create the spline surface and fit its geometry to the shape of the input mesh. For each face $f_{\mathcal{P}} \in F_{\mathcal{P}}$, an associated tensor product Bézier surface $\mathbf{b}_{f_{\mathcal{P}}}$ is generated. The Bézier points of these individual surfaces are computed from affine combinations of the Surface Spline control points. The task of surface fitting is now to find surface parameters (e. g., control points) such that the resulting Bézier surfaces approximate the surface of the input mesh as closely as possible.

In order to assess the approximation quality, we need a way to identify corresponding points on the input mesh and on the output spline surface. For this task,

we employ the coordinate representations and conversions as described in Section 4.2. These correspondences can be established either forwards or backwards:

- For *forward correspondences*, a point $\mathbf{p} \in \mathbb{A}$ is given which lies in the surface of the input mesh $\mathcal{I}$, i. e. in some triangle $f_\mathcal{I} \in F_\mathcal{I}$. By barycentric interpolation of the parameter values stored in the triangular face, an associated parameter value $\mathbf{u}_\mathcal{I}$ can be computed for $\mathbf{p}$. The corresponding face in the layout mesh is that face $f_\mathcal{L} \in F_\mathcal{L}$ for which $f_\mathcal{I} \in \operatorname{pre} f_\mathcal{L}$. Since the parametrization of $\mathcal{I}$ and $\mathcal{L}$ is identical, we can use the coordinates $\mathbf{u}_\mathcal{L} = \mathbf{u}_\mathcal{I}$ and identify the position of $\mathbf{p}$ by the pair $(f_\mathcal{L}, \mathbf{u}_\mathcal{L})$. Using Eqs. (4.1) to (4.3), we can convert this representation to a patch mesh face and coordinate by $(f_\mathcal{P}, \mathbf{u}_\mathcal{P}) = \operatorname{to}_\mathcal{P}(f_\mathcal{L}, \mathbf{u}_\mathcal{L})$. By evaluating the associated Bézier surface $\mathbf{b}_{f_\mathcal{P}}$ at these coordinates, we finally obtain the desired point on the spline surface, which we refer to as
$$\operatorname{fwd} \mathbf{p} = \mathbf{b}_{f_\mathcal{P}}(\mathbf{u}_\mathcal{P}) \ .$$
- In the case of *backward correspondences*, we start with a point on the Bézier surface, identified by a patch mesh face $f_\mathcal{P} \in F_\mathcal{P}$ and coordinate value $\mathbf{u}_\mathcal{P} \in [0,1]^2$. This representation is mapped back to the layout mesh by $(f_\mathcal{L}, \mathbf{u}_\mathcal{L}) = \operatorname{to}_\mathcal{L}(f_\mathcal{P}, \mathbf{u}_\mathcal{P})$. We now need to find the input mesh triangle which contains the parameter value $\mathbf{u}_\mathcal{L}$. To do so, we search for the triangular face $f_\mathcal{I} \in \operatorname{pre} f_\mathcal{L}$ where the parameter values $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \in [0,1]^2$ stored at its corners can form $\mathbf{u}_\mathcal{P}$ as a convex combination, i. e.
$$\mathbf{u}_\mathcal{P} = \alpha\mathbf{u}_1 + \beta\mathbf{u}_2 + \gamma\mathbf{u}_3$$
for some weights $\alpha, \beta, \gamma \geq 0$ such that $\alpha + \beta + \gamma = 1$. Using these barycentric coordinates, we can finally map back from parameter values to positions, given that the positions of the triangle corners are given by $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in \mathbb{A}$, and define
$$\operatorname{bwd}(f_\mathcal{P}, \mathbf{u}_\mathcal{P}) = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3 = \mathbf{p} \ .$$

4.4 Fitting Energy

This leaves us with two ways to measure the total approximation error between input surface and spline surface, depending on the domain of integration. If we integrate over the input surface, we obtain the following formulation: Let $\mathcal{S} \subset \mathbb{A}$ denote the set of all points which lie on the surface of the input mesh $\mathcal{I}$. Then, the

total approximation error of the spline fitting is measured by the *uniform fitting energy*

$$E_{\text{ufit}} = \frac{1}{A} \int_{\mathbf{p} \in \mathcal{S}} d\left(\mathbf{p}, \text{fwd}\, \mathbf{p}\right) \, \mathrm{d}\mathbf{p} \; . \tag{4.4}$$

where $d\colon \mathbb{A} \times \mathbb{A} \to \mathbb{R}_{\geq 0}$ is some suitable distance measure and A is the total surface area of the input mesh.

If, on the other hand, we decide to use the quad layout of the spline surface as the integration domain, we obtain a different energy functional which we call the *parametric fitting energy*

$$E_{\text{pfit}} = \sum_{f \in F_{\mathcal{P}}} \int_0^1 \int_0^1 d\left(\mathbf{b}_f(u,v), \text{bwd}(f,(u,v))\right) \, \mathrm{d}u \, \mathrm{d}v \; . \tag{4.5}$$

Note that in general, $E_{\text{ufit}} \neq E_{\text{pfit}}$. In some contexts, the integration domain is not relevant and we use E_{fit} to refer to either of the two energies.

Assuming a fixed input surface, E_{ufit} and E_{pfit} depend on the positions of the Bézier points, which we refer to as $\hat{\mathbf{b}} \in \mathbb{A}^{N_{\text{B}}}$. The positions $\hat{\mathbf{b}}$ in turn are derived from affine combination of the Surface Spline control points. If we gather all of the Surface Spline control points in a vector $\hat{\mathbf{c}} \in \mathbb{A}^{N_{\text{C}}}$, we can thus write the energies as functions in $\hat{\mathbf{c}}$, i. e. $E_{\text{ufit}}(\hat{\mathbf{c}})$ and $E_{\text{pfit}}(\hat{\mathbf{c}})$ and the task of finding the best approximating spline surface turns into the optimization problem

$$\hat{\mathbf{c}}^* = \operatorname*{argmin}_{\hat{\mathbf{c}} \in \mathbb{A}^{N_{\text{C}}}} E_{\text{fit}}(\hat{\mathbf{c}}) \; . \tag{4.6}$$

4.5 Optimization

Following Eck and Hoppe [EH96], we break down (4.6) into a more manageable form by

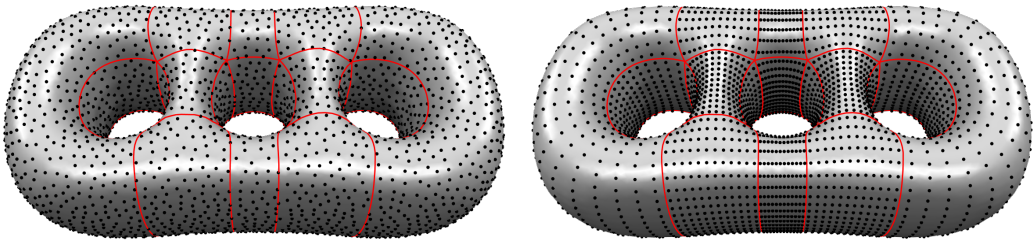
(*a*) approximating the integrals appearing in Eqs. (4.4) and (4.5) by sampling;

(*b*) choosing a suitable distance measure d; and

(*c*) exploiting the linear properties of the Bézier and spline surface representation.

This will allow us to pose Eq. (4.6) as a sparse linear least-squares problem, which can be solved efficiently.

First (Item (*a*)), we eliminate the integrals in the fitting energy functionals by approximating them with sums. For the uniform fitting energy (Eq. (4.4)), we



(a) **(b)**

Figure 4.3: Discrete approximation of the fitting energies by sampling. **(a)**: Uniform sampling of the surface $\mathcal{S}$ of the input mesh approximating E_{ufit}. **(b)**: Uniform sampling of the parameter domains of the resulting Bézier patches approximating E_{pfit}.

partition the surface $\mathcal{S}$ into n little pieces, each piece with an area A_i centered around a surface point $\mathbf{p}_i$ (with $i \in \{1, \dots, n\}$). We can then approximate the energy by

$$\begin{aligned} E_{\text{ufit}} &= \frac{1}{A} \int_{\mathbf{p} \in \mathcal{S}} d\left(\mathbf{p}, \text{fwd}\, \mathbf{p}\right) \, \text{d}\mathbf{p} \\ &\approx \frac{1}{A} \sum_{i=1}^{n} d\left(\mathbf{p}_i, \text{fwd}\, \mathbf{p}_i\right) \cdot A_i \ . \end{aligned}$$

If n becomes sufficiently large and the position samples $\mathbf{p}_i$ are uniformly distributed over $\mathcal{S}$ (see Fig. 4.3a), the areas of the little surface pieces approach $A_i \approx \frac{A}{n}$. Hence, we obtain

$$E_{\text{ufit}} \approx \frac{1}{n} \sum_{i=1}^{n} d\left(\mathbf{p}_i, \text{fwd}\, \mathbf{p}_i\right) \ . \tag{4.7}$$

For the approximation of the parametric fitting energy (Eq. (4.5)), we instead discretely sample the parameter domains of the patches (see Fig. 4.3b). On each patch, we distribute a regular, uniform grid of samples $(u_i, v_j) \in [0, 1]^2$ with $i, j \in \{1, \dots, n\}$. Every sample hence covers a parametric area of $R_{ij} = \frac{1}{n^2}$, and we can use the approximation

$$\begin{aligned} E_{\text{pfit}} &= \sum_{f \in F_{\mathcal{P}}} \int_0^1 \int_0^1 d\left(\mathbf{b}_f(u, v), \text{bwd}(f, (u, v))\right) \, \text{d}u \, \text{d}v \\ &\approx \sum_{f \in F_{\mathcal{P}}} \sum_{i=1}^{n} \sum_{j=1}^{n} d\left(\mathbf{b}_f(u_i, v_j), \text{bwd}(f, (u_i, v_j))\right) \cdot R_{ij} \\ &= \frac{1}{n^2} \sum_{f \in F_{\mathcal{P}}} \sum_{i=1}^{n} \sum_{j=1}^{n} d\left(\mathbf{b}_f(u_i, v_j), \text{bwd}(f, (u_i, v_j))\right) \ . \end{aligned}$$

Next, for Item (*b*), we choose the squared euclidean distance as our distance measure d. We can do this by interpreting the difference vector between the two points $\mathbf{x} - \mathbf{y} \in \mathbb{V}$ as a real coordinate vector $\mathbb{R}^n$ and just employ the standard euclidean norm:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2 \ .$$

Finally (Item (*c*)), remember that a tensor product Bézier surface $\mathbf{b}$ of degree (n, m) is evaluated at a parameter (u, v) by

$$\mathbf{b}(u, v) = \sum_{i=0}^{n} \sum_{i=0}^{m} B_i^n(u) B_j^m(v) \mathbf{b}_{ij} \ . \tag{4.8}$$

By fixing some parameter value (u, v) in Eq. (4.8), the Bernstein polynomials become just linear coefficients a_{ij} of the Bézier points:

$$\mathbf{b}(u, v) = \sum_{i=0}^{n} \sum_{i=0}^{m} a_{ij} \cdot \mathbf{b}_{ij} \ .$$

By some simple re-indexing, we can refer to the Bézier points and Bernstein coefficients using single indices $r \in \{1, \ldots, n_\mathrm{B}\}$ instead of double indices (i, j), i. e. $\mathbf{b}_r = \mathbf{b}_{ij}$ and $a_r = a_{ij}$. This allows us to write $\mathbf{b}(u, v)$ as a dot product

$$\mathbf{b}(u, v) = \begin{pmatrix} a_0 & \cdots & a_{n_\mathrm{B}} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_{n_\mathrm{B}} \end{pmatrix} \ .$$

By extending the left operand from a row vector to a matrix $\mathbf{S} \in \mathbb{R}^{n_\mathrm{S} \times n_\mathrm{B}}$ where each row corresponds to the Bernstein coefficients for some parameter, we can compute an entire column vector $\mathbf{s} \in \mathbb{A}^{n_\mathrm{S}}$ containing n_S samples of the Bézier surface at once:

$$\mathbf{s} = \begin{pmatrix} \mathbf{b}(u_1, v_1) \\ \vdots \\ \mathbf{b}(u_{n_\mathrm{S}}, v_{n_\mathrm{S}}) \end{pmatrix} = \mathbf{S} \cdot \begin{pmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_{n_\mathrm{B}} \end{pmatrix} \ .$$

Instead of computing samples from one single Bézier patch, we can compute samples from the entire spline surface using a single matrix multiplication in the same fashion. In order to do this, we concatenate the Bézier points of all individual Bézier surfaces into one big column vector $\hat{\mathbf{b}} \in \mathbb{A}^{N_\mathrm{B}}$. We multiply $\hat{\mathbf{b}}$ by a matrix $\hat{\mathbf{S}} \in \mathbb{R}^{N_\mathrm{S} \times N_\mathrm{B}}$ where each row contains some nonzero elements corresponding to the Bernstein coefficients of one coordinate sample from a single Bézier patch. All other elements are zero, thus $\hat{\mathbf{S}}$ is very sparse. The result of this multiplication is a column vector $\hat{\mathbf{s}} \in \mathbb{A}^{N_\mathrm{S}}$ containing N_S samples from the spline surface:

$$\hat{\mathbf{s}} = \hat{\mathbf{S}} \cdot \hat{\mathbf{b}} \ . \tag{4.9}$$

We now have a linear representation of spline surface samples $\hat{\mathbf{s}}$ in terms of the Bézier points $\hat{\mathbf{b}}$. However, our optimization problem (Eq. (4.6)) is posed in terms of the Surface Spline control points $\hat{\mathbf{c}}$, instead. Conveniently, the positions of our Bézier points derive from affine combinations of the Surface Spline control points, which also leads to a linear expression. Suppose all control points are given by a vector $\hat{\mathbf{c}} \in \mathbb{A}^{N_\mathrm{C}}$. Then, the vector $\hat{\mathbf{b}} \in \mathbb{A}^{N_\mathrm{B}}$ of all Bézier points is computed by

$$\hat{\mathbf{b}} = \hat{\mathbf{B}} \cdot \hat{\mathbf{c}} \tag{4.10}$$

where $\hat{\mathbf{B}} \in \mathbb{R}^{N_\mathrm{B} \times N_\mathrm{C}}$ is a matrix where each row contains the coefficients of the Surface Spline control points that contribute to the position of one Bézier point. In the Surface Spline construction, Bézier point positions are combined from only a small, constant number of control points. Hence, $\hat{\mathbf{B}}$ is sparse as well. Now, plugging Eq. (4.10) back into Eq. (4.9), we get

$$\hat{\mathbf{s}} = \hat{\mathbf{S}} \cdot \hat{\mathbf{b}} = \hat{\mathbf{S}} \cdot \hat{\mathbf{B}} \cdot \hat{\mathbf{c}} = \mathbf{A} \cdot \hat{\mathbf{c}} \tag{4.11}$$

where $\mathbf{A} = \hat{\mathbf{S}} \cdot \hat{\mathbf{B}} \in \mathbb{R}^{N_\mathrm{S} \times N_\mathrm{C}}$ is again a sparse matrix.

Let us put the above results together to derive an explicit form of the Surface Spline optimization problem using the discretized uniform fitting energy. First, we employ our choice of distance function, replacing d in Eq. (4.7) by the squared euclidean distance:

$$E_\mathrm{ufit} \approx \frac{1}{n_\mathrm{S}} \sum_{i=1}^{n} \|\mathbf{p}_i - \mathrm{fwd}\, \mathbf{p}_i\|_2^2 \ .$$

Instead of summing up the squared differences of individual sample pairs $\mathbf{p}_i$, $\mathrm{fwd}\, \mathbf{p}_i$, we can write this as the squared norm of a single vector difference: If we combine all input mesh samples and spline surface samples into column vectors $\hat{\mathbf{p}} = (\mathbf{p}_1, \ldots, \mathbf{p}_{N_\mathrm{S}})^\mathsf{T}$ and $\hat{\mathbf{s}} = (\mathrm{fwd}\, \mathbf{p}_1, \ldots, \mathrm{fwd}\, \mathbf{p}_{N_\mathrm{S}})^\mathsf{T}$, we get

$$E_\mathrm{ufit} \approx \frac{1}{n_\mathrm{S}} \|\hat{\mathbf{p}} - \hat{\mathbf{s}}\|_2^2 \ .$$

Now, recall that we can generate the output samples $\hat{\mathbf{s}}$ directly from the positions of the Surface Spline control points $\hat{\mathbf{c}}$ by multiplication with a matrix $\mathbf{A} \in \mathbb{R}^{N_\mathrm{S} \times N_\mathrm{C}}$ (cf. Eq. (4.11)). Hence, we can express E_ufit in terms of $\hat{\mathbf{c}}$ by

$$E_\mathrm{ufit}(\hat{\mathbf{c}}) \approx \frac{1}{n_\mathrm{S}} \|\hat{\mathbf{p}} - \mathbf{A} \cdot \hat{\mathbf{c}}\|_2^2 \ , \tag{4.12}$$

which brings the minimization of E_ufit to the form of a linear least squares problem. Ignoring the constant factor $\frac{1}{n_\mathrm{S}}$, we find a minimizer $\hat{\mathbf{c}}^*$ for Eq. (4.12) by solving the normal equations

$$\mathbf{A}^\mathsf{T}\mathbf{A} \cdot \hat{\mathbf{c}}^* = \mathbf{A}^\mathsf{T}\hat{\mathbf{p}} \ . \tag{4.13}$$

The parametric fitting energy E_{pfit} can easily be brought to the same form as E_{ufit} in Eq. (4.12). In general, any sampling scheme that creates corresponding input surface points and spline surface points obtained from fixed Bézier surface parameters can be expressed in this fashion. Note that changing the sampling strategy affects the values of $\mathbf{A}$ and will hence lead to different optimization results. Increasing the number of samples N_{S} will increase the number of rows in $\mathbf{A}$. This does however not affect the size of the linear system (4.13), whose complexity solely depends on the number of control points N_{C}.

4.6 Fairness Energy

The approach of spline surface fitting described in the previous sections has a notorious drawback: While the generated surfaces are indeed continuous (as is guaranteed by the Surface Spline construction), they commonly exhibit ripples or oscillations that are not present in the input surface.

These undesirable imperfections in the resulting surface are commonly counteracted by introducing some sort of *fairness energy* into the optimization which penalizes strong variation of surface tangents or higher-order properties. While plenty of such fairness energies have been proposed [Gre94; WN01], the most common choice is some variation of a *thin plate energy* [FS96; EH96; SWW+04; LRL06], such as

$$E_{\text{fair}} = \sum_{\mathbf{b}\in B} \int_0^1\int_0^1 \|\mathbf{b_{uu}}(u,v)\|_2^2 + 2\cdot\|\mathbf{b_{uv}}(u,v)\|_2^2 + \|\mathbf{b_{vv}}(u,v)\|_2^2 \,\mathrm{d}u\,\mathrm{d}v \quad (4.14)$$

where we denote by $B = \{\mathbf{b}_{f_\mathcal{P}} \mid f_\mathcal{P} \in F_\mathcal{P}\}$ the set of all Bézier patches constituting the spline surface and the subscripts of $\mathbf{b_{uu}}$, $\mathbf{b_{uv}}$ and $\mathbf{b_{vv}}$ denote second-order partial derivatives of $\mathbf{b}$. The popularity of the thin plate energy is due to the fact that, just like the fitting energy E_{fit}, it can be formulated as a quadratic function of the Surface Spline control points and hence can be minimized by solving a linear system.

