A Generalized Blending Scheme for Arbitrary Order of Continuity

by

Xiang Fang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor(s):	Stephen Mann Professor, School of Computer Science, University of Waterloo
Internal Member:	Christopher Batty Associate Professor, School of Computer Science, University of Waterloo
Internal Member:	Yuying Li Professor, School of Computer Science, University of Waterloo
Internal-External Member:	Sanjeev Bedi Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo
External Examiner:	Peter Salvi Associate professor, Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics

Author's Declaration

Abstract

In this thesis, new templates and formulas of blending functions, schemes, and algorithms are derived for solving the scattered data interpolation problem. The resulting data fitting scheme interpolates the positions and derivatives of a triangular mesh, and for each triangle of the mesh blends three triangular sub-surfaces, and creates a triangular patch. Similar to some existing schemes, the resulting surface inherits the derivatives of the sub-surfaces on the boundaries. In contrast with existing schemes, the new scheme has additional properties: The order of interpolated derivatives is extended to arbitrary values, and the restrictions of the sub-surfaces are relaxed. Then based on the properties of the new blending functions, an algorithm for constructing smooth triangular surfaces with global geometric continuity is described. The new blending functions and the scheme are then extended to multi-sided faces. The algorithm using these new blending functions accepts data sites formed by multi-sided polygons.

Acknowledgements

Throughout the writing of this dissertation I have received a great deal of assistance.

I would like to thank my supervisor, Professor Stephen Mann, for his guidance through each stage of the process as well as his patience.

I would like to thank Associate Professor Christopher Batty and Professor Sanjeev Bedi for being my PhD committee members, Professor Yuying Li for being my internal committee member, and Associate Professor Peter Salvi for being my external committee member.

I would like to thank Dr. David Ross Cheriton for his support of scholarship.

I would like to thank the support of NSERC for the funding.

I would like to thank the members of the Computer Graphics Lab for their help and advice.

At last, I would like to thank my parents for their support as always.

Dedication

This thesis is dedicated to my parents.

Table of Contents

List of Figures

List of Symbols

S_i Sub-functions. 18

$S^{n,d,p}$ Blended result. 18

d Blending order. 18

$f_i^{n,d,p}$ Blending functions. 18

n Blended terms. 18

p_i Inheritance indicators. 18

q^p Inherited terms. 18

Chapter 1

Introduction

The scattered data interpolation problem is to design methods to construct smooth surfaces that interpolate the locations and normals of vertices from data meshes. The data mesh is a combination of vertices assigned with locations and normals, as well as the edges and faces formed by these vertices. Data meshes can be constructed by sampling from specific surfaces, or generated manually with modelling software. The interpolation techniques are used to construct new surfaces that approximate the original surface, or construct surfaces of desired shapes controlled by the data meshes.

In general, the surfaces constructed for the scattered data interpolation problem are required to achieve a specific order of continuity. First order continuity is always required to ensure that the surfaces are visually smooth. Second order continuity is needed in the automotive industry to avoid kinks in reflection lines. Furthermore in some applications, third order continuity is needed to avoid sharp changes in jerk [10, 13].

Often the faces of the data meshes are triangular, and spline construction schemes with triangular Bézier patches are used. Triangular Bézier patches are specified by a finite number of points called *control points* that can be used as geometric shape handles. More specifically, a degree n triangular Bézier patch is

$$\sum P_{ijk} B^n_{ijk}(u, v, w) \tag{1.1}$$

where the P_{ijk} are the control points and the

$$B^n_{ijk}(u, v, w) = \frac{n!}{i!j!k!} u^i v^j w^k \tag{1.2}$$

are the multivariate Bernstein polynomials [3]. To construct a smooth surface, adjacent patches are required to meet with a specified order of continuity. These continuity constraints are met by forcing the control points from each patch to satisfy some geometric conditions. Such an approach has two major disadvantages. The first disadvantage is that when the continuity order gets higher, more control points (and higher degree patches) are required to build the patches. The second disadvantage is that these continuity conditions cause problems when control points are included in multiples conditions for different continuities of different boundaries. In this case the cross-boundary continuity conditions for all the boundaries surrounding a vertex creates a cycle of constraints that has no solution. This problem is known as the *vertex consistency problem*. Such problems add additional complexity to these schemes [19].

To avoid these disadvantages of the polynomial schemes, *surface blending schemes* were introduced. Surface blending schemes construct surfaces by blending multiple candidate patches into a new patch using specific weights. With appropriate weights, the resulting surface patch will have the same value and derivatives as the candidate patches in certain areas. The effect of the blending is to divide the task of establishing continuity across multiple boundaries and corners of the faces into several sub-tasks. A typical example is *Nielson's method*, which is applied to triangular faces, and blends three triangular surfaces into one. Each boundary of the result constructed by Nielson's method inherits the tangent field on a boundary of one of the blended surfaces. Details on Nielson's scheme are given in Section 2.6.1.

In this dissertation, I present a generalized template to construct the blending schemes, give typical examples of the triangular and binary blending schemes, and design the algorithms for solving the scattered data interpolation problem. More specifically, in Chapter 3, I give the generalized method to construct the blending functions with respect to the desired number of sub-surfaces, order of continuity, and weights that affect the shape. The term *sub-surfaces* is used to denote the functions to be blended since most of the time the blended functions are surfaces. In some settings the term *sub-function* may be used. In Chapter 4, I use my blending functions with three sub-surfaces to design a triangular surface blending scheme and compare my new scheme to Nielson's method. My new scheme has fewer restrictions than Nielson's method, and allows new interpolation algorithms to solve a more general scattered data interpolation problem. Furthermore, several binary blending schemes are used in the following discussions. All of these binary blending schemes have two sub-surfaces or sub-functions. These binary schemes focus on the value and derivatives only at two points, so they have fewer restrictions than the schemes having three or more sub-functions and inheritance indicators with larger and overlapped zero areas (for example the triangular blending scheme given above).

In Chapter 5, I design an interpolation algorithm for solving the scattered data interpolation problem of triangular data meshes. The algorithm constructs a surface for each element (vertex, edge, and face) of the data mesh. So there are two major tasks: constructing surfaces for elements and transferring the surface data to other elements. The first task requires constructing local coordinate systems for the elements, and the second task requires constructing *reparameterization functions* among the local coordinate systems. The resulting surfaces of the basic algorithm constructed in this chapter have continuous normals or curvatures. In Chapter 6, I modify the reparameterization functions between the edges and faces, and show that the new algorithm constructs surfaces that are parametrically continuous everywhere with respect to the corresponding coordinate systems constructed for elements in the algorithm.

In Chapter 7, I extend all previous work to the multi-sided case by increasing the number of elements that are blended with the generalized scheme, and updating the triangular surface construction algorithms to multi-sided versions. The extended algorithms fit surfaces to faces with more than three edges, and constructs surface with the same properties as the triangular algorithms.

Chapter 2

Background

In this chapter, the scattered data interpolation problem and necessary background knowledge are stated, and the method of Nielson is reviewed. In particular Nielson's blending functions and corresponding scheme will be stated, since a large part of the work in this dissertation is a generalization of these blending functions and scheme.

2.1 Scattered Data Interpolation Problem

The scattered data interpolation problem takes a data mesh as input, and constructs a smooth surface that interpolates the vertices of the data mesh as the output. I will use the terms *vertex*, *edge*, and *face* to describe the elements of the data mesh, and the terms *point/corner*, *boundary*, and *triangle/quadrilateral/polygon* to describe the elements of the surface and the domain.

Initially the case when the data mesh faces are all triangular is considered. The data mesh is given as a list of vertices and faces. All vertices are given in a sequence and each vertex has a position and a normal. Then each face is formed by three vertices. The vertices of a face are ordered so that the outside of the data mesh is on the counter-clockwise side of the triangle. The edges are not explicitly specified but are implicitly formed by two consecutive vertices in a triangle. The two vertices of an edge are not ordered, but for convenience it is assumed that one of them is the *head* of the edge, and another one is the *tail*. Finally I assume that the data mesh forms a closed manifold, so each vertex appears in at least two edges and two faces, and each edge appears in exactly two faces. With these assumptions, the formal definition of the triangular data mesh used in this dissertation is given in Definition 2.1.

Definition 2.1. *A triangular data mesh is a set* $\{\mathbb{V}, \mathbb{E}, \mathbb{F}\}$, *where*

- $\mathbb{V}$ *is a set of vertices.*
- $\mathbb{E}$ *is a set of edges, each element* $E \in \mathbb{E}$ *is formed by two vertices:*
$$E = \{V_0, V_1\} \subset \mathbb{V}, \tag{2.1}$$
where V_0 *and* V_1 *are distinct.*
- $\mathbb{F}$ *is a set of faces, each element* $F \in \mathbb{F}$ *is formed by three vertices:*
$$F = \{V_0, V_1, V_2\} \subset \mathbb{V}, \tag{2.2}$$
where V_0, V_1, *and* V_2 *are distinct, and the subsets of* F $\{V_0, V_1\}$, $\{V_1, V_2\}$, *and* $\{V_2, V_0\}$ *are elements of* $\mathbb{E}$.
- *For each* $V \in \mathbb{V}$, *there exist* $E \in \mathbb{E}$ *and* $F \in \mathbb{F}$ *such that* $V \in E$ *and* $V \in F$.
- *For each* $E \in \mathbb{E}$, *there exist exactly two faces* $F_1, F_2 \in \mathbb{F}$ *such that* $E \subset F_1$ *and* $E \subset F_2$.

The number of vertices in a face can be extended to values larger than three. In that case the data mesh is called *multi-sided*. One additional condition for the multi-sided case is that in a face, only the adjacent pairs of vertices form edges. The formal definition of the multi-sided data mesh is given in Definition 2.2.

Definition 2.2. *A multi-sided data mesh is a set* $\{\mathbb{V}, \mathbb{E}, \mathbb{F}\}$, *where*

- $\mathbb{V}$ *is a set of vertices.*
- $\mathbb{E}$ *is a set of edges, each element* $E \in \mathbb{E}$ *is formed by two vertices:*
$$E = \{V_0, V_1\} \subset \mathbb{V}, \tag{2.3}$$
where V_0 *and* V_1 *are distinct.*
- $\mathbb{F}$ *is a set of faces, each element* $F \in \mathbb{F}$ *is formed by* $n \geq 3$ *vertices:*
$$F = \{V_0, V_1, \ldots, V_{n-1}\} \subset \mathbb{V}, \tag{2.4}$$
where n *is the number of vertices in* F, V_0, V_1, …, V_{n-1} *are distinct, the subsets of* F $\{V_0, V_1\}$, $\{V_1, V_2\}$, …, $\{V_{n-1}, V_0\}$ *are elements of* $\mathbb{E}$, *and all other* 2*-element subsets of* F *are not elements of* $\mathbb{E}$.

- *For each $V \in \mathbb{V}$, there exist $E \in \mathbb{E}$ and $F \in \mathbb{F}$ such that $V \in E$ and $V \in F$.*
- *For each $E \in \mathbb{E}$, there exist exactly two faces $F_1, F_2 \in \mathbb{F}$ such that $E \subset F_1$ and $E \subset F_2$.*

In this dissertation it is assumed that all multi-sided faces of the data mesh are convex if projected to a certain plane. This property ensures that the algorithm for the multi-sided case always works. The concave case is discussed further in Section 9.1 of this dissertation.

These definitions are used in this dissertation. However manifolds with boundaries are also acceptable for my surface construction algorithms. Details on surfaces constructed for manifolds with boundaries are not discussed in this thesis, although one of the examples is for a manifold with a boundary.

Furthermore, there are two types of data interpolation: functional interpolation and parametric interpolation. A functional surface is one that can be written as $z = f(x, y)$, and a parametric surface has a bivariate parametric domain and three functions continuous $x(u, v)$, $y(u, v)$, and $z(u, v)$. The focus of this dissertation is on parametric interpolation, although at times reference will be made to functional interpolation.

2.2 Normal and Curvature

The *normal* of a surface at a point is the direction that is perpendicular to the surface at the point. Parametrically the normal is a vector that can be calculated by taking the cross product of the two partial derivatives of the surface. While at times normals are restricted to be unit length, in this dissertation I will only consider the direction.

The *curvature* of a curve at a point on the curve is a measure of how much the curve deviates from a straight line, or how much a surface deviates from a plane. For a plane curve, the curvature of the curve at a point is the inverse of the radius of the *osculating circle*. The osculating circle is the circle that best approximates the curve at a point. More specifically, given a point P on the curve, and two other points P_0 and P_1 on the curve and close to P, a circle can be uniquely determined (when the three points are co-linear, the straight line can be considered as a circle with infinite radius). The osculating circle is the limit of this circle when P_0 and P_1 approach P.

For a surface, there are two major types of curvature: *Gaussian curvature* and *mean curvature*. Both Gaussian and mean curvature are calculated from the *principal curvatures*. For a given point P on the surface, a *normal plane* of P is a plane that contains P and the normal vector of P. The intersection of the surface and a normal plane is a plane curve, and that curve has a curvature at P. Such curvature may vary for different normal planes of P. The two principal curvatures are the maximum and minimum values of these curvatures. The Gaussian curvature is the product of the two principal curvatures, and the mean curvature is the average of the two principal curvatures.

More formal definitions of the normal and the curvatures can be found in any book of elementary differential geometry [18].

By the definition of the normal and of the curvature, their values are determined by the surface itself. The normal and curvature may be calculated by a particular parameterization, but the normal and curvature do not change if the surface is reparameterized. More specifically, curve curvature and mean curvature can change sign based on the parameterization, but Gaussian curvature has the same sign for any parameterization. So in this dissertation I call the location, the normal, and the Gaussian curvature of a point on a surface as the *geometric values* of a surface, since they have the following properties:

- If the derivatives of a surface are uniquely determined, then its geometric values are also uniquely determined.
- The geometric value of a surface is independent from the parameterization of the surface.
- The geometric value may exist even when the derivatives do not exist.

These properties are used in the proofs of the continuity of geometric values in Chapter 4. Currently I only discussed the location, the normal, and the curvature. However if any other values satisfy these properties, similar theorems should exist.

Curvature plots of simple surfaces are used to illustrate Gaussian curvature. In the curvature plots, green denotes zero Gaussian curvature (where the surface curvature is flat in one direction), red denotes positive Gaussian curvature (at peaks and cups), and blue denotes negative Gaussian curvature (in regions of saddle points). Notice that for positive curvature the color varies from green to yellow then to red as the curvature increases, and for negative curvature the color varies from green to cyan then to blue as the curvature decreases. The Gaussian curvatures in this chapter are calculated numerically, and the size of the mesh and curvature bound of each example may vary.

As shown in Figure 2.1, the curvature plots of the cone and the cylinder are green everywhere since for each point on the surface one of the principal curvatures is zero, and thus the Gaussian curvature is zero. The curvature plot of the sphere is a constant shade of red since all principal curvatures as well as the Gaussian curvature are constant for all points on the sphere. For the torus, the surface has both positive (in the region close to the outer ring) and negative (in the region close to the inner ring) Gaussian curvature, as well as zero curvature along the top and bottom circles of the torus.

2.3 Isophotes

Isophotes are a tool to evaluate the quality of a surface. For a point P on the surface with normal n, define the "brightness" of the surface at P as the inner product of n with the normalized vector from the eye-point to P. Then an isophote is a curve formed by the surface points on the surface with certain value of brightness.

The continuity of the isophotes is dependent on the continuity of the surface. The order of continuity of the isophotes is always one less than the order of continuity of the surface. So for a C^0 surface its isophotes are disconnected, for a C^1 surface its isophotes are only C^0 and thus have kinks, and for a C^2 surface its isophotes are C^1. In Chapter 8 curvature and isophotes are both used to show the continuity of the resulting surfaces.

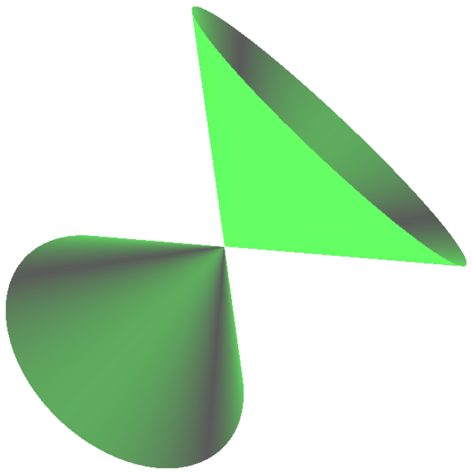
Figure 2.2 is a simple example illustrating isophotes. In the figure, the surface is two cylinder pieces of different radii joined with C^1 continuity along a common line. An eye-point close to the surface is chosen to make the kinks of the isophotes obvious. On this surface the isophotes are smooth on each cylinder piece. However the isophotes are not smooth across the boundary of two pieces since the surface is C^1- but not C^2-continuous along the common line.

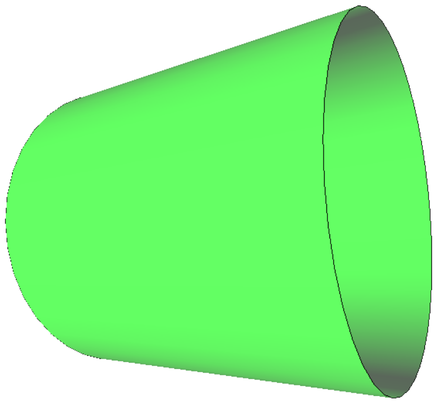
2.4 Smoothness and Continuity

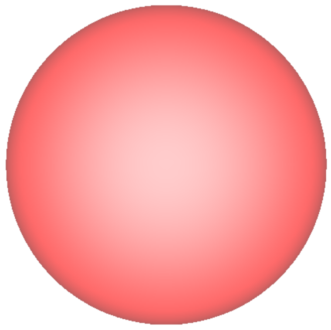
We want to construct smooth surfaces to solve the scattered data interpolation problem. I define "smooth" to mean adjacent surface patches meet with some level of continuity. In this dissertation two types of continuity are discussed:

- The continuity of geometric values.
- Local parametric continuity and global geometric continuity.

The meaning of geometric values is discussed in Section 2.2. The continuity of the geometric values can be proven by showing the equality of the derivatives. Then the


(a)


(b)


(c)

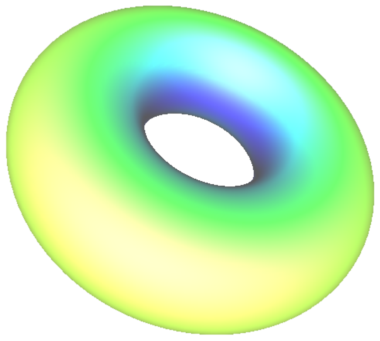

(d)

Figure 2.1: Four different geometric primitives shaded by the Gaussian curvature; (a) cone; (b) cylinder; (c) sphere; (d) torus.

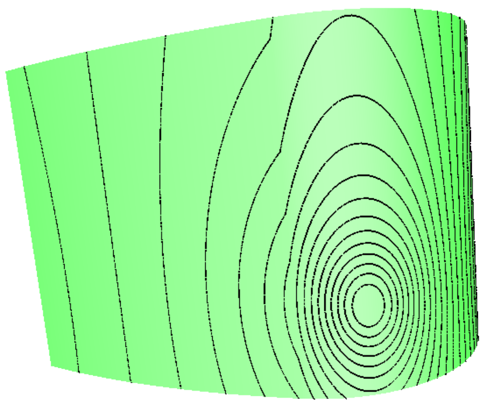


Figure 2.2: The isophotes of a C^1-continuous surface.

equality of the geometric values can be passed to different surfaces even with different parameterizations. In this dissertation only three types of geometric values are discussed: the locations for zero-th order continuity, normals for first order continuity, and curvatures for second order continuity.

For the second type of continuity, the term "parametric continuity" needs to be discussed with the surface parameterization. For the scattered data interpolation problem, most of the time the surface cannot be constructed using a single coordinate system. So the continuity needs to be considered locally. For a point P on a surface S, the surface S is called *parametrically continuous* at P if there exists a parameterization of S where the derivatives of S are continuous at P. Furthermore, if a surface S is parametrically continuous everywhere (possibly with different local parameterizations at different points on S), then S is said to be *geometrically continuous.*

2.5 Barycentric Coordinates

In this dissertation, surfaces interpolating triangular faces of the data mesh are defined over *barycentric coordinate systems.* For an arbitrary triangle, each point on the plane of the triangle can be considered as an affine combination with respect to the three corners of the triangle (an affine combination is a linear combination where the weights sum to 1). The barycentric coordinates of this point with respect to the triangle are the weights of that affine combination. The barycentric coordinates are usually represented by (u, v, w). Since the barycentric coordinates are the weights of an affine combination, the sum of u,

v, and w is always 1. Also

- If the point is located inside the triangle, then each of u, v, and w are greater than 0 and less than 1.
- If the point is located on a boundary (excluding the endpoints) of the triangle, then one of u, v, and w is 0, and the sum of the other two is 1.
- If the point is located at a corner of the triangle, then one of u, v, and w is 1, and the other two are 0.

Since the sum of the parameters of the barycentric coordinates is always 1, there are two independent degrees of freedom instead of three.

In this dissertation, all surfaces not defined over the barycentric coordinate system are defined over a *Cartesian coordinate system*. Since I use many Cartesian coordinate systems with different origins and basis vectors, I also refer to these as *local coordinate systems*.

2.6 Surface Blending Schemes

Surface blending schemes are the methods that create a surface by combining multiple *sub-surfaces*. For each point in the domain, the value of the resulting surface is obtained by combining corresponding values from the sub-surfaces with weights. The weights are functions that have the same domain as that of the surfaces, and are called *blending functions*. The sum of the blending functions is required to be 1, so the sum of a set of points weighted by the blending functions is an affine combination.

My schemes presented in this dissertation can be considered as *transfinite schemes*. Transfinite interpolation is a concept in numerical analysis that constructs functions over a planar domain in such a way that they match a given function on the boundary. This method is applied in geometric modelling, similar to the way I use it in this dissertation. Although my method can be considered as a transfinite scheme, in this dissertation I use the term "blending" since the focus of my work is on the functions used to combine the sub-surfaces. Some transfinite examples can be found in [23, 22, 4], for both triangular and multi-sided cases. Notice that these methods are transfinite, but may not be considered as the blending schemes.

2.6.1 Nielson's Blending Scheme

An example of a surface blending scheme is *Nielson's method* [17]. Nielson's method is designed for constructing first order smooth, piecewise triangular surface patches and uses a barycentric coordinate system as the domain for each triangular patch. Nielson's blending functions are given in Definition 2.3.

Definition 2.3. *For a point (u, v, w) in barycentric coordinates, Nielson's blending functions are defined as*

$$\begin{aligned} f_0^{Nielson}(u,v,w) &= \frac{vw}{uv+vw+wu}, \\ f_1^{Nielson}(u,v,w) &= \frac{wu}{uv+vw+wu}, \\ f_2^{Nielson}(u,v,w) &= \frac{uv}{uv+vw+wu}. \end{aligned} \tag{2.5}$$

Nielson's method is as follows. Assume three vertices V_0, V_1, and V_2 and their normals are given, as well as three parametric surfaces $S_0(u, v, w)$, $S_1(u, v, w)$, and $S_2(u, v, w)$ that satisfy the following conditions.

Condition 2.1. *For three vertices V_0, V_1, and V_2 and three surfaces S_0, S_1, and S_2,*

- *S_0, S_1, and S_2 interpolate the locations of V_0, V_1, and V_2,*

$$\begin{aligned} V_0 = S_0(1,0,0) = S_1(1,0,0) = S_2(1,0,0), \\ V_1 = S_0(0,1,0) = S_1(0,1,0) = S_2(0,1,0), \\ V_2 = S_0(0,0,1) = S_1(0,0,1) = S_2(0,0,1). \end{aligned} \tag{2.6}$$

- *S_0, S_1, and S_2 share the same boundaries,*

$$\begin{aligned} &S_0(u,v,0) = S_1(u,v,0) = S_2(u,v,0), && \textit{for } 0 < u, v < 1 \textit{ and } u+v=1, \\ &S_0(0,v,w) = S_1(0,v,w) = S_2(0,v,w), && \textit{for } 0 < v, w < 1 \textit{ and } v+w=1, \\ &S_0(u,0,w) = S_1(u,0,w) = S_2(u,0,w), && \textit{for } 0 < w, u < 1 \textit{ and } w+u=1. \end{aligned} \tag{2.7}$$

- *S_0 interpolates the normals of V_1 and V_2.*
- *S_1 interpolates the normals of V_2 and V_0.*
- *S_2 interpolates the normals of V_0 and V_1.*

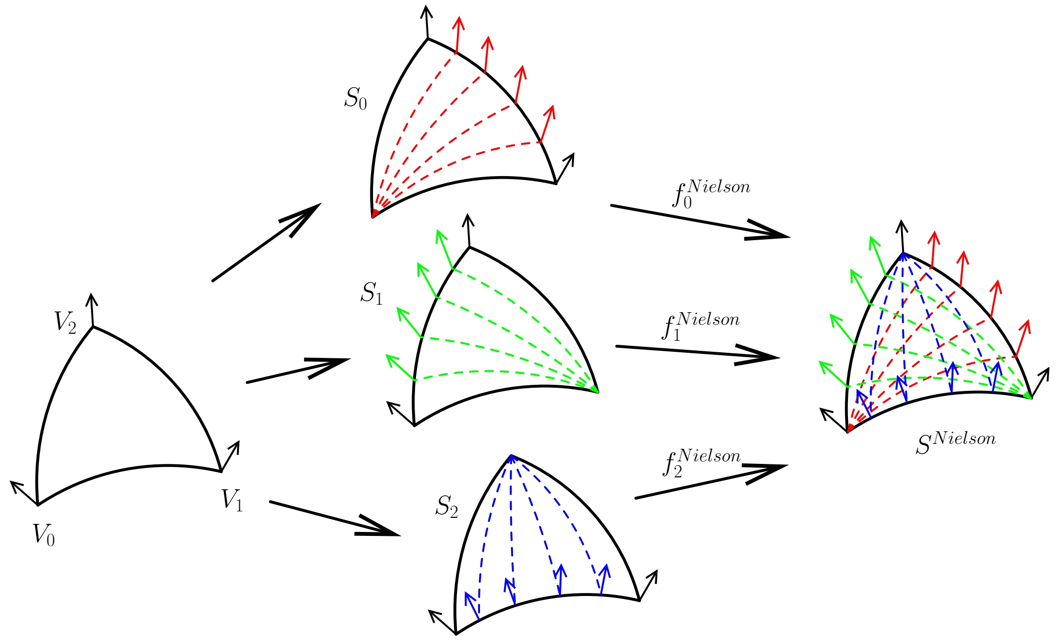


Figure 2.3: The process of Nielson's scheme.

The surfaces S_0, S_1, and S_2 are called sub-surfaces. Then the blended result is

$$S^{Nielson}(u, v, w) = \sum_{i=0}^{2} \Big[f_i^{Nielson}(u, v, w) S_i(u, v, w) \Big]. \tag{2.8}$$

The process of Nielson's scheme is shown in Figure 2.3.

Nielson proved that the boundaries of S inherit the tangent plane field of one of each sub-surface. More specifically, S and S_0 have the same tangent plane field along the boundary V_1V_2, S and S_1 have the same tangent plane field along the boundary V_2V_0, and S and S_2 have the same tangent plane field along the boundary V_0V_1. Also, Nielson provided a corresponding algorithm to solve the scattered data interpolation problem with continuous normals.

Notice that these blending functions contain a common denominator $uv + vw + wu$. When two of u, v, and w are zero (at the corners of the triangular domain), the denominator is zero, and Nielson's blending functions are not well defined. However since the limit of the sum of the blending functions is always 1, the limit of S at a corner exists when the three sub-surfaces share the same value at that corner. This issue at the corners is why the three sub-surfaces are required to interpolate all three data vertices. This interpolation of common data also applies to higher order derivatives. Since the normals are matched at the corners, the normals of the blended result exist geometrically. However, with Nielson's scheme, derivatives of order greater than one do not exist at the corners.

In this dissertation my work can be considered as an extension of Nielson's method with different blending function. My extension includes generalizing the blending functions to avoid the issue at the corners, allowing for higher continuity between patches, and generalizing to multi-sided patches.

2.6.2 Other Blending Schemes

Hagen and Pottmann [8] extended Nielson's method to second order continuity. Hagen and Pottmann's blending functions are

$$\begin{aligned} f_0^{HagenPottmann}(u,v,w) &= \frac{v^2w^2}{u^2v^2+v^2w^2+w^2u^2}, \\ f_1^{HagenPottmann}(u,v,w) &= \frac{w^2u^2}{u^2v^2+v^2w^2+w^2u^2}, \\ f_2^{HagenPottmann}(u,v,w) &= \frac{u^2v^2}{u^2v^2+v^2w^2+w^2u^2}. \end{aligned} \tag{2.9}$$

Hagen and Pottmann then used these blending functions to blend three patches, with the resulting surface having the same position, normals, and curvature along each boundary as one of the blended patches. The scheme provides the same locations and directions of the continuity as Nielson's scheme, but with the second order.

