Piecewise Smooth Surfaces with Features

by

Denis C. Kovacs

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May 2013

Denis Zorin

© Denis C. Kovacs

All Rights Reserved, 2013

Acknowledgements

First of all, I want to thank my family. Over the years, there were countless holidays that I spent working towards some imminent deadline. My family was always understanding and supportive and made sure all I had to worry about was my work.

My academic path has been a long and winding road, and I have encountered many great minds that heavily influenced the way I perceive and approach scientific problems.

My highschool teacher Lutz-Peter Tägert and my dad taught me to tinker and supported my early experiments in any way they could no matter how crazy the ideas became.

Prof. Folkmar Bornemann and Dr. Christian Ludwig sparked my interest in Numerical Methods and introduced me to the power of MATLAB. Prof. Armin Leutbecher's excellent seminar exposed me to dyadic fractions, the namesake for Dyadic T-meshes in Chapter 2. Prof. Christian Zenger and Prof. Hans-Joachim Bungartz further strengthened my interest in numerical methods and PDEs.

Andreas Krahnke and Harald Brunnhofer helped me scheme my jump across the pond.

I thank my Master's thesis advisor Prof. Christopher Beattie, who was patient with my humble attempts at research and was a great mentor. When I had difficulties chosing my PhD program, he told me that while all options were great schools, he recommended going to New York City for my "personal development" – the best advice anybody could have given me at this crucial point in my academic life.

During my studies, I had the privilege to do several internships in the graphics

industry. I am deeply indebted to Yury Uralski, Ashu Rege, Cem Cebenoyan and Ignacio Castaño (NVIDIA), Jason Mitchell, Joe Demers and Bay Raitt (who inspired the work in Chapter 2 during my summer at Valve), and Philip Schneider and Vivek Verma (ILM).

Christoph Groth, Michael Kiermaier, Johannes Haas and Evgeny Savelev shared parts of my academic journey, always two steps ahead, pulling me along, never letting me slow down.

At NYU, I found a lab that always provided a stimulating environment. I want to especially thank Ashish Myles, Julian Panetta, James Zhou, Ross Goroshin, Adam Gashlin, Pierre Franquin, Ofir Weber, Nico Pietroni, Koray Kavukcuoglu, Jason Reisman, Yotam Gingold, Elif Tosun, Adrian Secord, Harper Langston and Ilya Rosenberg for many enlightening discussions over the years.

Most importantly, I would like to thank my dissertation advisor, Prof. Denis Zorin, for believing in me and giving me the opportunity to follow my curiosity freely in some of the topics described below. It is always a pleasure to see his mind at work, especially before deadlines. More than anyone, he taught me to focus and to simplify, to re-visit ideas and iterate on them until they are polished, precise and coherent. He was also the perfect counterbalance to my frequent academic mood swings, providing much-needed critical input when I became too excited and the necessary impetus to continue when I was about to give up.

Abstract

The creation, manipulation and display of piecewise smooth surfaces has been a fundamental topic in computer graphics since its inception. The applications range from highest-quality surfaces for manufacturing in CAD to believable animations of virtual creatures in special effects, to virtual worlds rendered in real-time in computer games.

Our focus is on improving the a) mathematical representation and b) automatic construction of such surfaces from finely sampled meshes in the presence of features. Features can be areas of higher geometric detail in an otherwise smooth area of the mesh or sharp creases that contrast with the overall smooth appearance of an object.

In the first part, we build on techniques that define piecewise smooth surfaces to improve their quality in the presence of features. We present a crease technique suitable for real-time applications that increases the perceived visual detail of objects while maintaining a compact representation and efficient evaluation.

We then introduce a new subdivision scheme that allows the use of T-junctions for better local refinement. It thus reduces the need for extraordinary vertices, which can cause surface artifacts especially on animated objects.

In the second part, we consider the problem of building the control meshes of piecewise smooth surfaces so that the resulting surface closely approximates an existing data set (such as a 3D range scan), particularly in the presence of features. To this end, we introduce a simple modification that can be applied to a wide range of parameterization techniques to obtain an anisotropic parameterization. We show that a resulting quadrangulation can indeed better approximate the original surface.

Finally, we present a quadrangulation scheme that turns a data set into a quad mesh with T-junctions, which we then use as a T-Spline control mesh to obtain a smooth surface.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Smooth surface representations are ubiquitous in the computer graphics industry. Their uses range from CAD models in manufacturing to digital creatures and props in the special effects industry and games.

The majority of such models is created by 3D artists, either entirely on the computer or using reference meshes. Various "retopology tools" (e.g. in 3DCoat) have been developed to assist the artist in this process. Recently, tools like ZBrush and Mudbox have emerged that allow artists to "sculpt" models as if with clay rather than "engineering" them by modifying the vertices of an underlying mesh. For animation, such models also have to be translated into a smooth base surface, with the fine-level details added as displacement maps or normal maps.

In contrast, a fully automatic pipeline starts with the acquisition of 3D geometry and outputs a smooth surface closely approximating this data without the need for any user input. The pipeline consists of the following stages:

The *geometry acquisition* stage refers to the creation of the raw 3D input data, usually in the form of a triangle mesh. It can broadly be grouped into

two categories: scans and isosurface extraction. Scanners are usually based on laser or structured light measurements, while isosurface extraction converts volumetric data originating from CT scans, fluid dynamics simulations etc. into triangle meshes. For both categories, the resulting meshes are typically severely oversampled (they contain significantly more vertices than needed to describe the geometry) and have many irregular vertices. In contrast, the vertices of a fully regular triangle mesh away from its boundaries have valence 6, i.e. each vertex is connected to six other vertices by edges. A semi-regular mesh has predominantly regular vertices, with comparatively few exceptions; for quad meshes the regular vertices have valence 4. Meshes with few irregular vertices are desirable for approximation quality reasons (c.f. the fitting stage below).

The *parameterization* stage computes a mapping from the plane to the 3D mesh surface. In general, it is impossible to compute such a mapping without introducing angular or area distortion, a fact we are all familar with from maps of our Earth: in the common Mercator projection, Antarctica appears gigantic. Current state-of-the-art parameterization methods (e.g. [9]) support a number of important features. They compute a global parameterization, i.e. a single map from the plane to the mesh surface, as opposed to several local parameterizations that map the plane to a certain region on the surface. Parameterizations can be feature-aligned, such that integer parameter lines coincide with creases on the mesh. Cones are required to accommodate closed surfaces of genus other than 1; at cones, the angles around a vertex in the plane do not sum to 2π. Introducing further cones also reduces the distortion in the parameterization. A good parameterization scheme can automatically determine a small set of cones that results in a mapping with sufficiently low distortion. Another way to manage area distortion is to allow

for anisotropy in the parameterization (Chapter 4) by prescribing different scale factors along the two principal curvature directions.

The *quadrangulation* stage resamples the input mesh along a grid on the parametric plane and turns it into a quad-mesh. This resulting mesh is semi-regular; most vertices have regular valence 4 with the exception of the cone vertices described in the previous paragraph. To keep the quadrangulation as regular as possible, the parameterization algorithm should therefore introduce as few cones as possible while still achieving low enough distortion.

In the last stage, the quadrangulation is used as the control mesh structure for a higher-order surface representation (i.e. a piecewise polynomial surface that is smoother than piecewise linear elements). Common representations are Catmull-Clark surfaces and their various patch-based approximations and spline surface constructions such as T-splines. The surface quality for these schemes is generally poorer at irregular vertices than in regular areas. For close reproduction, an optimal fit (with respect to a certain metric) can be computed between the input mesh and the evaluated higher-order surface [62].

Fully automatic conversion from dense triangle meshes to higher-order surfaces is a topic of continuing research, and there is still a significant gap in surface quality between surfaces generated by fully automatic methods and surfaces created entirely manually or with human assistance.

1.1 Thesis Organization

This work is structured in two parts. The first part deals with the actual representation of higher-order surfaces. Chapter 2 extends one of the most commonly used

higher-order surfaces in computer graphics (Catmull-Clark subdivision surfaces) to allow more control in the mesh resolution of the underlying quad meshes by introducing T-joints. In Chapter 3, we discuss an extension to a common patch-based Catmull-Clark surface approximation that allows for selective sharp features such as creases.

The second part addresses various aspects of a fully automatic pipeline. Chapter 4 introduces a simple way to add anisotropy to existing parameterization techniques, replacing near-square quads by quads that are stretched along principle-curvature directions. We show that this improves the quality of quadrangulations when the number of quads is kept constant.

Finally, Chapter 5 discusses a fully automatic pipeline, starting from a densely sampled triangle mesh. We first compute a fine quadrangulation using a technique described in [9] and convert it into a coarse T-mesh. We use the T-mesh topology to define a T-spline surface that we fit to the original triangle mesh using a least-squares fit similar to [62].

Part I

Surface Representations

Chapter 2

Dyadic T-mesh Subdivision

This chapter describes our work in extending quad-based subdivision schemes such as Catmull-Clark subdivision and NURSS to support T-joints. T-joints allow for greater local control over mesh density without the need to introduce more extraordinary vertices.

2.1 Introduction

Subdivision surfaces are popular with 3D modeling artists for a variety of reasons. Most obviously, subdivision surfaces support control meshes of arbitrary topology and the predominant quad mesh structure is suitable for many modeling needs, but other, less obvious uses are no less important. Examination of common organic and mechanical models reveals that two uses are very common:

- Varying the mesh resolution between more and less feature-rich areas.
- Modifying connectivity to align features with control mesh edges;

These topological challenges are resolved using extraordinary vertices, most commonly in pairs of valence 3 and 5, redirecting the edge flow (Figure 1).

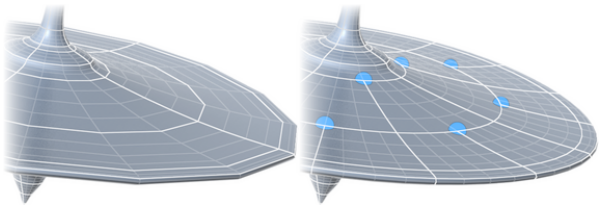


Figure 2.1: *T-joints can be used to keep tessellated faces equal sized and thus prevent under- or over-tessellation.*

At the same time, surface quality at the extraordinary vertices in general cannot match surface quality of the regular parts of the surface, where it reduces to polynomial patches (Figure 2.2). Placing extraordinary vertices at perceptually optimal locations where the reduction in quality is least objectionable is a difficult manual task. Even if a static mesh with carefully chosen extraordinary vertices looks fine, it might not behave well if the mesh is animated.

T-joints (Figure 2.3) offer an additional degree of freedom in choosing the control mesh connectivity. While T-joints do not solve the topology problem (a closed mesh with T-joints but without extraordinary vertices cannot have genus other than 1), the number of extraordinary points needed for other reasons can be greatly reduced.

A significant body of literature (Section 2.2) exists on construction of surfaces with T-joints, with T-splines being the most common construction. Yet, to the best of our knowledge, no subdivision schemes were proposed that operate directly on meshes with T-joints (T-meshes). While patch-based and hybrid patch/subdivision approaches exist, these are relatively complex and require significant separation between T-joints and extraordinary vertices.

Part of the difficulty in integrating T-joints with subdivision is that T-meshes have far richer space of local connectivity configurations compared to meshes with no T-joints. Handling these configurations is a challenge, even for subdivision rules involving few points.

In this chapter, we propose a simple subdivision scheme for *dyadic* T-meshes, with at most one T-joint per quad edge. An additional technical requirement on the connectivity is explained in Section 2.3. The dyadic requirement reduces the variety of possible local configurations, but still provides a rich space for modeling. Our main insight is that a subdivision scheme handling all these configurations for the class of meshes that we consider can be obtained in *factorized* form, based on the original Catmull-Clark and NURCC ideas, and only a small number of cases need to be handled.

We demonstrate that our scheme yields surfaces of good quality for a variety of local configurations including extraordinary vertices and T-joints (in fact, extraordinary vertices themselves can be T-joints) as shown in Figure 2.19, while treating the whole surface in a uniform way and imposing no restrictions on proximity of T-joints and extraordinary vertices.

We conjecture that our surfaces are G^1 at extraordinary points based on the numerical evidence, although a complete analysis is beyond the scope of this work.

We verified the practicality of our algorithm by implementing a plug-in prototype for Autodesk's Maya, together with a set of Maya Python scripts to aid the artist in the creation and manipulation of T-meshes, which we will make publicly available. We also include the central MATLAB code for computing the subdivision masks as an electronic supplement.

2.2 Related Work

The literature on subdivision surfaces is quite broad, but only few works touch on the question of T-joints. Similarly, much work has been done on T-splines and related T-mesh constructions, but with relatively little focus on subdivision schemes.

We refer the reader to a recent survey [17] for a general overview of recent work on subdivision; here we focus on most closely related work on subdivision schemes allowing nonuniform knot spacing on the one hand, and schemes for constructing smooth surfaces on T-meshes on the other hand.

Nonuniform subdivision schemes. Our work is based on one of the earliest schemes for extending nonuniform splines to arbitrary meshes [95] (NURSS), and its restricted version described in [94] (NURCC). In this work, the original factorized form of surface subdivision [20] is extended to arbitrary knot intervals. We show how to apply this factorization in the context of T-meshes. Alternative approaches to constructing subdivision surfaces with nonuniform knots are proposed in [74], [73], and [18]. [74] presents a scheme resulting in stationary subdivision matrices near extraordinary vertices (which yields explicit limit points formulas) while handling arbitrary knot intervals on opposite sides of faces. Factorized form plays an important role in extending higher-order uniform B-Splines to arbitrary control meshes ([116, 101]). The algorithm of [18], based on the factorized form of [19], describes a method for extending NURBS of arbitrary degree to arbitrary meshes with nonuniform knot intervals. In our work, our principal goal is *not* to handle arbitrary knots in full generality; rather, we focus on a restricted version of knot assignments, with matching knot intervals on opposite sides of faces

and dyadic relations between knots, making the minimal extension to conventional subdivision that enable T-mesh refinement.

T-joints on arbitrary meshes. There are several directions of work that introduce T-joints into arbitrary triangle and quad meshes.

Adaptive refinement of basis functions is the most straightforward approach to adding fine-scale degrees of freedom to the surface and, if we take a patch-based point of view, can be considered a restricted form of T-mesh constructions. Variations of this approach were proposed in [38, 117, 69, 55, 5] and many other works, leading to wavelet and multiscale surface representations.

[94] introduces *T-splines*, capable of handling a broad range of local T-configurations, and combines them with subdivision surfaces: a conforming arbitrary mesh can be refined by inserting T-joints in an arbitrary manner allowed by T-splines sufficiently far from extraordinary vertices – in general two steps of subdivision are required to achieve the needed separation. The parts of the surface with T-joints are handled using patches; subdivision is used near extraordinary vertices. The scheme can also handle T-meshes not originating from conforming meshes as long as sufficient separation between T-joints and extraordinary vertices is maintained.

A C^1 polynomial basis construction for T-meshes (PHT splines) was proposed in [33], and extended to meshes with extraordinary vertices in [65] (GPT splines). Due to more local basis function support, this scheme offers greater flexibility and a purely polynomial basis for T-meshes with few restrictions in the regular case, and admits simple analysis [32] (the situation with T-splines is far more complex, e.g., [66, 11, 72]). Extraordinary faces (faces which have at least one extraordinary vertex) cannot share a T-vertex. As these bases require multiple degrees of freedom

per vertex, further adaptation is needed in the context of geometric modeling. Our focus is on designing a scheme that can be easily used in the same context as Catmull-Clark subdivision is currently used.

An important recent application of T-meshes and T-splines is *isogeometric analysis* (see,e.g.,[27]), i.e. methods that use the same high-order basis for geometric modeling and simulation. [3] demonstrates that T-splines have substantial advantages for isogeometric analysis.

Analysis-suitable T-splines are introduced in [66]; restrictions on the T-mesh structure are imposed to ensure that the resulting T-spline spaces are linearly indepedent. Local refinement for T-splines for analysis purposes is studied in [91, 36]. [108] proposes a method for conversion of an arbitrary quad mesh to a control mesh for the analysis-suitable T-spline, which is C^2 away from extraordinary vertices but only C^0 at some of the edges at extraordinary vertices. In FEM simulation applications, lower order of smoothness is acceptable, as long as the approximation order is maintained; in modeling applications however it is essential to keep the surface quality high.

Finally, [75] describes a procedure for automatic conversion of an arbitrary mesh to a coarse T-mesh of quad patches, which often naturally have T-joints adjacent to extraordinary vertices, and require additional refinement to isolate them – another motivation for considering constructions with no such restriction.

2.3 Dyadic T-meshes and T-splines

In this section, we define the fundamental concepts we use to construct our subdivision scheme: *dyadic analysis-suitable* (DAS) T-meshes, T-spline spaces associated

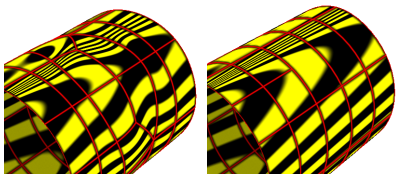


Figure 2.2: *Local refinement on cylinder surface approximated using redirected edge flow and T-joints.*

with regular DAS T-meshes, and a natural (but not very practical) way to construct subdivision rules for these. In the next section, we will show how NURSS rules of [95] can be adapted to DAS T-meshes.

Dyadic T-meshes. In a T-mesh, quad faces may have more than 4 vertices: each face has exactly four *corner* vertices; the remaining vertices are considered *T-joint* vertices with respect to this face. A T-mesh is *dyadic* if any two corner vertices are separated by no more than one T-vertex. While dyadic T-meshes restrict the variety of possible local configurations, the resulting space is still sufficiently rich for many important modeling tasks. The limitations and possible ways to overcome them are discussed in Section 2.8.

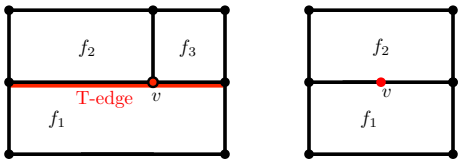


Figure 2.3: *a) Vertex v is a T-joint with respect to face f_1 and a corner for f_2 and f_3. b) Vertex v has valence 2 and is a T-joint w.r.t. both f_1 and f_2.*

A *T-edge* (Figure 2.3a) is a sequence of two edges connecting two corner vertices separated by a T-joint. A T-joint vertex can be a T-joint with respect to several faces, and can be *regular* or *extraordinary*. A regular T-joint has valence 3 (T-joint

with respect to one face), or valence 2 (T-joint with respect to both neighboring faces, see Figure 2.3b). All other T-joints are extraordinary.

For the purposes of constructing patches or subdivision surfaces, the edges of a T-mesh can be annotated by *knot intervals*. If there are no T-joints, the surface is required to be a non-uniform B-spline with the knots determined by knot intervals.

Following T-splines and NURCC, we require *knot interval consistency*: the sum of knot intervals on opposite T-edges of a face are equal. Furthermore we assume that two neighboring edges that form a T-edge to have equal knot intervals. These two restrictions form a system of linear constraints, the independent degrees of freedom are the basis of the nullspace of this system; if a user modifies any knot interval on the mesh, all dependent knot intervals are adjusted. To simplify the user interface, all knot intervals have values $2^k, k \in \mathbb{Z}$.

Analysis-suitable T-meshes and T-splines. Our scheme for regular grids is constructed to yield a restricted version of T-splines in the limit [94]. We associate a spline space $\mathcal{T}$ with a regular T-mesh by assigning a T-spline basis function to each vertex. It is constructed as a standard tensor-product non-uniform B-spline basis function, with the knot sequence determined by extending the edges at each vertex in four parametric directions by two knot intervals (we count a knot interval if the extension passes a vertex of the T-mesh or intersects an edge.) We call two sets of four knot intervals each the *knot vectors* of a T-spline basis function. The parametric location on the horizontal and vertical lines through the vertex constitutes the *basis cross* (Figure 2.5).

In general, the T-spline basis functions do not sum up to one (i.e. do not have the partition-of-unity property) and renormalization is required to achieve

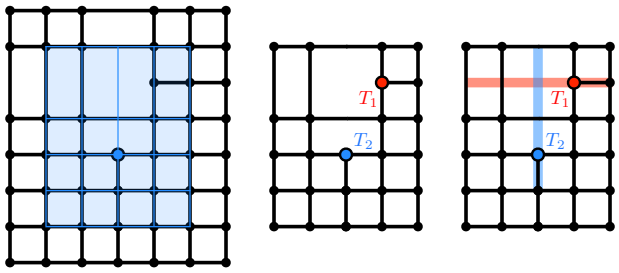


Figure 2.4: *Left: T-spline basis function support; Right: Analysis-suitable T-mesh condition: red and blue extensions of T-joints T_1 and T_2 intersect, so this mesh is not analysis-suitable.*

affine invariance of the basis. A mesh admits a *standard* T-spline basis, if the basis functions constructed as above do sum up to one. The necessary and sufficient conditions for a mesh to be standard formulated purely in terms of mesh connectivity are not known yet.

A simple *sufficient* condition for standard T-splines is based on T-joint extensions. A T-joint *extension* for a *regular* T-vertex can be regarded as a chain of 3 edges (Figure 2.4, right), one along an existing edge, the other one extending across the T-face to the opposite edge and the following face. T-meshes for which the extensions never intersect are called *analysis-suitable* ([66]) (Figure 2.4, right).

In this work, we consider only dyadic analysis-suitable T-meshes. A simple property of DAS T-meshes that immediately follows from the the requirement is that extensions do not intersect:

Proposition 1. *Any face of a DAS T-mesh has at most one T-joint.*

We call a DAS T-mesh face *regular* if it has no T-joints, and a *T-face* if it has one T-joint.

Analysis-suitable T-splines use the same basis functions and inherit a wealth of properties from B-splines: linear independence, partition of unity, affine invariance

and the convex hull property.

Subdivision of T-meshes. The topological subdivision rule on a T-mesh is the standard quadrisection of faces: every face is split into four quads, and every edge is split into two edges, with new control points introduced for every edge and face.

Subdivision rules are defined by masks. A *mask*, supported on a *stencil* S, is a collection of weights w_i assigned to the vertices of the mesh, that is used to obtain a new point P' from points P_i at vertices: $P' = \sum_i w_i P_i$. The stencil on a mesh annotated with knot intervals consists of a set of edges and vertices; the stencil vertices are vertices of the mesh to which nonzero weights are assigned, and stencil edges are edges whose knot intervals affect the weights w_i. In addition to subdivision rules, we need to define how knots are assigned to the edges of the refined mesh. In our case, the two edges obtained by refinement of an edge with knot interval d receive knot intervals $d/2$.

In the rest of this section, we only consider subdivision of regular DAS T-meshes: subdivision rules in this case are naturally derived from the nesting property of corresponding spline spaces. The regular case is a foundation of the extension to general T-meshes.

Nested analysis-suitable T-spline spaces. The subdivision rules in the case of B-splines are a consequence of the *nesting property* of the spline spaces defined for the grid M and once-subdivided grid M^1: each basis function B_i on M is a linear combination of the basis functions B^1_j on the grid M^1. In the case of T-meshes, the nestedness property for quadrisection does not always hold. However, for DAS T-meshes, it does:

Proposition 2. *T-spline space $\mathcal{T}$ associated with a regular dyadic analysis-suitable*

T-mesh T is contained in the space $\mathcal{T}^1$ associated with the once-subdivided mesh T^1.

Chapter 2.9 describes the proof.

This property means that subdivision rules abstractly can be defined in the standard way: given a surface f defined as a linear combination of T-spline basis functions on the coarse mesh $f = \sum_i P_i B_i$, we can replace each coarse basis function with $\sum_j w_{ij} B_j^1$, a linear combination of basis functions on the refined mesh, and by rearrangement of terms obtain expressions for control points of the same surface defined in terms of B_j^1:

$$f = \sum_i P_i \sum_j w_{ij} B_j^1 = \sum_j \left(\sum_i w_{ij} P_i \right) B_j^1 = \sum P_j^1 B_j^1$$

where P_j^1 are the control points on the fine mesh.

To turn this into a practical scheme, however, we need to obtain the weight w_{ij} of the fine-scale basis function B_j^1 in the decomposition of every coarse-scale basis function B_j. Unlike B-splines, for which only few distinct cases need to be considered, the situation is far more complex for T-meshes, even for DAS T-meshes. For a general T-mesh (even analysis suitable), the number of different possible connectivities in the support of a single coarse basis function is infinite. For DAS T-meshes, however, it is finite; the following proposition holds:

Proposition 3. *For a regular DAS T-mesh, the control mesh of a single patch of a T-spline surface corresponding to a face can only have a finite number of possible connectivities.*

This proposition is proved in the 2.10.

Computing subdivision coefficients for regular DAS T-meshes. Proposition 2 asserts that the spaces $\mathcal{T}$ and $\mathcal{T}^1$ are nested, and Proposition 3 suggests that a finite number of subdivision rules can be defined, but neither provide a way to compute coefficients w_{ij} for the subdivision rules.

For non-uniform B-Splines, one way to compute the coefficients w_{ij} is by performing knot insertion: Starting with a single control value 1 assigned to the vertex of the coarse-scale B-spline basis function B_i and zeros assigned to all other vertices, we can insert knot lines of a fine-scale basis function B_j^1, updating the control values accordingly. Once all knot lines are inserted, the control value of v_j, the vertex corresponding to the basis function B_j^1, will be w_{ij}.

For nested analysis-suitable T-meshes, in particular for DAS T-meshes, the subdivision masks can be computed in the same way ([92]. Section 2.4.2).

These observations give us an initial possible approach to defining a subdivision scheme for DAS T-meshes: enumerate all control meshes of a patch by Proposition 3, and then for each pair of a coarse and fine basis functions B_i and B_j^1 overlapping the patch, compute the coefficient w_{ij} using knot insertion.

However while the number of topologically distinct control meshes is finite, it is quite large: a scheme attempting to detect the local connectivity and compute the weights based on this would be quite complex. Even more importantly, it would be entirely unclear how to extend these rules to arbitrary T-meshes, as each local configuration in this case would potentially contain multiple extraordinary vertices.

In the next section we will develop a factorization following the general structure of Catmull-Clark subdivision, that allows for a compact and efficient evaluation and can be easily extended to support extraordinary vertices (Section 2.5).

2.4 Factorized Scheme on DAS T-meshes

Our next goal is to describe a small set of *practical* rules for subdivision of regular DAS T-meshes, that are equivalent to the rules obtained by knot insertion as described in a previous section, and that can be extended to arbitrary meshes.

Our rules use the factorization structure of Catmull-Clark [20] and NURSS [95] subdivision (with vertex and edge control points computed as linear combinations of face points and midpoints). However, unlike these schemes, we need to compute distinct face points for different vertices of a face (*modified* face points), as well as perform an initial step computing control points for *half-faces* of each T-face, which we discuss in more detail below. Although the modified face points are distinct from face points, quite remarkably the weights used in masks are the same as the NURSS weights, up to elimination of some weights or transfer of a weight from one point to another.

We construct our scheme so that it satisfies two main requirements:

- (A) on meshes with no T-joints it reduces to NURSS;
- (B) on regular DAS T-meshes, it reproduces analysis-suitable T-splines in the limit.

2.4.1 NURSS subdivision rules

We first describe the factorized form of the subdivision rules for nonuniform B-splines on regular grids, identical to the one used in [95] for generalization to arbitrary meshes. We refer to these rules as NURSS rules for brevity. The basic formulas for weights will be also used by our scheme.

The computation proceeds in two steps: first, new face points are computed, and intermediate points are assigned to edges (*edge midpoints*). In the second step, face points and midpoints are combined to obtain new positions of vertices, and new points for edges.

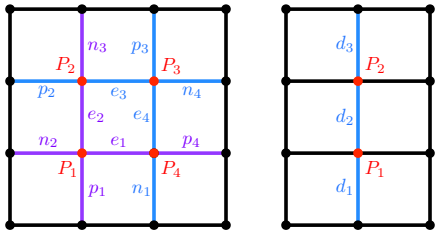


Figure 2.5: *Knot intervals (blue) used to compute weights for face points (left) and midpoints (right) on a regular mesh. Knot intervals for w_1 in purple.*

First stage. At the first stage, face points and edge midpoints are computed. The face points depend only on the four corner control points on the face, but the weights of these corners are determined by the knot intervals adjacent to the vertices of the face.

The edge midpoint is computed as

$$M = \frac{d_2 + 2d_3}{2(d_1 + d_2 + d_3)} P_1 + \frac{2d_1 + d_2}{2(d_1 + d_2 + d_3)} P_2 \tag{2.1}$$

and the face point as

$$F = w_1 P_1 + w_2 P_2 + w_3 P_3 + w_4 P_4 \tag{2.2}$$

The mask weight w_1 is given in terms of the knot intervals on the crosses through

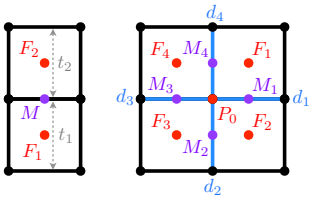


Figure 2.6: *Edge (left) and vertex (right) NURSS masks for a regular mesh; knot intervals in blue (gray for face extents), face points in red, midpoints in purple.*

each vertex (Fig. 2.5):

$$w_1 = \frac{(e_1 + 2p_4)}{2(n_2 + e_1 + p_4)} \frac{(e_2 + 2n_3)}{2(p_1 + e_2 + n_3)} \tag{2.3}$$

and the remaining weights w_2, w_3 and w_4 are obtained in a symmetric way. For regular meshes without T-joints, $p_i = n_i$, but we keep the notation separate as we need distinct values for T-meshes.

Second stage. At the second step, vertex and edge points for the refined mesh are computed. The edge points are computed using the face points F_1, F_2 on two sides of the edge, and the midpoint M:

$$E = \frac{t_2}{2(t_1 + t_2)} F_1 + \frac{t_1}{2(t_1 + t_2)} F_2 + \frac{1}{2} M \tag{2.4}$$

Similarly, we use edge midpoints M_i and face points F_i around the vertex:

$$\begin{aligned} V = & \frac{1}{4} P_0 + \\ & \frac{1}{4 d_{13} d_{24}} (d_2 d_3 F_1 + d_3 d_4 F_2 + d_4 d_1 F_3 + d_1 d_2 F_4) + \\ & \frac{1}{4 d_{13} d_{24}} (d_3 d_{24} M_1 + d_4 d_{31} M_2 + d_1 d_{42} M_3 + d_2 d_{13} M_4) \end{aligned} \tag{2.5}$$

where $d_{ij} = d_i + d_j$.

The full NURSS scheme for arbitrary meshes is discussed in Section 2.5.

2.4.2 Overview of our scheme

We start with an example illustrating the obstacles to applying NURSS rules on a T-mesh, and explain the intuition behind our modifications.

Recall that for DAS T-meshes (and regular grids in particular) subdivision rules can be obtained by repeated knot insertion as described in Section 2.3. For regular meshes or on T-meshes where the local structure is regular, this knot insertion produces the coefficients for non-uniform spline subdivision.

The coarse mesh in the support of a fine-scale basis function B_j^1 is shown in Fig. 2.7a, with a knot grid $[s_1, s_3, s_5] \times [t_1, t_3, t_5]$. Finer knot lines s_2 and s_4 need to be inserted into the support of any coarse-scale basis function B_i to determine the coefficient of control point P_i in the rule for computing P_j^1.

We observe that for the coefficients to be identical to the regular case (because the knot insertion process would be exactly the same) two conditions need to be satisfied: (1) The mesh already contains horizontal and vertical coarse knot lines $[s_1, s_3, s_5]$ and $[t_1, t_3, t_5]$; (2) the coarse T-mesh does not contain the new knot lines $[s_2, s_4], [t_2, t_4]$.

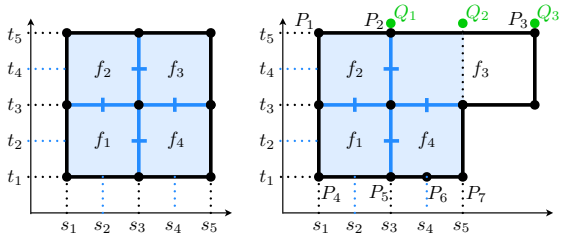


Figure 2.7: *Knot insertion for a vertex basis function (support and knot intervals in blue) on a) a regular mesh, b) a T-mesh*

Unfortunately, in general, both assumptions can be violated for a dyadic analysis-suitable T-mesh. In the T-mesh of Fig. 2.7b, the knot vectors of B_i, $i = 1, 2, 3$, associated with vertices v_i do not contain the knot line s_5. On the other hand, the knot vectors of B_i, for $i = 4 \dots 7$ already contain s_4. As a result, knot insertion will yield a non-standard coefficient w_{ij} for control points P_i, $i = 1 \dots 7$.

We use two different mechanisms to adjust the rules for situations of missing or extra knot lines.

- Accounting for missing knots is relatively easy: we simply perform temporary knot insertion, resulting in new control points Q_i, $i = 1, 2, 3$. Temporary points Q_i are used instead of corresponding control points P_i for the affected control point in the edge and vertex rules.

- To deal with fine-scale basis function knots already present in the mesh (s_4 w.r.t. support of B_i, i=1...7) we modify the regular and T-face masks (in this case for f_1 and f_4). These modifications require no recomputation of weights – rather some weights are zeroed out, or shifted to a different location (Section 2.4.4).

The intuition behind the second mechanism is based on the idea of *blocking* (Figure 2.8): For a coeffcient w_{ij} to be nonzero, the fine-scale basis function B_j^1 needs to be a part of the decomposition of B_i, which is only possible if $\operatorname{supp} B_j^1 \subset \operatorname{supp} B_i$. The presence of a T-joint T on an external edge of the stencil of the edge point of P, ensures that the basis function of the control point Q, does not contain the support of P. As a consequence, although the stencil in the absence of T had Q in it, it cannot be present in the edge stencil. The effects of blocking have to be propagated to the face points used to compute an edge or vertex point for

which a vertex in the standard stencil was blocked. This means that *modified* face points have to be computed, potentially *per corner* of a face. All blocking-related modifications can be summarized as follows:

A T-joint which is not a part of a face stencil, but is adjacent to a vertex v of a stencil, blocks the vertex of the stencil on the other side of v, and shifts its weight to v.

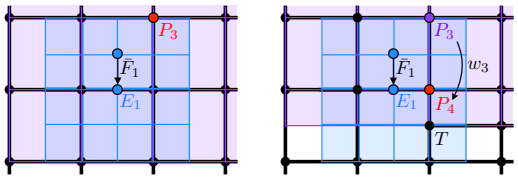


Figure 2.8: *Left: P_3's basis support (purple) overlaps E_1's basis support. Right: T blocks P_3's basis support (purple) with respect to edge point E_1's basis support (blue) – for $\bar{F}_1$ we shift w_3 to P_4.*

We emphasize that our rules were obtained by generalizing from a number of special cases and applying the blocking heuristics, not by a direct derivation from the knot-insertion rules. For this reason, our rules still require a proof of validity for *all* possible stencils which is briefly discussed in Section 2.7 and more completely in supplementary material.

We initially present our subdivision rules in the form closest to NURSS; this is however, not the most efficient way to implement these. In Section 2.6, we describe the changes needed for the *face-centric* view of the rules, with each face responsible for computing its contributions to verices and edges.

2.4.3 T-mesh subdivision masks

We start with face, edge and vertex masks and then describe face point modifications. The stencils of all cases remain the same, with two caveats: (1) T-faces require temporary split into half-faces (2) the edges with knot intervals n_i and p_i "sticking out" of the face (Fig. 2.9), may not be present in the mesh. In this case, consistently with definition of T-splines, we use the knot interval obtained by extending an edge of the face in the same direction to the next intersection with an edge (e.g. p_3 in Fig. 2.9a).