The evaluation of Eq. (4.14) is occasionally handled by sampling [LRL06], analogously to the discrete approximation of the fitting energy as described in Section 4.5. However, E_{fair} can also be analytically evaluated to a simple quadratic form which can be expressed in terms of the Surface Spline control points, i. e.

$$E_{\text{fair}}(\hat{\mathbf{c}}) = \hat{\mathbf{c}}^\mathsf{T}\mathbf{S}\hat{\mathbf{c}}$$

where $\mathbf{S} \in \mathbb{R}^{N_\text{C}\times N_\text{C}}$ is a symmetric positive-definite matrix whose construction is derived in Appendix A.

For a simultaneous optimization, the fitting energy E_{fit} and fairness energy E_{fair} are mixed together by a linear combination

$$E(\hat{\mathbf{c}}) = E_{\text{fit}}(\hat{\mathbf{c}}) + \sigma \cdot E_{\text{fair}}(\hat{\mathbf{c}}) \tag{4.15}$$

where $\sigma \in \mathbb{R}_{\geq 0}$ is a scaling factor that allows to control the strength of the fairing. The larger the value of σ, the more the shape of the spline surface is allowed to deviate from the shape of the input surface to allow for more smoothness.

Note that since Eq. (4.15) is a linear combination of quadratic functionals, it is itself quadratic (here, we again drop the factor $\frac{1}{N_{\text{S}}}$ from E_{fit}):

$$\begin{aligned} E(\hat{\mathbf{c}}) &= E_{\text{fit}}(\hat{\mathbf{c}}) + \sigma \cdot E_{\text{fair}}(\hat{\mathbf{c}}) \\ &= \|\hat{\mathbf{p}} - \mathbf{A}\hat{\mathbf{c}}\|_2^2 + \sigma \cdot \hat{\mathbf{c}}^{\mathsf{T}}\mathbf{F}\hat{\mathbf{c}} \\ &= (\hat{\mathbf{p}} - \mathbf{A}\hat{\mathbf{c}})^{\mathsf{T}} (\hat{\mathbf{p}} - \mathbf{A}\hat{\mathbf{c}}) + \sigma \cdot \hat{\mathbf{c}}^{\mathsf{T}}\mathbf{F}\hat{\mathbf{c}} \\ &= \hat{\mathbf{c}}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{A}\hat{\mathbf{c}} - 2\hat{\mathbf{p}}^{\mathsf{T}}\mathbf{A}\hat{\mathbf{c}} + \hat{\mathbf{p}}^{\mathsf{T}}\hat{\mathbf{p}} + \sigma \cdot \hat{\mathbf{c}}^{\mathsf{T}}\mathbf{F}\hat{\mathbf{c}} \\ &= \hat{\mathbf{c}}^{\mathsf{T}} \left(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \sigma \cdot \mathbf{F}\right) \hat{\mathbf{c}} - 2\hat{\mathbf{p}}^{\mathsf{T}}\mathbf{A}\hat{\mathbf{c}} + \hat{\mathbf{p}}^{\mathsf{T}}\hat{\mathbf{p}} \,. \end{aligned}$$

To find the minimum, we derive w. r. t. $\hat{\mathbf{c}}$ and set to zero, yielding

$$\begin{aligned} 2 \left(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \sigma \cdot \mathbf{F}\right) \hat{\mathbf{c}} - 2\hat{\mathbf{p}}^{\mathsf{T}}\mathbf{A} &= 0 \\ \left(\mathbf{A}^{\mathsf{T}}\mathbf{A} + \sigma \cdot \mathbf{F}\right) \hat{\mathbf{c}} &= \mathbf{A}^{\mathsf{T}}\hat{\mathbf{p}} \end{aligned} \tag{4.16}$$

which serves as a direct generalization of the normal equations (4.13).

4.7 Control Point Constraints

As mentioned previously in Section 3.1, if we want the Surface Spline surface to be both continuous and of low polynomial degree, we must enforce certain constraints on some of the Surface Spline control points. In particular, these constraints arise around layout nodes which have an even valence $n > 4$, i. e. $n \in \{6, 8, 10, \dots\}$. Each such constraint relates the $2n$ control points of the patches that directly surround the irregular n-sided region and is given by

$$\sum_{i=1}^{n} (-1)^i \left(\mathbf{c}_{i,1} - \mathbf{c}_{i,2}\right) \,. \tag{4.17}$$

Equation (4.17) is a linear expression of a single control point constraint. We can simultaneously express all constraints arising for a particular layout by a matrix-vector product

$$\mathbf{C} \cdot \hat{\mathbf{c}} = 0$$

where every row of the constraint matrix $\mathbf{C} \in \mathbb{R}^{N_\mathrm{I} \times N_\mathrm{C}}$ contains some coefficients ± 1 corresponding to a single constraint expression (4.17). The vector $\hat{\mathbf{c}} \in \mathbb{A}^{N_\mathrm{C}}$ is a vector of all N_C Surface Spline control points and N_I is the total number of irregular layout nodes with even valence.

In order to enforce these constraints, we restrict the solution of Eq. (4.16) by introducing Lagrange multipliers $\lambda \in \mathbb{R}^{N_\mathrm{I}}$, hence converting our objective function to the Lagrangian

$$L(\hat{\mathbf{c}}, \lambda) = E_\mathrm{fit}(\hat{\mathbf{c}}) + \sigma E_\mathrm{fair}(\hat{\mathbf{c}}) + \lambda^\mathsf{T} \mathbf{C} \hat{\mathbf{c}} \; . \tag{4.18}$$

Differentiating Eq. (4.18) w. r. t. $\hat{\mathbf{c}}$ and setting to zero yields the same terms as above for Eq. (4.15), plus one new term containing λ:

$$2 \left(\mathbf{A}^\mathsf{T} \mathbf{A} + \sigma \cdot \mathbf{F} \right) \hat{\mathbf{c}} - 2 \hat{\mathbf{p}}^\mathsf{T} \mathbf{A} + \mathbf{C}^\mathsf{T} \lambda = 0 \tag{4.19}$$

$$\left(\mathbf{A}^\mathsf{T} \mathbf{A} + \sigma \cdot \mathbf{F} \right) \hat{\mathbf{c}} + \frac{1}{2} \mathbf{C}^\mathsf{T} \lambda = \mathbf{A}^\mathsf{T} \hat{\mathbf{p}} \; , \tag{4.20}$$

while differentiating Eq. (4.18) w. r. t. λ yields just

$$\mathbf{C} \hat{\mathbf{c}} = 0 \; . \tag{4.21}$$

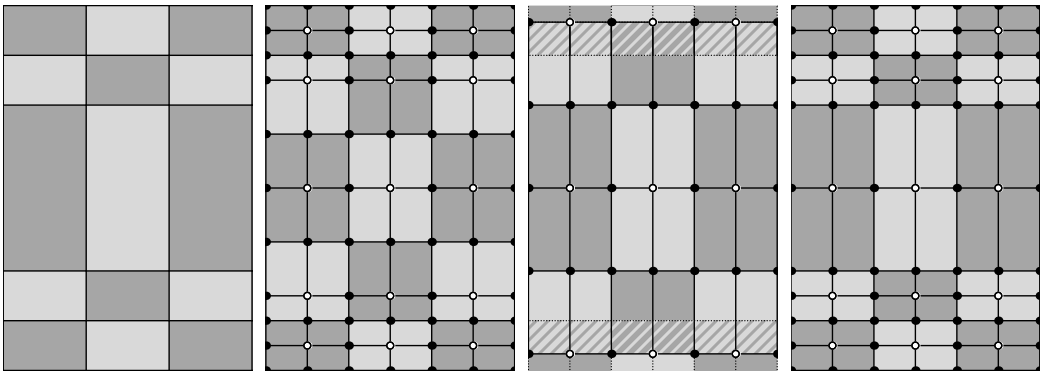
We can now combine Eqs. (4.20) and (4.21) into one equation by constructing a $(N_\mathrm{C} + N_\mathrm{I}) \times (N_\mathrm{C} + N_\mathrm{I})$ block matrix

$$\left(\begin{array}{c|c} \mathbf{A}^\mathsf{T} \mathbf{A} + \sigma \cdot \mathbf{F} & \frac{1}{2} \mathbf{C}^\mathsf{T} \\ \hline \frac{1}{2} \mathbf{C} & 0 \end{array} \right) \begin{pmatrix} \hat{\mathbf{c}} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{A}^\mathsf{T} \hat{\mathbf{p}} \\ 0 \end{pmatrix} \; . \tag{4.22}$$

By solving Eq. (4.22) for $\hat{\mathbf{c}}$ and λ, we obtain the control point positions of the optimal spline surface w. r. t. the fitting and fairness energies which fulfills the Surface Spline construction constraints and is hence tangent plane continuous everywhere.

4.8 Discussion

Experimental results of the spline surface fitting framework described in this chapter reveal that the major problem with this method are small surface oscillations. Often, the generated spline surfaces will be wavy or contain small artifacts such as cusps or foldovers which are not present in the input mesh. These problems are especially pronounced when the input quad layout comprises neighboring patches of greatly varying sizes. The obvious countermeasure to these artifacts, i. e. increasing the weight σ of the fairness energy E_fair, does unfortunately not adequately solve the problem: In most cases, σ must be increased to a significant magnitude



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

Figure 4.4: A layout with greatly varying patch sizes **(a)** cannot be regularly represented by a biquadtratic C^1 spline surface **(b)** where cross-boundary tangents (successive grid edges that cross a patch boundary) must have equal length. Moving the free Bézier points to match the given patch sizes results in foldovers (hatched) **(c)**. G^1 continuity **(d)** lifts this restriction and can hence reproduce the original configuration without distortions.

in order to flatten out the oscillation artifacts. Doing so will, however, cause the spline surface to greatly deviate from the target shape, especially near fine features which are lost due to excessive smoothing.

We have identified two major causes which contribute to these problems:

The first flaw is found in the formulation of the fairness energy E_{fair}, i. e. the thin plate energy (Eq. (4.14)). The problem here lies in the fact that the total curvature of individual Bézier patches is computed by integrating over their parameter domains, which erroneously assumes a uniform area parametrization. However, since the given layout may prescribe varying patch sizes, this assumption is violated, leading to small patches producing significantly less fairing energy than large patches, given the same curvature. Hence, optimized surfaces tend to minimize their fairing energy E_{fair} by storing most of their total curvature in small patches, thereby causing small oscillations. We discuss possible countermeasures to this problem in Chapter 6.

The second problem is rooted in geometric inflexibilities of the Surface Spline construction. Since in regular regions, individual Bézier patches are joined with C^1 continuity, they have identical partial derivatives across shared boundaries. As discussed in Section 2.3.1, this property translates into the midpoint condition on the Bézier grids, enforcing that triplets of Bézier points that straddle a patch boundary must be collinear and equidistant (cf. Eq. (2.15)). This equidistance constraint can be the cause of severe distortions and foldovers in the shape of the spline surface where patches with significantly differing sizes meet, as illustrated in Fig. 4.4: The C^1 spline surface (Fig. 4.4b) set to approximate the input lay-

out (Fig. 4.4a) is determined by one free Bézier point per patch (hollow dots). The remaining Bézier points (solid dots) are computed by averaging the positions of the free points. Thus, in order to create Bézier patches that have the same height as the center row of large layout patches, the free Bézier points of the rows below and above must be moved outwards. Since the neighboring rows are relatively short, the free Bézier points move beyond the subsequent boundary row, hence creating foldovers (hatched regions in Fig. 4.4c). These artifacts are by no means uncommon in practice: For foldovers to occur, it suffices that the widths of adjacent patches differ by a factor of two, which can occur quite often, even after some moderate uniform refinement. Our proposed remedy is to lift some of the restrictions of the Surface Spline model: If we give up C^1 continuity and instead move to the weaker condition of G^1 continuity, boundary tangents must still be collinear but they need no longer be of equal length. This allows to faithfully recreate layouts with varying patch sizes without introducing foldovers (see Fig. 4.4d). In Chapter 5, we derive a model for G^1 spline surfaces as an alternative to Surface Splines, allowing to use arbitrary tangent aspect ratios, hence avoiding most foldover artifacts.

Note how both the above problems are ultimately caused by non-uniform patch sizes. Hence, another obvious solution would be to refine the quad layout until all patch sizes are mostly uniform, i. e. by reducing the target arc length α_{max} (see Section 4.1). However, such a fine granularity of the quad layout is seldom desirable as it defeats the purpose of spline fitting, i. e. obtaining a high-level, simplified mesh representation which is easy to edit and has a low memory footprint.

Chapter 5

G^1 Bézier Splines

In this chapter, we derive a representation for tangent-plane continuous spline surfaces which, unlike Surface Splines, allows regular surface patches to join with G^1 instead of C^1 continuity. This added flexibility makes this model more suitable for approximating surfaces with embedded quad layouts comprising non-uniformly sized patches, as discussed in Section 4.8.

We will start with a brief characterization of our G^1 Bézier spline model in comparison to Surface Splines (Section 5.1). Then, we introduce the new degrees of freedom of our model, i. e. the G^1 aspect ratios, and how they arise from the connectivity of the patch mesh (Section 5.2). We then present a set of proposed constraints that encode G^1 continuity with the desired aspect ratios for joints in regular (Section 5.3) and irregular (Section 5.4) regions, and prove the existence of a solution to these constraints (Section 5.5).

5.1 Characterization

We propose our G^1 Bézier spline surface model as a generalization of the capabilities of Surface Splines. Due to their similarities, we can adapt most of our Surface Spline fitting framework (Chapter 4) to our new spline model, requiring only minor modifications. However, there are some notable differences:

- Surface Splines combine patches of different degrees: Regular layout regions are tiled with biquadratic (degree $(2, 2)$) tensor product Bézier surfaces. Around irregular nodes, bicubic (degree $(3, 3)$) patches are used. Our model, in contrast, uses bicubic patches in both regular and irregular regions. Using patches of the same degree everywhere simplifies the derivation of continuity constraints.

- Obviously, Surface Splines and our G^1 Bézier splines differ in their continuity guarantees: Surface Splines guarantee C^1 continuity across almost all layout arcs. Only arcs incident to an irregular node join with G^1 continuity. In our construction, all arcs are G^1 joints, regardless of the valence of their incident nodes. Still, the particular constraint formulations differ for regular and irregular arcs.

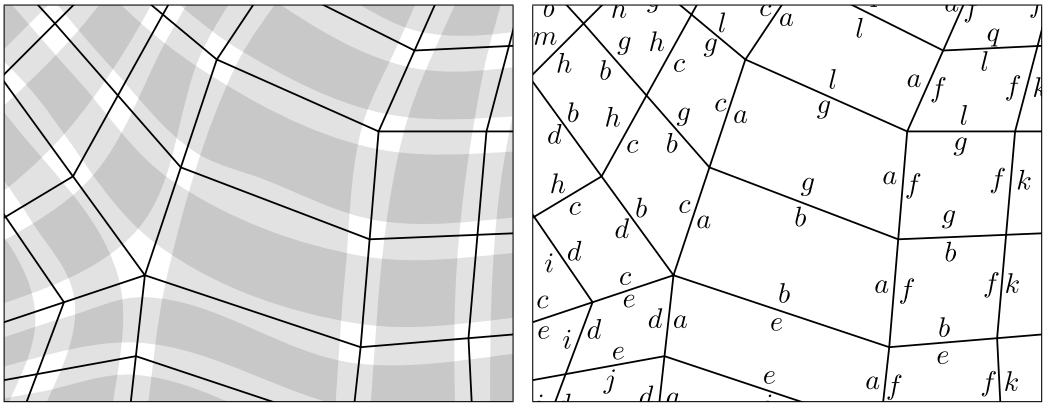
- For Surface Splines, Bézier point positions are computed from linear combinations of the positions of control mesh vertices. The interpolation coefficients are derived from an analytic solution to some fixed, pre-determined continuity constraints. Similarly, our G^1 construction is obtained by imposing certain continuity constraints for patch boundaries in terms of the Bézier points. We later show that a solution for a thus constrained Bézier network exists. We will also analyze the degrees of freedom of the resulting model and how to derive the positions of dependent Bézier points from the free variables.

- Although we show the existence of an analytic solution to our set of constraints, in practice, we still use the constrained form for surface fitting. The benefit of the constrained form is that we can easily incorporate features like sharp corners or feature edges by dropping continuity constraints for selected layout arcs, hence introducing additional degrees of freedom to the model. In our original Surface Spline fitting framework (Chapter 4), we optimize the positions of Surface Spline control points $\hat{\mathbf{c}}$ to obtain an optimal Bézier spline surface. For our new spline model on the other hand, we directly optimize the Bézier points $\hat{\mathbf{b}}$ subject to the given continuity constraints.

5.2 Aspect Ratios

The switch to G^1 continuity allows joints across regular arcs to use arbitrary tangent vector aspect ratios instead of the $1 : 1$ ratios required for C^1 continuity. However, in general, aspect ratios cannot be chosen independently for all individual arcs without overconstraining the surface definition. By restricting the number of variable aspect ratios based on the global connectivity of the layout mesh, we can guarantee consistency.

Key to our consistent choice of G^1 aspect ratios is a decomposition of the quad layout into quad strips. Remember the definition of an edge ring from Section 4.1: For an edge $e \in E_{\mathcal{P}}$ in our patch mesh, $\mathrm{er}\, e \subseteq E_{\mathcal{P}}$ denotes a chain of parallel edges propagating from e over the mesh which may either loop unto itself or end in some



(a) **(b)**

Figure 5.1: The patches of a quad layout arise from pairwise overlapping quad strips **(a)**. Assuming strips have uniform widths $a, b, c, \ldots$, half-arcs of layout patches are labeled with the width of their parallel constituting strip **(b)**.

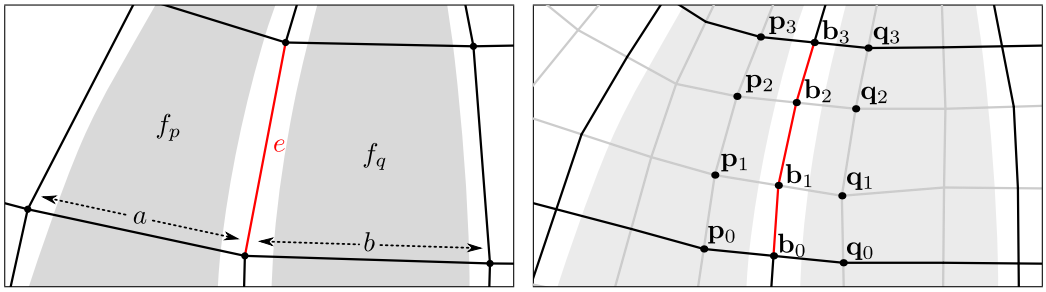
boundary edges. Based on this, we define the associated *quad strip* of an edge e as

$$\operatorname{strip} e = \{f \in F_{\mathcal{P}} \mid \exists\, e' \in \operatorname{er} e : f \operatorname{inc} e'\} \ ,$$

where the relation $f \operatorname{inc} e'$ denotes that face f is incident to edge e'. A quad strip is hence a set of adjacent faces where all shared edges are opposites of their successors. Every patch of a quad layout is thus given by the transversal intersection of two distinct quad strips (or by a transversal self-intersection of a single quad strip) [CK14a], as shown in Fig. 5.1a. If we make the assumption that all patches within a strip have roughly the same width, we can characterize each strip by a positive real-valued *strip width*. Thereby, aspect ratios between adjacent patches are implicitly given by the width ratios of the two corresponding adjacent strips (Fig. 5.1b).