For surface blending schemes, if all of the sub-surfaces are triangular Bézier patches of the same degree, the result of the blending scheme can be considered as a type of triangular Bézier patch, with some or all of the control points "split" by the blending functions, with each split control point being replaced with multiple points, one for each blending function per split control point. In this case the surface patch is also called a *hybrid patch*, and the blending functions are called *dividing functions*. Many surface patches for the scattered data problem can be considered as being constructed in this way.

The *hybrid cubic triangle Bézier patch* provided by Foley and Opitz [5] can be constructed by splitting the center control point of a cubic triangular Bézier patch into three with Nielson's blending functions in Definition 2.3 as dividing functions.

The *Triangular Gregory patch* provided by Gregory [6] can be constructed by splitting each of the three center control points of a quartic triangular Bézier patch into two with the dividing functions, and used with the exterior control points of a cubic triangular Bézier patch. For each boundary there exists a pair of dividing functions. For example, for barycentric coordinates (u, v, w), Gregory's dividing functions for the boundary where $u = 0$ are

$$\begin{aligned} f_0^{Gregory}(u,v,w) &= \frac{v(1-w)}{v(1-w)+w(1-v)}, \\ f_1^{Gregory}(u,v,w) &= \frac{w(1-v)}{v(1-w)+w(1-v)}. \end{aligned} \tag{2.10}$$

The Gregory patch can also be extended to multi-sided generalisations [12]. Similarly, the dividing functions

$$\begin{aligned} f_0^{Hybrid}(u,v,w) &= \frac{(1-u)v^2}{(1-u)v^2+(1-v)u^2}, \\ f_1^{Hybrid}(u,v,w) &= \frac{(1-v)u^2}{(1-u)v^2+(1-v)u^2} \end{aligned} \tag{2.11}$$

are used to construct a triangular, parametric, and hybrid patch/scheme [15].

The following schemes have different blending/dividing functions and control point layouts. They do not satisfy the conditions of the blending schemes. However the ideas of these schemes are similar to the blending schemes. So they can be treated as hybrid schemes.

The *PNG1 triangle patch* provided by Christoph, Kerstin, Dianne, and Gerald [2] can be constructed by splitting the center control point of a cubic triangular Bézier patch into six, and splitting each of the six control point on the boundaries into two.

Herron's patch [11] can be constructed by using the interior control points of a quartic triangular Bézier patch and the exterior control points of a cubic triangular Bézier patch together. The result of this patch can be considered as blending the surfaces of three quartic triangular Bézier patch into one with with Nielson's blending functions in Definition 2.3, as indicated in [16].

Gregory and Charrot's scheme [7] blends three patches with Nielson's blending functions. Each of the three patches is a "two-sided interpolant" that interpolates the cross-boundary information along two boundaries constructed by a method called *boolean sum*.

Some of the dividing functions have similar properties as my new blending functions, and variations on these schemes using the generalized blending functions in this dissertation can be devised. It is also possible to construct new hybrid patches with the new results. For these schemes with the blending/dividing functions replaced by the new blending function provided in this dissertation, it is unclear if the resulting scheme will have any additional properties. However all of these modified schemes would keep their original properties (such as interpolation of position and derivatives along particular boundaries).

2.6.3 Usage of Blending Schemes

The idea of the blending schemes is to divide the task of satisfying the continuity conditions across three or more boundaries of a single surface into satisfying the continuity condition across a single boundary of three or more sub-surfaces. This division makes the construction of the surfaces more flexible. For example, when constructing surfaces with cubic triangular Bézier patches, there are not enough degrees of freedom for the control points to meet the C^1 continuity conditions across all three boundaries. However with surface blending, it is possible to construct three sub-patches, with each patch meeting the continuity conditions across one boundary. Then blending these three sub-patches with the correct blending functions yields a blended result that meets the continuity conditions across all three boundaries.

It is this property of separating the continuity requirements along all three boundaries of the patch into three, separate problems that I am pursuing in my work. My main contributions are to relax the conditions on the sub-patches and more importantly to generalize the method to arbitrary orders of continuity along the boundaries.

Chapter 3

Generalized Blending Scheme

In this chapter, in Definition 3.3 I present a template for constructing new blending functions, and in Definition 3.4 I present new blending functions that can be used to interpolate an arbitrary order of derivatives on the locations specified by certain helper functions. The bulk of this chapter is devoted to proving mathematical properties of these blending functions and the resulting functions derived from them, including derivative inheritance and showing when these blending functions are well-defined. In Chapter 4, I will use these blending functions to develop data interpolation schemes for triangular data meshes. Furthermore I also extend the triangular scheme to the multi-sided faces in Chapter 7.

3.1 Definitions

Recall that a blending scheme takes multiple functions as inputs and blends them into a new function as the output using specific weights. With respect to the blending scheme, the inputs are sub-functions, the weights used in the blending process are blending functions, and the output function is the blended result.

The purpose of the blending scheme is to construct a new function that has the same derivatives as the input functions at the desired boundaries or points. For example, for the surface blending in Chapter 4, the new surface has the same derivatives along the boundaries of the triangle. I refer to this property as inheritance in this thesis. There are three terms that specify the properties of the blending scheme and the inheritance:

- The blended terms: the number of sub-functions to be blended.

- The blending order: the desired order of derivatives that the blended result is supposed to inherit.

- The locations of the inheritance points and which corresponding sub-functions should be inherited.

The third term is specified by a set of functions called the inheritance indicators. Each sub-function has a corresponding inheritance indicator, and when an inheritance indicator is 0 at a point, then the derivatives of its corresponding sub-function are inherited at that point. To ensure that the blended result is continuous, all inheritance indicators should be in the same order of continuity. Furthermore, with respect to inheritance indicators, the inherited terms is the number of sub-functions that the derivatives will be inherited from at certain points.

The following symbols are used in the analyses of this chapter:

- Let n denote the blended terms.
- Let d denote the blending order.
- Let p_i denote the inheritance indicators.
- Let q^p denote the inherited terms.
- Let S_i denote the sub-functions.
- Let $f_i^{n,d,p}$ denote the blending functions.
- Let $S^{n,d,p}$ denote the blended result.

3.2 Generalized Blending Functions

In this section, I present my generalized blending functions. I give the definition of a set of intermediate helper functions in Definition 3.3. Using the intermediate functions as the templates and combining them with a set of specific approaches, I also give the definition of my generalized blending functions in Definition 3.4, and the formula of the blended result in Definition 3.5.

Assume that the blended terms (a positive integer n) and the blending order (a non-negative integer d) are given. Let N denote the index set $\{0, 1, \dots, n-1\}$. Without loss of generality, assume that all inheritance indicators and sub-functions are defined over a vector space of degree two. Let I denote a common domain of inheritance indicators and sub-functions. The formal definition of the inheritance indicators is given in Definition 3.1.

Definition 3.1. *For $n > 0$ and $d \geq 0$, the inheritance indicators $\{p_i : I \to \mathbb{R}\}_{i \in N}$ is a set of functions that satisfy the following conditions:*

- *All p_i are C^d-continuous.*
- *For each pair of distinct $i, j \in N$ and any point $(x, y) \in I$, if $p_i(x, y) + p_j(x, y) = 0$, then $p_i(x, y) = p_j(x, y) = 0$.*

The first condition guarantees that the blending functions and the blended result defined below are also C^d-continuous, and the second condition is set to avoid zero denominators in the blending functions when the inheritance indicators are all non-zero.

Let p denote the set of the inheritance indicators. In this thesis, the symbols $\{p_i\}_{i \in N}$ and p are used to denote arbitrary inheritance indicators, although I will use different symbols for specific cases. For example the symbol t is used to denote the inheritance indicators of the triangular surface blending scheme in Chapter 4, and the symbol b is used to denote the inheritance indicators of the binary case. The inherited terms q^p is defined in Definition 3.2.

Definition 3.2. *With respect to a set of given inheritance indicators p, for each point $(x, y) \in I$, the inherited terms $q^p : I \to \mathbb{N}$ (denoted $q^p(x, y)$) is the number of zero entries in p at (x, y).*

The intermediate functions $\{g_i^{n,d}\}_{i \in N}$ defined in Definition 3.3 below are determined by the blended terms n and the blending order d. The parameters of the g_i will be the values of the inheritance indicators at a certain point. These intermediate functions will not be used directly in the blending scheme, but can be considered as the templates for constructing blending functions with the given inheritance indicators.

Definition 3.3. *For $n > 0$ and $d \geq 0$, the intermediate functions $\{g_i^{n,d} : \mathbb{R}^n \to \mathbb{R}\}_{i \in N}$ are defined as*

$$g_i^{n,d}(v) = \Big(\prod_{j \in N \setminus \{i\}} w_j\Big)\Big(\sum_{j \in N \setminus \{i\}} \frac{1}{w_i + w_j}\Big) / \Big(\sum_{j,k \in N, j \neq k} \prod_{l \in N \setminus \{j,k\}} w_l\Big), \tag{3.1}$$

where $v = (v_0, v_1, \ldots, v_{n-1})$ and $w_i = v_i^{d+1}$ for each $i \in N$.

The blending functions $\{f_i^{n,d,p}\}_{i\in N}$ defined in Definition 3.4 below are constructed by merging the intermediate functions $\{g_i^{n,d}\}_{i\in N}$ and the inheritance indicators p together. Each $f_i^{n,d,p}$ can also be considered as the output of the intermediate function $g_i^{n,d}$ with respect to certain inheritance indicators p.

Definition 3.4. *For $n > 0$ and $d \geq 0$, with respect to a set of given inheritance indicators p, the blending functions $\{f_i^{n,d,p} : I \to \mathbb{R}\}_{i\in N}$ are defined as*

$$f_i^{n,d,p}(x,y) = \lim_{(x',y')\to(x,y)} g_i^{n,d}(p_0(x',y'), p_1(x',y'), \ldots, p_{n-1}(x',y')). \tag{3.2}$$

A typical example of the blending functions can be found in Definition 4.3.

At last, the blended result $S^{n,d,p}$ is defined in Definition 3.5, with respect to the blending functions $f_i^{n,d,p}$ and the sub-functions S_i.

Definition 3.5. *For $n > 0$ and $d \geq 0$, with respect to a set of given inheritance indicators p, the blended result $S^{n,d,p}$ is defined as*

$$S^{n,d,p}(x,y) = \sum_{i\in N} \left[f_i^{n,d,p}(x,y) S_i(x,y) \right], \tag{3.3}$$

where S_i are C^d-continuous functions defined over I.

3.3 Inheritance Propositions

Recall that the inheritance is the property of the blending scheme that the blended result $S^{n,d,p}$ shares the same derivatives as the sub-functions S_i at certain points, and the inheritance indicators p show the locations of these points and which corresponding sub-functions should be inherited. In the following sections the relationships between the inheritance indicators p and the blended result $S^{n,d,p}$ are specified and proven. In general, for a certain point $(x, y) \in I$, the blending scheme works with respect to the following rules:

- If the inherited terms $q^p(x, y) = 0$, then the blending functions $\{f_i^{n,d,p}\}_{i\in N}$ and the blended result $S^{n,d,p}$ are well-defined, and $S^{n,d,p}$ does not inherit derivatives from any sub-functions.

- If the inherited terms $q^p(x, y) = 1$ and the inheritance indicator $p_i(x, y) = 0$, then the blending functions $\{f_i^{n,d,p}\}_{i\in N}$ and the blended result $S^{n,d,p}$ are well-defined, and $S^{n,d,p}$ inherits derivatives from the sub-function S_i.

- If the inherited terms $q^p(x,y) = 2$ and the inheritance indicators $p_i(x,y) = p_j(x,y) = 0$ with $i \neq j$, then the blending functions $\{f_i^{n,d,p}\}_{i\in N}$ and the blended result $S^{n,d,p}$ are well-defined if the sub-functions S_i and S_j satisfy some extra conditions, and in that case $S^{n,d,p}$ inherits derivatives from both S_i and S_j.

- If the inherited terms $q^p(x,y) > 2$, then the blending functions $\{f_i^{n,d,p}\}_{i\in N}$ and the blended result $S^{n,d,p}$ are not well-defined.

As shown in Proposition 3.1 below, in the first case the blended result is always well-defined, and no inheritance exists. Also the last case should be avoided in the blending scheme since the blended result is always not well-defined. So the following analyses focus on the second and the third cases.

Proposition 3.1. *If the inherited terms $q^p(x,y) = 0$, then for each $i \in N$, the blending function $f_i^{n,d,p}$ is well-defined at (x,y).*

Proof. By Definition 3.1, all denominators of $f_i^{n,d,p}(x,y)$ are non-zero since none of the inheritance indicators are zero at (x,y). Thus $f_i^{n,d,p}$ is well-defined at (x,y). □

Before analysing each of the second and third cases, I give two propositions that will be used in the following sections. Proposition 3.2 shows that the sum of $\{g_i^{n,d}\}_{i\in N}$ is 1.

Proposition 3.2. *For each point $v \in \mathbb{R}^n$,*

$$\lim_{v'\to v} \sum_{i\in N} g_i^{n,d}(v') = 1. \tag{3.4}$$

Proof.

$$\begin{aligned}
&\lim_{v'\to v} \sum_{i=0}^{n-1} g_i^{n,d}(v') \\
&= \lim_{v'\to v} \Big\{ \sum_{i\in N} \Big[\Big(\prod_{j\in N\backslash\{i\}} w'_j \Big) \Big(\sum_{j\in N\backslash\{i\}} \frac{1}{w'_i + w'_j} \Big) \Big] / \Big(\sum_{j,k\in N, j\neq k} \prod_{l\in N\backslash\{j,k\}} w'_l \Big) \Big\} \\
&= \lim_{v'\to v} \Big\{ \Big(\sum_{i\in N} \sum_{j\in N\backslash\{i\}} \frac{\prod_{k\in N\backslash\{i\}} w'_k}{w'_i + w'_j} \Big) / \Big(\sum_{j,k\in N, j\neq k} \prod_{l\in N\backslash\{j,k\}} w'_l \Big) \Big\} \\
&= \lim_{v'\to v} \Big\{ \Big(\sum_{i,j\in N, i\neq j} \frac{\prod_{k\in N\backslash\{i\}} w'_k + \prod_{k\in N\backslash\{j\}} w'_k}{w'_i + w'_j} \Big) / \Big(\sum_{j,k\in N, j\neq k} \prod_{l\in N\backslash\{j,k\}} w'_l \Big) \Big\} \\
&= \lim_{v'\to v} \Big\{ \Big(\sum_{i,j\in N, i\neq j} \prod_{k\in N\backslash\{i,j\}} w'_k \Big) / \Big(\sum_{j,k\in N, j\neq k} \prod_{l\in N\backslash\{j,k\}} w'_l \Big) \Big\} \\
&= 1,
\end{aligned} \tag{3.5}$$

where $w'_i = (v'_i)^{d+1}$ for each $i \in N$. □

Proposition 3.3 shows that the sum of the blending functions $\{f_i^{n,d,p}\}_{i\in N}$ is 1.

Proposition 3.3. *For each point $(x, y) \in I$,*

$$\sum_{i\in N} f_i^{n,d,p}(x, y) = 1. \tag{3.6}$$

Proof. By Proposition 3.2,

$$\begin{aligned}\sum_{i\in N} f_i^{n,d,p}(x, y) &= \lim_{(x',y')\to(x,y)} \sum_{i\in N} g_i^{n,d}(p(x', y')) \\ &= \lim_{v'\to p(x,y)} \sum_{i\in N} g_i^{n,d}(v') \\ &= 1.\end{aligned} \tag{3.7}$$

□

3.3.1 Singular Inheritance

In this section I show that the blended result inherits derivatives from a single sub-function when the inherited terms q^p is 1. For all propositions in this section, it is assumed that the following conditions are satisfied. Notice that the index i is used to denote a certain value and the index j is used to denote a general value in I.

Condition 3.1. *For a point $(x, y) \in I$ and an index i,*

- *For each $j \in N$, p_j and S_j are d-th differentiable at (x, y).*
- *$p_i(x, y) = 0$.*
- *$q^p(x, y) = 1$.*

Proposition 3.4 shows that the blending functions $f_j^{n,d,p}$ are well-defined at the point (x, y).

Proposition 3.4. *For a point $(x, y) \in I$ and an index i that satisfy Condition 3.1, for each $j \in N$, $f_j^{n,d,p}$ is well-defined at (x, y).*

Proof. By Definition 3.1, all denominators of $f_j^{n,d,p}(x,y)$ are non-zero since there is only one inheritance indicator that is zero at (x,y). Thus $f_j^{n,d,p}$ is well-defined at (x,y). $\square$

Proposition 3.5 shows that the value and derivatives of order less than or equal to d of the blending functions $f_j^{n,d,p}$ are all zero at the point (x,y) if $i \neq j$.

Proposition 3.5. *For a point $(x,y) \in I$ and an index i that satisfy Condition 3.1, for each $j \in N \setminus \{i\}$ and all integers $0 \leq a+b \leq d$,*

$$\frac{d^{a+b}}{dx^a dy^b} f_j^{n,d,p}(x,y) = 0. \tag{3.8}$$

Proof. Since $f_j^{n,d,p}$ contains a multiplier p_i^{d+1}, all terms of the directional derivatives of $f_j^{n,d,p}$ of order less than or equal to d contain a multiplier p_i with power at least 1. Then since $p_i(x,y) = 0$, these directional derivatives are all zero at (x,y). $\square$

Proposition 3.6 shows that the value of the blending function $f_i^{n,d,p}$ is 1 at the point (x,y).

Proposition 3.6. *For a point $(x,y) \in I$ and an index i that satisfy Condition 3.1,*

$$f_i^{n,d,p}(x,y) = 1. \tag{3.9}$$

Proof. By Proposition 3.5, for each $j \in N \setminus \{i\}$,

$$f_j^{n,d,p}(x,y) = 0. \tag{3.10}$$

Then by Proposition 3.3,

$$\begin{aligned} f_i^{n,d,p}(x,y) &= \sum_{j \in N} f_j^{n,d,p}(x,y) - \sum_{j \in N \setminus \{i\}} f_j^{n,d,p}(x,y) \\ &= 1 - \sum_{j \in N \setminus \{i\}} f_j^{n,d,p}(x,y) \\ &= 1. \end{aligned} \tag{3.11}$$

$\square$

Proposition 3.7 shows that the derivatives of order greater than zero and less than or equal to d of the blending function $f_i^{n,d,p}$ are all zero at the point (x,y).

Proposition 3.7. *For a point $(x, y) \in I$ and an index i that satisfy Condition 3.1, for all integers $1 \leq a + b \leq d$,*

$$\frac{d^{a+b}}{dx^a dy^b} f_i^{n,d,p}(x, y) = 0. \tag{3.12}$$

Proof. By Proposition 3.3 and Proposition 3.5,

$$\begin{aligned} \frac{d^{a+b}}{dx^a dy^b} f_i^{n,d,p}(x, y) &= \frac{d^{a+b}}{dx^a dy^b} \Big[\sum_{j \in N} f_j^{n,d,p}(x, y) - \sum_{j \in N \setminus \{i\}} f_j^{n,d,p}(x, y) \Big] \\ &= \frac{d^{a+b}}{dx^a dy^b} \Big[1 - \sum_{j \in N \setminus \{i\}} f_j^{n,d,p}(x, y) \Big] \\ &= - \sum_{j \in N \setminus \{i\}} \frac{d^{a+b}}{dx^a dy^b} f_j^{n,d,p}(x, y) \\ &= 0. \end{aligned} \tag{3.13}$$

□

Proposition 3.8 shows that the blended result $S^{n,d,p}$ is well-defined at the point (x, y).

Proposition 3.8. *For a point $(x, y) \in I$ and an index i that satisfy Condition 3.1, $S^{n,d,p}$ is well-defined at (x, y).*

Proof. The result follows from Proposition 3.4. □

Proposition 3.9 shows that the blended result $S^{n,d,p}$ has the same value and derivatives of order less than or equal to d as S_i at the point (x, y). In other words, the proposition shows the inheritance of $S^{n,d,p}$ from the sub-functions S_i.

Proposition 3.9. *For a point $(x, y) \in I$ and an index i that satisfy Condition 3.1, for all integers $0 \leq a + b \leq d$,*

$$\frac{d^{a+b}}{dx^a dy^b} S^{n,d,p}(x, y) = \frac{d^{a+b}}{dx^a dy^b} S_i(x, y). \tag{3.14}$$

Proof. By Proposition 3.5, Proposition 3.6, and Proposition 3.7,

$$
\begin{aligned}
\frac{d^{a+b}}{dx^a dy^b} S^{n,d,p}(x,y) &= \sum_{j\in N} \left[\frac{d^{a+b}}{dx^a dy^b}(f_j^{n,d,p} S_j)(x,y)\right] \\
&= \sum_{j\in N}\sum_{\alpha=0}^{a}\sum_{\beta=0}^{b} \left[h(\alpha,\beta)(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_j^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_j)(x,y)\right] \\
&= \sum_{\alpha=0}^{a}\sum_{\beta=0}^{b} \left[h(\alpha,\beta)(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_i^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_i)(x,y)\right] \\
&= h(0,0)\frac{d^{a+b}}{dx^a dy^b} S_i(x,y) \\
&= \frac{d^{a+b}}{dx^a dy^b} S_i(x,y),
\end{aligned}
\tag{3.15}
$$

where $h(\alpha,\beta)$ is the number of time the mixed derivative repeats in the sum. □

3.3.2 Binary Inheritance

In this section I show that if some extra conditions are met, the blended result inherits derivatives from two sub-functions when the inherited terms q^p is two. For all propositions in this section, it is assumed that the following conditions are satisfied. Notice that the indices i and j are used to denote certain values and the index k is used to denote a general value in I.

Condition 3.2. *For a point $(x,y)\in I$ and two distinct indices i and j,*

- *For each $k\in N$, p_k and S_k are d-th differentiable at (x,y).*
- $p_i(x,y)=0$.
- $p_j(x,y)=0$.
- $q^p(x,y)=2$.

Proposition 3.10 shows that the blending function $f_k^{n,d,p}$ is well defined at the point (x,y) when $i\neq k$ and $j\neq k$.

Proposition 3.10. *For a point $(x,y)\in I$ and two distinct indices i and j that satisfy Condition 3.2, for each $k\in N\setminus\{i,j\}$, $f_k^{n,d,p}$ is well-defined at (x,y).*

Proof. By Definition 3.1, all denominators of $f_k^{n,d,p}(x,y)$ are non-zero since there are only two inheritance indicators p_i and p_j that are zero at (x,y) and $p_i(x,y)+p_j(x,y)$ is not included in the denominators of $f_k^{n,d,p}(x,y)$. Thus $f_j^{n,d,p}$ is well-defined at (x,y). $\square$

Proposition 3.11 shows that the sum of two blending functions $f_i^{n,d,p}$ and $f_j^{n,d,p}$ is well defined at the point (x,y).

Proposition 3.11. *For a point $(x,y) \in I$ and two distinct indices i and j that satisfy Condition 3.2, $f_i^{n,d,p}+f_j^{n,d,p}$ is well-defined at (x,y).*

Proof.

$$\begin{aligned}
&(f_i^{n,d,p}+f_j^{n,d,p})(x,y)\\
=\;& \lim_{(x',y')\to(x,y)}\Big[g_i^{n,d}(p(x',y'))+g_j^{n,d}(p(x',y'))\Big]\\
=\;& \lim_{(x',y')\to(x,y)}\Big[\Big(\prod_{k\in N\setminus\{i\}}p_k(x',y')\Big)\Big(\sum_{k\in N\setminus\{i\}}\frac{1}{p_i(x',y')+p_k(x',y')}\Big)\\
&+\Big(\prod_{k\in N\setminus\{j\}}p_k(x',y')\Big)\Big(\sum_{k\in N\setminus\{j\}}\frac{1}{p_j(x',y')+p_k(x',y')}\Big)\Big]\\
&/\Big(\sum_{k,l\in N,k\neq l}\prod_{m\in N\setminus\{k,l\}}p_m(x,y)\Big)\\
=\;& \lim_{(x',y')\to(x,y)}\Big[\Big(\prod_{k\in N\setminus\{i\}}p_k(x',y')\Big)\Big(\sum_{k\in N\setminus\{i,j\}}\frac{1}{p_i(x',y')+p_k(x',y')}\Big)\\
&+\Big(\prod_{k\in N\setminus\{j\}}p_k(x',y')\Big)\Big(\sum_{k\in N\setminus\{i,j\}}\frac{1}{p_j(x',y')+p_k(x',y')}\Big)\\
&+\prod_{k\in N\setminus\{i,j\}}p_k(x',y')\Big(\frac{p_j(x',y')}{p_i(x',y')+p_j(x',y')}+\frac{p_i(x',y')}{p_j(x',y')+p_i(x',y')}\Big)\Big]\\
&/\Big(\sum_{k,l\in N,k\neq l}\prod_{m\in N\setminus\{k,l\}}p_m(x,y)\Big)\\
=\;& \lim_{(x',y')\to(x,y)}\Big[\Big(\prod_{k\in N\setminus\{i\}}p_k(x',y')\Big)\Big(\sum_{k\in N\setminus\{i,j\}}\frac{1}{p_i(x',y')+p_k(x',y')}\Big)\\
&+\Big(\prod_{k\in N\setminus\{j\}}p_k(x',y')\Big)\Big(\sum_{k\in N\setminus\{i,j\}}\frac{1}{p_j(x',y')+p_k(x',y')}\Big)\\
&+\prod_{k\in N\setminus\{i,j\}}p_k(x',y')\Big]/\Big(\sum_{k,l\in N,k\neq l}\prod_{m\in N\setminus\{k,l\}}p_m(x,y)\Big)\\
=\;& \Big[\Big(\prod_{k\in N\setminus\{i\}}p_k(x,y)\Big)\Big(\sum_{k\in N\setminus\{i,j\}}\frac{1}{p_i(x,y)+p_k(x,y)}\Big)\\
&+\Big(\prod_{k\in N\setminus\{j\}}p_k(x,y)\Big)\Big(\sum_{k\in N\setminus\{i,j\}}\frac{1}{p_j(x',y')+p_k(x,y)}\Big)\\
&+\prod_{k\in N\setminus\{i,j\}}p_k(x,y)\Big]/\Big(\sum_{k,l\in N,k\neq l}\prod_{m\in N\setminus\{k,l\}}p_m(x,y)\Big).
\end{aligned}\tag{3.16}$$

All denominators of $(f_i^{n,d,p}+f_j^{n,d,p})(x,y)$ are non-zero, and thus $f_i^{n,d,p}+f_j^{n,d,p}$ is well-defined at (x,y). $\square$

Proposition 3.12 shows that the value and derivatives of order less than or equal to d of the blending function $f_k^{n,d,p}$ are all zero at the point (x, y) if $i \neq k$ and $j \neq k$.

Proposition 3.12. *For a point $(x, y) \in I$ and two distinct indices i and j that satisfy Condition 3.2, for each $k \in N \setminus \{i, j\}$ and all integers $0 \leq a + b \leq d$,*

$$\frac{d^{a+b}}{dx^a dy^b} f_k^{n,d,p}(x, y) = 0. \tag{3.17}$$

Proof. The result holds by the same method as in Proposition 3.5. □

Proposition 3.13 shows that the sum of two blending functions $f_i^{n,d,p}$ and $f_j^{n,d,p}$ is 1 at the point (x, y).

Proposition 3.13. *For a point $(x, y) \in I$ and two distinct indices i and j that satisfy Condition 3.2,*

$$(f_i^{n,d,p} + f_j^{n,d,p})(x, y) = 1. \tag{3.18}$$

Proof. By Proposition 3.12, for each $k \in N \setminus \{i, j\}$,

$$f_k^{n,d,p}(x, y) = 0. \tag{3.19}$$

Then by Proposition 3.3,

$$\begin{aligned}(f_i^{n,d,p} + f_j^{n,d,p})(x, y) &= \sum_{k \in N} f_k^{n,d,p}(x, y) - \sum_{k \in N \setminus \{i,j\}} f_k^{n,d,p}(x, y) \\ &= 1 - \sum_{k \in N \setminus \{i,j\}} f_k^{n,d,p}(x, y) \\ &= 1.\end{aligned} \tag{3.20}$$

□

Proposition 3.14 shows that the derivatives of order greater than zero and less than or equal to d of the sum of two blending functions $f_i^{n,d,p}$ and $f_j^{n,d,p}$ are all zero at the point (x, y).