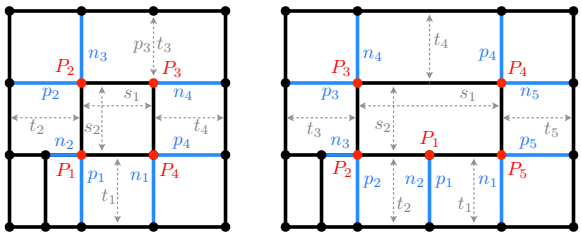


Figure 2.9: *Edge/knot interval labeling on dyadic T-meshes for a) regular faces, b) T-faces.*

Face masks. For regular quad faces, the masks (2.2) remain unchanged. Note however, that in the presence of T-joints in neighboring faces, n_i and p_i can in general be different now (Fig. 2.9a). In a T-face, due to blocking by the T-joint, P_2 and P_5 have zero weights w_2 and w_5. The remaining control points have the

following weights (see notation in Fig. 2.9b):

$$
\begin{aligned}
w_1 &= \frac{s_2 + 2t_4}{2(h_3 + s_2 + t_4)} \\
w_3 &= \frac{(s_1 + 2n_5)}{2(p_3 + s_1 + n_5)} \frac{(2p_2 + s_2)}{2(p_2 + s_2 + n_4)} \\
w_4 &= \frac{(2p_3 + s_1)}{2(p_3 + s_1 + n_5)} \frac{(2n_1 + s_2)}{2(n_1 + s_2 + p_4)}
\end{aligned}
\tag{2.6}
$$

where $h_3 = p_1 = n_2$ (we use a separate notation for h_3 as it will be needed for the extraordinary T-joint case).

The T-face point is given by

$$F = w_1 P_1 + w_3 P_3 + w_4 P_4 \tag{2.7}$$

For both edge and vertex points, we first temporarily split all T-faces with T-joints neighboring the edge or vertex point into half-faces, to which the standard masks can be applied. The difference to meshes with no T-joint is that the masks are applied to modified face control points.

Edge masks. For edge stencils on edge e, we first have to compute the auxiliary control points Q_i described in the introduction of this section. Specifically, if an edge endpoint is a T-joint w.r.t a neighboring T-face, we temporarily split it into two half-faces, introducing a knot opposite the T-joint, and compute 3 control points Q_i opposite the T-edge. The NURSS masks described in Eq.(2.4) are applied to modified face points of incident edges described in Section 2.4.4. The edge midpoint is identical to Equation 2.1.

Vertex masks. For the vertex stencil of a vertex v, we similarly first compute the auxiliary control points Q_i for all T-faces with T-joints adjacent to v, splitting them into half-faces. Note that two such T-faces may be adjacent, in which case five points Q_i are computed as a result of two knot insertions. We use the NURSS masks described in Eq.(2.5), and again apply them to modified face points. The midpoints are defined as in Eq. 2.1, but may use a Q_i instead of a P_i, if it is adjacent to a half-face (Figure 2.18b).

2.4.4 Face point modifications

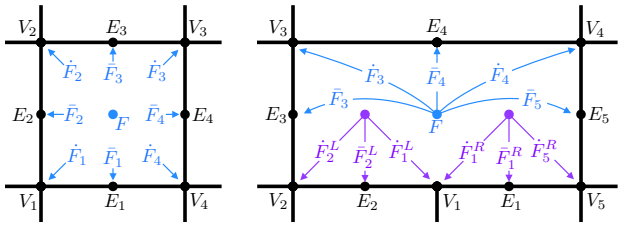


Figure 2.10: *Face (blue) and half-face points (purple) and their edge and vertex modifications.*

At the level of edge and face rules described above, there is no change from the non-T-joint case, except the rules are applied to the *modified* face points computed for each vertex and each edge of a face individually (Fig. 2.10). Furthermore, some face points used in the vertex and edge rules are face points of *half-faces*. These modified face points require minimal recomputation: they use the same simple combinations of NURSS face weights.

Half-face mask for split T-faces. Weights need to be computed for a half-face resulting from temporarily splitting a T-face. With point labeling shown in

(Fig. 2.9b), the weights w_i^R, w_i^L for the right and left half faces are obtained by the following substitutions into Equation 2.3:

$$\begin{aligned} w_i^R &: \{n_2, p_2, s_1, n_4, p_4\} \leftarrow \{h_1, s_1/2, s_1/2, \dot{n}_5, p_5\} \\ w_i^L &: \{n_2, p_2, s_1, n_4, p_4\} \leftarrow \{n_3, \dot{p}_3, s_1/2, s_1/2, h_2\} \end{aligned} \tag{2.8}$$

In the regular mesh case, h_1 and h_2 are equal to $d/2$ (we introduce a separate notation for $h_{1,2}$ to be able to describe the case of extraordinary vertices). $\dot{p}_3$ and $\dot{n}_5$ are derived from p_3 and n_5 but take into account a possible knot insertion on the edge, in which case their knot interval is halved.

Edge-modified face masks $\bar{\mathbf{F}}$. Possible presence of blocking T-joints on the neighboring edges p_i, n_i of a face requires a modification of the face mask before it can be used to compute an edge point. Two representative local configurations for regular and T-faces are shown in Fig. 2.11.

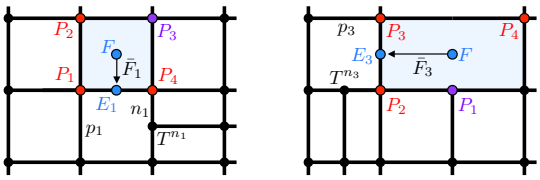


Figure 2.11: *Edge modification for a) regular and b) T-face masks; control points with non-zero weights shown in red, blocked control points with zero weight in purple.*

The regular face mask modified for edge e_1 can be written as

$$\begin{aligned} \bar{F}_1 = (w_1 + T^{p_1} w_2) P_1 \quad + \quad \overline{T^{p_1}} w_2 P_2 \quad + \\ \overline{T^{n_1}} w_3 P_3 \quad + \quad (T^{n_1} w_3 + w_4) P_4 \end{aligned} \tag{2.9}$$

where $T^e = 1$ if there is a T-joint on edge e (0 otherwise) and $\overline{T^e} = 1 - T^e$. The modifications for the remaining face edges are cyclically symmetric.

For T-faces, there are three different cases of face mask modifications. For edge e_4 opposite the T-joint, no modification is necessary since analysis-suitable condition ensures that no T-joint can be on n_4 or p_4. For edges e_1 and e_2 on the T-edge, the modifications are the same as for regular faces, applied to the half-face weights (2.8). For the side edges e_3 and e_5 we modify the T-face weights of Eq. 2.6 (Fig. 2.11b):

$$\begin{aligned} \bar{F}_3 = \overline{T^{n_3}} W_1 P_1 \quad &+ \quad T^{n_3} W_1 P_2 \quad + \\ (W_3 + T^{p_3} W_4) P_3 \quad &+ \quad \overline{T^{p_3} W_4} P_4 \end{aligned} \tag{2.10}$$

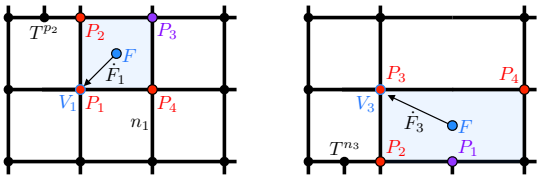


Figure 2.12: *Examples of face mask modifications for vertex stencils: non-blocked control points are shown in red, blocked control points with zero weight in purple.*

Vertex-modified face masks $\dot{\mathbf{F}}$. Similarly to the edge mask case, we modify a face mask in the presence of T-joints on the outer edges of a face 1-neighborhood which lead to blocking of some of the vertices:

$$\begin{aligned} \dot{F}_1 = w_1 P_1 \quad &+ \quad (w_2 + T^{p_2} w_3) P_2 \quad + \\ \overline{T^{p_2}}\,\overline{T^{n_1}} w_3 P_3 \quad &+ \quad (T^{n_1} w_3 + w_4) P_4 \end{aligned} \tag{2.11}$$

The modifications for the remaining face vertices are again cyclically symmetric.

For T-faces we again have three different cases: 1) For vertices v_2 and v_5 we use the regular vertex modifications in Eq. 2.11 for the respective adjacent half-faces. 2) For the T-joint v_1 we similarly compute the regular vertex modification for both half-faces. 3) For v_3, we have the following modified expression: if there is a T-joint on n_3, P_1 no longer contributes to V_3 as their respective basis functions no longer overlap. The weight w_1 is shifted to P_2 (Fig. 2.12b). (the case for v_4 is symmetric). The modification becomes:

$$\dot{F}_3 = \overline{T^{n_3}} w_1 P_1 + T^{n_3} w_1 P_2 + w_3 P_3 + w_4 P_4 \tag{2.12}$$

2.4.5 Boundaries

Our factorized scheme supports boundaries similar to Catmull-Clark. Conceptually we mirror the mesh for each border face along the border edge, thereby definining knot intervals, control points and T-joint tags past the border. In practice this is equivalent to evaluating non-uniform B-Spline boundary curves along the border. Note that for the auxiliary control points Q_i knot insertion on "both sides" of the border results in no change for the border vertex, i.e. (with the notation of Figures 2.7, 2.9) if e_3 is a border edge of a T-face, $Q_1 = P_3$.

2.5 Extraordinary Vertices

The support of meshes with arbitrary topology is one of the crucial advantages of subdivision surfaces: the local, factorized operations make generalization to

general meshes possible, while retaining the surface quality of splines for regular parts of the mesh. We complete our scheme by describing the generalization to arbitrary mesh topologies. To generalize the factorization to extraordinary vertices, we again take inspiration from NURSS, but show that it develops tangent plane discontinuities in certain situations. We then present our modifications that both restore tangent plane continuity in these cases and extend naturally to T-joints.

Thanks to the factorization, we do not need to impose restrictions on the proximity of T-joints and extraordinary vertices This means that a T-joint can be placed on an edge between two extraordinary vertices, and in fact can be extraordinary itself (Fig. 2.17a).

2.5.1 NURSS

We briefly describe the NURSS scheme in a simplified form for meshes with only quad faces and identical knot intervals for opposite quad edges, as we do not consider other types of meshes.

Beside the knot intervals on n_i and p_i we also need intervals on the next and previous edges (counterclockwise), respectively, labeled q_i and m_i (Fig. 2.13). Observe that if e.g. v_1 is a valence 4 vertex, $q_1 = n_2$ and $m_1 = p_4$. The (unnormalized) NURSS weights are:

$$w_1 = (e_1 + m_1 + p_4 + e_3 + q_3 + n_4) \cdot (e_2 + q_2 + n_3 + e_4 + m_4 + p_3)$$

$w_2 \dots w_4$ are cyclically symmetric. A face point F is then defined as:

$$F = \frac{\sum_{i=0}^{4} w_i P_i}{\sum_{i=0}^{4} w_i} \tag{2.13}$$

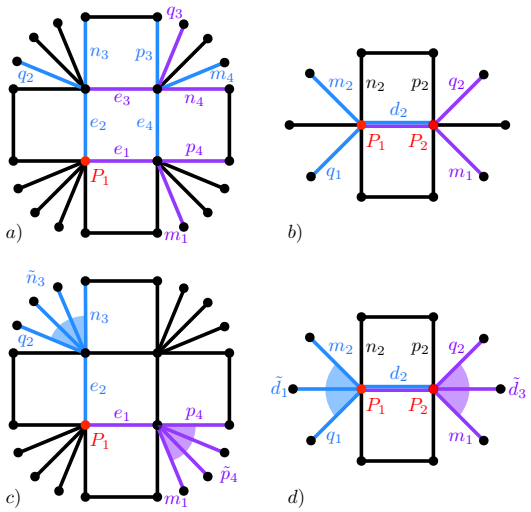


Figure 2.13: *a) & c) Knot intervals used in face point weight w_1's two factors (blue, purple) for original NURSS and our max-modified face masks. b) & d) Knot intervals used in midpoint's two control point weights (purple, blue) for P_1 and P_2.*

With knot interval labeling as in Fig. 2.13b), the edge midpoints are computed as

$$M = \frac{(d_2 + q_1 + m_2)P_2 + (d_2 + q_2 + m_1)P_1}{(d_2 + q_1 + m_2) + (d_2 + q_2 + m_1)}$$

The masks for edge points stay the same as in Eq. (2.4), but use the updated face and midpoint masks.

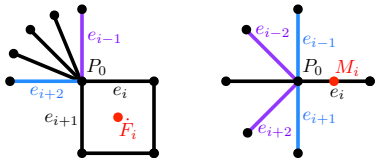


Figure 2.14: *Two factors (blue, purple) of face (f_i) and midpoint weights (m_i) for NURSS vertex mask.*

Vertex points are built from face points and edge midpoints of all surrounding faces:

$$V = \frac{n-3}{n} P_0 + \frac{3\sum_{i=1}^{n}(m_i M_i + f_i F_i)}{n\sum_{i=1}^{n}(m_i + f_i)} \tag{2.14}$$

with n the valence at the vertex and the midpoint weights m_i and face weights f_i as follows (using the notation in Fig. 2.14)

$$m_i = (e_{i+1} + e_{i-1})(e_{i+2} + e_{i-2})/2 \tag{2.15}$$

$$f_i = e_{i-1} e_{i+2} \tag{2.16}$$

2.5.2 Loss of tangent-plane continuity

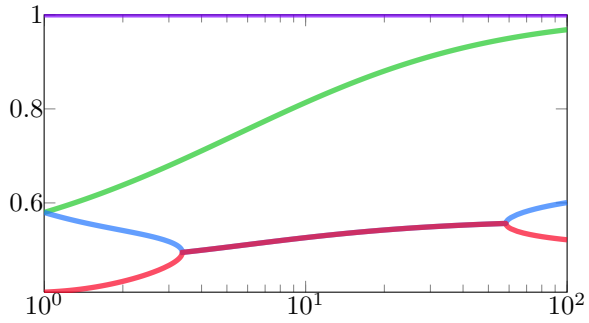


Figure 2.15: *Modulus of four dominant eigenvectors of NURSS subdivision matrix for valence $n = 6$ and unit knot intervals except one varying.*

In [95], the authors describe a pinching artifact at extraordinary vertices for general knot interval assignments due to different magnitudes of first and second eigenvalues of the subdivision matrix, but conjecture that the construction above nevertheless achieves tangent plane continuity. Unfortunately, we observe that in general, tangent plane continuity is not achieved. For analysis, we choose valence

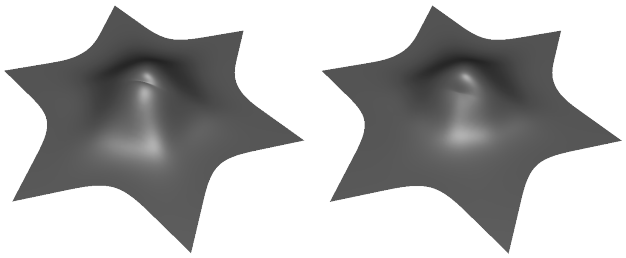


Figure 2.16: *Extr. vertex with $n = 6$. a) NURSS, b) our MAX modification.*

$n = 6$ with unit knot intervals except at one edge and vary the remaining interval. Figure 2.15 shows the modulus of the first four eigenvalues of the subdivision matrix. For knot intervals in the range 4-16, we can observe that the first subdominant eigenvalue stays real, while the second and third eigenvalues become complex. This results in a surface that does not have a unique tangent plane at the extraordinary vertex. Figure 2.16a clearly shows that there are two separate tangent planes at the extaordinary vertex. In the next section we describe a modification of the NURSS extraordinary vertex rules that 1) restores tangent plane continuity in the above situation (Figure 2.16b) and 2) supports DAS T-meshes.

2.5.3 Maximum formulas for extraordinary vertices

The face and midpoint generalizations in Section 2.5.1 can be viewed as replacing one knot interval with an average over two knot intervals in the vicinity of extraordinary vertices. The main idea of our maximum modification is to replace the averaging over two knot intervals with taking the maximum knot interval in the whole sector spanned by the two respective edges. We can view this also as

modifying p_i, n_i into $\tilde{p}_i, \tilde{n}_i$ as follows:

$$\begin{aligned} \tilde{p}_i &= \max\measuredangle[m_{i+1}, p_i] \\ \tilde{n}_i &= \max\measuredangle[q_{i-1}, n_i] \end{aligned} \tag{2.17}$$

where $\max\measuredangle[e_i, e_j]$ refers to the maximum over the knot intervals of all edges in the sector e_i to e_j, inclusive (Fig. 2.13c). We then substitute the p_i and n_i in Eq. (2.3) with $\tilde{p}_i$ and $\tilde{n}_i$ and re-normalize by the sum of all weights to get affine invariance.

$$F = \frac{\sum_{i=0}^{4} w_i P_i}{\sum_{i=0}^{4} w_i} \tag{2.18}$$

Note that we no longer compute the weight over both parallel edge lines. We similarly update the midpoint masks (Fig. 2.13d):

$$\begin{aligned} \tilde{d_1} &= \max\measuredangle[q_1, m_2] \\ \tilde{d_3} &= \max\measuredangle[q_2, m_1] \\ M &= \frac{d_2 + 2\tilde{d_3}}{2(\tilde{d_1} + d_2 + \tilde{d_3})} P_1 + \frac{2\tilde{d_1} + d_2}{2(\tilde{d_1} + d_2 + \tilde{d_3})} P_2 \end{aligned} \tag{2.19}$$

We keep the edge and vertex masks exactly as described above.

Extraordinary T-joints. Our algorithm can support T-joints with valences $>$ 4. Such configurations turn into T-faces with regular T-joints after one subdivision step (Fig. 2.17a). To support such T-meshes, we substitute h_1, h_2 in Eq. (2.8) and

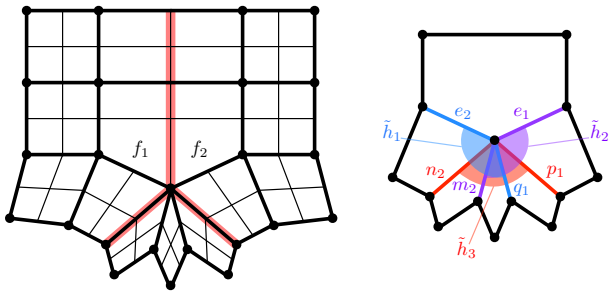


Figure 2.17: *a) Extraordinary T-joint with face/edge extensions and subdivided T-mesh; b) sectors for $\tilde{h}_1$(blue), $\tilde{h}_2$(purple), and $\tilde{h}_3$(red).*

h_3 in Eq. (2.6) by the following sector maxima, respectively (Fig. 2.17b):

$$\begin{aligned}\tilde{h}_1 &= \max\measuredangle[e_2, q_1] \\ \tilde{h}_2 &= \max\measuredangle[m_2, e_1] \\ \tilde{h}_3 &= \max\measuredangle[n_2, p_1]\end{aligned} \tag{2.20}$$

2.6 Algorithm

While we stayed close to the NURSS point of view for the presentation of our edge and vertex masks, a face-centric implementation is both more compact and faster. Section 2.4.4 already presented the face point modifications in a face-centric way. Here we outline an entirely face-centric algorithm (see Algorithm 1 for the pseudocode): the outer loop iterates once over all faces and computes all neccessary components of face, edge and vertex points. An additional renormalization of all vertex and edge points completes the computations. We provide a MATLAB implementation of mask computation in the supplement.

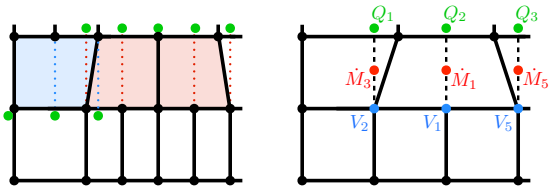


Figure 2.18: *Left: T-face chains. Right: midpoint masks involving auxiliary points Q_i.*

Auxiliary control points Q_i and T-face chains. In Section 2.4.3 we defined auxiliary points Q_i that are used to make the local configuration around a vertex or edge regular. These points Q_i were defined for each mask separately, and were always as a result of (one or two) knot insertions. We can collapse the computation of Q_i to once per *T-face chain*: We call a T-face chain a maximal set of neighboring T-faces, such that their T-joint is not part of the common edge of two neighboring T-faces and all T-edges on the same side. As an example, in Figure 2.18a) the left T-face builds a one-element T-face chain, while the middle and right T-faces build a two-element T-face chain. Similarly, we call the set of edges in a T-face chain opposite each T-edge a *T-face edge chain*. We can avoid the need to compute Q_i for different masks separately if we introduce all knots along a T-face edge chain simultaneously. This can be done efficiently using Lane-Riesenfeld. We do this in a pre-processing step at the beginning of the algorithm, and can then access the three relevant Q_i for each T-face without recomputation.

Face-centric midpoint evaluation. Since edge midpoints are shared between faces, we have to decide which face computes the midpoint contribution to an edge or vertex mask. We decide this "edge ownership" based on which face (or half-edge) index is smaller. A subtle complication is that the edge midpoints $\dot{M}_3, \dot{M}_5$

of e_3 and e_5 used for the vertex masks of v_2 and v_5 of a T-face require the control points Q_1, Q_3 instead of P_3, P_4. We make sure these are always computed in the respective T-face (again using edge ownership if two T-faces in the same T-face chain share the edge). Lastly, for each T-face we compute the split edge midpoint $\dot{M}_1$ which is used for the vertex point V_1 (Fig. 2.18):

$$\dot{M}_1 = \frac{s_2 + 2t_4}{2(\tilde{h}_3 + s_2 + t_4)} P_1 + \frac{2\tilde{h}_3 + s_2}{2(\tilde{h}_3 + s_2 + t_4)} Q_2 \tag{2.21}$$

2.7 Evaluation

Verifying limit surface in the regular case. We have verified that our subdivision rules are correct refinement rules for T-splines. For this it was sufficient to show that for a patch $P(u, v)$ corresponding to a face of a T-mesh, and an arbitrary valid connectivity of the control mesh around it, refinement using our rules keeps the surface unchanged, i.e. refining the control mesh once, to obtain a new T-mesh, and then constructing a T-spline surface from it, produces the same result as directly constructing the T-spline from the original mesh. As the number of possible connectivities of the control mesh of a patch on a regular grid is finite, we simply verified this property for all possibilities.

Surface quality. In Figure 2.19, we show a valence 5 extraordinary vertex with several possible local T-joint configurations, and compare to Catmull-Clark and NURSS with unequal knot intervals. We observe that the behavior of surfaces with T-joints is similar to that of Catmull-Clark, unless the knot spacing is unequal, in two directions. Then it is slightly worse and comparable to NURSS, inherit-

Algorithm 1 T-mesh subdivision pseudo-code

for all T-face chains **do**
  compute auxiliary points Q_i
end for
for all regular faces $f = [v_1, \ldots, v_4]$ **do**
  Compute face point $\mathbf{F}$, eq. (2.2)
  for all $i \in [1 \ldots 4]$ **do**
    compute
      edge-mod. $\bar{F}_i$ for edge e_i, eq. (2.9)
      vertex-mod. $\dot{F}_i$ for vertex v_i, eq. (2.11)
      midpoints M_i if edge owner, eq. (2.1)
      terms to add V_i and E_i, eqs. (2.4), (2.14)
    add the weight to total weight of V_i and E_i
  end for
end for
for all T-faces $f = [v_1, \ldots, v_5]$ **do**
  compute:
    face points $\mathbf{F}$, eq. (2.7)
    edge-modofied $\bar{F}_3, \bar{F}_5$, eq. (2.10)
    vertex-modified $\dot{F}_3, \dot{F}_4$, eq. (2.12)
    vertex-modified (half-faces) $\dot{F}_1^R, \dot{F}_1^L$ and $\dot{F}_2^R, \dot{F}_5^L$, eq. (2.11)
    edge-modified $\bar{F}_1^R, \bar{F}_2^L$, eq. (2.9)
    midpoints M_i if edge owner, eq. (2.1)
    midpoints $\dot{M}_{3,1,5}$ using $Q_{1,2,3}$, eqs. (2.1), (2.21)
    terms to add to V_i and E_i, eqs. (2.4), (2.14)
  add the weight to total weight of V_i and E_i
end for
for all edge points E_i and vertex points V_i **do**
  normalize by vertex's total weight
end for

ing its pinching artifact, a result of two unequal subdominant eigenvalues in the subdivision matrix. This can be seen in the lower quality of the reflection lines.

In Figure 2.20, we look at the effect of increasing the vertex valence. We observe that the quality is consistent with the quality of Catmull-Clark. We note that the quality for standard Catmull-Clark decreases quickly with valence but a variety of techniques were developed to improve quality, some of which are applicable in our

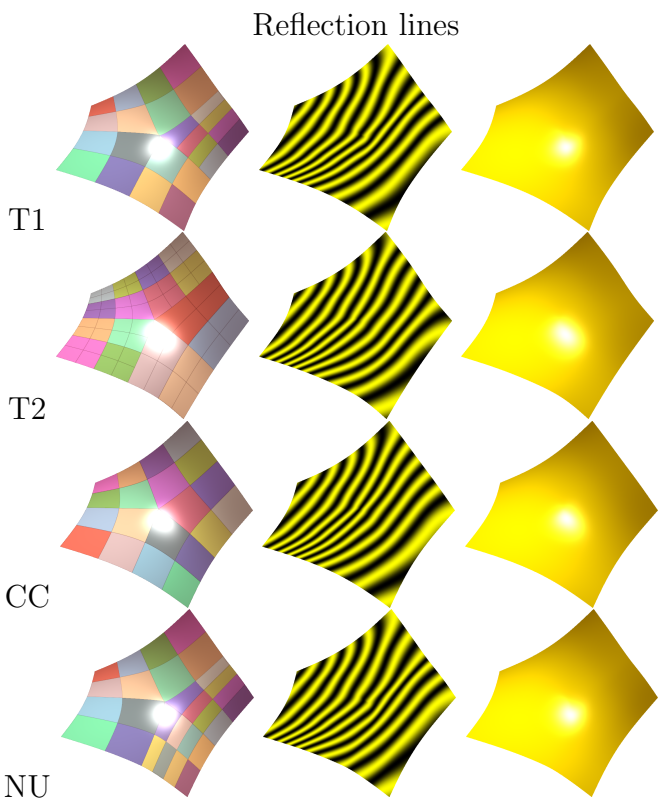


Figure 2.19: *A number of T-joint configurations at a vertex of valence 5: T1, T2 are different T-joint configurations. CC is the standard Catmull-Clark surface, and NU is NURSS.*

setting (as the scheme is stationary), although with greater difficulty.

Figure 2.1 shows how T-joints can be used to avoid under- or overtessellation without the introduction of extraordinary vertices, in the case where the control mesh faces vary greatly in size.

In Figure 2.21, we compare how similar mesh layouts are done with T-joints and conforming meshes with extraordinary vertices. Typically pairs of vertices of valence 3 and 5 need to be used to achieve the same layout.

Figure 2.22 shows several extraordinary vertices of valences 5 and 6 neighboring T-joints, transitioning to a coarser mesh from the fingers to the hand.

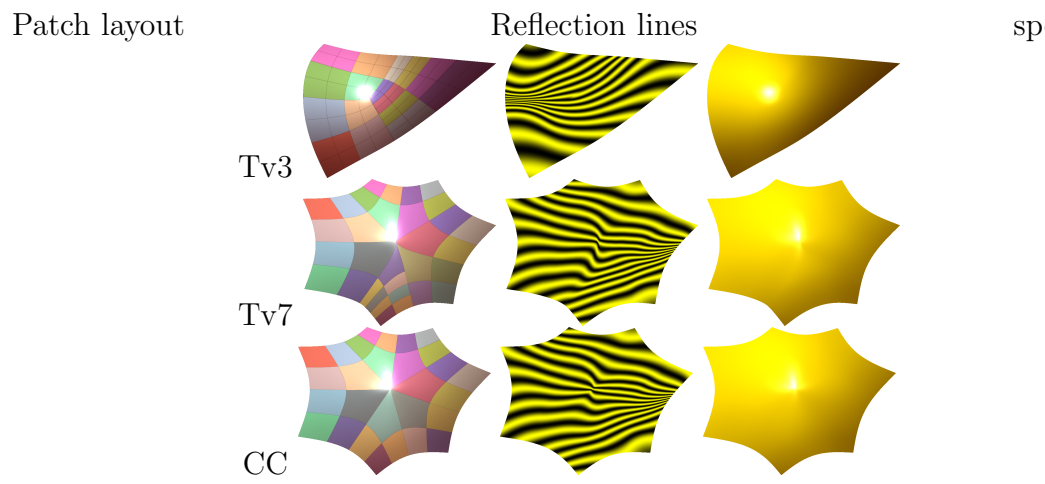


Figure 2.20: *Varying the valence: vertices of valence 3 and 7 are shown, with T-joints in incident edges. The last row shows Catmull-Clark for valence 7 for reference.*

Figure 2.23 highlights all T-joints used in Figure 1 to avoid the extraordinary vertices found in the original mesh from [7] to coarsen the mesh near the chin/neck and ear/cheek areas. It also shows an extraordinary T-joint.

2.7.1 Characteristic Maps and Tangent Plane Analysis

A complete analysis of tangent plane continuity at extraordinary vertices is relatively complex, due to the large number of configurations that need to be considered. However, with one additional assumption, a finite enumeration for moderate vertex valences is possible.

Recall that for a mesh a number of knot values can be chosen independently, with the rest determined by compatibility conditions. Specifically, if we assume that all independent knot intervals are set to 1, then a finite (although very large) enumeration of cases of self-reproducing connectivities near extraordinary vertex is possible.

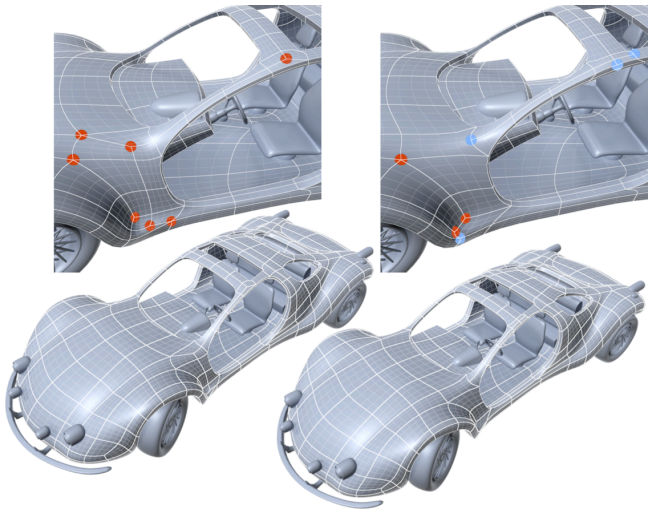


Figure 2.21: *Comparison of similar shapes with a mesh obtained using extraordinary vertices and a T-mesh.*

To determine the limit behavior at an extraordinary vertex, we can assume that enough subdivision steps have occured that the topology around the vertex is *self-similar*, i.e. the control mesh of the set of patches around the extraordinary vertex is the same at all subdivision levels. To characterize these topologies, we define a *spoke* to be the edge of the unsubdivided mesh(and all T-joints are regular) incident at the extraordinary vertex of interest. Then self-similar configurations are characterized by the following condintions:

- there is a single extraordinary vertex in the control mesh, and it is not a T-joint;
- knot intervals on spokes are equal;
- there are only T-joints along spokes. A row of faces along a spoke either all have T-joints on the spoke or none of them does.

These conditions allows us to characterize a configuration by a small number

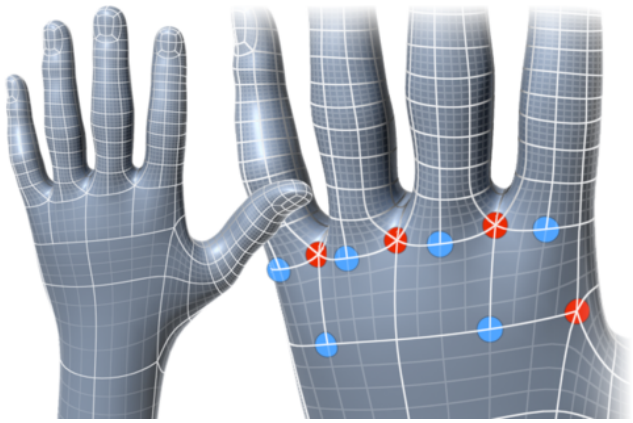


Figure 2.22: *Examples of subdivided T-meshes.*

of parameters (Fig. 2.24): 1) the valence, 2) the knot interval of the form $1/2^i$ for each spoke (all other knot intervals are determined by compatibilty constraints, and the intervals on adjacent spokes cannot differ by more than a factor of two) 3) whether there are T-joints on a spoke and to which side it stem is pointing.

The two-ring control mesh for the central ring of patches is obtained by taking the vertices of the union of quads forming 2×2 grids in each of k sectors for a vertex of valence k. We note that scaling all knots by the same amount does not change local surface behavior, so one of the knot intervals in the self-similar control mesh can be always chosen to be 1, and the rest set with respect to it.

We enumerate possible configurations by going over all combinations of parameter values and checking the analysis-suitable conditions. Of course the number of configurations grows exponentially, so the method is practical only for sufficiently low valences (up to $n = 9$).

We use the standard approach to verifying C^1 continuity for spline-based schemes [88, 80].

First, we construct the subdivision matrix mapping the control points of the

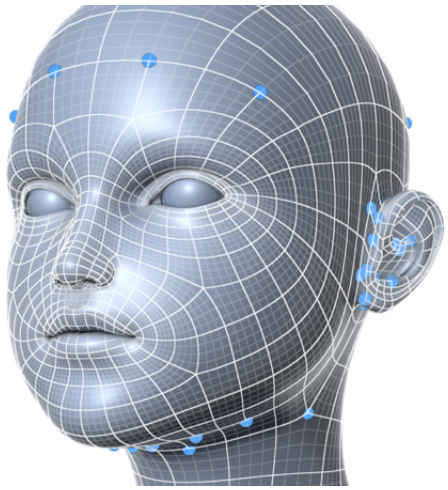


Figure 2.23: *Bottom right T-joint is an extraordinary T-joint.*

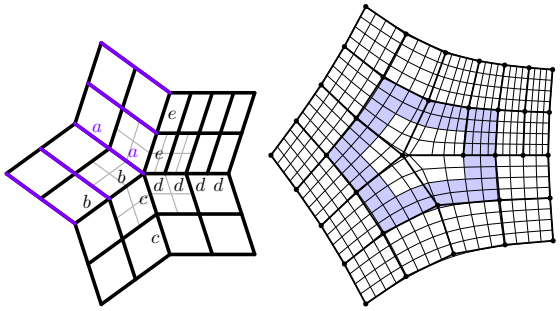


Figure 2.24: *Left: limit topology around an extraordinary vertex. Right: ring of Bezier patches extracted from the limit stencil.*

two-ring to the points of the two-ring on the next refinement level, and compute its subdominant eigenvalues and eigenvectors x^ℓ, $\ell = 1, 2$ with components x_i^ℓ. The two-dimensional control mesh with control points (x_i^1, x_i^2) define the control mesh for the *characteristic map* from the plane to the plane. Nonvanishing Jacobians and bijectivity of the characterisitic map are sufficient for C^1-continuity. The characteristic map is also self-similar (i.e., its values on a nested sequence of ring domains are obtained by scaling), so it is sufficient to examine it on a single ring domain. The ring domain is obtained as a set of patches forming outer rings after

two subdivision steps (Fig. 2.24). As there are no extraordinary vertices in the control meshes of these patches, all subdivision rules affecting the limit surface on these patches are just analysis-suitable T-spline rules, and patches are polynomial.