5.3 Regular G^1 Joints

A *regular* G^1 *joint* occurs at each arc connecting two regular nodes. Consider the following configuration, as illustrated in Fig. 5.2a: Let the arc e be the boundary between two adjacent patches f_p, f_q, belonging to two distinct quad strips with widths $a, b \in \mathbb{R}_{\geq 0}$. The spline surface pieces corresponding to f_p and f_q are the bicubic tensor-product Bézier surfaces $\mathbf{p}(u, v)$, $\mathbf{q}(u, v)$, which are to be joined with C^0 and G^1 continuity. As noted in Section 2.3.2, a C^0 joint is easily obtained by choosing equal boundary Bézier points. By doing that, we can use the following labeling for the grid points (see Fig. 5.2b): We refer to the center row



(a) **(b)**

Figure 5.2: **(a)**: Regular arc e between two regular nodes. The adjacent patch mesh faces f_p, f_q lie in quad strips with widths a, b. **(b)**: Labeling of the Bézier points for the associated regular G^1 condition.

of Bézier points shared between $\mathbf{p}$ and $\mathbf{q}$ by the labels $\mathbf{b}_0, \ldots, \mathbf{b}_3$. The points in the two adjacent rows on either side are called $\mathbf{p}_0, \ldots, \mathbf{p}_3$ and $\mathbf{q}_0, \ldots, \mathbf{q}_3$.

To establish G^1 continuity, we use Eq. (2.21) from Section 2.3.2, which expresses the reparametrization of $\mathbf{p}$ and $\mathbf{q}$ in terms of scalar connection functions α, β, γ:

$$\alpha(v) \cdot \mathbf{p}_\mathbf{u}(1, v) - \beta(v) \cdot \mathbf{q}_\mathbf{u}(0, v) + \gamma(v) \cdot \mathbf{p}_\mathbf{v}(1, v) = 0 \,. \tag{5.1}$$

By fixing the functions α, β, γ in Eq. (5.1), we derive a sufficient G^1 continuity condition. Here, we simply choose them as constants

$$\alpha(v) = \frac{1}{a} \,, \qquad \beta(v) = \frac{1}{b} \,, \qquad \gamma(v) = 0 \,.$$

Thus, Eq. (5.1) reduces to

$$b \cdot \mathbf{p}_\mathbf{u}(1, v) = a \cdot \mathbf{q}_\mathbf{u}(0, v) \,, \tag{5.2}$$

which reveals our geometric intuition behind this choice of connection functions: For Eq. (5.2) to hold, the lengths of the tangents $\mathbf{p}_\mathbf{u}$ and $\mathbf{q}_\mathbf{u}$ must be proportional to the strip widths a and b, respectively.

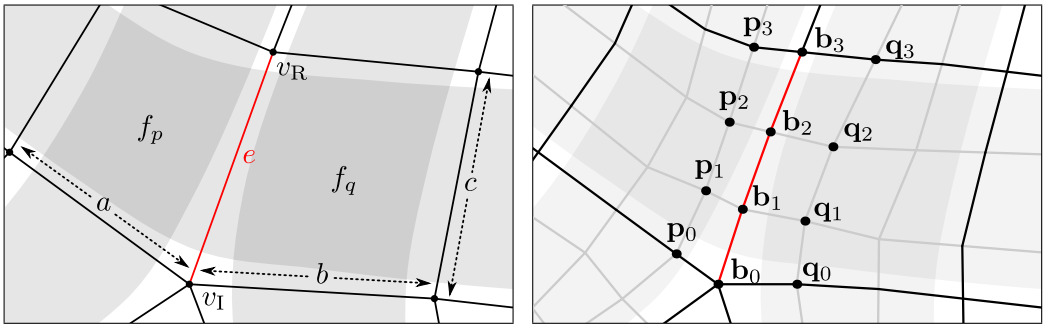
Expanding Eq. (5.2), we obtain the expression in terms of the Bézier points

$$b \cdot \sum_{j=0}^{3} B_j^3(v) \, (\mathbf{b}_j - \mathbf{p}_j) = a \cdot \sum_{j=0}^{3} B_j^3(v) \, (\mathbf{q}_j - \mathbf{b}_j)$$

which, due to the linear independence of the Bernstein polynomials $B_j^3(v)$, reduces to four separate equations

$$b \cdot (\mathbf{b}_j - \mathbf{p}_j) = a \cdot (\mathbf{q}_j - \mathbf{b}_j) \tag{RC}$$

for $j \in \{0, \ldots, 3\}$. Our sufficient G^1 condition can hence be enforced by four equality constraints in terms of Bézier points for each regular arc.



(a) **(b)**

Figure 5.3: (a): Irregular arc e between an irregular (v_I) and a regular (v_R) node. Widths of the parallel (a, b) and transversal (c) quad strips are shown. **(b)**: Labeling of the Bézier points involved in the irregular G^1 constraint.

5.4 Irregular G^1 Joints

Irregular G^1 joints arise at arcs which are incident to one irregular and one regular node, as shown in Fig. 5.3a (note that arcs incident to two irregular nodes cannot occur in our layouts since such configurations are eliminated by the layout mesh refinement, cf. Section 4.1). Let us denote the irregular and regular node by v_I and v_R, respectively, and the connecting arc by e. As before, we call the two incident patches f_p, f_q and their associated Bézier surfaces $\mathbf{p}(u,v)$, $\mathbf{q}(u,v)$. We also use the Bézier grid labeling from the previous section, identifying three rows of Bézier points $\mathbf{p}_0,\ldots,\mathbf{p}_3$, $\mathbf{q}_0,\ldots,\mathbf{q}_3$, and $\mathbf{b}_0,\ldots,\mathbf{b}_3$ (see Fig. 5.3b). Let the irregular node v_I have valence n with $n \geq 3$ and correspond to a position of $\mathbf{p}(1,0)$ and $\mathbf{q}(0,0)$ on the Bézier surfaces. We identify three relevant quad strips: The two distinct strips running parallel to e through f_p and f_q have widths a and b. Let the strip running transversally through e, thus including both f_p and f_q, have a width of c.

For such a configuration, Surface Splines derive their Bézier point construction from a reparametrization corresponding to the constant and quadratic connection functions [Pet95a]

$$\alpha(v) = 1\,, \qquad \beta(v) = 1\,, \qquad \gamma(v) = 2C_n(1-v)^2$$

where

$$C_n = \cos\left(\frac{2\pi}{n}\right)\,.$$

We now modify these connection functions by scaling them with the inverse widths of the quad strips a, b, c:

$$\alpha(v) = \frac{1}{a}\,, \qquad \beta(v) = \frac{1}{b}\,, \qquad \gamma(v) = \frac{2C_n}{c}(1-v)^2\,.$$

In order to transform from connection functions to explicit continuity constraints in terms of the Bézier points, we must insert our choice of α, β, γ into Eq. (5.1). To do this, we first translate all polynomial factors γ, $\mathbf{p_u}$, $\mathbf{p_v}$, $\mathbf{q_u}$ to scaled Bernstein form, which allows for a more terse notation and simpler multiplication of polynomials. For γ, we get

$$\gamma(v) = \frac{2C_n}{c}(1-v)^2 = \frac{2C_n}{c}\,[1,0,0] \ .$$

The cross-boundary derivatives become

$$\begin{aligned}\mathbf{p_u}(1,v) &= 3\sum_{j=0}^{3} B_j^3(v)\,(\mathbf{b}_j - \mathbf{p}_j) \\ &= 3\,[\mathbf{b}_0 - \mathbf{p}_0,\ 3(\mathbf{b}_1 - \mathbf{p}_1),\ 3(\mathbf{b}_2 - \mathbf{p}_2),\ \mathbf{b}_3 - \mathbf{p}_3]\end{aligned}$$

and, analogously

$$\mathbf{q_u}(0,v) = 3\,[\mathbf{q}_0 - \mathbf{b}_0,\ 3(\mathbf{q}_1 - \mathbf{b}_1),\ 3(\mathbf{q}_2 - \mathbf{b}_2),\ \mathbf{q}_3 - \mathbf{b}_3] \ .$$

Similarly, the versal derivative $\mathbf{p_v}$ translates to

$$\begin{aligned}\mathbf{p_v}(1,v) &= 3\sum_{i=0}^{2} B_j^2(v)\,(\mathbf{b}_{j+1} - \mathbf{b}_j) \\ &= 3\,[\mathbf{b}_1 - \mathbf{b}_0,\ 2(\mathbf{b}_2 - \mathbf{b}_1),\ \mathbf{b}_3 - \mathbf{b}_2] \ .\end{aligned}$$

To further simplify the notation, we introduce the following shorthands for the vectors of Bézier point differences

$$\mathbf{p}'_i = \mathbf{b}_i - \mathbf{p}_i \ , \qquad \mathbf{q}'_i = \mathbf{q}_i - \mathbf{b}_i \ , \qquad \mathbf{b}'_i = \mathbf{b}_{i+1} - \mathbf{b}_i \ , \tag{5.3}$$

bringing the derivatives to the form

$$\begin{aligned}\mathbf{p_u}(1,v) &= 3\,[\mathbf{p}'_0,\ 3\mathbf{p}'_1,\ 3\mathbf{p}'_2,\ \mathbf{p}'_3] \ , \\ \mathbf{q_u}(1,v) &= 3\,[\mathbf{q}'_0,\ 3\mathbf{q}'_1,\ 3\mathbf{q}'_2,\ \mathbf{q}'_3] \ , \\ \mathbf{p_v}(1,v) &= 3\,[\mathbf{b}'_0,\ 2\mathbf{b}'_1,\ \mathbf{b}'_2] \ .\end{aligned}$$

In order to substitute back into the G^1 condition, we break down Eq. (5.1) into its three terms

$$\underbrace{\alpha(v)\cdot\mathbf{p_u}(1,v)}_{A} - \underbrace{\beta(v)\cdot\mathbf{q_u}(0,v)}_{B} + \underbrace{\gamma(v)\cdot\mathbf{p_v}(1,v)}_{C} = 0$$

so we can substitute for each term separately:

$$
\begin{aligned}
A &= \frac{3}{a}\,[\mathbf{p}'_0,\ 3\mathbf{p}'_1,\ 3\mathbf{p}'_2,\ \mathbf{p}'_3]\ , \\
B &= \frac{3}{b}\,[\mathbf{q}'_0,\ 3\mathbf{q}'_1,\ 3\mathbf{q}'_2,\ \mathbf{q}'_3]\ , \\
C &= \frac{6C_n}{c}\,[1,\ 0,\ 0]\,[\mathbf{b}'_0,\ 2\mathbf{b}'_1,\ \mathbf{b}'_2]\ .
\end{aligned}
$$

The product of the two scaled Bernstein polynomials in C can be computed by a simple discrete convolution (cf. Section 2.2.2), resulting in a polynomial in five coefficients, i. e.

$$
C = \frac{6C_n}{c}\,[\mathbf{b}'_0,\ 2\mathbf{b}'_1,\ \mathbf{b}'_2,\ 0,\ 0]\ .
$$

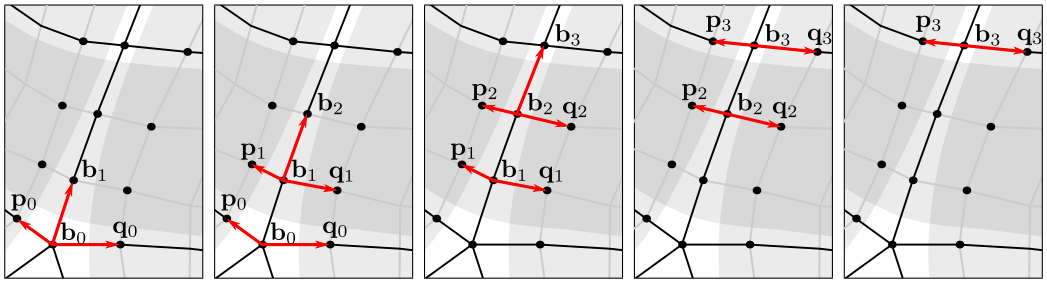
Next, we bring also the polynomials A and B to a representation in five coefficients by raising their degree by one. This is achieved by a convolution with the identity polynomial $[1, 1]$:

$$
\begin{aligned}
A &= \frac{3}{a}\,[\mathbf{p}'_0,\ \mathbf{p}'_0 + 3\mathbf{p}'_1,\ 3\mathbf{p}'_1 + 3\mathbf{p}'_2,\ 3\mathbf{p}'_2 + \mathbf{p}'_3, \mathbf{p}'_3]\ , \\
B &= \frac{3}{b}\,[\mathbf{q}'_0,\ \mathbf{q}'_0 + 3\mathbf{q}'_1,\ 3\mathbf{q}'_1 + 3\mathbf{q}'_2,\ 3\mathbf{q}'_2 + \mathbf{q}'_3, \mathbf{q}'_3]\ .
\end{aligned}
$$

Since the elements of the scaled Bernstein basis are linearly independent, the condition $A - B + C = 0$ holds if and only if all coefficients fulfill the same equality. Hence, we can compare the coefficients of A, B and C separately, which yields the following five constraints (after rearranging to $C = B - A$ and dropping the common factor of 3):

$$
\begin{aligned}
\tfrac{2C_n}{c}\mathbf{b}'_0 &= \tfrac{1}{b}\mathbf{q}'_0 - \tfrac{1}{a}\mathbf{p}'_0\ , && \text{(IC1}'\text{)} \\
\tfrac{4C_n}{c}\mathbf{b}'_1 &= \tfrac{1}{b}\,(\mathbf{q}'_0 + 3\mathbf{q}'_1) - \tfrac{1}{a}\,(\mathbf{p}'_0 + 3\mathbf{p}'_1)\ , && \text{(IC2}'\text{)} \\
\tfrac{2C_n}{c}\mathbf{b}'_2 &= \tfrac{1}{b}\,(3\mathbf{q}'_1 + 3\mathbf{q}'_2) - \tfrac{1}{a}\,(3\mathbf{p}'_1 + 3\mathbf{p}'_2)\ , && \text{(IC3}'\text{)} \\
0 &= \tfrac{1}{b}\,(3\mathbf{q}'_2 + \mathbf{q}'_3) - \tfrac{1}{a}\,(3\mathbf{p}'_2 + \mathbf{p}'_3)\ , && \text{(IC4}'\text{)} \\
0 &= \tfrac{1}{b}\mathbf{q}'_3 - \tfrac{1}{a}\mathbf{p}'_3\ . && \text{(IC5}'\text{)}
\end{aligned}
$$

Note the resemblance of these constraints to those given in [Pet95a] and [WN01] for biquartic Bézier surfaces. By expanding the differences $\mathbf{p}'_i$, $\mathbf{q}'_i$, $\mathbf{b}'_i$ according to Eq. (5.3), we finally obtain a set of constraints in terms of the Bézier points



(a) (IC1) **(b)** (IC2) **(c)** (IC3) **(d)** (IC4) **(e)** (IC5)

Figure 5.4: Bézier points participating in the constraints (IC1) to (IC5) for an irregular G^1 joint. Red arrows indicate difference vectors.

(individual constraints are illustrated in Figs. 5.4a to 5.4e):

$$\frac{2C_n}{c}(\mathbf{b}_1-\mathbf{b}_0) = \frac{1}{b}(\mathbf{q}_0-\mathbf{b}_0) - \frac{1}{a}(\mathbf{b}_0-\mathbf{p}_0) \ , \tag{IC1}$$

$$\frac{4C_n}{c}(\mathbf{b}_2-\mathbf{b}_1) = \frac{1}{b}(\mathbf{q}_0-\mathbf{b}_0+3\mathbf{q}_1-3\mathbf{b}_1) - \frac{1}{a}(\mathbf{b}_0-\mathbf{p}_0+3\mathbf{b}_1-3\mathbf{p}_1) \ , \tag{IC2}$$

$$\frac{2C_n}{c}(\mathbf{b}_3-\mathbf{b}_2) = \frac{1}{b}(3\mathbf{q}_1-3\mathbf{b}_1+3\mathbf{q}_2-3\mathbf{b}_2) - \frac{1}{a}(3\mathbf{b}_1-3\mathbf{p}_1+3\mathbf{b}_2-3\mathbf{p}_2) \ , \tag{IC3}$$

$$0 = \frac{1}{b}(3\mathbf{q}_2-3\mathbf{b}_2+\mathbf{q}_3-\mathbf{b}_3) - \frac{1}{a}(3\mathbf{b}_2-3\mathbf{p}_2+\mathbf{b}_3-\mathbf{p}_3) \ , \tag{IC4}$$

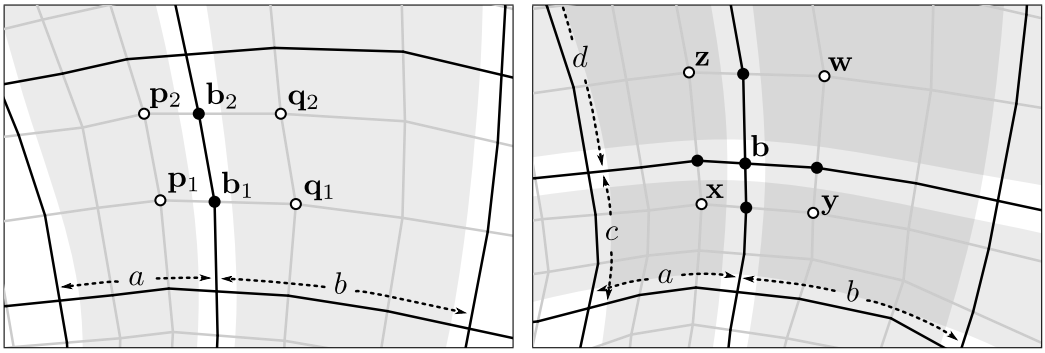
$$0 = \frac{1}{b}(\mathbf{q}_3-\mathbf{b}_3) - \frac{1}{a}(\mathbf{b}_3-\mathbf{p}_3) \ . \tag{IC5}$$

Note how these constraints are in fact a generalization of the regular G^1 joint constraints (RC): If we assume a valence of $n = 4$, we get $C_n = \cos\left(\frac{2\pi}{4}\right) = 0$ and hence the left-hand sides of the conditions (IC1) to (IC5) vanish. The remaining terms on the right-hand sides then reduce to linear combinations of the regular conditions (RC) for $j \in \{0, \ldots, 3\}$.

5.5 Explicit Solution

In Sections 5.3 and 5.4, we devised sufficient conditions for G^1 continuity between Bézier patches by choosing suitable connection functions and then deriving explicit Bézier point constraints. Since all constraints (RC) and (IC1) to (IC5) are linear in the positions of the Bézier points, it is clear that the trivial solution of setting all Bézier points to zero will always be feasible. However, the existence of non-trivial (i. e., non-degenerate) solutions to our chosen constraints is not immediately obvious.

In the following, we prove that in fact, non-trivial solutions for our G^1 surface constraints exist. In resemblance to Surface Splines, we derive a set of construc-



(a) **(b)**

Figure 5.5: Bézier point construction for regular G^1 joints. **(a)**: Interior boundary points $\mathbf{b}_j$ are computed by linear interpolation of the adjacent points $\mathbf{p}_j$, $\mathbf{q}_j$. **(b)**: Regular corner Bézier points $\mathbf{b}$ are computed by bilinear interpolation from the surrounding points $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{w}$.

tion rules that compute feasible Bézier point positions for the entire surface by linear combinations from a smaller set of control point positions.

