Robust Geometric Methods
for Surface Modeling and Manufacturing

by

Jordan Philip Smith

B.S. (UC Berkeley) 1996
M.S. (UC Berkeley) 2003

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Carlo H. Séquin, Chair
Professor Jonathan R. Shewchuk
Professor Paul K. Wright

Spring 2004

The dissertation of Jordan Philip Smith is approved:

Chair Date

Date

Date

University of California at Berkeley

Spring 2004

Robust Geometric Methods
for Surface Modeling and Manufacturing

by
Jordan Philip Smith

Abstract

Robust Geometric Methods
for Surface Modeling and Manufacturing

by

Jordan Philip Smith
Doctor of Philosophy in Computer Science

University of California at Berkeley

Professor Carlo H. Séquin, Chair

We investigate methods to aid in the process of going from a conceptual shape of a part to its physical realization. We present contributions to the "design for manufacture" problem in two major areas: surface model representation and process planning for layered manufacturing. In surface modeling, the focus is on subdivision surfaces and the analysis of their continuity behavior. We present a new algorithm for the exact evaluation of piecewise smooth Loop and Catmull-Clark surfaces near user-defined sharp features. This algorithm aids in the geometric process planning of subdivision surfaces for manufacturing. On the process planning side, we have developed an approach to layered manufacturing that produces parts with thin, dense walls filled with a loose web-like interior to reduce material usage and accelerate build times. The key algorithm in this process generates a conservative offset of a polyhedral surface by generating a stack of 2D slice contours, calculating individual offset contours, and forming appropriate Boolean combinations of these contours. We describe a numerically robust algorithm for constructing the generalized Voronoi diagram of polygonal slice contours which is then used to generate clean offset contours for each slice, which are combined between layers to create the conservative 3D offset surface.

Professor Carlo H. Séquin
Dissertation Committee Chair

Contents

List of Figures

Chapter 1

Introduction

1.1 Solids Modeling for Manufacturing

The goal of my research has been to aid in the process of going from a design idea to the physical manufacture of that shape. My work in the “design for manufacturing” (DFM) process has been twofold. First, I have been searching for a representation of shape geometry that is both easy and powerful for the designer and leads to a geometry that is unabiguous to the downstream manufacturing processes. Second, I have made contributions to the geometry processing necessary for certain types of free-form fabrication methods.

Shape design for consumer products have become more “organic” in nature. Even products such as the casings for desktop computers which could be simple rectilinear boxes are becoming more free-form to be more pleasing to the consumer’s eye. The design interface must allow for the intuitive description of smooth surfaces of arbitrary genus which may be interrupted with discontinuous crease features. Ideally the designer would specify a set of positional and differential geometry constraints for the surface, and an optimization process could be run to fill in the rest of the surface in a “pleasing way”. Subdivision surfaces are becoming an ideal substrate for specifying these types of free-form surfaces. These same surface primitives are used in film production to represented animated computer graphic characters in movies.

On the manufacturing end, the surface representation must describe an unambiguous solid that is watertight without self intersections. The generation of a process plan for automatic fabrication of a part, requires execution of many geometric queries and operations on this surface representation. These computer aided design (CAD) operations include sampling the position, tangents, and principle curvatures of the surface at a parametrice location; generating clean offset surfaces;

and making Boolean combinations of surfaces. These CAD operations must be numerically robust, so that the part can be automatically generated.

1.2 Free-Form Manufacturing

The types of manufacturing methods we are addressing fall into the category of free-form fabrication. Traditional manufacturing methods made it difficult to produce free-form surfaces due to difficult fixturing issues. However, new layered manufacturing technologies that build up both the desired part along with a disposable custom fixturing and support structure make it possible to construct intricate geometric surfaces. The free-form technologies that we have had practical experience with are Fused Deposition Modeling (FDM), wax deposition, and 3D Printing, but the geometric process planning techniques that we are focusing on tend to span all of the layered manufacturing processes. For layered manufacturing, a first requirement for the shape description is to have a clean, closed boundary representation of the model. Process planning then involves slicing the model with parallel planes perpendicular to the build axis, and performing 2D offsets and Boolean combinations among these slice planes. Knowledge of the differential geometry of the surface can be used to optimize the process planning, for instance tessellating more finely in areas of higher curvature to reduce discretization error. We will review concepts of the differential geometry of parametric surfaces in Chapter 2.

Numerically-controlled milling is a subtractive manufacturing method where the negative volume is removed from a piece of stock material, leaving behind the desired part. For round free-form parts, there are many difficulties with fixturing the work piece securely enough to resist the forces and torques of the cutting head without interfering with the cutting plan. While it is very difficult to clamp down a part composed of all free-form surfaces, techniques such as reference free part encapsulation (RFPE) [24] and five axis milling do allow for generating interesting surfaces. Process planning for N.C. milling also is much more difficult and involved that for layered manufacturing, but many of the same queries to the surface representation will be required. For example, the principle curvatures at a point on the surface dictate the maximum radius cutting tool that can be used for stock removal there and specify the tool path spacing needed to achieve a bound on scallop height. The scallop height is the height of excess material remaining between two parallel tool passes of a ball end mill.

With these downstream constraints of the manufacturing processes in mind, we would like to give the designer a part specification tool that describes shapes in such a way that the part

can be manufactured, while still giving the designer the freedom to explore interesting geometries. Subdivision surfaces are a very expressive and intuitive parametric boundary representation. A designer can guarantee that the limit surface is watertight simply by defining a closed polyhedral control mesh. We will discuss properties and some advancements in the field of subdivision surfaces that make them an ideal surface description for the "design for manufacturing" problem.

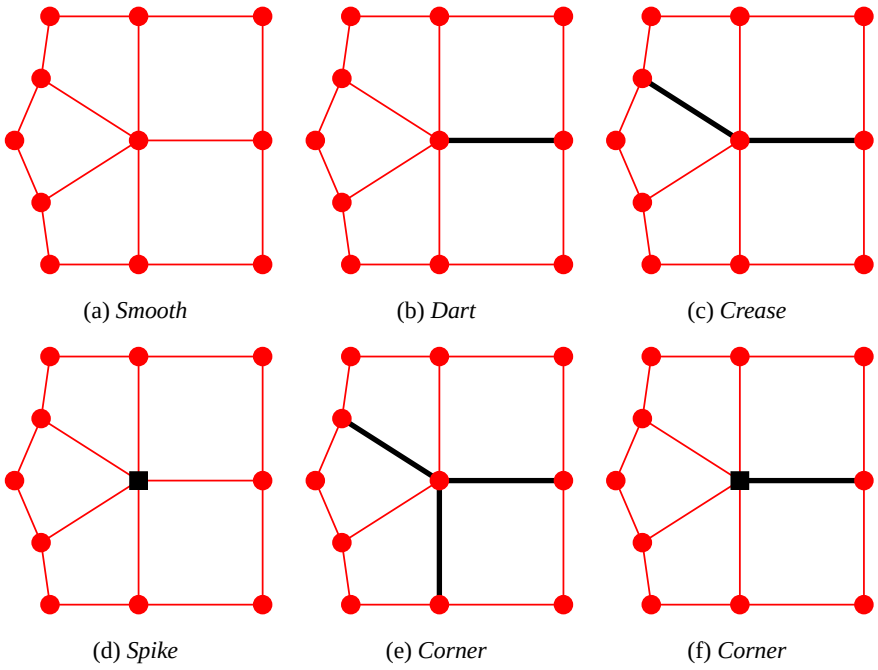
1.3 Manifold Surfaces

We would like to describe watertight, closed 2D manifold surfaces. Non-uniform rational B-splines (NURBS) have been the computer-aided design (CAD) industry standard for many years. The difficulty with NURBS is that they are patch based. A NURBS patch is specified by a rectilinear array of user specified control vertices weighted by tensor product basis function humps. To maintain the guaranteed continuity given by the basis functions for a NURBS surface of order k, $k-1$ of the parametric lines of control vertices must be shared from one patch to the next in both parametric directions. The result is that the valence of the vertices in the defining control mesh must all be valence $n = 4$. The topology of closed surfaces that meet this requirement is limited to the family of tori, which is not very interesting. To define more general surfaces of arbitrary mesh connectivity topology with NURBS, it is necessary to knit together multiple patches that must meet at an extraordinary vertex of valence $n \neq 4$. In this case, the control structure is not a 2D manifold mesh. At an extraordinary vertex, there are a multitude of control points, but they are not all true degrees of freedom. To achieve even the minimum C^0 positional continuity, many of these control points must be constrained to be functions of other control points. Achieving higher degrees of continuity, such as tangent plane continuity and curvature continuity, requires even more constraints on the control points. From a designer's point of view, it is preferable for all control vertices to be true degrees of freedom while still guaranteeing a given degree of continuity.

Subdivision surfaces [5, 15] solve this user interface problem. Subdivision surfaces are generalizations of uniform box spline schemes, e.g. B-splines, to meshes of arbitrary topology. Chapter 3 is an introduction to subdivision surfaces and the analysis of their continuity behavior. Box splines have the property that the same limit surface can be represented by a finer set of control vertices using the technique of knot insertion. A uniform subdivision scheme of a box spline performs a constant number of knot insertions per knot interval, usually one at the midpoint. The process of uniform knot insertion over the entire control structure defines a topological split of the control mesh, followed by a set of averaging rules to generate the control vertex positions in the

finer mesh, which are weighted by more local basis functions. Subdivision surface schemes such as the ones defined by Loop [15] and Catmull-Clark [5] generalize these averaging rules at vertices with extraordinary valences, while maintaining a certain degree of continuity. All of the control vertices of a subdivision surface are true, user-defined degrees of freedom, and any manipulation of their positions still maintains a closed 2-manifold shape with a certain level of parametric continuity. Conceptually after an infinite number of refinement steps, the control structure approaches the smooth limit surface, which is thus a closed 2-manifold surface.

Subdivision surfaces are ideal for creating organic looking smooth geometries, but it is often necessary or desirable for the designer to intentionally break this smoothness by adding sharp features. The control mesh can be tagged to specify that portions of the mesh be refined by different rules that will create sharp feature curves and points [10, 6, 4]. The types of extraordinary vertices present in the mesh are: *smooth*, *spikes*, *darts*, *creases*, and *corners* (Figure 1.1). While the surface topology at a smooth interior point is a full disc, the local surface topology of a point on one these sharp boundary features is a half disc.



(a) *Smooth* (b) *Dart* (c) *Crease*

(d) *Spike* (e) *Corner* (f) *Corner*

Figure 1.1: *Examples of the five different extraordinary vertex types for piecewise smooth Catmull-Clark surfaces. Black thick lines and squares denote sharp edges and vertices. Light thin lines and circles represent smooth edges and vertices.*

A dart vertex (Figure 1.1(b)) has one sharp edge incident on it, but it is still averaged by

the smooth Catmull-Clark vertex rule, forming a transition from the feature curve to the smooth surface. A crease vertex (Figure 1.1(c)) has two sharp edges incident on it, and it is smoothed by the B-spline subdivision curve rule. A spike vertex (Figure 1.1(d)) is tagged not to move, while the rest of the surface around it follows the standard Catmull-Clark rules, leading to geometry resembling the tip of a cone. A corner vertex is defined to be a vertex with 3 or more sharp edges incident on it (Figure 1.1(e)), or a vertex with 1 or more sharp edges where the vertex is also tagged to not move (Figure 1.1(f)). With this expanded set of tagging rules, it possible with one surface representation to describe shapes that range from completely smooth free-form shapes to faceted polyhedral shapes using the same closed control mesh.

1.4 Continuity Requirements

To know and control the degree of continuity of the surface is important for many reasons. From the design point of view, the need for smoothness could be for purely aesthetic reasons or functional reasons like reducing air resistance for car bodies. For manufacturing, properties of the surface such as the position, tangent plane, and principle curvatures at a parametric location may be used to generate and optimize process plans. The position is used to guide the tool that either adds or removes material to generate the surface. In the case of milling, the tangent plane is used to orient the tool head to optimally remove material and avoid gouging. Curvature, along with a global distance function over space, define the local feature size of a surface. This local feature size defines the largest ball end mill that can be used to carve out the surface. The curvature also specifies the density of tool paths necessary to achieve scalloping below a tolerance for a given size mill. For layered manufacturing methods, the curvature information of the surface can be used to adaptively tessellate the surface to allot more triangles to areas of the surface that require it. Adaptive tessellation reduces the amount of data necessary to be transferred to the manufacturer as well as reducing the amount of computing time needed to generate the low level process plans.

The above requirements strongly favor a surface description for which it is fast and easy to evaluate the position, tangent plane, and principle curvatures at arbitrary parametric locations. It is then necessary to calculate up to the second derivative of the surface with respect to the two parameter directions. The subdivision schemes discussed in this thesis will be C^2 continuous except at isolated extraordinary vertices, where they will be C^1 continuous. Until recently, it was unknown how to do exact evaluation of subdivision surfaces at arbitrary locations, but recent results have shown how to do evaluation of **smooth** Catmull-Clark [29] and smooth Loop [27] surfaces using

the eigen-decomposition of the subdivision matrix. In this thesis, we introduce a new technique, using the Jordan decomposition of the possibly defective subdivision, matrices for the evaluation of piecewise smooth Loop and Catmull-Clark subdivision with sharp tagging rules [10, 6], which are the RenderMan standard.