For each patch, nonegativity of the Jacobian can be verified explicitly, by computing the Jacobian as a polynomial and converting it to the Bezier form. Positivity of Bezier coefficients of the Jacobian is sufficient. Finally, global bijectivity can be inferred from local bijectivity by simple winding number tests as shown in [115].

2.8 Conclusions

We have demonstrated that for a restricted class of T-meshes it is possible to design a set of subdivision rules with a similar support and computation cost to the Catmull-Clark subdivision, and the complexity of stencils is only moderately higher compared to NURSS. The quality of surfaces is similar to Catmull-Clark near extraordinary vertices, although it degrades if knot intervals near extraordinary vertices vary greatly. Both a version of NURCC and analysis-suitable T-splines can be reproduced by our scheme.

Limitations. The most significant limitation of the proposed approach is that the T-mesh is required to be analysis-suitable. Requiring separation between T-joints limits the flexibility of T-joint insertion. On the other hand, this class of T-splines is best understood, and has a number of attractive properties not available for general T-splines.

Just as it is the case with T-NURCC, one can combine our scheme with general T-spline patching, provided that the separation between *non-standard* regions on the regular part of the surface and extraordinary vertices is high; note that no such

requirement needs to be imposed on the standard regions.

Our analysis of C^1-continuity shows that while our modification of NURSS increases the range of valences for which the scheme is C^1, the analysis is performed only under assumptions on independent knot intervals and for bounded valence. Even more significantly, for some of the higher-valence configurations C^1 conditions may still fail. Although practical implications of this are not high, this, along with degradation of surface quality suggests that more work is needed on improving NURSS rules.

A recent approach developed in [84] suggests that analysis of factorized schemes may be done without explicit analysis of the characteristic map; this opens up the possibility of analysis with restrictions on independent knot intervals.

Future work . In the future we want to extend the regular boundary rules of Section 2.4.5 to extraordinary vertices and define a full set of crease masks. We will also describe an extension of our scheme to support certain semi-standard T-spline configurations that will allow T-meshes such as a regular mesh with one quad split into four.

2.9 Subdivision and Nested Spaces

Proof of Proposition 2. We show that *dyadic analysis-suitable T-splines* form nested spaces under face quadrisection.

The extended T-mesh T_{ext} is defined as the T-mesh with all extensions included. We use *Corollary 8.10* from [61]:

Corollary 1. *Given two analysis-suitable T-meshes, T^1 and T^2, if $T^1_{ext} \subseteq T^2_{ext}$,*

then $\mathcal{T}^1 \subseteq \mathcal{T}^2$

We prove that the extended T-mesh and the extensions of its subdivisions are nested. Let the T-mesh be T^0 and subdivided T-mesh be T^1. Consider a face of T^0 containing a first-bay T-joint face extension (Figure 2.25a, face f1, middle extension). In this case, T^1 contains two edges covering the extension in f_1. Suppose a face of T^0 contains a second-bay face extension (e.g., side extensions in f_1 in Figure 2.25a or central extension in f_0). By enumerating possible ways to connect pairs of faces for which the same extension is first- and second-bay, one can observe that there are only two valid configurations for such pairs of faces in a DAS T-mesh, identical exactly to these examples: (f_1, f_0) or (f_2, f_1). In both cases, one can verify directly that the extended mesh of T^1 covers the extensions in f_0 and f_1.

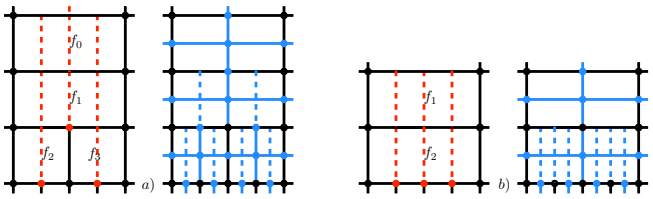


Figure 2.25: *a) Extensions in the original mesh (red dashed lines) are present in the extended subdivided mesh (blue) for DAS T-meshes. b) This is not the case for a non-nested space with three T-joints per edge in the original T-mesh (see extensions in* f_1*).*

Therefore, the DAS T-spline spaces are nested. As an example of spaces that are not nested, imagine an analysis-suitable T-mesh that allows three T-joints per edge (Figure 2.25b). The extended original T-mesh is not contained in the extended quadrisected T-mesh.

2.10 Stencil Enumeration

For the original T-mesh T^0 we define the T-spline space $\mathcal{T}^0$, its basis functions B_i^0 and control points P_i^0 associated with vertex v_i and support $S_i^0 = \text{supp}(B_i^0)$. Analogously we define for the refined T-mesh T^1, $\mathcal{T}^1$, B_j^1 and P_j^1 associated with vertex v_j, $S_j^1 = \text{supp}(B_j^1)$.

We enumerate a set of neighborhoods in T^0 of a vertex v_j from T^1, consisting of vertices in the stencil of v_j (*stencil candidates.*) In the next section we will show that the control points P_i^0 in these neighborhoods are in fact the only ones needed to compute P_j^1. We distinguish 5 cases with different neighborhood topology:

1. P_j^1 is a face control point;
2. P_j^1 is an edge control point and both edge vertices are regular;
3. P_j^1 is an edge control point and one edge vertex is a T-joint with respect to one or both faces bordering the edge;
4. P_j^1 is a vertex control point, v_j is a regular vertex;
5. P_j^1 is a vertex control point, v_j is a T-joint in T^0.

These 5 cases require different constructions of neighborhoods.

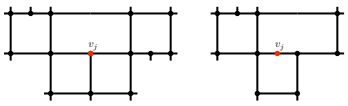


Figure 2.26: *Examples of a) T-vertex and b) T-edge stencils.*

Cases 1,2 and 4 are similar to Catmull-Clark - we define the stencils as the vertices of all faces bordering v_j. For an edge control point these are the faces

bordering the edge, while for a face control point it is the surrounding face. For cases 3 and 5, 1-neighborhood is not big enough: there are stencil control points outside the 1-neighborhood. For 5, we add the faces neighboring the T-face to the left and right of the T-joint (Figure 2.26a). Similarly for case 3, we only add the face on the same side of the T-joint as the edge on which v_j^1 is (Figure 2.26b).

For enumeration of all connectivities we can obtain as 1-neigborhoods (cases 1,2,4) or 1-neighborhod with additional edges specific choice of knot intervals is irrelevant. For consistency, we fix the basis cross of B_j^1 to unit intervals. Then any knot interval incident at v_j^1 (associated with an edge or a face extent) and covered by the knot grid of B_j^1 is also fixed. This leads to fixed knot intervals marked blue in Figure 2.27. It is easy to see that all T-joints in a stencil topology have to be oriented either all horizontally or all vertically. W.L.O.G., we assume the orientation is vertical. For cases 3 and 5 the orientation is fixed by the central T-joint, while for cases 1 and 4 we can choose one orientation (the other is symmetric). We assume all T-joints are oriented vertically. Only for case 2 do we have to consider both orientations. Each face in the stencil candidate can have at most one T-joint, which can be located on one of the two horizontal edges (in case 2, it can also be on one of the vertical edges) giving 3 possible states for each face (4 for case 2). In cases 1,3,4,5 we iterate for every face through the cases of a T-joint along the bottom and the top edge, observing all edge length constraints. For case 2, we consider T-joints on every edge (removing combinations that would result in horizontal and vertical T-joints). This yields a total of 171 stencil candidates. In the supplement, we explain how we verify that this enumeration is exhaustive.

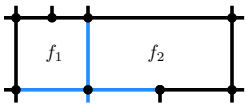


Figure 2.27: *Edge constraints on opposite sides resulting from the normalized basis cross.*

2.11 Proof of Exhaustive Enumeration of Subdivision Stencils

To compute the value of P_j^1 we need to define a stencil of control points P_i^0 that influence its value. We exhaustively enumerate the 1-ring neighborhood configurations (with a suitable extension at T-vertices and T-edges) of a vertex in all possible T-mesh configurations.

Here we show that such a configuration $\mathcal{N}$ in fact covers the stencil, i.e. no P_i^0 outside $\mathcal{N}$ can influence P_j^1.

We split the parametric plane outside $\mathcal{N}$ into two parts: (1) half-slabs H_k in which a basis function B_i^0 cannot affect P_j^1 since its cross would have to intersect two edges in $\mathcal{N}$ before S_i^0 would cover S_j^1, and (2) the remaining regions (*corner zones*) C_k for which we use a Lemma from [28].

Let

$$S_j^1 = [s_0 \dots s_1] \times [t_0 \dots t_1].$$

Half-slabs H_k are constructed for each of the four directions t,$-t$, s and $-s$. Consider a line ℓ_s in one of these directions (e.g., horizontal direction $-s$) that intersects the stencil candidate $\mathcal{N}$. Each such line is intersected by at least two edges of the stencil (this can be verified for each stencil candidate directly). Consider the set of points on ℓ_s separated from the *right* boundary of S_j^1 by two stencil

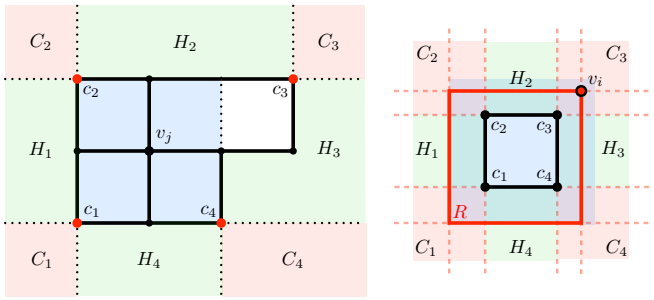


Figure 2.28: *Half-slab H_k (light green) and corner zones C_k (light red), corner points c_k (red), basis support S_j^1 of vertex v_j (blue). Right: minimal rectangle R that a quadrant of v_i's basis function needs to contain.*

edge intersections with ℓ_s. These points cannot be control points in the stencil. The union of such points for all lines ℓ_s form the half-slab H_1.

For every $\mathcal{N}$ there are four *corner points* where two H_k intersect. They bound one of the open regions C_k which we will call *corner zone*. Since the entire plane is covered by these regions:

$$\mathbb{R}^2 = \mathcal{N} \cup \bigcup_{k=1}^{t} H_k \cup C_k,$$

all that is left to do is to prove that any B_i^0 centered in any corner zone C_k can not contain S_j^1. To show this, we use *Lemma 4.2* from [28]. The vertex v_i is called an active T-mesh node if it is sufficiently far away from any boundary that there are enough knots to define its basis function. $\mathrm{TF}(v_i)$ is the *tiled floor* of v_i, i.e. the support of B_i^0 excluding the 5x5 grid of knot lines.

Lemma 4. *Let $\mathcal{M}$ be an AS T-mesh and v_i an active T-mesh node. Then* $\mathrm{TF}(v_i)$ *does not contain any T-mesh node.*

As a consequence of this Lemma, in the case of dyadic T-meshes, all T-mesh vertices v_i in a quadrant Q of a basis function B_i^0, have to be a subset of one of

two possible configurations (see Figure 2.29).

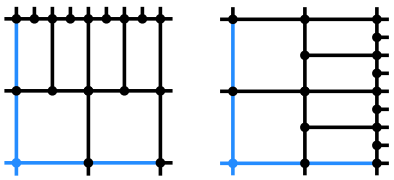


Figure 2.29: *Possible quadrants for DAS T-spline basis functions: visible part of the basis cross marked blue.*

This can be seen as follows: There can be no vertices or crossing edges on the basis cross other than the two that define the basis cross (otherwise the basis cross would be shorter), so the densest regular grid we can define on Q has 3×3 knots. W.l.o.g. let us assume there are only vertical T-edges in this grid. It contains at least one T-joint with a horizontal stem – otherwise the support of the basis function B_i^0 would be smaller. Wherever the T-joint is located, its face and edge extension together span the entire s-span of Q, making it impossible to add a vertical T-joint anywhere without violating the analysis-suitable rule that no horizontal and vertical T-joint extensions intersect.

So in one quadrant, analysis-suitable T-meshes can only have either horizontal or vertical T-joints, but not both. Since we are only considering dyadic T-meshes, there can be at most one T-joint per edge. Hence the densest Q given a fixed basis cross is defined by the cascaded T-joint pattern shown in Figure 2.29. It is clear that the s and t knots of any other refinement are contained in the knots of this Q.

We can hence conclude that there is one dimension along which there can be no more than 3 knots in Q.

To prove that no $v_i \in C_k$ exists such that $S_i^0 \supseteq S_j^1$, we show that the relevant

quadrant Q of a basis function B_i^0 associated with such a v_i requires at least 4 knots along both dimensions.

A necessary condition for $S_i^0 \supseteq S_j^1$ is that Q contains the rectangle R spanned by v_i and the corner of S_j^1 diagonally opposite to v_i (Figure 2.28 right).

We then collect the knots of all vertices in R, and include the s- and t-extents of R to ensure that Q indeed contains R. Recall that the total knot count cannot exceed three in both dimensions simultaneously.

For each previously listed stencil, however, we verified by this simple counting scheme that for each stencil C_k, the number of knots in R is always ≥ 4 in each dimension. Hence, there are no $v_i \in C_k$ such that their support contains S_j^1, and there are no outside control points P_i^0 that affect P_j^1.

To verify our factorization reduces to analysis-suitable T-Splines, now all we have to do is to verify that it yields the same results we obtain with the refinement formulas in Section 2.3 for every vertex of every stencil connectivity enumerated above.

2.12 Explicit Enumeration of Dyadic T-mesh Subdivision Stencils

Figure 2.30: *DAS T-mesh face stencils.*

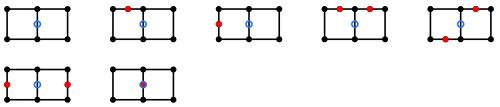


Figure 2.31: *DAS T-mesh edge stencils.*

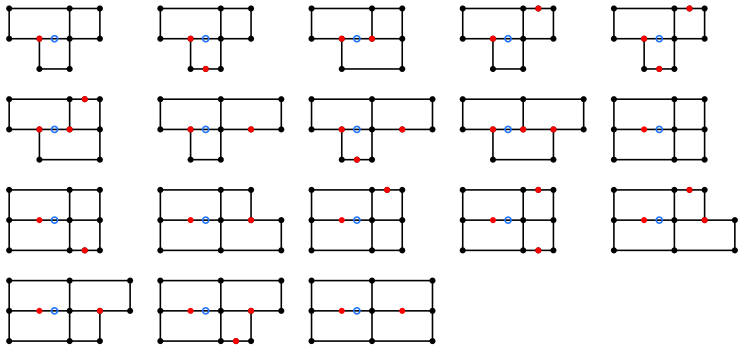


Figure 2.32: *DAS T-mesh T-edge stencils.*

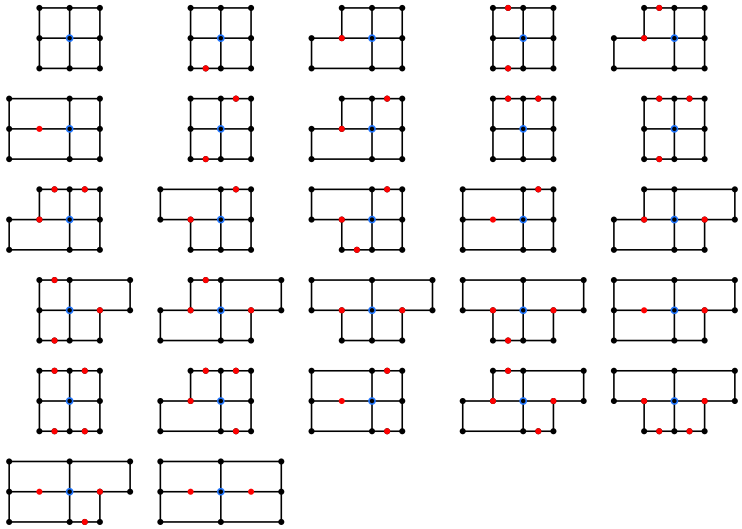


Figure 2.33: *DAS T-mesh vertex stencils.*

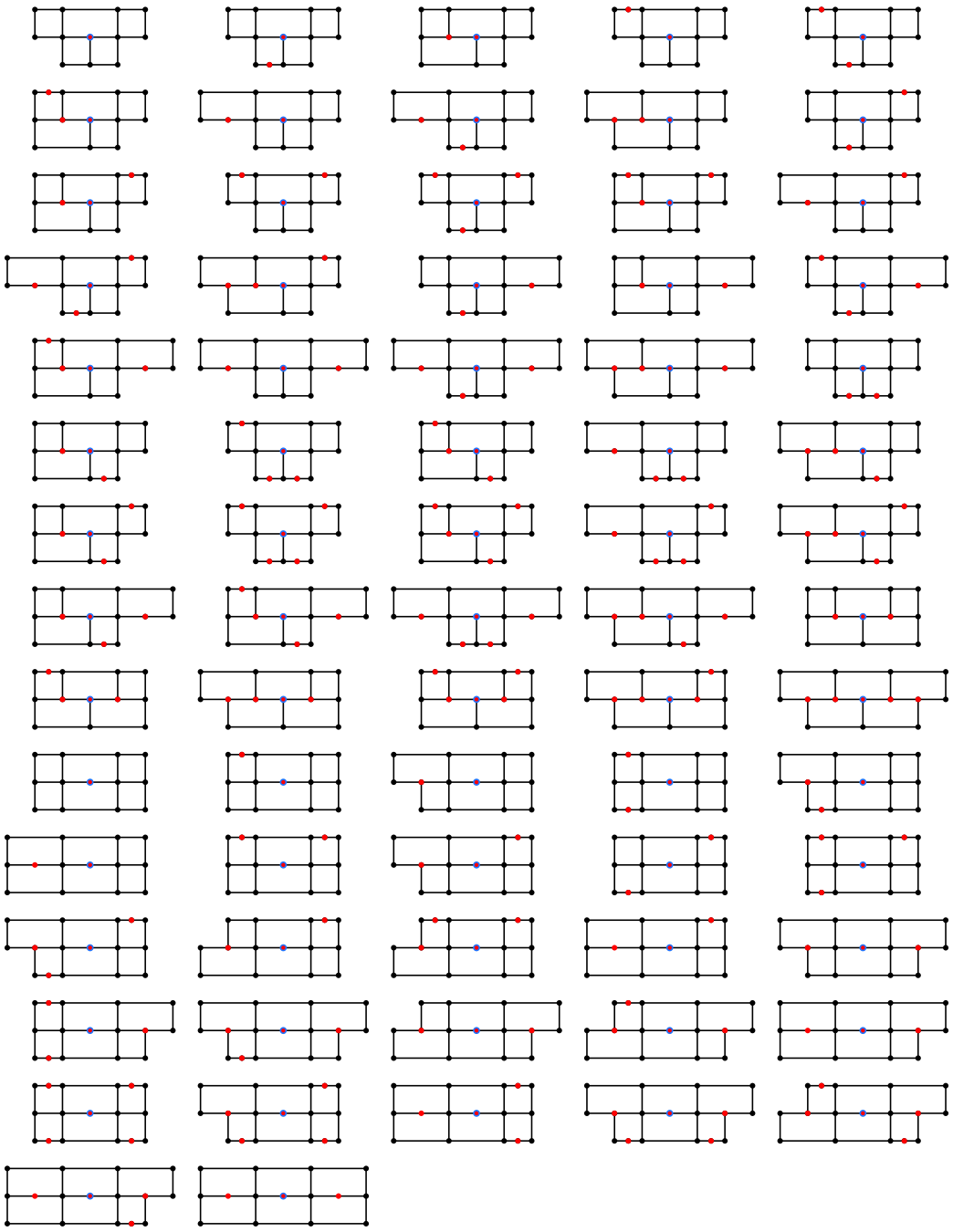


Figure 2.34: *DAS T-mesh T-vertex stencils.*

2.13 Control Meshes of Characteristic Maps for Dyadic T-meshes

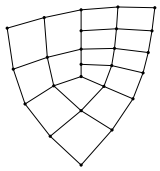


Figure 2.35: *DAS T-mesh valence 3 characteristic map.*

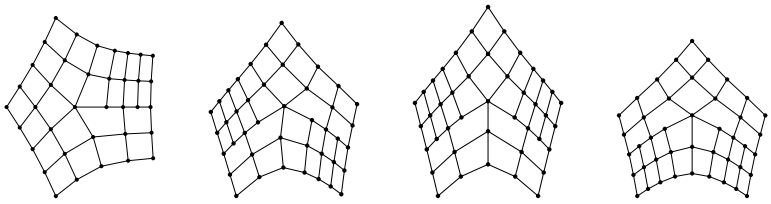


Figure 2.36: *DAS T-mesh valence 5 characteristic maps.*

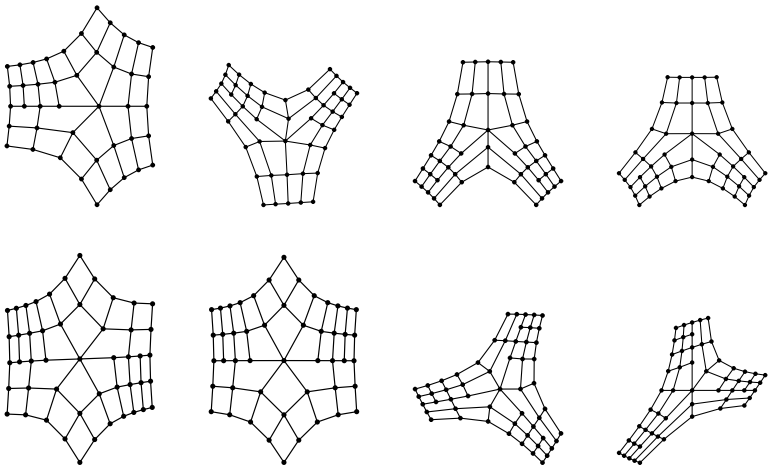
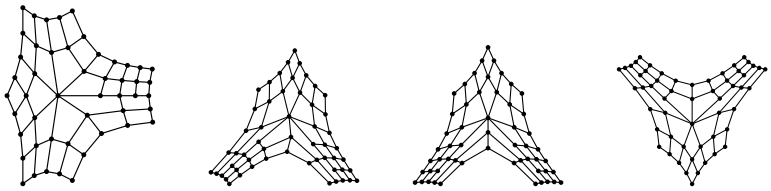


Figure 2.37: *DAS T-mesh valence 6 characteristic maps.*



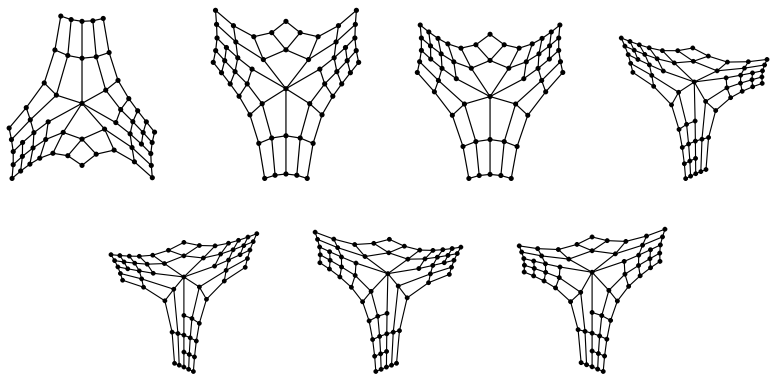


Figure 2.38: *DAS T-mesh valence 7 characteristic maps.*

Chapter 3

Real-Time Creased Approximate Subdivision Surfaces with Displacements

Here we extend the patch-based Approximate Catmull-Clark scheme to support sharp features such as creases and corners and derive stencils for the valence 2 border case.

This work was published as [Denis Kovacs, Jason Mitchell, Shanon Drone, and Denis Zorin. Real-time Creased Approximate Subdivision Surfaces. *I3D 09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, pages 155 160, 2009].

3.1 Introduction

High-order surfaces are a very compact representation of smooth objects: a small number of control points completely defines a surface. This feature makes high-order surfaces appealing when memory is limited, for example, in applications running on game consoles. High-order surfaces naturally support level-of-detail, allowing for a flexible quality-performance tradeoff, essential in the cases when an application must guarantee interactive performance across a variety of target hardware platforms.

At the same time, high-order surfaces require significantly higher computational resources for evaluation, which, until recently have limited their use in games. As graphics processing unit (GPU) compute density continues to outstrip memory and memory bandwidth, high-order surfaces have become an increasingly attractive option.

Catmull-Clark subdivision surfaces are the dominant high-order surface type used in feature films, particularly in the area of character modeling [21] [34]. Modeling with Catmull-Clark surfaces is familiar and intuitive to artists and the limit surface behaves well when the control mesh is animated. Recently, a simple and efficient, yet high-quality bicubic approximation of Catmull-Clark surfaces (ACC) suitable for integration into game engines was introduced [67]. Our goal is to extend ACC to support piecewise smooth surfaces with creases and boundaries with corners [56]. Surfaces with crease features are common in applications, which makes it essential to retain high visual surface quality of ACC near such points. Loop and Schaefer's original paper presents a construction for a smooth boundary (excluding a common situation described in Section 3.5). This construction can be used anywhere except at vertices where multiple creases meet, near corner vertices

Figure 3.1: *The Heavy Weapons Guy from the game* Team Fortress 2 *modeled as a Catmull-Clark subdivision surface and rendered with our technique. The character and his weapon contain sharp features which require crease support to render correctly. In the bottom image, the black lines indicate patch edges with tagged crease edges highlighted in green.*

on the boundaries, or on interior creases.

There are two important differences between corner points and interior points:

- Catmull-Clark surfaces may not have well-defined tangents at corner points, and the tangents of modified Catmull-Clark surfaces [6] turn out to be unsuitable for use in tangent Bezier patches for many meshes;
- Vertices of the control mesh tagged as corners need to be interpolated.

We demonstrate shading artifacts that result from using incorrect tangents at corner vertices, and present a formula for tangents that leads to good visual quality.

Interpolation of corner control points may result in artifacts ("overhangs") both in subdivision surfaces and their bicubic approximation. We discuss how these can be avoided. In addition to extending ACC, we discuss integration of our implementation with a production game engine, Valve's Source engine (Section 3.7). We present performance comparisons of instanced versus native tessellation as well as a comparison with results published by Ni et al. [76] for a C^1 scheme in Section 3.8.

3.2 Related Work

Our work is a direct extension of the work of Loop and Schaefer [67] and we refer the reader to that paper for a more detailed discussion of related work; here we present a brief summary. The central idea of using separate tangent and position fields to define visually smooth geometry without constructing a C^1 surface explicitly was introduced by Vlachos et al. [107]. A number of algorithms were proposed in the past to generate smooth (typically C^1) piecewise polynomial surfaces, but few attempted to match the visual quality of Catmull-Clark surfaces. A C^1 (almost everywhere C^2) patch approximation of Catmull-Clark surfaces was proposed by Peters et al. [79], but requires one or two additional subdivision steps. G^2-continuity is achieved by Loop et al. [68] but requires evaluation of relatively complex high-order patches. Techniques for direct evaluation of subdivision surfaces on GPUs [8, 99] require additional subdivision steps or multiple passes [12].

An important recent approach for smoothing quad meshes with C^1 patches was presented by Ni et al. [76]. In this work, all extraordinary quads (with at least one vertex of valence $\neq 4$) are converted to four triangular patches of total degree

4 each. As we target relatively low polygon count models common in games, many faces in such models tend to be extraordinary, which makes the lower complexity of control point setup and patch evaluation of ACC more appealing in our setting.

Subdivision surfaces with creases were introduced by Hoppe et al. [49], and rules for interior-independent creases and corners that ensure tangent plane continuity were introduced by Biermann et al. [6]. As we discuss in Section 3.4, these rules do not necessarily meet the needs of our application. Boubekeur [10] adds smooth crease curves to PN triangles as well as additional parameters for crease shape control, but does not consider corners.

As we discuss in Section 3.7, we have implemented two tessellation schemes on current hardware: **instanced tessellation** and ATI **native tessellation**. Instanced tessellation is available in shader model 3.0 hardware with vertex texture fetch capabilities such as NVIDIA GeForce 8x00 and ATI RADEON HD 2x00 while native tessellation is available on ATI RADEON HD 2x00 and later GPUs as well as the XBox 360 [60] [103]. In instanced tessellation, rendering multiple pretessellated meshes containing parametric and index data makes it possible to use the vertex shader to evaluate tessellated surface position [41]. In Grün as well as Ni et al., hardware instancing is used to accelerate rendering of PN-triangles and a patch-based C^1 surface construction respectively [44][76]. As we discuss in Section 3.8, instancing and native tessellation methods suffer from performance issues related to memory usage and memory bandwidth needed to transmit control point data. In order to eliminate this control point storage and transmission cost, three pipeline stages are added after the vertex shader in Microsoft's DirectX 11 as shown in Figure 3.2: the *hull shader*, the *tessellator* and the *domain shader* [42]. Developers will typically map vertex animation operations such as skinning to the

vertex shader, basis transformations such as ACC to the hull shader and high order surface evaluation to the domain shader. Each new stage has remained programmable so that developers can customize the functionality to suit their needs, for example extending ACC to support creases as we have done.

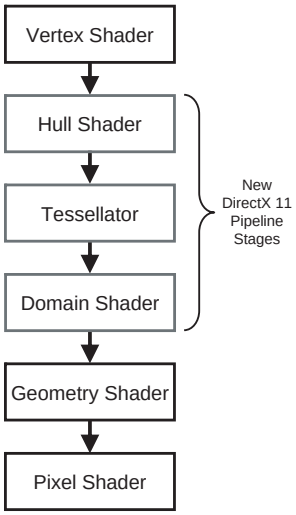


Figure 3.2: *DirectX 11 Pipeline.*

3.3 Bicubic Approximation of Catmull-Clark surfaces

We briefly review the construction of [67] to set up the notation. *Geometry patches* are Bezier patches of bidegree 3, defined by a grid of 16 control points (Figure 3.3): 4 *corner*, 8 *edge* and 4 *interior* points. Their positions are determined using fixed-weight masks depending on the valence (see [67] for mask definitions). The weights are chosen so that each edge point is the midpoint of two adjacent interior points, and each corner point is the centroid of the adjacent endpoints of all nearby patches.

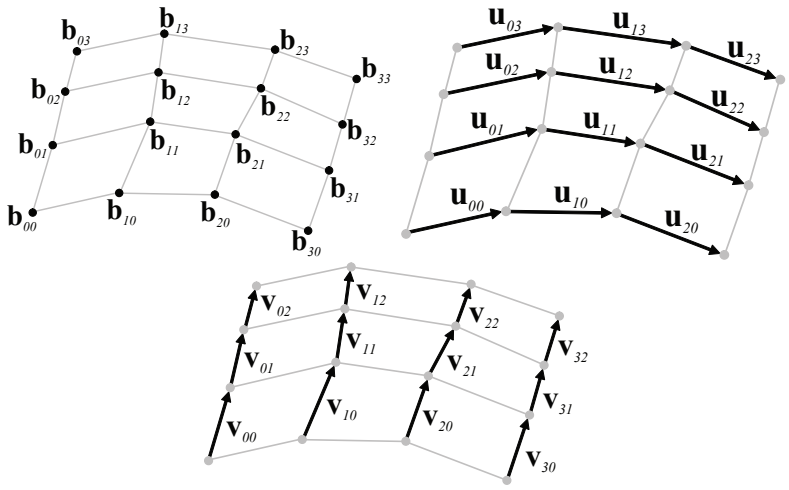


Figure 3.3: *Control points of geometry patches and control vectors of tangent patches.*

For boundaries, the weights for corners and edge points on the boundary are chosen to produce a B-spline curve on the boundary, with the exception of vertices of valence 2, which are forced to be corners (i.e., have 2 distinct tangents). The interior points are determined in the same way as for patches non-adjacent to the boundary, with the boundary vertex regarded as an interior vertex of valence $2k$, where k is the number of incident patches, which we call *face valence*. Informally, a smooth boundary vertex is regarded as interior vertex with a half-ring of incident patches.

Tangent patches are Bezier patches of degree (2,3), with 12 control vectors (Figure 3.3). Separate patches ∂_u and ∂_v are defined for parametric directions u and v; the formulas used for control vectors are the same, so we consider only the ∂_v tangent patch. The *corner vectors* are obtained using Catmull-Clark tangent mask weights applied to the ring of edge and face neighbor vertices of the corner point, with signs reversed for corners $\mathbf{v}_{02}$ and $\mathbf{v}_{32}$. For boundary points, tangent

masks of the modified Catmull-Clark scheme [6] are used.

All *edge* and *interior* control vectors are obtained from the geometry patch directly using the standard formula $\mathbf{v}_{ij} = 3\big(\mathbf{b}_{i,j+1} - \mathbf{b}_{ij}\big)$, excluding edge control vectors along the edges with two control points. These vectors are defined using correction factors that ensure tangent field continuity

$$\begin{aligned}\mathbf{v}_{1j} &= 3\big(\mathbf{b}_{1,j+1} - \mathbf{b}_{1j}\big) + \frac{1}{3}\big(2c_0\mathbf{u}_{1j} - c_1\mathbf{u}_{0j}\big)\\ \mathbf{v}_{2j} &= 3\big(\mathbf{b}_{2,j+1} - \mathbf{b}_{2j}\big) + \frac{1}{3}\big(2c_0\mathbf{u}_{2j} - c_1\mathbf{u}_{1j}\big)\end{aligned} \tag{3.1}$$

We re-derive these formulas in a slightly more general form to construct tangent fields at corner vertices in Section 3.4. Interior and edge control vectors are computed in the same way for patches adjacent to creases or boundaries.

3.4 Creases and Corners

A control mesh for a piecewise smooth surface of the type described in [6] has a number of edges tagged as *crease edges*. A vertex of a crease edge can be tagged as either a *crease smooth vertex* (default), or a *crease corner vertex*. Crease corner vertices are interpolated, and the crease curves may have two distinct tangents at the corner. We consider only the case of interior-independent sharp crease curves, which are completely defined by the control points on the crease.

If a vertex has more than two incident crease edges, we always tag it as a corner, so that it is interpolated, although some of the incident crease curves may have common tangents. In this setup, locally near a crease vertex, we can split the control mesh into independent parts (sectors), each of which can be treated as a surface with boundary (Figure 3.4). For smooth crease vertices, the rules of Loop

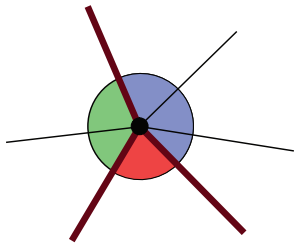


Figure 3.4: *Three sectors defined at a crease vertex.*

and Schaefer are used, excluding the case of face valence 2, for which Loop and Schaefer use a special-case corner rule.

To define corner rules, we first consider what can be regarded as the desired behavior at corner vertices.