5.5.1 Regular G^1 Joints

Any Bézier point on a grid boundary corresponding to a regular arc is involved in either one or two regular G^1 constraints: Bézier points at grid corners are subject to two transversal constraints. For the remaining Bézier points on the grid boundaries, just a single constraint (RC) applies. In the latter case, the position of the boundary Bézier point can immediately be computed by just rearranging (RC) to

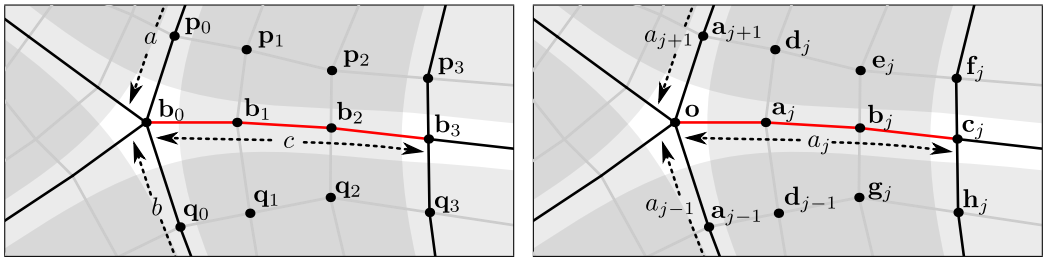
$$\mathbf{b}_j = \frac{b}{a+b}\mathbf{p}_j + \frac{a}{a+b}\mathbf{q}_j \ , \tag{5.4}$$

i. e. by an affine combination of the two adjacent interior Bézier points (Fig. 5.5a).

In the former case, the position of a corner Bézier point $\mathbf{b}$ corresponding to a regular node is computed by bilinear interpolation from the four diagonally surrounding interior Bézier points $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{w}$ (see Fig. 5.5b) by weighting each point with the widths of its two opposing quad strips, i. e.

$$\mathbf{b} = \frac{bd \cdot \mathbf{x} + ad \cdot \mathbf{y} + bc \cdot \mathbf{z} + ac \cdot \mathbf{w}}{(a+b)(c+d)} \ . \tag{5.5}$$

It is easily verified that Bézier points constructed according to Eqs. (5.4) and (5.5) conform to constraint (RC). For patches that only border to regular arcs, the two above rules imply a simple construction scheme: We can arbitrarily



(a) **(b)**

Figure 5.6: Circular re-labeling of the Bézier points and strip widths involved in an irregular G^1 joint. The names used in Section 5.4 **(a)** are replaced by a unique set of new labels **(b)**.

choose the positions of the four interior points of each Bézier grid. The remaining boundary and corner Bézier points are entirely determined by affine combinations of the former. Hence, this scheme allows for four degrees of freedom per regular Bézier patch.

5.5.2 Irregular G^1 Joints

Irregular G^1 joints arise only across the arcs incident to an irregular node. Since the continuity conditions for each arc involve three rows of Bézier grid points, the constraints interlock around the central node and hence cannot be considered independently of each other. Hence, we regard the construction of the Bézier grids of the patches surrounding an irregular node as a single, unified problem.

Consider the following setup: Given an irregular node v of valence $n \geq 3$, we identify the n outgoing arcs by $e_0, \ldots, e_{n-1}$ in counterclockwise order. For the Bézier points, we adopt the following unique labeling (see Fig. 5.6): The common center Bézier point is called $\mathbf{o}$. For each arc e_j, the subsequent Bézier points along the corresponding grid boundary are denoted by $\mathbf{a}_j$, $\mathbf{b}_j$, $\mathbf{c}_j$. The interior Bézier points diagonally adjacent to $\mathbf{o}$ (often also referred to as *twist points* or *twist coefficients*) are called $\mathbf{d}_j$, and the following Bézier points on the same row are labeled $\mathbf{e}_j$ and $\mathbf{f}_j$. For the Bézier points opposite to $\mathbf{e}_j$ and $\mathbf{f}_j$, we use the labels $\mathbf{g}_j$ and $\mathbf{h}_j$. In addition, we denote the strip widths of the n quad strips passing by the central node v by $a_0, \ldots, a_{n-1}$ where a_j indicates the strip width of $\operatorname{strip} e_j$. In the following, we adopt the convention that all indices are understood to be taken modulo n, e. g. $\mathbf{a}_n = \mathbf{a}_0$, or $\mathbf{a}_{-1} = \mathbf{a}_{n-1}$, etc. In our new labeling scheme, the

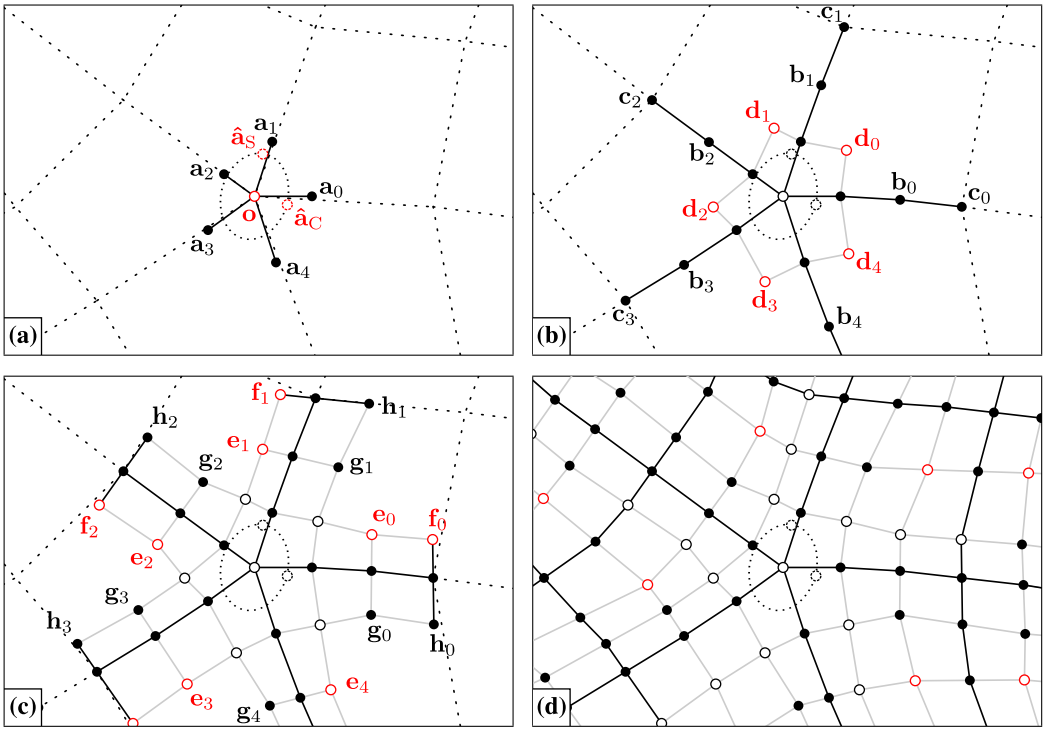


Figure 5.7: Incremental construction of Bézier points around an irregular node. In each step, new free control points (hollow red dots) are introduced, determining the positions of the dependent Bézier points (solid black dots).

conditions (IC1) to (IC5) translate to

$$\frac{2C_n}{a_j}\,(\mathbf{a}_j - \mathbf{o}) = \frac{1}{a_{j-1}}\,(\mathbf{a}_{j-1} - \mathbf{o}) - \frac{1}{a_{j+1}}\,(\mathbf{o} - \mathbf{a}_{j+1})\ , \tag{IC1}$$

$$\begin{aligned}\frac{4C_n}{a_j}\,(\mathbf{b}_j - \mathbf{a}_j) = {}& \frac{1}{a_{j-1}}\,(\mathbf{a}_{j-1} - \mathbf{o} + 3\mathbf{d}_{j-1} - 3\mathbf{a}_j) \\ & - \frac{1}{a_{j+1}}\,(\mathbf{o} - \mathbf{a}_{j+1} + 3\mathbf{a}_j - 3\mathbf{d}_j)\ ,\end{aligned} \tag{IC2}$$

$$\begin{aligned}\frac{2C_n}{a_j}\,(\mathbf{c}_j - \mathbf{b}_j) = {}& \frac{1}{a_{j-1}}\,(3\mathbf{d}_{j-1} - 3\mathbf{a}_j + 3\mathbf{g}_j - 3\mathbf{b}_j) \\ & - \frac{1}{a_{j+1}}\,(3\mathbf{a}_j - 3\mathbf{d}_j + 3\mathbf{b}_j - 3\mathbf{e}_j)\ ,\end{aligned} \tag{IC3}$$

$$0 = \frac{1}{a_{j-1}}\,(3\mathbf{g}_j - 3\mathbf{b}_j + \mathbf{h}_j - \mathbf{c}_j) - \frac{1}{a_{j+1}}\,(3\mathbf{b}_j - 3\mathbf{e}_j + \mathbf{c}_j - \mathbf{f}_j)\ , \tag{IC4}$$

$$0 = \frac{1}{a_{j-1}}\,(\mathbf{h}_j - \mathbf{c}_j) - \frac{1}{a_{j+1}}\,(\mathbf{c}_j - \mathbf{f}_j)\ . \tag{IC5}$$

As above, we now derive a series of rules to construct the Bézier network around the irregular node from affine combinations of a set of free control points. The construction proceeds in three major steps, working its way outwards from the center node towards the Bézier points on the outermost ring. Before we delve into the detailed derivation of the construction steps, we give a brief rundown: First,

the central point $\mathbf{o}$ is chosen and the adjacent ring of boundary points $\mathbf{a}_j$ are determined based on the shape of an ellipsoid given by two additional control points $\hat{\mathbf{a}}_\mathrm{C}$ and $\hat{\mathbf{a}}_\mathrm{S}$ (Fig. 5.7a). Next, the twist points $\mathbf{d}_j$ are chosen. Together with $\mathbf{o}$ and the $\mathbf{a}_j$, the $\mathbf{d}_j$ completely determine the positions of the remaining sets of boundary Bézier points $\mathbf{b}_j$ and $\mathbf{c}_j$ (Fig. 5.7b). Finally, the remaining tangent Bézier points $\mathbf{e}_j$, $\mathbf{f}_j$ can be chosen for each arc, which fixes the corresponding opposite tangent points $\mathbf{g}_j$, $\mathbf{h}_j$ (Fig. 5.7c). After these constructions, all Bézier points involved in the G^1 joints across irregular arcs have been placed. The remaining parts of the Bézier network can then be filled in by regular G^1 joints (Fig. 5.7d), as described in the previous section.

Center Point

As a first step, we choose a position for the center point $\mathbf{o}$, which represents our first degree of freedom. Since all subsequent points are constructed relative to the center point, changing $\mathbf{o}$ just results in a translation of the entire configuration. Hence, without loss of generality, we assume $\mathbf{o} = 0$ in the following to simplify some calculations.

First Ring

Next, we need to determine the positions for the first ring of boundary Bézier points $\mathbf{a}_0, \ldots, \mathbf{a}_{n-1}$ around $\mathbf{o}$. In particular, we want to find all points $\mathbf{a}_0, \ldots, \mathbf{a}_{n-1}$ that fulfill constraint (IC1), which can be rearranged to

$$\tfrac{2C_n}{a_j}\mathbf{a}_j = \tfrac{1}{a_{j-1}}\mathbf{a}_{j-1} + \tfrac{1}{a_{j+1}}\mathbf{a}_{j+1}$$

and, by substituting $\frac{1}{a_j}\mathbf{a}_j = \hat{\mathbf{a}}_j$,

$$2C_n \cdot \hat{\mathbf{a}}_j = \hat{\mathbf{a}}_{j-1} + \hat{\mathbf{a}}_{j+1} \,. \tag{5.6}$$

By gathering all points $\hat{\mathbf{a}}_j$ in a vector $\hat{\mathbf{a}} = (\hat{\mathbf{a}}_0, \ldots, \hat{\mathbf{a}}_{n-1})^\mathsf{T}$, we can represent Eq. (5.6) by a matrix-vector product

$$2C_n \cdot \hat{\mathbf{a}} = \begin{pmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & 1 & 0 & 1 & \\ & & \ddots & \ddots & \ddots \\ 1 & & & 1 & 0 \end{pmatrix} \hat{\mathbf{a}} = \mathbf{C} \cdot \hat{\mathbf{a}} \tag{5.7}$$

where $\mathbf{C} \in \mathbb{R}^{n \times n}$. Note how Eq. (5.7) takes the form of an eigenvector problem: All feasible solutions $\hat{\mathbf{a}}$ are eigenvectors of $\mathbf{C}$ to the eigenvalue $2C_n$. To find the eigenvectors, we exploit the fact that $\mathbf{C}$ is a *circulant matrix*, i. e. that all rows

of $\mathbf{C}$ are subsequent cyclic permutations of the first row. For circulant matrices, eigenvalues and eigenvectors have well-known closed-form expressions [Gra06], based on the entries $c_0, \ldots, c_{n-1}$ of the first row. In general, $\mathbf{C}$ has the eigenvalues and corresponding eigenvectors

$$\lambda_j = \sum_{k=0}^{n-1} c_k \cdot \omega_j^k \quad \text{and} \quad \mathbf{v}_j = \left(\omega_j^0,\ \omega_j^1,\ \ldots,\ \omega_j^{n-1}\right)^\mathsf{T}$$

for $j \in \{0, \ldots, n-1\}$ with

$$\omega_j^k = e^{\frac{2\pi i j k}{n}} = \cos\frac{2\pi jk}{n} + i \sin\frac{2\pi jk}{n} \ .$$

In our case, we have $c_1 = c_{n-1} = 1$ and $c_k = 0$ for all other indices. Hence, $\mathbf{C}$ has the eigenvalues $\lambda_j = \omega_j^1 + \omega_j^{n-1}$. In particular, for $j = 1$, we get

$$\begin{aligned}
\lambda_1 &= \omega_1^1 + \omega_1^{n-1} \\
&= e^{\frac{2\pi i}{n}} + e^{\frac{2\pi i(n-1)}{n}} \\
&= e^{\frac{2\pi i}{n}} + e^{-\frac{2\pi i}{n}} \\
&= \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n} + \cos\frac{2\pi}{n} - i\sin\frac{2\pi}{n} \\
&= 2\cos\frac{2\pi}{n} = 2C_n \ ,
\end{aligned}$$

thus λ_1 is the eigenvalue we are interested in. The corresponding complex-valued eigenvector $\mathbf{v}_1 \in \mathbb{C}^n$ is given by the coefficients

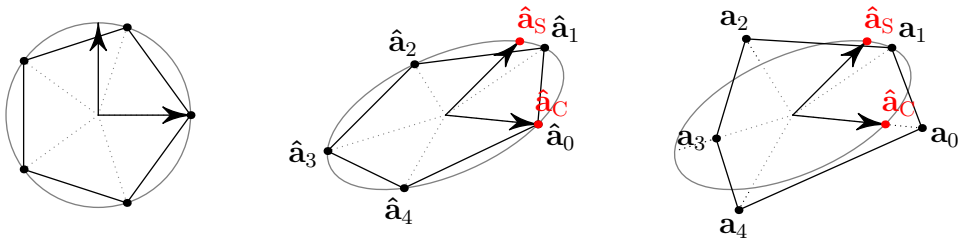
$$\mathbf{v}_1^{[k]} = \cos\frac{2\pi k}{n} + i\sin\frac{2\pi k}{n}$$

where the superscript $[k]$ indicates the k-th coefficient of $\mathbf{v}_1$. If $\mathbf{v}_1$ is an eigenvector of $\mathbf{C}$ for λ_1, then so is each linear combination of $\Re(\mathbf{v}_1)$ and $\Im(\mathbf{v}_1)$ [HR03]. Hence, the two-dimensional eigenspace of $\mathbf{C}$ for the eigenvalue $\lambda_1 = 2C_n$ is spanned by the two real-valued eigenvectors $\mathbf{v}_\mathrm{C}, \mathbf{v}_\mathrm{S} \in \mathbb{R}^n$, given by

$$\mathbf{v}_\mathrm{C}^{[k]} = \cos\frac{2\pi k}{n} \quad \text{and} \quad \mathbf{v}_\mathrm{S}^{[k]} = \sin\frac{2\pi k}{n} \ .$$

Therefore, if we choose two control points $\hat{\mathbf{a}}_\mathrm{C}, \hat{\mathbf{a}}_\mathrm{S} \in \mathbb{A}$ and use them as linear weights for $\mathbf{v}_\mathrm{C}$ and $\mathbf{v}_\mathrm{S}$, we get a solution for Eq. (5.7):

$$\mathbf{C} \cdot (\hat{\mathbf{a}}_\mathrm{C} \cdot \mathbf{v}_\mathrm{C} + \hat{\mathbf{a}}_\mathrm{S} \cdot \mathbf{v}_\mathrm{S}) = 2C_n \cdot (\hat{\mathbf{a}}_\mathrm{C} \cdot \mathbf{v}_\mathrm{C} + \hat{\mathbf{a}}_\mathrm{S} \cdot \mathbf{v}_\mathrm{S}) \ ,$$



(a) **(b)** **(c)**

Figure 5.8: A regular polygon inscribed in the unit circle **(a)** is mapped to an affine regular polygon determined by two control points $\hat{\mathbf{a}}_C$ and $\hat{\mathbf{a}}_S$ **(b)**. A concentric scaling of individual vertices yields the final points $\mathbf{a}_0, \ldots, \mathbf{a}_{n-1}$ **(c)**.

implying

$$\hat{\mathbf{a}} = \hat{\mathbf{a}}_C \cdot \mathbf{v}_C + \hat{\mathbf{a}}_S \cdot \mathbf{v}_S$$

or, for each individual element of $\hat{\mathbf{a}}$,

$$\hat{\mathbf{a}}_j = \cos\left(\frac{2\pi j}{n}\right)\hat{\mathbf{a}}_C + \sin\left(\frac{2\pi j}{n}\right)\hat{\mathbf{a}}_S \ . \tag{5.8}$$

Note the geometric interpretation of Eq. (5.8): The points $\hat{\mathbf{a}}_j$ correspond to the image of a regular polygon inscribed in the unit circle after a linear transformation (see Figs. 5.8a and 5.8b). Such objects, known as *affine regular polygons*, and their connection with circulant matrices are well understood [BGS65; Cox92].

We now undo our previous substitution $\hat{\mathbf{a}}_j = \frac{1}{a_j}\mathbf{a}_j$, resulting in

$$\mathbf{a}_j = a_j \cos\left(\frac{2\pi j}{n}\right)\hat{\mathbf{a}}_C + a_j \sin\left(\frac{2\pi j}{n}\right)\hat{\mathbf{a}}_S \ . \tag{5.9}$$

which corresponds to a concentric scaling of the individual corners of the affine regular polygon by independent scale factors a_j (Fig. 5.8c). With Eq. (5.9), we can construct all Bézier points $\mathbf{a}_j$ of the innermost ring from just the two control points $\hat{\mathbf{a}}_C$, $\hat{\mathbf{a}}_S$, i. e. two degrees of freedom.

As linear combinations of two points, the positions $\mathbf{a}_j$ all lie in a common plane around the center vertex $\mathbf{o}$. Since the tangent directions at the Bézier patch corners are given by $\mathbf{a}_j - \mathbf{o}$, all tangent vectors are coplanar as well and hence the spline surface has a consistent tangent plane at the center node.