Chapter 4 shows how the 2D subdivision averaging rules can be viewed as a difference equation over a 1D signal, and how to use this notion to derive analytic forms for the eigenvalues and eigenvectors of the subdivision matrix. Chapter 5 describes how to transform the circulant and Toeplitz shaped subdivision matrices to block diagonal form, which is an alternate way of understanding the eigen-decomposition nature of these matrices. Chapter 6 describes in general the Jordan decomposition based exact evaluation method for subdivision surfaces. Chapters 7 and 8 fill in the details for Loop and Catmull-Clark subdivision respectively. Chapter 9 provides a description of our *half-edge* structure used to represent the subdivision mesh.

1.5 Robustness of Geometric Algorithms

We want to make the physical generation of 3D geometric models as easy and robust as printing 2D postscript documents. This requires a number of geometric process planning algorithms. Ideally, we would like these algorithms to execute and complete without human intervention. One of the biggest problems with geometric algorithms is numeric robustness. Subdivision surfaces are good because the closed manifold structure stays throughout the tessellation process. Further more, the local linear averaging and limit position masks are less prone to numerical round off errors than evaluating high degree polynomials. In terms of robustness of algorithms, errors in the numeric calculation of subdivision vertex positions will not prevent the tessellation algorithm from completing because the algorithm does not make logical decisions based on these values.

However, many of the other algorithms involved in process planning do make logical decisions based on numeric calculations. One such algorithm is the creation of clean offset curves or surfaces. Offset curves are used in layered manufacturing and milling to define contour-parallel tool paths for tracing out volumes to be filled or removed. In our paper about thin-walled layered manufacturing [19], we show how the generation of a 3D offset surface can be used to optimize the build time and material usage in FDM. This algorithm makes a thin stiff wall near the surface and uses a looser honeycomb weave for the interior. This greatly reduces the amount of material used to build the model, which in turn greatly reduces the building time on the FDM machine. This is because the material is extruded at a constant feed rate, so the build time is proportional to the

volume of material used. The generation of a clean 3D offset surface is a very hard problem, but fortunately for layered manufacturing the output tool paths are 2D slice planes so we were able to reduce our problem to 2D curve offsetting. We slice the original polyhedral model into planes, which each contain a set of polygonal contours. For each plane, we generate the 2D offset curves of this set of polygonal contours, using a robust algorithm that I implemented. Boolean combinations of the slice contours and 2D offset curves from a number of slice layers n proportional to the desired wall thickness are then combined to create slice contours that correspond to an inwardly conservative $2\frac{1}{2}$D approximation to the true 3D offset surface.

I will only focus on my contribution to the thin-walled algorithm which is the robust generation of 2D offset curves from sets of polygonal contours. The naive approach to creating offset curves is to take each input edge segment and vertex and generate their offset line segment or circular arc, respectively, creating one parametrically continuous offset curve. Unfortunately, some of these offset segments will intersect each other creating loops in the offset curve. A de-looping algorithm must then be run to detect self intersections, split the curve into separate loops at the intersection points, and then at the end decide which loops are spurious and should be deleted. The difficulty here is detecting the global interaction of when one local offset front intersects another front from a topologically distant input source. At this point the offset curve must change topology. Making logical decisions based on generated geometry, like the local fronts, is dangerous because the exact values of those entities may not be representable on a computer in a finite number of bits. Then all decisions made using these inputs will have some degree of uncertainty to them. Even if we are unconcerned with slight positional and topological inaccuracies in our output as long as they are self consistent, it still may not be possible to implement an algorithm that always runs to completion under these assumptions. Geometric algorithms are usually designed using the assumption of the real Euclidean plane, but computers can only represent entities that lie on a discrete grid. While the input vertices may be on this discrete grid, geometry generated by offsetting them will pass through a continuum of points on the real plane. Hence points generated by intersecting these offset curves may not be representable. The real problem arises if the algorithm makes decisions based on multiple geometric relationships that are consistent in the Euclidean plane but are not necessarily consistent given floating point computation. If the algorithm depends on the consistency of these relationships, then it is liable to either crash or produce garbage output.

An alternate approach for constructing the offset curves is to first generate a generalized Voronoi diagram where the input sites are the vertices and edge segments. The Voronoi diagram is a topological mesh that encodes the global distance function. The generated Voronoi vertices are

equidistant from three input sites. The Voronoi diagram when viewed in 3D, with the Z-dimension being the distance from the input contours, looks like a mountain. Offsetting the input contours is equivalent to creating a level-set slice of the Voronoi mountain which can be done as robustly as slicing the original polyhedral input. The difficult part is generating the Voronoi diagram. The algorithm to generate the Voronoi diagram depends on two geometric predicates. A predicate is a decision function that returns a value that is either positive, negative, or zero in the case of a tie. The first predicate describes whether three ordered input sites inscribe a counterclockwise oriented circle. The second predicate checks whether a fourth site penetrates the circle inscribed within three other input sites. The algorithm depends on the consistency of the logical answers returned by these two separate predicates. If we represented these circles as a generated center point and radius, then we would have the same numeric difficulties found in the intersection based approach. Instead, we always return to the original input site data when evaluating a predicate. We are also careful to always pass the vertices in the same order for both predicates. For efficiency, an interval arithmetic filter is used try to get a quick answer using floating point arithmetic. If the interval of uncertainty does not include zero then the result can be trusted. Otherwise, a slow, infinite precision calculation is used. The package that we used for these extended precision calculations is mpfun [3]. Chapter 10 describes the numerically robust generalized Voronoi diagram of polygonal contours used to generate 2D offset curves.

Chapter 2

Differential Geometry of Surfaces

2.1 Introduction

Differential geometry of a 2-manifold [11] or surface embedded in 3D is the study of the intrinsic properties of the surface as well as the effects of a given parameterization on the surface. For the discussion below, we will consider the local behavior of a surface S at a single point p with a given local parameterization (u, v), see Figure 2.1. We would like to calculate the principle curvature values (κ_1, κ_2), the maximum and minimum curvature values at p, as well as the local orthogonal arc length parameterization which is aligned with the directions of maximum and minimum curvature $\{E_1, E_2\}$.

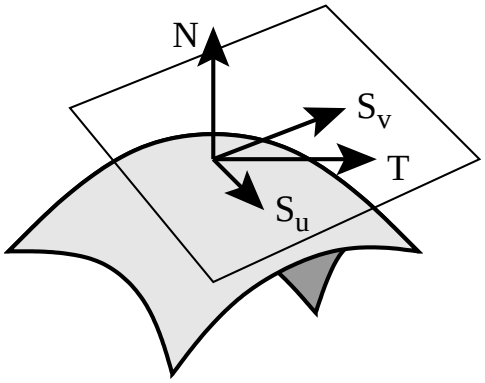


Figure 2.1: *Tangent plane of S*

2.2 The First Fundamental Form

With the given parameterization, we can compute a pair of tangent vectors $\begin{bmatrix} S_u & S_v \end{bmatrix} = \begin{bmatrix} \frac{\partial S}{\partial u} & \frac{\partial S}{\partial v} \end{bmatrix}$ assuming the u and v directions are distinct. These two vectors locally parameterize the tangent plane to S at p. An arbitrary tangent vector T can be constructed as a linear combination of $\begin{bmatrix} S_u & S_v \end{bmatrix}$ scaled by infinitesimal coefficients $U = \begin{bmatrix} \partial u & \partial v \end{bmatrix}^t$.

$$T = \partial u S_u + \partial v S_v = \begin{bmatrix} S_u & S_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.1}$$

Consider rotating the direction of the tangent T around p by setting $\begin{bmatrix} \partial u & \partial v \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix}$. In general, the magnitude and direction of T will vary based on the skew and scale of the basis vectors $\begin{bmatrix} S_u & S_v \end{bmatrix}$, see Figure 2.3(a). The distortion of the local parameterization is described by the metric tensor or the first fundamental form I_S. This measures the length of a tangent vector on a given parametric basis by examining the quantity $T \cdot T$.

$$\begin{aligned}
T \cdot T &= (S_u \cdot S_u)\,\partial u^2 + 2\,(S_u \cdot S_v)\,\partial u \partial v + (S_v \cdot S_v)\,\partial v^2 && (2.2)\\
&= E\partial u^2 + 2F\partial u\partial v + G\partial v^2 && (2.3)\\
&= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} S_u \\ S_v \end{bmatrix} \cdot \begin{bmatrix} S_u & S_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} && (2.4)\\
&= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} S_u \cdot S_u & S_u \cdot S_v \\ S_u \cdot S_v & S_v \cdot S_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} && (2.5)\\
&= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} E & F \\ F & G \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} && (2.6)\\
&= U^t I_S U && (2.7)
\end{aligned}$$

Equation 2.7 defines I_S.

$$I_S = g_{ij} = \begin{bmatrix} S_u \\ S_v \end{bmatrix} \cdot \begin{bmatrix} S_u & S_v \end{bmatrix} \tag{2.8}$$

$$= \begin{bmatrix} S_u \cdot S_u & S_u \cdot S_v \\ S_u \cdot S_v & S_v \cdot S_v \end{bmatrix} \tag{2.9}$$

$$= \begin{bmatrix} E & F \\ F & G \end{bmatrix} \tag{2.10}$$

If an orthonormal arc length parameterization (s, t) is chosen then there will be no local metric distortion and I_S will reduce to the identity matrix.

The determinant of I_S is proportional the local differential area square because it is equal to the square of the area of the parallelogram bordered by $\{S_u, S_v\}$.

$$Det\,(I_S) = (S_u \cdot S_u)\,(S_v \cdot S_v) - (S_u \cdot S_v)^2 = \|S_u \times S_v\|^2 \tag{2.11}$$

2.3 The Second Fundamental Form

The unit normal N of a surface S at p is the vector perpendicular to S, i.e. the tangent plane of S, at p. N can be calculated given a general nondegenerate parametrization (u, v).

$$N = \frac{S_u \times S_v}{\|S_u \times S_v\|} \tag{2.12}$$

The curvature of a surface S at a point p is defined by the rate that N rotates in response to a unit tangential displacement as in Figure 2.2. A normal section curve at p is constructed by intersecting S with a plane normal to it, i.e a plane that contains N and a tangent direction T. The curvature of this curve is the curvature of S in the direction T. The curvature κ of a curve is the reciprocal of the radius ρ of the best fitting osculating circle, i.e. $\kappa = \frac{1}{\rho}$. The directional derivative N_T of N in the direction T is a linear combination of the partial derivative vectors $\begin{bmatrix} N_u & N_v \end{bmatrix} = \begin{bmatrix} \frac{\partial N}{\partial u} & \frac{\partial N}{\partial v} \end{bmatrix}$ and is parallel to the tangent plane. By differentiating Equation 2.12 we obtain Equations [2.13,2.14].

$$N_u = \frac{(S_{uu} \times S_v) + (S_u \times S_{vu})}{\|S_u \times S_v\|} + (S_u \times S_v) \frac{\partial \|S_u \times S_v\|^{-1}}{\partial u} \tag{2.13}$$

$$N_v = \frac{(S_{uv} \times S_v) + (S_u \times S_{vv})}{\|S_u \times S_v\|} + (S_u \times S_v) \frac{\partial \|S_u \times S_v\|^{-1}}{\partial u} \tag{2.14}$$

$$N_T = \partial u N_u + \partial v N_v = \begin{bmatrix} N_u & N_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.15}$$

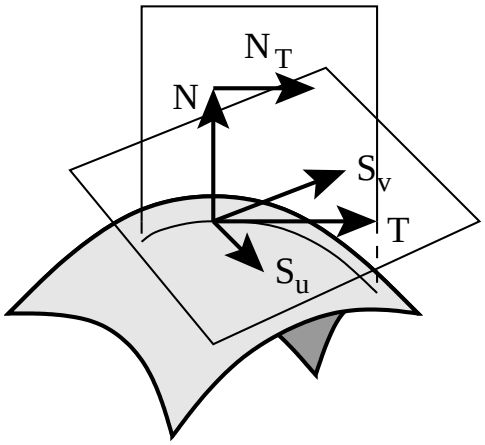


Figure 2.2: *Normal section of S*

As with the tangent vector T, N_T is subject to scaling by the metric of the parameter space. In an arc length parameter direction s, $-S_s \cdot N_s$ is the curvature of the normal section. Given a general parameterization (u, v), we can compute $-T \cdot N_T$. This quantity defines the second fundamental form II_S which describes curvature information distorted by the local metric.

$$\begin{aligned} -T \cdot N_T &= -(S_u \cdot N_u)\,\partial u^2 - (S_u \cdot N_v + S_v \cdot N_u)\,\partial u \partial v \\ &\quad -(S_v \cdot N_v)\,\partial v^2 && (2.16) \\ &= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} -S_u \\ -S_v \end{bmatrix} \cdot \begin{bmatrix} N_u & N_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} && (2.17) \\ &= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} -S_u \cdot N_u & -S_u \cdot N_v \\ -S_v \cdot N_u & -S_v \cdot N_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} && (2.18) \\ &= U^t II_S U && (2.19) \end{aligned}$$

Equation 2.19 defines II_S. II_S can be simplified to a set of quanties which are easier to compute by substituting the expressions in Equations [2.13,2.14] for N_u and N_v respectively. Two of the terms of the entries of the matrix in Equation 2.22 derived from Equations [2.13,2.14] vanish because the dot product of orthogonal vectors is zero. Further simplification using properties of the box product of vectors yields Equation 2.23. Equation 2.24 is derived by subsituting N from Equation 2.12 and shows that II_S can be computed simply by the dot product of the second partial derivatives of S with N. We assume that $S_{uv} = S_{vu}$ for all surfaces S that we will consider. Hence Equation 2.24 becomes Equation 2.25, so II_S is a symmetric matrix.