Proposition 3.14. *For a point $(x, y) \in I$ and two distinct indices i and j that satisfy Condition 3.2, for all integers $1 \leq a + b \leq d$,*

$$\frac{d^{a+b}}{dx^a dy^b} (f_i^{n,d,p} + f_j^{n,d,p})(x, y) = 0. \tag{3.21}$$

Proof. By Proposition 3.3 and Proposition 3.12,

$$\begin{aligned}\frac{d^{a+b}}{dx^a dy^b}(f_i^{n,d,p}+f_j^{n,d,p})(x,y) &= \frac{d^{a+b}}{dx^a dy^b}\Big[\sum_{k\in N} f_k^{n,d,p}(x,y) - \sum_{k\in N\setminus\{i,j\}} f_k^{n,d,p}(x,y)\Big] \\ &= \frac{d^{a+b}}{dx^a dy^b}\Big[1 - \sum_{k\in N\setminus\{i,j\}} f_k^{n,d,p}(x,y)\Big] \\ &= -\sum_{k\in N\setminus\{i,j\}} \frac{d^{a+b}}{dx^a dy^b} f_k^{n,d,p}(x,y) \\ &= 0.\end{aligned} \tag{3.22}$$

□

Proposition 3.15 shows that the blended result $S^{n,d,p}$ is well-defined at the point (x, y) if an extra condition is satisfied.

Proposition 3.15. *For a point $(x, y) \in I$ and two distinct indices i and j that satisfy Condition 3.2, if $S_i(x, y) = S_j(x, y)$, then $S^{n,d,p}$ is well-defined at (x, y).*

Proof. By Proposition 3.12 and Proposition 3.13,

$$\begin{aligned}S^{n,d,p}(x,y) &= \sum_{k\in N}\Big[f_k^{n,d,p}(x,y)S_k(x,y)\Big] \\ &= f_i^{n,d,p}(x,y)S_i(x,y) + f_j^{n,d,p}(x,y)S_j(x,y) \\ &= (f_i^{n,d,p}+f_j^{n,d,p})(x,y)S_i(x,y) \\ &= S_i(x,y).\end{aligned} \tag{3.23}$$

Then $S^{n,d,p}$ is well-defined at (x, y). □

Proposition 3.16 shows that the blended result $S^{n,d,p}$ has the same value and derivatives of order less than or equal to d as both S_i and S_j when they share the same value and derivatives at the point (x, y). In other words, the proposition shows the inheritance of $S^{n,d,p}$ from the sub-functions S_i and S_j.

Proposition 3.16. *For a point $(x, y) \in I$ and two distinct indices i and j that satisfy Condition 3.2, if for all integers $0 \le a + b \le d$,*

$$\frac{d^{a+b}}{dx^a dy^b}S_i(x,y) = \frac{d^{a+b}}{dx^a dy^b}S_j(x,y), \tag{3.24}$$

then

$$\frac{d^{a+b}}{dx^a dy^b}S^{n,d,p}(x,y) = \frac{d^{a+b}}{dx^a dy^b}S_i(x,y) = \frac{d^{a+b}}{dx^a dy^b}S_j(x,y). \tag{3.25}$$

Proof. By Proposition 3.12, Proposition 3.13, and Proposition 3.14,

$$\begin{aligned}
&\frac{d^{a+b}}{dx^a dy^b} S^{n,d,p}(x,y) \\
&= \sum_{k\in N} \Big[\frac{d^{a+b}}{dx^a dy^b} (f_k^{n,d,p} S_k)(x,y) \Big] \\
&= \sum_{k\in N} \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \Big\{ h_1(\alpha,\beta) \\
&\quad \Big[(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_k^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_k)(x,y) \Big] \Big\} \\
&= \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \Big\{ h_1(\alpha,\beta) \\
&\quad \Big[(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_i^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_i)(x,y) \\
&\quad +(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_j^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_j)(x,y) \Big] \Big\} \\
&= \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \Big\{ h_1(\alpha,\beta) \\
&\quad \Big[(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_i^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_i)(x,y) \\
&\quad +(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_j^{n,d,p})(x,y)(\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_j)(x,y) \Big] \Big\} \\
&= \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \Big\{ h_1(\alpha,\beta) h_2(\alpha,\beta) \\
&\quad \Big[(\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_i^{n,d,p})(x,y) + (\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} f_j^{n,d,p})(x,y) \Big] \Big\} \\
&= \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \Big\{ h_1(\alpha,\beta) h_2(\alpha,\beta) \Big[\frac{d^{\alpha+\beta}}{dx^\alpha dy^\beta} (f_i^{n,d,p} + f_j^{n,d,p})(x,y) \Big] \Big\} \\
&= h_1(0,0) h_2(0,0) \\
&= \frac{d^{a+b}}{dx^a dy^b} S_i(x,y),
\end{aligned} \tag{3.26}$$

where $h_1(\alpha,\beta)$ is the number of time the mixed derivative repeats in the sum, and

$$h_2(\alpha,\beta) = (\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_i)(x,y) = (\frac{d^{a+b-\alpha-\beta}}{dx^{a-\alpha} dy^{b-\beta}} S_j)(x,y). \tag{3.27}$$

Then

$$\frac{d^{a+b}}{dx^a dy^b} S^{n,d,p}(x, y) = \frac{d^{a+b}}{dx^a dy^b} S_i(x, y) = \frac{d^{a+b}}{dx^a dy^b} S_j(x, y). \tag{3.28}$$

□

3.4 Summary

In this chapter, I defined new blending functions in Definition 3.4 and the blended result in Definition 3.5. I then showed that the blended result inherits the derivatives from the sub-functions based on the distributions of zero values of the inheritance indicators. These results are used in this dissertation multiple times.

- In Chapter 4, I will give typical inheritance indicators and use these blending functions to construct a triangular surface blending scheme, and compare my new scheme with Nielson's scheme.
- In Chapter 5, I will use both triangular and binary schemes to design an interpolation algorithm to solving the triangular scattered data interpolation problem.
- In Chapter 7, I will extend the triangular surface blending scheme to the multi-sided version, and give the updated algorithm.

Chapter 4

Triangular Blending Scheme

In this chapter, I will present a triangular surface blending scheme using the blending functions generating templates defined in Chapter 3, and present the typical definitions and propositions with respect to the triangular case.

4.1 Definitions

Most of the variables and functions used in this chapter have the same symbols as given in Section 3.1. In this chapter these variables and functions are now used in a specific blending scheme, which sets the values and names of some of the variables and functions as follows:

- For the triangular case, $n = 3$.
- The symbol $t = (t_0, t_1, t_2)$ will be used to denote the triangular inheritance indicators.
- S_0, S_1, and S_2 are the sub-functions. In this chapter they are surfaces, so they are also called the sub-surfaces.
- q^t is the inherited terms with respect to the triangular inheritance indicators t.
- $f_0^{3,d,t}$, $f_1^{3,d,t}$, and $f_2^{3,d,t}$ are the triangular blending functions.
- $S^{3,d,t}$ is the blended result of the triangular surface blending scheme.

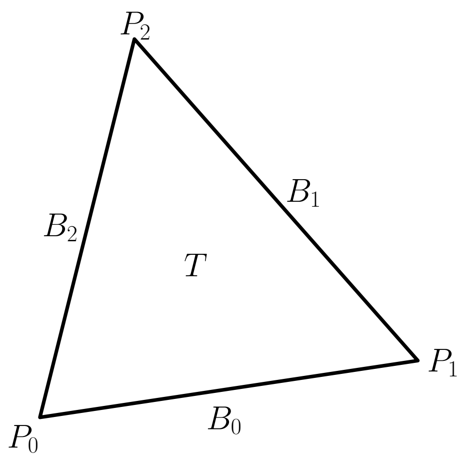


Figure 4.1: The layout of the blending domain T.

For the blending scheme with respect to the surfaces, the portion of the blended result that is formed by the interpolated points and boundaries should be considered since the continuity conditions are established with respect to its boundaries and corners. The corresponding subset of its domain is called a blending domain. For the triangular case, the corresponding blending domain is defined in Definition 4.1.

Definition 4.1. *A triangular blending domain T is a closed subset of the vector space on which the inheritance indicators, blending functions, and the sub-surfaces are defined. T is formed by three points P_0, P_1, and P_2, and three boundaries B_0, B_1, and B_2. The two endpoints of B_0 are P_0 and P_1, the two endpoints of B_1 are P_1 and P_2, and the two endpoints of B_2 are P_2 and P_0.*

Notice that the boundaries in Definition 4.1 are not restricted to be straight. However in this chapter they are assumed to be straight. More specifically, the blending domain T in this chapter is the interior part of the barycentric coordinate domain. Recall that the barycentric coordinates of a point inside the triangular domain is usually represented by (u, v, w) where $u + v + w = 1$. Without loss of generality, assume that P_0 has barycentric coordinates $(1, 0, 0)$, P_1 has barycentric coordinates $(0, 1, 0)$, and P_2 has barycentric coordinates $(0, 0, 1)$. The layout of T is shown in Figure 4.1.

Also notice that the blend result of the triangular surface blending scheme is always at least C^d-continuous in the interior part of T. So all of the discussions and proofs focus on the boundaries and corners.

4.2 Triangular Blending Scheme

With respect to T, to fulfill the goal of the triangular blending scheme, a set of inheritance indicators need to be designed with two conditions. The first condition is that the inheritance indicators are defined over T and the second condition is that each inheritance indicator is zero only on the corresponding boundary. The triangular inheritance indicator t is defined in Defintion 4.2.

Definition 4.2. *The triangular inheritance indicator $t = (t_0, t_1, t_2)$ is defined as*

$$\begin{aligned} t_0(u,v,w) &= u, \\ t_1(u,v,w) &= v, \\ t_2(u,v,w) &= w. \end{aligned} \tag{4.1}$$

Recall that as defined in Definition 3.2, the inherited terms q^t is the function that shows the number of zero inheritance indicators at certain points. Thus the values of q^t with respect to the locations of the points are determined.

Proposition 4.1. *The value of the inherited terms q^t in the blending domain T is*

- *$q^t(u,v,w) = 0$ if (u,v,w) is located inside the interior part of T.*
- *$q^t(u,v,w) = 1$ if (u,v,w) is located on a boundary (excluding the endpoints) of T.*
- *$q^t(u,v,w) = 2$ if (u,v,w) is located at a corner of T.*

Proof. The result follows from the definition of q^t. □

With the triangular inheritance indicators, the triangular blending functions can be expanded from Definition 4.3:

Definition 4.3. *The d-th triangular blending functions with respect to the triangular inheritance indicators t are*

$$\begin{aligned} f_0^{3,d,t}(u,v,w) &= \beta\gamma\left(\frac{1}{\alpha+\beta} + \frac{1}{\alpha+\gamma}\right)\left(\frac{1}{\alpha+\beta+\gamma}\right), \\ f_1^{3,d,t}(u,v,w) &= \gamma\alpha\left(\frac{1}{\beta+\gamma} + \frac{1}{\beta+\alpha}\right)\left(\frac{1}{\alpha+\beta+\gamma}\right), \\ f_2^{3,d,t}(u,v,w) &= \alpha\beta\left(\frac{1}{\gamma+\alpha} + \frac{1}{\gamma+\beta}\right)\left(\frac{1}{\alpha+\beta+\gamma}\right), \end{aligned} \tag{4.2}$$

where $\alpha = u^{d+1}$, $\beta = v^{d+1}$, and $\gamma = w^{d+1}$.

I make the following two definitions that state the positions and derivatives are matched at the corners. They are the two conditions that ensure the existence of the blended result and its derivatives at the corners.

Definition 4.4. *The sub-surfaces S_0, S_1 and S_2 are value matched with respect to the blending domain T if each pair of adjacent sub-surfaces share the same values at corresponding corners:*

$$\begin{aligned} S_0(P_1) &= S_1(P_1), \\ S_1(P_2) &= S_2(P_2), \\ S_2(P_0) &= S_0(P_0). \end{aligned} \tag{4.3}$$

Definition 4.5. *The sub-surfaces S_0, S_1 and S_2 are d-th matched with respect to the blending domain T if each pair of adjacent sub-surfaces share the same derivatives of order less than or equal to d at corresponding corners. More specifically, for all integers $0 \leq a+b \leq d$,*

$$\begin{aligned} \frac{d^{a+b}}{du^a dv^b d} S_0(P_1) &= \frac{d^{a+b}}{du^a dv^b} S_1(P_1), \\ \frac{d^{a+b}}{du^a dv^b d} S_1(P_2) &= \frac{d^{a+b}}{du^a dv^b} S_2(P_2), \\ \frac{d^{a+b}}{du^a dv^b d} S_2(P_0) &= \frac{d^{a+b}}{du^a dv^b} S_0(P_0). \end{aligned} \tag{4.4}$$

The blended result $S^{3,d,t}$ is then defined as in Definition 4.6.

Definition 4.6. *With three sub-surfaces S_0, S_1, and S_2 defined over the blending domain T, the blended result of the d-th triangular blending scheme with respect to the triangular inheritance indicators t is defined as*

$$S^{3,d,t}(u,v,w) = \sum_{i=0}^{2} \left[f_i^{3,d,t}(u,v,w) S_i(u,v,w) \right]. \tag{4.5}$$

The following theorems are given based on the propositions in Chapter 3 with respect to the surface blending. Theorem 4.1 shows that the blended result $S^{3,d,t}$ is well-defined if the sub-surfaces are value matched with respect to the blending domain.

Theorem 4.1. *If the sub-surfaces S_0, S_1, and S_2 are value matched with respect to the blending domain T, then the blended result $S^{3,d,t}$ is well-defined.*

Proof. The result follows Proposition 3.8 and Proposition 3.15. □

Theorem 4.2 shows that the blended result inherits values on boundaries from corresponding sub-surfaces.

Theorem 4.2. *The blended result $S^{3,d,t}$ inherits the derivatives of order less than or equal to d from a sub-surface on the corresponding boundary excluding the endpoints. More specifically, for each $i \in \{0,1,2\}$, a non-corner point $P \in B_i$, and all integers $0 \le a+b \le d$,*

$$\frac{d^{a+b}}{du^a dv^b} S^{3,d,t}(P) = \frac{d^{a+b}}{du^a dv^b} S_i(P). \tag{4.6}$$

Proof. By Proposition 4.1, $q^t(P) = 1$. Then the result follows Proposition 3.9. $\square$

Theorem 4.3 shows that the blended result inherits values at the corners from corresponding adjacent pairs of sub-surfaces when the sub-surfaces are d-th matched with respect to the blending domain.

Theorem 4.3. *If the sub-surfaces S_0, S_1, and S_2 are d-th matched with respect to the blending domain T, then the blended result $S^{3,d,t}$ has well-defined derivatives of order less than or equal to d in T, and inherits the derivatives of order less than or equal to d from both adjacent sub-surfaces at the corresponding corner. More specifically, for all integers $0 \le a+b \le d$,*

$$\begin{aligned}
\frac{d^{a+b}}{du^a dv^b} S^{3,d,t}(P_1) &= \frac{d^{a+b}}{du^a dv^b} S_0(P_1) = \frac{d^{a+b}}{du^a dv^b} S_1(P_1),\\
\frac{d^{a+b}}{du^a dv^b} S^{3,d,t}(P_2) &= \frac{d^{a+b}}{du^a dv^b} S_1(P_2) = \frac{d^{a+b}}{du^a dv^b} S_2(P_2),\\
\frac{d^{a+b}}{du^a dv^b} S^{3,d,t}(P_0) &= \frac{d^{a+b}}{du^a dv^b} S_2(P_0) = \frac{d^{a+b}}{du^a dv^b} S_0(P_0).
\end{aligned} \tag{4.7}$$

Proof. By Proposition 4.1, $q^t(P_0) = q^t(P_1) = q^t(P_2) = 2$. Then the result follows Proposition 3.16. $\square$

The previous theorems are given with respect to parametric properties. Furthermore, some geometric properties are given. The following discussion is based on the concept of "geometric value" in Section 2.2. Recall that the geometric values are the locations for the zero-th order, the normals for the first order, and the curvatures for the second order. The locations always exist when the blended result is well-defined. The following two theorems state the geometric properties of the blended surfaces with respect to the normals and curvatures.

Theorem 4.4. *Assume a blending order $d \geq 1$ is given, and each pair of adjacent sub-surfaces share the same normals at the corresponding corner. More specifically, assume*

- *The normals of the sub-surfaces S_0 and S_1 are same at the point P_1.*
- *The normals of the sub-surfaces S_1 and S_2 are same at the point P_2.*
- *The normals of the sub-surfaces S_2 and S_0 are same at the point P_0.*

Then the blended result $S^{3,d,t}$ has well-defined normals in the blending domain T, and the normals of $S^{3,d,t}$ are continuous. $S^{3,d,t}$ inherits the normals at the corners from the corresponding pair of adjacent sub-surfaces.

Proof. The blended result $S^{3,d,t}$ inherits the first order derivatives from a sub-surface on the corresponding boundary, so it also inherits the normals from the sub-surface. For a corner, the limit of its normal along the direction of a boundary is the normal at the corner of the corresponding sub-surface. Then since the normals of the two adjacent sub-surfaces at that corner are same, then the normal of the blended result exist at that corner, and it is continuous. The value of the normal at that corner is inherited from the corresponding pair of adjacent sub-surfaces. $\square$

Theorem 4.5. *Assume a blending order $d \geq 2$ is given, and each pair of adjacent sub-surfaces share the same curvatures at the corresponding corner. More specifically, assume*

- *The curvatures of the sub-surfaces S_0 and S_1 are same at the point P_1.*
- *The curvatures of the sub-surfaces S_1 and S_2 are same at the point P_2.*
- *The curvatures of the sub-surfaces S_2 and S_0 are same at the point P_0.*

Then the blended result $S^{3,d,t}$ has well-defined curvatures in the blending domain T, and the curvatures of $S^{3,d,t}$ are continuous. $S^{3,d,t}$ inherits the curvatures at the corners from the corresponding pair of adjacent sub-surfaces.

Proof. The blended result $S^{3,d,t}$ inherits the first and second order derivatives from a sub-surface on the corresponding boundary, so it also inherits the curvatures from the sub-surface. For a corner, the limit of the curvature of the surface along the direction of a boundary is the curvature at the corner of the corresponding sub-surface. Then since the curvatures of the two adjacent sub-surfaces at that corner are same, then the curvature of the blended result exists at that corner, and it is continuous. The value of the normal at that corner is inherited from the corresponding pair of adjacent sub-surfaces. $\square$

With these definitions and theorems, I can state the triangular surface blending scheme here. With a given blending order d, the d-th triangular surface blending scheme accepts three triangular sub-surfaces S_0, S_1, and S_2 parameterized by the barycentric coordinates (defined over T) as inputs. It then construct a blended result $S^{3,d,t}$ in Definition 4.6 by applying the triangular blending functions in Definition 4.3 as the output.

The output blended result surface $S^{3,d,t}$ has different properties for different conditions that the input sub-surfaces meet.

- $S^{3,d,t}$ is well-defined in the barycentric coordinates except the corners.
- $S^{3,d,t}$ inherits the derivatives of order less than or equal to d from a sub-surface on the corresponding boundary of the barycentric coordinates excluding the endpoints.
- If S_0, S_1, and S_2 are value matched, then $S^{3,d,t}$ is well-defined at the corners of the barycentric coordinates.
- If S_0, S_1, and S_2 are d-th matched, then $S^{3,d,t}$ has well-defined derivatives of order less than or equal to d in the barycentric coordinates, and inherits the derivatives of order less than or equal to d from both adjacent sub-surfaces at the corresponding corner.

With respect to the geometric values, the blended result $S^{3,d,t}$ also has the following properties:

- If $d \geq 1$ and each pair of adjacent sub-surfaces share the same locations and normals at the corresponding corner, then $S^{3,d,t}$ has well-defined normals in the barycentric coordinates, and the normals of $S^{3,d,t}$ are continuous. $S^{3,d,t}$ inherits the normals at the corners from the corresponding pair of adjacent sub-surfaces.
- If $d \geq 2$ and each pair of adjacent sub-surfaces share the same locations, normals, and curvatures at the corresponding corner, then $S^{3,d,t}$ has well-defined curvatures in the barycentric coordinates, and the curvatures of $S^{3,d,t}$ are continuous. $S^{3,d,t}$ inherits the curvatures at the corners from the corresponding pair of adjacent sub-surfaces.

4.3 Comparison to Nielson's Scheme

In this section the new triangular surface blending scheme is compared with Nielson's method. For any $d \geq 1$, the new triangular surface blending scheme constructs surfaces with the the normals inherited from the sub-surfaces. In that case the resulting surfaces have the same properties as Nielson's. The difference between them is the conditions of the sub-surfaces. Recall that as in Condition 2.1, for three vertices V_0, V_1, and V_2 as well as three sub-surfaces S_0, S_1, and S_2, the conditions for Nielson's method are

- S_0, S_1, and S_2 interpolate the locations of V_0, V_1, and V_2.
- S_0, S_1, and S_2 share the same boundaries.
- S_0 interpolates the normals of V_1 and V_2.
- S_1 interpolates the normals of V_2 and V_0.
- S_2 interpolates the normals of V_0 and V_1.

The conditions for the new triangular surface blending scheme are

- S_0 and S_1 interpolate the location and normal of V_1.
- S_1 and S_2 interpolate the location and normal of V_2.
- S_2 and S_0 interpolate the location and normal of V_0.

The new scheme has fewer restrictions on the sub-surfaces. The reason is that Nielson's blending functions contain denominators $uv + vw + wu$ and a problem occurs when the barycentric coordinates of a point has two zero entries when located at a corner of T. In that case the values of the denominators are zero, and the blended result does not exist. However, if all three sub-surfaces used in Nielson's method share the same values at all three corners, then the blended result is well-defined. For the new scheme, the denominators of the new blending functions have the property that for any point in the blending domain, at least of one of the blending functions does not have zero denominators. For example, at the corner $(1, 0, 0)$, all denominators $\alpha+\beta$, $\alpha+\gamma$, and $\alpha+\beta+\gamma$ of the blending function $f_0^{3,d,t}$ in Definition 4.3 are non-zero. That leads to a result that the blended result exists when the two corresponding sub-surfaces of the other two blending functions share the same values at that corner. For the new scheme, the blended result is well-defined everywhere if the sub-surfaces are value matched with respect to all three corners of T.

The reduced conditions lead to a possibility to construct the triangular blending algorithm in Chapter 5. Consider two triangular faces from a data mesh shown in Figure 4.2, formed by vertices $V_0V_1V_2$ and $V_0V_2V_3$. To construct a smooth surface for such a data

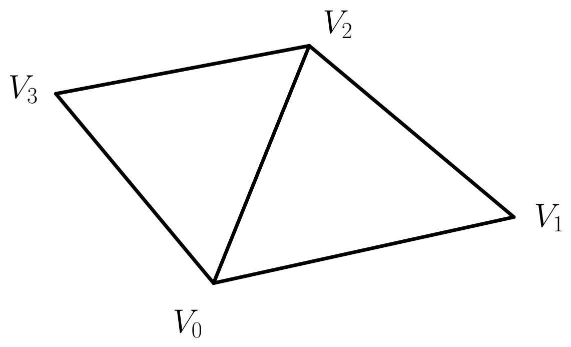


Figure 4.2: Two adjacent faces from a data mesh.

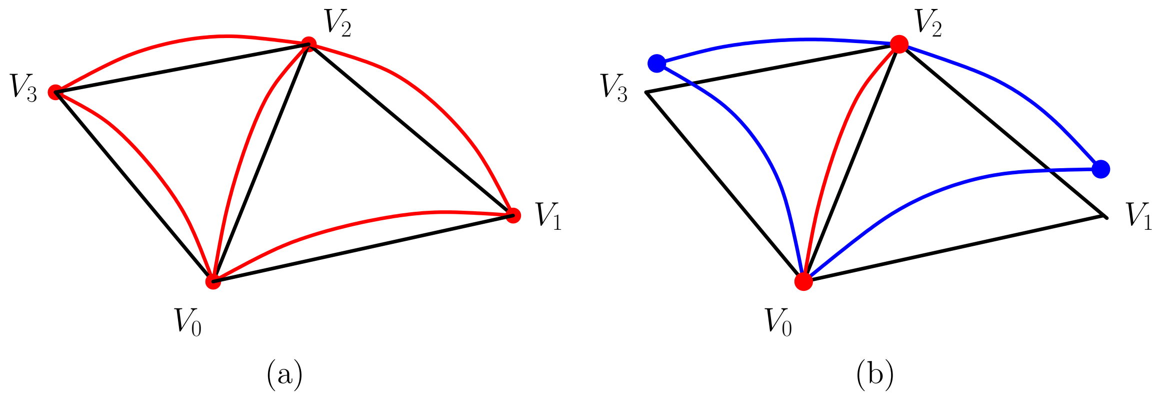


Figure 4.3: The differences between the requirements for the sub-surfaces; (a) Nielson's method; (b) new scheme.

mesh, the resulting surfaces for these two faces need to join smoothly, thus the two sub-surfaces with respect the edge $E = \{V_0, V_2\}$ also need to join smoothly. Figure 4.3 shows the differences between the requirements of constructing sub-surfaces for Nielson's scheme and the new scheme.

As shown in Figure 4.3(a), the two sub-surfaces for the two faces with respect to their common edge are shown by the red points and curves. According to the conditions for Nielson's method, the three sub-surfaces of a face share the same boundaries. Then all red curves are restricted. At least two steps are required to construct these sub-surfaces: determine the common boundaries interpolating the vertices first, then construct sub-surfaces interpolating these boundaries and for each edge ensure the two sub-surfaces from both side are joined smoothly. As a result, the two sub-surfaces are restricted by all four vertices and five edges. These vertices and edges also influence sub-surfaces of adjacent faces.

Figure 4.3(b) shows the requirements for the new blending scheme. According to the new conditions, the boundaries of the three sub-surfaces of a face do not need to match and for each of them the number of vertices that need to be interpolated is reduced to two. That leads to a result that only the boundaries and points coloured by red are restricted, and the boundaries and points coloured by blue have extra degrees of freedom. Since the red part only influences these two sub-surfaces, they can be constructed "independently".

Furthermore their process of constructing can be reversed. It is possible to construct a smooth surface interpolating the two vertices V_0 and V_2, then cut two joined triangular pieces as the sub-surfaces from them, instead of constructing two triangular sub-surfaces interpolating the vertices and ensure them to be joined smoothly. The algorithm presented in Chapter 5 is constructed based on this idea. In the new algorithm the blending of three triangular sub-surfaces can be considered as the blending of three larger surfaces that are constructed with respect to each edge. They can be treated as three quadrangular surfaces. Figure 4.4 shows this process of blending for a single triangular face. The red, green, and blue surfaces are constructed over pairs of triangle faces, one for each corresponding edge. The solid portions are the triangular surfaces that are actually blended for the triangular face.

4.3.1 Blending Functions

Nielson's blending function can be extended to increase the degree of continuity (an example is the blending functions used by Hagen and Pottmann in Equation 2.9). For the triangular case, the formula of the extended Nielson's functions are

$$\begin{aligned} f_0^{Nielson,d}(u,v,w) &= \frac{\beta\gamma}{\alpha\beta+\beta\gamma+\gamma\alpha}, \\ f_1^{Nielson,d}(u,v,w) &= \frac{\gamma\alpha}{\alpha\beta+\beta\gamma+\gamma\alpha}, \\ f_2^{Nielson,d}(u,v,w) &= \frac{\alpha\beta}{\alpha\beta+\beta\gamma+\gamma\alpha}, \end{aligned} \tag{4.8}$$

where $\alpha = u^{d+1}$, $\beta = v^{d+1}$, and $\gamma = w^{d+1}$. Note that a straightforward generalization of Nielson's blending functions would use $\alpha = u^d$, $\beta = v^d$, and $\gamma = w^d$. For use in my scheme, the powers need to be one higher for the same order of continuity.

The extended Nielson's blending functions satisfies the conditions in Chapter 4 (for the triangular case, each one of them is well-defined at a corresponding corner). That

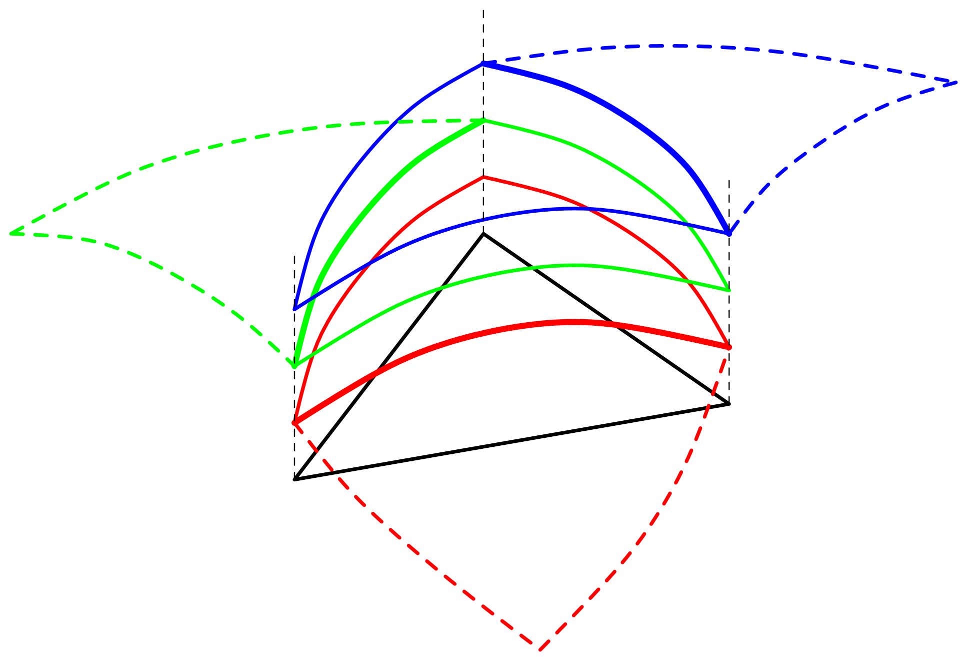


Figure 4.4: The extended triangular surface blending scheme.

means the extended Nielson's blending functions may also be used in place of my blending functions in the algorithms introduced in the following chapters. I discuss this further in Section 9.1.2.