Second Ring

Next, we construct the twist points, i. e. the Bézier points $\mathbf{d}_j$ diagonally adjacent to the previously derived points $\mathbf{a}_j$, in a way that fulfills the second constraint (IC2).

Since by our previous construction of $\mathbf{a}_j$, constraint (IC1) now holds, we can substitute (IC1) into (IC2) and obtain, after some rearranging,

$$\tfrac{1}{a_{j-1}}\mathbf{d}_{j-1} + \tfrac{1}{a_{j+1}}\mathbf{d}_j = \left(\tfrac{1}{a_{j-1}} + \tfrac{1}{a_{j+1}} - \tfrac{2}{3}\tfrac{C_n}{a_j}\right)\mathbf{a}_j + \tfrac{4}{3}\tfrac{C_n}{a_j}\mathbf{b}_j \ . \tag{5.10}$$

Multiplying both sides with $\frac{1}{a_j}$ gives

$$\tfrac{1}{a_{j-1}a_j}\mathbf{d}_{j-1} + \tfrac{1}{a_j a_{j+1}}\mathbf{d}_j = \left(\tfrac{1}{a_{j-1}a_j} + \tfrac{1}{a_j a_{j+1}} - \tfrac{2}{3}\tfrac{C_n}{a_j^2}\right)\mathbf{a}_j + \tfrac{4}{3}\tfrac{C_n}{a_j^2}\mathbf{b}_j$$

where we substitute $\hat{\mathbf{d}}_j = \frac{1}{a_j a_{j+1}}\mathbf{d}_j$ and the entire right-hand side by $\mathbf{r}_j$, allowing us to simplify to

$$\hat{\mathbf{d}}_{j-1} + \hat{\mathbf{d}}_j = \mathbf{r}_j \ . \tag{5.11}$$

By gathering the individual terms into column vectors $\hat{\mathbf{d}} = (\hat{\mathbf{d}}_0, \ldots, \hat{\mathbf{d}}_{n-1})^\mathsf{T}$ and $\mathbf{r} = (\mathbf{r}_0, \ldots, \mathbf{r}_{n-1})^\mathsf{T}$, we can express the system of equations (5.11) by a matrix-vector product

$$\begin{pmatrix} 1 & & & & 1 \\ 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \hat{\mathbf{d}}_0 \\ \vdots \\ \hat{\mathbf{d}}_{n-1} \end{pmatrix} = \mathbf{D} \cdot \hat{\mathbf{d}} = \mathbf{r}$$

where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is again a circulant matrix. Now, somewhat surprisingly, it turns out that the matrix $\mathbf{D}$ is only invertible if n is odd. For an even valence n of the irregular node, it is possible to form the last row of $\mathbf{D}$ by summing up the first $n-1$ rows with alternating signs, hence $\operatorname{rank}\mathbf{D} = n-1 \neq n$ and thus $\mathbf{D}$ is singular. For a fixed right-hand side $\mathbf{r}$, it is then impossible in general to find a solution for $\hat{\mathbf{d}}$. This famous problem, known as the *vertex enclosure problem*, *N-vertex problem* or *parity phenomenon* is a major difficulty for spline constructions that aim to smoothly fill in surfaces between a prescribed network of boundary curves [Wij86] (see also [HL96] for a more thorough analysis).

Luckily, in our case, the boundary curves are not fixed yet: The terms $\mathbf{r}_j$ depend on $\mathbf{a}_j$ and $\mathbf{b}_j$ and while we did already fix all $\mathbf{a}_j$ in the previous step, the $\mathbf{b}_j$ are still free to choose. Hence, we can avoid the dreaded vertex enclosure problem by working the other way around: Instead of trying to find $\hat{\mathbf{d}}_j$ consistent with some prescribed $\mathbf{r}_j$, we go ahead and choose $\hat{\mathbf{d}}_j$ arbitrarily and then compute suitable $\mathbf{r}_j$ accordingly.

Going back to Eq. (5.10), we isolate $\mathbf{b}_j$, resulting in

$$\mathbf{b}_j = \tfrac{3}{4}\tfrac{a_j}{C_n}\left(\tfrac{1}{a_{j-1}}\mathbf{d}_{j-1} + \tfrac{1}{a_{j+1}}\mathbf{d}_j\right) - \left(\tfrac{3}{4}\tfrac{a_j}{C_n}\left(\tfrac{1}{a_{j-1}} + \tfrac{1}{a_{j+1}}\right) - \tfrac{1}{2}\right)\mathbf{a}_j \tag{5.12}$$

which yields the following construction rules: Choose the positions of the twist points $\mathbf{d}_0, \ldots, \mathbf{d}_{n-1}$ freely (n degrees of freedom). Then, compute the positions of $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}$ according to Eq. (5.12). Together, the constructed points fulfill condition (IC2).

Third Ring

The remaining constraints (IC3), (IC4), (IC5) govern the placement of the outermost Bézier points $\mathbf{c}_j$, $\mathbf{e}_j$, $\mathbf{f}_j$, $\mathbf{g}_j$, and $\mathbf{h}_j$. The right-hand sides of the three constraints stem from the coefficients of a quartic Bézier curve after degree raising and hence contain some redundant terms. To resolve this redundancy, we first use constraint (IC5) to eliminate the coefficients $\mathbf{c}_j$, $\mathbf{f}_j$, and $\mathbf{h}_j$ from constraint (IC4) such that only terms in $\mathbf{b}_j$, $\mathbf{e}_j$, and $\mathbf{g}_j$ remain. Next, we use the newly reduced constraint (IC4) to eliminate the former three terms from the right-hand side of (IC3) where the remaining coefficients are $\mathbf{a}_j$, $\mathbf{d}_{j-1}$, and $\mathbf{d}_j$. Together, the reduced constraints read

$$\tfrac{2}{3}\tfrac{C_n}{a_j}(\mathbf{c}_j - \mathbf{b}_j) = \tfrac{1}{a_{j-1}}(\mathbf{d}_{j-1} - \mathbf{a}_j) - \tfrac{1}{a_{j+1}}(\mathbf{a}_j - \mathbf{d}_j)\,, \qquad \text{(RIC3)}$$

$$0 = \tfrac{1}{a_{j-1}}(\mathbf{g}_j - \mathbf{b}_j) - \tfrac{1}{a_{j+1}}(\mathbf{b}_j - \mathbf{e}_j)\,, \qquad \text{(RIC4)}$$

$$0 = \tfrac{1}{a_{j-1}}(\mathbf{h}_j - \mathbf{c}_j) - \tfrac{1}{a_{j+1}}(\mathbf{c}_j - \mathbf{f}_j)\,. \qquad \text{(RIC5)}$$

Having already fixed $\mathbf{a}_j$, $\mathbf{b}_j$, $\mathbf{d}_{j-1}$, and $\mathbf{d}_j$, we can now easily compute $\mathbf{c}_j$ from constraint (RIC3). After that, all n boundary curves, each given by the Bézier points $\mathbf{o}$, $\mathbf{a}_j$, $\mathbf{b}_j$, $\mathbf{c}_j$ are fully determined. The remaining coefficients govern the cross-boundary tangents around $\mathbf{b}_j$ and $\mathbf{c}_j$: $\mathbf{e}_j$ and $\mathbf{g}_j$ are the opposite Bézier points on either side of $\mathbf{b}_j$ and are related by constraint (RIC4). Similarly, (RIC5) constrains the tangent points $\mathbf{f}_j$ and $\mathbf{h}_j$ around $\mathbf{c}_j$. Note how (RIC4) and (RIC5) take the form of regular G^1 joints, which is desired: $\mathbf{c}_j$ is the corner point associated with a regular node adjacent to the center node v across the arc e_j. The constraints (RIC4) and (RIC5) are thus consistent with the regular Bézier point construction (as given in Section 5.5.1) around $\mathbf{c}_j$. This consistency is a direct result of our careful choice of the connection functions γ (see Section 5.4) which let the influence of the versal boundary derivatives vanish at the outer nodes.

To complete our G^1 construction, we choose arbitrary positions for the tangent Bézier points $\mathbf{e}_j$ and $\mathbf{f}_j$. Then, we use (RIC4) and (RIC5) to determine the opposite points $\mathbf{g}_j$ and $\mathbf{h}_j$ accordingly. After that, the regular G^1 configuration at the opposite corner is fully determined by the four points $\mathbf{b}_j$, $\mathbf{c}_j$, $\mathbf{e}_j$, $\mathbf{f}_j$. Hence, by choosing $\mathbf{g}_j$ and $\mathbf{h}_j$, we steal two degrees of freedom from each pair of opposite Bézier patches.

This concludes our derivation of explicit construction rules for the Bézier points around an irregular node. Note how we have only considered the boundary

and tangent Bézier points along the n irregular arcs. For each bicubic patch, there remain four points in the diagonally opposite corner of the irregular node which are entirely independent of our previous constructions. These remaining points are involved in regular G^1 corner constraints and hence contribute one more degree of freedom per patch.

5.5.3 Degrees of Freedom

To reveal the total degrees of freedom of our bicubic G^1 spline surface construction, let us recap the construction rules derived in Sections 5.5.1 and 5.5.2.

At each regular layout node, four Bézier patches meet at the intersection of two regular G^1 joints, allowing the free placement of the four twist Bézier points of the adjacent patches. Thus, we have four degrees of freedom per regular node.

For irregular layout nodes, we derived the following free variables: First, the position of the center node may be arbitrarily chosen. Two additional control points determine the orientation and shape of the inner ring of Bézier points on the boundary. Then, all twist Bézier points around the center node can be chosen individually. Assuming a valence n of the central node, this implies $3 + n$ degrees of freedom for the placement of the interior nodes. However, our construction carries over some constraints into the vicinity of the surrounding nodes: While the surrounding regular nodes would normally each have four degrees of freedom, the G^1 conditions for the irregular arcs introduce additional constraints, eliminating two degrees of freedom of each opposite node. In total, this brings down the degrees of freedom per irregular node to $3 - n$.

Thus, in general, for a patch mesh with regular vertices $V_{\mathcal{P}\mathrm{R}}$ and irregular vertices $V_{\mathcal{P}\mathrm{I}}$, the total number of degrees of freedom is given by

$$4 \cdot |V_{\mathcal{P}\mathrm{R}}| + \sum_{v \in V_{\mathcal{P}\mathrm{I}}} (3 - \operatorname{val} v)$$

where $\operatorname{val} v$ denotes the valence of vertex v.

The construction rules derived in this section have shown that the G^1 continuity constraints proposed in Sections 5.3 and 5.4 indeed have non-trivial solutions, i. e. can be fulfilled by non-degenerate spline surfaces. The free variables of this construction, a set of control points, determine the positions of all Bézier points of the individual patches. If the chosen G^1 aspect ratios are fixed, all Bézier point construction rules reduce to linear combinations of the control points. In that case, this spline model is suitable for a linear least-squares fitting approach such as the one described in Chapter 4.

Chapter 6

Surface Smoothing

In order to suppress oscillations in the fitted spline surface, it is common to introduce an additional fairness energy term that penalizes patches with strong curvature. We adopt the popular thin plate energy for this purpose. For tensor product Bézier surfaces, this energy can be analytically discretized into a quadratic function of the Bézier points and thus lends itself to numerical optimization.

Furthermore, we show that the thin plate energy for Bézier patches in its usual parametric form is not suitable for smoothing spline surfaces with non-uniform patch sizes. As a possible remedy, we propose a simple modification based on a scale heuristic obtained from the input mesh that retains the quadratic form of the fairness energy.

6.1 Thin Plate Energy

The *thin plate energy* (*TPE*) for a tensor product Bézier surface $\mathbf{b}(u, v)$ is obtained by integrating the squared magnitudes of the (mixed) second-order partial derivatives over the parameter domain $[0, 1]^2$, i. e. by

$$E = \int_0^1 \int_0^1 \|\mathbf{b_{uu}}(u, v)\|_2^2 + 2\,\|\mathbf{b_{uv}}(u, v)\|_2^2 + \|\mathbf{b_{vv}}(u, v)\|_2^2 \;\mathrm{d}u\,\mathrm{d}v\;.$$

If $\mathbf{b}$ is a Bézier surface of degree (n, n), the partial derivatives $\mathbf{b_{uu}}$, $\mathbf{b_{uv}}$ and $\mathbf{b_{vv}}$ are again Bézier surfaces of reduced degrees $(n-2, n)$, $(n-1, n-1)$ and $(n, n-2)$, respectively (recall Eq. (2.9) from Section 2.2.3), whose Bézier points derive from iterated forward differences on the Bézier grid of $\mathbf{b}$. Notably, the resulting grid

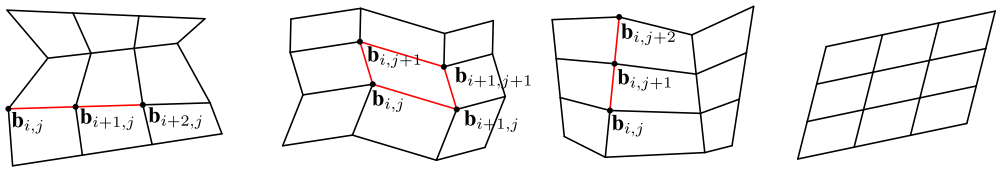


Figure 6.1: Control point conditions for zero-TPE Bézier surfaces. **(a)**: Evenly-spaced collinear grid rows. **(b)**: Parallelogram-shaped grid cells. **(c)**: Evenly-spaced collinear grid columns. **(d)**: All conditions combined.

points are given by vectors proportional to

$$\mathbf{b}_{\mathbf{uu},i,j} = \Delta^{20}\mathbf{b}_{i,j} = \mathbf{b}_{i,j} - 2\mathbf{b}_{i+1,j} + \mathbf{b}_{i+2,j} \ , \tag{6.1}$$

$$\mathbf{b}_{\mathbf{uv},i,j} = \Delta^{11}\mathbf{b}_{i,j} = \mathbf{b}_{i,j} - \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j+1} + \mathbf{b}_{i+1,j+1} \ , \tag{6.2}$$

$$\mathbf{b}_{\mathbf{vv},i,j} = \Delta^{02}\mathbf{b}_{i,j} = \mathbf{b}_{i,j} - 2\mathbf{b}_{i,j+1} + \mathbf{b}_{i,j+2} \ . \tag{6.3}$$

Note how $E = 0$ if and only if $\mathbf{b}_{\mathbf{uu}}(u,v) = \mathbf{b}_{\mathbf{uv}}(u,v) = \mathbf{b}_{\mathbf{vv}}(u,v) = 0$ for all parameters $(u,v) \in [0,1]^2$. Since $\mathbf{b}_{\mathbf{uu}}$, $\mathbf{b}_{\mathbf{uv}}$ and $\mathbf{b}_{\mathbf{vv}}$ are Bézier surfaces, this can only be the case if all their Bézier points are also zero. Hence, by setting Eqs. (6.1) to (6.3) to zero, we obtain Bézier point conditions for Bézier surfaces with minimal TPE:

$$\mathbf{b}_{i+1,j} = \tfrac{1}{2}\left(\mathbf{b}_{i,j} + \mathbf{b}_{i+2,j}\right) \ , \tag{6.4}$$

$$\mathbf{b}_{i+1,j} - \mathbf{b}_{i,j} = \mathbf{b}_{i+1,j+1} - \mathbf{b}_{i,j+1} \ , \tag{6.5}$$

$$\mathbf{b}_{i,j+1} = \tfrac{1}{2}\left(\mathbf{b}_{i,j} + \mathbf{b}_{i,j+2}\right) \ . \tag{6.6}$$

These conditions have quite obvious geometric interpretations which reveal the properties Bézier surfaces assume when exposed to significant TPE smoothing: Equation (6.4) states that for each triplet of successive points $\mathbf{b}_{i,j}$, $\mathbf{b}_{i+1,j}$, $\mathbf{b}_{i+2,j}$ in a row of the Bézier grid of $\mathbf{b}$, the point $\mathbf{b}_{i+1,j}$ in the middle must lie in the center of gravity of the two adjacent points $\mathbf{b}_{i,j}$, $\mathbf{b}_{i+2,j}$ (Fig. 6.1a). Over the entire grid, this condition causes each row to only consist of collinear, equidistantly spaced Bézier points. Equation (6.6) expresses the same notion for columns of the Bézier grid (Fig. 6.1c). The remaining Eq. (6.5) relates the four Bézier points surrounding each grid cell by posing that the bottom edge connecting $\mathbf{b}_{i,j}$ and $\mathbf{b}_{i+1,j}$ must be parallel and of the same length as the top edge between $\mathbf{b}_{i,j+1}$ and $\mathbf{b}_{i+1,j+1}$ (Fig. 6.1b), hence enforcing that each grid cell is a parallelogram.

In conjunction, Eqs. (6.4) to (6.6) restrict the entire Bézier grid of $\mathbf{b}$ to the affine image of a uniform regular grid pattern (see Fig. 6.1d), consisting only of parallel, evenly spaced grid rows and columns. Hence, optimizing the shape of a

Bézier patch by minimizing E converges towards a flat, uniformly parametrized surface which is free of twisting and tapering.

In order to evaluate E, integrals of the squared norms of the three second derivative surfaces must be computed. As it turns out, for each derivative surface, the integral reduces to a quadratic function in the Bézier points of $\mathbf{b}$. If we take $\mathbf{b}$ to represent the column vector of all its Bézier points, the TPE can be written as

$$\begin{aligned} E &= \mathbf{b}^\mathsf{T}\mathbf{F}_{\mathbf{uu}}\mathbf{b} + 2 \cdot \mathbf{b}^\mathsf{T}\mathbf{F}_{\mathbf{uv}}\mathbf{b} + \mathbf{b}^\mathsf{T}\mathbf{F}_{\mathbf{vv}}\mathbf{b} \\ &= \mathbf{b}^\mathsf{T}\left(\mathbf{F}_{\mathbf{uu}} + 2\mathbf{F}_{\mathbf{uv}} + \mathbf{F}_{\mathbf{vv}}\right)\mathbf{b} \\ &= \mathbf{b}^\mathsf{T}\mathbf{F}\mathbf{b}\,. \end{aligned}$$

The derivation of the quadratic coefficients given by the matrices $\mathbf{F}_{\mathbf{uu}}$, $\mathbf{F}_{\mathbf{uv}}$ and $\mathbf{F}_{\mathbf{vv}}$ is somewhat involved and can be found in Appendix A. The sum of the TPEs of multiple Bézier patches can easily be converted into a similar quadratic form by a suitable concatenation of the Bézier point vectors and coefficient matrices, making it possible to evaluate the total TPE of a Bézier spline surface by a single vector-matrix-vector product. By another simple transformation, the total TPE can be expressed in terms of Surface Spline control points instead of individual Bézier points.

As noted in Section 4.6, the least-squares fitting of spline surfaces already involves the minimization of a quadratic fitting energy. Since the TPE is also quadratic, it can be easily incorporated into the existing optimization problem: By just adding a scaled contribution of the TPE to the fitting energy, a joint optimization of fitting and smoothness is achieved without affecting the problem structure. This straightforward compatibility with least-squares optimization has contributed to the broad popularity of the TPE in the context of spline surface fitting.

6.2 Normalization

In its usual definition, the thin plate energy of a Bézier patch is defined by integrating the norms of second-order derivatives over the parameter domain $[0, 1]^2$. As stated in the previous section, this definition evaluates to a quadratic function in terms of iterated forward differences on the Bézier grid. However, due to the fixed domain of integration, thin plate energies measured among different patches with non-uniform sizes are not comparable, as we will show in the following. This can lead to a biased distribution of curvature when optimizing globally for minimum fairness energy of a spline surface.