$$II_S = h_{ij} = \begin{bmatrix} S_u \\ S_v \end{bmatrix} \cdot \begin{bmatrix} -N_u & -N_v \end{bmatrix} \tag{2.20}$$

$$= \begin{bmatrix} -S_u \cdot N_u & -S_u \cdot N_v \\ -S_v \cdot N_u & -S_v \cdot N_v \end{bmatrix} \tag{2.21}$$

$$= \begin{bmatrix} -\frac{S_u \cdot (S_{uu} \times S_v)}{\|S_u \times S_v\|} & -\frac{S_u \cdot (S_{uv} \times S_v)}{\|S_u \times S_v\|} \\ -\frac{S_v \cdot (S_u \times S_{vu})}{\|S_u \times S_v\|} & -\frac{S_v \cdot (S_u \times S_{vv})}{\|S_u \times S_v\|} \end{bmatrix} \tag{2.22}$$

$$= \begin{bmatrix} \frac{S_{uu} \cdot (S_u \times S_v)}{\|S_u \times S_v\|} & \frac{S_{uv} \cdot (S_u \times S_v)}{\|S_u \times S_v\|} \\ \frac{S_{vu} \cdot (S_u \times S_v)}{\|S_u \times S_v\|} & \frac{S_{vv} \cdot (S_u \times S_v)}{\|S_u \times S_v\|} \end{bmatrix} \tag{2.23}$$

$$= \begin{bmatrix} S_{uu} \cdot N & S_{uv} \cdot N \\ S_{vu} \cdot N & S_{vv} \cdot N \end{bmatrix} \tag{2.24}$$

$$= \begin{bmatrix} S_{uu} \cdot N & S_{uv} \cdot N \\ S_{uv} \cdot N & S_{vv} \cdot N \end{bmatrix} \tag{2.25}$$

$$= \begin{bmatrix} L & M \\ M & N \end{bmatrix} \tag{2.26}$$

Equations [2.25,2.26] can be substituted for II_S to yield alternate formulas for $-T \cdot N_T$.

$$-T \cdot N_T = (S_{uu} \cdot N)\,\partial u^2 + 2\,(S_{uv} \cdot N)\,\partial u \partial v + (S_{vv} \cdot N)\,\partial v^2 \tag{2.27}$$

$$= L\partial u^2 + 2M\partial u\partial v + N\partial v^2 \tag{2.28}$$

$$= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} S_{uu} \cdot N & S_{uv} \cdot N \\ S_{vu} \cdot N & S_{vv} \cdot N \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.29}$$

$$= \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} L & M \\ M & N \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.30}$$

II_S contains information about the curvature of S, but it will be distorted by the metric if the given parameterization is not an orthonormal arc length parameterization. It is still necessary to counteract this distortion in order to compute the curvature of the S.

2.4 Coordinate Transformations

Our purpose in studying differential geometry and the fundamental forms has been to compute the principle curvature values (κ_1, κ_2) and directions (S_{k1}, S_{k2}) of a parametric surface S at a point p. Thus far we have derived II_S which contains curvature information, but it can be distorted by the metric of the parameterization. I_S describes this metric information. We must combine II_S and I_S to find the arc length curvature values.

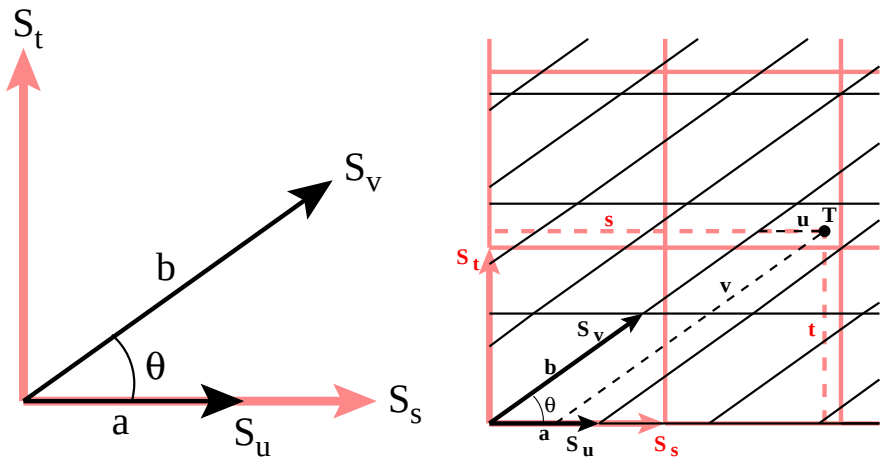
The given parameterization (u, v) defines a set of basis tangent vectors $\begin{bmatrix} S_u & S_v \end{bmatrix}$. We can construct an orthonormal basis $\begin{bmatrix} S_s & S_t \end{bmatrix}$ due to an arc length parameterization (s, t) where S_s and S_u are aligned, see Figure 2.3(a). A point T can be expressed in either coordinate system, see Figure 2.3(b).

$$T = \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} S_u \\ S_v \end{bmatrix} = \begin{bmatrix} \partial s & \partial t \end{bmatrix} \begin{bmatrix} S_s \\ S_t \end{bmatrix} \tag{2.31}$$

The coordinates $\begin{bmatrix} \partial s & \partial t \end{bmatrix}$ measure the length of T as opposed to the coordinates $\begin{bmatrix} \partial u & \partial v \end{bmatrix}$.

$$T \cdot T = \partial s^2 + \partial t^2 \neq \partial u^2 + \partial v^2 \tag{2.32}$$

We would like to work in the $\begin{bmatrix} S_s & S_t \end{bmatrix}$, so we must find the transformations to and from $\begin{bmatrix} S_u & S_v \end{bmatrix}$. The quantities a, b, and θ in Figure 2.3(a) are defined by $\begin{bmatrix} S_u & S_v \end{bmatrix}$.



(a) *Constructing an orthonormal basis on top of the given parametric basis*

(b) *A point T in (u, v) and (s, t) coordinates*

Figure 2.3: *Effects of the metric*

$$\begin{bmatrix} S_u \\ S_v \end{bmatrix} = \begin{bmatrix} a & 0 \\ b\cos\theta & b\sin\theta \end{bmatrix} \begin{bmatrix} S_s \\ S_t \end{bmatrix} \tag{2.33}$$

$$= A \begin{bmatrix} S_s \\ S_t \end{bmatrix} \tag{2.34}$$

$$\begin{bmatrix} S_s \\ S_t \end{bmatrix} = \frac{1}{ab\sin\theta} \begin{bmatrix} b\sin\theta & 0 \\ -b\cos\theta & a \end{bmatrix} \begin{bmatrix} S_u \\ S_v \end{bmatrix} \tag{2.35}$$

$$= A^{-1} \begin{bmatrix} S_u \\ S_v \end{bmatrix} \tag{2.36}$$

The first fundamental form I_S can be represented by these vector transformations.

$$I_S = \begin{bmatrix} S_u \\ S_v \end{bmatrix} \cdot \begin{bmatrix} S_u & S_v \end{bmatrix} \tag{2.37}$$

$$= A \begin{bmatrix} S_s \\ S_t \end{bmatrix} \cdot \begin{bmatrix} S_s & S_t \end{bmatrix} A^t \tag{2.38}$$

$$= A \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} A^t \tag{2.39}$$

$$= AA^t \tag{2.40}$$

$$= \begin{bmatrix} a^2 & ab\cos\theta \\ ab\cos\theta & b^2 \end{bmatrix} \tag{2.41}$$

We can also derive transformations between coordinates of the different sets of basis vectors by substituting Equations [2.34,2.36] respectively into Equation 2.31.

$$\begin{bmatrix} \partial s & \partial t \end{bmatrix} = \begin{bmatrix} \partial u & \partial v \end{bmatrix} A \tag{2.42}$$

$$\begin{bmatrix} \partial u & \partial v \end{bmatrix} = \begin{bmatrix} \partial s & \partial t \end{bmatrix} A^{-1} \tag{2.43}$$

To verify that this makes sense, consider $T \cdot T$ again and transform the coordinates using Equations [2.42,2.43].

$$T \cdot T = \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} S_u \\ S_v \end{bmatrix} \cdot \begin{bmatrix} S_u & S_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.44}$$

$$= \begin{bmatrix} \partial u & \partial v \end{bmatrix} I_S \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.45}$$

$$= \begin{bmatrix} \partial s & \partial t \end{bmatrix} A^{-1} I_S \left(A^{-1}\right)^t \begin{bmatrix} \partial s \\ \partial t \end{bmatrix} \tag{2.46}$$

$$= \begin{bmatrix} \partial s & \partial t \end{bmatrix} A^{-1} \left(AA^t\right) \left(A^{-1}\right)^t \begin{bmatrix} \partial s \\ \partial t \end{bmatrix} \tag{2.47}$$

$$= \begin{bmatrix} \partial s & \partial t \end{bmatrix} \begin{bmatrix} \partial s \\ \partial t \end{bmatrix} \tag{2.48}$$

$$= \partial s^2 + \partial t^2 \tag{2.49}$$

It is now possible to transform our given parameters to coordinates on an orthonormal basis where we can measure lengths. We can also transform coordinates on the orthonormal basis back to coordinates on our given parameterization.

2.5 Curvature

The curvature at a point p on the surface S in any tangential direction T is $-T \cdot N_T$.

$$-T \cdot N_T = \begin{bmatrix} \partial u & \partial v \end{bmatrix} \begin{bmatrix} S_u \\ S_v \end{bmatrix} \cdot \begin{bmatrix} -N_u & -N_v \end{bmatrix} \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.50}$$

$$= \begin{bmatrix} \partial u & \partial v \end{bmatrix} II_S \begin{bmatrix} \partial u \\ \partial v \end{bmatrix} \tag{2.51}$$

This function is parameterized by the given parameterization. We would like to study the function that sweeps a unit length tangent around p, so it is more convenient to use arc length coordinates. We must transform coordinates so that setting the arc length coordinates $\begin{bmatrix} \partial s & \partial t \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \end{bmatrix}$ sweeps out the curvature values.

$$-T \cdot N_T = \begin{bmatrix} \partial s & \partial t \end{bmatrix} A^{-1} II_S \left(A^{-1}\right)^t \begin{bmatrix} \partial s \\ \partial t \end{bmatrix} \tag{2.52}$$

$$= \begin{bmatrix} \partial s & \partial t \end{bmatrix} II_{\hat{S}} \begin{bmatrix} \partial s \\ \partial t \end{bmatrix} \tag{2.53}$$

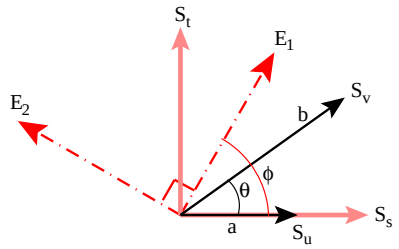


Figure 2.4: *The relationship between the principle parameter directions* (E_1, E_2)*, the given general parameter directions* (S_u, S_v)*, and the constructed orthonormal basis* (S_s, S_t)*in the tangent plane to* S *at* p

The curvature function will have maximum and minimum values that occur at directions that are orthogonal to each other, $\{E_1, E_2\}$ in Figure 2.4. The eigenvalues of $II_{\hat{S}}$ are the principle curvature values (κ_1, κ_2). The eigenvectors of $II_{\hat{S}}$ are the coordinates of the principle curvature directions in the arc length coordinates. These coordinates will not be directly useful because we only know the $\begin{bmatrix} S_u & S_v \end{bmatrix}$ vectors, so we would actually prefer the eigenvectors in coordinates on this basis.

$$II_{\hat{S}} = A^{-1} II_S \left(A^{-1}\right)^t \tag{2.54}$$

$$= \frac{1}{a^2 b^2 \sin^2\theta} \begin{bmatrix} b\sin\theta & 0 \\ -b\cos\theta & a \end{bmatrix} \begin{bmatrix} L & M \\ M & N \end{bmatrix} \begin{bmatrix} b\sin\theta & -b\cos\theta \\ 0 & a \end{bmatrix} \tag{2.55}$$

$$= \frac{1}{a^2 b^2 \sin^2\theta} \cdot \begin{bmatrix} b^2\sin^2\theta & b\sin\theta\left(-b\cos\theta L + aM\right) \\ b\sin\theta\left(-b\cos\theta L + aM\right) & b^2\cos^2\theta L - 2ab\cos\theta M + a^2 N \end{bmatrix} \tag{2.56}$$

To solve for the eigenvalues, we must form the characteristic equation, Equation 2.58 (see Appendix B.2.2). The principle curvature values are the eigenvalues.

$$0 = \left(a^2b^2 - (ab\cos\theta)^2\right)\lambda^2 - \left(b^2 L - 2ab\cos\theta M + a^2 N\right)\lambda + \left(LN - M^2\right) \tag{2.57}$$

$$= \left(EG - F^2\right)\lambda^2 - \left(GL - 2FM + EN\right)\lambda + \left(LN - M^2\right) \tag{2.58}$$

2.6 The Weingarten Operator

An alternate way of computing the principle curvature values and directions is by using the Weingarten Operator W, also known as the shape operator. W is the inverse of the first fundamental form multiplied by the second fundamental form.

$$W = I_S^{-1} II_S \tag{2.59}$$

I_S^{-1} is the inverse of the first fundamental form.