The reason that I did not use the extended Nielson's blending functions is that for same cases, the blending functions may need to be extended again (for example the shape adjusting in Section 9.1.3). The power of the parameters may be unbalanced. The functions

$$\begin{aligned} f_0(u,v,w) &= \frac{v^2w^2}{u^2v^4+v^2w^2+w^4u^2}, \\ f_1(u,v,w) &= \frac{w^4u^2}{u^2v^4+v^2w^2+w^4u^2}, \\ f_2(u,v,w) &= \frac{u^2v^4}{u^2v^4+v^2w^2+w^4u^2} \end{aligned} \tag{4.9}$$

with different powers are an example. At the corner where $(u,v,w) = (1,0,0)$ all three functions and their limits are not well-defined. However for the new triangular blending functions in this chapter, there exist variations

$$\begin{aligned} f_0(u,v,w) &= v^2w^2\Big(\frac{1}{u^2+v^2}+\frac{1}{u^2+w^2}\Big)\Big(\frac{1}{u^2+v^4+w^4}\Big), \\ f_1(u,v,w) &= w^4u^2\Big(\frac{1}{v^4+w^4}+\frac{1}{v^2w^2+u^2w^2}\Big)\Big(\frac{1}{u^2+v^4+w^4}\Big), \\ f_2(u,v,w) &= u^2v^4\Big(\frac{1}{w^2v^2+u^2v^2}+\frac{1}{w^4+v^4}\Big)\Big(\frac{1}{u^2+v^4+w^4}\Big), \end{aligned} \tag{4.10}$$

and

$$\begin{aligned} f_0(u,v,w) &= v^2w^2\Big(\frac{w^2}{u^2+v^2}+\frac{v^2}{u^2+w^2}\Big)\Big(\frac{1}{u^2+v^4+w^4}\Big), \\ f_1(u,v,w) &= w^4u^2\Big(\frac{1}{v^4+w^4}+\frac{1}{v^2+u^2}\Big)\Big(\frac{1}{u^2+v^4+w^4}\Big), \\ f_2(u,v,w) &= u^2v^4\Big(\frac{1}{w^2+u^2}+\frac{1}{w^4+v^4}\Big)\Big(\frac{1}{u^2+v^4+w^4}\Big). \end{aligned} \tag{4.11}$$

Both sets of them always have one well-defined function or limit for each corner of the domain. So my new blending functions always satisfy the requirement of having at least one well-defined basis function at the corners, while some variations of the extended Nielson blending functions to not have this property. That means the new blending functions are more possible to be further extended. Such property is not a big advantage, but might be useful in the future plan.

Chapter 5

Triangular Blending Algorithm

In Section 4.2, I gave a method of constructing a triangular patch that interpolates position and derivatives at the corners and boundaries of a triangle. In this chapter, I will extend the method to solve the triangular scattered data interpolation problem, and show how to construct a set of patches that interpolate the vertices of a mesh, where adjacent patches meet smoothly. I call this algorithm the *basic triangular blending algorithm*.

The input of the algorithm is a triangular data mesh, and the output of the algorithm is a surface that interpolates the vertices of the data mesh. The resulting surface of the algorithm is a combination of a group of blended results, in which each blended result is the output of a triangular surface blending scheme with respect to a face of the data mesh. The continuity (of arbitrary order) across the boundaries of the adjacent pairs of blended results and at the corners of the blended results is fulfilled by constructing and dividing intermediate surfaces across the boundaries.

In Section 5.1 I give the definitions of the symbols used in the algorithm and the reparameterization functions. Then in Section 5.2 I give the complete process of the algorithm. At last in the Section 5.3 I show that the algorithm constructs surfaces with continuous normals (with $d \geq 1$), and surfaces with continuous curvatures (with $d \geq 2$) everywhere.

5.1 Definitions

In the algorithm introduced in this chapter, each element (vertex, edge, or face) of the data mesh has a corresponding coordinate system, and a surface constructed with respect to it. These surfaces may need to be reparameterized by the coordinate systems of other elements. In this thesis, two elements are said to be parametrically adjacent if in the algorithm there is reparameterization between their coordinate systems. More specifically,

- a vertex is parametrically adjacent to its neighbouring edges.
- an edge is parametrically adjacent to its neighbouring vertices and faces.
- a face is parametrically adjacent to its neighbouring edges.

The surface of an element may need to be reparameterized by the coordinate system of another element, and these two elements may not be parametrically adjacent. For example the surface of a face may need to be reparameterized by the coordinate system of a neighbouring vertex. Such reparameterization cannot be done directly. The face surface needs to be reparameterized by the coordinate system of a common neighbouring edge, then by the coordinate systems of the vertex. The path of the reparameterization may contain more than three elements. To describe these relationships, the following definitions are given.

- Let $\mathcal{F}^X$ denote the coordinate system of an element X.
- Let S^X denote the surface of an element X.
- If two elements X and Y are parametrically adjacent, let S^X_Y denote the surface of X reparameterized by $\mathcal{F}^Y$.
- If each pair of adjacent elements in the sequence $X, Z_0, Z_1, \ldots, Z_{n-1}, Y$ are parametrically adjacent, let $S^X_{Z_0,Z_1,\ldots,Z_{n-1},Y}$ denote the surface of X reparameterized via the path $X \to Z_0 \to Z_1 \to \cdots \to Z_{n-1} \to Y$, at last reparameterized by $\mathcal{F}^Y$.

Furthermore, in the algorithm all reparameterizations are bijective. Now the following functions are defined with respect to the reparameterization methods used in the algorithm.

- If two elements X and Y are parametrically adjacent, let $r^{X,Y}$ denote the function that maps the points in $\mathcal{F}^X$ into $\mathcal{F}^Y$.
- If each pair of adjacent elements in the sequence $X, Z_0, Z_1, \ldots, Z_{n-1}, Y$ are parametrically adjacent, let $r^{X,Z_0,Z_1,\ldots,Z_{n-1},Y}$ denote the function that maps the points in $\mathcal{F}^X$ via the path $X \to Z_0 \to Z_1 \to \cdots \to Z_{n-1} \to Y$, at last into $\mathcal{F}^Y$:

$$r^{X,Z_0,Z_1,\ldots,Z_{n-1},Y} = r^{Z_{n-1},Y} \circ \cdots \circ r^{Z_0,Z_1} \circ r^{X,Z_0}. \tag{5.1}$$

From the definitions,

- If two elements X and Y are parametrically adjacent, then

$$S_Y^X(x,y) = S^X(r^{Y,X}(x,y)). \tag{5.2}$$

- If each pair of adjacent elements in the sequence $X, Z_0, Z_1, \ldots, Z_{n-1}, Y$ are parametrically adjacent, then

$$S^X_{Z_0,Z_1,\ldots,Z_{n-1},Y}(x,y) = S^X(r^{Y,Z_{n-1},\ldots,Z_1,Z_0,X}(x,y)). \tag{5.3}$$

Notice that all of the definitions are given for the parametric interpolation. For a functional interpolation all surfaces can be constructed with the coordinates with the same plane and no reparameterization is required (except the reparameterization between the local and barycentric coordinate systems).

5.2 The Algorithm

The idea of the algorithm is to construct the sub-surfaces of the triangular blending scheme by splitting the surfaces constructed with respect to the edges, and construct the edge surfaces by blending adjacent vertex surfaces. Thus the algorithm contains five steps:

1. For each vertex V, construct a local coordinate system $\mathcal{F}^V$ and the surface S^V.
2. For each edge $E = \{V_0, V_1\}$, construct a local coordinate system $\mathcal{F}^E$ and reparameterize the surfaces S^{V_0} and S^{V_1} to obtain $S_E^{V_0}$ and $S_E^{V_1}$.
3. For each edge $E = \{V_0, V_1\}$, construct the surface $S^E = S^{\{V_0,V_1\}}$ by blending $S_E^{V_0}$ and $S_E^{V_1}$.
4. For each face $F = \{V_0, V_1, V_2\}$, construct a barycentric coordinate system $\mathcal{F}^F$ and reparameterize the surfaces $S^{\{V_0,V_1\}}$, $S^{\{V_1,V_2\}}$, and $S^{\{V_2,V_0\}}$ to obtain $S_F^{\{V_0,V_1\}}$, $S_F^{\{V_1,V_2\}}$, and $S_F^{\{V_2,V_0\}}$.
5. For each face $F = \{V_0, V_1, V_2\}$, construct the surface $S^F = S^{\{V_0,V_1,V_2\}}$ by blending $S_F^{\{V_0,V_1\}}$, $S_F^{\{V_1,V_2\}}$, and $S_F^{\{V_2,V_0\}}$.

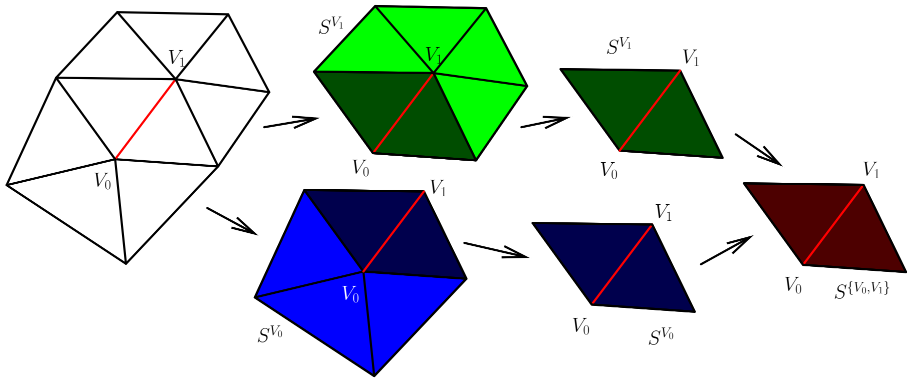


Figure 5.1: The construction of an edge surface.

The construction process of the edge surfaces and face surfaces are illustrated in Figure 5.1 and Figure 5.2. In Figure 5.1, the edge labelled red is the edge that the surface is constructed for, and the mesh on the left shows its two vertices and their neighbours. The green and blue polygons are the vertex surfaces constructed for the two vertices of the edge in the first step, and the darker portions are the surface pieces that are blended in the next step. The construction of the edge surface is completed by blending these two darker portions, giving the red quadrilateral on the right of Figure 5.1.

Now we combine three edge patches to obtain the surface patch for the triangle in the mesh. Figure 5.2 shows the mesh triangle (in the center) and the three edge patches to be blended, where the darker portions of the three edge patches are what get blended in step 5.

Then the output of the algorithm is the combination of all face surfaces S^F. For each step of the algorithm, there are multiple ways to construct surfaces or reparameterizations that satisfy the conditions. In the following sections an example is provided for each step.

5.2.1 Vertex Surface Construction

The first step is to construct a surface for each vertex. To meet the minimal requirements of the scattered data interpolation problem, the surface constructed in this step should interpolate the location and the normal of the vertex. Additional requirements are imposed that the surface interpolates the locations of all neighbours of the vertex. Satisfying this second condition is optional but improves the shape of the resulting surface of the algorithm.

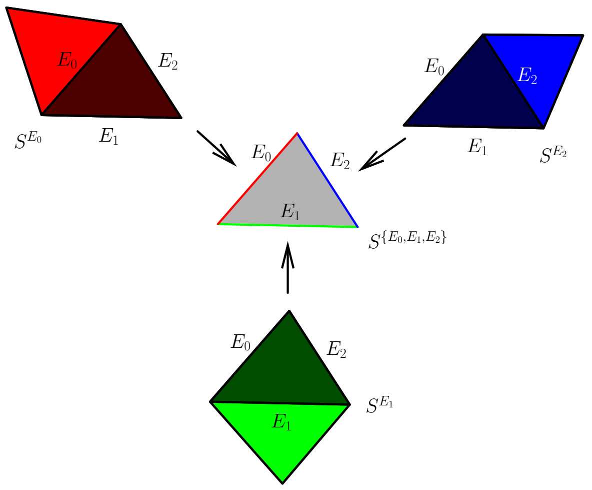


Figure 5.2: The construction of a face surface.

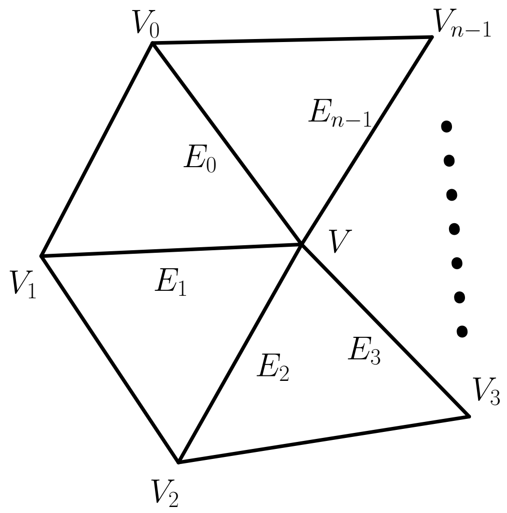


Figure 5.3: A vertex and its adjacent elements.

Consider a vertex V and its neighbours shown in Figure 5.3. Let n be the number of neighbours of V, and $V_0, V_1, \ldots, V_{n-1}$ be the neighbours of V located counter-clockwise around V with respect to the normal of V. The coordinate system $\mathcal{F}^V$ of the vertex V is constructed by the following steps.

- Let the location of V be the origin of $\mathcal{F}^V$.
- Let the direction of the normal of V be the direction of the z-axis of $\mathcal{F}^V$.
- Let the direction from V to the projection of V_0 onto the xy-plane be the direction of the x-axis of $\mathcal{F}^V$, and the direction of the y-axis of $\mathcal{F}^V$ be the direction of the x-axis rotated by 90 degrees counter-clockwise.
- Let the length between V and the projection of V_0 be the unit length of $\mathcal{F}^V$.

Then $\mathcal{F}^V$ is determined, and the locations and normals of the vertices with respect to $\mathcal{F}^V$ are known. To construct a surface that interpolates these vertices, the Polynomial Least [1] [9] is used to build the surface $S^V(x, y)$. The Polynomial Least constructs a surface functionally that interpolates positions and derivatives in an error minimizing way. Here the Least is used to interpolate the positions of V and V_i and the normal at V all specified relative to $\mathcal{F}^V$.

5.2.2 Vertex to Edge Reparameterization

The second step is to construct a local coordinate system for each edge, and reparameterize the surfaces constructed in the first step by these coordinate systems. Consider an edge $E = \{V_0, V_1\}$ and its neighbours as shown in Figure 5.4. Without loss of generality, assume that V_0 is the head of E and V_1 is the tail of E. Let V_2 and V_3 be their neighbours on each side of E. With the method described in the first step, construct two surface S^{V_0} and S^{V_1} for the vertices V_0 and V_1.

The difference between the surfaces of the vertices and the surfaces of the edges is that the vertex surfaces are constructed functionally, but the edge surfaces are constructed parametrically. Thus the coordinate systems of the edges are not placed in the space of the data mesh. So the only restrictions for the coordinate system of an edge E are:

- The origin $(0, 0)$ is mapped to the head of the edge (V_0).
- The point $(0, 1)$ is mapped to the tail of the edge (V_1).

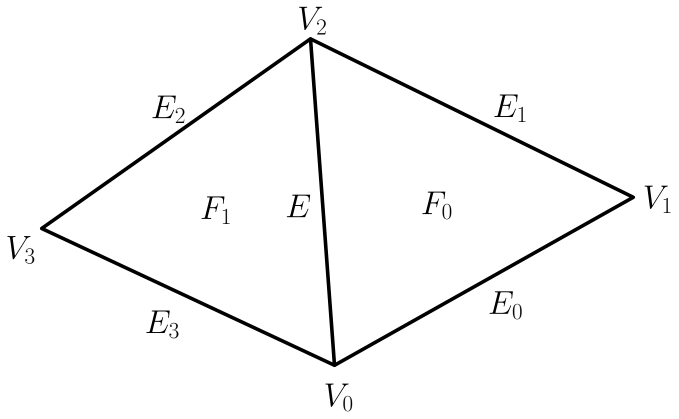


Figure 5.4: An edge and its adjacent elements.

Then the reparameterizations between the coordinate systems of the vertices and the edges satisfy the following conditions.

- The projection of the edge in the coordinate system of the vertex is mapped to the segment from $(0,0)$ to $(0,1)$ in the coordinate system of the edge.
- The reparameterization is linear.

For example, consider the vertex V_0. Let (x_{head}, y_{head}) and (x_{tail}, y_{tail}) denote the locations of the head and the tail of the edge E in $\mathcal{F}^{V_0}$. Then

$$\begin{aligned} r^{E,V_0}(x,y) &= (x_{head} + x(y_{tail} - y_{head}) + y(x_{tail} - x_{head}), \\ &\quad\; y_{head} + x(x_{head} - x_{tail}) + y(y_{tail} - y_{head})), \end{aligned} \tag{5.4}$$

and the reparameterized surface of V_0 is

$$S_E^{V_0}(x,y) = S^{V_0}(r^{E,V_0}(x,y)). \tag{5.5}$$

By the same method the surface $S_E^{V_1}$ can be constructed.

5.2.3 Edge Surface Construction

The third step is to construct the surfaces of the edges. Consider the edge E shown in Figure 5.4. The edge surface S^E is constructed by blending with two sub-surfaces. With $S_E^{V_0}$ and $S_E^{V_1}$ constructed in the second step, the surface S^E is

$$S^E(x,y) = f_0^{2,d}(x,y) S_E^{V_0}(x,y) + f_1^{2,d}(x,y) S_E^{V_1}(x,y), \tag{5.6}$$

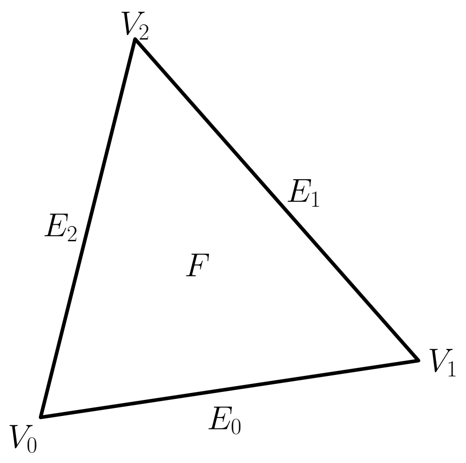


Figure 5.5: A face and its adjacent elements.

where $f_0^{2,d}$ and $f_1^{2,d}$ are defined in Definition 3.4 with $n = 2$, and p_i is

$$\begin{array}{rcl} p_0(x,y) & = & y, \\ p_1(x,y) & = & 1-y. \end{array} \tag{5.7}$$

5.2.4 Edge to Face Reparameterization

The fourth step is to construct a barycentric coordinate system for each face, and reparameterize the surfaces constructed in the third step by these coordinate systems. Consider a face $F = \{V_0, V_1, V_2\}$ and its neighbours as shown in Figure 5.5. $E_0 = \{V_0, V_1\}$, $E_1 = \{V_1, V_2\}$, and $E_2 = \{V_2, V_0\}$ are the edges of the face. With the method described in the third step, construct three surfaces S^{E_0}, S^{E_1}, and S^{E_2} for the edges E_0, E_1, and E_2.

Without loss of generality, let the barycentric coordinate system satisfy the following conditions.

- The point $(1,0,0)$ is mapped to the vertex V_0.
- The point $(0,1,0)$ is mapped to the vertex V_1.
- The point $(0,0,1)$ is mapped to the vertex V_2.

Then reparameterize the surface of each edge of E_0, E_1, and E_2. For example, Figure 5.6 shows $\mathcal{F}^{E_0}$ constructed in the second step. Choose a point V_2' such that the triangle

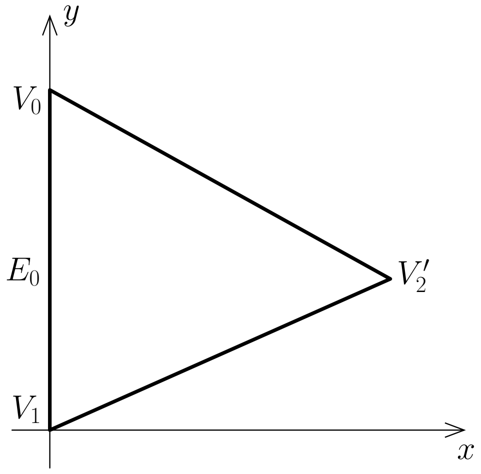


Figure 5.6: The coordinate system of E_0.

$\triangle V_0V_1V_2'$ in $\mathcal{F}^{E_0}$ is similar to the triangle $\triangle V_0V_1V_2$ in the data mesh. Let (x_{V_0}, y_{V_0}), (x_{V_1}, y_{V_1}), and$(x_{V_2'}, y_{V_2'})$ denote the locations of V_0, V_1, and V_2' in $\mathcal{F}^{E_0}$. Then

$$\begin{aligned} r^{F,E_0}(u,v) &= (ux_{V_0} + vx_{V_1} + (1-u-v)x_{V_2'}, \\ &\quad\; uy_{V_0} + vy_{V_1} + (1-u-v)y_{V_2'}), \end{aligned} \tag{5.8}$$

and the reparameterized surface of E_0 is

$$S_F^{E_0}(u,v) = S^{E_0}(r^{F,E_0}(u,v)). \tag{5.9}$$

By the same method the surface $S_F^{E_1}$ and $S_F^{E_2}$ can also be constructed.

5.2.5 Face Surface Construction

The last step is to construct the surfaces of the faces. Consider the face F shown in Figure 5.5. The edge surface S^E is constructed by a surface blending with three targets. With $S_F^{E_0}$, $S_F^{E_1}$, and $S_F^{E_2}$ constructed in the fourth step, the surface S^F is

$$S^F(u,v) = \sum_{i=0}^{2} f_i^{3,d}(u,v) S_F^{E_i}(u,v), \tag{5.10}$$

where $f_0^{3,d}$, $f_1^{3,d}$, and $f_2^{3,d}$ are the triangular blending functions defined in Chapter 4.

5.3 Surface Continuity

In this section the continuity of the resulting surface constructed by the algorithm described in Section 5.2 is specified and the corresponding proofs are given.

The analyses are based on the location of the points on the surface. There are three kinds of continuity to be discussed.

- Face continuity: the continuity at points located inside the triangles.
- Edge continuity: the continuity at points located on the boundaries.
- Vertex continuity: the continuity at points located at the corners.

5.3.1 Face Continuity

For the triangular blending scheme, the interior part of the blended result is C^d-continuous if the three sub-surfaces are C^d-continuous. For the binary blending scheme, the blended result is C^d-continuous if the two sub-surfaces are C^d-continuous. For each vertex, the vertex surfaces are constructed by a polynomial interpolation scheme, and it is always C^∞-continuous. Thus the resulting surface of the algorithm is always at least C^d-continuous inside the triangles.

5.3.2 Edge Continuity

For a point located on a boundary between two blended results, the resulting surface is parametrically continuous at the point with respect to the coordinate system of the corresponding edge.

Theorem 5.1. *For each edge E of the data mesh, the resulting surface constructed by the blending algorithm in Section 5.2 is C^d-continuous across the corresponding boundary (excluding the endpoints) of E, with respect to $\mathcal{F}^E$.*

Proof. Without loss of generality, consider the edge E_0 and the face F shown in Figure 5.5. By Section 5.2.5, the resulting surface with respect to F is

$$S^F(u,v) = \sum_{i=0}^{2} f_i^{3,d,t}(u,v) S_F^{E_i}(u,v). \tag{5.11}$$

Reparametrize S^F with respect to $\mathcal{F}^{E_0}$ by the inverse function $r^{E_0,F}$ of r^{F,E_0} given in Equation 5.8:

$$S^F_{E_0}(x,y) = \sum_{i=0}^{2} f_i^{3,d,t}(r^{E_0,F}(x,y))S^{E_i}_{F,E_0}(x,y). \tag{5.12}$$

Let $t'_i(x,y) = t_i(r^{E_0,F}(x,y))$ and $t' = (t'_0, t'_1, t'_2)$, then for $i \in \{0,1,2\}$,

$$f_i^{3,d,t}(r^{E_0,F}(x,y)) = f_i^{3,d,t'}(x,y). \tag{5.13}$$

Furthermore,

$$S^F_{E_0}(x,y) = \sum_{i=0}^{2} f_i^{3,d,t'}(x,y)S^{E_i}_{F,E_0}(x,y). \tag{5.14}$$

Thus the surface $S^F_{E_0}$ can be considered as a blended result of a d-th triangular blending scheme with respect to the inheritance indicators t'. Since the reparameterization between F and E_0 interpolates the boundary of E_0, then $t'_0(x,y) = 0$ and $q^{t'}(x,y) = 1$ when (x,y) is located on that boundary excluding the endpoints. Then by Proposition 3.9, for all integers $0 \le a+b \le d$,

$$\frac{d^{a+b}}{dx^a dy^b}S^F_{E_0}(x,y) = \frac{d^{a+b}}{dx^a dy^b}S^{E_0}_{F,E_0}(x,y) = \frac{d^{a+b}}{dx^a dy^b}S^{E_0}(x,y), \tag{5.15}$$

for all (x,y) on the boundary of E_0.

By the same method, the surface with respect to another adjacent face also has the same d-th derivatives with respect to $\mathcal{F}^{E_0}$. Then the two surfaces are joined with C^d-continuity. That result also holds for the other two edges of the face and the other faces. $\square$

Since the surface is C^d-continuous across the boundaries, then it is also known that the surface has continuous normals (with $d \ge 1$) and continuous curvatures (with $d \ge 2$) across the boundaries.

5.3.3 Vertex Continuity

The algorithm introduced in this chapter does not construct surfaces that have C^d-continuity at the corners. However the normals and curvatures exist.

Theorem 5.2. *For each vertex V of the data mesh, the resulting surface constructed by the blending algorithm in Section 5.2 has continuous normals at V if $d \geq 1$, and has continuous curvatures at V if $d \geq 2$.*

Proof. When $d \geq 1$, the surface of an edge inherits the first order derivatives from two vertex surfaces at two corners respectively, then the surface inherits the normals. Then for the surfaces of faces, at a corner the two adjacent sub-surfaces share the same normals since they inherit the normal from the same vertex. Thus the normal at that corner of the face surface exists and it is equal to the normal of the corresponding vertex surface at the vertex point. Furthermore all face surfaces around the vertex share the same normal, so the normals are continuous at V.

By the same method, the result holds for curvatures when $d \geq 2$. □

5.4 Comparison to Nielson's Results

To show the differences between the new triangular blending scheme and Nielson's scheme, I compared the results constructed by the new algorithm in this chapter and the algorithm formed by Nielson's scheme. The examples of the new algorithm are constructed with different order of continuity (with $d = 0$, $d = 1$, and $d = 2$). The ellipsoid and perturbed mesh shown in Figure 5.7 are used as the inputs.

For the ellipsoid mesh, Figure 5.8 and Figure 5.9 shows the surfaces constructed by the four methods, and Figure 5.10 shows scaled views of them. While Nielson's method appears to have better shape, note that the curvature discontinuities are higher in Nielson's method than in the surface constructed with the G^1 $(d = 1)$ new algorithm.

For the perturbed mesh, Figure 5.11 and Figure 5.12 shows the surfaces constructed by the four methods, and Figure 5.13 shows scaled views of them. For this example, the surface constructed by the G^1 $(d = 1)$ new algorithm has both better shape and curvature plots.

Notice that for the new algorithm with $d = 1$, the results constructed are G^1-continuous, and the isophotes are just C^0-continuous. However with the new algorithm the G^1 surface is very close to be G^2, so the kinks in the isophotes are hard to see in the figure. In the scaled images shown in part (b) of Figure 5.10 and Figure 5.13, such kinks can be seen in Nielson's scheme, although they are still hard to see in the G^1 new algorithm.