To motivate the problem, let us consider a simple example of a bicubic Bézier surface $\mathbf{b}(u, v)$ which is only curved along its u parameter direction. The TPE

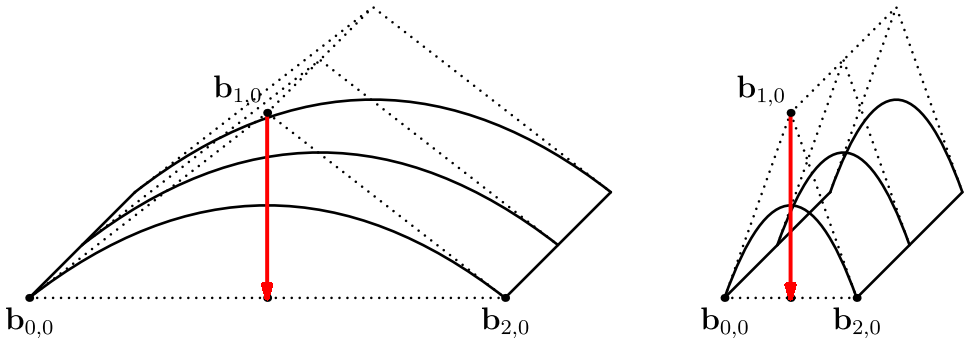


Figure 6.2: The thin plate energy for Bézier patches measures how far grid points deviate from the centroid of their neighbors **(a)**. By scaling the Bézier patch, the curvature of the surface increases but the offset vectors remain unchanged **(b)**.

generated by $\mathbf{b}$ hence depends on the magnitude of the second-order derivative $\mathbf{b_{uu}}$, which is in turn computed from iterated forward differences on the Bézier grid. As discussed in the previous section, the relevant values are proportional to

$$\Delta^{20}\mathbf{b_{uu},}_{0,j} = \mathbf{b}_{1,j} - \frac{1}{2}\left(\mathbf{b}_{0,j} + \mathbf{b}_{2,j}\right) \ ,$$

which has the geometric interpretation of an offset vector pointing from the position of the center grid point $\mathbf{b}_{1,j}$ to the midpoint of the two adjacent grid points $\mathbf{b}_{0,j}$, $\mathbf{b}_{2,j}$ (see Fig. 6.2a). Now, suppose we keep $\mathbf{b}_{1,j}$ stationary while moving $\mathbf{b}_{0,j}$ and $\mathbf{b}_{2,j}$ towards the midpoint $\frac{1}{2}\left(\mathbf{b}_{0,j} + \mathbf{b}_{2,j}\right)$. In doing so, the offset vector $\Delta^{20}\mathbf{b_{uu},}_{0,j}$ does not change, hence also $\mathbf{b_{uu}}(u, v)$ remains constant. However, as the outer Bézier points move closer towards each other, the apex of the resulting Bézier surface (Fig. 6.2b) gradually becomes pointier, i. e. increases in curvature. Since the thin plate energy only depends on the magnitudes of the offset vectors, it fails to represent this increase in surface curvature. As it turns out, the TPE favors large Bézier patches: For small patches, a given magnitude of offset vectors produces a relatively high curvature. With the same offset vectors, a larger patch bends only slightly but produces the same thin plate energy.

In the context of spline fitting, this result explains the susceptibility of the output surface to small oscillations: Since small patches generate less energy when bent, the total energy is best minimized by distributing the total curvature of the spline surface primarily over the smaller patches.

In order to counteract this non-intuitive effect of the thin plate energy, we propose a normalization of the energies of individual Bézier patches based on their sizes. Since the actual patch sizes depend non-linearly on the Bézier points, we instead opt for a simple constant heuristic: As each Bézier patch is optimized

to approximate a given region of the input mesh, we measure the arc lengths of the corresponding embedded quadrilaterals in the input mesh. Under the additional assumption that Bézier patches are roughly rectangular, we can describe the shape of each patch by two scalar sizes w and h, indicating width and height. Based on these measurements, we offer the following scaling strategies for the discretized thin plate energy:

- **No scaling**: This is the usual parametric thin plate energy without any corrective scaling, i. e. the contribution of each patch is

$$E = \mathbf{b}^\mathsf{T} \left(\mathbf{F_{uu}} + 2\mathbf{F_{uv}} + \mathbf{F_{vv}} \right) \mathbf{b} \; .$$

- **Inverse size**: By dividing the directional components of E by the patch length in the corresponding direction, we increase the response of small patches to strong curvature, thereby counteracting the bias in the parametric thin plate energy. We use

$$E = \mathbf{b}^\mathsf{T} \left(\frac{\mathbf{F_{uu}}}{w} + 2\frac{\mathbf{F_{uv}}}{\sqrt{wh}} + \frac{\mathbf{F_{vv}}}{h} \right) \mathbf{b} \; .$$

- **Inverse squared size**: In some cases, it can be beneficial to overcompensate the thin plate energy scaling, thereby intentionally biasing the curvature distribution towards larger patches. For this purpose, we propose the scaling given by

$$E = \mathbf{b}^\mathsf{T} \left(\frac{\mathbf{F_{uu}}}{w^2} + 2\frac{\mathbf{F_{uv}}}{wh} + \frac{\mathbf{F_{vv}}}{h^2} \right) \mathbf{b} \; .$$

Based on the shape of the input object and the provided quad layout embedding, different scaling schemes might be suitable. Hence, we leave this choice to the user as a design parameter. In Section 7.4, we compare the effects of different scaling strategies in a practical approximation setting.

Chapter 7

Results

In the following, we present experimental results from our spline fitting framework. We compare the approximation error achieved by different spline models and sampling strategies and showcase the effects of different choices of fairness energies. We conclude this chapter with performance measurements and a discussion of interesting failure cases of our algorithm.

7.1 Approximation Error

In this section, we focus on the quantitative evaluation of the approximation quality achieved by spline surfaces. For this purpose, we employ the following distance measures: Given an input surface $I \subset \mathbb{A}$ and a spline surface $S \subset \mathbb{A}$ approximating I, we define the *maximum distance* by

$$d_{\max}(I, S) = \max_{\mathbf{x}_I \in I} \min_{\mathbf{x}_S \in S} \{d(\mathbf{x}_I, \mathbf{x}_S)\} \ ,$$

i. e. the directed Hausdorff distance from I to S. Our distance metric d is the Euclidean distance. Since $d_{\max}$ only captures the greatest deviation from I to S, we also use the more descriptive *quadratic mean distance* or *root mean square distance*

$$d_{\text{rms}}(I, S) = \sqrt{\int_{\mathbf{x}_I \in I} \min_{\mathbf{x}_S \in S} \{d(\mathbf{x}_I, \mathbf{x}_S)\}^2 \ \mathrm{d}A}$$

which is closer to an average distance but still penalizes strong outliers.

In order to avoid inconsistencies due to different input object sizes, we give all distance measurements as percentages of the length of the bounding box diagonal of the respective input mesh.

Table 7.1 gives an overview of different measurements for G^1 Bézier spline approximations of a variety of input meshes (shown in Fig. 7.1). We indicate

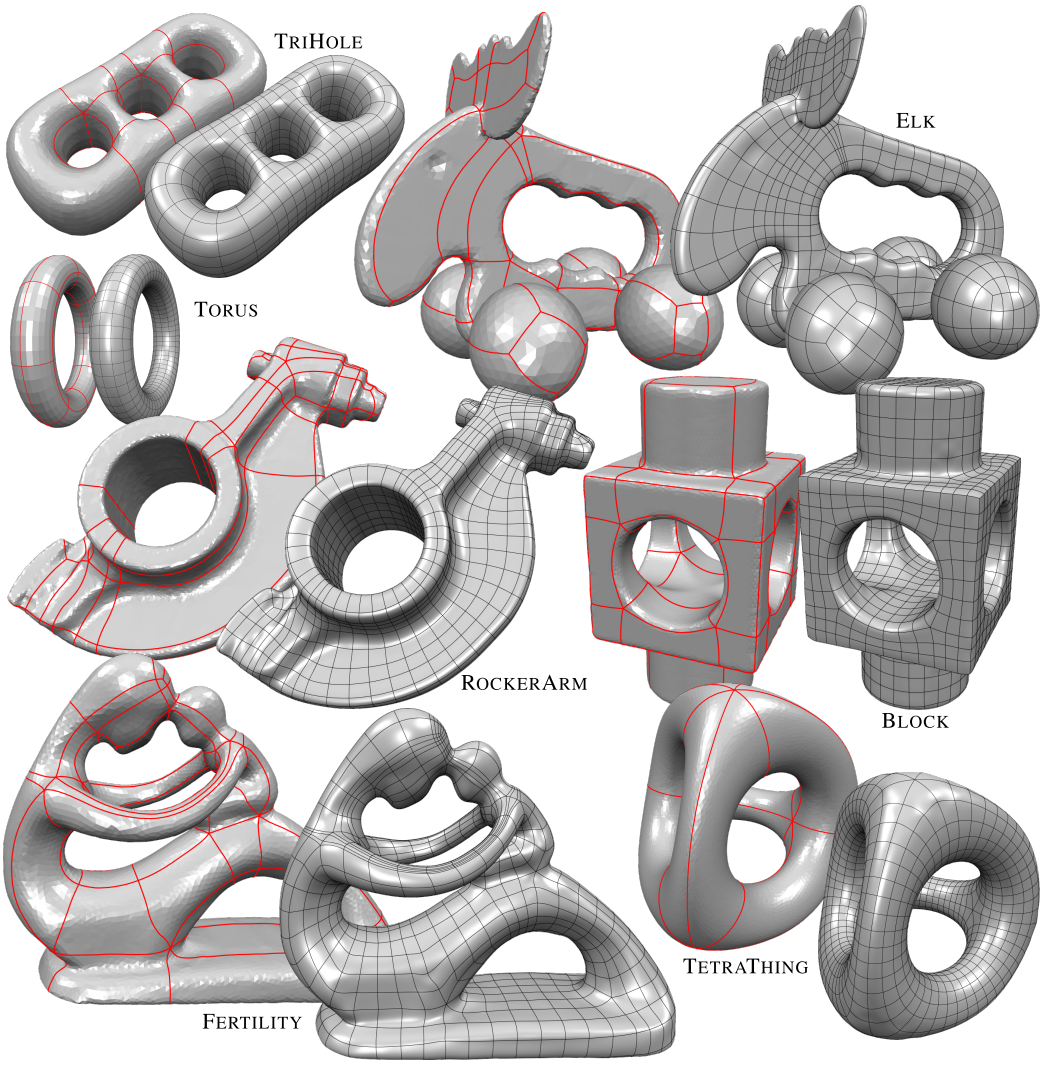


Figure 7.1: Input meshes (with embedded quad layouts, red) and fitted spline surfaces for a variety of models (Bézier patch boundaries shown in black). For mesasurements, see Table 7.1.

the number of patches $|F_{\mathcal{L}}|$ in the input layout and the number of Bézier patches $|F_{\mathcal{P}}|$ obtained after a refinement to a target arc length of $\alpha_{\max} = 0.04$ (i. e. 4 % of the length of the bounding box diagonal). During fitting, we use a weight of $\sigma = 0.01$ for the contribution of the fairness energy term E_{fair}, which is evaluated using the inverse patch size scaling strategy. For all fitted spline surfaces, we measure d_{rms} and $d_{\max}$. All quad layouts embedded in the input meshes have been generated using the methods of Campen *et al.* [CBK12; CK14b]. A notable outlier in our results is the maximum distance $d_{\max}$ measured for the ELK mesh.

Table 7.1: Comparative measurements for G^1 Bézier spline fitting on different input meshes (see Fig. 7.1). Common parameters are $\alpha_{\max} = 0.04$, $\sigma = 0.01$, using inverse size scaling and 4 fitting iterations.

	$\lvert F_{\mathcal{L}}\rvert$	$\lvert F_{\mathcal{P}}\rvert$	d_{rms}	$d_{\max}$	t_{in}	t_{smp}	t_{sol}	t
BLOCK	76	1644	0.033 %	0.296 %	0.20 s	2.1 s	12.9 s	15.2 s
ELK	80	1309	0.099 %	2.022 %	0.10 s	1.5 s	10.3 s	11.9 s
FERTILITY	72	1455	0.035 %	0.440 %	0.25 s	1.9 s	11.6 s	13.7 s
ROCKERARM	115	1602	0.048 %	0.518 %	0.69 s	5.3 s	13.1 s	19.1 s
TETRATHING	12	1710	0.023 %	0.210 %	0.54 s	3.4 s	12.9 s	16.8 s
TORUS	32	672	0.022 %	0.091 %	0.02 s	0.7 s	4.9 s	5.6 s
TRIHOLE	20	824	0.023 %	0.148 %	0.37 s	2.6 s	6.2 s	9.2 s

The relatively high approximation error is due to the shape of the ELK's antlers: Since the rim of the narrow antler plate has a comparably strong curvature, it is subject to significant smoothing and hence suffers some shrinkage.

We also evaluate the effectiveness of our non-symmetric G^1 Bézier spline model against a comparable symmetric spline model. To do so, we repeatedly run the spline fitting algorithm on the same input mesh while varying the target arc length parameter $\alpha_{\max}$, leading to different refinements of the input layout. As $\alpha_{\max}$ decreases, the number of Bézier patches $|F_{\mathcal{P}}|$ increases and the patch sizes approach uniformity. For each refined patch layout, we fit two spline surfaces: For the first surface, we use the G^1 Bézier spline model as described in Chapter 5 where we estimate the strip widths from arc length measurements on the input mesh. For the second surface, we set all aspect ratios to 1, thereby enforcing C^1 continuity across all regular joints. Results of this procedure for the ROCKERARM and TRIHOLE model are shown in Fig. 7.2: As expected, the benefit of non-symmetric G^1 Bézier splines over their symmetric C^1 counterpart is most pronounced for low refinement settings where non-uniform patch sizes are common. At their peak, the non-symmetric constraints achieve a 20 % better RMS approximation error than the symmetric construction for the ROCKERARM model and roughly 40 % for the TRIHOLE model.

Finally, we are interested in the effectiveness of our choice of quad layout generation algorithm. In order to draw a direct comparison with the results of Eck and Hoppe, we reimplement their Surface Spline fitting procedure [EH96]. However, instead of using their decimation-based layout generation technique, we use the curvature-aligned embedded quad layouts generated by the Dual Loops meshing approach [CBK12; CK14b] as base domain for the Surface Spline construction. For the approximation of the TRIHOLE model, Eck and Hoppe achieve approximation errors of $d_{\text{rms}} = 0.07\,\%$ and $d_{\max} = 0.59\,\%$ using a Surface Spline consisting of 2848 Bézier patches (Eck and Hoppe count 178 B-Spline surfaces, each

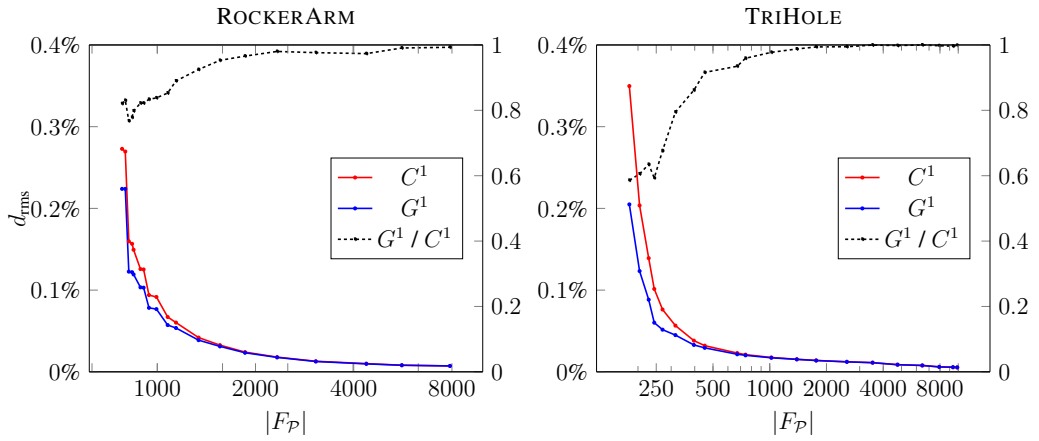


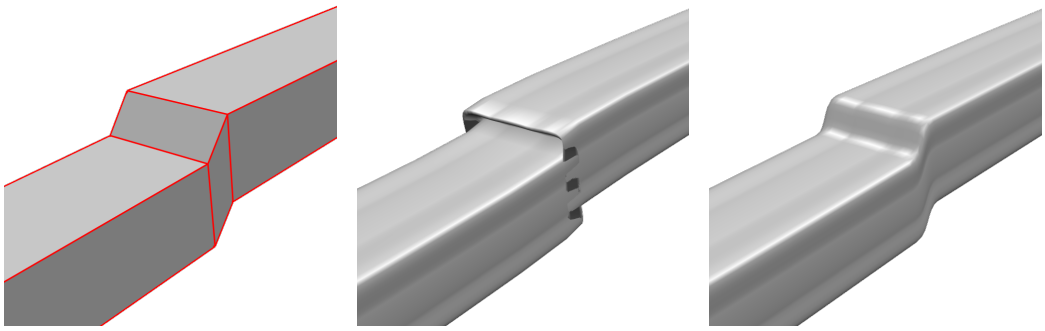
Figure 7.2: RMS approximation error (relative to bounding box diagonal) for different refinement densities (i. e., number of Bézier patches $|F_{\mathcal{P}}|$) of the same input mesh, enforcing C^1 (red) and G^1 (blue) continuity in regular regions. The dotted line shows the quotient of both graphs.

corresponding to 4×4 Bézier patches). For the same model, but using a quad layout generated using Dual Loops meshing, we need significantly fewer Bézier patches to achieve a better approximation error: Using just 710 Bézier patches, the fitted Surface Spline has approximation errors $d_{\text{rms}} = 0.04\,\%$ and $d_{\text{max}} = 0.27\,\%$ w. r. t. the input mesh.

7.2 Spline Model

Fig. 7.3 illustrates a typical failure case for C^1 spline surfaces: For an input layout where a ring of extremely narrow patches is situated between two rings of wide patches (Fig. 7.3a), an approximating C^1 surface produces undesirable foldovers (Fig. 7.3b). These visual artifacts result from the collinearity condition for adjacent Bézier patch tangents (see our discussion in Section 4.8): In relation to their size, the boundary tangent vectors of the outer two patches are relatively large. The C^1 condition enforces identical tangents on the inner, smaller patches, causing their shapes to overshoot their prescribed extents. The G^1 construction, on the other hand, allows for arbitrary tangent length ratios and can hence interpolate the given layout accurately (Fig. 7.3c).

For the Block model (Fig. 7.4a), we demonstrate the possibility to reconstruct spline surfaces with sharp features such as creases or corners. In the extracted intermediate layout mesh, the user can mark selected edges as feature edges (Fig. 7.4b). Across such edges, no tangent plane continuity constraints are generated, allowing the formation of angled features (Fig. 7.4c).