$$I_S^{-1} = \left(AA^t\right)^{-1} = \left(A^{-1}\right)^t A^{-1} \tag{2.60}$$

$$= \frac{1}{EG - F^2} \begin{bmatrix} G & -F \\ -F & E \end{bmatrix} \tag{2.61}$$

We will prove that W has the same eigenvalues as $II_{\hat{S}}$, i.e. the same principle curvature values. We will also prove that the eigenvectors of W are transformed versions of the eigenvectors of $II_{\hat{S}}$ to coordinates on the given basis $\begin{bmatrix} S_u & S_u \end{bmatrix}$.

$$II_{\hat{S}} = V\Lambda V^{-1} \tag{2.62}$$

$$II_S = AII_{\hat{S}}A^t \tag{2.63}$$

$$W = I_S^{-1} II_S \tag{2.64}$$

$$= \left(A^{-1}\right)^t A^{-1} II_S \tag{2.65}$$

$$= \left(A^{-1}\right)^t A^{-1} AII_{\hat{S}}A^t \tag{2.66}$$

$$= \left(A^{-1}\right)^t II_{\hat{S}}A^t \tag{2.67}$$

$$W = \left(A^{-1}\right)^t V\Lambda V^{-1}A^t \tag{2.68}$$

Equation 2.68 is the eigendecomposition of W. The diagonal matrix Λ has the eigenvalues of W on its diagonal. It is the same eigenvalue matrix as for $II_{\hat{S}}$, so the eigenvalues of W are the principle curvature values. $\left(A^{-1}\right)^t V = \left(V^{-1}A^t\right)^{-1}$, so the columns of $\left(A^{-1}\right)^t V$ are the eigenvectors of W. This is a transformed version of V, the eigenvectors of $II_{\hat{S}}$ over the arc length basis. The transformation A^{-1} transforms the arc length coordinates to coordinates over the given basis, see Equation 2.43.

$$V = \begin{bmatrix} s_1 & s_2 \\ t_1 & t_2 \end{bmatrix} \tag{2.69}$$

$$\left(A^{-1}\right)^t V = \left(A^{-1}\right)^t \begin{bmatrix} s_1 & s_2 \\ t_1 & t_2 \end{bmatrix} \tag{2.70}$$

$$= \begin{bmatrix} u_1 & u_2 \\ v_1 & v_2 \end{bmatrix} \tag{2.71}$$

To solve for the principle curvature values and directions using the given parameterization, we substitute Equations [2.26,2.61] into Equation 2.59 and simplify to derive W written as the coefficients of the first and second fundamental forms.

$$W = I_S^{-1} II_S \tag{2.72}$$

$$= \frac{1}{EG - F^2} \begin{bmatrix} G & -F \\ -F & E \end{bmatrix} \begin{bmatrix} L & M \\ M & N \end{bmatrix} \tag{2.73}$$

$$= \frac{1}{EG - F^2} \begin{bmatrix} GL - FM & GM - FN \\ EM - FL & EN - FM \end{bmatrix} \tag{2.74}$$

Then we use the representation of W in Equation 2.74 and compute the eigendecomposition. The eigenvalues are the principle curvatures, and the eigenvectors are the coordinate coefficients over the given parametric basis vectors.

$$\kappa_{1,2} = \lambda_{1,2} = \frac{GL - 2FM + EN}{2\left(EG - F^2\right)} \mp \frac{\sqrt{\left(GL - 2FM + EN\right)^2 - 4\left(EG - F^2\right)\left(LN - M^2\right)}}{2\left(EG - F^2\right)} \tag{2.75}$$

$$\kappa_M = \frac{\kappa_1 + \kappa_2}{2} = \frac{GL - 2FM + EN}{2\left(EG - F^2\right)} = \frac{Trace\left(W\right)}{2} \tag{2.76}$$

$$\kappa_G = \kappa_1 \kappa_2 = \frac{LN - M^2}{EG - F^2} = Det\left(W\right) \tag{2.77}$$

2.7 Minimizing Total Bending Energy

A minimal surface is a surface that minimizes the integral of an area weighted energy functional over the surface. A physical example of a minimal surface is a soap bubble film spanning a wire boundary loop which minimizes surface area, Equation 2.78.

$$E_{area} = \int_S \partial A \tag{2.78}$$

For the soap film, the pressure on either side of the surface must be equal for it to be at equilibrium, so the mean curvature $\kappa_M = 0$ as every point on the surface is a saddle point with equal and opposite min and max curvatures, $\kappa_1 = -\kappa_2$. We will show that a soap film also minimizes total

bending energy. Total bending energy is defined as the area weighted integral of the magnitudes of the minimum and maximum curvatures, Equation 2.79.

$$E_{bend} = \int_S \kappa_1^2 + \kappa_2^2 \partial A \tag{2.79}$$

The total curvature quantity $\kappa_1^2 + \kappa_2^2$ can be computed from the mean κ_M and Gaussian κ_G curvatures.

$$\kappa_1^2 + \kappa_2^2 = \left(\kappa_1^2 + 2\kappa_1\kappa_2 + \kappa_2^2\right) - 2\kappa_1\kappa_2 \tag{2.80}$$

$$= 4\left(\frac{\kappa_1^2 + 2\kappa_1\kappa_2 + \kappa_2^2}{4}\right) - 2\kappa_1\kappa_2 \tag{2.81}$$

$$= 4\left(\frac{\kappa_1 + \kappa_2}{2}\right)^2 - 2\kappa_1\kappa_2 \tag{2.82}$$

$$= 4\kappa_M^2 - 2\kappa_G \tag{2.83}$$

We will show that for S with a fixed topology, minimizing the bending energy is equivalent to minimizing the mean curvature energy.

$$E_{bend} = \int_S \kappa_1^2 + \kappa_2^2 \partial A \tag{2.84}$$

$$= \int_S 4\kappa_M^2 - 2\kappa_G \partial A \tag{2.85}$$

$$= 4\int_S \kappa_M^2 \partial A - 2\int_S \kappa_G \partial A \tag{2.86}$$

$$= 4E_{mean} - 2E_{Gauss} \tag{2.87}$$

Equation 2.87 shows that E_{bend} is proportional to the sum of E_{mean} and E_{Gauss}, so it is not at first obvious that minimizing E_{bend} is the same as minimizing E_{mean}. However, the Gauss-Bonnet theorem states that E_{Gauss} for a given topology is a constant reguardless of the geometric embedding of the surface, Section 2.7.1. Hence the change in E_{mean} is proportional to the change in E_{bend}, so minimizing E_{mean} is equivalent to minimizing E_{bend}. Recalling the soap film example where the mean curvature $\kappa_M = 0$ everywhere, this surface then minimizes bending energy.

2.7.1 The Gauss-Bonnet Theorem

The Gauss-Bonnet theorem states that the area weighted integral of Gaussian curvature E_{Gauss} is proportional to the Euler characteristic $\chi(S)$ of a manifold surface S. $\chi(S)$ is a topolog-

ical invariant of S, so E_{Gauss} is independent of the particular geometric embedding of the topology S. $\chi(S)$ is a simple function of the number of vertices V, edges E, and faces F for any valid tessellation of the manifold S, Equation 2.88.

$$\chi(S) = V - E + F \tag{2.88}$$

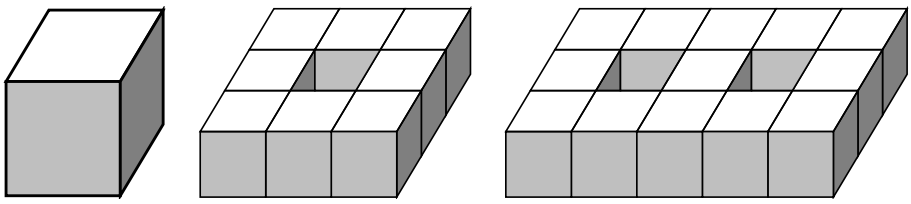
More simply, $\chi(S)$ is a function of the genus G of the manifold. The formulas for $\chi(S)$ of an orientable surface (a surface that can be embedded in $R3$ without self-intersections) Equation 2.89 and a nonorientable surface Equation 2.90 are slightly different, but both depend on the genus only.

$$\chi_{orientable}(S) = 2 - 2G \tag{2.89}$$

$$\chi_{nonorientable}(S) = 2 - G \tag{2.90}$$

The Gauss-Bonnet theorem states that E_{Gauss} is proportional $\chi(S)$, Equation 2.91, so all surfaces of a given genus G have the same E_{Gauss}.

$$E_{Gauss} = \int_S \kappa_G \partial A = 2\pi\chi(S) \tag{2.91}$$

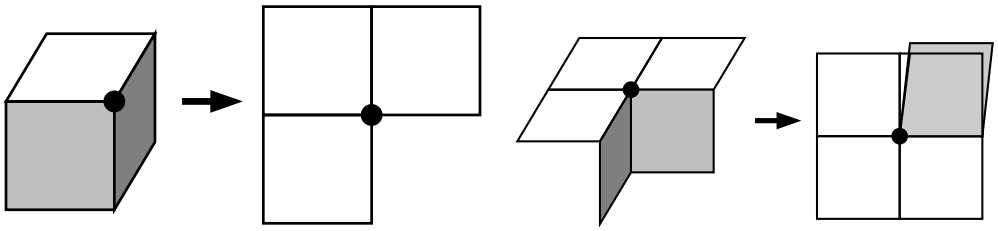


(a) *Genus 0* (b) *Genus 1* (c) *Genus 2*

Figure 2.5: *Orientable manifolds of different genus*

E_{Gauss} is the sum of the angle deficit at every point on the surface. This is easy to understand in the discrete case. Consider orientable polyhedral objects made up of squares only, Figure 2.5. $G = 0$ is a cube which has 8 valence $n = 3$ vertices on the corners, Figure 2.5(a). $G = 1$ is a torus which also has 8 valence $n = 3$ vertices on the corners plus an additional 8 valence

$n = 5$ saddle vertices around the hole, Figure 2.5(b). All other vertices in the torus are valence $n = 4$. With every additional genus handle, 8 new valence $n = 5$ vertices are added, Figure 2.5(c).



(a) *Valence* $n = 3$ (b) *Valence* $n = 5$

Figure 2.6: *Angle deficit created by flattening the local disc around a vertex. For* $n < 4$ *there is a deficit, and for* $n > 4$ *there is a surplus so the squares overlap.*

The angle deficit of a vertex is defined by its valence n, i.e. the number of squares it is incident on, Figure 2.6. The angle at the corner of a square is $\frac{\pi}{2}$. The angle deficit is $2\pi - n\frac{\pi}{2}$. For $n = 3$ the angle deficit is $\frac{\pi}{2}$, Figure 2.6(a). For $n = 4$ the angle deficit is 0. For $n = 5$ the angle deficit is $-\frac{\pi}{2}$, Figure 2.6(b). Equation 2.95 shows that the total angle deficit for the different genus objects in Figure 2.5 follow the Gauss-Bonnet theorem.

$$E_{Gauss} = 8Deficit\,(3) + 8GDeficit\,(5) \tag{2.92}$$
$$= 8\frac{\pi}{2} - 8G\frac{\pi}{2} \tag{2.93}$$
$$= 2\pi\,(2 - 2G) \tag{2.94}$$
$$= 2\pi\chi_{orientable}\,(S) \tag{2.95}$$

2.8 Conclusion

We have shown how to calculate the position, tangent vectors and normal, and principle curvature directions and values for a parametric surface with C^2 continuity. In Chapter 3, we show that subdivision surfaces are C^2 continuous except at a discrete set of extraordinary points. Chapters 4-8 derive the Jordan decompositions of the subdivision matrices for piecewise smooth Loop and Catmull-Clark subdivision surfaces, and present a new exact evaluation algorithm for arbitrary parameter locations over these general 2-manifold surfaces. Combining the results from this chapter with the evaluation algorithm, we have a full parameterization of the subdivsion surface with full

differential information up to curvature.

Chapter 3

Eigen-Analysis of Stationary Subdivision Schemes

3.1 Introduction

A subdivision manifold (curve or surface) is a smooth limit manifold defined by repeated averaging by a subdivision scheme over a control mesh of control vertices. Each step of subdivision applies a topological split followed by a set of averaging rules that define the vertex positions in the next generation. Most subdivision schemes are generalizations of box spline knot insertion schemes, and so they share the continuity behavior of these parametric functions in their regular areas. A box spline is constructed by convolving a box filter a specified number of times in a specified number of parametric directions. A convolution is an integral function, so each convolution pass adds a degree of differential continuity. For example, uniform bi-cubic B-splines are $N_{3,3}$ box splines, so they are constructed by convolving three times in both u and v parametric directions. The resulting surface is C^2 continuous in both parameters because the position, first, and second derivatives are continuous.