Further examples of surfaces constructed using my scheme and algorithm are given in Chapter 8. Note that Nielson's scheme is restricted to triangular meshes. As we will see

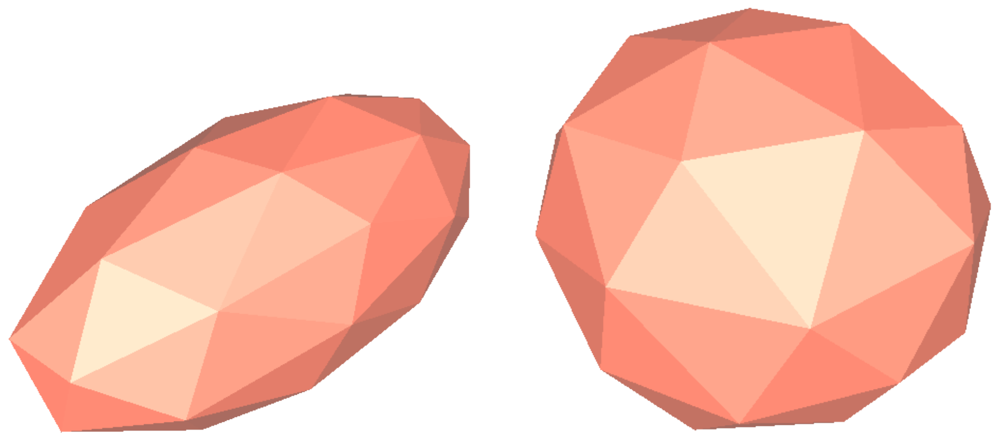

(a) (b)

Figure 5.7: The input mesh for scheme comparison; (a) ellipsoid; (b) perturbed.

in Chapter 7 and Chapter 8, my scheme and algorithm can be generalized to multi-sided faces.

5.5 Summary

In this chapter I presented the complete process of the triangular surface blending algorithm, and show that the algorithm constructs surfaces with continuous normals and curvature. Currently the order of continuity is limited to 2. Then I compared some examples provided by this algorithm with Nielson's results. In the next chapter I will show that the continuity can be improved to the parametric continuity with arbitrary order.

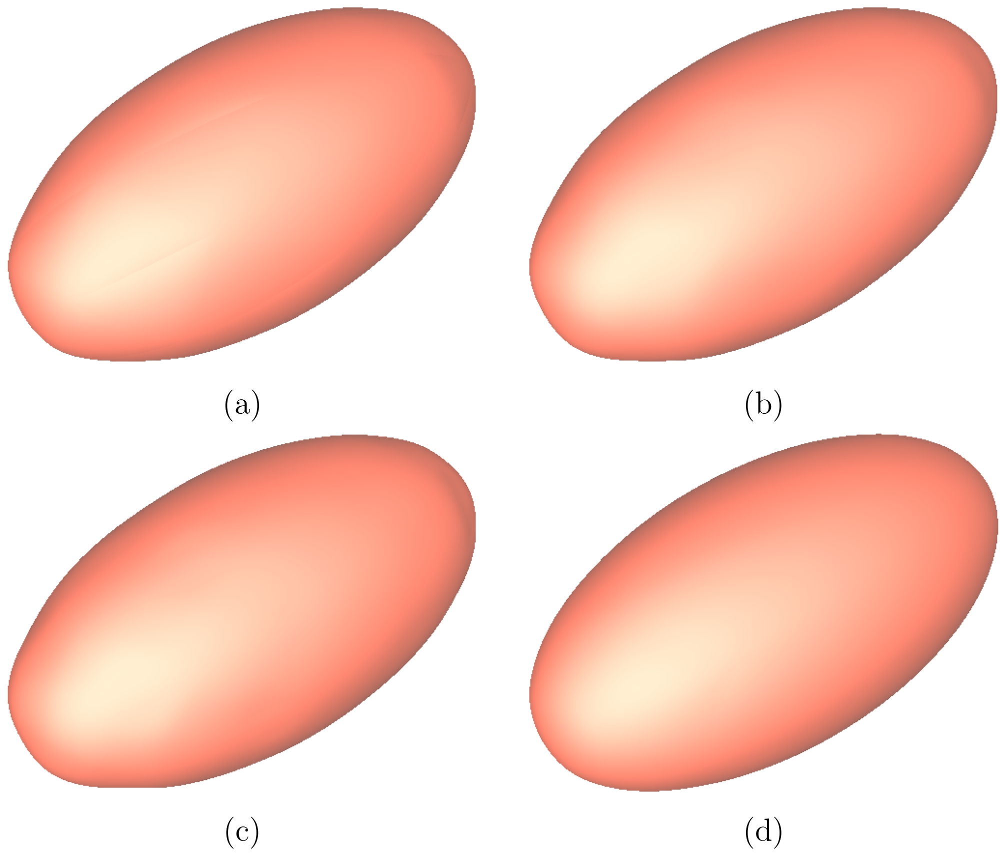


Figure 5.8: The shaded images of the surfaces constructed for the ellipsoid mesh; (a) improved algorithm with $d = 0$; (b) improved algorithm with $d = 1$; (c) improved algorithm with $d = 2$; (d) Nielson's algorithm.

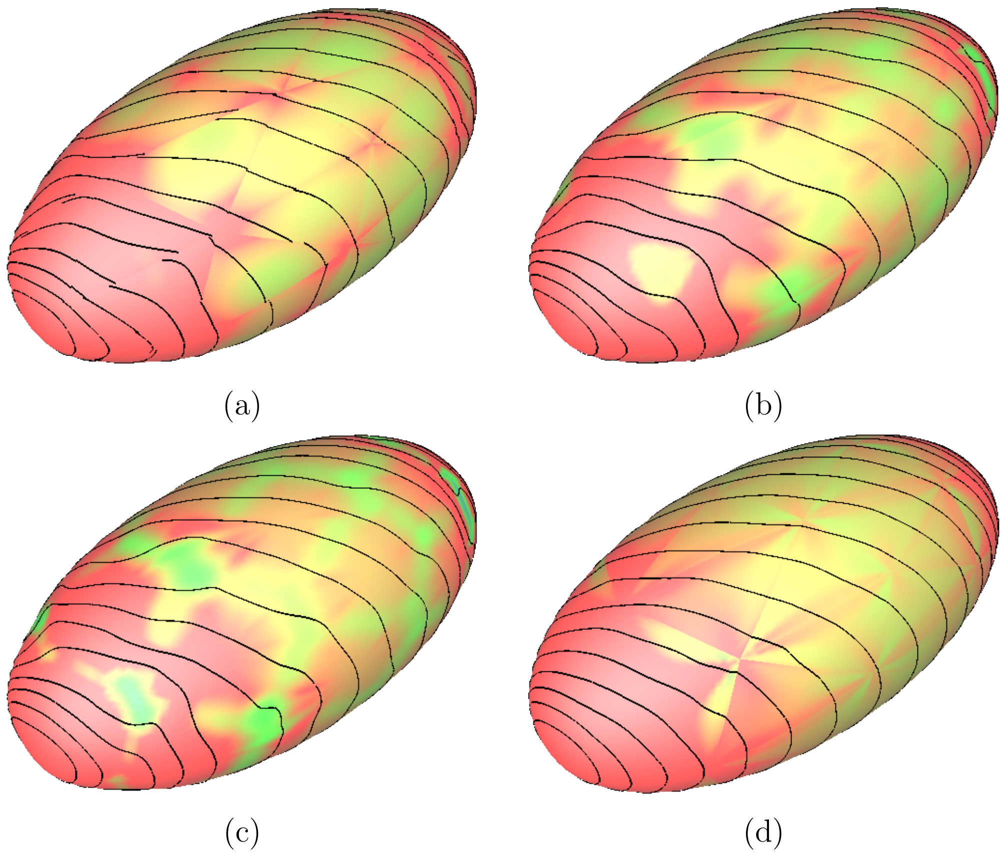


Figure 5.9: The curvature plots and isophotes of the surfaces constructed for the ellipsoid mesh; (a) improved algorithm with $d = 0$; (b) improved algorithm with $d = 1$; (c) improved algorithm with $d = 2$; (d) Nielson's algorithm.

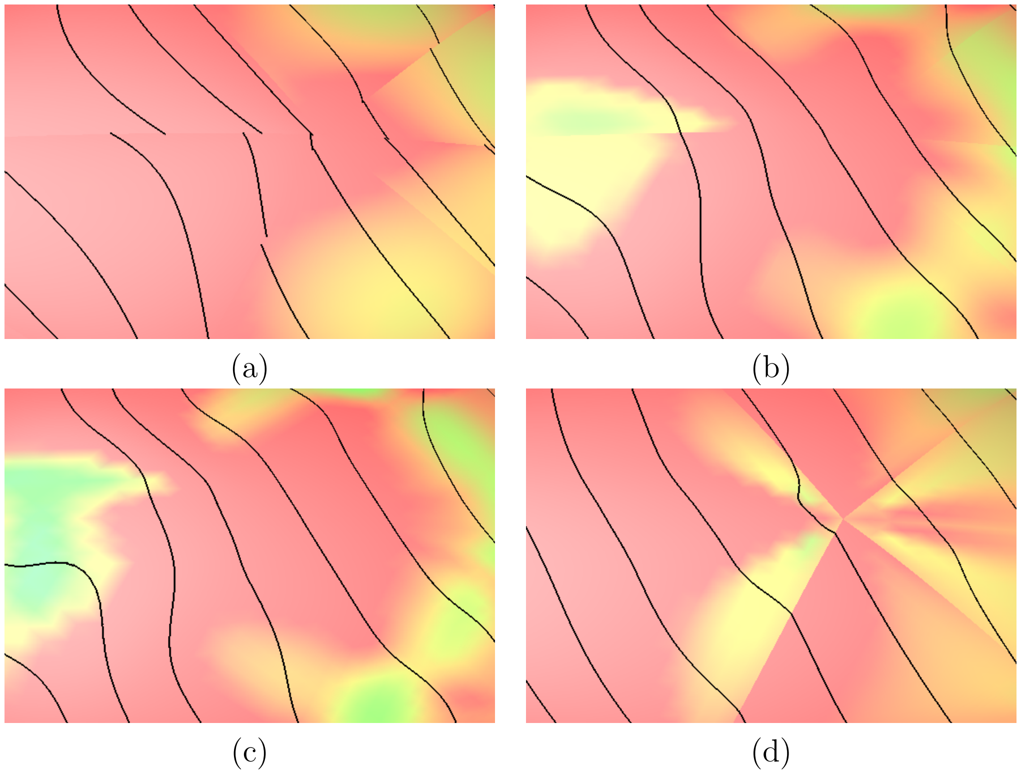
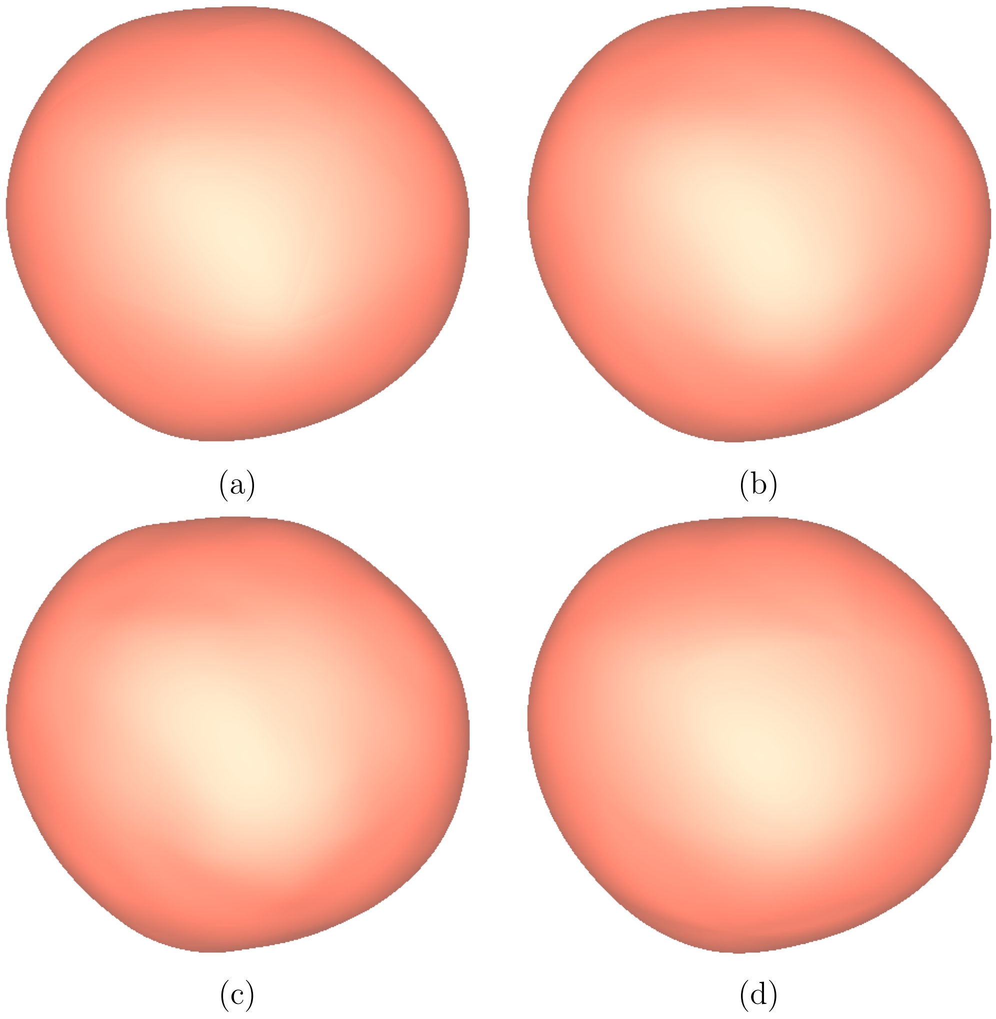
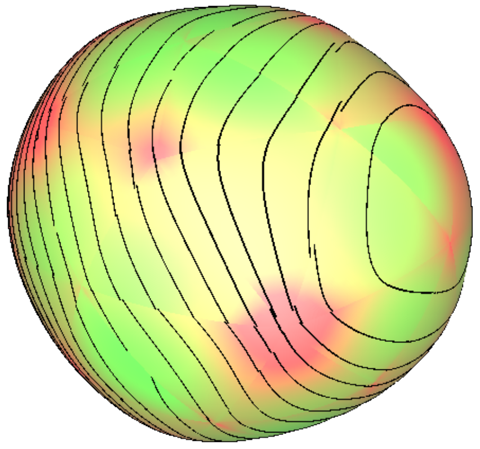

(a)
(b)
(c)
(d)

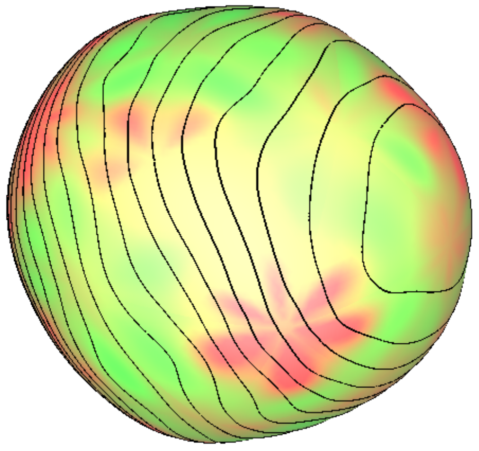
Figure 5.10: The scaled views of the curvature plots and isophotes of the surfaces constructed for the ellipsoid mesh; (a) improved algorithm with $d = 0$; (b) improved algorithm with $d = 1$; (c) improved algorithm with $d = 2$; (d) Nielson's algorithm.

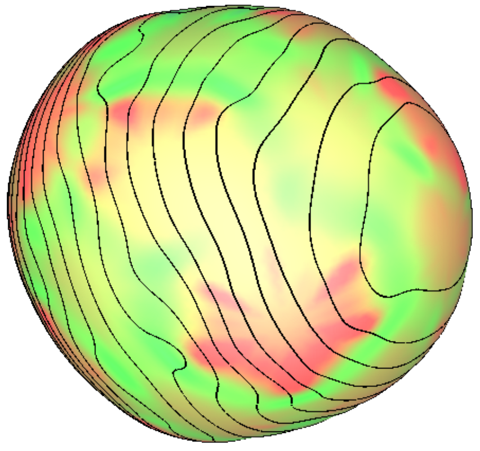


(a) (b) (c) (d)

Figure 5.11: The shaded images of the surfaces constructed for the perturbed mesh; (a) improved algorithm with $d = 0$; (b) improved algorithm with $d = 1$; (c) improved algorithm with $d = 2$; (d) Nielson's algorithm.


(a)


(b)


(c)

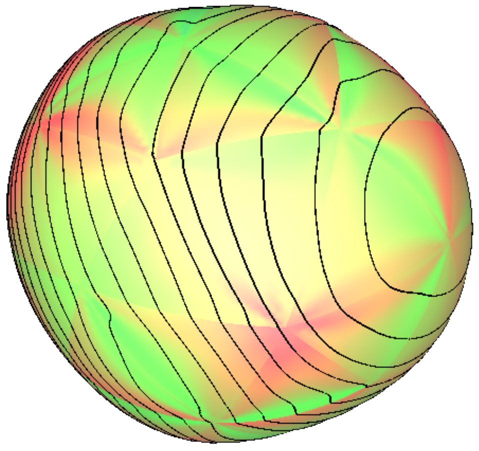

(d)

Figure 5.12: The curvature plots and isophotes of the surfaces constructed for the perturbed mesh; (a) improved algorithm with $d = 0$; (b) improved algorithm with $d = 1$; (c) improved algorithm with $d = 2$; (d) Nielson's algorithm.

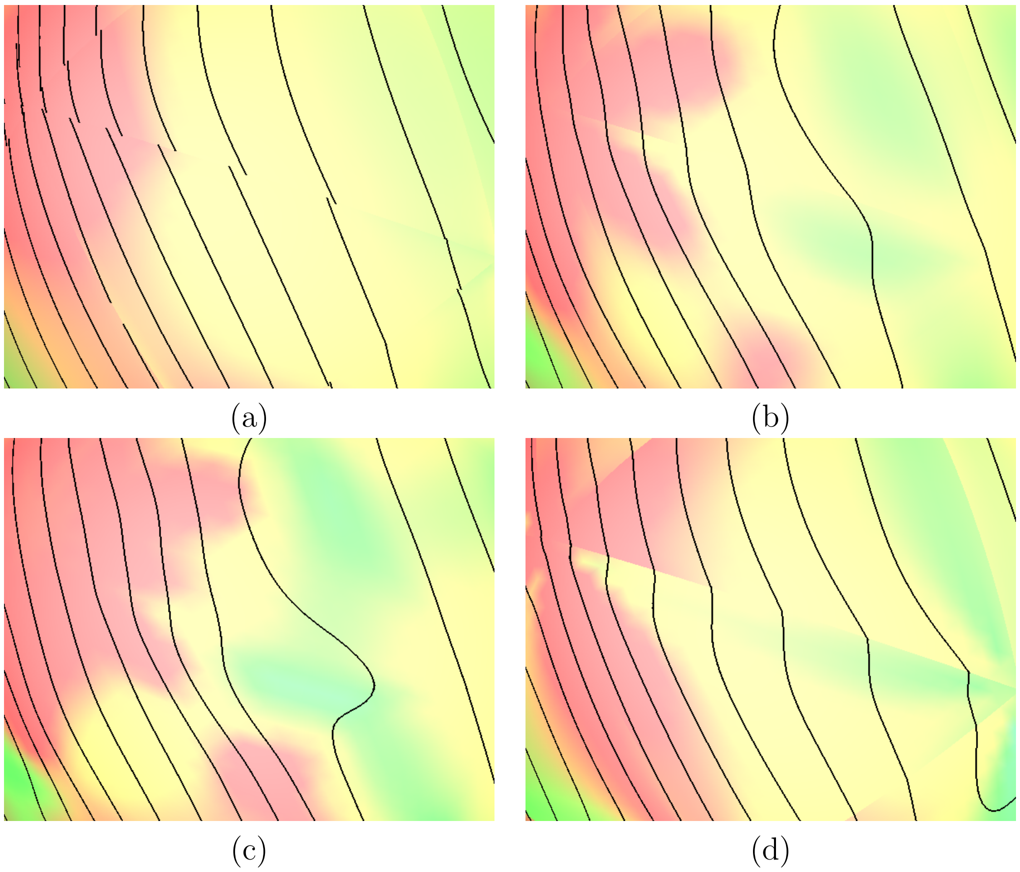


Figure 5.13: The scaled views of the curvature plots and isophotes of the surfaces constructed for the perturbed mesh; (a) improved algorithm with $d = 0$; (b) improved algorithm with $d = 1$; (c) improved algorithm with $d = 2$; (d) Nielson's algorithm.

Chapter 6

Parametric Continuity Improvement

The d-th triangular algorithm provided in Chapter 5 constructs surfaces only with continuous normals when $d \geq 1$ or curvatures when $d \geq 2$ at the locations of the vertices. In this chapter the conditions of parametric continuity and an extended algorithm are developed. The surfaces constructed by the extended algorithm are parametrically continuous everywhere.

This chapter focuses on parametric interpolation, although the results hold for functional interpolation without modification.

In Chapter 5, I proved that the surfaces constructed by the blending algorithm are parametrically continuous on the edges. Also the continuity properties always hold in the interior part of each blended result. What remains is to show the parametric continuity at the vertices. The goal of this chapter is to show that the resulting surfaces are parametrically continuous at the vertices, with respect to the coordinate system of each vertex. A major problem is that with the current reparameterization function in the algorithm, the derivatives do not exist at the vertices, and the surface is not well-defined at other locations, with respect to the coordinate system of vertices. So the first task is to update the reparameterization functions.

Parametric continuity at the vertices is obtained in a way similar to Theorem 5.1. The proof of this theorem shows that a piece of blended result reparameterized by the coordinate system of a chosen element has the same derivatives with respect to that coordinate system. Thus all adjacent blended results have the same derivatives and are joined with parametric continuity. For a single piece of the blended result, the construction and proof of parametric continuity contains three steps:

1. Choose an element from the data mesh as the base.

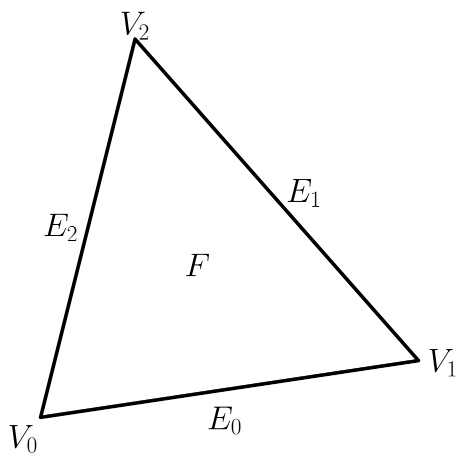


Figure 6.1: The layout of a face and its neighbours.

2. Reparameterize the blended result and its sub-surfaces by the coordinate system of the chosen element.
3. Show that the reparameterized sub-surfaces satisfy the conditions of the triangular blending scheme. Then the reparameterized blended result inherits the derivatives from one of sub-surfaces.

For the first step, the element chosen is the vertex itself. The second step is more involved, since the vertices and faces are not parametrically adjacent. That means the surfaces parameterized by the coordinate systems of the faces cannot be reparameterized by the coordinate system of the vertices directly. An intermediate element is required to establish that reparameterization. For a vertex and one of its adjacent faces, the possible intermediate elements are the two adjacent edges. The choice of the edge for each sub-surface depends on the corresponding derivatives.

Consider the elements shown in Figure 6.1. In the following analyses the relationship between the derivatives of the blended result with respect to the face F reparameterized by $\mathcal{F}^{V_0}$ and the surface of V_0 are considered. Without loss of generality, assume that V_0 is the head of both E_0 and E_2. Then Proposition 6.1 shows that the surface of an edge reparameterized by the coordinate system of an adjacent vertex has the same derivatives as the surface of that vertex.

Proposition 6.1. *The surface S^E of an edge E constructed in Section 5.2.3 has the same derivatives of order less than or equal to d as the surface S^V of an adjacent vertex V constructed in Section 5.2.1 with respect to $\mathcal{F}^V$ at the point of the vertex.*

Proof. Without loss of generality, consider the edge E_0 in Figure 6.1. Constructing the surface $S^{E_0}(x,y)$ with the method in Section 5.2.3 gives

$$S^{E_0}(x,y) = f_0^{2,d,b}(x,y)S_{E_0}^{V_0}(x,y) + f_1^{2,d,b}(x,y)S_{E_0}^{V_1}(x,y). \tag{6.1}$$

Reparameterizing S^{E_0} by $\mathcal{F}^{V_0}$, the reparameterized surface is

$$S_{V_0}^{E_0}(x,y) = f_0^{2,d,b}(r^{V_0,E_0}(x,y))S^{V_0}(x,y) + f_1^{2,d,b}(r^{V_0,E_0}(x,y))S_{E_0,V_0}^{V_1}(x,y). \tag{6.2}$$

Let $b_0'(x,y) = b_0(r^{V_0,E_0}(x,y))$, $b_1'(x,y) = b_1(r^{V_0,E_0}(x,y))$, and $b' = (b_0', b_1')$, then

$$\begin{array}{rcl} f_0^{2,d,b}(r^{V_0,E_0}(x,y)) & = & f_0^{2,d,b'}(x,y), \\ f_1^{2,d,b}(r^{V_0,E_0}(x,y)) & = & f_1^{2,d,b'}(x,y). \end{array} \tag{6.3}$$

Furthermore,

$$S_{V_0}^{E_0}(x,y) = f_0^{2,d,b'}(x,y)S^{V_0}(x,y) + f_1^{2,d,b'}(x,y)S_{E_0,V_0}^{V_1}(x,y). \tag{6.4}$$

Thus the surface $S_{V_0}^{E_0}$ is a blend of a d-th binary blending scheme with respect to the inheritance indicators t'. Since the reparameterization between E_0 and V interpolates the point of V_0, then $b_0'(V_0) = 0$ and $q^{b'}(V_0) = 1$. Then by Proposition 3.9, for all integers $0 \le a+b \le d$,

$$\frac{d^{a+b}}{dx^a dy^b} S_{V_0}^{E_0}(V_0) = \frac{d^{a+b}}{dx^a dy^b} S^{V_0}(V_0). \tag{6.5}$$

□

Then by Section 5.2.5, the blended result of F is

$$S^F(u,v) = \sum_{i=0}^{2} f_i^{3,d,t}(u,v) S_F^{E_i}(u,v). \tag{6.6}$$

Now reparameterize S^F with respect to $\mathcal{F}^{V_0}$. Since V_0 and F are not adjacent,

- reparametrize the sub-surface $S_F^{E_0}$ via the path $E_0 \to F \to E_0 \to V_0$.
- reparametrize the sub-surface $S_F^{E_1}$ via one of the paths $E_0 \to F \to E_0 \to V_0$ and $E_2 \to F \to E_2 \to V_0$. Without loss of generality, choose $E_0 \to F \to E_0 \to V_0$.

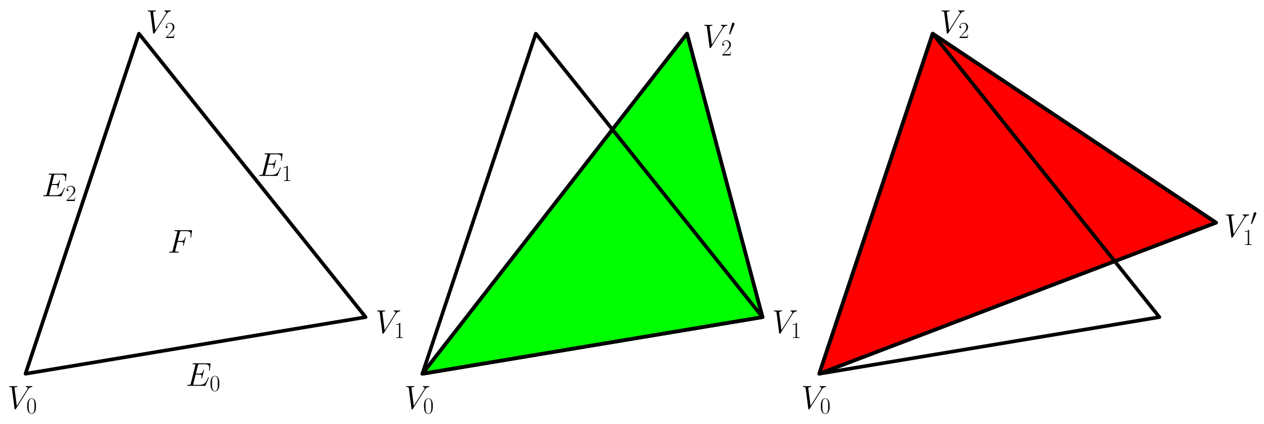


Figure 6.2: The elements inside the domain of V_0.

- reparametrize the sub-surface $S_F^{E_2}$ via the path $E_2 \to F \to E_2 \to V_0$.

Thus by Proposition 6.1, the reparametrized sub-surface $S_{V_0}^{E_0}$ and $S_{V_0}^{E_2}$ have the same derivatives at V_0.

$$\frac{d^{a+b}}{dx^a dy^b} S_{V_0}^{E_0}(V_0) = \frac{d^{a+b}}{dx^a dy^b} S^{V_0}(V_0) = \frac{d^{a+b}}{dx^a dy^b} S_{V_0}^{E_2}(V_0). \tag{6.7}$$

The two surfaces are d-th matched at this corner. However the blended result S^F is reparametrized by the triangular domain T of the face F. Thus, most of the time the reparameterization functions r^{F,E_0,V_0} and r^{F,E_2,V_0} are not the same. That means the corresponding triangular portions in the domain of V_0 do not match.