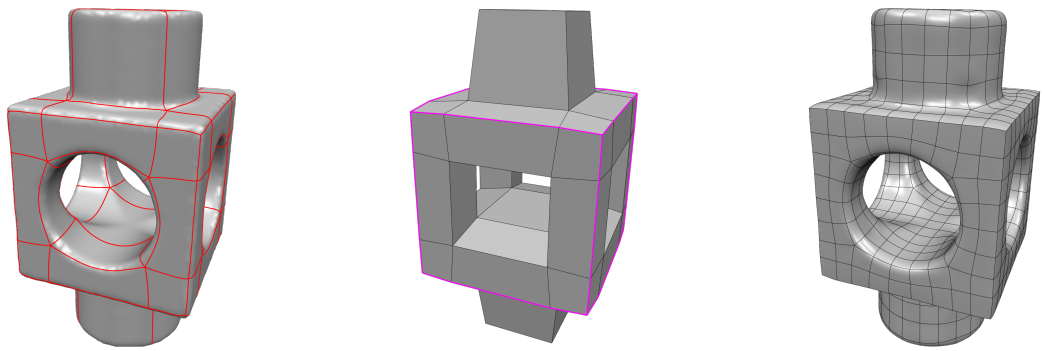


(a) **(b)** **(c)**

Figure 7.3: Approximating an input layout **(a)** with wildly varying patch sizes by a C^1 cubic spline surface leads to foldover artifacts **(b)**, which are remedied by our G^1 spline surface model **(c)**.

7.3 Sampling Strategies

In Section 4.4, we gave two alternative forms of the fitting energy, implemented by different sampling strategies: The parametric fitting energy E_{pfit} which integrates the approximation error over the parameter domain of the spline surface and the uniform fitting energy E_{ufit} which takes the integral over the surface of the input mesh. For the sake of comparing the two alternatives, we run a similar procedure as for the comparison of the C^1 and G^1 constraints: For the ROCKERARM and BLOCK models, we measure the approximation error d_{rms} resulting from an optimization with either of the two energy functionals. As the results depicted in



(a) **(b)** **(c)**

Figure 7.4: For the BLOCK model **(a)**, feature edges (purple) are marked in the layout mesh **(b)**. Feature edges enforce only C^0 continuity between adjacent Bézier patches and hence produce sharp creases in the spline surface **(c)**.

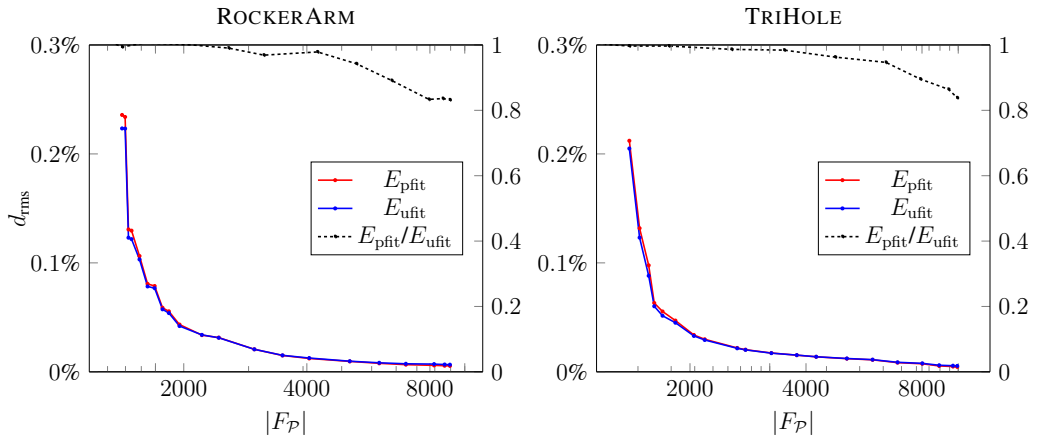


Figure 7.5: Comparison of the RMS fitting error d_{rms} resulting from the optimization of different fitting functionals: Parametric fitting energy E_{pfit}, uniform fitting energy E_{ufit}.

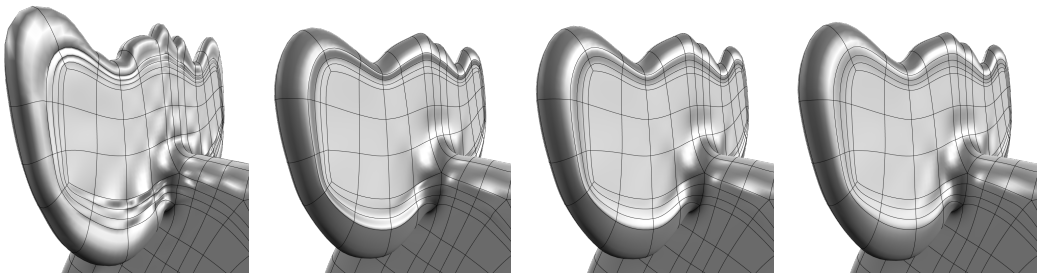
Fig. 7.5 suggest, the choice of fitting energy makes next to no difference in terms of approximation quality, even for coarse, non-uniform layout refinements.

For very small target edge lengths $\alpha_{\max}$ (resulting in large numbers of Bézier patches), we measure a drop in the relative approximation error of the parametric energy E_{pfit}. This, however, has a simple explanation: For the computation of the uniform fitting energy E_{ufit}, the number of samples used is independent of the layout refinement and hence constant over all measurements. In contrast, for the computation of the parametric fitting energy E_{pfit}, a constant number of samples is generated for each patch. Hence, the difference in approximation quality measured for large patch numbers is merely due to a denser, more accurate sampling for E_{pfit}.

7.4 Smoothness

We now turn to a comparison of different weighting schemes for the fairness energy (cf. Section 6.2). A suitable fairness energy should minimize the wrinkles and oscillation artifacts of the least squares fitting while still preserving the shape of small features that are present in the input object. We expect an optimal fitted spline surface to strike a sensible balance between approximation error and overall surface smoothness. It is difficult to find a quantifiable measure for this combined objective. Hence, we resort to just a visual comparison of the effects of different weighting schemes.

Figure 7.6 shows a detail of the Elk model which has been approximated by a G^1 Bézier spline surface. Without the addition of a fairness energy term, the patches on the back side of the antler plate produce significant oscillation arti-



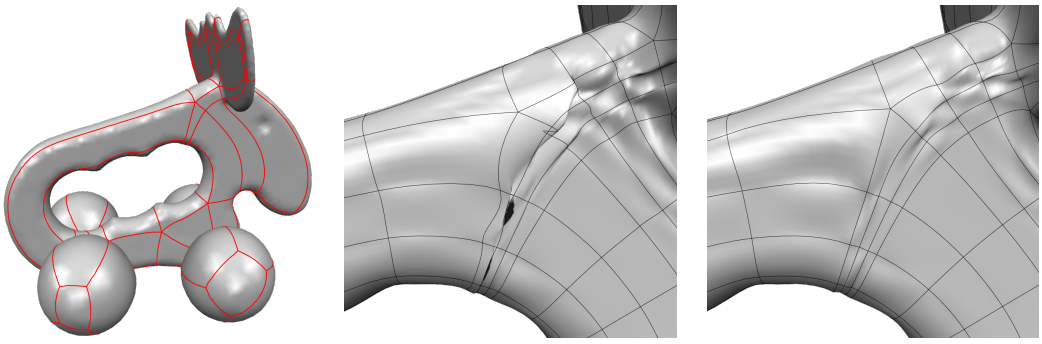
(a) **(b)** **(c)** **(d)**

Figure 7.6: Oscillation artifacts on the back side of the ELK's antlers. **(a)**: No smoothing. **(b)**: Unmodified TPE. **(c)**: TPE with inverse patch size scaling. **(d)**: TPE with inverse squared patch size scaling.

facts (Fig. 7.6a). Figures 7.6b to 7.6d show the same G^1 surface optimized under the addition of some fairness energy intended to suppress the unwanted wrinkles. Different weighting strategies are used: For Fig. 7.6b, we use the unscaled parametric fitting energy. Figures 7.6c and 7.6d depict the results of using the inverse size and inverse squared size weighting strategies, respectively. In all three cases, the added smoothing causes a noticeable loss of detail in the spline surface, most apparent in the spiky features in the center region of the antlers. At the same time, the smoothing fails to completely eliminate the wrinkle artifacts: In Fig. 7.6b, the outer ring of small Bézier patches still exhibits a visible crease. Moving on to inverse size and inverse squared size weighting, the intensity of the remaining artifacts gradually reduces (Figs. 7.6c and 7.6d).

7.5 Performance

Our algorithm executes in a matter of seconds for typical input models. As indicated by our measurements in Table 7.1, all tested input meshes require a total processing time t of less than 20 seconds. We also measure the time spent in different stages of the algorithm: t_{in} indicates the time taken for the processing of the input data including the extraction and refinement of the layout mesh. t_{smp} measures the time spent sampling the input mesh and spline surface for corresponding points and t_{sol} gives the time required to set up and solve the constrained linear least squares system. Evidently, the total run time of the algorithm is dominated by the solver (t_{sol}) while the contribution of the input processing and refinement (t_{in}) is almost negligible.



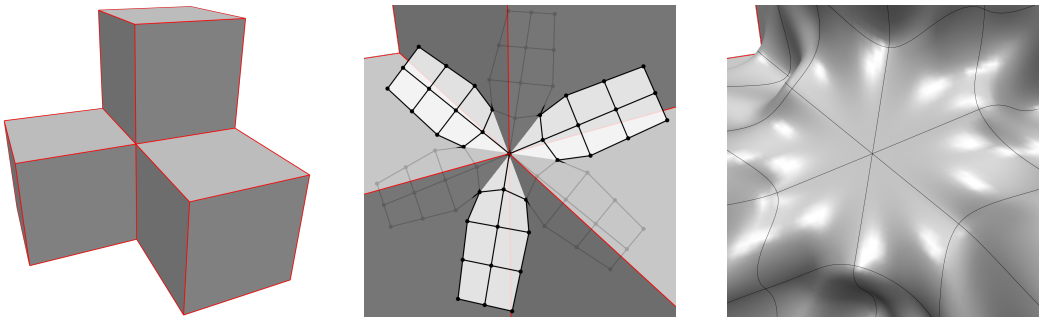
(a) **(b)** **(c)**

Figure 7.7: The the ELK model **(a)** is susceptible to foldover artifacts in the C^1 spline approximation **(b)**. Using G^1 Bézier splines mitigates the issue somewhat but small surface oscillations remain a problem **(c)**.

7.6 Failure Cases

In practice, foldover artifacts arise for relatively low refinement densities in some of our test meshes, such as ELK, where a narrow layout strip near the shoulder provokes the issue (Fig. 7.7a). Using cubic C^1 interpolation, the fitted spline surface exhibits very unpleasant self-intersections (Fig. 7.7b). By using G^1 splines, the problem of foldovers is mitigated but some very noticeable oscillation artifacts remain (Fig. 7.7c). In this case, there are two complicating factors that contribute to this problem: Firstly, our G^1 spline model rests on the assumption that quad strips have an approximately uniform width. Judging from Fig. 7.7c, this assumption is obviously violated: The quad strip passing left by the irregular layout node on the ELK's shoulder visibly narrows towards the bottom. Secondly, the quad strip widths used for the aspect ratios of the G^1 constraints are estimated by measuring arc lengths of the quad layout embedded in the input mesh. However, there is no guarantee that this heuristic choice of the strip widths leads to an optimal approximating surface. Due to these circumstances, our G^1 spline model fails to accurately reflect the geometry of the patch configuration in its continuity constraints, hence resulting in a non-optimal approximation.

In such cases, we have two options to still achieve a more adequate approximation: We can request a finer layout mesh refinement (by decreasing $\alpha_{\max}$), resulting in more uniform patch sizes and more degrees of freedom, or we can increase the influence σ of the fairness energy term E_{fair} and try to smooth out the small oscillations. Both remedies have their drawbacks: Refining the layout increases the complexity of the output spline surface, hence reducing its degree of abstraction. Stronger smoothing, on the other hand, runs the risk of suppressing desired features at similar frequencies as the unwanted oscillations. In general,



(a) **(b)** **(c)**

Figure 7.8: **(a)**: Layout with an irregular node surrounded by non-coplanar arcs. The Bézier networks **(b)** of the approximating spline surface becomes locally flat, causing visible artifacts **(c)**.

finding suitable refinement and smoothing parameters $\alpha_{\max}$ and σ involves a lot of guesswork and can be quite time-consuming.

Our derivation of construction rules for G^1 Bézier splines in Section 5.5 has shown that for irregular G^1 joints, the choice of some control points has non-local influence and can affect adjacent patches. Due to this, there are certain input configurations that exceed the flexibility of a bicubic Bézier spline model and hence cannot be effectively approximated, as has also been noted by [GZ94]. An example for such a problematic configuration is shown in Fig. 7.8a: At the central common node of the three visible cube elements, a 6-valent irregular node arises. Note how the six incident arcs correspond to alternatingly convex and concave edges of the model and do not lie in a common plane. Since the G^1 spline model enforces a consistent tangent plane everywhere, the region around the irregular node is approximated by a diagonal regression plane, determining the tangent Bézier points $\mathbf{a}_j$ which alternatingly lie above or below their corresponding arcs (Fig. 7.8b). Due to the symmetry of the input configuration, the twist Bézier points $\mathbf{d}_j$ are placed diagonally in between the $\mathbf{a}_j$, lying in the same tangent plane. Together with the center node, the $\mathbf{a}_j$ and $\mathbf{d}_j$ completely determine the positions of all boundary Bézier points on the six irregular arcs. Since the $\mathbf{a}_j$ and $\mathbf{d}_j$ are coplanar, so are all affine combinations of them, hence all Bézier patch boundaries around the irregular node are straight lines. Even worse, due to the continuity constraints at opposite patch corners, the coplanarity propagates even further into the tangent Bézier points of the subsequent Bézier patches.

The coplanar configuration is visible in the resulting spline surface (Fig. 7.8c) as a roughly hexagonal, locally flat region protruding from the surface around the irregular node. Although the situation in Fig. 7.8 shows a rather contrived example, somewhat more moderate instances of the same problem occur in practice. Generally, irregular nodes located at non-simple saddle points (e. g. *monkey*

saddles) of the input surface suffer from the insufficient flexibility of the spline model. An example are the 6-valent interior nodes of the TETRATHING model, which tend to introduce excessively flat regions to the spline surface at low refinement levels.

Chapter 8

Future Work

To further improve the quality of the spline approximation, we propose several directions for future research, mainly focusing on improvements of the optimization strategy and the capabilities of the spline model.

Adaptive Refinement

Our spline fitting algorithm uses a global target arc length to control the granularity of the layout mesh refinement and hence the number of patches. As an alternative approach, an iterative adaptive refinement strategy could be used: Starting with a very coarse initial patch mesh, the algorithm would repeatedly refine layout regions with high local approximation error until a desired error upper bound is satisfied. We expect our G^1 Bézier splines to be especially suitable for such local refinement schemes as the generated quad layouts are expected to exhibit strongly varying patch sizes.

Surface Optimization

Our experiments have shown that the heuristic of measuring G^1 aspect ratios on the input mesh is generally acceptable and leads to improved results in comparison to fixed C^1 aspect ratios. However, it is not yet clear whether this heuristic taps the full potential of the G^1 Bézier spline model. Likely, even better surface approximations could be achieved if we optimized the control points and the aspect ratios of the spline surface simultaneously. In such a simultaneous optimization setting, the previously quadratic objective function would become quartic and non-convex, hence complicating an efficient numerical solution.

An obvious problem in the current approximation setting are artifacts of the least-squares fitting which manifest themselves as visible oscillations in the output surface. Here, we see two possible solutions: While the default thin plate energy

applies a global smoothing by forcing all surface curvatures towards zero, it might prove more effective to measure curvature on the input surface and optimize the curvature of the spline to locally match the measured data. The second, more radical alternative is to abandon the least-squares fitting altogether and move on to a non-linear optimization setting instead.

Spline Model

The aspect ratios used for the G^1 conditions in our G^1 Bézier spline model are derived from the assumption that strip widths in the embedded quad layout have approximately uniform widths. However, for many input meshes, especially those with quite complex geometry, the arising quad layouts have strips whose thickness varies across the mesh, hence violating our assumption. In such cases, the additional flexibility offered by G^1 Bézier splines is severely restricted.

To overcome such restrictions, the assumption of uniform quad strip width needs to be abandoned. We suspect that a more granular control over the G^1 aspect ratios might be beneficial. It is however not obvious how additional degrees of freedom for the choice of G^1 aspect ratios should be introduced.

A distinctive advantage of Surface Splines is their capacity for intuitive editing of the resulting spline surface by manipulating control points. In contrast, manual editing of a G^1 Bézier spline surface is currently not viable: Some of the free control points of the model exert non-local influence in highly unpredictable ways. In order to make G^1 Bézier splines more suitable for interactive manipulation, the model must be brought to an alternative form using different control points, preferably such that all remaining Bézier points can be derived only from convex combinations.

When spline surfaces are used to model objects with shiny or reflective surface materials (e. g. car bodies), G^1 continuity typically no longer suffices: Since specular highlights depend on the surface normals which, in turn, depend on first-order derivatives, the intensity of visible highlights will vary with only G^0 continuity across the surface. For our spline surfaces, this leads to noticeable artifacts in the specular highlights resembling bilinear interpolation. The only way to avoid such defects is to move on to higher orders of G^r continuity of the spline surface. Since already for Surface Splines, the step from C^1 to C^2 continuity comes with a significant increase in complexity, we expect the derivation of a G^2 Bézier spline model to become similarly involved. However, for our spline model to have practical merit, this move is unavoidable.

Chapter 9

Conclusion

In this report, we have outlined a general framework for the evaluation of spline surface fitting methods. Our experimental results have shown that the results of conventional spline reconstruction methods can be significantly improved by using state-of-the-art high-quality quad layouts.

Our novel G^1 Bézier spline model, generalizing Surface Splines, allows for non-symmetric continuity constraints between patches. In an analysis of our spline model, we reveal its degrees of freedom and derive explicit construction rules in terms of affine combinations of control points. In our experiments, the G^1 Bézier splines achieve a further increase in approximation quality and visual fidelity over comparable C^1 constructions, especially for quad layouts with non-uniformly sized patches. We envision a variety of future research opportunities related to G^1 Bézier splines e. g. regarding the simultaneous non-linear optimization of aspect ratios and control points, a restricted formulation using fewer degrees of freedom more similar to Surface Splines or a generalization to higher orders of G^r continuity.

Unwanted oscillations in the reconstructed spline surfaces remain a major problem. We have discussed a variety of smoothing strategies to mitigate such artifacts. However, none of these strategies have ultimately proven suitable to suppress oscillations without eliminating desired features due to excessive smoothing. For a fully automatic spline reconstruction pipeline, more sophisticated feature-aware smoothing techniques need to be developed.