We will show that for stationary subdivision schemes, we can analyze the continuity behavior of the scheme with only knowing the matrix of stationary averaging rules. A stationary subdivision scheme, e.g. box splines, is where the subdivision mask matrix S of averaging rules is constant at every generation of subdivision. S is applied to a local vector of control points P^g at a generation g to generate the next generation of control points P^{g+1} around a given control point (Figure 3.1 and Equation 3.1). The local neighborhood P^g at any generation g is then the intial

neighborhood P^0 multiplied by g applications of S (Equation 3.2). The limit behavior of a stationary subdivision scheme can be analyzed by examining the structure of the eigen-decomposition of S (Equation 3.3). For a review of eigen-decompositions see Appendix B.2.4.

$$P^{g+1} = SP^g \tag{3.1}$$

$$P^g = S^g P^0 \tag{3.2}$$

$$S = V\Lambda V^{-1} \tag{3.3}$$

$$S^g = V\Lambda^g V^{-1} \tag{3.4}$$

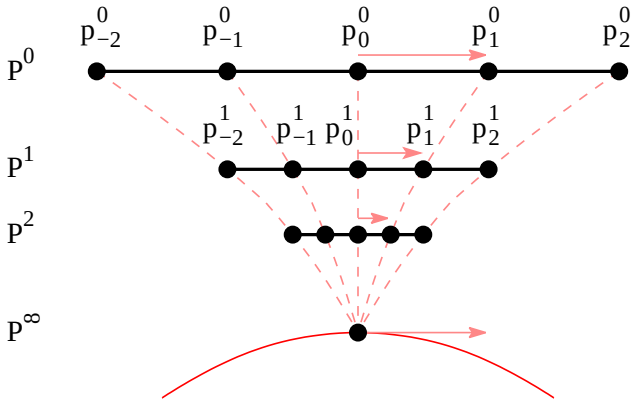


Figure 3.1: *The local neighborhood of points P^g in the subdivision mask S at different generations $\{g : 0 \rightarrow \infty\}$ that converge to the limit position. The arrows show the finite difference between points in the parameter direction u, and the limit of that difference is the first derivative.*

3.2 Subdivision Limit Position

Each application of S shrinks the local neighborhood P until after an infinite number of subdivisions, the limit surface is reached (Figure 3.1 and Equation 3.5). If the subdivision manifold is C^0 continuous then all the vertices in the local neighborhood vector P^∞ will shrink to the same point.

$$P^\infty = \lim_{g\rightarrow\infty} P^g = S^\infty P^0 \tag{3.5}$$

If each of the rows of S, i.e. the averaging rule for each vertex in the subdivision mask, sum to one with the weights being positive or negative, then S will have one eigenvector consisting

of all ones $v_0 = \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}^t$ and at least one eigenvalue equal to one $\lambda_0 = 1$. Proof: Assume $v_0 = \begin{bmatrix} 1 & 1 & ... & 1 \end{bmatrix}^t$, then the result of multiplying Sv_0 is a vector whose elements are each the sum of the corresponding row of S which we defined to always be one. Hence, $Sv_0 = v_0$ and v_0 is an eigenvector of S. Furthermore, the scalar or eigenvalue λ_0 must be one. $Sv_0 = \lambda_0 v_0 = 1v_0 = v_0$.

If $1 = \lambda_0 > \lambda_i \{i : 1, n\} \geq 0$, then S is C^0 continuous. Consider the eigen-decomposition of S with v_0 and λ_0 substituted in (Equation 3.8).

$$S = V \Lambda V^{-1} \tag{3.6}$$

$$= \begin{bmatrix} v_0 & v_1 & ... & v_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \lambda_0 & 0 & ... & 0 \\ 0 & \lambda_1 & & ... \\ ... & & ... & \\ 0 & ... & & \lambda_n \end{bmatrix} \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_1^{-1} & \rightarrow \\ ... & \\ v_n^{-1} & \rightarrow \end{bmatrix} \tag{3.7}$$

$$= \begin{bmatrix} 1 & v_1 & ... & v_n \\ 1 & \downarrow & & \downarrow \\ ... & & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & \lambda_1 & & ... \\ ... & & ... & \\ 0 & ... & & \lambda_n \end{bmatrix} \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_1^{-1} & \rightarrow \\ ... & \\ v_n^{-1} & \rightarrow \end{bmatrix} \tag{3.8}$$

To find the limit neighborhood P^∞, we must compute S^∞ (Equation 3.5). Fortunately, the eigen-decomposition of S reduces this to computing Λ^∞ (Equation 3.9). Raising the diagonal matrix Λ to a power is the same as raising its diagonal elements to that power, and since $1 > \lambda_i \{i : 1, n\} \geq 0$ then $\lambda_i^\infty = 0$ (Equation 3.11). Mulitplying $V\Lambda^\infty$ maintains the first column of V, v_0 which is all ones, and the rest of the matrix becomes all zeroes (Equation 3.12). The final multiplication copies the top row of V^{-1}, v_0^{-1}, to all the other rows (Equation 3.13).

$$S^{\infty} = V\Lambda^{\infty}V^{-1} \tag{3.9}$$

$$= \begin{bmatrix} 1 & v_1 & ... & v_n \\ 1 & \downarrow & & \downarrow \\ ... & & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} 1^{\infty} & 0 & ... & 0 \\ 0 & \lambda_1^{\infty} & & ... \\ ... & & ... & \\ 0 & ... & & \lambda_n^{\infty} \end{bmatrix} \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_1^{-1} & \rightarrow \\ ... & \\ v_n^{-1} & \rightarrow \end{bmatrix} \tag{3.10}$$

$$= \begin{bmatrix} 1 & v_1 & ... & v_n \\ 1 & \downarrow & & \downarrow \\ ... & & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 0 & & ... \\ ... & & ... & \\ 0 & ... & & 0 \end{bmatrix} \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_1^{-1} & \rightarrow \\ ... & \\ v_n^{-1} & \rightarrow \end{bmatrix} \tag{3.11}$$

$$= \begin{bmatrix} 1 & 0 & ... & 0 \\ 1 & 0 & ... & 0 \\ ... & ... & ... & ... \\ 1 & 0 & ... & 0 \end{bmatrix} \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_1^{-1} & \rightarrow \\ ... & \\ v_n^{-1} & \rightarrow \end{bmatrix} \tag{3.12}$$

$$= \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_0^{-1} & \rightarrow \\ ... & \\ v_0^{-1} & \rightarrow \end{bmatrix} \tag{3.13}$$

Because all of the rows of S^{∞} are identical, all of the vertices of the limit local neighborhood P^{∞} converge to the same position and S is thus C^0 continuous (Equation 3.14). The limit position weight mask is the row of inverse eigenvector matrix V^{-1} associated with the eigenvalue $\lambda_0 = 1$.

$$P^{\infty} = \begin{bmatrix} v_0^{-1} & \rightarrow \\ v_0^{-1} & \rightarrow \\ ... & \\ v_0^{-1} & \rightarrow \end{bmatrix} P^0 \tag{3.14}$$

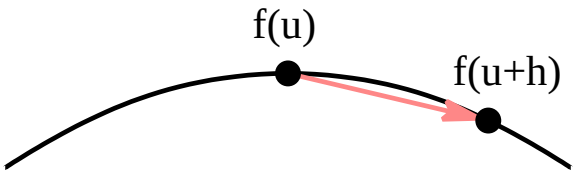


Figure 3.2: *The derivative of a function* $f\left(u\right)$ *is* $\lim_{h\to 0}\frac{f(u+h)-f(u)}{h}$

3.3 Subdivision Derivatives

It is also possible to compute derivatives of subdivision manifolds directly from the subdivision mask S. Recall the definition of the derivative of a function $f\left(u\right)$ in Equation 3.15.

$$\frac{\partial f\left(u\right)}{\partial u}=\lim_{h\to 0}\frac{f\left(u+h\right)-f\left(u\right)}{h} \tag{3.15}$$

Figure 3.2 shows that as $h\to 0$ the point $f\left(u+h\right)$ approaches $f\left(u\right)$ within the manifold until it is computing the difference vector of two points that coincide. Analogously in Figure 3.1, two control points p_i^g and p_{i+1}^g along a parameter direction u in the local neighborhood approach each other and the limit position as $g\to\infty$. Hence the limit of the finite difference between these control points normalized by the parametric distance in u is the derivative of the subdivision manifold with respect to u, but the quantity in Equation 3.16 will always be zero. Only the components scaled by the eigenvalue $\lambda_0=1$ will exist as $g\to\infty$, and that value is the limit position for all points p_i^∞ so it will cancel due to the subtraction.

$$\lim_{g\to\infty}\frac{p_{i+1}^g-p_i^g}{\|p_{i+1}^0-p_i^0\|} \tag{3.16}$$

If we scale this quantity by $\frac{1}{\lambda_1^g}$ for the next highest eigenvalue λ_1 then the components scaled by λ_1 will not vanish, components scaled by $\lambda_i<\lambda_1$ will vanish, and componets scaled by $\lambda_0=1$ that would diverge will cancel due to the subtraction. Equation 3.17 defines the first derivative of the subdivision manifold for a parameter direction u.

$$\frac{\partial S}{\partial u}=\lim_{g\to\infty}\frac{1}{\lambda_1^g}\frac{p_{i+1}^g-p_i^g}{\|p_{i+1}^0-p_i^0\|} \tag{3.17}$$

The limit of the finite difference between different pairs of points along the same parameter direction u will be the same if the subdivision manifold is C^1 continuous. By similar reasoning, a second finite difference between these quantities scaled by $\frac{1}{\lambda_2^g}$ defines the second derivative with respect to u (Equation 3.18). Further derivatives are calculated by performing finite differences of the previous

derivate and scaling by the appropriate eigenvalue (Equation 3.19).

$$\frac{\partial^2 S}{\partial u^2} = \lim_{g\to\infty} \frac{1}{\lambda_2^g} \frac{\left(p_{i+1}^g - p_i^g\right) - \left(p_i^g - p_{i-1}^g\right)}{\|p_{i+1}^0 - p_i^0\|^2} \tag{3.18}$$

$$\frac{\partial^3 S}{\partial u^3} = \lim_{g\to\infty} \frac{1}{\lambda_3^g} \frac{\left(p_{i+2}^g - 2p_{i+1}^g + p_i^g\right) - \left(p_{i+1}^g - 2p_i^g + p_{i-1}^g\right)}{\|p_{i+1}^0 - p_i^0\|^3} \tag{3.19}$$

3.4 Example: Uniform Cubic B-Spline Curves

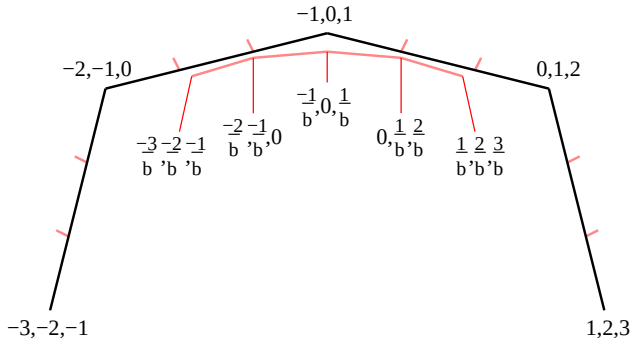


Figure 3.3: *Set of points P^0 in the subdivision mask S for a parameter step of $\frac{1}{b}$ for a uniform cubic B-spline, and the next generation P^1*

As a concrete example, we will analyze the subdivision curve that is the same as a uniform cubic B-spline curve. The techniques applied to curves or surfaces are analogous, but the examples are much smaller and easier to understand for curves.

Consider the case of subdividing near a control vertex of a uniform cubic B-spline curve in parameter steps of $\frac{1}{b}$ for an integer $b > 1$. We will consider a subdivision mask S over a local neighborhood of five vertices (Figure 3.3). S multiplies a vector of input blossom control vertices P^g at generation g and produces a vector of output blossom control vertices P^{g+1} at generation $g+1$. Equations 3.21 and 3.22 show the blossoming knot labels for the control points for generations P^g and P^{g+1}, respectively. These knot labels are used in Figure 3.3 to identify the blossom vertices.

$$P^{g+1} = SP^g \tag{3.20}$$

$$P^g = \begin{bmatrix} V_{-3,-2,-1} & V_{-2,-1,0} & V_{-1,0,1} & V_{0,1,2} & V_{1,2,3} \end{bmatrix}^t \tag{3.21}$$

$$P^{g+1} = \begin{bmatrix} V_{-\frac{3}{b},-\frac{2}{b},-\frac{1}{b}} & V_{-\frac{2}{b},-\frac{1}{b},0} & V_{-\frac{1}{b},0,\frac{1}{b}} & V_{0,\frac{1}{b},\frac{2}{b}} & V_{\frac{1}{b},\frac{2}{b},\frac{3}{b}} \end{bmatrix}^t \tag{3.22}$$

The rows of S are the subdivision averaging rules for each of the control vertices. The weights can be derived from the blossom averaging rules. Notice that there is a bilateral symmetry

around $V_{-1,0,1}$ in the parameter space, so we only need to derive the weights for $V_{-\frac{1}{b},0,\frac{1}{b}}$, $V_{0,\frac{1}{b},\frac{2}{b}}$, and $V_{\frac{1}{b},\frac{2}{b},\frac{3}{b}}$. Figure 3.4 shows the set of blossom points that must be calculated to derive the weights. Equations 3.23-3.29 show the interpolation along the individual blossom number lines.