Figure 6.2 shows the face F inside the domain of V_0, as well as the two different reparameterization results of the domain of F via different paths. With respect to Section 5.2.2 and Section 5.2.4, the triangular domain T of the face F is mapped to the green triangle $\triangle V_0V_1V_2'$ by the reparameterization function r^{F,E_0,V_0}, and T is mapped to the red triangle $\triangle V_0V_1'V_2$ by r^{F,E_2,V_0}. In that case the only point matched is the point of V_0, and the reparametrized blended result is well-defined only at that point, and the derivatives are meaningless. To make the derivatives meaningful, the reparametrized blended result should be well-defined at least in an open set that includes the point of V_0.

To meet such a requirement, there are two possible methods:

- Choose a portion in the domain of V_0 and do two additional reparameterizations from that portion to both the green and red triangles.

- Modify the reparameterization process to force the two mappings r^{F,E_0,V_0} and r^{F,E_2,V_0} to match in an open set that includes the point of V_0.

The first method constructs resulting surfaces with parametric continuity only at the location of the vertex (with respect to the coordinate system of the vertex). The second method extends the continuous area to an open set containing the vertex.

6.1 Vertex Continuity Method

In this section the first reparameterization method is used. In Figure 6.2, the triangle $\triangle V_0V_1V_2$ is chosen as the corresponding matched portion. However the reparameterizations from the triangle $\triangle V_0V_1V_2$ to the green triangle $\triangle V_0V_1V_2'$ or the red triangle $\triangle V_0V_1'V_2$ do not keep the derivatives at the point of V_0. The reparameterization in Section 5.2.4 is modified to solve the problem as detailed in Section 6.1.1.

The new algorithm provided in this section is called the *improved triangular blending algorithm with vertex continuity.*

6.1.1 Modified Edge to Face Reparameterization

In this section an example of the reparameterization between the edge E_2 and the face F is given. Figure 6.3 shows $\mathcal{F}^{E_2}$ constructed in the algorithm. The point V_1' is the point obtained by mapping the location of V_1 in $\mathcal{F}^{V_0}$ to $\mathcal{F}^{E_2}$ with the reparameterization function r^{V_0,E_2}, and the point V_1'' is the point obtained by mapping the location of V_1 in $\mathcal{F}^{V_2}$ to $\mathcal{F}^{E_2}$ with the reparameterization function r^{V_2,E_2}. Recall that most of the time V_1' and V_1'' are not the same.

Notice that the triangle $\triangle V_0V_1'V_2$ is mapped to the blue triangle $\triangle V_0V_1V_2$ in Figure 6.2, in $\mathcal{F}^{V_0}$. So if the domain T of the face F is mapped to $\triangle V_0V_1'V_2$ then it is also mapped to the blue triangle $\triangle V_0V_1V_2$ in $\mathcal{F}^{V_0}$. Such a reparameterization satisfies the conditions in $\mathcal{F}^{V_0}$, however it does not satisfy the corresponding conditions of V_2. A reparameterization satisfying the conditions of both vertices V_0 and V_2 is required. To obtain such a reparameterization, a binary blending is used.

Let (x_{V_0}, y_{V_0}), $(x_{V_1'}, y_{V_1'})$, $(x_{V_1''}, y_{V_1''})$, and (x_{V_2}, y_{V_2}) denote the locations of V_0, V_1', V_1'', and V_2 in $\mathcal{F}^{E_2}$. Then define two sub-functions $r^{F,E_2}_{V_0}$ and $r^{F,E_2}_{V_2}$ as

$$\begin{aligned} r^{F,E_2}_{V_0}(u,v) &= (ux_{V_0} + vx_{V_1'} + (1-u-v)x_{V_2}, \\ &\quad uy_{V_0} + vy_{V_1'} + (1-u-v)y_{V_2}), \end{aligned} \tag{6.8}$$

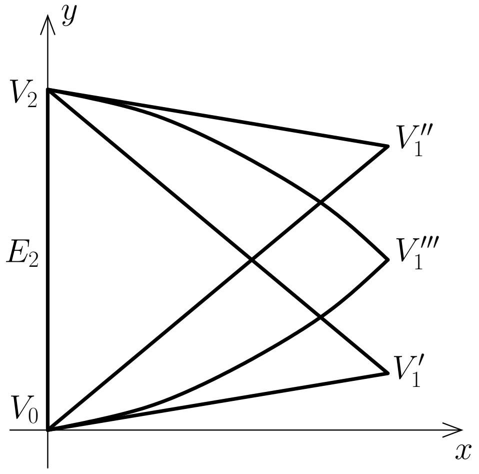


Figure 6.3: The elements inside the domain of E_2.

and

$$\begin{aligned} r_{V_2}^{F,E_2}(u,v) &= (ux_{V_0} + vx_{V_1''} + (1-u-v)x_{V_2}, \\ & \quad uy_{V_0} + vy_{V_1''} + (1-u-v)y_{V_2}). \end{aligned} \tag{6.9}$$

Then the modified reparameterization function r^{F,E_2} is

$$r^{F,E_2}(u,v) = f_0^{2,d,b^r}(u,v) r_{V_0}^{F,E_2}(u,v) + f_1^{2,d,b^r}(u,v) r_{V_2}^{F,E_2}(u,v), \tag{6.10}$$

where $b^r = (b_0^r, b_1^r)$, and b_0^r and b_1^r are defined as

$$\begin{aligned} b_0^r(u,v) &= 1 - u - \frac{v}{2}, \\ b_1^r(u,v) &= u + \frac{v}{2}. \end{aligned} \tag{6.11}$$

Then by Proposition 3.9, r^{F,E_2} inherits the derivatives of order less than or equal to d from the sub-function $r_{V_0}^{F,E_2}$ at the point of V_0, and inherits the derivatives of order less than or equal to d from the sub-function $r_{V_2}^{F,E_2}$ at the point of V_2.

Then in Figure 6.4, the red portion is the result of mapping the triangular domain T of the face F by the modified r^{F,E_2,V_0}. The reparameterization from the red portion to the blue triangle in Figure 6.2 does not change the derivatives at the point of V_0.

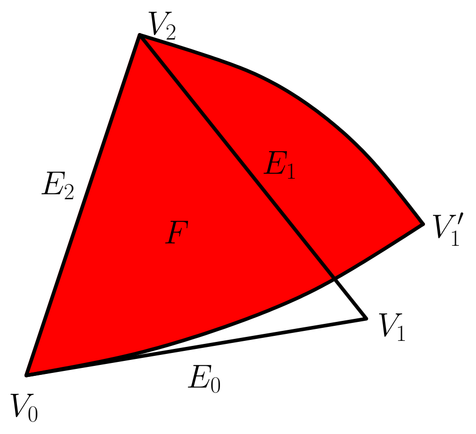


Figure 6.4: The elements inside the domain of V_0.

6.1.2 Improved Vertex Continuity

With the modified reparameterization functions, for a point located at a corner of the faces, the resulting surface is parametrically continuous at the point with respect to the coordinate system of the corresponding vertex.

Theorem 6.1. *For each vertex V of the data mesh, the resulting surface constructed by the blending algorithm in Section 5.2 with the reparameterization function in Section 6.1.1 is C^d-continuous at the corresponding point of V, with respect to $\mathcal{F}^V$.*

Proof. Without loss of generality, consider the vertex V_0 and the face F shown in Figure 6.1, and its image in the domain of V_0 shown in Figure 6.2. By Section 5.2.5, the resulting surface with respect to F is

$$S^F(u,v) = \sum_{i=0}^{2} f_i^{3,d,t}(u,v) S_F^{E_i}(u,v). \tag{6.12}$$

Reparameterize S^F with respect to $\mathcal{F}^{V_0}$. Let r^{E_0} denote the reparameterization from the triangle of F to the modified green portion, and r^{E_2} denote the reparameterization from the blue triangle to the modified red portion. Let $S^F_{\sim V_0}$ denote the reparametrized blended result, then

$$\begin{aligned} S^F_{\sim V_0}(x,y) \quad = \quad & f_0^{3,d,t}(r^{V_0,E_0,F}(r^{E_0}(x,y)))S^{E_0}_{V_0}(x,y)+ \\ & f_1^{3,d,t}(r^{V_0,E_0,F}(r^{E_0}(x,y)))S^{E_1}_{F,E_0,V_0}(x,y)+ \\ & f_2^{3,d,t}(r^{V_0,E_2,F}(r^{E_2}(x,y)))S^{E_2}_{V_0}(x,y). \end{aligned} \tag{6.13}$$

A new inheritance indicator t' is defined as

$$\begin{aligned} t'_0(x,y) &= t_0(r^{V_0,E_0,F}(r^{E_0}(x,y))), \\ t'_1(x,y) &= t_1(r^{V_0,E_0,F}(r^{E_0}(x,y))), \\ t'_2(x,y) &= t_2(r^{V_0,E_2,F}(r^{E_2}(x,y))), \end{aligned} \tag{6.14}$$

and $t' = (t'_0, t'_1, t'_2)$, then

$$\begin{aligned} S^F_{\sim V_0}(x,y) = \; & f_0^{3,d,t'}(x,y) S^{E_0}_{V_0}(x,y) + \\ & f_1^{3,d,t'}(x,y) S^{E_1}_{F,E_0,V_0}(x,y) + \\ & f_2^{3,d,t'}(x,y) S^{E_2}_{V_0}(x,y). \end{aligned} \tag{6.15}$$

Thus the surface $S^F_{\sim V_0}$ can be considered as a blended result of a d-th triangular blending scheme with respect to the inheritance indicators t'. Since all reparameterizations interpolate the point of V_0, then $t'_0(V_0) = 0$, $t'_2(V_0) = 0$, and $q^{t'}(V_0) = 2$. Then by Proposition 6.1,

$$\frac{d^{a+b}}{dx^a dy^b} S^{E_0}_{V_0}(V_0) = \frac{d^{a+b}}{dx^a dy^b} S^{V_0}(V_0) = \frac{d^{a+b}}{dx^a dy^b} S^{E_2}_{V_0}(V_0). \tag{6.16}$$

Then by Proposition 3.16, for all integers $0 \le a+b \le d$,

$$\frac{d^{a+b}}{dx^a dy^b} S^F_{\sim V_0}(x,y) = \frac{d^{a+b}}{dx^a dy^b} S^{V_0}(x,y). \tag{6.17}$$

By the same method, the surface with respect to other adjacent faces also has the same d-th derivatives with respect to $\mathcal{F}^{V_0}$. Then all surfaces are joined with C^d-continuity. This continuity result holds for the other two vertices of the face and the other faces. $\square$

6.2 Open Set Method

In many analyses the local parametric continuity at a singular point is not sufficient. Most of the time the continuity inside an open set around the point is desired. The parametric continuity conditions of the vertices are extended in this section. The edge to face reparameterization is updated once more, and for each vertex, the resulting surface is parametrically continuous inside an open set around it.

The new algorithm provided in this section is called the *improved triangular blending algorithm with open set continuity.*

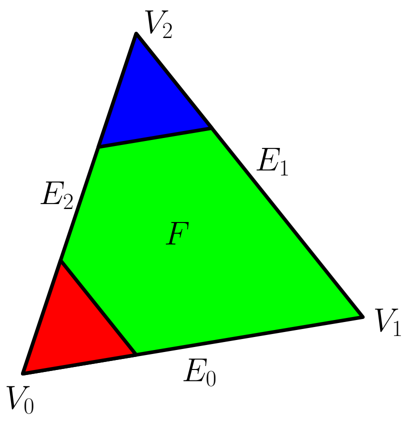


Figure 6.5: The layout of a face and its neighbours.

6.2.1 Modified Edge to Face Reparameterization

Consider the elements shown in Figure 6.1. In Section 6.1.1, the reparameterization guarantees that the derivatives of $r^{V_0,E_0,F}$ and $r^{V_0,E_2,F}$ match at the point of V_0. In this section the reparameterization makes $r^{V_0,E_0,F}$ and $r^{V_0,E_2,F}$ have the same values inside an open set around the point of V_0.

As shown in Figure 6.6, the meanings of V_0, V_1', V_1'', and V_2 are the same as in Figure 6.3. Then the points V_a, V_b, V_c, and V_d are the three equal points of the segment V_0V_1', V_0V_2, and V_2V_1''. The combination of the areas shaded by red, blue, and green is the desired reparameterization image of the triangular domain T shown in Figure 6.5. The shaded areas in T are mapped to the areas with the same colors. Particularly the red and blues portions should be mapped uniformly to guarantee the equality of the two reparameterization functions for each vertex. Furthermore the green portion should be joined with the red and blue portions with C^d-continuity. This reparameterization can be constructed with a binary blending scheme.

Let (x_{V_0}, y_{V_0}), $(x_{V_1'}, y_{V_1'})$, $(x_{V_1''}, y_{V_1''})$, and (x_{V_2}, y_{V_2}) denote the locations of V_0, V_1', V_1'', and V_2 in $\mathcal{F}^{E_2}$. Then define two sub-functions $r_{V_0}^{F,E_2}$ and $r_{V_2}^{F,E_2}$ as

$$\begin{aligned} r_{V_0}^{F,E_2}(u,v) &= (ux_{V_0} + vx_{V_1'} + (1-u-v)x_{V_2}, \\ &\quad uy_{V_0} + vy_{V_1'} + (1-u-v)y_{V_2}), \end{aligned} \tag{6.18}$$

and

$$\begin{aligned} r_{V_2}^{F,E_2}(u,v) &= (ux_{V_0} + vx_{V_1''} + (1-u-v)x_{V_2}, \\ &\quad uy_{V_0} + vy_{V_1''} + (1-u-v)y_{V_2}). \end{aligned} \tag{6.19}$$

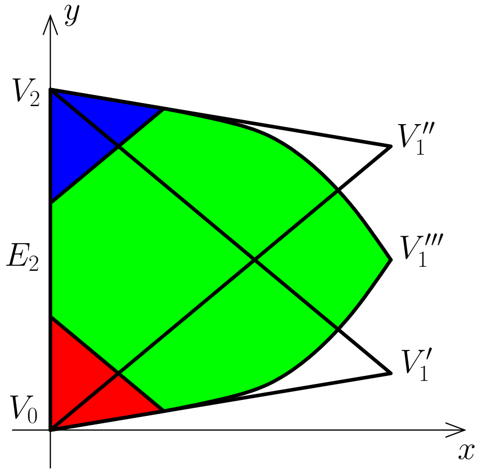


Figure 6.6: The elements inside the domain of E_2.

Then the modified reparameterization function r^{F,E_2} is

$$\begin{aligned} r^{F,E_2}(u,v) &= r^{F,E_2}_{V_0}(u,v), && \text{if } u+v < \frac{1}{3}, \\ r^{F,E_2}(u,v) &= r^{F,E_2}_{V_2}(u,v), && \text{if } v > \frac{2}{3}, \\ r^{F,E_2}(u,v) &= f_0^{2,d,b^r}(u,v) r^{F,E_2}_{V_0}(u,v) + f_1^{2,d,b^r}(u,v) r^{F,E_2}_{V_2}(u,v), && \text{otherwise.} \end{aligned} \tag{6.20}$$

where $b^r = (b_0^r, b_1^r)$, and b_0^r and b_1^r are defined as

$$\begin{aligned} b_0^r(u,v) &= u+v-\frac{1}{3}, \\ b_1^r(u,v) &= \frac{2}{3}-v. \end{aligned} \tag{6.21}$$

Then by Proposition 3.9, r^{F,E_2} inherits the derivatives of order less than or equal to d from the sub-function $r^{F,E_2}_{V_0}$ at the point of V_0, and inherits the derivatives of order less than or equal to d from the sub-function $r^{F,E_2}_{V_2}$ at the point of V_2.

6.2.2 Improved Continuity Around the Vertex

The continuity inside the open set is shown in this section. If the surface piece is C^d-continuous with respect to the face domain T, then it is also C^d-continuous with respect to the barycentric coordinate system of the vertex since that coordinate system is now well-defined. Also the proof of parametric continuity at the vertex is the same as in Section 6.1.2, as shown in Theorem 6.2.

Theorem 6.2. *For each vertex V of the data mesh, the resulting surface constructed by the blending algorithm in Section 5.2 with the reparameterization function in Section 6.2.1 is C^d-continuous at the corresponding point of V, with respect to $\mathcal{F}^V$.*

Proof. Same as the proof of Theorem 6.1 in Section 6.1.2. □

The last step is to show the result for the edges. This proof for the edges is similar to the proof in Section 5.3.2.

Theorem 6.3. *For each vertex V of the data mesh, the resulting surface constructed by the blending algorithm in Section 5.2 with the reparameterization function in Section 6.2.1 is C^d-continuous on the boundaries of the edges adjacent to the vertex, with respect to $\mathcal{F}^V$.*

Proof. Without loss of generality, consider the edge E_0 and the face F shown in Figure 6.1. By Section 5.2.5, the resulting surface with respect to F is

$$S^F(u,v) = \sum_{i=0}^{2} f_i^{3,d,t}(u,v) S_F^{E_i}(u,v). \tag{6.22}$$

Reparametrize S^F with respect to $\mathcal{F}^{V_0}$ by the function $r^{V_0,E_0,F}$:

$$S_{E_0,V_0}^F(x,y) = \sum_{i=0}^{2} f_i^{3,d,t}(r^{V_0,E_0,F}(x,y)) S_{F,E_0,V_0}^{E_i}(x,y). \tag{6.23}$$

Let $t'_i(x,y) = t_i(r^{V_0,E_0,F}(x,y))$ and $t' = (t'_0, t'_1, t'_2)$, then for $i \in \{0,1,2\}$,

$$f_i^{3,d,t}(r^{V_0,E_0,F}(x,y)) = f_i^{3,d,t'}(x,y). \tag{6.24}$$

Furthermore,

$$S_{E_0,V_0}^F(x,y) = \sum_{i=0}^{2} f_i^{3,d,t'}(x,y) S_{F,E_0,V_0}^{E_i}(x,y). \tag{6.25}$$

Thus the surface S_{E_0,V_0}^F can be considered as a blended result of a d-th triangular blending scheme with respect to the inheritance indicators t'. Since the reparameterization between F and E_0 interpolates the boundary of E_0, then $t'_0(x,y) = 0$ and $q^{t'}(x,y) = 1$ when (x,y) is located on the that boundary excluding the endpoints. Then by Proposition 3.9, for all integers $0 \leq a+b \leq d$,

$$\frac{d^{a+b}}{dx^a dy^b} S^F_{E_0,V_0}(x,y) = \frac{d^{a+b}}{dx^a dy^b} S^{E_0}_{F,E_0,V_0}(x,y) = \frac{d^{a+b}}{dx^a dy^b} S^{E_0}_{V_0}(x,y), \tag{6.26}$$

for all (x,y) on the boundary of E_0.

By the same method, the surface with respect to another adjacent face also has the same d-th derivatives with respect to $\mathcal{F}^{V_0}$. Then the two surfaces are joined with C^d-continuity. □

6.3 Summary

In this chapter I gave two methods to modify the reparameterization functions of the triangular surface blending algorithm, and call these two modified algorithms the improved triangular blending algorithm with vertex continuity and improved triangular blending algorithm with open set continuity. Both methods construct surfaces that are parametrically continuous at each point. The difference is that for a vertex of the input, the surfaces of the first algorithm are parametrically continuous only at the vertex itself, but the surfaces of the second algorithm are parametrically continuous inside an open set that includes the vertex.

Chapter 7

Multi-Sided Extension

In this chapter the scheme and algorithm for the case when the blended terms n is not restricted to 3 is discussed. In that case the scheme and algorithm can be used to process multi-sided inputs. The new results are still based on Chapter 3, so the behaviours of the multi-sided scheme and algorithm are similar to the triangular ones. The differences are listed in the following sections.

7.1 Multi-Sided Blending Scheme

For the multi-sided case, the blending domain T is defined with $n \geq 3$ in Definition 7.1.

Definition 7.1. *An n-sided blending domain T is a closed subset of the local coordinate system that is formed by n points $\{P_0, P_1, \ldots, P_{n-1}\}$ and n boundaries $\{B_0, B_1, \ldots, B_{n-1}\}$. For each $i \in \{0, 1, \ldots, n-2\}$, the two endpoints of B_i are P_i and P_{i+1}, and the two endpoints of B_{n-1} are P_{n-1} and P_0.*

Since the number of boundaries may be larger than three, the face domain cannot be parameterized by a barycentric coordinate system. So for the multi-sided case, all surfaces are parameterized by local coordinate systems.

Consider the face domain T inside the corresponding local coordinate system. In this chapter it is assumed that the boundaries of T are all straight, and the polygon formed by these boundaries is convex. (Although there is no such requirement for the data meshes, and the vertices of a face do not even have to be co-planar. The only assumption is that

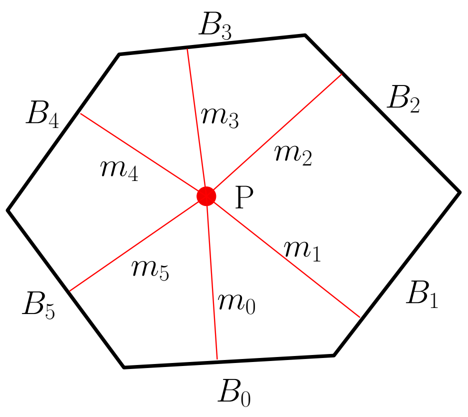


Figure 7.1: An example of the multi-sided inheritance indicators.

the projection of the faces into some plane are convex). Thus a new set of inheritance indicators can be defined by

Definition 7.2. *The n-sided inheritance indicators $m = (m_0, m_1, \ldots, m_{n-1})$ with respect to T are defined as for each $i \in N$, $m_i(x, y)$ is the Euclidean distance from (x, y) to the boundary B_i.*

Figure 7.1 shows an example of the multi-sided inheritance indicators.

Then by Definition 3.2, for each point in T, the corresponding value of the inherited terms q^m is known, and the corresponding blending functions $f_i^{n,d,m}$ for $i \in N$ are also known by Definition 3.4. This leads to the following theorem.

Theorem 7.1. *For n vertices $V_0, V_1, \ldots, V_{n-1}$, if the sub-surfaces $S_0, S_1, \ldots, S_{n-1}$ satisfy the conditions*

- *S_0 interpolates the locations and normals of V_0 and V_1.*
- *S_1 interpolates the locations and normals of V_1 and V_2.*
- *S_2 interpolates the locations and normals of V_2 and V_3.*
- ...
- *S_{n-2} interpolates the locations and normals of V_{n-2} and V_{n-1}.*
- *S_{n-1} interpolates the locations and normals of V_{n-1} and V_0.*

then the blended result S

$$S^{n,d,m}(x,y) = \sum_{i \in N} f_i^{n,d,m}(x,y) S_i(x,y) \tag{7.1}$$

is well-defined. Then on each edge $E = \{V_i, V_{i+1}\}$ where $i \in \{0, 1, \ldots, n-2\}$, $S^{n,d,m}$ has the same location and normal as the sub-surface S_i, and on the edge $E = \{V_{n-1}, V_0\}$, $S^{n,d,m}$ has the same location and normal as the sub-surface S_{n-1}. At last at each vertex V_i where $i \in \{1, 2, \ldots, n-1\}$, $S^{n,d,m}$ has the same location and normal as the sub-surfaces S_{i-1} and S_i, and at the vertex V_0, $S^{n,d,m}$ has the same location and normal as the sub-surfaces S_{n-1} and S_0.

Proof. The result can be obtained by the methods used in the proofs of Theorem 4.2 and Theorem 4.3. □

7.2 Multi-Sided Algorithm

The multi-sided blending algorithm has the same five steps as for the triangular algorithm given in Section 5.2. The first three steps are same as the triangular case, but the last two steps need to be updated since the face surfaces are now parameterized by local coordinate systems.

1. For each vertex V, construct a local coordinate system $\mathcal{F}^V$ and the surface S^V.
2. For each edge $E = \{V_0, V_1\}$, construct a local coordinate system $\mathcal{F}^E$ and reparameterize the surfaces S^{V_0} and S^{V_1} to obtain $S_E^{V_0}$ and $S_E^{V_1}$.
3. For each edge $E = \{V_0, V_1\}$, construct the surface $S^E = S^{\{V_0,V_1\}}$ by blending $S_E^{V_0}$ and $S_E^{V_1}$.
4. For each face $F = \{V_0, V_1, \ldots, V_{n-1}\}$, construct a local coordinate system $\mathcal{F}^F$ and reparameterize the surfaces $S^{\{V_0,V_1\}}$, $S^{\{V_1,V_2\}}$, $\ldots$, $S^{\{V_{n-2},V_{n-1}\}}$, and $S^{\{V_{n-1},V_0\}}$ to obtain $S_F^{\{V_0,V_1\}}$, $S_F^{\{V_1,V_2\}}$, $\ldots$, $S_F^{\{V_{n-2},V_{n-1}\}}$, and $S_F^{\{V_{n-1},V_0\}}$.
5. For each face $F = \{V_0, V_1, \ldots, V_{n-1}\}$, construct the surface $S^F = S^{\{V_0,V_1,\ldots,V_{n-1}\}}$ by blending $S_F^{\{V_0,V_1\}}$, $S_F^{\{V_1,V_2\}}$, $\ldots$, $S_F^{\{V_{n-2},V_{n-1}\}}$, and $S_F^{\{V_{n-1},V_0\}}$.

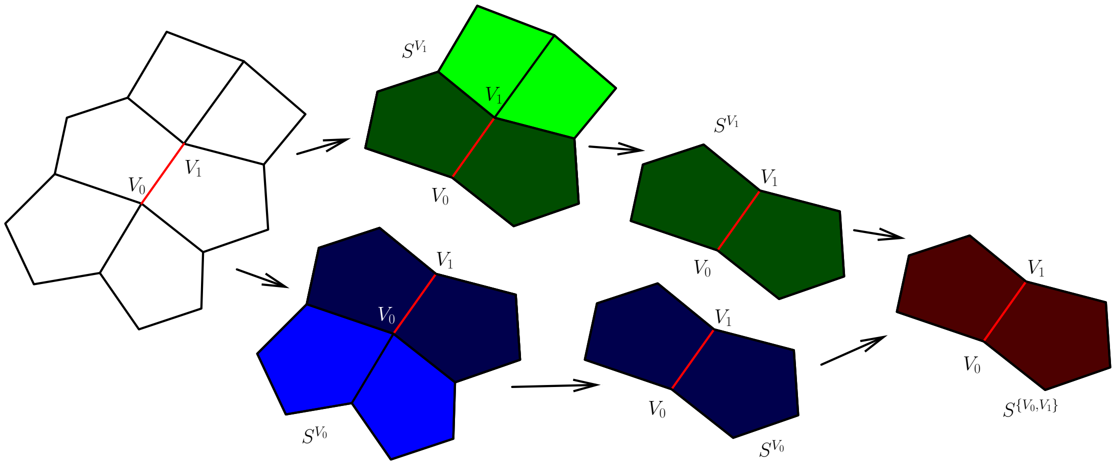


Figure 7.2: The construction of an edge surface for the multi-sided case.

The construction process of the edge surfaces and faces surfaces are illustrated in Figure 7.2 and Figure 7.3. The construction is similar to the triangular case shown in Figure 5.1 and Figure 5.2. In Figure 7.2, the edge labelled red is the edge that the surface is constructed for, and the mesh on the left shows its two vertices and their neighbours. The green and blue polygons are the vertex surfaces constructed for the two vertices of the edge in the first step, and the darker portions are the surface pieces that are blended in the next step. The construction of the edge surface is completed by blending these two darker portions, giving the red polygon on the right of Figure 7.2.

Now we need to combine multiple edge patches to obtain the surface patch for the face in the mesh. Figure 7.3 shows the mesh face (in the center) and the edge patches to be blended, where the darker portions of the edge patches are what get blended in step 5.

7.2.1 Edge to Face Reparameterization

A local coordinate system needs to be constructed for each face first. Consider a face F shown in Figure 7.4. Let n be the number of adjacent vertices and edges of V, and the vertices $V_0, V_1, \ldots, V_{n-1}$ and the edges $E_0, E_1, \ldots, E_{n-1}$ are located counter-clockwise around F with respect to the normal of F. The coordinate system $\mathcal{F}^F$ of the face F is constructed by the following steps. Notice that the vertices $V_0, V_1, \ldots, V_{n-1}$ are not required to be co-planar.

- Let the location of the center of the vertices $V_0, V_1, \ldots, V_{n-1}$ be the origin of $\mathcal{F}^F$.