Surface behavior near crease corners Intuitively, one expects the smooth surface to "follow" the control mesh; this natural requirement, combined with crease independence from the interior leads to unexpected difficulties at crease corner vertices. If one requires the surface to have a well-defined tangent plane at corners, and the tangent curves do not depend on control points away from the mesh crease, then the tangents of the two crease curves meeting at the corner determine the tangent plane of the surface. Furthermore, there are two types of possible local surface behavior [6]: convex and concave corners (Figure 3.5). One can observe that neither of these options matches the control mesh behavior: one intuitively expects the normal to the surface to be as close as possible to being perpendicular to the incident mesh edges. In contrast, at interior vertices, the tangent masks effectively average the tangent directions, so the resulting normal can be regarded as the average of normals of planes spanned by all possible pairs of incident edges.

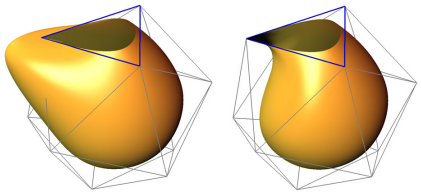


Figure 3.5: *Left: concave corner. Right: convex corner. [6]*

The situation is substantially different at corner vertices at the crease, as the crease independence requirement forces a tangent plane independent of interior control points. For the type of surface shown in Figure 3.5, this results in surface normals nearly parallel to mesh edges. For Bezier-interpolated normals, the problem is further exacerbated (Figure 3.6). Two possible approaches to resolving this contradiction are:

- Relax the tangent plane continuity requirement at crease corner vertices: introduce *cones*, so that a different normal corresponds to each edge direction at the corner vertex, but the normal is continuous everywhere else;
- Allow normal directions that are dependent on interior control points.

Depending on the desired appearance, either option may be suitable. However, in the context of ACC, the first option turns out to be unusable because of the limitations of the Bezier representation.

Cones Making corner crease vertices into cones presents two difficulties: first, there are a few smooth configurations for which a cone is not the best possible behavior. Second, even more importantly, *it is impossible to produce a single-point normal discontinuity with nondegenerate tangent Bezier patches.* Indeed, tangents $\mathbf{t}_i(s)$, $i = u, v$, along the boundary of a patch parametrized by s, are given

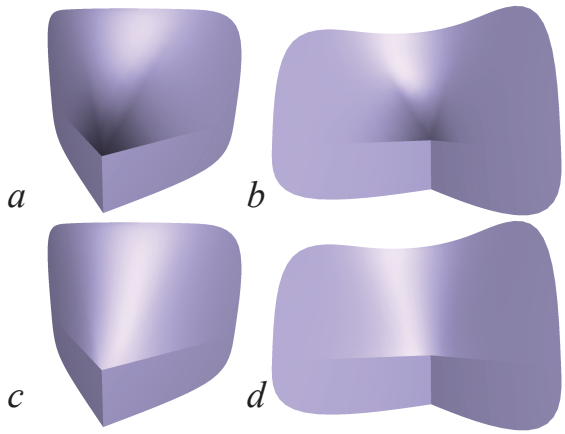


Figure 3.6: *Comparison of different tangent definitions for a corner: a,b: Edge tangents are computed as linear combinations of two crease tangents (modified Catmull-Clark). c,d: Our scheme used for tangent control vectors for the same control meshes.*

by cubic or quadratic polynomials, so (non-unit length) normal $\mathbf{n}(u) = \mathbf{t}_u \times \mathbf{t}_v$ is a polynomial of degree 5. The condition that normals on two sides of the boundary are collinear can be expressed as $\mathbf{n}(u) \times \widehat{\mathbf{n}}(u) = D(u) = 0$, where $\widehat{\mathbf{n}}$ is the normal computed for the same boundary curve for the adjacent patch. $D(u)$ is a polynomial of degree at most 10. If the normals are not collinear and nondegenerate at the corner vertex $u = 0$, then $D(u) \neq 0$ at $u = 0$, and can vanish at most at 10 points along the boundary, which makes normal continuity at *all* boundary points away from the corner vertex impossible. $D(0) = 0$ implies either collinear normals (which means the point is not a cone), or singular parameterization. It is clear how one can construct cones by collapsing control points of a Bezier patch to a single control point at one of the patch boundaries.

As singular parametrization is highly undesirable for tessellation (especially based on instancing as discussed in Section 3.7), and texture mapping, we consider the latter option impractical.

Interior-dependent tangent patches We adopt the second possible option, that is, introduce dependency of normals at crease corners on the mesh interior. Recall that the primary reason to use an interior-independent construction for edge and corner points on the crease is to ensure that patches on different sides of creases match perfectly. At the same time, the normals are discontinuous across creases, so no such constraint is essential for tangent patches, although it is still desirable that the normals on the crease depend only on control points on the same side. This observation leads to the following overall approach:

- Compute geometry patches in the same way as boundary patches in [67], but make sure that the crease corner point $\mathbf{c}$ is interpolated;
- Define a suitable tangent plane P for $\mathbf{c}$;
- Project crease edges incident at $\mathbf{c}$ to P, and obtain tangents for interior edges incident at c by interpolation of two normals.

To define the tangent plane at interior and smooth crease vertices, the tangents are obtained by applying fixed modified Catmull-Clark tangent weights, and the normal is computed from the tangents. In the case of corners, this solution is not available, as the modified Catmull-Clark surface of [6] the tangent plane is spanned by the two crease tangents, and the original Catmull-Clark surface in general is not tangent-plane continuous.

Instead, we use the average of the normals to geometry patches directly. If we choose the indices in each patch so that the crease corner vertex is $\mathbf{b}_{00}$, and number patches from 0 to k,

$$\mathbf{n} = \operatorname{norm}\Big(\sum_{i=0}^{k-1} \operatorname{norm}\big(\big(\mathbf{b}_{10}^{(i)} - \mathbf{b}_{01}^{(i)}\big) \times \big(\mathbf{b}_{10}^{(i)} - \mathbf{b}_{00}^{(i)}\big)\big)\Big) \tag{3.2}$$

where $\text{norm}(\mathbf{x}) = \mathbf{x}/|\mathbf{x}|$. In the case when normals average to zero, there is no meaningful common normal, and we use the cross-product of two tangents.

While this direct procedure is more expensive than computing a linear average of tangents using fixed weights, we found that the results are substantially better for all choices of averaging of tangents with which we have experimented. If the expense of this calculation is a concern, for all control point configurations we have examined in practical models just averaging the normal to the first and last geometry patches still yields a tangent plane superior to other alternatives.

Tangent interpolation Once the tangent plane P with unit normal $\mathbf{n}$ is defined, we project the crease tangents $\mathbf{t}_0^{init} = \mathbf{p}_0 - \mathbf{c}$ and $\mathbf{t}_k^{init} = \mathbf{p}_k - \mathbf{c}$, where $\mathbf{p}_0$ and $\mathbf{p}_k$ are two crease vertices adjacent to $\mathbf{c}$, and k is its face valence, to the tangent plane P to obtain two basis tangents $\mathbf{t}_i^P = \text{norm}\big(\mathbf{t}_i^{init} - (\mathbf{n} \cdot \mathbf{t}_i^{init})\mathbf{n}\big)$, $i = 0, k$. Next, we interpolate these tangents to obtain corner control vectors for each tangent patch incident at the crease corner vertex, in a way that insures tangent plane continuity.

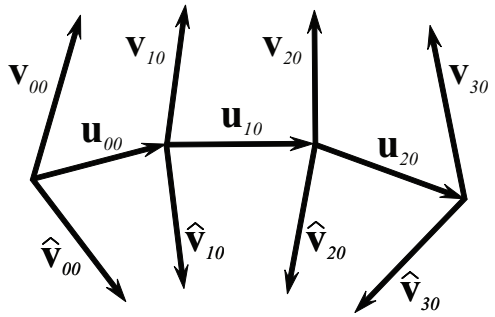


Figure 3.7: *Control vectors of tangent fields* $\mathbf{u}(t)$, $\mathbf{v}(t)$ *and* $\widehat{\mathbf{v}}(t)$.

We use a slightly more general form of patch continuity conditions of [67] used to derive (3.1). Consider three tangent fields defined on an edge e shared by two patches: the quadratic ∂_u field $\mathbf{u}(t)$, and two cubic ∂_v fields $\mathbf{v}(t)$ and $\widehat{\mathbf{v}}(t)$

(Figure 3.7). To ensure normal field continuity across the edge, these three fields have to be linearly dependent at each t. We can multiply $\mathbf{u}(t)$ by any linear function $(a+bt)$ without changing its direction. If three linear polynomials $(a+bt)\mathbf{u}(t)$, $\mathbf{v}(t)$, $\widehat{\mathbf{v}}(t)$ are linearly dependent, but pairwise independent (which is the case whenever there are no degeneracies in the patches), we can write the dependency condition as $(a+bt)\mathbf{u}(t) = c\mathbf{v}(t) + d\widehat{\mathbf{v}}(t)$, with $c, d \neq 0$. As a and b are arbitrary, we can set c to one. As the choice of order between $\mathbf{v}$ and $\widehat{\mathbf{v}}$ is arbitrary, it is natural to require that $c = d$, and c can be chosen to be 1. As for polynomials to coincide, their Bezier points have to coincide, we arrive at the conditions of a form similar to [67], but with undefined a and b.

$$a\mathbf{u}_{00} = \mathbf{v}_{00} + \widehat{\mathbf{v}}_{00},\ b\mathbf{u}_{20} = \mathbf{v}_{30} + \widehat{\mathbf{v}}_{30} \tag{3.3}$$

$$\frac{1}{3}\big(b\mathbf{u}_{00} + 2a\mathbf{u}_{10}\big) = \mathbf{v}_{10} + \widehat{\mathbf{v}}_{10},\ \frac{1}{3}\big(2b\mathbf{u}_{10} + a\mathbf{u}_{20}\big) = \mathbf{v}_{20} + \widehat{\mathbf{v}}_{20} \tag{3.4}$$

As in [67], equations (3.4) can be satisfied by a suitable choice of $\mathbf{v}_{10}$, $\mathbf{v}_{20}$, $\widehat{\mathbf{v}}_{10}$, and $\widehat{\mathbf{v}}_{20}$ as functions of other control vectors. We index the edges e incident at the crease corner $\mathbf{c}$ counterclockwise starting with a crease edge. The vectors $\mathbf{u}_{00}$, $\widehat{\mathbf{v}}_{00}$, and $\mathbf{v}_{00}$ are tangents along an edge e_j, the previous edge e_{j-1} and the next edge e_{j+1}, respectively. If we denote them by $\mathbf{t}_j$, $\mathbf{t}_{j-1}$ and $\mathbf{t}_{j-1}$, each of the equations (3.3) has the form

$$\mathbf{t}_{j+1} = a_j\mathbf{t}_j - \mathbf{t}_{j-1}$$

$j = 1 \ldots k-1$, where k is the face valence of $\mathbf{c}$. In [67] two choices of formulas for $\mathbf{t}_j$ (Catmull-Clark tangent formulas for interior vertices and smooth boundaries) satisfy these equations for a fixed a. Three additional natural assumptions deter-

mine a unique answer: (1) $a_j = a$ independent of j; (2) $\mathbf{t}_0 = \mathbf{t}_0^P$ and $\mathbf{t}_k = \mathbf{t}_k^P$ so that the tangent control vectors on the crease are aligned with the crease curve tangents as well as possible; (3) if these tangents are of equal length, we expect all inferred t_j $j = 1 \ldots k$ are of equal length. In this case,

$$\mathbf{t}_i = \frac{1}{\sin\theta}\big(\sin\frac{(i-k)\theta}{k}\mathbf{t}_0^P + \sin\frac{i\theta}{k}\mathbf{t}_k^P\big) \tag{3.5}$$

where θ is the angle between the tangents $\mathbf{t}_0$ and $\mathbf{t}_k$. The angle θ is measured in counterclockwise direction if we look at the tangent plane from the direction of the averaged normal $\mathbf{n}$. Equation (3.5) leads to numerical difficulties if θ is close to π, although resulting tangent vectors are well-behaved even in the limit $\theta = \pi$. If t_0 and t_k are normalized to be of the same length, which we take to be the average of their lengths, one can write this expression in a less symmetric but more stable form as

$$\mathbf{t}_i = \cos\frac{i\theta}{k}\mathbf{t}_0^P + \sin\frac{i\theta}{k}\mathbf{n} \times \mathbf{t}_0^P \tag{3.6}$$

Formulas (3.2) and (3.6) define tangent control vectors at crease corner vertices. We have found this approach to be quite robust and insensitive to perturbations and degenerate cases. Examples of applying these formulas are shown in Figures 3.6 and 3.14 (right).

3.5 Smooth Creases for Face Valence 1

Smooth creases can be thought of as boundaries inside the mesh. Loop and Schaefer describe geometry patch stencils and limit tangents for boundaries, but choose

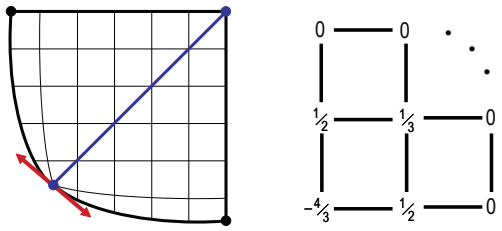


Figure 3.8: *Face valence 1 crease vertices.*

to define a crease vertex of valence 2 to be a corner, defining a special-case rule. However, we have observed in practice (Figure 3.8) that it is often desirable to have smooth boundaries with face valence 1 vertices.

Defining the crease to be a uniform B-Spline curve as in the other cases results in the problem that now the tangents and bitangents of the geometry patch are collinear and no longer define a normal.

The limit normal, however, still exists and can be found by considering tangents along the curve $\gamma(s) = B(s, s)$ where $B(u, v)$ is the Bezier patch. A direct computation shows that up to higher order terms, the tangent is $\gamma'(s) = s\mathbf{t}_3 + O(s^2)$, where $\mathbf{t}_3$ can be computed from the control points using the mask in Figure 3.8. This allows us to obtain the normal to the surface at the smooth vertex with $k = 1$ as the cross product of the tangent to the crease curve and $\mathbf{t}_3$.

In practice, one can avoid a special-case code for the normal computation by perturbing the boundary tangents instead, setting them to $\mathbf{t}_0 + \epsilon\mathbf{t}_3$ and $\mathbf{t}_1 + \epsilon\mathbf{t}_3$ respectively. We use $\epsilon = 10^{-4}$. As a result, the tangent control vectors at the boundary are no longer perfectly collinear; we could not find any cases where a perturbation of this magnitude would cause artifacts, yet it is sufficiently large to make the normal computation stable. The result of applying this procedure on a car model are shown in Figures 3.15 and 3.16.

3.6 Artifacts and How to Fix Them

Just as for subdivision surfaces, many different types of visual quality defects can be identified. We address the two most significant types of artifacts that occur near corners.

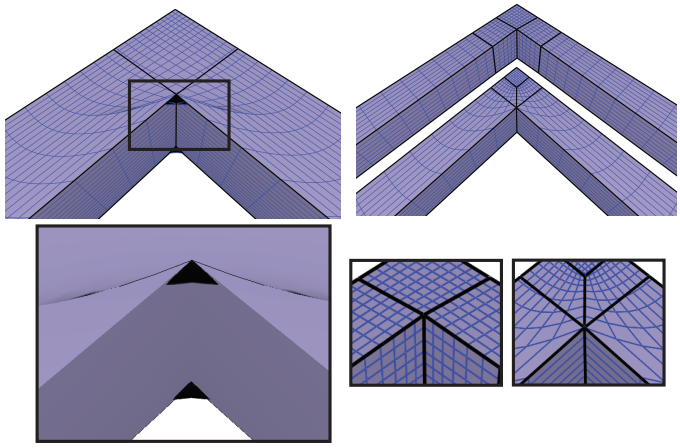


Figure 3.9: *Left: An ACC surface with an overhang. Right: a modification of the control mesh eliminating the overhang.*

Overhangs Crease corner points are interpolated by an ACC surface, while nearby noncorner mesh vertices are moved to averaged positions of their neighbors. This, in turn, affects placement of interior and edge Bezier points near a corner. As the edge Bezier points determine tangent directions at patch boundaries, for highly nonuniform meshes, extreme shifts of tangents may result in patches with angles between tangents exceeding 180 degrees. For reasons discussed in [6], a Bezier patch, which always has convex corners in the parametric domain, cannot have a concave corner, so the patch folds over and approaches the boundary curves from the smaller angle side, developing *overhangs* shown in Figure 3.9a. The problem disappears if the designer chooses a more uniform set of quads near the crease

corner or increases its valence (Figure 3.9b).

Lack of smoothness near convex corners For highly non-planar corners of low valence, in some cases, the bicubic patches cannot approximate the behavior of the subdivision surface well, even if a least-squares fit is used as close as possible to the surface. In this case, the angle between actual normals of geometry patches is significant and cannot be fully masked by using tangent patches (Figure 3.10). In this case, increasing the valence of the crease corner or decreasing the size of adjacent faces solves the problem.

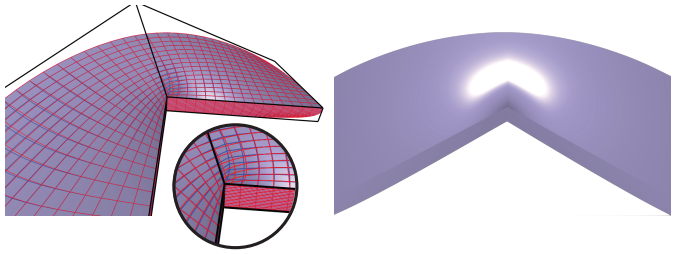


Figure 3.10: *Non-smooth appearance near a corner. Note the mismatch between the Catmull-Clark surface (red) and bicubic patches (blue), and the sharp angle between parametric lines on adjacent patches.*

3.7 Implementation

We have implemented the above extensions to ACC in Valve's Source engine. We have mapped the DirectX 11 pipeline onto DirectX 9, including instanced and native tessellation codepaths where the vertex shader and hull shader are implemented in software on the CPU and the remaining stages are executed on the GPU.

The CPU-side vertex shading operations include skinning, vertex morphing and other operations which are appropriate to perform at the control mesh level. Post-

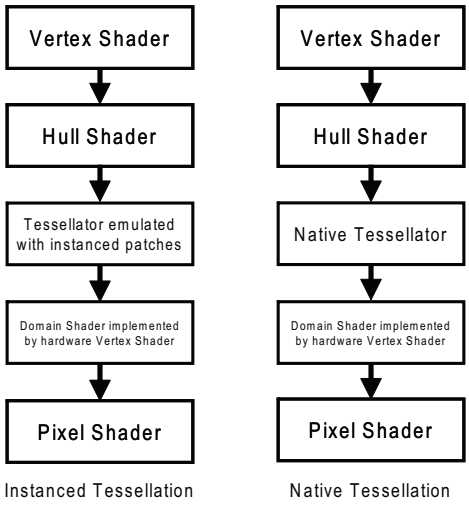


Figure 3.11: *DirectX 11 pipeline mapped onto instanced and native tessellation on DirectX 9.*

transform control mesh vertices are then sent to the software hull shader where they are mapped to a set of Bezier patches using our technique. This data is copied asynchronously to GPU memory. Domain points are instantiated with appropriate mesh connectivity on the GPU using either hardware instancing or ATI's native hardware tessellator. After this on-chip data amplification, the vertex shader—playing the role of *domain shader*—evaluates Bezier patch point positions and tangent frames using Bezier control points fetched from the floating point texture generated earlier by the CPU-side hull shader.

Native Tessellation ATI's hardware tessellator instantiates the vertex shader at u, v points in the $[0..1]^2$ domain and provides the shader with access to all of the "super-primitive" data from the input vertices [60] [103]. The shader can use the input super-primitive data and the Bezier patch data to evaluate patch attributes. The remainder of the graphics pipeline is unchanged, so an implementor need only alter existing vertex shaders and vertex buffer layouts. In Valve's Source engine,

the changes necessary to add this functionality to existing production-tested vertex shaders were minimal, though we did run into some limitations of the DirectX 9 vertex shader programming model, particularly the size of the general purpose register file, as we discuss in section 3.8.2.

Instanced Tessellation It is also possible to emulate tessellation by using instancing hardware to replicate a pretessellated patch across the input mesh, one instance for each ACC patch [44] [76]. The vertex shader then evaluates the bicubic patch in the same manner as in the native tessellated version. Due to some implementation details dictated by the ATI native tessellator interface, the vertex shader instruction counts are not identical between the two codepaths, as we discuss in the next section.

3.7.1 Displacement Mapping

In addition to approximating the Catmull-Clark limit surface, it is possible to compactly represent high frequency detail by displacing the vertices from the approximate limit surface [26] [58]. We have written an extractor which processes the Catmull-Clark control mesh and a separate high-resolution detail mesh to generate a scalar displacement map relative to our approximation to the Catmull-Clark subdivision surface [23]. Without displacement, each invocation of the domain shader performs 30 fetches of packed control point data and the inclusion of an additional data fetch to access a displacement map has negligible incremental performance impact. Likewise, the handful of arithmetic operations necessary to displace the vertex relative to the approximate limit surface are insignificant. Since the runtime cost is essentially free, the real barriers to the adoption of displacement map-

ping are the additional memory burden of storing the displacement maps and the tool investment necessary to integrate displacement mapping into a studio's art pipeline. There are available commercial tools such as ZBrush, MudBox and others which can output heightfield textures suitable for use as displacement maps. In these tools, however, the height maps are computed relative to the Catmull-Clark limit surface. In our case, of course, we are rendering an approximation made up of bicubic patches using a separate normal field which is designed to provide plausible lighting despite the fact that the geometry patches are not necessarily C^1 at patch boundaries. As a result, we have written a displacement map extraction tool which uses the creased ACC geometry and normal fields in the extraction process to ensure that the displacement maps are computed relative to the approximate limit surface. In Figure 3.12, we see a Vortigaunt character from the game *Half-Life 2* rendered as an approximate Catmull-Clark subdivision surface. In the left column of images, we see the smooth approximate limit surface shaded with a simple Phong shader, using a normal map to provide some detail in the lighting. In the right column of images, displacement mapping has been applied to the character to add surface detail.

3.8 Performance

We have analyzed the performance of both the CPU conversion of the Catmull-Clark quad mesh to the bicubic approximation and the GPU evaluation of the resulting bicubic patches. In our case, the Catmull-Clark to ACC evaluation was done on the CPU, though it can be performed on the GPU instead [76]. Naturally, the patch evaluation is handled on the GPU, after the data has been amplified

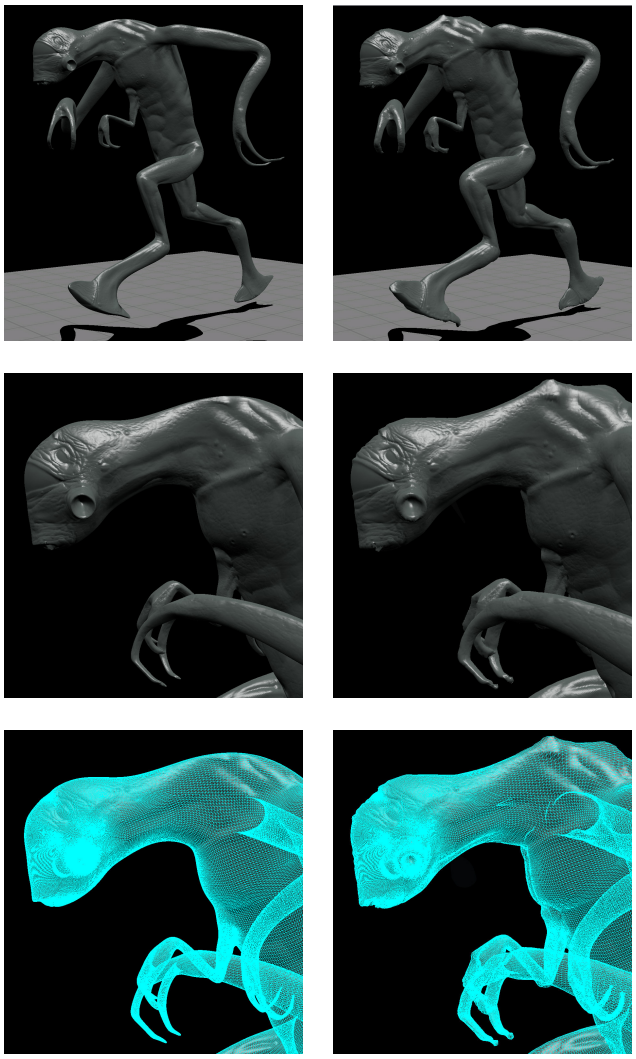


Figure 3.12: ***Left column:*** *A Vortigaunt from* Half-Life 2 *rendered as an approximate Catmull-Clark subdivision surface.* ***Right column:*** *The Vortigaunt rendered with displacement mapping. The bottom images show wireframe to illustrated the post-tessellated mesh density.*

either through instancing or using ATI's native tessellator.

3.8.1 CPU Performance

The primary bottleneck in the CPU conversion from Catmull-Clark to ACC is the computation of the tangent patches, accounting for more than half of the total CPU time. We can avoid this cost on regular patches since these patches do not necessarily require separate tangent patches. Hence, the performance is dependent on the mix of valences in the mesh being converted, where a mesh consisting of only regular patches could see as much as a twofold performance increase over a wholly extraordinary mesh. We then vectorized the conversion math using CPU SIMD operations. In addition we created lookup tables for commonly computed values and unrolled much of the looping math. This roughly doubled performance over our original implementation. Third, we parallelized the computation. Since each patch evaluation is independent of all other patch evaluations, it splits across multiple cores easily. Using parallelization, we achieved an additional performance improvement of around **3.5x** on 4 cores for meshes between 1000 and 10,000 patches.

3.8.2 GPU Performance

We now compare performance of instanced tessellation and native tessellation for a variety of datasets and analyze GPU bottlenecks at various stages of the pipeline.

In Table 3.1, we compare instanced and native tessellation performance of the datasets shown in Figure 3.13 using the ATI RADEON 4870 X2, which is able to run both codepaths. Due to differences in the hardware interfaces, the native tessellation shader uses roughly 16% more instructions than the instanced patch shader. In our measurements, we have seen that the instanced patch performance is

frequently as much as twice as fast as the native hardware tessellation performance. We conclude that the performance improvement seen when using instanced patches is not entirely related to shader length, but rather, related to driver or hardware implementation. (Both perform the same number of texture operations.) We have also compared our system's performance to that of Ni et al. [76] as shown in Table 3.2. (In this case, we have used a GeForce 8800 GT in order to remain consistent with the hardware used by Ni et al.)

Table 3.1: ***Performance Comparisons*** *- The Car, Ship and Poly models contain 1164, 5180 and 10618 quad faces. Performance numbers are in frames per second, measured on an Intel Quad Core Q9450 2.66GHz and ATI RADEON 4870 X2. N = number of tessellated vertices per control mesh edge.*

	Native Tessellation			**Instanced Tessellation**		
Mesh	N=3	N=9	N=15	N=3	N=9	N=15
Car	1344	1296	589	1550	1301	846
Ship	1245	326	137	1196	473	222
Poly	747	160	65	532	304	132

We have also found that ATI's native tessellation path seems to be more impacted by the rest of the pipeline state, notably the number of interpolators output from the vertex shading unit to triangle setup, as well as the complexity of the pixel shader. The numbers in Table 3.1 were generated with a vertex shader which outputs two interpolators to a trivial pixel shader. If we output five interpolators to a 22 instruction pixel shader, we measure a 1.8x to 2x performance hit when using native tessellation. The instanced tessellation path sees no such performance penalty.

Using NVPerfHUD, we have determined that both the instanced and native hardware tessellation shaders are vertex texture fetch bound. Each invocation of

Table 3.2: *Instanced results using our technique for the Sword and Frog datasets which contain 138 and 1292 quad faces respectively. Performance data from [Ni et al. 2008] is shown on the right.*

	Instanced Results			**Ni et al.**		
Mesh	N=9	N=17	N=33	N=9	N=17	N=33
Sword	1480	1480	539	965	965	703
Frog	728	256	63	392	226	87

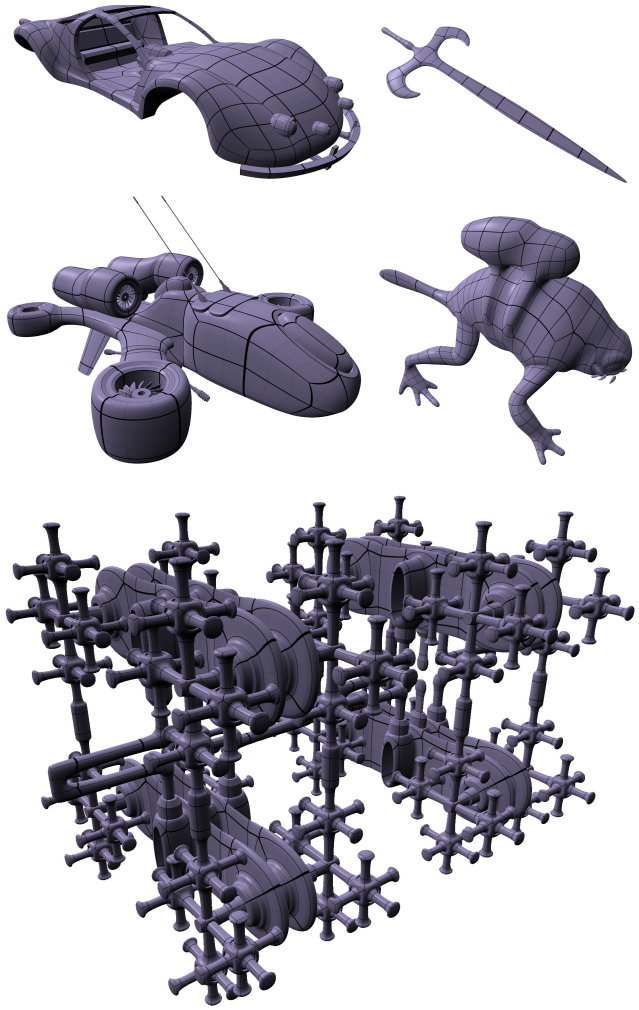


Figure 3.13: *Car, Sword, Ship, Rocket Frog and Poly models.*

the domain shader performs 30 fetches of packed control point data (12 for the control points, and 9 for each of the two tangent patches). For regular patches (with all vertices of valence 4), we can avoid fetching the tangent patch control points and use the de Casteljau algorithm to compute both positions and normals. This saves 18 texture fetches for these patches at the expense of drawing regular and extraordinary patches with two API calls rather than one. In this case, we measured a 20% (1.9ms) performance improvement in GPU evaluation cost at N = 33 for the rocket frog mesh by splitting evaluation of regular and extraordinary patches. In addition, we avoid calculating tangent patches for regular patches when converting from Catmull-Clark to ACC in the hull shader. In our implementation, this saves an additional 0.26 ms on the rocket frog.

In practice, care should be taken with small meshes (<2000 patches) with few regular patches. Separating regular and extraordinary vertices requires two API calls—as opposed to just one—and the CPU-side overhead of this extra API call can outweigh the savings gained by reducing the shader load for regular patches. Additionally, we found it advantageous (and in some cases necessary) to keep the use of vertex shader general purpose registers (GPRs) to a minimum, particularly when combining patch evaluation with some of the more advanced vertex shaders that we have used in shipping games such as *Team Fortress 2*, *Portal*, *Left 4 Dead* and the *Half-Life 2* series. To reduce the number of GPRs, we split the loading and evaluation of the control point patch and the loading and evaluation of the tangent patches, allowing GPRs to be reused between position evaluation and tangent evaluation. This made the implementation somewhat awkward, but the DirectX 9 vertex shader programming model simply exhausted its general purpose register bank without such shader massaging.

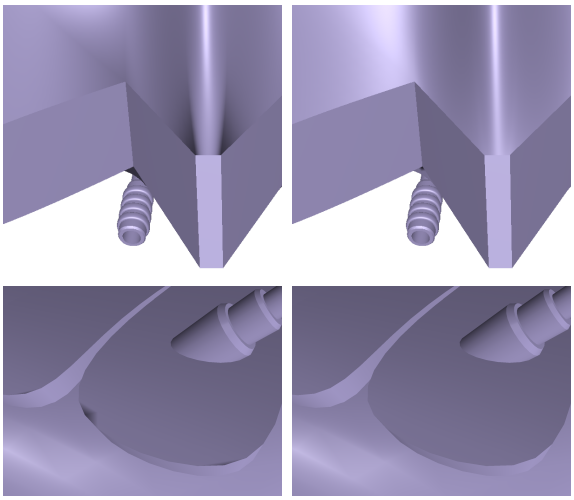


Figure 3.14: *Top: an example of corner artifacts on the spaceship model (left) eliminated by our technique (right). Bottom: an example of face valence 1 smooth crease vertex with vanishing normal (left) and with the normal computed using our technique (right).*

3.9 Future Work

In the future, we intend to address the topic of *adaptive subdivision*, which is critical for performance and LOD, particularly as we move beyond isolated character and object meshes and to complex terrain geometry. Given the new programming model introduced in DirectX 11, we expect that it will be necessary to develop new error metrics and schemes for determining the appropriate level of detail for a given primitive or primitive edge, particularly when performing displacement mapping. In addition to the optimizations described in Section 3.8, we would like to explore culling operations appropriate to a displaced patch representation. Compelling speedups have been reported by Lee et al. through the use of normal masks [58] [114].

Overall, we have obtained high quality results with our extensions to ACC, although the artifacts (Section 3.6) require designers to adapt the models. While

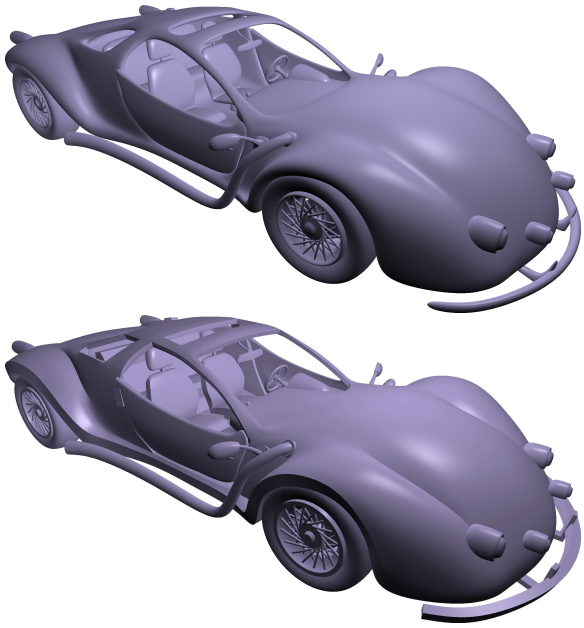


Figure 3.15: Top: *A car model rendered with smooth ACC.* Bottom: *The same model with creases and corners added using our method.*

some of the limitations are fundamental (it is impossible to create cones or approximate well the subdivision surface near certain types of corners without refinement), one can hope to design techniques to deal with some of these problems automatically; this is a promising direction for future work.

We have integrated our technique with the Source engine's skeletal and facial morphing systems as shown in Figure 3.1. Going forward, we plan to explore additional animation techniques including fluid simulation, cloth simulation and free-form deformation (FFD) of the low-resolution quad mesh. We anticipate having to make changes to such simulations based on the fact that we are animating a subdivision surface control mesh rather than the final polygonal primitives to be displayed.