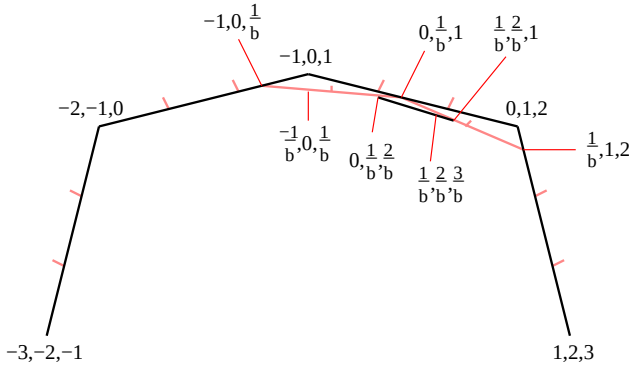


Figure 3.4: *Blossoming points used to derive the subdivisions rules for a parameter step of* $\frac{1}{b}$ *for a uniform cubic B-spline*

$$V_{-1,0,\frac{1}{b}} = \frac{3b-(2b+1)}{3b}V_{-2,-1,0} + \frac{2b+1}{3b}V_{-1,0,1} \tag{3.23}$$

$$V_{0,\frac{1}{b},1} = \frac{3b-(b+1)}{3b}V_{-1,0,1} + \frac{b+1}{3b}V_{0,1,2} \tag{3.24}$$

$$V_{\frac{1}{b},1,2} = \frac{3b-1}{3b}V_{0,1,2} + \frac{1}{3b}V_{1,2,3} \tag{3.25}$$

$$V_{-\frac{1}{b}0\frac{1}{b}} = \frac{2b-(b-1)}{2b}V_{-1,0,\frac{1}{b}} + \frac{b-1}{2b}V_{0,\frac{1}{b},1} \tag{3.26}$$

$$V_{0,\frac{1}{b},\frac{2}{b}} = \frac{2b-(b+2)}{2b}V_{-1,0,\frac{1}{b}} + \frac{b+2}{2b}V_{0,\frac{1}{b},1} \tag{3.27}$$

$$V_{\frac{1}{b},\frac{2}{b},1} = \frac{2b-2}{2b}V_{0,\frac{1}{b},1} + \frac{2}{2b}V_{\frac{1}{b},1,2} \tag{3.28}$$

$$V_{\frac{1}{b},\frac{2}{b},\frac{3}{b}} = \frac{b-3}{b}V_{0,\frac{1}{b},\frac{2}{b}} + \frac{3}{b}V_{\frac{1}{b},\frac{2}{b},1} \tag{3.29}$$

In Equations 3.30, 3.31, and3.32, the formulas for $V_{-\frac{1}{b},0,\frac{1}{b}}$, $V_{0,\frac{1}{b},\frac{2}{b}}$, and $V_{\frac{1}{b},\frac{2}{b},\frac{3}{b}}$ are expanded to be weighted sums of the original control vertices.

$$V_{-\frac{1}{b},0,\frac{1}{b}} = \frac{b^2-1}{6b^2}V_{-2,-1,0} + \frac{2(2b^2+1)}{6b^2}V_{-1,0,1} + \frac{b^2-1}{6b^2}V_{0,1,2} \tag{3.30}$$

$$V_{0,\frac{1}{b},\frac{2}{b}} = \frac{b^2-3b+2}{6b^2}V_{-2,-1,0} + \frac{4(b^2-1)}{6b^2}V_{-1,0,1} + \frac{b^2+3b+2}{6b^2}V_{0,1,2} \tag{3.31}$$

$$V_{\frac{1}{b},\frac{2}{b},\frac{3}{b}} = \frac{(b-3)(b-2)(b-1)}{6b^3}V_{-2,-1,0} + \frac{4b^3-22b+18}{6b^3}V_{-1,0,1} + \frac{b^3+6b^2+11b-18}{6b^3}V_{0,1,2} + \frac{6}{6b^3}V_{1,2,3} \tag{3.32}$$

Using the bilateral parametric symmetry, we can then write down the subdivision matrix S (Equation 3.33).

$$S = \frac{1}{6b^3}\begin{bmatrix} 6 & b^3+6b^2+11b-18 & 4b^3-22b+18 & (b-3)(b-2)(b-1) & 0 \\ 0 & b\left(b^2+3b+2\right) & 4b\left(b^2-1\right) & b\left(b^2-3b+2\right) & 0 \\ 0 & b\left(b^2-1\right) & 2b\left(2b^2+1\right) & b\left(b^2-1\right) & 0 \\ 0 & b\left(b^2-3b+2\right) & 4b\left(b^2-1\right) & b\left(b^2+3b+2\right) & 0 \\ 0 & (b-3)(b-2)(b-1) & 4b^3-22b+18 & b^3+6b^2+11b-18 & 6 \end{bmatrix} \tag{3.33}$$

To verify our formula for S, we let $b = 2$ and get the standard midpoint subdivision mask (Equation 3.34).

$$S_{b=2} = \frac{1}{8}\begin{bmatrix} 1 & 6 & 1 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 1 & 6 & 1 \end{bmatrix} \tag{3.34}$$

3.4.1 Eigen-Decomposition of S

To study the limit behavior of the curve, we need to consider $P^g = S^g P^0$ as $g \to \infty$. The most efficient way to raise a stationary square matrix S to a high integer power is by diagonalizing it into its eigen-decomposition (Equation 3.35).

$$\begin{aligned} S =& V\Lambda V^{-1} \\ =& \begin{bmatrix} 1 & -1 & 1 & 1 & 0 \\ 1 & -\frac{1}{2} & \frac{2}{11} & 0 & 0 \\ 1 & 0 & -\frac{1}{11} & 0 & 0 \\ 1 & \frac{1}{2} & \frac{2}{11} & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{b} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{b^2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{b^3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{b^3} \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & \frac{11}{6} & -\frac{22}{6} & \frac{11}{6} & 0 \\ 1 & -3 & 3 & -1 & 0 \\ 0 & -1 & 3 & -3 & 1 \end{bmatrix} \end{aligned} \tag{3.35}$$

S in Equation 3.33 is parameterized by the number of steps b per parametric unit. Note that only the eigenvalues $\{\lambda_0, \lambda_1, \lambda_2, \lambda_3, \lambda_3\} = \left\{1, \frac{1}{b}, \frac{1}{b^2}, \frac{1}{b^3}, \frac{1}{b^3}\right\}$ of the matrix Λ in Equation 3.35 depend on b. The eigenvectors, columns of matrix V, are constants that are defined by uniform cubic B-splines curves independent of our choice of equal parametric step size.

The magnitude of the eigenvalues determine the speed at which the subdivision process converges to its limit position which is parameterized by our choice of b. For standard midpoint subdivision, $b = 2$ because we are spliting a parametric edge at the midpoint into two sub-edges. For $b = 2$, $\lambda_{b=2} = \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right\}$. For $b = 3$, we are inserting two new points along each parametric edge and creating three sub-edges. There is more refinement at each step and faster convergence which is reflected by the set of smaller eigenvalues $\lambda_{b=3} = \left\{1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}, \frac{1}{27}\right\}$. For $b = 4$, we are inserting three new points along each parametric edge and creating four sub-edges. The eigenvalues are $\lambda_{b=4} = \left\{1, \frac{1}{4}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}\right\}$. Note that these eigenvalues are the squares of the midpoint subdivision eigenvalues, $\lambda_{b=4} = \lambda_{b=2}^2$. In fact, setting $b = 4$ is equivalent to doing two steps of $b = 2$, (Equations 3.36-3.42).

$$S_{b=2}^2 = S_{b=2} S_{b=2} \tag{3.36}$$
$$= \left(V \Lambda_{b=2} V^{-1}\right) \left(V \Lambda_{b=2} V^{-1}\right) \tag{3.37}$$
$$= V \Lambda_{b=2} \left(V^{-1} V\right) \Lambda_{b=2} V^{-1} \tag{3.38}$$
$$= V \Lambda_{b=2} \Lambda_{b=2} V^{-1} \tag{3.39}$$
$$= V \Lambda_{b=2}^2 V^{-1} \tag{3.40}$$
$$= V \Lambda_{b=4} V^{-1} \tag{3.41}$$
$$= S_{b=4} \tag{3.42}$$

3.4.2 Limit Position

The eigen-structure of S for the B-spline case shown in Equation 3.35 meets all the requirements described in Section 3.2, so we can use Equation 3.14 to calculate the limit position $V_{0,0,0}$ (Equation 3.45). v_0^{-1} is the top row of V^{-1} from Equation 3.35.

$$V_{0,0,0} = v_0^{-1} \cdot P^0 \tag{3.43}$$
$$= \begin{bmatrix} 0 & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & 0 \end{bmatrix} \begin{bmatrix} V_{-3,-2,-1} \\ V_{-2,-1,0} \\ V_{-1,0,1} \\ V_{0,1,2} \\ V_{1,2,3} \end{bmatrix} \tag{3.44}$$
$$= \frac{1}{6} V_{-2,-1,0} + \frac{4}{6} V_{-1,0,1} + \frac{1}{6} V_{0,1,2} \tag{3.45}$$

In preparation to take derivatives, we will also repeat the steps of Section 3.2 for the uniform cubic B-spline example. Because we know the eigenvector matrix V and its inverse V^{-1}, we can remultiply the eigen-decomposition matrices to form S^g parameterized by its eigenvalues (Equation 3.49). Note that for any choice of b the set of eigenvalues is of the form $\{1, \lambda_1, \lambda_2, \lambda_3, \lambda_3\}$ where $\lambda_i = \frac{1}{b^i}$.

$$S = V \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & 0 & \lambda_3 \end{bmatrix} V^{-1} \tag{3.46}$$

$$S^g = V \Lambda^g V^{-1} \tag{3.47}$$

$$= V \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \lambda_1^g & 0 & 0 & 0 \\ 0 & 0 & \lambda_2^g & 0 & 0 \\ 0 & 0 & 0 & \lambda_3^g & 0 \\ 0 & 0 & 0 & 0 & \lambda_3^g \end{bmatrix} V^{-1} \tag{3.48}$$

$$S^g = \frac{1}{6} \begin{bmatrix} 6\lambda_3^g & 1 + 6\lambda_1^g + 11\lambda_2^g - 18\lambda_3^g & 4 - 22\lambda_2^g + 18\lambda_3^g & 1 - 6\lambda_1^g + 11\lambda_2^g - 6\lambda_3^g & 0 \\ 0 & 1 + 3\lambda_1^g + 2\lambda_2^g & 4 - 4\lambda_2^g & 1 - 3\lambda_1^g + 2\lambda_2^g & 0 \\ 0 & 1 - \lambda_2^g & 4 + 2\lambda_2^g & 1 - \lambda_2^g & 0 \\ 0 & 1 - 3\lambda_1^g + 2\lambda_2^g & 4 - 4\lambda_2^g & 1 + 3\lambda_1^g + 2\lambda_2^g & 0 \\ 0 & 1 - 6\lambda_1^g + 11\lambda_2^g - 6\lambda_3^g & 4 - 22\lambda_2^g + 18\lambda_3^g & 1 + 6\lambda_1^g + 11\lambda_2^g - 18\lambda_3^g & 6\lambda_3^g \end{bmatrix} \tag{3.49}$$

Note that the entries of S^g from Equation 3.49 are weighted sums of λ_i^g. Also note that the constant terms of each row are identical, so that when we let $g \to \infty$ we get the matrix where all of the rows are v_0^{-1}, Equation 3.50. And as expected, we derive the formula for the limit position $V_{0,0,0}$, Equation 3.52.

$$S^{\infty} = \frac{1}{6}\begin{bmatrix} 0 & 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 & 0 \end{bmatrix} \tag{3.50}$$

$$P^{\infty} = S^{\infty} P^{0} \tag{3.51}$$

$$V_{0,0,0} = \frac{1}{6} V_{-2,-1,0} + \frac{4}{6} V_{-1,0,1} + \frac{1}{6} V_{0,1,2} \tag{3.52}$$

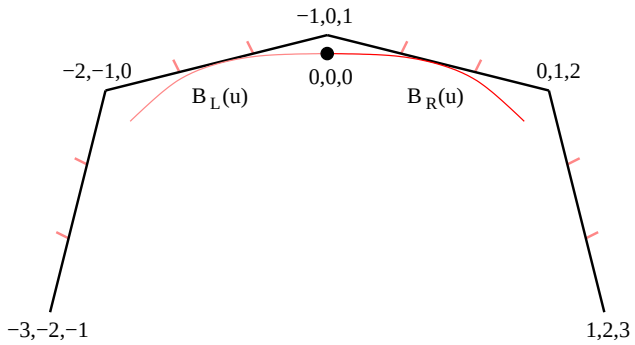


Figure 3.5: *Left and right B-spline curve segments* $B_L(u)$ *and* $B_R(u)$ *that meet at the subdivision limit vertex* $V_{0,0,0}$

The limit position of our subdivision process $V_{0,0,0}$ for the uniform cubic B-spline should be equal to the left B-spline segment $B_L(u)$ evaluated at $u = 1$ and also to the right B-spline segment $B_R(u)$ evaluated at $u = 0$ (Figure 3.5).

$$B_L(u) = \tfrac{(1-u)^3}{6} V_{-3,-2,-1} + \tfrac{3u^3-6u^2+4}{6} V_{-2,-1,0} + \tfrac{-3u^3+3u^2+3u+1}{6} V_{-1,0,1} + \tfrac{u^3}{6} V_{0,1,2} \tag{3.53}$$

$$B_R(u) = \tfrac{(1-u)^3}{6} V_{-2,-1,0} + \tfrac{3u^3-6u^2+4}{6} V_{-1,0,1} + \tfrac{-3u^3+3u^2+3u+1}{6} V_{0,1,2} + \tfrac{u^3}{6} V_{1,2,3} \tag{3.54}$$

$$V_{0,0,0} = B_L(1) = B_R(0) = \frac{1}{6} V_{-2,-1,0} + \frac{4}{6} V_{-1,0,1} + \frac{1}{6} V_{0,1,2} \tag{3.55}$$

Equations 3.45 and 3.55 show that the eigen-analysis of the subdivision matrix S matches the behavior of the known B-spline polynomial.