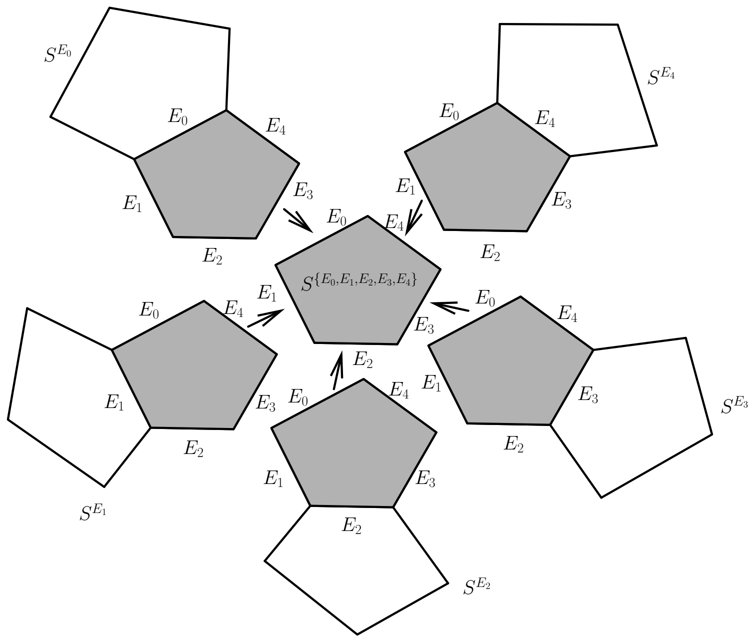


Figure 7.3: The construction of a face surface for the multi-sided case.

- Let the direction of the average of the normals of the vertices $V_0, V_1, \ldots, V_{n-1}$ be the direction of the z-axis of $\mathcal{F}^F$ (it is assumed that the sum of the normals are not zero).
- Let the direction from V to the projection of V_0 onto the xy-plane be the direction of the x-axis of $\mathcal{F}^F$, and the direction of the y-axis of $\mathcal{F}^F$ is the direction of the x-axis rotated by 90 degrees counter-clockwise.
- Let the length between V and the projection of V_0 be the unit length of $\mathcal{F}^F$.

Now reparameterize the surface of each edge. Similar to the reparameterizations between the surfaces of vertices and edges, only the edge being reparameterized needs to be interpolated, and the reparameterization is linear. For example, consider the edge E_0. Let (x_{head}, y_{head}) and (x_{tail}, y_{tail}) denote the locations of the head and the tail of the edge E_0 in $\mathcal{F}^F$. Then the vector $v_y = (x_{tail} - x_{head}, y_{tail} - y_{head})$ is mapped to the y-unit vector of $\mathcal{F}^{E_0}$, and $v_x = (y_{tail} - y_{head}, x_{head} - x_{tail})$ is mapped to the x-unit vector of $\mathcal{F}^{E_0}$. For any point (x, y) in $\mathcal{F}^F$, let $v = (x - x_{head}, y - y_{head})$, then

$$r^{F,E_0}(x, y) = \left(\frac{v \cdot v_x}{v_x \cdot v_x}, \frac{v \cdot v_y}{v_y \cdot v_y}\right). \tag{7.2}$$

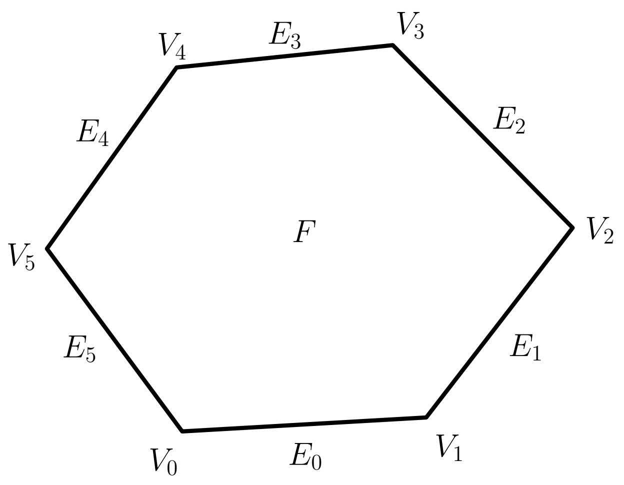


Figure 7.4: A face and its edges.

and the reparameterized surface of E_0 is

$$S_F^{E_0}(x, y) = S^{E_0}(r^{F,E_0}(x, y)). \tag{7.3}$$

By the same method the surface of other edges can also be constructed. Then for any $i \in N$,

$$S_F^{E_i}(x, y) = S^{E_i}(r^{F,E_i}(x, y)). \tag{7.4}$$

7.2.2 Face Surface Construction

The last step is to construct the surfaces of the faces. Consider the face F shown in Figure 7.4. The surface S^F is constructed by a surface blending with n targets. With $S_F^{E_0}$, $S_F^{E_1}$, . . ., and $S_F^{E_{n-1}}$ constructed with Equation 7.4, the surface S^F is constructed by

$$S^F(x, y) = \sum_{i \in N} f_i^{n,d,m}(x, y) S_F^{E_i}(x, y). \tag{7.5}$$

7.3 Modified Edge to Face Reparameterization

This section discusses the multi-sided version of the edge to face reparameterization of Section 6.1.1. Figure 7.5 shows the coordinate system $\mathcal{F}^E$ that will be constructed for an

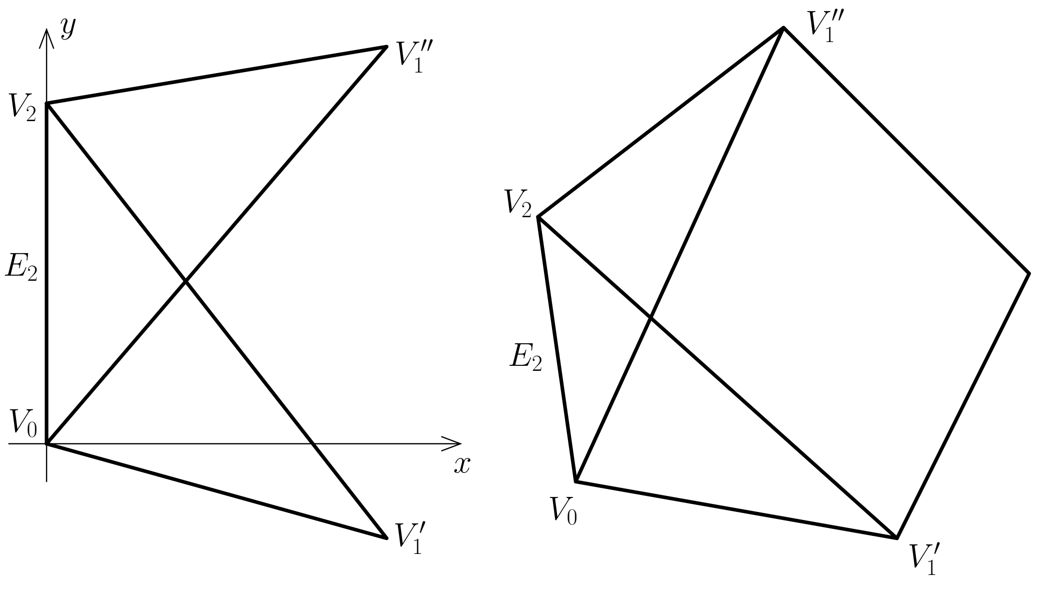


Figure 7.5: The reparameterization from edge to face.

edge E and an adjacent face F. The point V_0' is the point obtained by mapping the location of V_0 in $\mathcal{F}^{V_h}$ to $\mathcal{F}^E$ with the reparameterization function $r^{V_h,E}$, and the point V_1'' is the point obtained by mapping the location of V_1 in $\mathcal{F}^{V_t}$ to $\mathcal{F}^E$ with the reparameterization function $r^{V_t,E}$. Recall that most of the time V_1' and V_1'' are not the same.

Similar to the triangular case, two sub-reparameterizations need to be constructed. For the first sub-reparameterization the triangle $\triangle V_h V_0 V_t$ in $\mathcal{F}^F$ should be mapped to $\triangle V_h V_0' V_t$ in $\mathcal{F}^E$, and for the second sub-reparameterization the triangle $\triangle V_t V_h V_1$ in $\mathcal{F}^F$ should be mapped to $\triangle V_t V_h V_1'$ in $\mathcal{F}^E$. For any point (x, y) in $\mathcal{F}^F$, let (u_0, v_0, w_0) and (u_1, v_1, w_1) denote its barycentric coordinates with respect the triangles $\triangle V_h V_0 V_t$ and $\triangle V_t V_h V_1$ respectively. Let (x_{V_h}, y_{V_h}), (x_{V_t}, y_{V_t}), $(x_{V_0'}, y_{V_0'})$, and $(x_{V_1'}, y_{V_1'})$ denote the locations of V_h, V_t, V_0', and V_1' in $\mathcal{F}^E$. Then define two sub-reparameterizations $r_{V_h}^{F,E}$ and $r_{V_t}^{F,E}$ as

$$\begin{aligned} r_{V_h}^{F,E}(x,y) &= (u_0 x_{V_h} + v_0 x_{V_0'} + w_0 x_{V_t}, \\ &\quad\; u_0 y_{V_h} + v_0 y_{V_0'} + w_0 y_{V_t}), \end{aligned} \tag{7.6}$$

and

$$\begin{aligned} r_{V_t}^{F,E}(x,y) &= (u_1 x_{V_t} + v_1 x_{V_h} + w_1 x_{V_{V_1'}}, \\ &\quad\; u_1 y_{V_t} + v_1 y_{V_h} + w_1 y_{V_{V_1'}}). \end{aligned} \tag{7.7}$$

Then the modified reparameterization function $r^{F,E}$ is

$$r^{F,E}(x,y) = f_0^{2,d,b^r}(x,y)r_{V_h}^{F,E}(x,y) + f_1^{2,d,b^r}(x,y)r_{V_t}^{F,E}(x,y), \tag{7.8}$$

where $b^r = (b_0^r, b_1^r)$, and $b_0^r(x,y)$ is the distance from V_h to (x,y) and $b_1^r(x,y)$ is the distance from V_t to (x,y) in $\mathcal{F}^F$.

7.4 Summary

In this chapter I extended the surface blending scheme, the interpolation algorithm, and the parametric modification to the multi-sided case. The surfaces constructed by the multi-sided algorithm have the same order of continuity as the surfaces constructed by the triangular algorithm of Section 5.2.

Chapter 8

Examples

In Chapter 5, I gave some examples of my algorithm (more specifically, the improved algorithm in Chapter 6) on triangular meshes to compare my scheme to Nielson's scheme. In this chapter I give examples of surfaces constructed by the basic algorithm in Chapter 5, the improved algorithm in Chapter 6, and the multi-sided algorithm in Chapter 7. The remaining examples shown in this chapter are constructed by the blending algorithm with the blending order $d = 2$. With these examples I will show that the surfaces constructed by the algorithms with $d = 2$ have continuous normals and curvatures. As indicated in the chapter of the improved algorithms, the surfaces constructed by the improved algorithm are also locally continuous. For both the triangular and the multi-sided algorithms, several meshes as well as the corresponding shaded images, and the curvature plots as well as the isophotes are given to show the overall shapes and the continuity of curvatures.

8.1 Basic and Improved Algorithms

When $d = 2$, both the basic and improved algorithms construct surfaces with continuous curvature. The difference between these two algorithms is that the surfaces constructed by the improved algorithm are locally parametrically continuous at the vertices. The differences in the constructed surfaces are not obvious when the angles of the input mesh are close to flat. On a data mesh with sharper angles, the differences between the normals of vertices and the normals of adjacent faces are larger, and the advantages of the improved algorithm become clear. For comparison two meshes shown in Figure 8.1 and Figure 8.3 with sharp angles are given for the triangular and multi-sided algorithms respectively. Figure 8.2 and Figure 8.4 show the corresponding surfaces constructed by the basic and

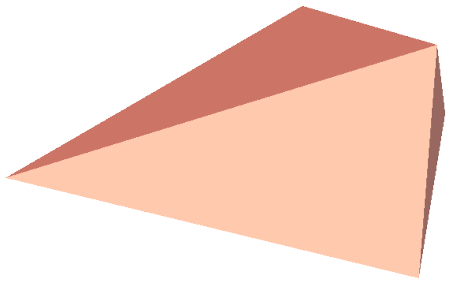


Figure 8.1: The triangular sharp mesh.

improved algorithms. Observe that the curvature plots of the improved algorithm do not have the "gaps" toward the vertices. Also, the basic algorithm has a large area of undesired negative curvature in Figure 8.4 that the improved algorithm does not have. For the isophotes, the results of the improved algorithm have isophotes of better shapes.

Because of the superior performance of the improved algorithms, the remaining examples in this chapter will be shown using only the improved algorithms.

8.2 Examples of the Improved Algorithm

In this section examples for different meshes are presented. All examples are constructed with the improved algorithm presented in Chapter 6 and Chapter 7 with the blending order $d = 2$. Notice that all the constructed surfaces in these examples have continuous normals and curvatures.

I start with a simple example. Figure 8.5 shows two simple triangulated grids, sampled from the fourth Franke's function [20]

$$f_4(x,y) = \frac{1}{3}e^{-\frac{81}{16}((x-\frac{1}{2})^2+(y-\frac{1}{2})^2)} \tag{8.1}$$

with densities 4×4 and 8×8 in the region $[0,1] \times [0,1]$, and Figure 8.6 shows the shaded images and the curvature plots as well as the isophotes of the resulting surfaces of two densities constructed by the triangular improved algorithm. Observe that taking higher density of sampling improves the shape of the resulting surface. However with the higher sampling, while the curvature looks better, the isophotes suggest there may be other issues with the surface quality.

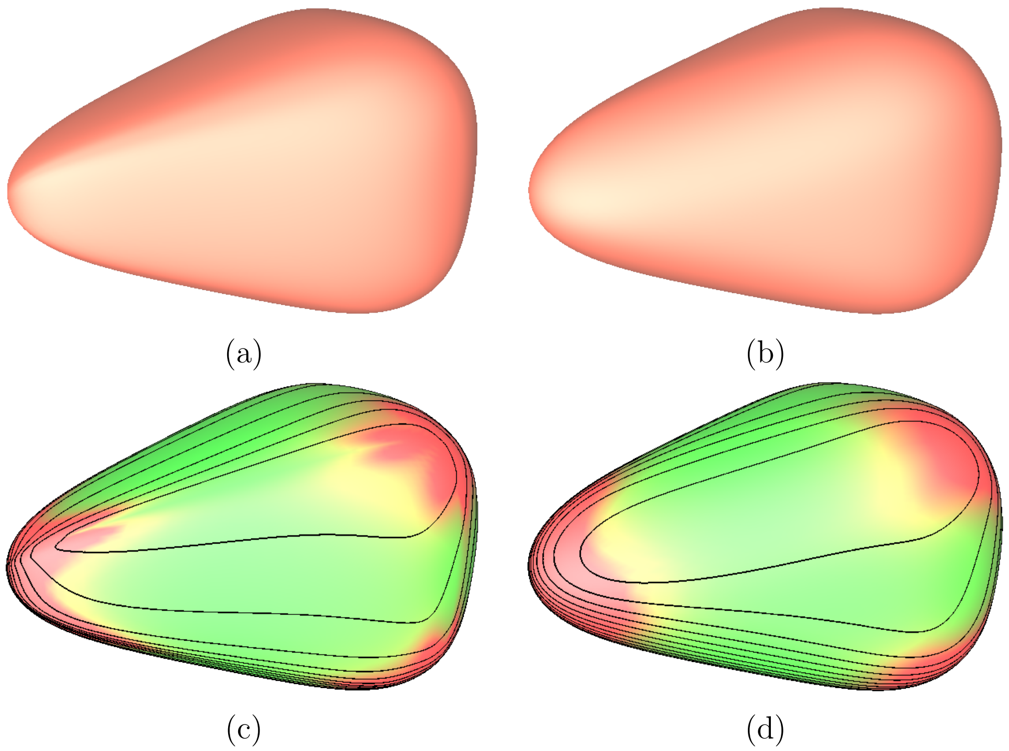


Figure 8.2: The surfaces constructed for the triangular sharp mesh by the basic and improved algorithms; (a) shaded image of the basic algorithm; (b) shaded image of the improved algorithm; (c) curvature plot and isophotes of the basic algorithm; (d) curvature plot and isophotes of the improved algorithm.

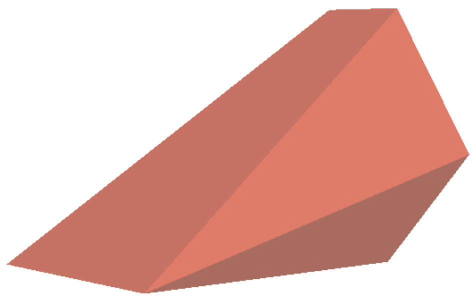


Figure 8.3: The quadrangular sharp mesh.

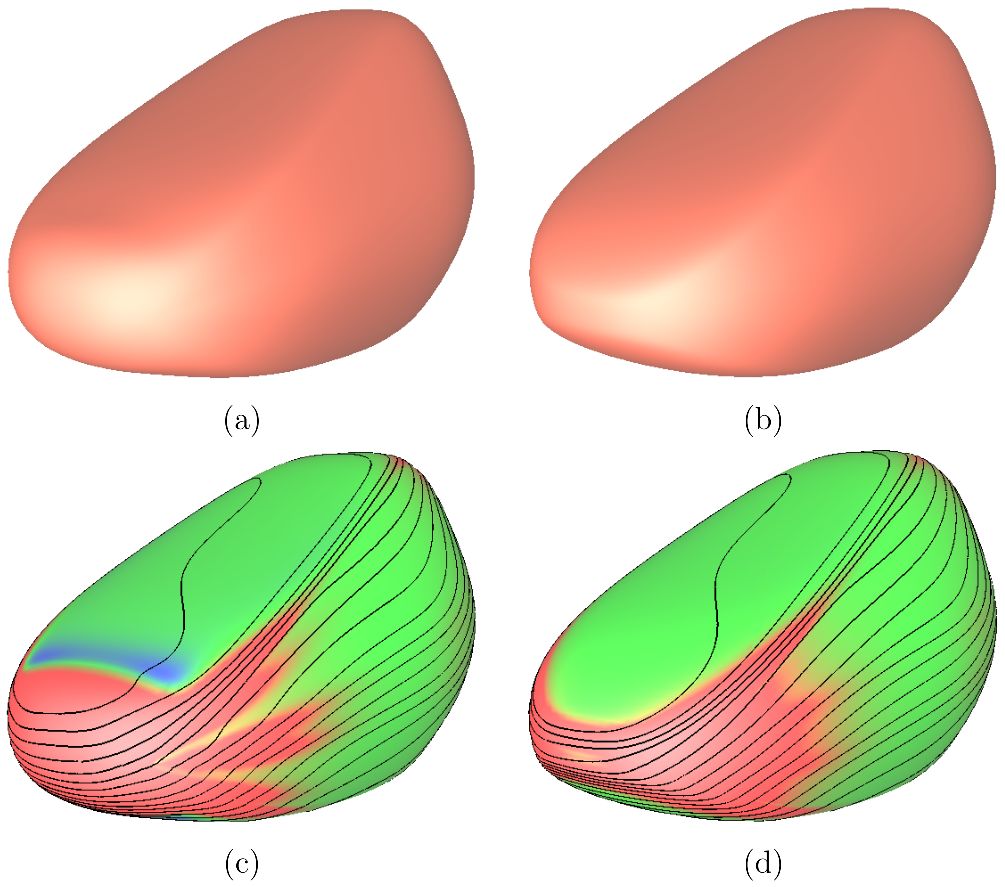

(a) (b) (c) (d)

Figure 8.4: The surfaces constructed for the quadrangular sharp mesh by the basic and improved algorithms; (a) shaded image of the basic algorithm; (b) shaded image of the improved algorithm; (c) curvature plot and isophotes of the basic algorithm; (d) curvature plot and isophotes of the improved algorithm.

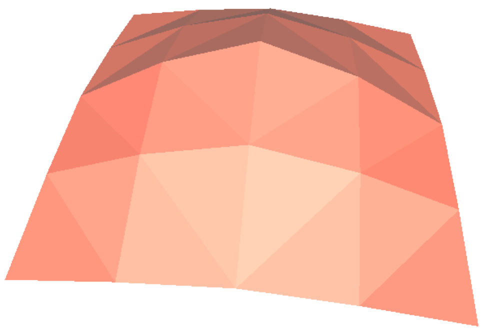
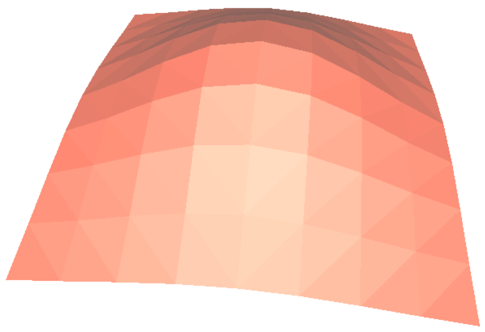

(a)

(b)

Figure 8.5: The simple meshes; (a) simple mesh of a density 4×4; (b) simple mesh of a density 8×8.

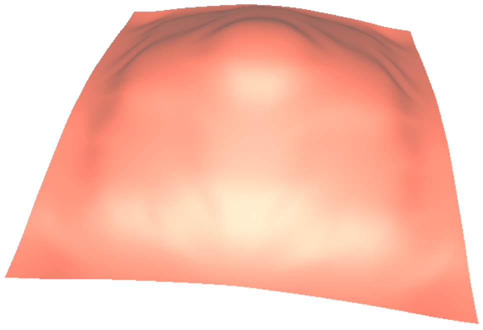
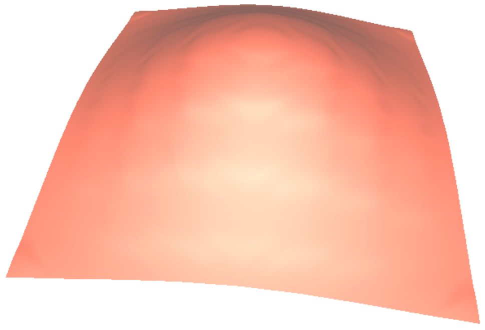
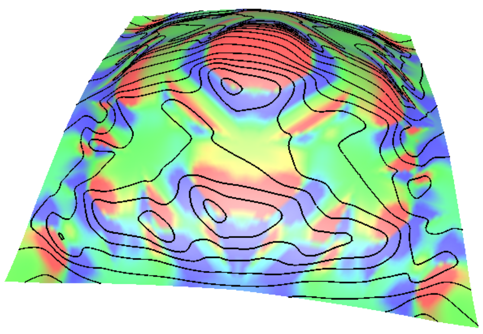
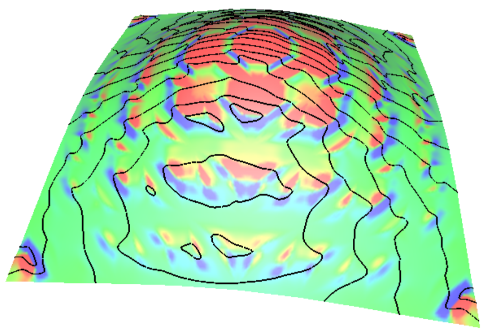

(a)

(b)

(c)

(d)

Figure 8.6: The surfaces constructed for the simple meshes by the improved algorithm; (a) shaded image of a density 4×4; (b) shaded image of a density 8×8; (c) curvature plot and isophotes of a density 4×4; (d) curvature plot and isophotes of a density 8×8.

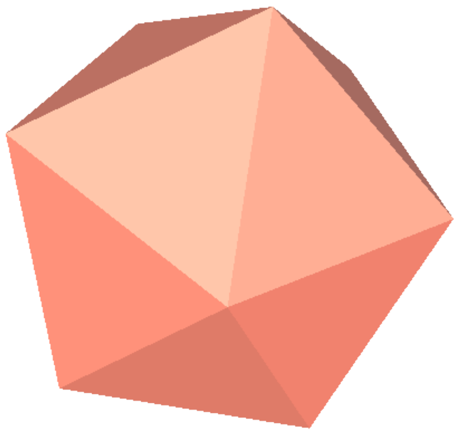


Figure 8.7: The icosahedron mesh.

Figure 8.7 and Figure 8.8 show a icosahedral data mesh and the surface constructed by the triangular algorithm for this mesh. The distance from each vertex to the center of the icosahedron is 1. Note that the high curvature near the mesh vertices and the relatively flat regions near the edges and centers of the triangles. The result shows that a regular input provides an output with better shape.

Figure 8.9 and Figure 8.10 show the data mesh and the results of an ellipsoid. The ellipsoid of Figure 8.9(b) is created by starting with a Pentakis icosidodecahedron of radius 1 (Figure 8.9(a)), and scaling it by a factor of 2 in one direction. The normals are then recalculated with respect to the ellipsoid. The constructed surface in Figure 8.10 does not have too extreme of curvature variation. Also since the ellipsoid mesh is convex, the resulting surface does not have negative Gaussian curvatures.

Figure 8.11 and Figure 8.12 show a perturbed data mesh and its results. The perturbed mesh is created by randomly shifting the vertices of a mesh of Pentakis icosidodecahedron of radius 1.5. The normal of each vertex is calculated by taking the average of the normals of its adjacent faces. The output become more non-uniform since the input is not close to a regular shape.

Figure 8.13 and Figure 8.14 show the data mesh and the results of a triangulated torus. The sampled torus has an outer radius 1.3 and an inner radius 0.7. The vertices are sampled by taking 12 around the outer ring, and taking 6 around the tube. Unlike the earlier data meshes, this data mesh has regions of both positive and negative Gaussian curvature. While the curvature plot of the resulting surface is not as smooth as that of a

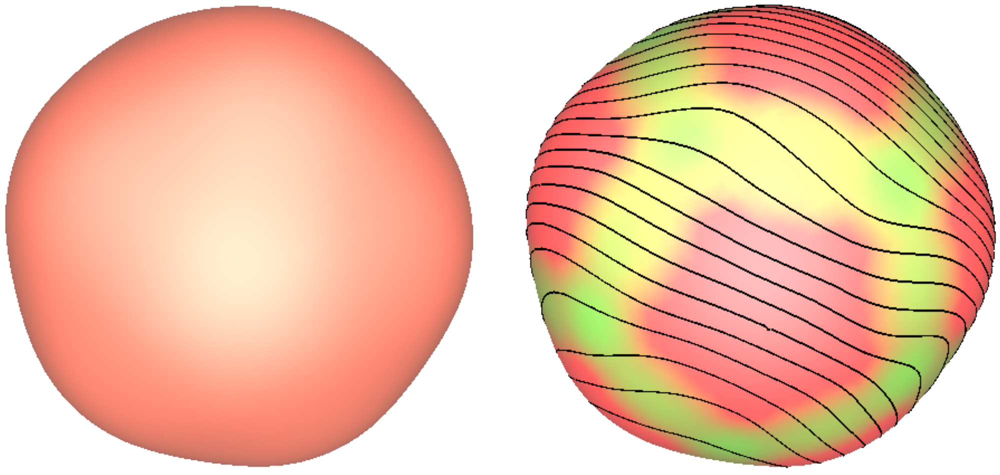

(a) (b)

Figure 8.8: The surface constructed for the icosahedron mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

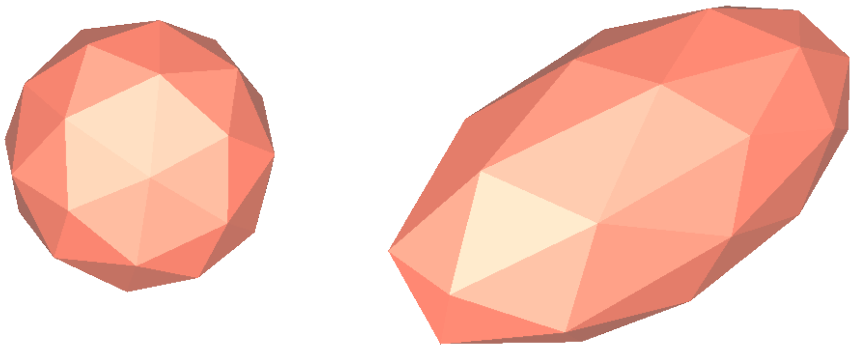

(a) (b)

Figure 8.9: The input meshes; (a) pentakis icosidodecahedron mesh; (b) ellipsoid mesh.

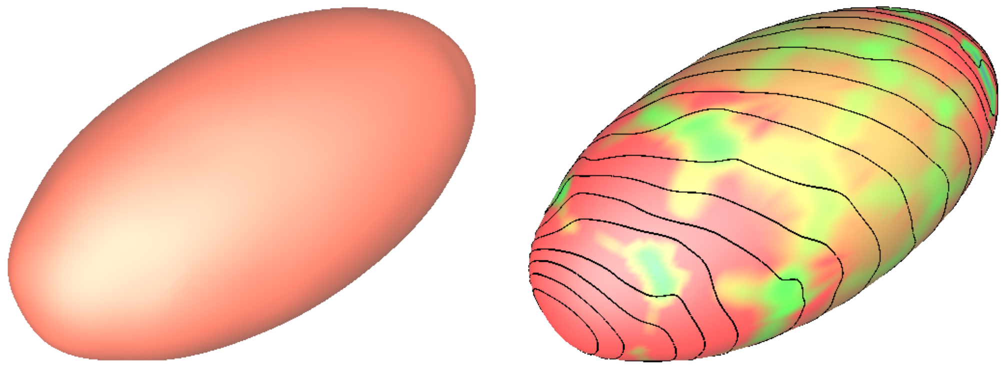

(a) (b)

Figure 8.10: The surface constructed for the ellipsoid mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

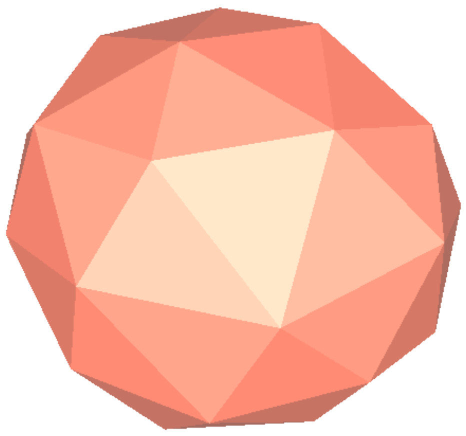


Figure 8.11: The perturbed mesh.