Appendix A

Thin Plate Energy Discretization

Consider a single tensor product Bézier surface $\mathbf{b} : [0,1]^2 \to \mathbb{A}$ of degree (n, n). The parametric thin plate energy for $\mathbf{b}$ is given by

$$E = \int_0^1 \int_0^1 \|\mathbf{b_{uu}}(u,v)\|_2^2 + 2\,\|\mathbf{b_{uv}}(u,v)\|_2^2 + \|\mathbf{b_{vv}}(u,v)\|_2^2 \;\mathrm{d}u\,\mathrm{d}v \tag{A.1}$$

where $\mathbf{b_{uu}}$, $\mathbf{b_{uv}}$, $\mathbf{b_{vv}}$ indicate the (mixed) second-order partial derivatives of $\mathbf{b}$ w. r. t. the parameters u and v. We can split the integral in Eq. (A.1) into three terms

$$E = E_{\mathrm{uu}} + 2 \cdot E_{\mathrm{uv}} + E_{\mathrm{vv}}$$

with

$$E_{uu} = \int_0^1 \int_0^1 \|\mathbf{b_{uu}}(u,v)\|_2^2 \;\mathrm{d}u\,\mathrm{d}v \tag{A.2}$$

etc. Since the terms E_{uu}, E_{uv}, E_{vv} are quite similar, we focus our attention in the following on the derivation of E_{uu}. If we assume that the vector space $\mathbb{V}$ underlying our affine space $\mathbb{A}$ is represented by a D-dimensional cartesian coordinate space $\mathbb{R}^D$, the squared norm of a vector $\mathbf{v} \in \mathbb{V}$ just evaluates to a sum of squares:

$$\|\mathbf{v}\|_2^2 = \sum_{d=1}^{D} \left(\mathbf{v}^{[d]}\right)^2 \;,$$

allowing us to split the integral in Eq. (A.2) again into D real-valued terms which can be integrated independently. Hence, in the following, we pretend $\mathbf{b}$ just consists of a single real-valued component, simplifying Eq. (A.2) to

$$E_{\mathrm{uu}} = \int_0^1 \int_0^1 \mathbf{b_{uu}}(u,v)^2 \;\mathrm{d}u\,\mathrm{d}v \;.$$

We can now expand the partial derivatives of the Bézier surface:

$$\int_0^1\int_0^1 \left(\frac{\partial^2}{\partial u^2} \sum_{i=0}^{n}\sum_{j=0}^{n} B_i^n(u) B_j^n(v) \mathbf{b}_{ij} \right)^2 \mathrm{d}u\, \mathrm{d}v$$
$$= \int_0^1\int_0^1 \left(n(n-1) \frac{\partial^2}{\partial u^2} \sum_{i=0}^{n-2}\sum_{j=0}^{n} B_i^{n-2}(u) B_j^n(v) \Delta^{20}\mathbf{b}_{ij} \right)^2 \mathrm{d}u\, \mathrm{d}v$$

where the Δ^{20} denote forward differences on the Bézier grid. Multiplying out the squared term yields

$$n^2(n-1)^2 \int_0^1\int_0^1 \sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n} B_i^{n-2}(u) B_j^n(v) B_k^{n-2}(u) B_l^n(v) \Delta^{20}\mathbf{b}_{ij} \Delta^{20}\mathbf{b}_{kl}\, \mathrm{d}u\, \mathrm{d}v$$
$$= n^2(n-1)^2 \sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n} \Delta^{20}\mathbf{b}_{ij} \Delta^{20}\mathbf{b}_{kl} \int_0^1\int_0^1 B_i^{n-2}(u) B_j^n(v) B_k^{n-2}(u) B_l^n(v)\, \mathrm{d}u\, \mathrm{d}v$$
$$= n^2(n-1)^2 \sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n} \Delta^{20}\mathbf{b}_{ij} \Delta^{20}\mathbf{b}_{kl} \int_0^1 B_i^{n-2}(u) B_k^{n-2}(u)\, \mathrm{d}u \int_0^1 B_j^n(v) B_l^n(v)\, \mathrm{d}v$$
$$= n^2(n-1)^2 \sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n} \Delta^{20}\mathbf{b}_{ij} \Delta^{20}\mathbf{b}_{kl} \cdot \mathcal{B}_{i,k}^{n-2} \cdot \mathcal{B}_{j,l}^{n} \,.$$

Note how we have shortened the separated integral factors to $\mathcal{B}_{i,k}^{n-2}$ and $\mathcal{B}_{j,l}^{n}$ which have the same general form

$$\mathcal{B}_{i,j}^n = \int_0^1 B_i^n(t) B_j^n(t)\, \mathrm{d}t$$
$$= \int_0^1 \binom{n}{i} t^i (1-t)^{n-i} \binom{n}{j} t^j (1-t)^{n-j}$$
$$= \binom{n}{i}\binom{n}{j} \int_0^1 t^{i+j} (1-t)^{2n-i-j} \,. \tag{A.3}$$

As observed by [FR88], the integral in Eq. (A.3) turns out to be an instance of the Euler beta function

$$\beta(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\, \mathrm{d}t$$

with arguments $x = i + j + 1$ and $y = 2n - i - j + 1$. The Euler beta function has the known closed-form solution

$$\beta(x, y) = \frac{(x-1)!\,(y-1)!}{(x+y-1)!}\ .$$

Substituting back into Eq. (A.3) yields

$$\begin{aligned}
\mathcal{B}_{i,j}^n &= \binom{n}{i}\binom{n}{j}\beta(i+j+1,\ 2n-i-j+1) \\
&= \binom{n}{i}\binom{n}{j}\frac{(i+j)!\,(2n-i-j)!}{(2n+1)!} \\
&= \binom{n}{i}\binom{n}{j}\frac{(i+j)!\,(2n-i-j)!}{(2n+1)!}\frac{1}{\binom{2n}{i+j}}\frac{(2n)!}{(i+j)!\,(2n-i-j)!} \\
&= \binom{n}{i}\binom{n}{j}\frac{1}{2n+1}\frac{1}{\binom{2n}{i+j}} \\
&= \frac{1}{2n+1}\frac{\binom{n}{i}\binom{n}{j}}{\binom{2n}{i+j}}\ .
\end{aligned}$$

Using this, we can discretize the thin plate energy as

$$\begin{aligned}
&n^2(n-1)^2\sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n}\Delta^{20}\mathbf{b}_{ij}\Delta^{20}\mathbf{b}_{kl}\cdot\mathcal{B}_{i,k}^{n-2}\cdot\mathcal{B}_{j,l}^{n} \\
&= n^2(n-1)^2\sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n}\Delta^{20}\mathbf{b}_{ij}\Delta^{20}\mathbf{b}_{kl}\frac{1}{2n-3}\frac{\binom{n-2}{i}\binom{n-2}{k}}{\binom{2n-4}{i+k}}\frac{1}{2n+1}\frac{\binom{n}{j}\binom{n}{l}}{\binom{2n}{j+l}} \\
&= \sum_{i,k=0}^{n-2}\sum_{j,l=0}^{n}F_{\mathrm{uu}}^{ijkl}\Delta^{20}\mathbf{b}_{ij}\Delta^{20}\mathbf{b}_{kl} \qquad \text{(A.4)}
\end{aligned}$$

with

$$F_{\mathrm{uu}}^{ijkl} = \frac{n^2(n-1)^2}{(2n-3)(2n+1)}\frac{\binom{n-2}{i}\binom{n-2}{k}}{\binom{2n-4}{i+k}}\frac{\binom{n}{j}\binom{n}{l}}{\binom{2n}{j+l}}\ . \qquad \text{(A.5)}$$

We now change the indexing scheme on the Bézier grid from using index pairs (i, j), (k, l) to single indices p, q, i. e. $\mathbf{b}_p = \mathbf{b}_{i,j}$ and $\mathbf{b}_q = \mathbf{b}_{k,l}$, simplifying Eq. (A.4) to

$$\sum_p\sum_q F_{\mathrm{uu}}^{pq}\Delta^{20}\mathbf{b}_p\Delta^{20}\mathbf{b}_q \qquad \text{(A.6)}$$

Since the iterated forward differences $\Delta^{20}\mathbf{b}_p$, $\Delta^{20}\mathbf{b}_q$ are linear combinations of the grid points, each can be represented by a dot product

$$\Delta^{20}\mathbf{b}_p = \mathbf{d}_p^{\mathsf{T}}\mathbf{b} \qquad \text{(A.7)}$$

where $\mathbf{d}_p^\mathsf{T}$ is a row vector containing the coefficients that contribute to $\Delta^{20}\mathbf{b}_p$ and $\mathbf{b}$ is a column vector containing all Bézier points of the patch. Inserting Eq. (A.7) into Eq. (A.6), we obtain

$$\begin{aligned} E_{\mathrm{uu}} &= \sum_p \sum_q F_{\mathrm{uu}}^{pq} \mathbf{d}_p^\mathsf{T} \mathbf{b} \mathbf{d}_q^\mathsf{T} \mathbf{b} \\ &= \sum_p \sum_q F_{\mathrm{uu}}^{pq} \mathbf{b}^\mathsf{T} \mathbf{d}_p \mathbf{d}_q^\mathsf{T} \mathbf{b} \\ &= \mathbf{b}^\mathsf{T} \left(\sum_p \sum_q F_{\mathrm{uu}}^{pq} \mathbf{d}_p \mathbf{d}_q^\mathsf{T} \right) \mathbf{b} \\ &= \mathbf{b}^\mathsf{T} \mathbf{F}_{\mathbf{uu}} \mathbf{b} \end{aligned}$$

which is a quadratic expression in $\mathbf{b}$, determined by the matrix $\mathbf{F}_{\mathbf{uu}}$ whose coefficients are computed from the terms F_{uu}^{pq} as given by Eq. (A.5) and from the outer products $\mathbf{d}_p \mathbf{d}_q^\mathsf{T}$ of the forward difference vectors. Since all factors F_{uu}^{pq} are non-negative and symmetric (i. e. $F_{\mathrm{uu}}^{pq} = F_{\mathrm{uu}}^{qp}$), $\mathbf{F}_{\mathbf{uu}}$ is a symmetric positive semidefinite matrix.

Analogously to the above derivations, the other mixed derivative terms E_{uv} and E_{vv} can be represented by matrices $\mathbf{F}_{\mathbf{uv}}$ and $\mathbf{F}_{\mathbf{vv}}$, allowing us to write the thin plate energy of the Bézier patch as

$$\begin{aligned} E &= \mathbf{b}^\mathsf{T} \mathbf{F}_{\mathbf{uu}} \mathbf{b} + 2 \cdot \mathbf{b}^\mathsf{T} \mathbf{F}_{\mathbf{uv}} \mathbf{b} + \mathbf{b}^\mathsf{T} \mathbf{F}_{\mathbf{vv}} \mathbf{b} \\ &= \mathbf{b}^\mathsf{T} \left(\mathbf{F}_{\mathbf{uu}} + 2 \cdot \mathbf{F}_{\mathbf{uv}} + \mathbf{F}_{\mathbf{vv}} \right) \mathbf{b} \\ &= \mathbf{b}^\mathsf{T} \mathbf{F} \mathbf{b} \,. \end{aligned}$$

The energies of multiple Bézier patches can be computed using a single operation: By concatenating the Bézier points of individual patches to a single column vector $\hat{\mathbf{b}}$, the total energy is given by the product $\hat{\mathbf{b}}^\mathsf{T} \hat{\mathbf{F}} \hat{\mathbf{b}}$ where $\hat{\mathbf{F}}$ is a diagonal block matrix obtained by a concatenation of the individual thin plate matrices $\mathbf{F}$ along the diagonal.

If a Surface Splines construction is used, the Bézier points $\hat{\mathbf{b}}$ are derived from linear combinations of the Surface Spline control points $\hat{\mathbf{c}}$, i. e. by $\hat{\mathbf{b}} = \mathbf{B}\hat{\mathbf{c}}$ (cf. Section 4.5). In that case, the thin plate energy of the surface can be computed directly from the control points by

$$\begin{aligned} E_{\mathrm{fair}}(\hat{\mathbf{c}}) &= \left(\mathbf{B}\hat{\mathbf{c}} \right)^\mathsf{T} \hat{\mathbf{F}} \left(\mathbf{B}\hat{\mathbf{c}} \right) \\ &= \hat{\mathbf{c}}^\mathsf{T} \mathbf{B}^\mathsf{T} \hat{\mathbf{F}} \mathbf{B} \hat{\mathbf{c}} \\ &= \hat{\mathbf{c}}^\mathsf{T} \mathbf{S} \hat{\mathbf{c}} \end{aligned}$$

where $\mathbf{S}$ is again symmetric and positive semi-definite.

Bibliography

[BGS65] E. R. Berlekamp, E. N. Gilbert, and F. W. Sinden. “A Polygon Problem”. In: *The American Mathematical Monthly* 72.3 (1965) (cited on page 52).

[Béz67] Pierre Bézier. “Définition numérique des courbes et surfaces II”. In: *Automatisme* 12 (1967) (cited on page 17).

[Boo62] Carl de Boor. “Bicubic Spline Interpolation”. In: *Journal of Mathematical Physics* 41.3 (1962) (cited on page 17).

[CBK12] Marcel Campen, David Bommes, and Leif Kobbelt. “Dual Loops Meshing: Quality Quad Layouts on Manifolds”. In: *ACM Transactions on Graphics* 31.4 (2012) (cited on pages 20, 23, 64, 65).

[CK14a] Marcel Campen and Leif Kobbelt. “Dual Strip Weaving: Interactive Design of Quad Layouts Using Elastica Strips”. In: *ACM Transactions on Graphics* 33.6 (2014) (cited on pages 20, 23, 41).

[CK14b] Marcel Campen and Leif Kobbelt. “Quad Layout Embedding via Aligned Parameterization”. In: *Computer Graphics Forum* 33.8 (2014) (cited on pages 20, 64, 65).

[Cas63] Paul de Casteljau. *Courbes et surfaces à pôles*. Tech. rep. Paris: A. Citroën, 1963 (cited on page 17).

[CC78] Edwin Catmull and James Clark. “Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes”. In: *Computer-Aided Design* 10.6 (1978) (cited on page 18).

[Cox92] H. S. M. Coxeter. “Affine Regularity”. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*. Vol. 62. 1. Springer. 1992 (cited on page 52).

[DBG+06] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. “Spectral Surface Quadrangulation”. In: *ACM Transactions on Graphics* 25.3 (2006) (cited on page 19).

[DS78] Daniel Doo and Malcolm Sabin. “Behaviour of Recursive Division Surfaces near Extraordinary Points”. In: *Computer-Aided Design* 10.6 (1978) (cited on page 18).

[EH96] Matthias Eck and Hugues Hoppe. “Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. ACM, 1996 (cited on pages 19, 21, 29, 33, 65).

[Far02] Gerald Farin. *Curves and Surfaces for CAGD*. 5th edition. Morgan Kaufman, 2002 (cited on page 3).

[FR88] R. T. Farouki and V. T. Rajan. “Algorithms for Polynomials in Bernstein Form”. In: *Computer Aided Geometric Design* 5.1 (1988) (cited on pages 6, 78).

[FS96] Gregory E. Fasshauer and Larry L. Schumaker. “Minimal Energy Surfaces Using Parametric Splines”. In: *Computer Aided Geometric Design* 13.1 (1996) (cited on page 33).

[FH02] Michael Floater and Kai Hormann. “Parameterization of triangulations and unorganized points”. In: *Tutorials on Multiresolution in Geometric Modelling*. Springer, 2002, pp. 287–316 (cited on page 20).

[GR74] William J. Gordon and Richard F. Riesenfeld. “B-Spline Curves and Surfaces”. In: *Computer Aided Geometric Design* 167 (1974) (cited on page 17).

[Gra06] Robert M. Gray. “Toeplitz and Circulant Matrices: A Review”. In: *Foundations and Trends in Communications and Information Theory* 2.3 (2006) (cited on page 51).

[GZ94] John A. Gregory and Jianwei Zhou. “Filling polygonal holes with bicubic patches”. In: *Computer Aided Geometric Design* 11.4 (1994) (cited on page 71).

[Gre94] Günther Greiner. “Variational Design and Fairing of Spline Surfaces”. In: *Computer Graphics Forum* 13.3 (1994) (cited on pages 21, 33).

[HR03] Guershon Harel and Jeffrey M. Rabin. “Polygons Whose Vertex Triangles Have Equal Area”. In: *The American Mathematical Monthly* 110 (2003) (cited on page 51).

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. "Piecewise Smooth Surface Reconstruction". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. ACM, 1994 (cited on page 21).

[HL96] Josef Hoschek and Dieter Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, 1996 (cited on pages 3, 53).

[JLW10] Zhongping Ji, Ligang Liu, and Yigang Wang. "B-Mesh: A Modeling System for Base Meshes of 3D Articulated Shapes". In: *Computer Graphics Forum* 29.7 (2010) (cited on page 20).

[LRL06] Wan-Chiu Li, Nicolas Ray, and Bruno Lévy. "Automatic and Interactive Mesh to T-Spline Conversion". In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Eurographics Association, 2006 (cited on pages 21, 33).

[LLS01] Nathan Litke, Adi Levin, and Peter Schröder. "Fitting Subdivision Surfaces". In: *Proceedings of the Conference on Visualization '01*. VIS '01. IEEE Computer Society, 2001 (cited on page 21).

[Pet94] Jörg Peters. "Constructing C^1 Surfaces of Arbitrary Topology Using Biquadratic and Bicubic Splines". In: *Designing Fair Curves and Surfaces*. Ed. by Nickolas S. Sapidis. SIAM, 1994 (cited on pages 6, 18).

[Pet95a] Jörg Peters. "Biquartic C^1-Surface Splines over Irregular Meshes". In: *Computer-Aided Design* 27.12 (1995) (cited on pages 18, 43, 45).

[Pet95b] Jörg Peters. "C^1-Surface Splines". In: *SIAM Journal on Numerical Analysis* 32.2 (1995) (cited on page 18).

[Pet96] Jörg Peters. "Curvature Continuous Spline Surfaces over Irregular Meshes". In: *Computer Aided Geometric Design* 13.2 (1996) (cited on page 19).

[Pet02] Jörg Peters. "Geometric Continuity". In: *Handbook of Computer Aided Geometric Design*. Ed. by Gerald Farin, Josef Hoschek, and Myung-Soo Kim. Elsevier, 2002 (cited on page 11).

[PBP02] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002 (cited on page 3).

[RLL+06] Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. "Periodic Global Parameterization". In: *ACM Transactions on Graphics* 25.4 (2006) (cited on page 20).

[Rei95] Ulrich Reif. “A Unified Approach to Subdivision Algorithms near Extraordinary Vertices”. In: *Computer Aided Geometric Design* 12.2 (1995) (cited on page 18).

[SR03] J. Sánchez-Reyes. “Algebraic Manipulation in the Bernstein Form Made Simple via Convolutions”. In: *Computer-Aided Design* 35.10 (2003) (cited on page 6).

[SZB+03] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri. “T-Splines and T-NURCCs”. In: *ACM Transactions on Graphics* 22.3 (2003) (cited on page 17).

[SWW+04] Xiquan Shi, Tianjun Wang, Peiru Wu, and Fengshan Liu. “Reconstruction of Convergent Smooth B-Spline Surfaces”. In: *Computer Aided Geometric Design* 21.9 (2004) (cited on page 33).

[Sta98] Jos Stam. “Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values”. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’98. New York, NY, USA: ACM, 1998 (cited on page 18).

[TDN+12] Julien Tierny, Joel Daniels II, Luis Gustavo Nonato, Valerio Pascucci, and Cláudio T. Silva. “Interactive Quadrangulation with Reeb Atlases and Connectivity Textures”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.10 (2012) (cited on page 20).

[Ver75] Kenneth James Versprille. “Computer-Aided Design Applications of the Rational B-Spline Approximation Form”. PhD thesis. Syracuse University, 1975 (cited on page 17).

[WN01] Geir Westgaard and Horst Nowacki. “Construction of Fair Surfaces over Irregular Meshes”. In: *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*. SMA ’01. ACM, 2001 (cited on pages 21, 33, 45).

[Wij86] Jarke J. van Wijk. “Bicubic Patches for Approximating Non-Rectangular Control-Point Meshes”. In: *Computer Aided Geometric Design* 3.1 (1986) (cited on page 53).

Index