3.4.3 First Derivative

The limit of the finite difference between adjacent vertices in the local neighborhood will be the same if the subdivision process is C^1 continuous. Differencing p^g_{i+1} and p^g_i is the same as

differencing the corresponding rows $i+1$ and i of S^g and then applying that new matrix to P^0.

$$D_u = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.56}$$

$$\frac{\partial S^g}{\partial u} = D_u S^g \tag{3.57}$$

$$= \frac{1}{2} \begin{bmatrix} -2\lambda_3^g & -\lambda_1^g - 3\lambda_2^g + 6\lambda_3^g & 6\lambda_2^g - 6\lambda_3^g & \lambda_1^g - 3\lambda_2^g + 2\lambda_3^g & 0 \\ 0 & -\lambda_1^g - \lambda_2^g & 2\lambda_2^g & \lambda_1^g - \lambda_2^g & 0 \\ 0 & -\lambda_1^g + \lambda_2^g & -2\lambda_2^g & \lambda_1^g + \lambda_2^g & 0 \\ 0 & -\lambda_1^g + 3\lambda_2^g - 2\lambda_3^g & -6\lambda_2^g + 6\lambda_3^g & \lambda_1^g + 3\lambda_2^g - 6\lambda_3^g & 2\lambda_3^g \end{bmatrix} \tag{3.58}$$

By scaling Equation 3.58 by $\frac{1}{\lambda_1^g}$ and letting $g \to \infty$, only the terms with λ_1 remain. They are the same for every row (Equation 3.59), so the subdivision process is C^1.

$$\frac{\partial S^\infty}{\partial u} = \lim_{g \to \infty} \frac{1}{\lambda_1^g} \frac{\partial S^g}{\partial u} = \frac{1}{2} \begin{bmatrix} 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 \end{bmatrix} \tag{3.59}$$

$$\frac{\partial P^\infty}{\partial u} = \frac{\partial S^\infty}{\partial u} P^0 \tag{3.60}$$

$$\frac{\partial V_{0,0,0}}{\partial u} = -\frac{1}{2} V_{-2,-1,0} + \frac{1}{2} V_{0,1,2} \tag{3.61}$$

The first derivative of our subdivision process for the uniform cubic B-spline should be equal to that of the left B-spline segment $\frac{\partial B_L(u)}{\partial u}$ evaluated at $u = 1$ and also to that of the right B-spline segment $\frac{\partial B_R(u)}{\partial u}$ evaluated at $u = 0$.

$$\frac{\partial B_L(u)}{\partial u} = -\frac{(1-u)^2}{2} V_{-3,-2,-1} + \frac{3u^2 - 4u}{2} V_{-2,-1,0} + \frac{-3u^2 + 2u + 1}{2} V_{-1,0,1} + \frac{u^2}{2} V_{0,1,2} \tag{3.62}$$

$$\frac{\partial B_R(u)}{\partial u} = -\frac{(1-u)^2}{2} V_{-2,-1,0} + \frac{3u^2 - 4u}{2} V_{-1,0,1} + \frac{-3u^2 + 2u + 1}{2} V_{0,1,2} + \frac{u^2}{2} V_{1,2,3} \tag{3.63}$$

$$\frac{\partial B_L(1)}{\partial u} = \frac{\partial B_R(0)}{\partial u} = -\frac{1}{2} V_{-2,-1,0} + \frac{1}{2} V_{0,1,2} \tag{3.64}$$

Equations 3.61, and 3.64 show that the eigen-analysis of the subdivision matrix S matches the behavior of the known B-spline polynomial.

3.4.4 Second Derivative

The same finite differencing can be done to the first derivative matrix to compute second derivatives to see if the process is C^2 continuous.

$$D_{uu} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.65}$$

$$\frac{\partial^2 S^g}{\partial u^2} = D_{uu}\frac{\partial S^g}{\partial u} \tag{3.66}$$

$$= \begin{bmatrix} \lambda_3^g & \lambda_2^g - 3\lambda_3^g & -2\lambda_2^g + 3\lambda_3^g & \lambda_2^g - \lambda_3^g & 0 \\ 0 & \lambda_2^g & -2\lambda_2^g & \lambda_2^g & 0 \\ 0 & \lambda_2^g - \lambda_3^g & -2\lambda_2^g + 3\lambda_3^g & \lambda_2^g - 3\lambda_3^g & \lambda_3^g \end{bmatrix} \tag{3.67}$$

By scaling Equation 3.67 by $\frac{1}{\lambda_2^g}$ and letting $g \to \infty$, only the terms with λ_2 remain. They are the same for every row (Equation 3.68), so the subdivision process is C^2.

$$\frac{\partial^2 S^\infty}{\partial u^2} = \lim_{g\to\infty} \frac{1}{\lambda_2^g}\frac{\partial^2 S^g}{\partial u^2} = \begin{bmatrix} 0 & 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 & 0 \end{bmatrix} \tag{3.68}$$

$$\frac{\partial^2 P^\infty}{\partial u^2} = \frac{\partial^2 S^\infty}{\partial u^2} P^0 \tag{3.69}$$

$$\frac{\partial^2 V_{0,0,0}}{\partial u^2} = V_{-2,-1,0} - 2V_{-1,0,1} + V_{0,1,2} \tag{3.70}$$

The second derivative of our subdivision process for the uniform cubic B-spline should be equal to that of the left B-spline segment $\frac{\partial^2 B_L(u)}{\partial u^2}$ evaluated at $u = 1$ and also to that of the right B-spline segment $\frac{\partial^2 B_R(u)}{\partial u^2}$ evaluated at $u = 0$.

$$\frac{\partial^2 B_L(u)}{\partial u^2} = (1-u)V_{-3,-2,-1} + (3u-2)V_{-2,-1,0} + (-3u+1)V_{-1,0,1} + uV_{0,1,2} \tag{3.71}$$

$$\frac{\partial^2 B_R(u)}{\partial u^2} = (1-u)V_{-2,-1,0} + (3u-2)V_{-1,0,1} + (-3u+1)V_{0,1,2} + uV_{1,2,3} \tag{3.72}$$

$$\frac{\partial^2 B_L(1)}{\partial u^2} = \frac{\partial^2 B_R(0)}{\partial u^2} = V_{-2,-1,0} - 2V_{-1,0,1} + V_{0,1,2} \tag{3.73}$$

Equations 3.70 and 3.73 show that the eigen-analysis of the subdivision matrix S matches the behavior of the known B-spline polynomial.

3.4.5 Third Derivative

We can iterate again to check if the subdivision process is C^3 continuous.

$$D_{uuu} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \tag{3.74}$$

$$\frac{\partial^3 S^g}{\partial u^3} = D_{uuu} \frac{\partial^2 S^g}{\partial u^2} \tag{3.75}$$

$$= \begin{bmatrix} -\lambda_3^g & 3\lambda_3^g & -3\lambda_3^g & \lambda_3^g & 0 \\ 0 & -\lambda_3^g & 3\lambda_3^g & -3\lambda_3^g & \lambda_3^g \end{bmatrix} \tag{3.76}$$

We scale Equation 3.76 by $\frac{1}{\lambda_3^g}$ and let $g \to \infty$. The rows are not the same (Equation 3.77), so the subdivision process is not C^3.

$$\frac{\partial^3 S^\infty}{\partial u^3} = \lim_{g \to \infty} \frac{1}{\lambda_3^g} \frac{\partial^3 S^g}{\partial u^3} = \begin{bmatrix} -1 & 3 & -3 & 1 & 0 \\ 0 & -1 & 3 & -3 & 1 \end{bmatrix} \tag{3.77}$$

$$\frac{\partial^3 P^\infty}{\partial u^3} = \frac{\partial^3 S^\infty}{\partial u^3} P^0 \tag{3.78}$$

$$= \begin{bmatrix} -V_{-3,-2,-1} + 3V_{-2,-1,0} - 3V_{-1,0,1} + V_{0,1,2} \\ -V_{-2,-1,0} + 3V_{-1,0,1} - 3V_{0,1,2} + V_{1,2,3} \end{bmatrix} \tag{3.79}$$

The third derivative of the uniform cubic B-spline is discontinuous at the junction of the left and right B-spline segments, $\frac{\partial^3 B_L(1)}{\partial u^3} \neq \frac{\partial^3 B_R(0)}{\partial u^3}$. However, the formulas for the left and right third derivatives computed by eigen-analysis and with the polynomials are the same.

$$\frac{\partial^3 B_L(u)}{\partial u^3} = -V_{-3,-2,-1} + 3V_{-2,-1,0} - 3V_{-1,0,1} + V_{0,1,2} \tag{3.80}$$

$$\frac{\partial^3 B_R(u)}{\partial u^3} = -V_{-2,-1,0} + 3V_{-1,0,1} - 3V_{0,1,2} + V_{1,2,3} \tag{3.81}$$

$$\frac{\partial^3 B_L(1)}{\partial u^3} \neq \frac{\partial^3 B_R(0)}{\partial u^3} \tag{3.82}$$

3.5 Subdivision Surfaces

We now have a general tool for analyzing the continuity behavior of any subdivision mask matrix S. The same techniques work for subdivision surfaces as well. The C^0 limit position analysis is exactly the same as that for curves, so the averaging rules must sum to one and $1 = \mu_0 = \lambda_0$ and v_0 is all ones.

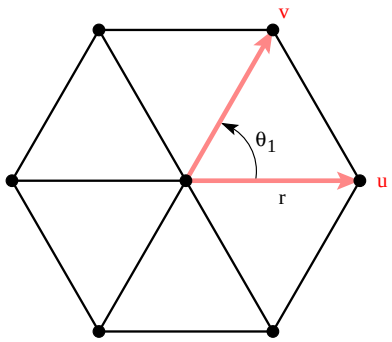


Figure 3.6: *Two parametric directions chosen for a subdivision surface*

The derivatives differ in that there are two parameter directions (u, v), so there are multiple types of derivatives per degree of differentiation. The subspace for first derivatives is 2D with basis vectors $\{S_u, S_v\}$, so for C^1 continuity $1 > \mu_1 = \lambda_1 = \lambda_2$ with linearly independent eigenvectors $\{v_1, v_2\}$. The finite differencing from Equation 3.16 also becomes slightly more complicated because not all pairs of vertices in the subdivision mask lie along the same parameter direction. For generality, we will use a polar coordinate system with its origin at the central vertex of the subdivision mask. We choose two parameter directions u ($\theta = 0$) and v ($\theta = \theta_1$) from the central vertex (Figure 3.6), then for C^1 continuity the directional derivatives must satisfy Equation 3.92.

Equation 3.92 is derived by a change of coordinates from the subdivision mask coordinates (u, v) which are unit length with an angle θ_1 between them to a set of polar coordinates (r, θ) (Figure 3.7). Equations 3.84-3.88 use a reference orthonormal coordinate system (s, t) to derive the transformation from (r, θ) to (u, v) in Equation 3.88.

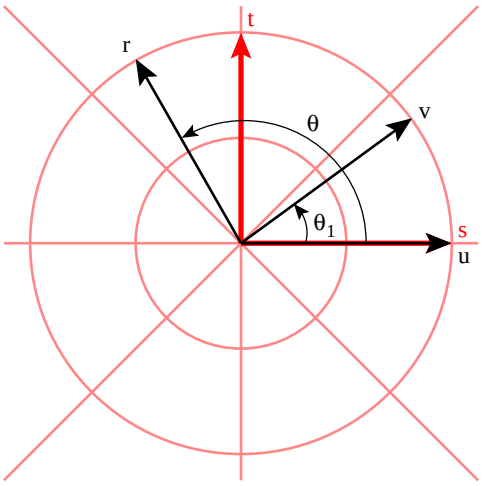


Figure 3.7: *Change of coordinates between the subdivision mask directions* (u, v)*, an orthogonal basis* (s, t)*, and polar coordinates* (r, θ)

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \tag{3.83}$$

$$\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} 1 & \cos\theta_1 \\ 0 & \sin\theta_1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.84}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sin\theta_1} \begin{bmatrix} \sin\theta_1 & -\cos\theta_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} \tag{3.85}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sin\theta_1} \begin{bmatrix} \sin\theta_1 & -\cos\theta_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \tag{3.86}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{r}{\sin\theta_1} \begin{bmatrix} \sin\theta_1\cos\theta - \sin\theta\cos\theta_1 \\ \sin\theta \end{bmatrix} \tag{3.87}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{r}{\sin\theta_1} \begin{bmatrix} \sin(\theta_1 - \theta) \\ \sin\theta \end{bmatrix} \tag{3.88}$$

With this coordinate transformation, we can derive the directional derivative in any direction θ using the chain rule (Equations 3.89-3.92). The relationship in Equation 3.92 must be satisfied for a finite difference taken in a different direction θ in Figure 3.6 for C^1 continuity.

$$\frac{\partial u}{\partial r} = \frac{\sin(\theta_1 - \theta)}{\sin\theta_1} \tag{3.89}$$

$$\frac{\partial v}{\partial r} = \frac{\sin\theta}{\sin\theta_1} \tag{3.90}$$

$$\frac{\partial S(u,v)}{\partial r} = \frac{\partial S(u,v)}{\partial u}\frac{\partial u}{\partial r} + \frac{\partial S(u,v)}{\partial v}\frac{\partial v}{\partial r} \tag{3.91}$$

$$\frac{\partial S(r,\theta)}{\partial r} = \frac{\sin(\theta_1 - \theta)}{\sin\theta_1}\frac{\partial S(u,v)}{\partial u} + \frac{\sin\theta}{\sin\theta_1}\frac{\partial S(u,v)}{\partial v} \tag{3.92}$$