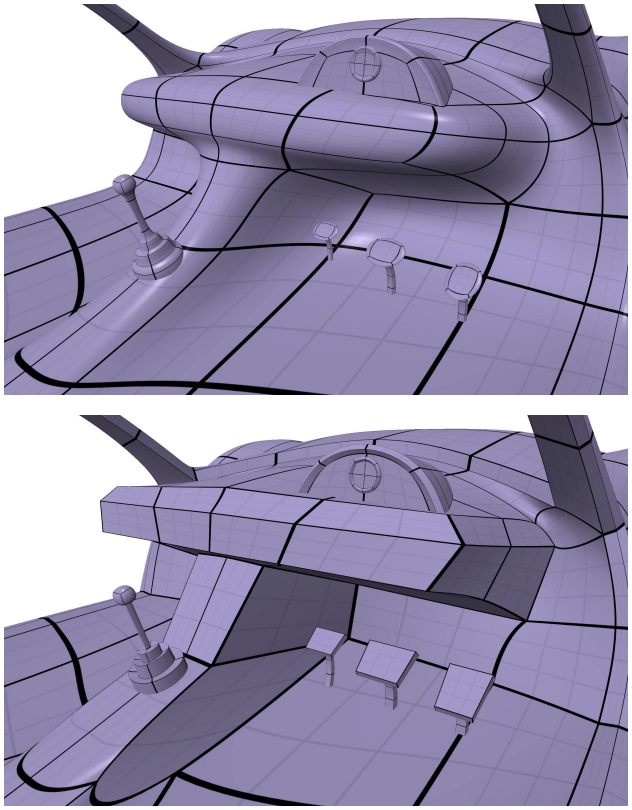


Figure 3.16: Top: *The dashboard of our car model rendered with ACC.* Bottom: *The same model with creases and corners added using our method.*

Part II

Mesh Generation

Chapter 4

Anisotropic Quadrangulation

In this chapter, we present a simple and efficient technique to add curvature-dependent anisotropy to harmonic and feature-aligned parameterizations and improve the approximation error of the resulting quadrangulations. We use a metric derived from the shape operator that results in a more uniform error distribution, decreasing the error near features.

This work was published as [Denis Kovacs, Ashish Myles, and Denis Zorin. Anisotropic quadrangulation. *Computer Aided Geometric Design*, 28(8):449 462, 2011. Solid and Physical Modeling 2010].

4.1 Introduction

Most common techniques for generating meshes from range scans and volumetric data produce irregular meshes with complex connectivity. A surface can be stored in a much more compact form, simplifying and speeding up rendering and processing if it is converted to a predominantly regular mesh, with only a small number of irregular vertices and faces. It is desirable to minimize the number of vertices

in the semiregular mesh, while keeping it close to the original mesh.

Recent quadrangulation algorithms use a *global parameterization* of a mesh; the new mesh is obtained using a regular sampling pattern in the plane. Quite often, the parameterization is optimized to be as isometric possible. However, isometric parameterizations may be far from optimal for surface remeshing, if the goal is to obtain a surface as close as possible to the original for a given number of faces. For example, a cylinder can be mapped isometrically to the plane, resulting in a uniform sampling pattern on the surface. It can, however, also be meshed with single long quads stretched along the axial direction, with the

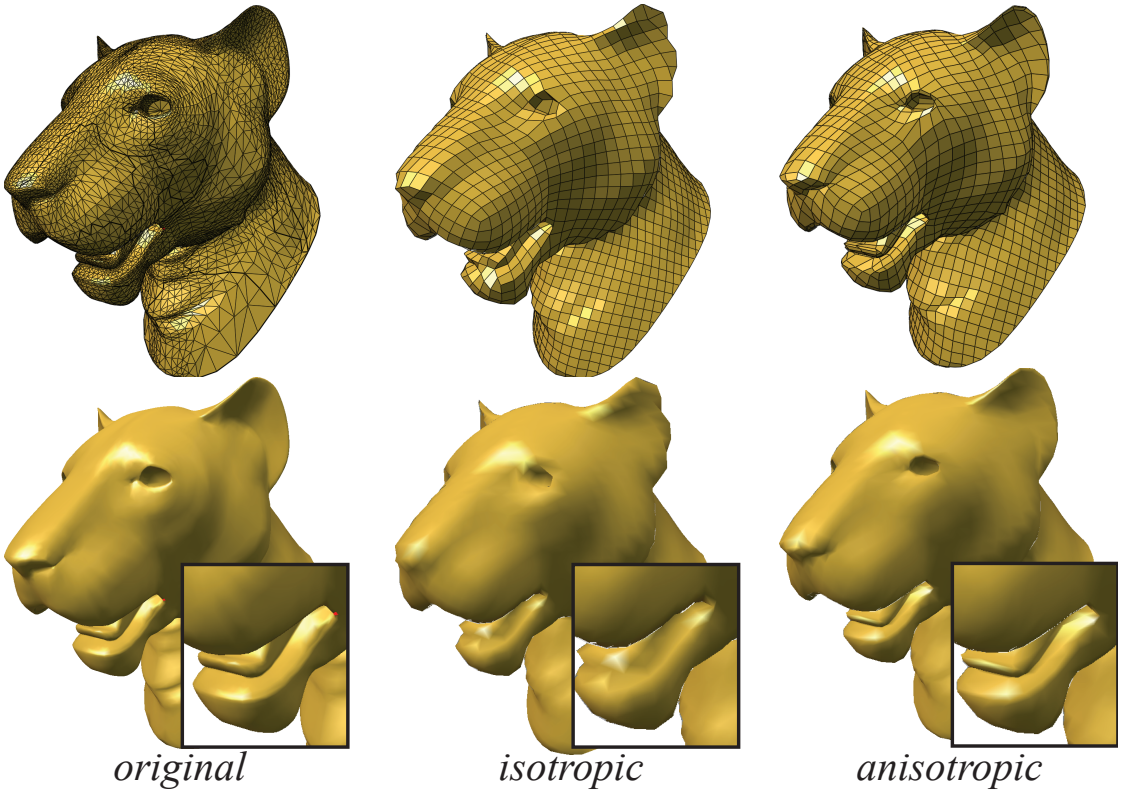


Figure 4.1: *Quadrangulations of a lion head model. Left: the original model; middle: isotropic feature-aligned quadrangulation (25% reduced); right: anisotropic feature-aligned quadrangulation.*

same approximation error. We call quadrangulations that adapt the quad aspect ratio to the surface shape *anisotropic.* We present a *simple* and *robust* method for computing anisotropic quadrangulations with quad aspect ratios adapted to local curvature, obtaining a good surface approximation with fewer quads.

Our method utilizes a curvature-based surface metric and computes the parameterization using this metric, rather than the Euclidean metric. Our approach is compatible with most parameterization methods that only rely on intrinsic quantities and vector fields on the surface.

Defining a metric for meshes is conceptually simple: we assign a new length to each edge. However, each edge length has to satisfy local triangle inequality constraints. It is a surprisingly difficult task to ensure that no inequality is violated, and while it may still be possible to compute a parameterization, the results may not have the desired anisotropic behavior (Section 4.5). We solve this problem using the idea of a high-dimensional embedding [82, 14]: the Euclidean metric in the higher-dimensional space defines the new edge lengths for the mesh. The embedded vertex coordinates consist of the original positional and normal coordinates, making the new edge length computation straightforward.

4.2 Related Work

The literature on parameterization, remeshing and quadrangulation is vast; [82], [14] and [31] are the most closely related to our work. Our key observation is that the high-dimensional embedding proposed in [14] to obtain anisotropic quadrangulations with the quad aspect ratio determined by the ratio of principal curvatures can be applied in the context of a particular class of parameterization techniques,

and yields robust results while preserving fine surface features.

There are many related works considering optimal anisotropic meshes in function approximation context (some recent work includes [1] [15] and [71]). Starting from [31], anisotropic mesh generation is often based on defining a suitable metric for the desired approximation measure, so that the isotropic triangulatulation in this metric results in optimal approximation. Our approach can be viewed as an application of the same general idea to the surface quadrangulation problem for a particular choice of error measure.

Many recent quadrangulation methods (in contrast to the work based on the construction of base complexes by simplification [39, 59, 54, 30]) have similar structure: a global parameterization is obtained by solving equations for gradients of parametric functions, and a new mesh is generated by following parametric lines. The two main categories of methods of this type are harmonic and feature-aligned.

Harmonic and conformal methods (for brevity we will we refer to both as harmonic) are robust, efficient and typically produce good results even for complex meshes for a suitable choice of singularities and boundary conditions. Some quadrangulation methods use harmonic maps directly [35, 105]. These methods can be viewed as minimizing nonconformality of the map, while allowing significant area scaling; nonlinear methods such as [96, 100] are needed to guarantee a one-to-one parameterization. Extreme area distortion is reduced by adding singularities (or "cones") to the parameterization, with several methods for automatic placement of singularities proposed in [35, 4, 100]. These techniques allow explicit user control over the number of irregular points on the mesh. The downside of harmonic techniques, especially in the context of remeshing, is that non-intrinsic shape information is not used directly.

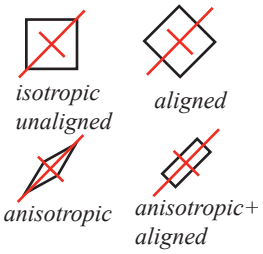


Figure 4.2: *Quad alignment and anisotropy.*

The shape information can be taken into account in two distinct ways to minimize the approximation error. Locally, a smooth shape can be characterized by its shape operator. Figure 4.2 show two ways of taking the shape operator into account (with principal curvature directions scaled by inverse principal curvatures shown in red).

A "perfect" quad of a given area approximating a surface is *aligned*, i.e., has edges parallel to principal curvature directions and *anisotropic* i.e., has aspect ratio inversely proportional to the ratio of principal curvatures. This corresponds to two classes of feature-aware parameterization techniques.

Feature-alignment methods [85, 52, 9] adapt the parameterization to the shape by aligning new mesh elements with a feature field, typically derived from the principal curvature direction field, either by smoothing, or interpolation of salient features. The singularities of the parameterization are determined by the singularities of the field, so the feature field cannot match the actual curvature field too closely: substantial smoothing is needed to keep the number of singularities small. The shape of the quads generated by these techniques tends to be uniform, rather than anisotropic: one can view these techniques as minimizing non-isometry, while aligning with the feature field. [9] permits a degree of anisotropy, penalizing

changes in length less than changes in the direction, but without relating these to curvature.

In geometric modeling, *anisotropic parameterization* was introduced as signal-specialized parameterization [90, 104]. This work uses a metric derived from the Hessian of the signal to adapt the parameterization to a signal defined on the surface; in particular, the surface itself can be used as the signal. Zayer et al. [112, 110, 111] describe a general class of parameterization methods based on solving a generalized Laplace or Poisson equation using a tensor field, which can be interpreted as a metric tensor. An elegant formulation for related quasi-conformal maps based on Beltrami factors described in [113]. The interpolation and stiffness properties of anisotropic linear triangles in finite-element context are discussed in detail in [98].

[22] derives bounds on the Haussdorf-distance approximation of manifolds using a metric closely related to the one that we use.

We show how to use a metric defined on a surface to obtain anisotropic versions of global quadrangulation algorithms, both harmonic and feature-aligned, and demonstrate the improvements in surface approximation that can be obtained in this way. To the best of our knowledge, metric-based techniques were not yet applied to quadrangulation, although [105] suggests that this is possible by altering the Laplace equation coefficients without suggesting a specific way to compute the metric.

We emphasize that we view using anisotropic metric as complementary to curvature-alignment approaches, rather than alternative to these. Curvature-alignment methods allow to obtain a geometrically meaningful set of singularities and coarse alignment with the shape; anisotropy helps to resolve sharp features

locally with fewer vertices, and allows to keep the number of parameterization singularities low.

4.3 Anisotropic Metric

The main idea of our approach is to define a new metric (that is, new edge lengths) on a mesh, and use an isometry-approximating parameterization based on these edge lengths for quadrangulation. The discrete metric is given by Equation (4.12). Our goal in this section is to explain the motivation for this choice. First, we discuss the *local error* and the choice of the best approximating quad; under the assumptions that we make, and similarly to previous work, the optimal quad is aligned with principal curvature directions, and has aspect ratio proportional to the ratio of principal curvatures.

Second, we discuss how local errors can be combined together to obtain equations for the parameterization of the whole surface. We show that isometry in the shape-operator corresponds to optimal *equidistributed* error.

Definitions Important local properties of a parameterization are captured by the *metric tensor*. Suppose a surface A is defined by a function $\mathbf{b} : \mathbf{R}^2 \to A \subset \mathbf{R}^3$ (Figure 4.3).

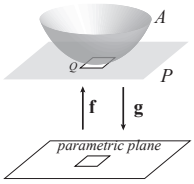


Figure 4.3: *Notation*

A *surface parameterization* is the inverse map from the surface to the plane $\mathbf{g} : A \to \mathbf{R}^2$. In our exposition, it is convenient to fix a surface point $\mathbf{p}$ and the tangent plane P at this point. *We assume that an orthonormal coordinate system is fixed in the tangent plane.* Unless otherwise noted, all tangent vectors are expressed in this coordinate system. We denote the 2 three-dimensional coordinate vectors of this system $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2]$. Then a two-dimensional vector $\mathbf{v}$ in the coordinate plane corresponds to three-dimensional vector $\mathbf{w} = \mathbf{D}\mathbf{v}$, and conversely, $\mathbf{v} = \mathbf{D}^T\mathbf{w}$. The differential $\nabla\mathbf{b}$ is a linear map from the parametric plane to P.

$\nabla\mathbf{b}$ defines the metric tensor in the parametric plane representing the metric of the surface. The dot product of two vectors in the parametric plane is given by the Euclidean dot product in the tangent plane P:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{b}} := (\nabla\mathbf{b}\mathbf{u}, \nabla\mathbf{b}\mathbf{v})_P = \mathbf{u}^T(\nabla\mathbf{b})^T\nabla\mathbf{b}\mathbf{v} \tag{4.1}$$

i.e. the metric tensor is given by the 2×2 symmetric matrix

$$M(\mathbf{b}) = (\nabla\mathbf{b})^T\nabla\mathbf{b} \tag{4.2}$$

For a vector $\mathbf{v} = \mathbf{q}_2 - \mathbf{q}_1$ in the parametric domain defined by a pair of close points $\mathbf{q}_1$ and $\mathbf{q}_2$, the quadratic form $\mathbf{v}^T M(\mathbf{b})\mathbf{v}$ is, in the limit, the squared length of the image of $\mathbf{v}$: $|\mathbf{b}(\mathbf{q}_2) - \mathbf{b}(\mathbf{q}_1)|^2$.

4.3.1 Normal approximation error

The local normal approximation error measure (e.g. [24]) is similar to the gradient error measure in finite elements [31]. This error corresponds more closely to

the perceived visual quality of an approximation, compared to, for example, the distance between points on the surface. For the purposes of defining a pointwise error, we consider an idealized setting: (1) The surface has well-defined curvature, with nonvanishing Gaussian curvature. (2) For a parameterization $\mathbf{g}$, we consider the approximation of the surface by a collection of small quads. Each quad Q is a parallelogram obtained by mapping a square Q_p of edge length h from a regular grid in the plane to the tangent plane of the surface at a point $\mathbf{g}^{-1}(\mathbf{c}) = \mathbf{b}(\mathbf{c})$, using $\nabla\mathbf{b}$. (3) We assume the surface to be well-approximated by a quadratic function over the tangent plane over each quad.

We define the error for a quad Q in the tangent plane P with normal $\mathbf{n}_Q$ as the square of the average of the deviation of the normal on the part of the surface $A(Q)$ projected to the quad Q along $\mathbf{n}_Q$.

$$E_Q^2 = \frac{1}{\text{Area}(Q)} \int_{A(Q)} \|\mathbf{n}_S(\mathbf{q}) - \mathbf{n}_Q\|^2 d\mathbf{q} \tag{4.3}$$

Next, we show how in the limit of small quads this error measure is related to the shape operator. Let (u, v) be local coordinates in the parametric plane centered at a point $g(\mathbf{p}_0)$, corresponding to $\mathbf{p}_0$ on the surface. The linear approximation to the surface normal over $A(Q)$ is $\mathbf{n}_0 + \mathbf{D}\nabla\mathbf{n}\,\mathbf{p}$, where $\mathbf{p}$ is the vector in the parametric plane in (u, v) coordinates. The $\nabla\mathbf{n}$ is the differential of the unit normal $\mathbf{n} = \mathbf{n}(u, v)$; as any directional derivative of the unit normal is perpendicular to it, it is in the tangent plane, so we assume $\nabla\mathbf{n}$ expressed in the tangent plane coordinates $\mathbf{D}$. Let $\mathbf{n}_0$ be the normal at $\mathbf{p}_0$; we assume that quad Q is tangent to the surface at $\mathbf{p}_0$, i.e. $\mathbf{n}_0 = \mathbf{n}_Q$. We express the shape operator S as a 2×2 matrix mapping vectors in the parametric plane to vectors in the tangent plane, in $\mathbf{D}$ coordinates. By definition of the shape operator, $S\nabla\mathbf{b} = -\nabla\mathbf{n}$. We rewrite

the expression for the normal as $\mathbf{n}_0 - \mathbf{D}S\nabla\mathbf{b}\mathbf{p}$. Then the pointwise squared error is given by

$$E_{pt}^2 = (\mathbf{n} - \mathbf{n}_0)^2 = \mathbf{p}^T \nabla\mathbf{b}^T S^T S \nabla\mathbf{b}\mathbf{p} \tag{4.4}$$

where we used frame orthonormality $\mathbf{D}^T\mathbf{D} = I$ to eliminate $\mathbf{D}$.

We assume that the surface is tangent to the quad at the center, (we need to expand the quad in two directions to make this true for an arbitrary tangent point), integrating E_{pt} over the quad Q in the tangent plane, we obtain

$$\begin{aligned} E_Q^2 &= \frac{1}{\text{Area}(Q)} \int_Q (\mathbf{n} - \mathbf{n}_0)^2 \det \nabla\mathbf{b}\, dudv \\ &= \frac{h^4}{12}\text{Tr}(\nabla\mathbf{b}^T S^T S \nabla\mathbf{b}) = \frac{h^4}{12}\text{Tr}(S^T S \nabla\mathbf{b}\nabla\mathbf{b}^T) \\ &= \frac{h^4}{12}\text{Tr}(S^2 M(\mathbf{g})^{-1}), \end{aligned} \tag{4.5}$$

where we use $\det \nabla\mathbf{b} = \text{Area}(Q)$ and $\nabla\mathbf{b} = \nabla\mathbf{g}^{-1}$.

We conclude that

$$E_Q^2 = \frac{h^4}{12}\text{Tr}\left(S^2 M(\mathbf{g})^{-1}\right), \tag{4.6}$$

approximates the integral of previously defined quad error up to $O(h^5)$ for each quad.

E_Q is highly similar to the gradient interpolation error for linear elements [98], yet there is an important distinction. As discussed in [98, 13], that error has a strong dependence on the shape of the element in the physical space (in our case, the shape of the approximating quad).

Specifically, if a square is mapped to the tangent plane using a map $\mathbf{b}$ with

metric S^{-2}, and the edges of the quad form a large angle in the tangent plane, the error, instead of being independent of curvature as suggested by (4.6) and (4.8) may be of order ah^2, where a is the ratio of max to min curvature; so the error distribution over the surface is clearly nonuniform. The fact that the quads we consider are tangent to the surface changes this behavior. However, in this work we are primarily concerned with the case when arbitrary anisotropy is not allowed. Rather we limit it to moderate values (typically no more than 5). We also note that under our assumptions, differing from those in e.g. [31], the error is the same for hyperbolic and elliptic points with identical principal curvatures. If the vertices of quads are expected to interpolate the surface, optimality conditions in the hyperbolic case are different.

Uniform-error parameterization and shape operator metric A natural approach to define an optimal parameterization given a pointwise local error is to require the error to have the same value ϵ over the whole surface, and minimize ϵ. This is however distinct from most common methods that define a global energy as an integral measure of a local error over the surface. Integrating the local error E_Q over the surface results in difficult-to-solve equations. Remarkably, *equalizing* the error in our case leads to a simple condition on the error, if one of the constraints of the problem is relaxed.

Denote $H = M(\mathbf{g})^{-1}$. Then the optimal uniform-error parametrization solves the following constrained problem:

$$\text{Minimize } \epsilon, \text{ subject to } \ \mathrm{Tr} S^2 H = \epsilon, \ \text{ and } \ H = M(\mathbf{g})^{-1} \text{ everywhere.} \qquad (4.7)$$

This problem is difficult to solve directly; instead, one can define an "ideal" metric H, solving the minimal uniform-error density optimization problem with H as a free variable, without the constraint $H = M(\mathbf{g})^{-1}$.

In addition, to the constraint above, we constrain the total area the image of the surface has in the parametric plane. This additional constraint is necessary as otherwise the trivial solution of the problem is to set H to zero. This constraint has the following form:

$$\int_A \det \nabla \mathbf{g} dA = \int_A \det H^{-\frac{1}{2}} dA.$$

Then the Lagrange function with multipliers λ and μ for the constrained minimization of ϵ is

$$\epsilon + \int_A \lambda \mathrm{Tr}(S^2 H) + \mu \det H^{-\frac{1}{2}} dA.$$

We compute the L^2-gradient of this expression with respect to H, using the identities $\partial \mathrm{Tr} A^T B / \partial A = B$, and $\partial \det A / \partial A = det A (A^{-1})^T$, and symmetry of H, we get

$$\lambda S^2 + \frac{1}{2} \mu H^{-1} \det H^{-\frac{3}{2}} = 0$$

i.e., $H = kS^{-2}$. Substituting into $\mathrm{Tr} S^2 H = \epsilon$, we get $k = \epsilon/2$, i.e., the scale factor is independent of H.

We conclude that the "ideal" parametrization has metric given by

$$M(\mathbf{g}) = cS^2, \tag{4.8}$$

with c independent of the point. In particular, the error bound is the same (under

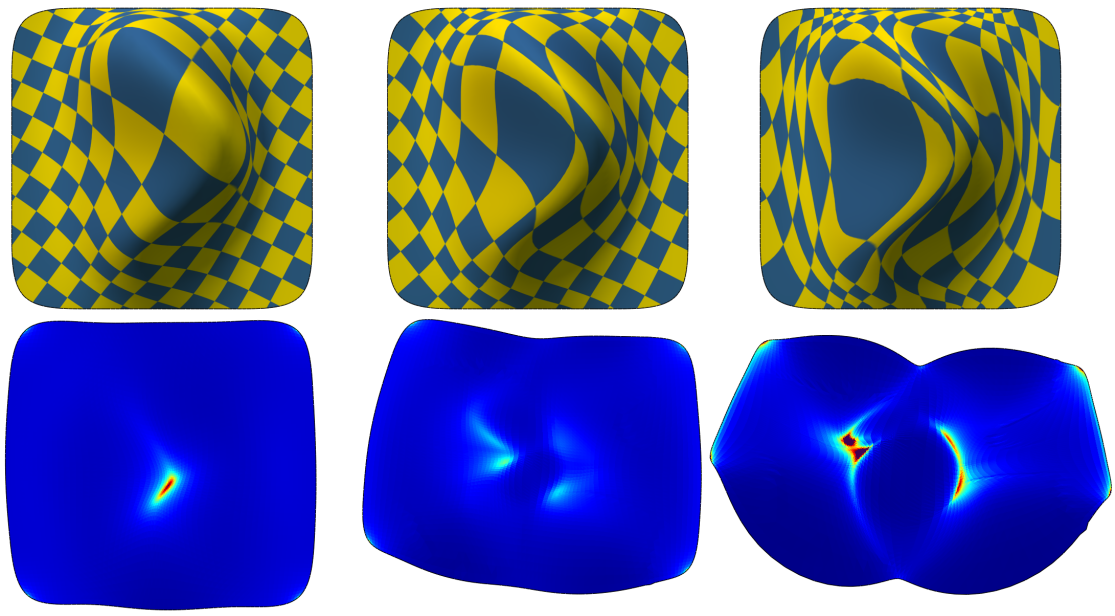


Figure 4.4: *Top left to right: a conformal map, a map with a small amount of anisotropy added ($\alpha = 3$), and large amount of anisotropy ($\alpha = 0.1$), where the metric tensor for the parameterization is $\alpha^2 I + S^2$. Bottom left to right: corresponding uv maps color-coded by inverse parametric triangle area.*

restrictive assumptions on approximating quads outlined below) for all parameterization differing by a rotation of the parametric plane (Figure 4.5).

In general, S^2 may have small or zero eigenvalues, and using it alone as a metric is not desirable, as this would result in infinitely long or thin quads. We can limit the possible quad aspect ratios by using $G(\alpha) = \alpha^2 I + S^2$ as the metric.

We conclude that *a uniform normal error parameterization g of a surface with nonzero Gaussian curvature has a metric tensor coinciding with the square of the shape operator up to a globally constant scale factor*, in other words, it is isometric in the metric defined by the shape operator.

Embeddings The Nash embedding theorems state that every Riemannian manifold can be isometrically embedded into a (sufficiently high-dimensional) Euclidean

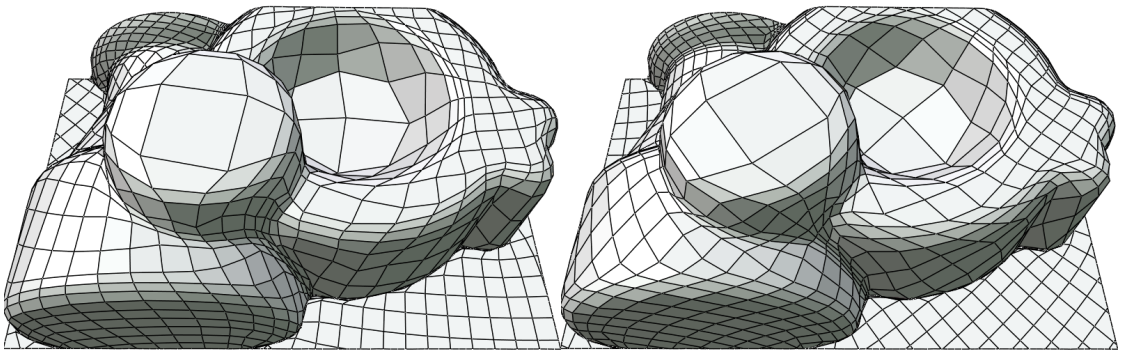


Figure 4.5: *The right model shows the result of rotating the anisotropic parameterization 45 degrees. Observe that the mesh elements remain stretched along the features.*

space, no matter what the metric might be.

The direct approach (cf. [110]) to obtain parametrizations with respect to modified metric is to derive the equations for the parametrization directly in terms of the metric tensor, and choose discretizations for the tensor and the parametric functions.

However, any surface equipped with an arbitrary metric can be embedded in a (usually higher-dimensional) Euclidean space in which the metric coincides with the induced metric (see Figure 4.6). This allows us to recast the problem of computing an isometric parameterization g of A with a given metric to that of computing an isometric parameterization $\bar{g}$ of the embedded surface $\bar{A}$ in the standard metric. Explicitly constructing such an embedding for a general tensor may be difficult. Fortunately, for the specific tensor we use a direct embedding construction is possible, and yields substantially better results as we demonstrate in the next sections.

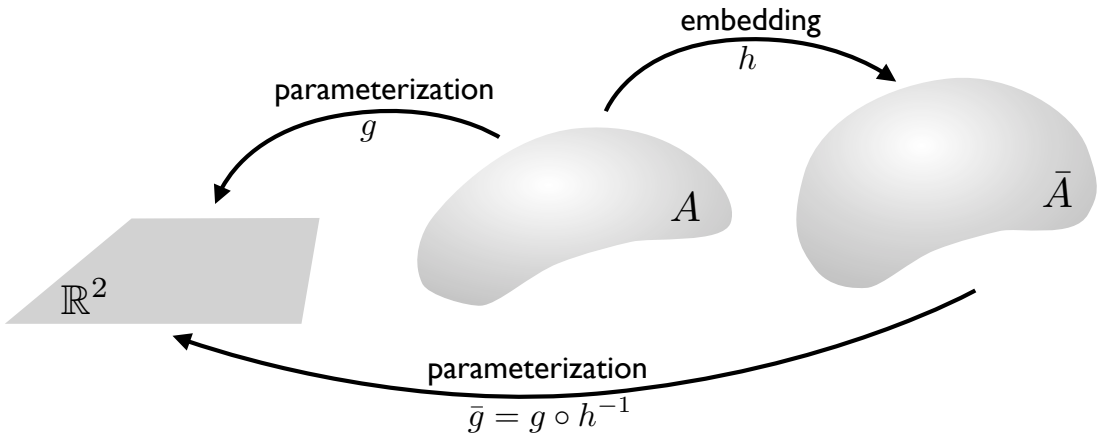


Figure 4.6: *An embedding h which has the desired metric G makes it possible to replace construction of g as close as possible to metric G with construction of $\bar{g}$ as close as possible to isometry.*

4.4 Anisotropic Parameterization

The observation of the previous section reduces the problem of finding an equidistributed error parameterization to that of finding an isometric parameterization in a different metric. Most currently used techniques can be regarded as approximations to the isometric parameterization in Euclidean metric, and can be naturally generalized if the shape-operator metric can be computed robustly and accurately, as discussed in Section 4.5.

We present anisotropic extensions for two parameterization techniques, harmonic, following [100] and feature-aligned, following [9]. As we have discussed in Section 4.2, the advantage of the former is more direct and explicit control over the number of singularities, while the latter yields parameterizations better aligned with mesh features, and typically closer to isometric.

We regard both harmonic and feature-aligned parameterizations as two types of efficient approximations to isometric maps (in the latter case with additional

condition of feature alignment) and demonstrate how these can be combined with anisotropy.

Isometric parameterization and harmonic maps Isometric parameterizations do not exist for surfaces with nonzero Gaussian curvature: at best, we can hope to approximate an isometric parameterization. Minimizing the deviation of the metric tensor from identity leads to nonlinear systems of equations for which no robust and efficient solvers are available. For this reason, many techniques replace direct isometry optimization with various types of factorizations.

Most commonly, harmonic maps, leading to linear systems, are used to minimize the angle distortion, subject to the boundary conditions; harmonic parameterizations often result in high area distortion. The idea of a number of recent methods [45, 35, 4, 100] is to use harmonic maps with singularities to define a parameterization, and to reduce the area distortion by introducing singularities and optimizing the singularity placement.

For the simplest case of a surface with disk topology, a harmonic map minimizes the Dirichlet energy

$$E = \int_A (\nabla u)^2 + (\nabla v)^2 dA \tag{4.9}$$

where u and v are parametric coordinates, and ∇ is the surface gradient. Computing u and v requires solving the linear Laplace-Beltrami equations $\Delta u = 0$ and $\Delta v = 0$.

Anisotropic harmonic maps In case of isometry, conformal maps are defined by the condition $M(\mathbf{g}) = cI$; they preserve the ratio of the singular values of the identity tensor I exactly. *Anisotropic conformal maps* satisfying $M(\mathbf{g}) = cG(\alpha)$

have similar behavior in the shape operator metric. Intuitively, an anisotropic conformal map takes a small circle in the parametric plane to an ellipse in the tangent plane of the surface, with axes aligned with the principal curvature directions, and its aspect ratio is determined by the ratio of principal curvatures. The effects of such a map, compared to a conformal map, are illustrated in Figure 4.4. The anisotropic harmonic map is a least-squares approximation to the anisotropic conformal map.

Isometric feature-aligned maps Feature-aligned maps [52, 9] use a feature cross-field, which locally can be regarded as a pair of orthogonal unit vectors $(\mathbf{u}, \mathbf{v})$ to define the target directions for the surface gradients of parametric coordinates ∇u and ∇v. If the desired gradient directions for coordinate functions are fixed, finding the as-isometric-as-possible parameterization can be formulated as a linear optimization problem minimizing misalignment with the feature field *and* deviation of the gradient magnitude from the unit length:

$$E = \int_A (\nabla u - \mathbf{u})^2 + (\nabla v - \mathbf{v})^2 dA \tag{4.10}$$

As $\mathbf{u}$ and $\mathbf{v}$ are orthogonal, perfect minimization of this energy corresponds to an isometric parameterization.

To obtain the *anisotropic feature-aligned parameterization*, we remap the feature field on the original surface A to be orthogonal in the new metric $M(\mathbf{g}) = cG(\alpha)$ and compute a feature-aligned least-squares isometric (w.r.t. this new metric) parameterization of A.

4.5 Discrete Metric

To complete our construction, it remains to define a discrete metric $G(\alpha)$ by assigning new lengths to each edge (4.12). While a variety of techniques can be used, we found that the results can be quite sensitive to the choice of technique.

There are two approaches to discretize the continuous theory described in the previous section:

- we can either work on the original surface A (Figure 4.6) and change the metric according to a discrete estimation of the shape operator,
- or we can construct the embedding $\tilde{A}$ explicitly and use the actual edge lengths as the discrete metric.

Using the metric $G(\alpha)$ directly The shape operator S can either be estimated per vertex [25, 83, 53] or per triangle [89, 43].

For example, to discretize the Laplace-Beltrami equation that needs to be solved to find the minimum of the Dirichlet energy (4.9), one can use piecewise-linear elements for the parametrization, and constant metric tensors defined per triangle (for vertex-based shape operator estimators, we can average the tensors at the three vertices).

To simplify the derivation, we assume that the embedding realization $\mathbf{h}$ of the metric $G(\alpha)$ is known (the equations we obtain will depend on the metric tensor only, so $\mathbf{h}$ is not used for discretization). This means that the differential of $\mathbf{h}$ satisfies $\nabla\mathbf{h}^T\nabla\mathbf{h} = G(\alpha)$. As before, we assume that an orthonormal frame is defined on the tangent planes of A and $\tilde{A}$, and all differentials are expressed in in these coordinates.

We express the parameterization differential $\nabla\bar{\mathbf{g}}$ on $\tilde{A}$ in terms of the parameterization differential of the original surface $\nabla\mathbf{g}$ as $\nabla\bar{\mathbf{g}} = \nabla\mathbf{g}\nabla\mathbf{h}^{-1}$ (see Figure 4.6). The Dirichlet energy density $\nabla u^2 + \nabla v^2$ can be written in matrix form for the map $g = (u, v)$ as $\operatorname{Tr}\nabla\mathbf{g}\nabla\mathbf{g}^T$. Then for the Dirichlet energy density of the map $\bar{\mathbf{g}}$ we have

$$\operatorname{Tr}\nabla\bar{\mathbf{g}}\nabla\bar{\mathbf{g}}^T = \operatorname{Tr}\nabla\mathbf{g}\nabla\mathbf{h}^{-1}(\nabla\mathbf{h}^{-1})^T\nabla\mathbf{g}^T = \operatorname{Tr}\nabla\mathbf{g}G(\alpha)^{-1}\nabla\mathbf{g}^T \tag{4.11}$$

The last equation can be expanded as $\nabla u G(\alpha)^{-1}\nabla u^T + \nabla v G(\alpha)^{-1}\nabla v^T$. (Note that we consider ∇u and ∇v row vectors, so the terms in the this expression are norms with respect to metric $G(\alpha)^{-1}$.)

Minimizing this energy leads to the generalized Laplacian equations for parametrization of the form $\operatorname{div}(G(\alpha)^{-1}\nabla g) = 0$, identical to the equations obtained in [110] with $C = G(\alpha)^{-1}$. Finite-element discretization of (4.11) is essentially identical to the Euclidean metric case, if $G(\alpha)$ is constant per triangle.

We can show that this discretization reduces to simply rescaling edge lengths per element using the metric tensor for this triangle, and computing the element matrix based on these new lengths.

As the metric tensors assigned to two adjacent triangles do not necessarily yield identical results for scaling of the common edge, each edge has two distinct scaled lengths; the examples in Section 4.7 demonstrate the effect of this mismatch.