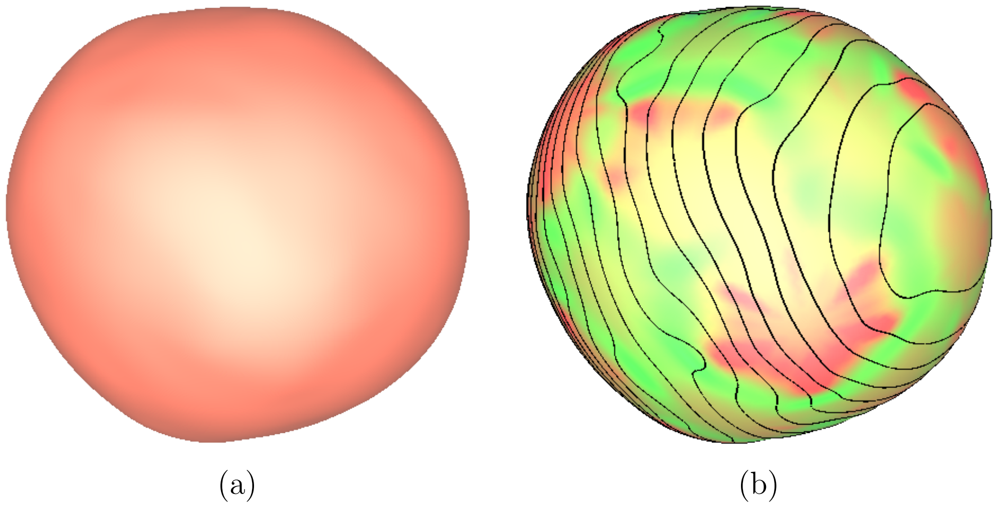


Figure 8.12: The surface constructed for the perturbed mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

torus, this curvature plot does have the expected positive and negative curvature variation.

Figure 8.15 and Figure 8.16 show a dodecahedral data mesh and the surface constructed by the multi-sided algorithm for this mesh. The distance from each vertex to the center of the dodecahedron is 1.66. While there is some curvature concentration, notice the lack of negative curvature in the example. This result shows that the multi-sided algorithm provides similar output as the triangular algorithm for the regular input.

Figure 8.17 and Figure 8.18 show a randomly generated mesh (called a "mixed" mesh) with a mix of 3- and 4-sided faces, where the vertices have been shifted off the surface of

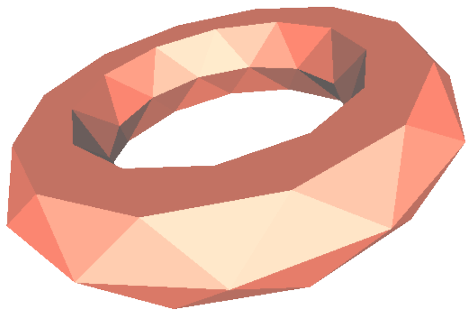


Figure 8.13: The triangular torus mesh.

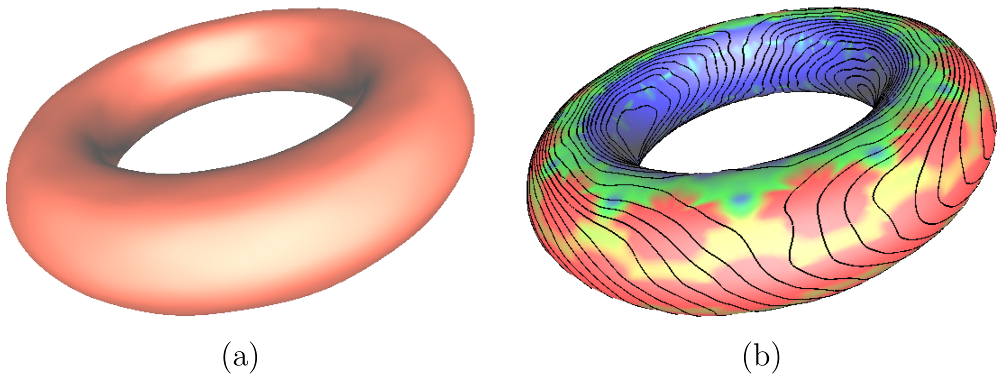

(a) (b)

Figure 8.14: The surface constructed for the triangular torus mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

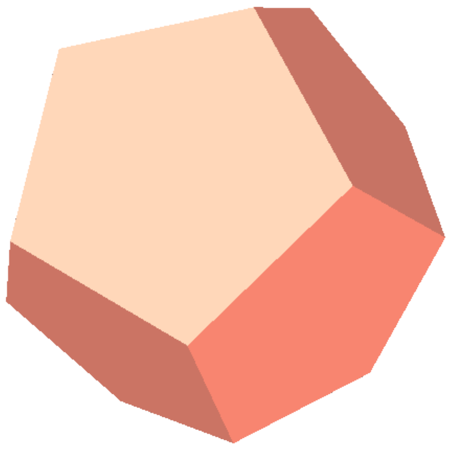


Figure 8.15: The dodecahedron mesh.

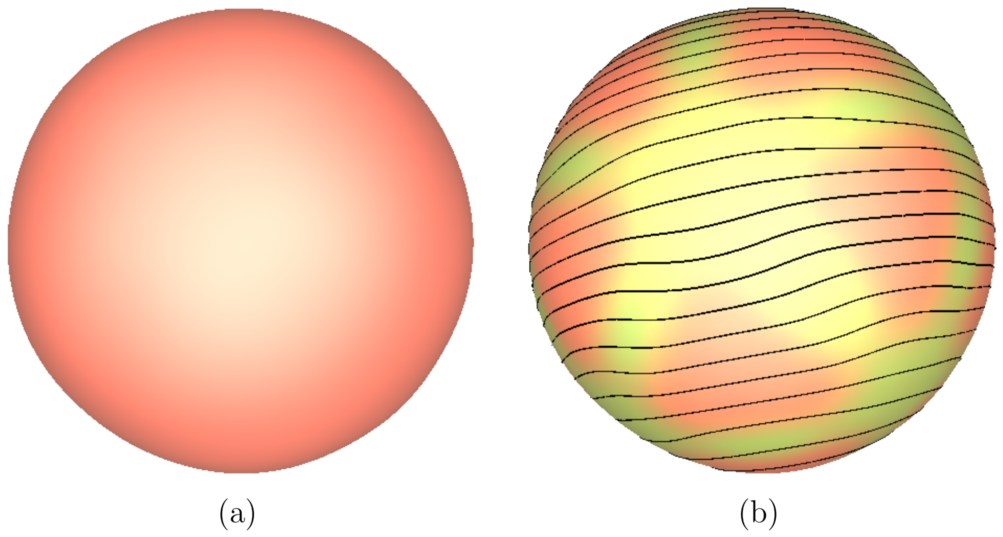


Figure 8.16: The surface constructed for the dodecahedron mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

a sphere, and its resulting surface. Similarly, notice the lack of negative curvature in the example. This result shows that the multi-sided algorithm provides similar output as the triangular algorithm for the irregular input.

Figure 8.19 and Figure 8.20 show a 4-sided mesh sampled from a torus and the result of the multi-sided algorithm fit to this mesh. The sampled torus has an outer radius 1.25 and an inner radius 0.75. The vertices are sampled by taking 16 around the outer ring, and taking 8 around the tube. This result shows the expected mix of positive and negative curvatures, and also shows that the multi-sided algorithm provides similar output as the triangular algorithm for the surface with negative Gaussian curvature.

Overall, while my algorithms are performed reasonably well on these data sets, the constructed surfaces showed curvature concentration that is typical of this type of data fitting scheme.

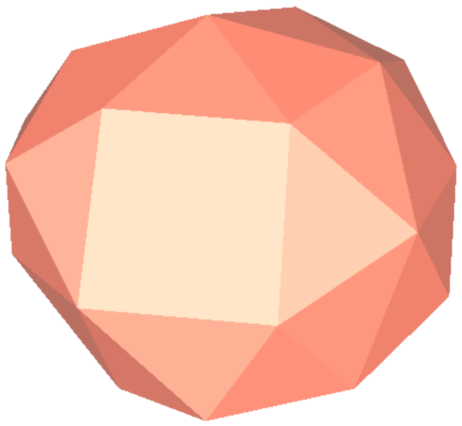


Figure 8.17: The mixed mesh.

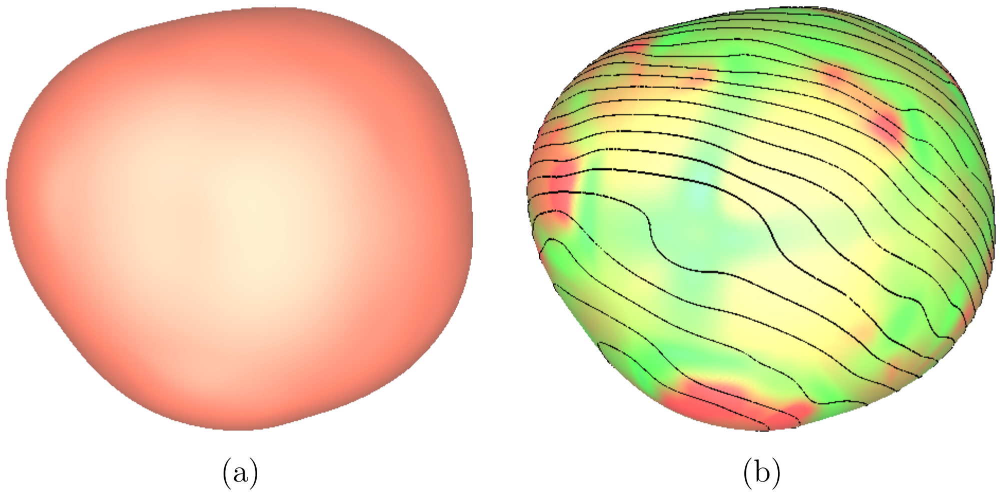


Figure 8.18: The surface constructed for the mixed mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

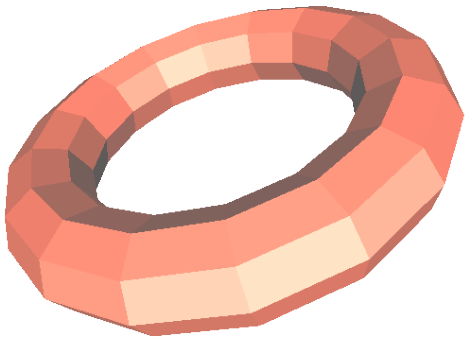


Figure 8.19: The quadrangular torus mesh.

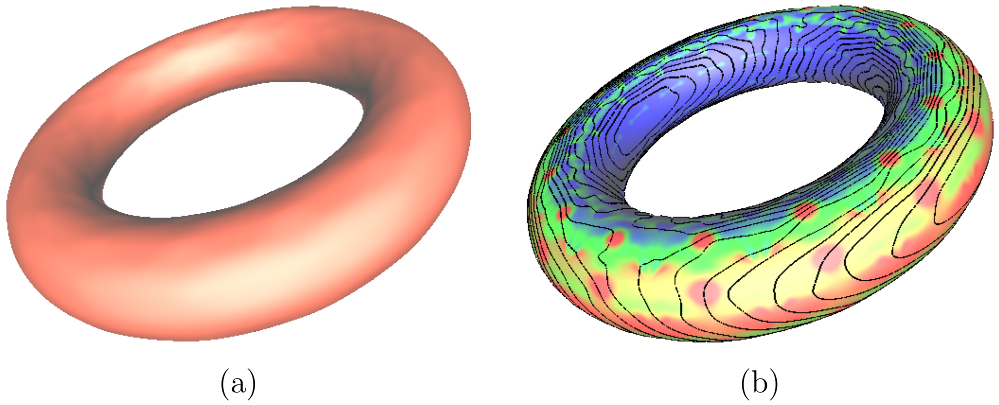


Figure 8.20: The surface constructed for the quadrangular torus mesh by the improved algorithm; (a) shaded image; (b) curvature plot and isophotes.

Chapter 9

Conclusions and Future Work

In this dissertation the ideas for solving the scattered data interpolation problem are discussed. Central to my work is the definition and the usage of the new blending functions defined in Chapter 3. The base case can be considered as an extension of the blending functions of Nielson's method [17], so the properties of my blending functions are similar to the properties of Nielson's blending functions. The new blending functions work for triangular interpolation, blend three triangular sub-surfaces, and provide a resulting surface of the same shape. Similar to Nielson's method, the resulting surface inherits the derivatives of the sub-surfaces on the boundaries. More specifically, for each boundary there is a corresponding sub-surface, and the derivatives of the resulting surface are equal to the derivatives of the sub-surface on that boundary. Furthermore, the new scheme has extra properties:

- Although I only constructed C^2 surfaces, the order of derivatives is extended to arbitrary values. That leads to a result that the scheme can provides surfaces that are continuous of arbitrary order. Such value only affects the formulas of the new blending functions, and does not cause any extra cost when constructing the surface.
- The restrictions of the sub-surfaces are relaxed. For Nielson's method all of the three sub-surfaces need to interpolate all of the three vertices of the triangular face. However with the new blending functions, each sub-surface only need to interpolate the two vertices of the corresponding edge. That property provides more flexibility when constructing the sub-surface. For the minimal requirement, each sub-surface can be constructed with respect to the edge, instead of the whole face. Furthermore, the two sub-surfaces of the two adjacent faces with respect to the same edge can be constructed together.

The basic scheme of the new blending functions only provides the continuity inside the triangles and across the boundaries. So based on the properties of the new blending functions, an algorithm for constructing smooth triangular surfaces with globally geometric continuity is described. The algorithm accepts any triangular data sites as input, and provides surfaces that are geometric continuous of arbitrary order everywhere.

Then I extended the new blending functions and surface construction to the multi-sided case. The formulas of an arbitrary number of new blending functions are given and the corresponding scheme accepts the same number of sub-surfaces. The algorithm using these new blending functions is also given. The new algorithm accepts data sites formed by multi-sided polygons.

At last the discussion of parametric continuity of the resulting surfaces is given. It is proven that for each point on the resulting surface, there exists a corresponding coordinate system. The resulting surface is parametric continuous with respect to the coordinate system. The order of parametric continuity is equal to the order of geometric continuity.

9.1 Future Work

This section discusses future work, which is divided into two categories: improving the shape of the resulting surfaces, and updating the scheme and algorithm to accept more possible inputs. These updates are presented with respect to the complete surface blending algorithm. Furthermore, the single blending scheme may also be applied to other applications.

9.1.1 Functional Interpolation

The algorithms introduced in this thesis are constructed for parametric interpolation. The reparameterization operations in the algorithm are necessary since each element of the input data mesh has its own local coordinates, and most of the time it is impossible to make them share the same coordinates. However for functional data (for example the input meshes in Figure 8.5), all elements can share the same coordinates. In this case all reparameterization operations are unnecessary. Furthermore the operations of parametric improvement in Chapter 6 are not required, and the interpolation results always have the properties of local parametric continuity. With this approach, the surfaces will likely have better shape, and metrics like order of convergence could used to evaluate the method. Investigating this variation of the method is an area of future work.

9.1.2 Blending Functions

In this dissertation I used my blending functions to create new schemes and algorithms. However such choice is not unique. In Section 4.3 I mentioned that the Nielson's blending functions can be extended to increase the degree of continuity. Consider the multi-sided approach in Chapter 7, Nielson's functions can also be extended to blending more than three sub-surfaces. Combine these two directions, it provides a new set of intermediate functions

$$g_i^{Nielson,n,d}(v) = \Big(\prod_{j\in N\setminus\{i\}} w_j\Big)/\Big(\sum_{j\in N}\prod_{k\in N\setminus\{j\}} w_k\Big), \tag{9.1}$$

where $v = (v_0, v_1, \ldots, v_{n-1})$ and $w_i = v_i^{d+1}$ for each $i \in N$. These functions are also called *special side blending functions* [22].

If the numerator and denominator of the extended Nielson's functions are both divided by the product of all parameters $\prod_{i\in N} w_i$, the functions become

$$s_i(v) = \Big(\frac{1}{w_i}\Big)/\Big(\sum_{j\in N}\frac{1}{w_j}\Big), \tag{9.2}$$

where $v = (v_0, v_1, \ldots, v_{n-1})$ and $w_i = v_i^{d+1}$ for each $i \in N$. Thus, the extended Nielson's blending functions are the *Shepard's functions* [21], an inverse distance weighting method to solve multivariate interpolation problem. This shows that the Shepard's functions might be substituted into my algorithms and replace the roles of my blending functions. Validating these possibilities of the substitution is an area of future work.

9.1.3 Surface Shape Improvement

For the surface blending algorithms presented in this dissertation, only the continuity of the resulting surfaces are discussed. So the resulting surfaces may not have good shape. As show in the examples in Figure 5.8 and Figure 5.11, the G^1 surfaces do not visually have better shape than Nielson's results. (Although by some other criteria such as the discontinuity of curvature and isophotes the shape may be quantified.) Note that Nielson uses a side-vertex method to construct his sub-surfaces. In Figure 9.1, I constructed a surface using Nielson's scheme and I constructed a second surface using Nielson's side-vertex sub-surfaces but blended these sub-surfaces using the blending functions given by Equation 4.2 with $d = 0$. In this figure, the two surfaces are almost identical, suggesting that the sub-surfaces that I used in my dissertation are suboptimal. Thus a major direction of

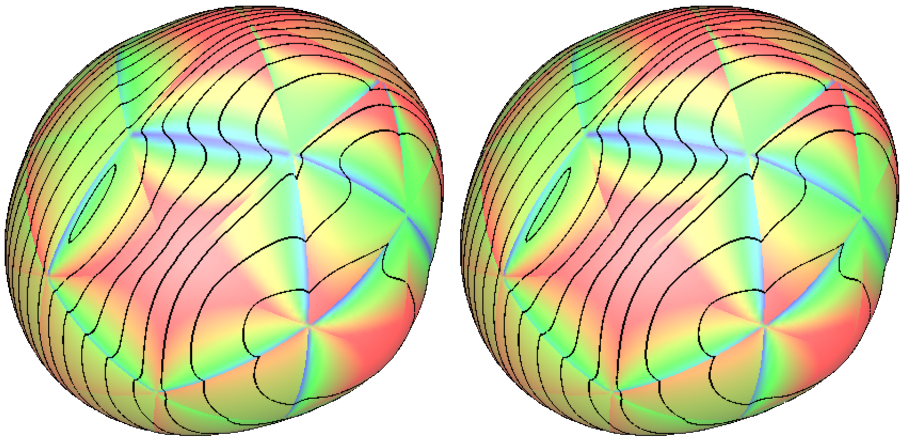


Figure 9.1: The blending results of Nielson's sub-surfaces; left: by Nielson's blending functions; right: by the new triangular functions with $d = 0$.

the future work is to improve the shape of the resulting surfaces. In the blending algorithm presented in Chapter 5, the shape of the final resulting surface is mainly determined by three points:

- The method to construct the vertex surfaces.
- The reparameterization functions between the coordinate systems of the adjacent elements.
- The choice of the inheritance indicators.

For the vertex surface construction, currently the polynomial Least method is used to construct the vertex surface. For now the vertex surface only interpolates the vertex itself and its neighbours. Potentially, more data could be interpolated, and additional guidance could be provided by the user. The algorithm should work not only the information carried by the input data mesh, but also with customized shape parameters provided by the users. Further, there are many shape parameters in the sub-surfaces. An obvious next step is to use various techniques to improve the shape of the constructed surface.

Another possible method to improve the shapes of the resulting surfaces is to introduce extra sub-surfaces to blend. These surfaces should affect the final shapes of the resulting surface, but should not change the derivatives on the boundaries. There are two possible methods to achieve this goal. The first method is to assign small but non-zero inheritance indicators to these sub-surfaces. Another method is to directly modify the blending functions to avoid increasing the number of inheritance indicators. For example for the triangular case, for an additional fourth sub-surface, the blending functions are changed to

$$\begin{aligned}
f_0^{3,d,t}(u,v,w) &= \beta\gamma\Big(\frac{1}{\alpha+\beta}+\frac{1}{\alpha+\gamma}\Big)\Big(\frac{1}{\alpha+\beta+\gamma+\alpha\beta\gamma}\Big), \\
f_1^{3,d,t}(u,v,w) &= \gamma\alpha\Big(\frac{1}{\beta+\gamma}+\frac{1}{\beta+\alpha}\Big)\Big(\frac{1}{\alpha+\beta+\gamma+\alpha\beta\gamma}\Big), \\
f_2^{3,d,t}(u,v,w) &= \alpha\beta\Big(\frac{1}{\gamma+\alpha}+\frac{1}{\gamma+\beta}\Big)\Big(\frac{1}{\alpha+\beta+\gamma+\alpha\beta\gamma}\Big), \\
f_3^{3,d,t}(u,v,w) &= \alpha\beta\gamma\Big(\frac{1}{\alpha+\beta+\gamma+\alpha\beta\gamma}\Big),
\end{aligned} \tag{9.3}$$

where $\alpha = u^{d+1}$, $\beta = v^{d+1}$, and $\gamma = w^{d+1}$. The blending function $f_3^{3,d,t}$ is the desired one that affect the shape but not the derivatives. The idea is to use the first three blending functions to achieve continuity, and to use $f_3^{3,d,t}$ to blend in a surface to improve the shape.

9.1.4 Input Extension

For the multi-sided blending algorithm, it is assumed that each face of the input data mesh is projected into a convex polygon. The reason for this assumption is that the current inheritance indicators for each face domain are constructed with simple distances. For some special case such inheritance indicators cannot be applied. An example is shown in Figure 9.2. Suppose that two edges E_0 and E_1 are co-planar, then they are also co-planar on the face domain after projected. With current inheritance indicators, the blending scheme always provides the same weight for two different edges E_0 and E_1. Such a result is not acceptable since their corresponding sub-surfaces will conflict each other.

To avoid this problem, a new method to construct the inheritance indicators is required. One method to explore is to construct a polar parameterization for each point inside the polygon, then choose the lengths of the shortest paths from the point to the edges as the values of the inheritance indicators.

9.1.5 Additional Applications

Various applications of my surface construction remain to be explored. One application that stands out is filling holes around extraordinary points in subdivision surfaces. My n-sided patch construction could blend the surfaces created by the subdivision scheme around the n-sided hole, creating a patch that meets these surfaces with top order continuity. Since

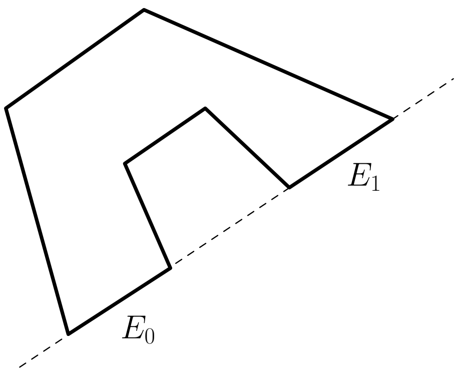


Figure 9.2: A convex face with two edges co-planar.

schemes for filling such holes already exist, the main issues to explore would be those of shape and a comparison to existing methods.

Finally, my method has potential applications where controlling jerk is important in CNC machining and in computer animation, where at times C^3 surfaces are required.

References

[1] Carl De Boor and Amos Ron. Computational aspects of polynomial interpolation in several variables. *Mathematics of Computation*, 58(198):705 – 727, 1992.

[2] Fünfzig Christoph, Müller Kerstin, Hansford Dianne, and Farin Gerald. PNG1 triangles for tangent plane continuous surfaces on the GPU. *Proceedings of Graphics Interface 2008*, pages 219 – 226, 2008.

[3] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 3rd edition, 1993.

[4] Michael S. Floater and Francesco Patrizi. Transfinite mean value interpolation over polygons. *Numerical Algorithms*, 85:995–1003, 2020.

[5] Thomas A. Foley and Karsten Opitz. Hybrid cubic bézier triangle patches. *Mathematical Methods in Computer Aided Geometric Design II*, pages 275 – 286, 1992.

[6] John A. Gregory. Smooth interpolation without twist constraints. *Computer Aided Geometric Design*, pages 71 – 87, 1974.

[7] John A. Gregory and Peter Charrot. A C^1 triangular interpolation patch for computer-aided geometric design. *Computer Graphics and Image Processing*, 13(1):80 – 87, 1980.

[8] Hans Hagen and Helmut Pottmann. Curvature continuous triangular interpolants. *Mathematical Methods in Computer Aided Geometric Design*, pages 373 – 384, 1989.

[9] Kirk Haller and Stephen Mann. Error sensitive multivariate polynomial interpolation. *University of Waterloo Technical Report*, 2021.

[10] Ali Hashemian, Pengbo Bo, and Michael Barton. Reparameterization of ruled surfaces: Toward generating smooth jerk-minimized toolpaths for multi-axis flank cnc milling. *Computer-Aided Design*, 127:102868, 2020.

[11] Gary Herron. Smooth closed surfaces with discrete triangular interpolants. *Computer Aided Geometric Design*, 2(4):297 – 306, 1985.

[12] Gerben J. Hettinga and Jiri Kosinka. Multisided generalisations of gregory patches. *Computer Aided Geometric Design*, 62:166–180, 2018.

[13] Xin Jiang, Yifei Hu, Guanying Huo, Cheng Su, Bolun Wang, Hexiong Li, LiYong Shen, and Zhiming Zhen. Asymmetrical pythagorean-hodograph spline-based C^4 continuous local corner smoothing method with jerk continuous feedrate scheduling along linear toolpath. *The International Journal of Advanced Manufacturing Technology*, 121:5731 – 5754, 2021.

[14] Stephen Mann. Cubic precision Clough-Tocher interpolation. *Computer Aided Geometric Design*, 16(2):85 – 88, 1999.

[15] Stephen Mann and Matthew Davidchuk. A parametric hybrid triangular bézier patch. *Proceedings of the International Conference on Mathematical Methods for Curves and Surfaces II Lillehammer*, pages 335–342, 1997.

[16] Stephen Mann, Charles Loop, Michael Lounsbery, David Meyers, James Painter, Tony Derose, and Kenneth Sloan. A survey of parametric scattered data fitting using triangular interpolants. *Curve and Surface Design*, pages 145–172, 1992.

[17] Gregory Nielson. A transfinite, visually continuous, triangular interpolant. *Geometric Modeling: Algorithms and New Trends*, pages 235 – 246, 1987.

[18] Barrett O'Neill. *Elementary Differential Geometry.* Academic Press, 2nd edition, 2006.

[19] Jörg Peters. Joining smooth patches around a vertex to form a C^k surface. *Computer Aided Geometric Design*, 9(5):387 – 411, 1992.

[20] Franke Richard. A critical comparison of some methods for interpolation of scattered data. 1979.

[21] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. *Proceedings of the 1968 23rd ACM National Conference*, pages 517 – 524, 1968.

[22] Tamás Várady, Alyn Rockwood, and Péter Salvi. Transfinite surface interpolation over irregular n-sided domains. *Computer-Aided Design*, 43(11):1330–1340, 2011.

[23] Alex Vlachos, Jörg Peter, Chas Boyd, and Jason L. Mitchell. Curved PN triangles. *Bi-Annual Conference Series*, 2001, 2001.

Glossary

***d*-th matched** The properties of the sub-surfaces that the adjacent pair of sub-surfaces share the same d-th derivatives at the corner of the blending domain. 35

blended result The result provided by the blending schemes. 17

blended terms The number of sub-functions blended by a blending scheme. 17

blending domain A closed polygon formed by the zero domain of the inheritance indicators. 33

blending function The weight used in the blending schemes. 17

blending order The order of derivative that the blended result inherits. 18

inheritance A property of the blending schemes that the blended result has the same derivatives as the sub-functions on certain locations. 17

inheritance indicator A set of functions indicates the locations on which the derivatives should be inherited. 18

inherited terms The number of sub-functions from which the derivatives are inherited for certain locations. 18

parametrically adjacent The property of a pair of elements of the data mesh that there exists reparameterization between their coordinate systems in the blending algorithm. 45

sub-function The the functions that blended by the blending schemes. 17

sub-surface The sub-functions which are also surfaces. 32

triangular blending function The blending function for the triangular blending scheme. 34

triangular inheritance indicator The inheritance indicator for the triangular blending scheme. 34

value matched The properties of the sub-surfaces that the adjacent pair of sub-surfaces share the same location at the corner of the blending domain. 35