We can instead enforce consistent edge lengths by averaging the two lengths obtained by using either per-vertex or per element shape operators. However, it proves to be fundamentally difficult to achieve a consistent discrete metric in this way which satisfies the triangle inequality for general meshes. The reason for this

can be seen from Figure 4.7. Suppose a triangle has bad alignment (long edge along principal direction with larger curvature). If the metric length of each edge β is determined as the average of two lengths $(\sqrt{\beta^T M_1 \beta} + \sqrt{\beta^T M_2 \beta})/2$, and the singular values of M_i are 1 and k^2, except M_3 for which they are 1 and $(1+a)k^2$, then for large l, a can be at most $4/(lk)$ before the triangle inequality is violated. So any averaging method is likely to fail even for small curvature variation: for $k = 10$ and $l = 10$, for instance, only 4% variation is possible across an edge.

Constructing an embedding An attractive alternative is to define an embedding of the surface such that the Euclidean metric on the surface for this embedding yields an approximation to the desired metric [13]. For the shape operator, the relevant embedding is the *Gauss map*: $f(\mathbf{p}) = \mathbf{n}(\mathbf{p}) \in \mathbf{R}^3$, because $S = \nabla \mathbf{n}$, i.e. S^2 is exactly the metric tensor of the Gauss map.

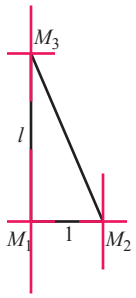


Figure 4.7: *A triangle with aspect ratio l, with 3 metric tensors M_i at its vertices.*

The shape operator satisfies $S\mathbf{v} = \nabla_{\mathbf{v}} \mathbf{n}$ for a tangent vector $\mathbf{v}$. Applied to edge vectors $\beta_{ij} = \mathbf{p}_i - \mathbf{p}_j$ on a triangle mesh (in a coordinate frame $\mathbf{D}$ on the triangle) expressed in a local orthonormal frame $\mathbf{D}$, it can be discretized by $S\beta_{ij} = \mathbf{D}^T(\mathbf{n}_j - \mathbf{n}_j) = \mathbf{D}^T \Delta \mathbf{n}_{ij}$. Then the squared shape operator metric S^2 is given by

$$\beta_{ij}^T S^2 \beta_{ij} = |\Delta \mathbf{n}_{ij}|^2$$

In other words, *the optimal metric edge length is simply the distance between endpoints of the edge in the Gauss map image of the mesh.* Note that so far, we only considered the embedding of the mesh into a two-dimensional sphere given by mapping each vertex v_i to its normal $\mathbf{n}_i$. This, however, is not sufficient to obtain the metric $G(\alpha) = \alpha^2 I + S^2$.

We therefore embed the mesh into $\mathbf{R}^6$, with a vertex v_i mapped to the point $(\alpha\mathbf{p}_i, \mathbf{n}_i)$, where α is a scale factor controlling the aspect ratios. In this case, the Euclidean metric in $\mathbf{R}^6$ yields

$$\begin{aligned} l_{ij}^2 &= (\alpha\mathbf{p}_i - \alpha\mathbf{p}_j)^2 + (\mathbf{n}_i - \mathbf{n}_j)^2 \\ &= \beta_{ij}^T(\alpha^2 I + S^2)\beta_{ij} = \beta_{ij}^T G \beta_{ij} \end{aligned} \tag{4.12}$$

i.e. it corresponds to a linear combination of isometry and normal error metrics. This defines the metric tensor G in terms of metric edge lengths l_{ij}. Since every mesh triangle is embedded in Euclidean space, the metric edge lengths satisfy the triangle inequality *by construction.*

Remapping the cross-fields Conceptually, parametrizing the surface $\tilde{A}$ embedded in six dimensions is not different from parametrizing a surface in three dimensions. One could remap the salient points on A to $\tilde{A}$, using the natural map $\mathbf{p} \to (\mathbf{p}, \mathbf{n})$, and then compute the feature cross-field directly on $\tilde{A}$. However, in practice we observe that the surface $\tilde{A}$ is much "bumpier" (Figure 4.8) i.e., has greater oscillations of the Gaussian curvature, due to higher variation of the shape operator included in the metric. The cross-field optimization procedure of [9] tends to place cones at Gaussian curvature extrema, which results in large numbers of

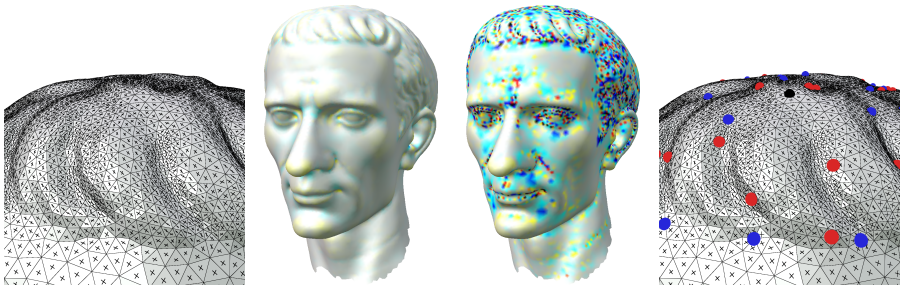


Figure 4.8: *Left: Standard Gaussian curvature distribution in* $\mathbf{R}^3$ *yields a cross-field with fewer singularities on the head of the Julius model. Right: Gaussian curvature of the six-dimensional embedding exhibits too many peaks and yields large clusters of singularities on the head. For visualization, the crossfield has been linearly mapped from the tangent space of the six dimensional manifold back to that of the three-dimensional manifold.*

cones. Instead, we perform cross field optimization in three dimensions as before, and remap the resulting cross field to $\tilde{A}$.

Say for a triangle T the linear transform from T to $\tilde{T}$ in some two-dimensional local coordinate systems is C, and the two orthogonal directions of the cross-field are $\mathbf{u}$ and $\mathbf{v} = \mathbf{u}^{\perp}$. First, we obtain a nonorthogonal cross field on the six-dimensional surface using vectors $\pm C\mathbf{u}$ and $\pm C\mathbf{v}$. However, to achieve near-isometry, the crossfield needs to be orthogonal. We consider normalized vectors $\mathbf{u}' = C\mathbf{u}/\|C\mathbf{u}\|$ and $\mathbf{v}' = C\mathbf{v}/\|C\mathbf{v}\|$, and compute an orthonormal pair $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$, such that $(\mathbf{u}' - \bar{\mathbf{u}})^2 + (\mathbf{v}' - \bar{\mathbf{v}})^2$ is minimized. We observe that if we combine $\mathbf{u}'$ and $\mathbf{v}'$ into a matrix Q, this is equivalent to finding the closest rotation matrix R to Q.

In the case of general matrices Q with entries q_{ij}, the angle α between the x axis and the direction of $\mathbf{u}$ is given by

$$\alpha = \arctan \frac{q_{21} - q_{12}}{q_{11} + q_{22}}$$

In the case of columns of unit length, this expression can be further simplified, and the resulting construction admits the following simple geometric interpretation. Consider bisectors of the two pairs of angles formed by $\mathbf{u}'$ and $\mathbf{v}'$. This bisectors are perpendicular (and in fact represent the rotation with the *largest* deviation from Q). The smallest deviation from Q is obtained by $\pi/4$ rotation.[1]

Controlling aspect ratios The parameter α can be used to control the maximal distortion either globally or locally. We found that the method is stable even for very small values of α, which allow quads to stretch a lot. The singular values of the tensor are $\alpha+\kappa_1^2$ and $\alpha+\kappa_2^2$, and the aspect ratio of the images of infinitesimal quads is $\sqrt{(\alpha+\kappa_1^2)/(\alpha+\kappa_2^2)}$,where we assume $|\kappa_1| > |\kappa_2|$. By choosing

$$\alpha = \sqrt{\frac{r \max\, \kappa^2 - \min\, \kappa^2}{r-1}} \tag{4.13}$$

globally, we can keep the aspect ratio below r. This is, however, a very conservative choice, which may eliminate the advantages of the method for surfaces with very nonuniform curvature.

4.6 Implementation

The idea of using a shape-operator metric can be integrated with any quadrangulation approach that only relies on the surface metric: the main change required is to modify the metric-dependent quantities to use (4.12); additionally, for methods using vector or tensor fields on surfaces, the fields need to be remapped as described in the previous section.

[1] In [78], it was observed that cross-fields are most naturally interpreted as symmetric 4-tensors; this yields an alternative approach to remapping fields.

The standard linear FEM discretization of the Laplace-Beltrami operator L involves the computation of cotangent weights. These weights can be derived using only edge lengths: for a triangle with sides a, b, c and angle $\gamma = \angle(a, b)$ we can compute

$$\cot(\gamma) = \frac{a\ b\ \cos(\gamma)}{a\ b\ \sin(\gamma)} = \frac{a\ b\ \cos(\gamma)}{2\ A}$$

We can then write the triangle area A as

$$A = \frac{1}{4}\sqrt{(a+b-c)(a-b+c)(-a+b+c)(a+b+c)}$$

and use the cosine rule $\cos(\gamma) = (a^2 + b^2 - c^2)/(2\ a\ b)$ to arrive at the final form:

$$\cot(\gamma) = \frac{(a^2+b^2-c^2)}{\sqrt{(a+b-c)(a-b+c)(-a+b+c)(a+b+c))}}$$

The details of both harmonic and feature-aligned mixed-integer parameterization can be found in [4, 100] and [52, 9] respectively. Here we present only a brief overview, to point out the aspect of algorithms that were modified.

For both methods, we start with computing a normal field (we use the robust method of [53]) and compute and smooth the scaling function α, followed by evaluating the metric lengths l_{ij} using (4.12). Once the global parameterization is computed, we generate a quad mesh by tracing parametric lines $u = i$, and $v = i$ where i is an integer, and determine quad vertex positions at integer u/v locations by linearly interpolating the original mesh vertices.

Anisotropic harmonic parameterization The main steps in this case are:

- iteratively optimize cone locations solving the Laplace equation for the scale

factors using metric edge length l_{ij}, or specify singularity locations manually;

- cut the mesh into a disk;
- quantize singularity indices to $k\pi/2$ (if not specified by hand), and singularity positions to integer locations;
- use harmonic parameterization with cotangent weights computed from l_{ij} to obtain a global mesh parameterization matching across the seams of the cut.

The main distinction compared to the original method is computing all metric quantities (cotangent weights in particular) using lengths obtained in (4.12).

Anisotropic feature-aligned parameterization In this case, we start with constructing the 3D feature cross-field:

- identify *salient* triangles and fix their cross-field directions;
- compute a global smooth feature cross-field using the quadratic mixed integer optimization of [9];
- detect singularities and cut the mesh into a disk so that the cut passes through all singularities;
- label globally consistent $\mathbf{u}$ and $\mathbf{v}$ directions on the cut mesh;
- minimize the fit energy for parameterization gradients to $\mathbf{u}$ and $\mathbf{v}$, enforcing constraints along the cuts and constraining the changes in coordinates across cuts to be integer.

The last step may be repeated multiple times with increasing weights in the energy to eliminate inverted triangles in the parameterization (stiffening).

For anisotropic feature-aligned parameterization, we remap the cross-fields on each triangle to the new metric to the using the approach described in Section 4.4. Each triangle T of the mesh for surface A corresponds to a triangle $\tilde{T}$ on the mesh for surface $\tilde{A}$, with rescaled edge lengths l_{ij}. The linear transformation C is uniquely determined by the affine transformation mapping $\tilde{T}$ to T.

4.7 Results

Comparison of different metric discretizations First, we demonstrate the robustness and feature sensitivity of our technique (Figure 4.9). We compare to an approach similar to that of [110] described in Section (4.5). This method results in significant smoothing of the metric, and, as a consequence, sharper features are not captured (Figure 4.9b.)

We attempt to set the scaled edge lengths again by averaging the lengths computed using per-vertex shape operators at two endpoints (Figure 4.9d,e). We observe that even for modest anisotropy, for a large number of facets the triangle inequality is violated; refining the mesh in most cases eliminates the triangle inequality violations, but a large number of iterations may be needed and resulting quadrangulation suffers from metric smoothing similar to the per-triangle case (Figure 4.9e).

Figure 4.5 shows the effects of rotating parametric axes for anisotropic harmonic parameterization of a shape which does not require adding cones or cuts. Note that the parameterization automatically squeezes quads to the lines of high curvature: the mesh elements appear to preserve their orientation, while rotating in the parametric domain.

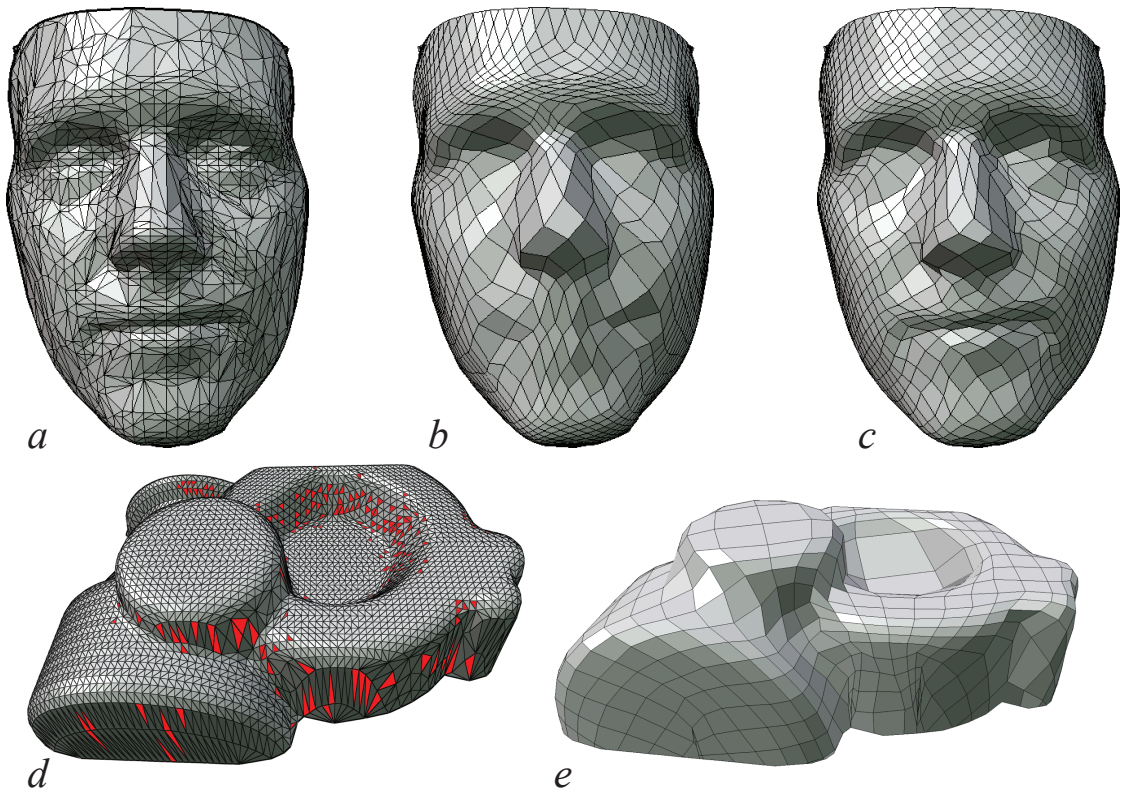
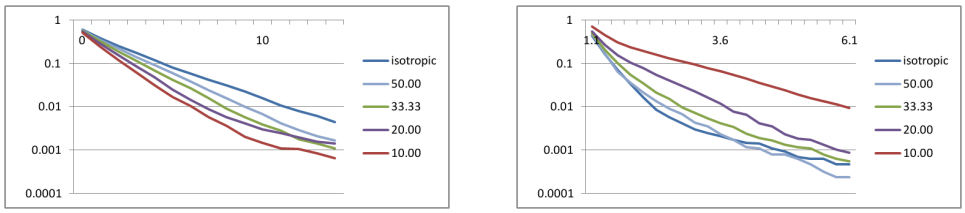


Figure 4.9: *Comparison of different ways of specifying metric lengths (a) the original face mesh; (b) face-tensor-based; (c) our method; (d) vertex-tensor averaging, triangles not satisfying metric inequality; (e) after refinement, metric inequality is satisfied, but quadrangulation misses some features.*

Impact of α Figure 4.10 demonstrates the effect of decreasing the scale factor α. Smaller α improve the normal error distribution by permitting quads with larger aspect ratios.

Comparisons with isotropic quadrangulation Our primary comparison is to the mixed-integer quadrangulation of [9] with no anisotropy. Figure 4.11, Figure 4.14, and Figure 4.15 show feature-aligned quadrangulations for a number of models.

For two models, we also compare to the periodic global parameterization (PGP)



Normal error Aspect ratio

Figure 4.10: *Impact of α on the normal approximation error and the quad aspect ratios observed on the Julius model shown in Figure 4.15, top-left. Left: In log scale for a given abscissa β (in % of max. normal error) the fraction of vertices with error above β. Right: In log scale the fraction of quads with aspect ratio above the abscissa.*

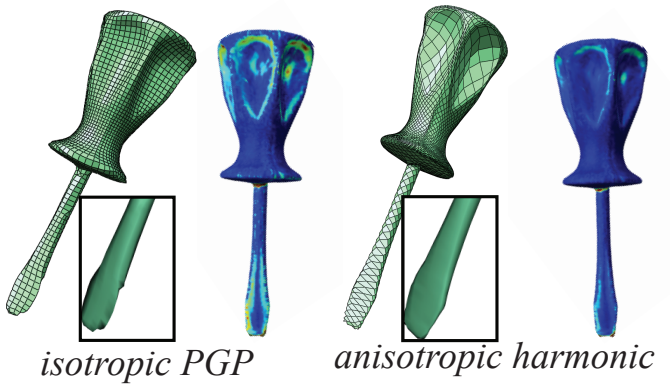


Figure 4.11: *Periodic global parameterization and (unaligned) harmonic anisotropic parameterization. Normal error distribution is shown in pseudocolor.*

(Figure 4.11 and Figure 4.12). We observe that under some conditions, *unaligned* anisotropic harmonic quadrangulation produces better results compared to aligned but isotropic quadrangulation.

As our main target application is approximating the original meshes with semiregular meshes with good visual quality, the ultimate criterion in this case (vs., for example, remeshing for finite element simulation) is the appearance of the resulting models. For this reason, we present smoothly shaded images of the

remeshed models in Figure 4.15, along with a pseudocolor rendering of the pointwise normal error (dark red corresponds to maximal error, dark blue to no error). We choose relatively coarse quadrangulations to make the errors more apparent. The number of facets in the original models, the number of quads as the fraction of the original model size, and the number of singularities are summarized in the following table.

model	facets	reduced to	cones
lion head	16674	17%	41
Julius	39168	28%	25
screwdriver	54300	3%	20
Stanford bunny	111364	3.5%	32
rocker arm	20088	8%	26
Omotondo	10000	25%	36
Max Planck	50790	35%	15

We emphasize that our technique aims to make the error distribution more uniform, *not* to minimize an integral error measure, hence it is difficult to quantify the relative quality of the result by a single number. In pseudocolor visualizations in Figure 4.15, one can observe greater uniformity in pointwise error. A consistent increase in uniformity is also confirmed by the plots of the pointwise error distribution: these plots show, for a given abscissa β, (in percent of the max possible error in normal), the fraction of vertices with error above β in log scale. Plots for anisotropic models are in red and for isotropic in blue. Higher slope corresponds to more even error distribution.

Sharp features As Figure 4.12 demonstrates, anisotropic harmonic quadrangulation can handle models with sharp features robustly, even with no feature alignment. The mesh for the fan disk model has only 8 singularities, i.e., the whole surface is mapped to the surface of the cube. Although for noise-free the quality of the result is inferior to the one that can be obtained by explicitly constraining the parameterization to be aligned with sharp edges as described in [9], for scanned meshes similar to the screwdriver example (Figure 4.15) when the edges of the mesh are not aligned with sharp features of the underlying geometry.

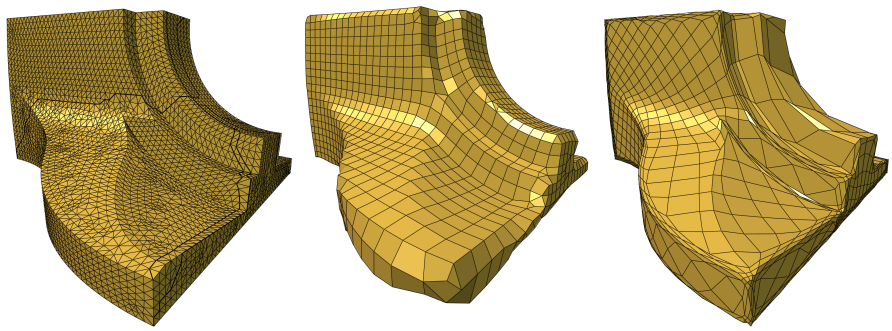


Figure 4.12: *Quadrangulation of a model with sharp features. From left to right: the original model, remeshing using PGP, and remeshing using anisotropic harmonic map. Both remeshed models retain approximately 20% of faces of the original model. 8 singularities are used for the anisotropic map, i.e., the model is parametrized over the surface of a cube.*

For certain types of models, it may be highly desirable to preserve sharp features. For feature-based parameterization, one can explicitly integrate perfectly sharp feature edges into the process, by forcing the field to be aligned with these edges and forcing one of parametric coordinates to be constant along these edges. This typically requires introducing a sufficient number of singular vertices.

In the context of our method, one can introduce parameterization discontinuities along sharp edges *without* introducing extraordinary vertices, at the expense

of introducing collapsed quads on a regular mesh. Figure 4.13 shows a case where sharp features were tagged along the connection of the model to the plane, and degenerate triangles were inserted along these creases. For normal calculations the creases were treated as internal boundaries.

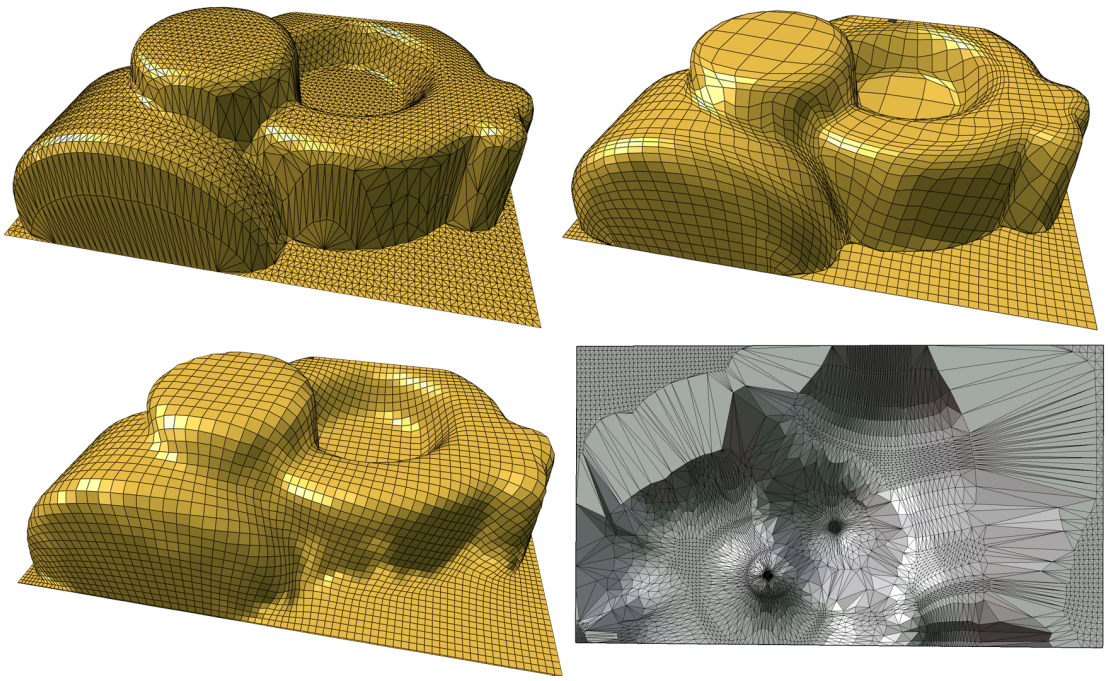


Figure 4.13: *Quadrangulation of a model with sharp features, with additional edges inserted at creases; no singularities are used. Top: the original model and our quadrangulation with 25% of faces. Bottom left: a harmonic map quadrangulation with the same number of faces; Bottom right: the original mesh in the parametric domain. Note the extremely stretched bands of triangles: these are thin triangles inserted along the sharp feature.*

4.8 Conclusion

The most appealing features of the proposed method are its robustness, its simplicity and its compatibility with a number of other approaches.

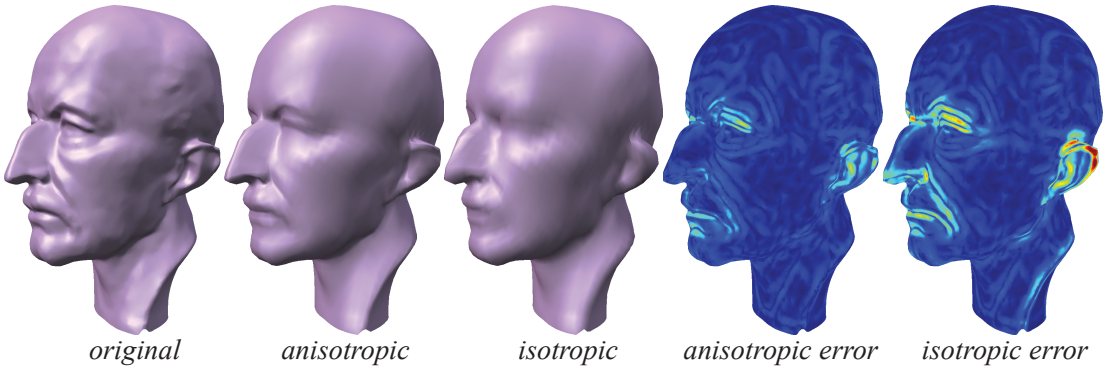


Figure 4.14: *Anisotropic quadrangulation preserves essential features even for extreme simplification (3%). The two rightmost images show the normal error distribution.*

As we generate quads with possibly large angles the resulting meshes are in general not suitable for solving equations on surfaces, unless the aspect ratio is limited to a moderate value; even with this restriction we can still expect a reduction in the number of quads needed for a given approximation quality (Figure 4.10).

While we do provide control over maximal aspect ratios, it is far from a complete solution, especially in cases of rapid edge length variation.

The method takes advantage of the possibility of discretizing the shape operator metric using a high-dimensional embedding. We would like to extend this to approximate embedding discretizations for arbitrary metric tensors.

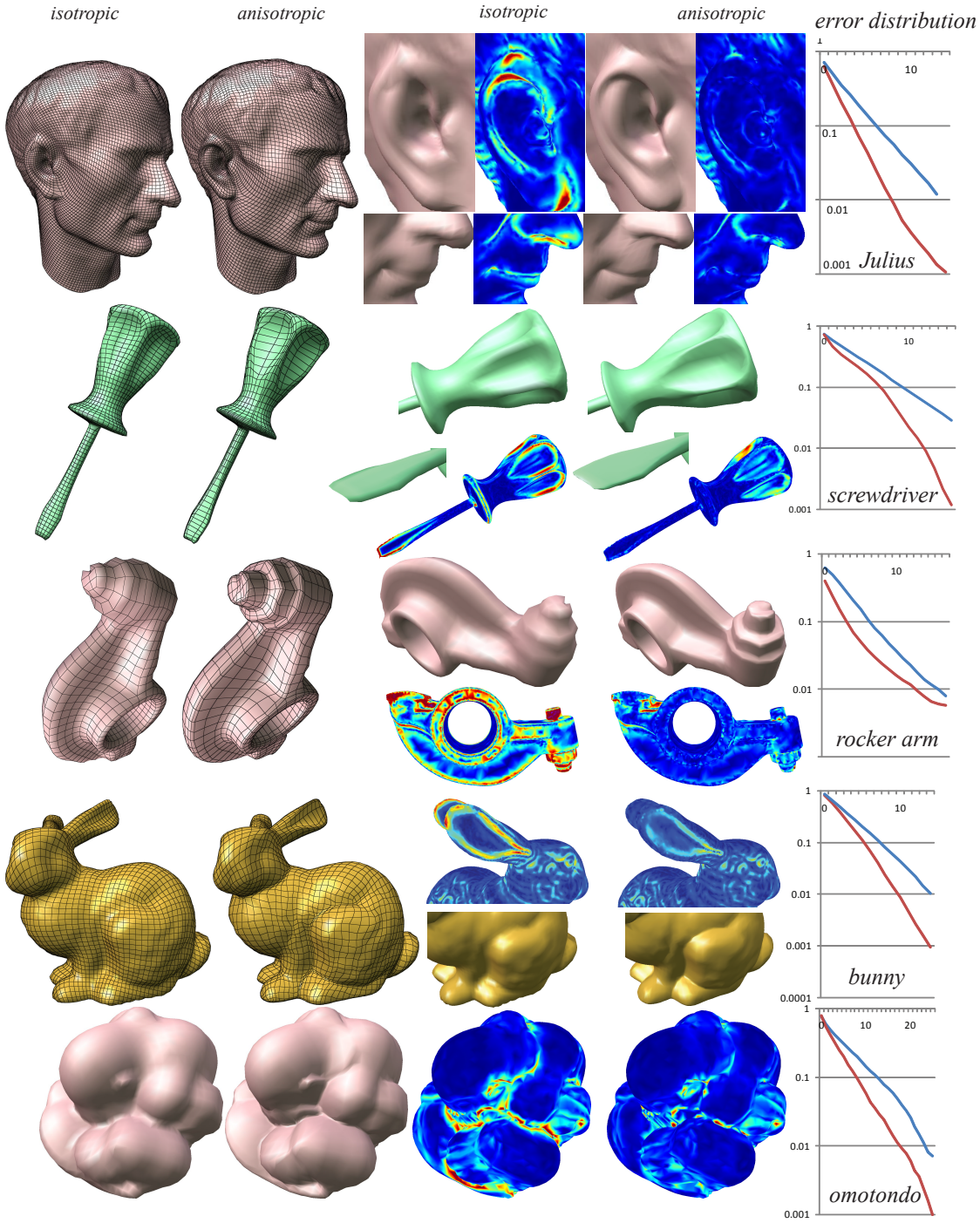


Figure 4.15: *Isotropic and anisotropic feature-aligned quadrangulations and error visualization. Error plots show in log scale for a given abscissa β (in % of max. normal error) the fraction of vertices with error above β.*

Chapter 5

Feature-Aligned T-meshes

This chapter describes a fully automatic pipeline to parameterize and remesh an arbitrary triangle input mesh into a quad mesh with T-joints. The T-mesh is used as the control mesh for a smooth T-spline surface, which is then least-squares-fit to the original input mesh.

This work was published as [A. Myles, N. Pietroni, D. Kovacs, and D. Zorin. Feature-aligned T-meshes. *ACM Transactions on Graphics (TOG)*, 29(4):117, 2010].

5.1 Introduction

Subdivision surfaces, surface splines, and related multiresolution and regularly sampled surface representations are far more compact and efficient than general meshes and simplify many geometric modeling and processing algorithms. Converting arbitrary meshes to this type of representations is difficult because of many conflicting requirements for such conversions.

Most regularly-sampled surface representations consist of patches forming a

domain mesh, with a regular pattern of samples for each patch. We focus on quadrilateral patches, as these are most commonly used.

Important requirements include (cf. [9]):

1. **Patch quality:** Patches should be well-shaped, with minimal skew and bounded aspect ratio.

2. **Approximation:** Each patch should approximate the original mesh well.

3. **Mesh complexity:** The domain mesh should have as few vertices as possible, while satisfying other constraints.

4. **Orientation and Alignment:** In areas with well-pronounced consistent curvature directions, patch parametric lines should follow the curvature; patch boundaries should be aligned with sharp features and smooth surface boundaries.

Existing techniques offer a tradeoff between *alignment* with features and isometry and the *number* of patches in the domain mesh. Techniques allowing to keep the number of patches small have only restricted forms of alignment control, while many recent algorithms with good alignment control often yield a larger number of patches in the domain mesh.

Quite often the tradeoff between the number of patches in the coarse mesh and alignment is fundamental, and not a feature of any specific algorithm: the maximal patch size in a local area is determined by the distance between nearby feature lines, which can be quite small. This local size restriction propagates globally (Figure 5.2) if the patch boundaries are aligned with feature lines, resulting in

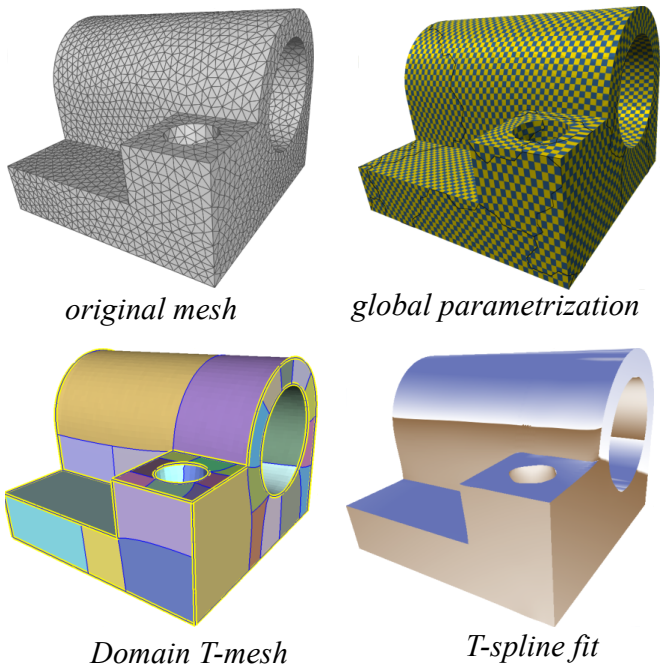


Figure 5.1: *Transforming the* `joint` *mesh into a T-spline surface.*

domain meshes with numbers of patches growing far larger than the complexity of the object suggests.

In this work, we propose an approach to constructing domain meshes consisting of small numbers of patches while maintaining good feature alignment. Our approach is based on using domain *T-meshes*, in which the intersection of two faces may be not the whole edge or vertex, but a part of an edge. T-meshes dramatically change the relation between the total number of patches needed and the local feature size making it possible to align patches with the field without restricting their size. Thus we generate one-to-two orders of magnitude fewer patches than the coarsest quadrangulations aligned to the same features.

We show that feature-aligned coarse T-meshes are naturally obtained using recently developed global parametrization techniques for quadrangulation [52, 9].

While T-meshes offer more flexibility, they also retain many desirable features of domain meshes with no T-joints (*conforming meshes*). Several high-order constructions (T-splines/NURCCs [94] and PT-splines [64, 33]) are available, with natural refinement structure allowing for multiresolution [93]. Adaptive structured meshes, a subset of T-meshes, are widely used in simulation; many local constructions developed for adaptive meshes, such as finite-volume and finite-element discretizations, can be transferred to general T-meshes ([2]).

Overview. Our approach consists of the following main components: (1) global parametrization construction; (2) constrained parametrization optimization, aiming to improve T-mesh structure while maintaining field alignment; (3) construction of an initial patch layout and its optimization, and optionally, reconstruction of a T-spline approximation to the original mesh.

We use a global parametrization method closely following [9], with some important changes discussed in Section 5.4, aiming to improve the quality of the feature cross-field guiding the parametrization.

The parametrization optimization step aims to reduce the number of T-joints in the domain mesh, by changing the global parametrization so that more singularities are on the same parametric lines. (Section 5.6).

We construct an initial patch layout with a number of patches within a constant factor from the minimal possible for a given number singularities. This initial layout may contain patches with bad approximation quality, unnecessary T-joints, and with unbalanced areas. We use a greedy constrained optimization strategy to move the patch boundaries while maintaining alignment to obtain the final layout (Section 5.5).

Finally, we fit a T-spline approximation to the surface, using the optimized domain T-mesh as the T-spline domain. The resulting surface can be either used directly (if the original mesh is well approximated by a piecewise-smooth surface), or used as the base for a displaced surface.

5.2 Related Work

The literature on parameterization, quadrangulation and conversion to high-order surfaces is quite extensive, and we survey only the most closely related work. Broader reviews can be found in [50, 97].

A number of methods [39, 59, 54, 70, 30, 29, 81, 102] use simplification techniques for constructing a conforming domain mesh. These techniques make it possible to obtain very coarse domain meshes, with good user control over the domain mesh size. While some degree of feature alignment is possible (cf. [59], [70]), it is limited by the difficulty of preserving features in simplification. Other methods use global harmonic or conformal parametrizations with singularities [45, 35, 105, 4, 100, 57]. While some of these methods offer a degree of control over the size and structure of the domain mesh (e.g., [35]), feature alignment is limited to determining positions of parametrization singularities. [51] describes an algorithm for adding alignment and orientation control to the parametrization, but the domain mesh is still constructed independently of geometry features.

Field-alignment techniques [85, 52, 9] adapt the parameterization to the shape by fitting the parametrization gradient to smoothed principal curvature directions, or more generally, to a smooth *cross-field* capturing surface features. The topological structure of the field (singularities and separating lines) indirectly determines

how fine the domain mesh can be. Reducing the number of singularities is often difficult without significant smoothing of the field. Furthermore, as the examples in Figure 5.2 show, even sparsely placed singularities do not guarantee that they are connected together in a way that allows constructing a coarse domain mesh. The method that we propose is based on field alignment. However, similar to simplification-based methods, we aim to produce coarse domain meshes, even for geometry with relatively complex features, while maintaining alignment.

In geometric modeling, *T-meshes* were considered primarily in the context of T-splines, T-NURCCs [94, 93], and PT-splines [64, 63, 33]. [62] demonstrated how to use periodic global parametrization (PGP) of [85] to fit T-spline surfaces to meshes. An important feature of PGP is its ability to introduce T-joints during the parametrization process. However, the complexity of the resulting domain mesh is still determined by the topological structure of the field, with significant smoothing required to make it simpler. [40] demonstrate how to use motorcycle graphs to partition a quad mesh into rectangular patches allowing T-joints, and prove bounds on the possible number of patches, but the quality of the patch layout cannot be controlled. [16] constructs rectangular geometry images (effectively, a T-mesh) by partitioning a mesh into approximately rectangular patches and parametrizing each on a rectangle, but the patches are not adapted to the geometry. [46] constructs a T-spline from an arbitrary mesh using global conformal parametrization. In this extreme case it is possible to have effectively a single-patch domain mesh for an arbitrary surface. As is the case with other harmonic methods, feature alignment control is limited to parametrization singularity placement.

The quality of the feature-aligned quadrangulation depends on the quality of feature detection, a difficult problem for many classes of meshes. A number of

techniques for defining and detecting feature lines were proposed: [77, 48, 109]. We use ridges and valleys computed from smoothed curvature values obtained using the robust estimation of [53] to determine which curvature directions should be considered salient (Section 5.4).

5.3 Field-aligned Quadrangulations

To motivate our approach, we consider constraints imposed on a conforming quadrangulation by field alignment. These considerations are not specific to any particular quadrangulation method. Recall that a cross-field (4-rosy field or 4-symmetry field) [47, 78, 86] is a quadruple of tangent vectors assigned to each surface point. A quadrangulation algorithm aligns the edges of the quad with the vectors of this field, so that no quad has singularities in the interior. As a consequence, a field singularity has to be a quad vertex, and there are quad edges following field integral lines starting at singularities (*separating lines*) (Figure 5.2, right). Chains of quadrangulation edges starting at singularities have to end at singularities, as we can always extend a chain past a regular vertex.

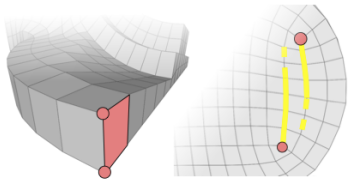


Figure 5.2: *Left: close singularities result in a strip of small quads. Right: a singularity close to separating line.*

The most fundamental restriction on the size of mesh patches is imposed by the *distance between field singularities*, as no quad can contain singularities inside.

In many cases, it is essential to place singularities close to each other for the field to follow features (Figure 5.2, left). If the quad size changes smoothly and is close to constant over the mesh, a local size restriction becomes global.

More generally, it is not essential for two singularities to be close to each other for the quad size to be constrained: *it is sufficient for two separating lines starting at these singularities to be close* (Figure 5.2, right). As there have to be quad edges along each separating line, at best, we can produce long and thin quads bounded by these lines.

Both cases can be either due to the structure of surface features (like singularities at two close corners) or be an artifact of constructing a smooth field from the salient feature lines. In the first case, a coarse mesh may be fundamentally incompatible with being aligned with features. In the second case, the feature field can be changed to improve the parametrization without changing the quality of the alignment.

By allowing T-joints in the domain mesh, we make it possible to switch to larger-size quads away from closely spaced singularities, and terminate chains of quad edges following separating lines early, removing both restrictions. Section 5.5 describes our algorithm for T-mesh construction. By detecting closely spaced separating lines and adding constraints to parametrization to snap them together, we reduce the number of nonessential T-joints in the resulting domain mesh (Section 5.6).

The need for rounding. For a general cross-field it is possible that integral lines starting at singularities may pass arbitrarily close to each other, or any integral line may pass arbitrarily close to itself. For example, if we tilt the natural parametric

lines on the torus so that the slope is irrational, any integral line of this field will be an infinite spiral around the torus.

To be able to obtain a valid quadrangulation, we need to ensure that the integral lines are closed or end at singularities. In [52, 9] this is ensured by a rounding procedure we discuss in greater detail in Section 5.4, which requires deviation of the quadrangulation lines from the original field. Creating larger quads for the domain mesh requires moving singularities further, resulting in non-aligned quadrangulations and higher distortion. T-meshes avoid the need for extreme rounding while still allowing to obtain large patches.

5.4 Feature-aligned Parametrization

The starting point for our T-mesh construction is the global parametrization of [9]. We briefly summarize the algorithm and the main differences in our version, as the structure of the algorithm is essential for introducing singularity constraints described in Section 5.6.

The algorithm computes a *global parametrization* of a mesh M, i.e., an assignment of planar (u, v) coordinates to each *triangle corner* (A triangle corner is a pair (f, w) where f is a triangle of the mesh and w is one of f's vertices). The mapping to the plane defined by these coordinates is one-to-one and orientation-preserving on each triangle. In addition, we assume that the whole mesh is mapped to a topological disk. More precisely, each vertex gets the same (u, v) coordinates in all incident triangles, excluding vertices along a *cut*, a connected graph C of mesh edges, such that $M \setminus C$ is topologically equivalent to a disk. The algorithm proceeds in several steps:

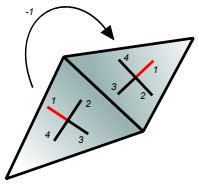


Figure 5.3: *Matchings between cross-fields in adjacent triangles.*

1. The shape operator is estimated on all triangles, and *salient* triangles are detected. For a salient triangle, principal curvature directions are likely to correspond to a feature; we discuss how salient triangles are detected below.

2. A cross-field, represented on each triangle T by the angle θ_T between one of four field directions and a reference edge of T, as well as integer *matchings* (Figure 5.3) on each edge, is optimized to minimize a measure of field smoothness. A matching determines corresponding directions on two triangles. Matching -1 means that direction 4 in triangle 2 corresponds to 1 in triangle 1. Matchings can be arbitrary (i.e., not necessarily mapping closest directions to each other). The directions are fixed on salient triangles, and the matchings are restricted to be integers. [9] describes an efficient greedy mixed-integer solver that we use to solve the optimization problem.

3. The cut C passing through all singularities of the field is computed.

4. The cross-field is made consistent: the angles representing the field are changed so that the matchings across all non-cut edges are zero. It is possible to achieve this if the cut passes through all singularities (we refer to [86, 9] for details).

5. As the matchings are all zero at non-cut edges, if we arbitrarily label one

of the directions of the field on a triangle T_o $\mathbf{u}_{T_0}$ (the target vector for ∇u), the label can be consistently propagated to all other facets. The 90-degree rotated vectors of the cross-field are labeled $\mathbf{v}_T$. The vectors $\mathbf{u}_T$ and $\mathbf{v}_T$ are the target values for the gradients of the parametric coordinates on the triangle T.

6. The parametrization is computed as a solution to the constrained minimization problem

$$\sum_{\text{triangles } T} \text{area}(T) \left(\|\nabla u - h\mathbf{u}_T\|^2 + \|\nabla v - h\mathbf{v}_T\|^2 \right) \to \min \tag{5.1}$$

where the scale factor h sets the correspondence between the length scale of the parametric domain and the surface.

The constraints imposed on (u, v) values correspond to transitions across seams: we want the match across seams to be the same as for the guiding cross-field: if the $\mathbf{u}_T$ direction across a seam is transformed to a $\mathbf{v}_T$ direction, then the parametric directions are transformed in the same way. More precisely, if two triangles T and T' share a cut edge e, with parametric positions of endpoint corners $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$ on one side of the cut, and p_1' and p_2' on the other side, these are related by

$$p_1' = R_e p_1 + t_e, \qquad p_2' = R_e p_2 + t_e$$

where R_e is a $k_e\pi/2$ rotation defined by the matching k_e of the cross-field on the edge, and t_e is an unknown translation.

For models with sharp features, apart from transition constraints, constraints

are imposed on parametric coordinates of vertices on sharp edges: for example, if the direction of a sharp edge is close to $\mathbf{u}_T$, its vertices are constrained to have the same v coordinates.

To make a conforming quadrangulation possible, [9] require that the translational parts of transition maps t_e to be integers. In addition, all parametrization singularities are required to be at integer locations. These constraints ensures that the cross field formed by ∇u and ∇v does not have infinite separating lines of the type discussed in Section 5.3. When the quadrangulation is generated by tracing the integer parametric lines on the surface, rounding ensures that the field singularities are at quad corners and that quad edges are continued seamlessly across cut edges of the mesh. Note that compared to "unrounded" global parametrization that minimizes (5.1) with no constraints, for large values of h rounding forces the parametric line directions further away from the cross-field directions.

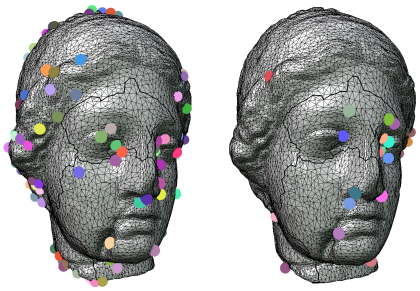
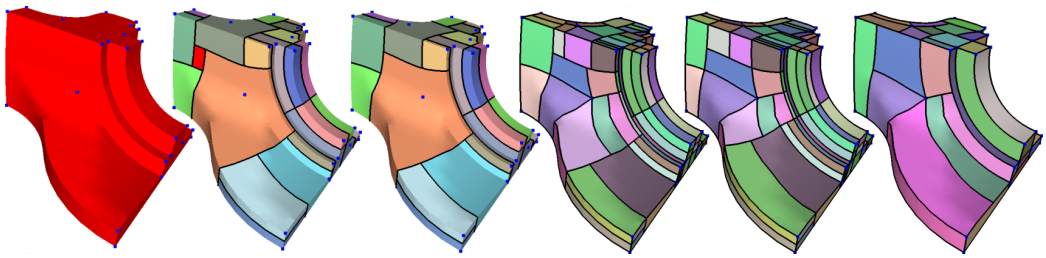


Figure 5.4: *Facet-based (left) vs. vertex-based cross-field optimization (right). For close salient fields, the vertex-based field optimization produces 34 singularities vs. 139 for facet-based.*

Quite often, the algorithm described above yields parametrizations with inverted triangles; as a result the parametrization has more singularities than the original field, and these singularities are not at integer locations. To solve this problem, following [9], constrained energy optimization is repeated several times



(a) (b) (c) (d) (e) (f)

Figure 5.5: *Main steps of the T-mesh patch layout construction. (a) An initial set of vertices are placed at singularities. (b) Cells with field-aligned edges are uniformly expanded from singularities, until no further expansion is possible. (c) Holes between cells are closed by adjusting cell boundaries. (d) Cells are split into quad patches. (e) T-joints are eliminated whenever possible by moving cell boundaries. (f) The number and shape of the cells are optimized to minimize an energy and satisfy the constraints.*

with gradually increasing (*stiffened*) weights assigned to the terms corresponding to triangles in areas with high parametric distortion.

Our algorithm differs from the algorithm of [9] in three main respects.

Salient feature detection. In [9], salient feature detection is based on thresholding per-triangle total curvature and shape operator anisotropy (the ratio of principal curvatures). Instead, following [77, 48] we use ridges and valleys, computed from a smoothed curvature field to identify salient facets and vertices. Ridges also require thresholding, and we use ridge strength as described in [77].

Field optimization. While we found that the triangle-based cross-field optimization produces good results in the case of meshes with well-shaped triangles, we also observed that a large number of singularities is often formed for surfaces with lower triangle quality. Instead of using tangent vectors at facets, we define a tangent plane at each vertex v, and a cross-field at v, with a reference direction for

the angle θ_v chosen to be the projection of an edge connected to v to the tangent plane. Similarly to the triangle cross-field case, we define matchings on edges, but this time these indicate correspondences between cross-fields at two incident vertices, rather than triangles. For subsequent parametrization, the cross-field at vertices is converted to a facet-based field by averaging the cosines and sines of quadruple the angles in a common reference frame; this is justified by the 4^{th}-order tensor formulation of [78].

Translation rounding. For T-mesh construction, rounding of translation variables and singularity positions is not fundamentally required, as the separating lines starting at singularities can be terminated at T-joints. However, we do perform a modest amount of rounding (on the order of triangle size of the original mesh) to make it possible to generate a fine-scale conforming quadrangulation that we use to implement our T-mesh construction algorithms, as explained in Section 5.5. As all singularity position changes are relatively small, we do not need to use the mixed-integer solver for setting singularity positions: they are all adjusted simultaneously as in [52].

5.5 Construction and Optimization of Domain T-meshes

Our goal is to construct a T-mesh with a small number of faces, with edges following the parametric lines of the global parametrization constructed in Section 5.4 and satisfying a number of quality constraints. In this section for simplicity we assume that the parametrization is fixed, and describe how the T-mesh can be con-

structed. In the next section we discuss a method for adjusting the parametrization to eliminate some non-essential T-joints.

The main steps of the construction are summarized in Figure 5.5. All steps in this section use the fine quadrangulation generated in the previous section.

5.5.1 Field-aligned T-meshes and operations on them

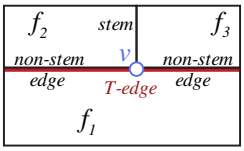


Figure 5.6: Notation. *Vertex v is a T-joint with respect to face f_1, but a corner for f_2 and f_3.*

To describe our algorithms we introduce the basic terminology for T-meshes. We consider quadrilateral T-meshes: Conceptually, every face of this mesh is a quad, but some of the quad edges may be split into several subedges by T-joints. Each vertex is one of three types: labeled *T-joint* (always of valence 3), *regular* (valence 4 in the interior, 3 on the boundary) or *extraordinary* (non-T-joint interior vertices of valence different from 4). For exactly one edge incident at a T-joint vertex v we mark its endpoints at v as *stem*, and the other two as *nonstem*. Thus, we say that a vertex is T-joint with respect to a face if it is incident to two non-stem edges comprising the face (Figure 5.6).

We distinguish between mesh edges and *T-edges*. Each face has exactly four *corner* vertices – those that are not T-joint with respect to the face; T-edges are unions of edges between two sequential corner vertices of a face. We assume that no face is glued to itself: the starting and ending vertices of a T-edge are always distinct.

Suppose we have a global parametrization defined. A *field-aligned T-mesh* is a mesh whose edges are curves on the surface satisfying two requirements: (a) each is a subset of a parametric line; (b) the field is nonsingular on each face, except possibly at the corners.

Field-aligned edge moves. The most basic operation used by our algorithms is *moving* a T-edge or edge along the field of parametric lines. For a parametrization with no cuts, this corresponds to simple translation of the edge in the parametric plane. For each endpoint w of an edge e not located at a singularity, there is a unique parametric line $\ell(w)$ passing through w and orthogonal to the line of the edge in the parametric domain. We define a *valid move* to be a repositioning of the T-edge on the surface so that the new endpoints w_1' and w_2' are on $\ell(w_1)$ and $\ell(w_2)$, and move by the same amount along these lines, and the edge remains aligned with a parametric line. Furthermore, no singularity is contained in the curvilinear rectangle with corners w_1, w_2, w_2' and w_1'.

Attached sets of T-edges. For a given T-edge e, we call a T-edge e' attached to e, if their intersection contains at least one edge. If E is a set of edges, then $A(E)$ is the set of all edges attached to edges in E. The *attached set* $A^c(e)$ (Figure 5.7) is the transitive closure of A for an edge e. If a T-edge e is moved while maintaining alignment with the parametrization, all T-edges in $A^c(e)$ need to be moved by the same amount if no new faces in the T-mesh are created.

In addition to simple attachment, we define *regular attachment*. An edge e' is regularly attached to e if it is attached to it, or it is on the same parametric line and shares a regular endpoint with e. Similarly, the regularly attached set of edges $RA^c(e)$ is defined as the transitive closure of the regular attachment relation.

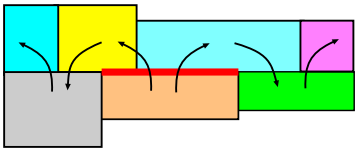


Figure 5.7: *The attached set of an edge (marked in red).*

Implementation. While all operations with T-edges can be implemented by tracing parametric lines on the surface, the implementation is considerably simplified by generating a fine (with quad size on the order of triangle size or smaller) quadrangulation of the original surface using the global parametrization. The downside of this approach is that it requires a moderate amount of rounding at the parametrization stage. In practice, we found that the quadrangulation can be chosen to be sufficiently fine for this not to lead to folds not removable by stiffening (see Section 5.7). In this case, the edge moves are no longer continuous but are discretized at the resolution of the fine quad mesh.

5.5.2 Initial T-mesh construction

As singularities of the field have to be vertices of the mesh, it is natural to start the construction using singularities as the initial set of vertices with no faces attached.

Singularity cell expansion. In the absence of parametric lines to align with, a natural and commonly used approach would be to use a Voronoi partition on the mesh to get a mesh of k-gons, and apply one step of Catmull-Clark subdivision. We mimic a simple Voronoi partition construction but force the edges of cell to be field-aligned. An initial curved k-gonal patch is defined by tracing integral lines of the parameterization gradient very close to singularity. (On the quad mesh, this

tracing reduces to following edges along the fine quadrangulation in 1-neighborhood of the singularity.) If the singularity index is $i/4$, $i \leq 2$ then $k = 4 - i$. We refer to k as singularity valence. As a result we get a set of field-aligned edges, which are moved away from the singularity at a constant speed in parametric units, until the T-edges of the cell become attached to other T-edges and cannot be moved without shrinking other cells, or reach the mesh boundary.[1]

Unlike Voronoi cells, the field-aligned cells need not fill the whole mesh, leaving some *hole quads* uncovered. It is possible to show (see the Electronic Appendix) that the local configuration of edges at any hole quad is of the type shown in Figure 5.8a.

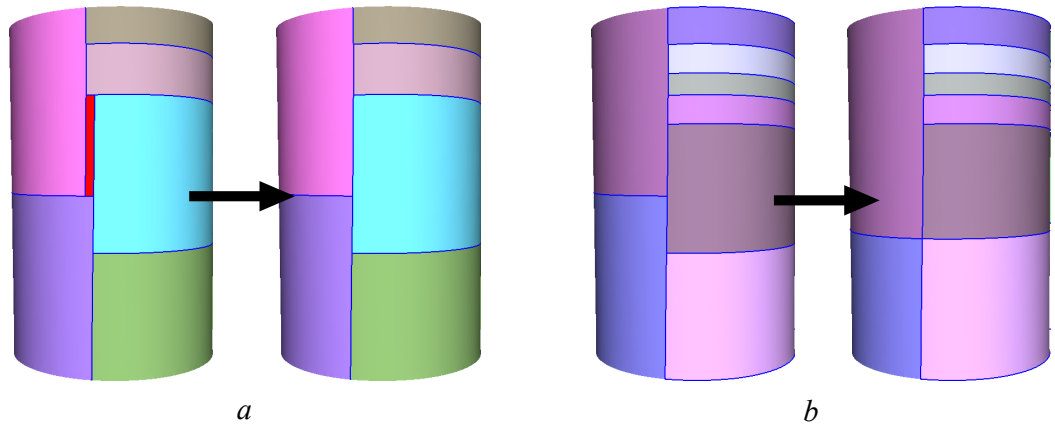


Figure 5.8: *(a) Closing a hole in the initial mesh; (b) regularization step.*

Hole closing. Most of these holes can be eliminated by moving some of the cell boundaries. For the hole-elimination step, the hole quads are sorted by size, with

[1]Note that while the shape of an initial cell of this type on a regular grid is identical to an L^∞ disk (i.e. a square), resulting cells are *not* Voronoi cells with respect to L^∞ metric: Our cells always have coordinate-aligned edges, while the L^∞ metric cells may have diagonal edges.

small holes first. A single hole-closing operation, shown in Figure 5.8a, proceeds as follows. First, a pair (e_1, e_2) of opposite edges bounding the hole is selected. We select the pair of edges along the longer parametric dimension of the hole quad, with parametric distance d between them. The attachment sets $A^c(e_1)$ and $A^c(e_2)$ are either disjoint or coincide, as they are defined as transitive closures. In the latter case, we say that this pair of edges fails *the loop condition* (Figure 5.9), and consider the other pair of edges of the hole quad.

If the attachment sets are distinct, we determine the maximal valid move distances d_1^{max} and d_2^{max} for $A^c(e_1)$ and $A^c(e_2)$, in the direction towards the interior of the quad, defined as minima of the valid move distances of the edges in each set. If $d_1^{max} + d_2^{max} \geq d$, we set $d_1 = \min(d_1^{max}, d/2)$, and $d_2 = d - d_1$, and move the attached sets $A^c(e_1)$ and $A^c(e_2)$ to close the hole. If $d_1^{max} + d_2^{max} < d$, no valid move in this direction closes the hole, and we consider the opposite pair of edges. If neither pair can be used, we create an additional cell to fill the hole. The result of the initial T-mesh construction is a parametrization-aligned T-mesh, but with k-gonal faces.

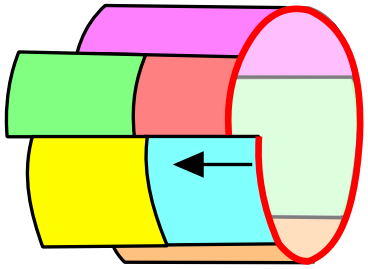


Figure 5.9: *Loop condition.*

Once all cells are expanded to the maximal extent, and all holes are filled, the

k-gonal cells are split into parametrization-aligned quads, by tracing k parametric lines from the central singularity to the boundaries of cells. All three-valent vertices of the resulting mesh with two incident edges along the same parametric line become T-joints. If there are no holes in the mesh, the total number of cells in the mesh we obtain is $\sum_{\text{singularity } v}(4-4i_v)$, where i_v is the index of the singularity at v. The number of holes is bounded from above by the same number, as there is at most one hole at each k-gonal cell corner.

T-mesh regularization. The regularization step reduces the number of T-joints in a mesh by an operation similar to closing holes, collapsing some edges separating two T-joints (Figure 5.8b). We find all edges in a mesh with two non-stem endpoints w_1 and w_2 at T-joints, such that the stem edges at these T-joints are on opposite sides of the edge. These edges are sorted by length, with shorter edges eliminated first. Let e_1 and e_2 be the stem edges at w_1 and w_2. Then we apply the same procedure as for the hole filling to the pair (e_1, e_2) except we use the regularly attached sets $RA^c(e_1)$ and $RA^c(e_2)$ instead of the attached sets, to avoid creating new T-joints in the process of removing old. A similar $RA^c(e_1) \neq RA^c(e_2)$ needs to be checked to verify validity of the move.

5.5.3 T-mesh optimization

The operations used in construction of the initial mesh are pure connectivity operations, not taking into account any quality criteria, other than reducing the number of T-joints at the regularization step. As the next step we optimize the patch layout. Our overall goal is to create the largest possible patches, while maintaining good patch quality. Our approach is similar to mesh simplification and improve-

ment techniques: we define a set of operations on the set of faces of the T-mesh preserving the validity of mesh, and define an energy we want to minimize by a sequence of these operations, while satisfying a set of constraints.

Energy and constraints. We choose an energy favoring larger well-shaped patch sizes; for an individual patch, we use its perimeter-area ratio to balance the priorities of avoiding small patches while favoring square-shaped patches. Since we quadrangulate finely to minimize distortion due to rounding, we approximate actual lengths by parametric lengths. The energy of individual patches needs to be combined in a global energy function; we found that the ℓ_1 norm of the vector of parametric perimeter-area ratios yields the best results compared to ℓ_2 and ℓ_∞:

$$E_{area} = \sum_P \frac{1}{\mathcal{L}(P)} + \frac{1}{\mathcal{W}(P)} = \sum_P \frac{\mathcal{P}(P)}{2\mathcal{A}(P)}$$

where the summation is over patches P; and $\mathcal{L}$, $\mathcal{W}$, $\mathcal{P}$, and $\mathcal{A}$ are the parametric length, width, perimeter, and area operators, respectively. The constraints are as important as the energy itself: the T-mesh is likely to be useful in a much more limited context with no constraints on patch quality. The choice of constraints depends on the goal of constructing the T-mesh. If the primary goal is to partition the surface into a small number of logically rectangular domains, similarly to [16] or [40] the optimization can be done without constraints. If, however, we would like to use the T-mesh as a coarse control mesh for a high-order or multiresolution surface representation, controlling the approximation error and patch aspect ratios are likely to play an important role.

This leads us to our two main constraints (additional optional constraints, e.g. maximal patch size can be imposed if desired).

Geometric approximation constraint. For each face of the T-mesh, we estimate how well the surface can be approximated by a smooth piecewise polynomial surface on this patch. While we could use a globally smooth surface approximation directly (T-NURCCs or PT-splines), determining the precise approximation error is expensive as it requires solving a global linear system. Instead, we use a simple Bezier curve network fit to approximate the per-patch error locally and efficiently. For each face, we fit 8 Bezier curves (4 aligned with each parametric direction) to surface points uniformly sampled in the parametric domain (we use an 8×8 grid of samples). Each Bezier curve interpolates the boundary samples, so the fit reduces to solving a 2×2 system of linear equations per curve. We compute the approximation to L^2-norm of the error ϵ_P of the fit, as the sum

$$\epsilon_P^2 \approx \frac{1}{2}\frac{\mathcal{A}(P)}{n^2}\sum_{i=0}^{3}\sum_{j=0}^{7}\left(\mathbf{b}_i(hj) - \mathbf{p}_{2i,j}\right)^2 + \sum_{j=0}^{3}\sum_{i=0}^{7}\left(\mathbf{b}_j(hi) - \mathbf{p}_{i,2j}\right)^2$$

where $\mathbf{p}_{ij}$ are the samples, and $\mathbf{b}_i(t)$ and $\mathbf{b}_j(s)$ are the Bezier curves along two parametric directions, with parameters t and s in the range $[0, 1]$ on the face, and $h = 1/7$.

Patch aspect ratio constraint. While in many cases constraining geometric error results in automatic restrictions on the aspect ratio, the patches on nearly cylindrical areas of the surface may become very long. In other cases, the energy may favor creating very thin and narrow patches in flat areas to increase the area of nearby patches. To limit these effects we impose an additional constraint on the patch aspect ratio.

T-mesh modification operations. We use three operations for T-mesh modification: two connectivity-modifying, and one only affecting the patch size, two of which have a single length parameter.

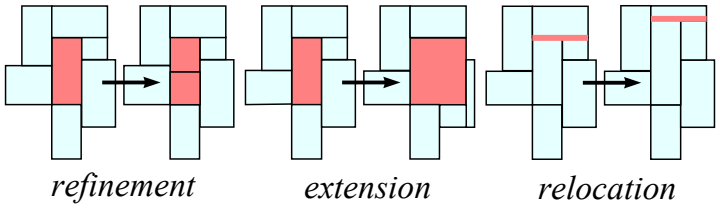


Figure 5.10: *Refinement, extension, and relocation operations.*

The *Refinement* operation acts on a (T-edge,face) pair (e, f), splitting the face f into equal halves by inserting a new edge along the parametric direction perpendicular to e, generally creating two T-joints (Figure 5.10). Refinement always decreases the patch size, and increases the number of patches.

The *Extension* operation acts on a T-edge/face pair (e, f), extending the face f across the edge e into adjacent faces. It increases the size of one patch, while other patches shrink, or even eliminated. A face f (Figure 5.10) is extended to the maximal length possible across the T-edge e, so that we do not modify faces with no T-edges attached to the T-edge e. Depending on local connectivity, it may increase or decrease the number of patches, although it most commonly decreases the energy most when it reduces to a merge of several faces. We define this operation in a more general way, as we found that in some cases this less constrained operation produces better quality T-meshes.

The *Relocation* operation acts on an edge e. The attachment set $RA^c(e)$ is found and all edges in the set are moved by the same distance a, positive or negative, not exceeding the maximally valid distance in this direction.

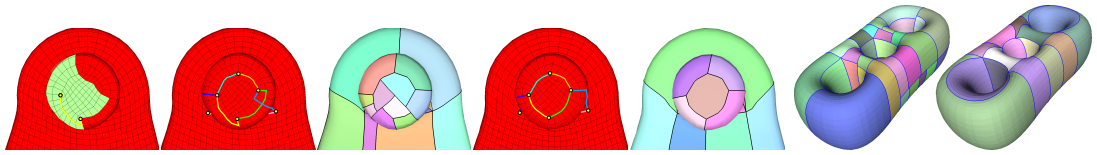


Figure 5.11: *Singularity alignment process: (a): detecting a mismatch between adjacent singularities; (b): a part of the singularity adjacency graph (observe "near-misses"); (c): T-mesh before alignment; (d): singularities and separating lines after alignment; (e): T-mesh after alignment; (f):* `holes3` *before alignment; (g):* `holes3` *after alignment with no T-joints.*

Complete optimization algorithm. We impose the constraints using a multiplicative penalty method (cf. [106]) by combining them with the energy function:

$$E_{total} = \sum_P \frac{\mathcal{P}}{2\mathcal{A}(P)} \left(1 + w(\alpha_P/\alpha_0 - 1)\right) \left(1 + w(\epsilon_P/\epsilon_0 - 1)\right)$$

where α_P is the parametric domain aspect ratio of patch P, ϵ_P is the geometric error estimate described above, α_0 and ϵ_0 are user-specified upper bounds for the constraints. The function $w(t)$ is chosen to be zero if $t < 0$, and increases rapidly for $t > 0$, we use t^3. Multiplicative penalty functions are similar to the more common additive penalties (taking log of the energy converts them to additive) but have the advantage of not requiring to choose a proper scale factor.

The complete optimization algorithm proceeds as follows.

For each (edge,face) pair of the T-mesh, we consider *Refinement* and *Extension* operations, and for each edge the *Relocation* operation. For each parametrized operation (*Extension* and *Relocation*) we determine the parameter range corresponding to valid moves, and find the parameter value corresponding to the maximal decrease in energy. Among all operations, we choose the operation that results in the maximal decrease in energy, and perform this operation. The process is

iterated until the energy cannot be further decreased. Since the energy decreases at every step, the algorithm always terminates.

Reevaluating all possible operations at every iteration would be prohibitively expensive. Instead, we update the invalidated operations incrementally.

We assign a timestamp to all edges and faces of the mesh (initially zero). In the beginning for all faces and edges, we generate all potential operations, compute the energy change resulting from each operation, and place the operations on the priority queue, with the energy decrease as the priority (energy-increasing operations are discarded). All operations are also given a timestep zero.

Then we repeatedly perform the operation with highest priority, unless the faces and edges it affects have a later timestamp than the operation, in which case it is discarded. All edges and faces modified as a result of the operation get a new timestamp t, and a new set of operations is generated for these facets and edges and pushed on the priority queue with timestamp t, if they decrease the energy.

5.6 Singularity Alignment

The T-mesh construction algorithm of Section 5.5 is limited by the fact that the parametrization is fixed, and patch boundaries stay aligned with parametric lines. However, only important feature lines (sharp edges in particular) are fixed by the geometry. The cross-field and the parametric lines of the global parameterization away from features can be modified to improve the quality of the mesh, and decrease the number of T-joints. The problem of separating lines passing within short distance of each other, (discussed in Section 5.3; Figure 5.2, right), can be reduced by adjusting the global parameterization.

Overview. We observe that ideally we want the separating lines starting at a singularity to terminate at a nearby singularity, if it passes sufficiently close to it. The algorithm that we describe in this section identifies close singularities and detects "near-misses", constructs the *singularity adjacency graph*, and solves for a new parametrization with additional constraints that force perfect alignment for identified pairs; the steps of the process are shown in Figure 5.11.

Defining a singularity adjacency graph. We observe that the first step of the initial T-mesh construction algorithm, with minor modifications, provides a mechanism for detecting "near-misses" of field separating lines. As before, we expand a cell from each singularity. However, instead of growing all cells at once, we expand the cell, until each edge reaches an adjacent singularity, a boundary, or another edge of the same cell. Singularities on the boundary of the maximally expanded cell for a singularity c are considered adjacent to c. This relation is not necessarily reciprocal. We construct a singularity adjacency graph connecting by edges all adjacent singularities. Example graphs are shown in Figure 5.11 and 5.18. Each edge (c^1, c^2) is annotated with a separating line *mismatch* (the parametric length from the singularity c^2 to the closest parametric line starting at c^1) and *distance* (the parametric L^∞ distance between the singularities) illustrated in Figure 5.12.

We want to change the parametrization so that a parametric line starting at c^1 passes through c^2 i.e., so that the mismatch becomes zero at a maximal number of edges of the adjacency graph.

Constructing constraints. If a mesh can be parametrized without seams, the requirement of singularity alignment easily translates into a constraint on parametriza-

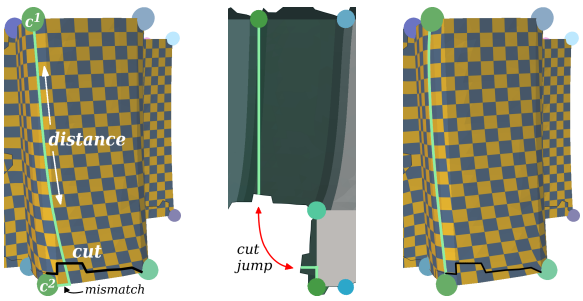


Figure 5.12: *A path connecting two misaligned singularities before and after alignment.*

tion: two singularities should be on the same parametric line, i.e. share the same u or v value. The constrained optimization framework of [9] that we are using makes adding such constraints easy.

For parametrizations with cuts, the situation is more complicated. A parametric line on the surface undergoes a jump to a different point and direction in the parametric space when it crosses a cut (Figure 5.12). While the rotation is entirely determined by the cross-field to which the parametrization is aligned, the positional jump depends on the parametrization itself. When we add a constraint on singularity coordinates, the parametrization may change, changing the jumps at cut edges. The resulting constraint will depend not only on the pair of singularities (c^1,c^2), but also on the (variable) translational parts of the transforms at the cut edges we cross, t_e in Section 5.4. E.g., if the cut is crossed once, and the crossing is at a cut edge e with associated transform $p' = R_e p + t_e$, where $p = (u, v)$ is a parametric point, then the constraint is $(R_e p^1 + t_e)_u = p^2_u$, if the aligned parameter line at c^2 is along the u coordinate direction, and p^1 and p^2 are parametric positions of c^1 and c^2), and the subscript u means taking the u coordinate.

In the general case, consider a path crossing cut edges e_i $i = 0, \ldots m$ between c^1

and c^2. Assuming the final direction of the path is u, then the complete constraint has the form

$$\left((\Pi_{i=0}^{m} R_{m-i}) p^1 + \sum_{i=1}^{m} \Pi_{j=i}^{m} R_{m-j} t_i \right)_u = p_u^2$$

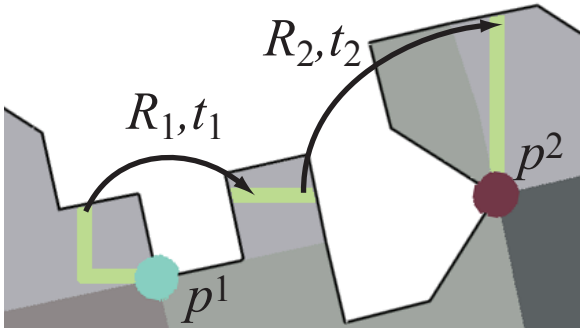


Figure 5.13: *Forming a constraint for a pair of singularities.*

we still have a single linear constraint, but involving a larger number of variables t_i, p^1 and p^2 (Figure 5.13).

We observe that the form of the constraint depends on the choice of path between singularities. One can show that for two paths P_1 and P_2 connecting two singularities and such that the loop formed by P_1 and P_2 does not enclose any singularities and encloses a topological disk, the constraints are equivalent This allows us to choose a path between singularities consisting of two segments of parametric lines, one passing through c_1 and the other through c_2, tracing the boundary of the rectangle with c_1 and c_2 at diagonal corners.

Filtering singularity constraints. The singularity constraints can be redundant. As these are homogeneous constraints with zero right-hand side, they cannot be incompatible, so we eliminate the redundant ones with Gaussian elimination.

We choose a threshold for the minimal aspect ratio of the rectangle for an

adjacent pair of singularities, and remove all constraints exceeding this threshold (we set it in the range 5-10); choosing this threshold high enough ensures that the field does not deviate too far from soft creases. A singularity can satisfy only a single constraint along each outgoing parametric line, so if several constraints correspond to a direction, we choose the one with the closest singularity (smallest parametric L^∞ distance). The singularity constraints can interact with sharp edge constraints: if a singularity has a sharp edge constraint in a particular parametric direction, we remove singularity constraints in this direction.

5.7 Results and Comparisons

Our algorithm is automatic once various thresholds are set. Other than mixed integer quadrangulation parameters, we use (1) aspect ratio threshold for cone alignment pair selection, (2) desired max aspect ratio of T-patches, and (3) max geometric approximation error. Optionally, manual adjustments can be made to singularity placements as described in [9]. This was done for the `maxplanck` head to make the placement of singularities more symmetric.

We evaluate the results of our algorithm in several ways. The main criterion is the number of patches in the final mesh subject to the constraints on aspect ratio and geometric approximation quality.

For several meshes, we compare domain T-meshes we obtain to the minimal number of patches we could obtain in a conforming mesh using mixed-integer quadrangulation, with the same feature field and the number of stiffening iterations bounded by 40. We observe that in most cases, especially involving alignment, the main restriction on quad size is due to the lack of robustness with respect to large

rounding. Numerical data on the number of faces in different meshes are presented in Table 5.1, and Figures 5.16 and 5.15 show the results for several meshes. We emphasize that these coarsest meshes are not necessarily suitable as control meshes for a T-spline or other fit. Rather, these are useful as seamlessly and smoothly aligned rectangular geometry images, for texturing, or solving equations on the surface.

Periodic Global Parametrization (PGP) [85], does not have a target quad size limitation due to foldovers PGP avoids the need for rounding, and the need for stiffening, but as the target quad size becomes coarser, the quality and alignment of the mesh rapidly deteriorates. Figure 5.14 shows a parametrization obtained using PGP with approximately the same number of quads as our T-mesh shown in Figure 5.16. (Caveat: the best effort was made to smooth the field for the PGP parametrization, but it is unclear if the quality of the field matched the quality of the cross-field we used for our result.)

For high-order approximation, the geometric error constraint has to be taken into account. Figure 5.17 shows the effect of decreasing the geometric error constraint for one model and decreasing the aspect ratio constraint.

In Figure 5.18 we show the full singularity adjacency graph and its pruning. Singularity alignment is highly useful for eliminating most of the near misses in matches. At the same time, we observe that one cannot expect to eliminate most of the T-joints in the mesh using this method due to two reasons. First, we filter the singularity alignment constraints by feasibility and by the mismatch-to-distance ratio, to avoid deviation from the feature field. As a result, the total number of valid alignments we enforce is relatively small, compared to the total number of T-joints. Second, the resulting mesh is optimized using our general T-mesh

					unaligned						sing.		aligned with a.r. constraint				aligned with geom. constraint			
Model	faces	N_q	N_s	time$_{fld}$	niter	time/iter	N_Q	N_P	N_v	T-joints	pairs	N_Q	N_P	time$_P$	N_v	T-joints	N_P	time$_P$	N_v	T-joints
holes3	11776	154	16	25.1	1	0.590	15943	47	69	52	29	13211	20	0.173	16	0	232	53.5	401	346
sculpt	7342	1255	16	9.00	2	0.295	11553	65	106	82	24	10792	38	0.725	58	36	104	12.9	146	88
joint	8766	776	23	8.85	2	0.629	13715	51	70	42	23	13440	45	0.701	59	32	68	9.17	100	68
fandisk	14454	4185	32	27.1	8	1.02	54421	63	93	56	6	51897	50	1.31	71	38	153	15.3	193	76
casting	36828	47845	98	250	7	12.1	756338	272	403	294	51	745319	233	70.7	326	218	297	114	429	296
screw	9596	1034	10	10.8	4	1.36	13384	40	74	60	4	14353	18	6.72	30	20	182	15.4	235	102
rockerarm	20088	1076	24	49.7	4	1.50	16311	87	141	108	24	16129	54	1.58	79	50	195	73.3	276	162
screwdriver	54300	906	20	352	11	12.94	5445	55	94	74	18	5306	40	1.59	62	40	118	41.5	165	90
maxplanck	50790	2198	15	555	9	1.82	36725	57	96	72	12	36725	50	2.92	84	60	671	232	1224	1086
bojito	82332	1837	72	960	6	3.02	32974	351	523	360	94	31987	165	8.98	230	146	403	76.7	554	318
fertility	27954	914	43	152	8	1.05	14211	172	265	194	39	14211	146	1.31	247	188	420	67.5	581	334
elephant	49918	6395	96	468	31	5.48	59342	396	635	486	44	57762	281	4.44	449	344	366	42.6	587	450

Table 5.1: *Column titles: N_q is the number of quads in the coarsest quadrangulation; N_s is the number of singularities; time$_{fld}$ is the time to smooth the cross-field; for parameterization, niter is the number of stiffening iterations and time/niter is the average time taken per iteration; N_Q is the number of quads in the fine quadrangulation used for patch generation; N_P is the number of patches; time$_P$ is the time to generate the T-mesh; N_v is the number of vertices in the T-mesh.*

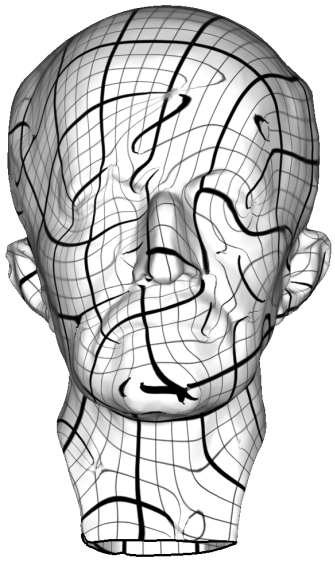


Figure 5.14: *PGP parameterization with target size chosen to match our number of patches in Figure 5.16*

optimization procedure, which has to trade T-joint creation for better geometric approximation or aspect ratio.

We observe that the number of control points in the T-mesh and the number of patches is within a factor of 2-4 of the number of singularities in the original mesh. We believe that with constraints imposed on patches it is difficult to improve on these numbers: the best possible number one can expect is approximately equal to the number of singularities. In this case, every quad of the mesh is supposed to have corners at singularities, which is extremely difficult to achieve for typically highly nonuniform singularity locations produced by feature-aligned fields.

Fitting T-splines. Once the T-mesh is constructed it can be used to define a T-spline/T-NURCCs surface which can be fitted to the original mesh. Our approach to fitting T-spline surfaces is identical to that of [62], with two important differences. Because our T-mesh is constructed by tracing parametrization lines,

the sums of parametric intervals on the opposite sides of each face are guaranteed to be equal if we simply use parametric length to determine the knot interval for each edge. This eliminates the need to introduce the extraordinary vertices at T-joints which were necessary to handle the T-meshes constructed using PGP.

The second difference is that we pass the information about sharp edges to the T-spline construction, and insert degenerate faces with zero knot intervals along sharp edges.

We obtain meshes suitable for fitting in our framework by setting the geometric error to 0.02 of the model diameter. Figure 5.19 shows the fit for several models. The L_2/L_∞ relative errors for the models shown in the figure are: for `joint`, 0.08%/2%, for `sculpt`, 0.1%/2%, for `botijo`, 0.1%/1% and for `fertility`, 0.06%/0.5%.

Performance. Performance numbers for different stages of the process are included in Table 5.1. The time needed for construction of the initial T-mesh patch layout is negligible in all cases and we ignore it. For larger meshes, in general, the dominant cost is mixed-integer field optimization (vertex-based version is more expensive than facet, and the running time of mixed-integer optimization increases faster than linearly due to correlation in the number of vertices and number of integer variables). This potentially may be alleviated by rounding in groups or the method described in [87]. For some meshes, the parametrization cost dominates, because of a large number of stiffening iterations. For smaller meshes, the cost of field optimization is far lower, and T-mesh layout may be the most significant expense. The cost of T-mesh optimization strongly depends on the number of patches produced, so it raises substantially, when lots of small patches are re-

quired. For example, for the `maxplanck` mesh, which has a lot of geometric detail at different scales, a very large number of patches is needed to obtain the same error. For this type of meshes, direct high-order patch approximation is not appropriate, and a displacement map or a hierarchical approximation is needed so that fewer patches can be used. The cost also raises if a very fine quad mesh is used (as it was necessary for `casting`). This is not a fundamental problem of the method, and it is primarily due to a suboptimal implementation of searching for an optimal parameter values for the parametrized optimization. We note that this numbers are strongly implementation, compiler and hardware dependent: for example, in our setup we were unable to match field optimization timings presented in [9], although the same MI code was used. The T-mesh optimization algorithm complexity is hard to estimate theoretically, so its scaling is difficult to predict. We have observed that it does not depend much on the number of singularities, but has a stronger dependence on geometric complexity of the shape (as more complex shapes require finer patches).

Limitations. There are several important limitations to our method.

Field quality. To the greatest extent, the quality of the T-mesh is determined by the quality of the field. Current techniques do not allow automatic enforcement of symmetries, and in many cases lead to unnecessary bending and twisting of patch boundaries. (often observed for organic shapes lacking sharp edge alignment constraints).

In an effort to eliminate unnecessary T-joints, our technique for singularity alignment changes parametric placement of singularities, but does not alter their position on the surface. Adjusting singularity positions whenever possible (if a sin-

gularity is located in an isotropic flat area) has potential for considerably improving the quality of the layouts.

Optimality. There is no guarantee that our result is close to the global optimum. For smaller meshes, the results seem hard to improve. For more complex meshes better quality seems possible. In all cases, a significant reduction in energy was achieved.

Robustness. There are three limiting factors in robustness of our method. The most substantial restriction is the need to obtain a locally one-to-one parametrization. Even with no rounding, for complex models with sharp features, foldovers in the parametrization are common, and stiffening iterations are not guaranteed to eliminate these. Another problem is detection of sharp features. In this work, we relied on the ability to tag sharp features based on the dihedral angles, or on having relatively rounded features. Finally, the global impact of sharp edges and cone alignment is difficult to predict as these constraints could force the parameterization to collapse due to global dependencies. The T-mesh construction is quite robust, but can produce excessive number of small patches for low-quality input fields.

Scalability. While we were able to obtain parametrizations for meshes of moderate size (approximately 100 thousand triangles), the main scalability bottleneck is the field optimization. The running time of the mixed-integer solver for larger meshes is dominated by the Gaussian elimination step for the constraints. Furthermore the number of iterations of the solver is proportional to the number of singularities, which, in turn, grows with geometric complexity of the model.

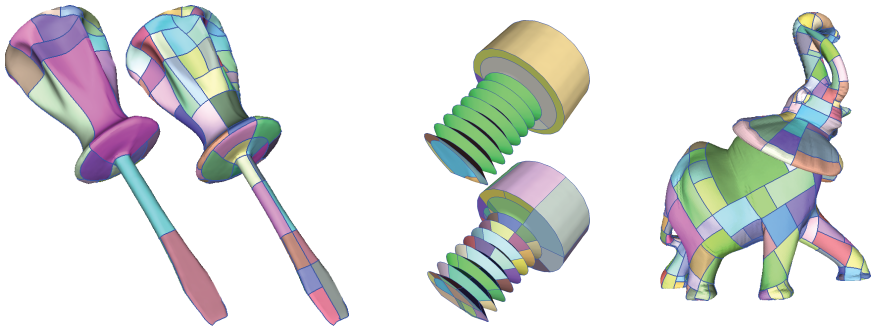


Figure 5.15: *The* `screw` *and* `screwdriver` *with patches of maximal size and obeying geometric error constraints; the* `elephant` *with geometric error optimization.*

5.8 Conclusions

The method presented in this work demonstrates the possibility of constructing coarse domain meshes while maintaining feature alignment, if T-joints are allowed in the mesh. Domain meshes constructed in this way are a natural fit for T-spline surfaces and related high-order constructions. Clearly, our T-mesh construction algorithm can be improved and extended in many ways to achieve more compact T-mesh structures with fewer nonessential T-joints. Furthermore, we observe that in many ways the quality of patch layouts is determined by the initial cross-field, and improving the quality of these fields is an important direction for future work.

While many algorithms and constructions for conforming meshes can be easily extended to T-meshes, overall, the theory and algorithms for this type of meshes is far less developed, presenting many interesting questions for future research.

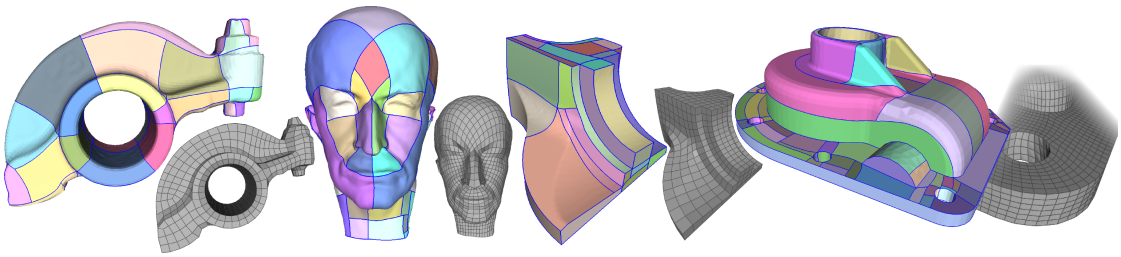


Figure 5.16: *Maximal patch sizes obtained by our algorithm while maintaining aspect ratio constraint 2.5. From left to right:* `rockerarm`*,* `fandisk`*,* `maxplanck`*, and* `casting`*. Smaller images show the coarsest quad meshes we could obtain.*

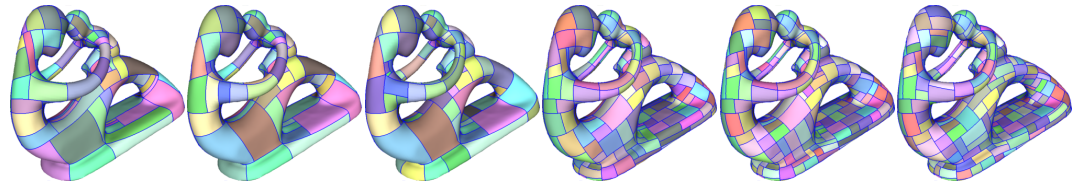


Figure 5.17: *Effects of changing the minimum thresholds for geometric error ϵ_0 and aspect ratio α_0 for the* `fertility` *mesh. Geometric error is measured relatively to the diameter of the bounding box of the mesh.*

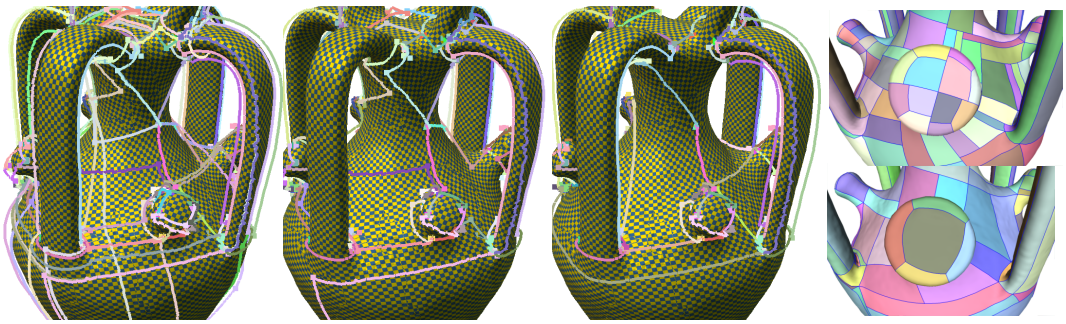


Figure 5.18: *From left to right: a full singularity adjacency graph for 72 singularities for the* `botijo` *mesh (209 edges); adjacency graph pruned by compatibility criterion (120 edges); adjacency graph pruned by aspect ratio 3 followed by compatibility (94 edges); comparison of aligned and non-aligned T-meshes.*

Figure 5.19: *Several T-spline models obtained by least-squares fitting from T-meshes generated by our algorithm.*

Bibliography

[1] V. Babenko, Y. Babenko, A. Ligun, and A. Shumeiko. Asymptotical behavior of the optimal linear spline interpolation error of C^2 functions. *East journal on approximations*, 12(1):71, 2006.

[2] Y. Bazilevs, VM Calo, JA Cottrell, JA Evans, TJR Hughes, S. Lipton, MA Scott, and TW Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 2009.

[3] Y. Bazilevs, VM Calo, JA Cottrell, JA Evans, TJR Hughes, S. Lipton, MA Scott, and TW Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010.

[4] Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum*, 27(2):449–458, 2008.

[5] M. Bertram, M.A. Duchaineau, B. Hamann, and K.I. Joy. Generalized B-spline subdivision-surface wavelets for geometry compression. *Visualization and Computer Graphics, IEEE Transactions on*, 10(3):326–338, 2004.

[6] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise Smooth Subdivision Surfaces With Normal Control. *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 113–120, 2000.

[7] Bitmapworld. Little girl head 3d model (http://www.turbosquid.com/3d-models/polygonal-head-little-girl-3d-model/302581), 2006.

[8] J. Bolz and P. Schröder. Rapid Evaluation of Catmull-Clark Subdivision Surfaces. *Proceedings of the seventh international conference on 3D Web technology*, pages 11–17, 2002.

[9] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Trans. Graph.*, 28(3):77, 2009.

[10] Tamy Boubekeur, Patrick Reuter, and Christophe Schlick. Scalar Tagged PN Triangles. *EUROGRAPHICS 2005 (Short Papers)*, pages 17–20, 2005.

[11] A. Buffa, D. Cho, and G. Sangalli. Linear independence of the T-spline blending functions associated with some particular T-meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1437–1445, 2010.

[12] M. Bunnell. Adaptive Tessellation of Subdivision Surfaces With Displacement Mapping. *GPU Gems 2*, pages 109–122, 2005.

[13] G.D. Cañas and S.J. Gortler. On Asymptotically Optimal Meshes by Coordinate Transformation. *Proceedings of 15th International Meshing Roundtable*, 2006.

[14] G.D. Cañas and S.J. Gortler. Surface remeshing in arbitrary codimensions. *The Visual Computer*, 22(9):885–895, 2006.

[15] W. Cao. An interpolation error estimate on anisotropic meshes in R^n and optimal metrics for mesh refinement. *SIAM J. Numer. Anal*, 45(6):2368–2391, 2007.

[16] N.A. Carr, J. Hoberock, K. Crane, and J.C. Hart. Rectangular multi-chart geometry images. In *Symposium on Geometry Processing*, page 190. Eurographics Association, 2006.

[17] T.J. Cashman. Beyond Catmull-Clark? a survey of advances in subdivision surface methods. In *Computer Graphics Forum*, volume 31, pages 42–61, 2012.

[18] T.J. Cashman, U.H. Augsdörfer, N.A. Dodgson, and M.A. Sabin. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. *ACM Transactions on Graphics (TOG)*, 28(3):46, 2009.

[19] T.J. Cashman, N.A. Dodgson, and M.A. Sabin. A symmetric, non-uniform, refine and smooth subdivision algorithm for general degree B-splines. *Computer Aided Geometric Design*, 26(1):94–104, 2009.

[20] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[21] Edwin Catmull and James Clark. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer-Aided Design*, pages 350–355, 1978.

[22] K.L. Clarkson. Building triangulations using ε-nets. In *Proc. thirty-eighth annual ACM symposium on Theory of computing*, pages 326–335. ACM New York, NY, USA, 2006.

[23] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 115–122, 1998.

[24] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914, 2004.

[25] D. Cohen-Steiner and J.M. Morvan. Restricted delaunay triangulations and normal cycle. In *Proc. nineteenth annual symposium on Computational geometry*, pages 312–321. ACM New York, NY, USA, 2003.

[26] Robert L. Cook. Shade Trees. *SIGGRAPH Comput. Graph.*, 18(3):223–231, 1984.

[27] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons Inc, 2009.

[28] L. Beiro da Veiga, A. Buffa, D. Cho, and G. Sangalli. Analysis-suitable t-splines are dual-compatible. *Computer Methods in Applied Mechanics and Engineering*, 249252(0):42 – 51, 2012. Higher Order Finite Element and Isogeometric Methods.

[29] Joel Daniels, Claudio T. Silva, and Elaine Cohen. Localized quadrilateral coarsening. *Computer Graphics Forum*, 28(5):1437–1444, 2009.

[30] Joel Daniels II, Claudio T Silva, and Elaine Cohen. Semiregular quadrilateralonly remeshing from simplified base domains. *Computer Graphics Forum*, 28(5):1427–1435, July 2009.

[31] EF D'Azevedo and RB Simpson. On optimal triangular meshes for minimizing the gradient error. *Numerische Mathematik*, 59(1):321–348, 1991.

[32] J. Deng, F. Chen, and Y. Feng. Dimensions of spline spaces over T-meshes. *Journal of Computational and Applied Mathematics*, 194(2):267–283, 2006.

[33] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng. Polynomial splines over hierarchical T-meshes. *Graphical Models*, 70(4):76–86, 2008.

[34] Tony DeRose, Michael Kass, and Tien Truong. Subdivision Surfaces in Character Animation. *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, 1998.

[35] S. Dong, P.T. Bremer, M. Garland, V. Pascucci, and J.C. Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25(3):1057–1066, 2006.

[36] M.R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer methods in applied mechanics and engineering*, 199(5-8):264–275, 2010.

[37] Qiang Du and Desheng Wang. Anisotropic centroidal voronoi tessellations and their applications. *SIAM J. Sci. Comput.*, 26(3):737–761, 2005.

[38] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd*

annual conference on Computer graphics and interactive techniques, pages 173–182. ACM, 1995.

[39] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *SIGGRAPH 1995*, pages 173–182, 1995.

[40] D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry*, 22(4):569–592, 1999.

[41] Tom Forsyth. Practical Displacement Mapping. In *Game Developers Conference*, 2003.

[42] Kev Gee. DirectX 11 Tessellation. In *Microsoft GameFest*, 2008.

[43] E. Grinspun, Y. Gingold, J. Reisman, and D. Zorin. Computing discrete shape operators on general meshes. In *Computer Graphics Forum*, volume 25, pages 547–556. Blackwell Synergy, 2006.

[44] Holger Grün. Efficient Tessellation on the GPU Through Instancing. *Journal Of Game Development*, 1(3), 2005.

[45] Xianfeng Gu and Shing-Tung Yau. Global conformal surface parameterization. In *Proc. 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 127–137, 2003.

[46] Y. He, K. Wang, H. Wang, X. Gu, and H. Qin. Manifold T-spline. *Geometric Modeling and Processing-GMP 2006*, pages 409–422, 2006.

[47] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *SIGGRAPH 2000*, pages 517–526, 2000.

[48] K. Hildebrandt, K. Polthier, and M. Wardetzky. Smooth feature lines on surface meshes. In *Symposium on Geometry Processing*, page 85. Eurographics Association, 2005.

[49] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise Smooth Surface Reconstruction. *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 295–302, 1994.

[50] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. *SIGGRAPH Course Notes*, 2007.

[51] J. Huang, M. Zhang, J. Ma, X. Liu, L. Kobbelt, and H. Bao. Spectral quadrangulation with orientation and alignment control. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2008.

[52] F. Kälberer, M. Nieser, and K. Polthier. QuadCover: Surface Parameterization using Branched Coverings. *Computer Graphics Forum*, 26(3):375–384, 2007.

[53] E. Kalogerakis, P. Simari, D. Nowrouzezahrai, and K. Singh. Robust statistical estimation of curvature on discretized surfaces. In *Symposium on Geometry Processing*, pages 13–22, 2007.

[54] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM Trans. Graph.*, 22(3):350–357, 2003.

[55] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 271–278. ACM Press/Addison-Wesley Publishing Co., 2000.

[56] Denis Kovacs, Jason Mitchell, Shanon Drone, and Denis Zorin. Real-time Creased Approximate Subdivision Surfaces. *I3D '09: Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, pages 155–160, 2009.

[57] Denis Kovacs, Ashish Myles, and Denis Zorin. Anisotropic quadrangulation. *Computer Aided Geometric Design*, 28(8):449 – 462, 2011. Solid and Physical Modeling 2010.

[58] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced Subdivision Surfaces. *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94, 2000.

[59] A.W.F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 1998*, pages 95–104, 1998.

[60] Matt Lee. Next-Generation Graphics Programming on XBox 360. In *Microsoft GameFest*, 2006.

[61] M. Li and M.A. Scott. On the nesting behavior of t-splines. Technical Report 11-13, ICES, 2011.

[62] W.C. Li, N. Ray, and B. Lévy. Automatic and interactive mesh to T-spline conversion. In *Symposium on Geometry Processing*, page 200. Eurographics Association, 2006.

[63] X. Li, J. Deng, and F. Chen. Surface modeling with polynomial splines over hierarchical T-meshes. *The Visual Computer*, 23(12):1027–1033, 2007.

[64] X. Li, J. Deng, and F. Chen. Polynomial splines over general T-meshes. *The Visual Computer*, pages 1–10, 2009.

[65] X. Li, J. Deng, and F. Chen. Polynomial splines over general T-meshes. *The Visual Computer*, 26(4):277–286, 2010.

[66] X. Li, J. Zheng, T.W. Sederberg, T.J.R. Hughes, and M.A. Scott. On linear independence of T-spline blending functions. *Computer Aided Geometric Design*, 29(1):63–76, 2012.

[67] Charles Loop and Scott Schaefer. Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches. *ACM Trans. Graph.*, 27(1):1–11, 2008.

[68] Charles Loop and Scott Schaefer. G^2 Tensor Product Splines Over Extraordinary Vertices. *Computer Graphics Forum*, 27(5):1373–1382, 2008.

[69] M. Lounsbery, T.D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics (TOG)*, 16(1):34–73, 1997.

[70] M. Marinov and L. Kobbelt. Automatic generation of structure preserving multiresolution models. *Computer Graphics Forum*, 24(3):479–486, September 2005.

[71] Jean-Marie Mirebeau. Optimal meshes for finite elements of arbitrary order. *Constructive Approximation*, 32:339–383, 2010.

[72] B. Mourrain. On the dimension of spline spaces on planar t-subdivisions. *Arxiv preprint arXiv:1011.1752*, 2010.

[73] K. Müller, C. Fünfzig, L. Reusche, D. Hansford, G. Farin, and H. Hagen. Dinus: Double insertion, nonuniform, stationary subdivision surfaces. *ACM Transactions on Graphics (TOG)*, 29(3):25, 2010.

[74] K. Müller, L. Reusche, and D. Fellner. Extended subdivision surfaces: Building a bridge between NURBS and Catmull-Clark surfaces. *ACM Transactions on Graphics (TOG)*, 25(2):268–292, 2006.

[75] A. Myles, N. Pietroni, D. Kovacs, and D. Zorin. Feature-aligned T-meshes. *ACM Transactions on Graphics (TOG)*, 29(4):117, 2010.

[76] T. Ni, Y. Yeo, A. Myles, V. Goel, and J. Peters. GPU Smoothing of Quad Meshes. *International Conference on Shape Modeling and Applications*, pages 3–9, 2008.

[77] Y. Ohtake, A. Belyaev, and H.P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *International Conference on Computer Graphics and Interactive Techniques*, pages 609–612. ACM New York, NY, USA, 2004.

[78] J. Palacios and E. Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3):55, 2007.

[79] J. Peters. Patching Catmull-Clark Meshes. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 255–258, 2000.

[80] J. Peters and U. Reif. Analysis of algorithms generalizing b-spline subdivision. *SIAM Journal on Numerical Analysis*, 35(2):728–748, 1998.

[81] Nico Pietroni, Marco Tarini, and Paolo Cignoni. Almost isometric mesh parameterization through abstract domains. *IEEE Trans. Visualization and Computer Graphics*, 99(RapidPosts), 2009.

[82] H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. *Lecture Notes in Computer Science*, pages 560–572, 2004.

[83] H. Pottmann, J. Wallner, Q.X. Huang, and Y.L. Yang. Integral invariants for robust geometry processing. *Computer Aided Geometric Design*, 2008.

[84] H. Prautzsch and Q. Chen. Analyzing midpoint subdivision. *Computer Aided Geometric Design*, 2011.

[85] N. Ray, W.C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460–1485, 2006.

[86] N. Ray, B. Vallet, W.C. Li, and B. Lévy. N-Symmetry direction field design. *ACM Trans. Graph.*, 27:2, 2008.

[87] Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. Geometry-aware direction field processing. *ACM Trans. Graph.*, 29(1):1–11, 2009.

[88] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

[89] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 486–493, 2004.

[90] P.V. Sander, S.J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parameterization. *Eurographics Workshop on Rendering*, 2002, 2002.

[91] MA Scott, X. Li, TW Sederberg, and TJR Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213:206–222, 2012.

[92] M.A. Scott, X. Li, T.W. Sederberg, and T.J.R. Hughes. Local refinement of analysis-suitable t-splines. *Computer Methods in Applied Mechanics and Engineering*, 213216(0):206 – 222, 2012.

[93] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 276–283. ACM, 2004.

[94] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 477–484. ACM, 2003.

[95] T.W. Sederberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform recursive subdivision surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 387–394. ACM, 1998.

[96] A. Sheffer and E. de Sturler. Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening. *Engineering with Computers*, 17(3):326–337, 2001.

[97] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):171, 2006.

[98] J.R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. *11th International Meshing Roundtable*, pages 115–126, 2002.

[99] L.J. Shiue, I. Jones, and J. Peters. A Realtime GPU Subdivision Kernel. *Proceedings of ACM SIGGRAPH 2005*, 24(3):1010–1015, 2005.

[100] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal equivalence of triangle meshes. *ACM Trans. Graph.*, 27:77:1–77:11, August 2008.

[101] J. Stam. On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree. *Computer Aided Geometric Design*, 18(5):383–396, 2001.

[102] Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo, and Enrico Puppo. Practical quad mesh simplification. *Computer Graphics Forum*, 29(2), 2010.

[103] Natasha Tatarchuk. Real-Time Tessellation on the GPU. In *SIGGRAPH Advanced Real-Time Rendering in 3D Graphics and Games Course*, 2007.

[104] G. Tewari, J. Snyder, P.V. Sander, S.J. Gortler, and H. Hoppe. Signal-specialized parameterization for piecewise linear reconstruction. *Symposium on Geometry Processing*, pages 55–64, 2004.

[105] Y. Tong, P. Alliez, D. Cohen-Steiner, and M. Desbrun. Designing quadrangulations with discrete harmonic forms. *Symposium on Geometry Processing*, pages 201–210, 2006.

[106] A. Torn and A. Zilinskas. Global Optimization, volume 350 of. *Lecture Notes in Computer Science*, 1989.

[107] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason Mitchell. Curved PN Triangles. *I3D 2001: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 159–166, 2001.

[108] W. Wang, Y. Zhang, M.A. Scott, and T.J.R. Hughes. Converting an unstructured quadrilateral mesh to a standard T-spline surface. *Computational Mechanics*, pages 1–22, 2011.

[109] T. Weinkauf and D. Günther. Separatrix Persistence: Extraction of Salient Edges on Surfaces Using Topological Methods. In *Computer Graphics Forum*, volume 28, pages 1519–1528. Blackwell Publishing Ltd, 2009.

[110] R. Zayer, C. Rossl, and H.P. Seidel. Discrete Tensorial Quasi-Harmonic Maps. *Proceedings of Shape Modeling and Applications*, pages 276–285, 2005.

[111] R. Zayer, C. Rössl, and H.P. Seidel. Setting the boundary free: A composite approach to surface parameterization. 2005.

[112] Rhaleb Zayer, Christian Rössl, and Hans-Peter Seidel. r-Adaptive parameterization of surfaces. Technical report, 2004.

[113] W. Zeng, F. Luo, S. T. Yau, and X. D. Gu. Surface quasi-conformal mapping by solving beltrami equations. In *Proceedings of the 13th IMA International Conference on Mathematics of Surfaces XIII*, pages 391–408, Berlin, Heidelberg, 2009. Springer-Verlag.

[114] Hansong Zhang and Kenneth E. Hoff, III. Fast Backface Culling Using Normal Masks. *SI3D '97: Proceedings of the 1997 Symposium on Interactive 3D graphics*, pages 103–106, 1997.

[115] D. Zorin. A method for analysis of c 1-continuity of subdivision surfaces. *SIAM Journal on Numerical Analysis*, 37(5):1677–1708, 2000.

[116] D. Zorin and P. Schröder. A unified framework for primal/dual quadrilateral subdivision schemes. *Computer Aided Geometric Design*, 18(5):429–454, 2001.

[117] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 259–268. ACM Press/Addison-Wesley Publishing Co., 1997.