The subspace for second derivatives is 3D with basis vectors $\{S_{uu}, S_{uv} = S_{vu}, S_{vv}\}$, so for C^2 continuity $\mu_1 > \mu_2 = \lambda_3 = \lambda_4 = \lambda_5 > ...\lambda_n \geq 0$ with linearly independent eigenvectors $\{v_3, v_4, v_5\}$. For C^2 continuity, the second directional derivative in any direction θ must match the linear combination of the second derivatives and the mixed derivative with respect to (u, v) (Equation 3.98). We derive this formula by applying the chain rule to Equation 3.92 in Equations 3.93-3.98.

$$\frac{\partial^2 u}{\partial r^2} = 0 \tag{3.93}$$

$$\frac{\partial^2 v}{\partial r^2} = 0 \tag{3.94}$$

$$\begin{aligned}\frac{\partial^2 S(u,v)}{\partial r^2} = &\left(\frac{\partial^2 S(u,v)}{\partial u^2}\frac{\partial u}{\partial r} + \frac{\partial^2 S(u,v)}{\partial u \partial v}\frac{\partial v}{\partial r}\right)\frac{\partial u}{\partial r} + \frac{\partial S(u,v)}{\partial u}\frac{\partial^2 u}{\partial r^2} \\ &+ \left(\frac{\partial^2 S(u,v)}{\partial v \partial u}\frac{\partial u}{\partial r} + \frac{\partial^2 S(u,v)}{\partial v^2}\frac{\partial v}{\partial r}\right)\frac{\partial v}{\partial r} + \frac{\partial S(u,v)}{\partial v}\frac{\partial^2 v}{\partial r^2}\end{aligned} \tag{3.95}$$

$$\frac{\partial^2 S(u,v)}{\partial r^2} = \left(\frac{\partial u}{\partial r}\right)^2 \frac{\partial^2 S(u,v)}{\partial u^2} + \frac{\partial u}{\partial r}\frac{\partial v}{\partial r}\left(\frac{\partial^2 S(u,v)}{\partial u \partial v} + \frac{\partial^2 S(u,v)}{\partial v \partial u}\right) + \left(\frac{\partial v}{\partial r}\right)^2 \frac{\partial^2 S(u,v)}{\partial v^2} \tag{3.96}$$

$$\frac{\partial^2 S(u,v)}{\partial r^2} = \left(\frac{\partial u}{\partial r}\right)^2 \frac{\partial^2 S(u,v)}{\partial u^2} + 2\frac{\partial u}{\partial r}\frac{\partial v}{\partial r}\frac{\partial^2 S(u,v)}{\partial u \partial v} + \left(\frac{\partial v}{\partial r}\right)^2 \frac{\partial^2 S(u,v)}{\partial v^2} \tag{3.97}$$

$$\frac{\partial^2 S(r,\theta)}{\partial r^2} = \frac{\sin^2(\theta_1 - \theta)}{\sin^2\theta_1}\frac{\partial^2 S(u,v)}{\partial u^2} + \frac{2\sin^2(\theta_1 - \theta)\sin^2\theta}{\sin^2\theta_1}\frac{\partial^2 S(u,v)}{\partial u \partial v} + \frac{\sin^2\theta}{\sin^2\theta_1}\frac{\partial^2 S(u,v)}{\partial v^2} \tag{3.98}$$

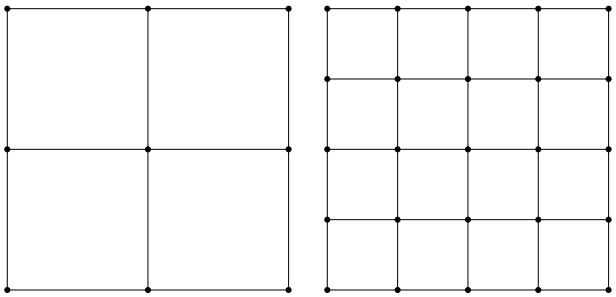
The mixed derivative first in direction u and then in another arbitrary direction θ follows the same rule as the first derivative (Equation 3.100).

$$\frac{\partial^2 S(u,v)}{\partial u \partial r} = \frac{\partial^2 S(u,v)}{\partial u^2}\frac{\partial u}{\partial r} + \frac{\partial^2 S(u,v)}{\partial u \partial v}\frac{\partial v}{\partial r} \tag{3.99}$$

$$\frac{\partial^2 S(r,\theta)}{\partial u \partial r} = \frac{\sin(\theta_1 - \theta)}{\sin\theta_1}\frac{\partial^2 S(u,v)}{\partial u^2} + \frac{\sin\theta}{\sin\theta_1}\frac{\partial^2 S(u,v)}{\partial u \partial v} \tag{3.100}$$

3.6 Catmull-Clark Subdivision Surfaces

Catmull-Clark subdivision surfaces [5] are a generalization of $N_{3,3}$ box splines, i.e. uniform bi-cubic tensor product B-splines, for extraordinary vertices with valence $n \neq 4$. The topological split creates new vertices at the edge midpoints and at the face centers and connects them forming all quadrilateral faces in the next generation (Figure 3.8). All the extraordinary vertices, those specified in the control mesh and the face centers of non-valence 4 faces in the control mesh, are created after one generation of subdivision. After two generations of subdivision, every quadrilateral will have at most one extraordinary vertex.



(a) *Generation* g (b) *Generation* $g+1$

Figure 3.8: *One generation of the Catmull-Clark subdivision topological split*

There are different averaging rules for the three types of created vertices (Figure 3.9). A face-center vertex F_i^{g+1} gets the average of the face's vertices (Equation 3.101). An edge-midpoint vertex E_i^{g+1} gets the average of the edge's endpoints and the two face-center vertices of the two adject faces (Equation 3.102). A child vertex V_0^{g+1} gets a weighted average of its parent vertex, the centroid of the neighboring edge vertices, and the centroid of the face-center vertices (Equation 3.103).

$$F_i^{g+1} = \frac{1}{n}\sum_{i=1}^{n} V_i^g \tag{3.101}$$

$$E_i^{g+1} = \frac{1}{4}\left(V_0^g + E_i^g + F_{i-1}^{g+1} + F_i^{g+1}\right) \tag{3.102}$$

$$V_0^{g+1} = \frac{(n-2)\,V_0^g + \frac{1}{n}\sum_{i=1}^{n} E_i^g + \frac{1}{n}\sum_{i=1}^{n} F_i^{g+1}}{n} \tag{3.103}$$

Both the face and edge rules are derived from midpoint knot insertion to bi-cubic B-splines. The generalized vertex rule also agrees with midpoint knot insertion mask when $n = 4$.

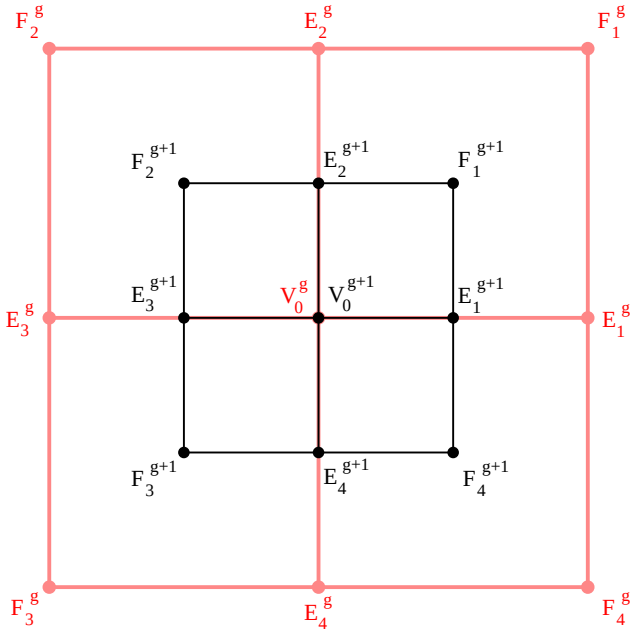


Figure 3.9: *The 1-ring mask for one step of Catmull-Clark subdivision*

3.6.1 Analysis of a Catmull-Clark Regular Vertex

The limit surface around a valence $n = 4$ regular vertex is an $N_{3,3}$ box spline, so our analysis should show it to be C^2 continuous. The 1-ring subdivision mask S is shown in Figure 3.9 and Equation 3.104.

$$S = \frac{1}{64}\begin{bmatrix} 36 & 6 & 1 & 6 & 1 & 6 & 1 & 6 & 1 \\ 24 & 24 & 4 & 4 & 0 & 0 & 0 & 4 & 4 \\ 16 & 16 & 16 & 16 & 0 & 0 & 0 & 0 & 0 \\ 24 & 4 & 4 & 24 & 4 & 4 & 0 & 0 & 0 \\ 16 & 0 & 0 & 16 & 16 & 16 & 0 & 0 & 0 \\ 24 & 0 & 0 & 4 & 4 & 24 & 4 & 4 & 0 \\ 16 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 0 \\ 24 & 4 & 0 & 0 & 0 & 4 & 4 & 24 & 4 \\ 16 & 16 & 0 & 0 & 0 & 0 & 0 & 16 & 16 \end{bmatrix} \tag{3.104}$$

Equation 3.105 shows the eigen-decomposition of S. Note that the magnitude of the eigenvalues $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}, \frac{1}{16}\right\}$ are consistent with pattern for midpoint subdivision of B-spline curves $\mu_i = 2^{-i}$, and that the algebraic multiplicities are consistent with the derivatives for surfaces.

$$\begin{aligned} S =& V\Lambda V^{-1} \\ =& \begin{bmatrix} 1 & 0 & 0 & -\frac{1}{4} & 0 & -\frac{1}{4} & 0 & 0 & \frac{1}{4} \\ 1 & -1 & 1 & \frac{1}{4} & -1 & \frac{1}{4} & -1 & -\frac{1}{2} & -\frac{1}{2} \\ 1 & -2 & 1 & 1 & 0 & 0 & 4 & 1 & 1 \\ 1 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & -\frac{1}{2} \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & 1 \\ 1 & 1 & -1 & \frac{1}{4} & -1 & \frac{1}{4} & 1 & \frac{1}{2} & -\frac{1}{2} \\ 1 & 2 & -1 & 1 & 0 & 0 & -4 & -1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & -\frac{1}{2} \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} \end{bmatrix} \cdot \\ & \frac{1}{36}\begin{bmatrix} 16 & 4 & 1 & 4 & 1 & 4 & 1 & 4 & 1 \\ 0 & 0 & -3 & -12 & -3 & 0 & 3 & 12 & 3 \\ 0 & 12 & 0 & -12 & -6 & -12 & 0 & 12 & 6 \\ -32 & 4 & 13 & 4 & -5 & 4 & 13 & 4 & -5 \\ -8 & -8 & 1 & 10 & 1 & -8 & 1 & 10 & 1 \\ -32 & 4 & -5 & 4 & 13 & 4 & -5 & 4 & 13 \\ 0 & 0 & 3 & -6 & 3 & 0 & -3 & 6 & -3 \\ 0 & -12 & 0 & 12 & -12 & 12 & 0 & -12 & 12 \\ 16 & -8 & 4 & -8 & 4 & -8 & 4 & -8 & 4 \end{bmatrix} \end{aligned} \tag{3.105}$$

The weights for the limit position V_0^∞ are the elements of v_0^{-1} (Equation 3.106). By taking the limit of finite differences of rows of S^∞ in the patterns shown in Figure 3.10, it can be shown that the surface satisfies Equations 3.92 and 3.98, and it is thus C^2 continuous (Equations 3.107-3.111).

$$V_0^\infty = \frac{16V_0^g + 4\sum_{i=1}^{n} E_i^g + \sum_{i=1}^{n} F_i^g}{36} \tag{3.106}$$

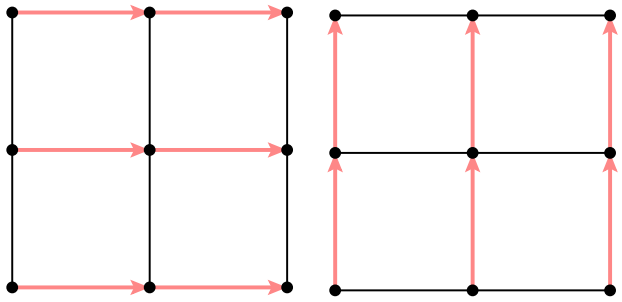
$$S_u = \frac{(F_4^g + 4E_1^g + F_1^g) - (F_2^g + 4E_3^g + F_3^g)}{12} \tag{3.107}$$

$$S_v = \frac{(F_1^g + 4E_2^g + F_2^g) - (F_3^g + 4E_4^g + F_4^g)}{12} \tag{3.108}$$

$$S_{uu} = \frac{(F_4^g + 4E_1^g + F_1^g)}{6} - \frac{(2E_4^g + 8V_0^g + 2E_2^g)}{6} + \frac{(F_2^g + 4E_3^g + F_3^g)}{6} \tag{3.109}$$

$$S_{vv} = \frac{(F_1^g + 4E_2^g + F_2^g)}{6} - \frac{(2E_1^g + 8V_0^g + 2E_3^g)}{6} + \frac{(F_3^g + 4E_4^g + F_4^g)}{6} \tag{3.110}$$

$$S_{uv} = \frac{(F_1^g + F_3^g) - (F_2^g + F_2^g)}{4} \tag{3.111}$$



(a) $\frac{\partial S}{\partial u}$ (b) $\frac{\partial S}{\partial v}$

Figure 3.10: *Finite differences for Catmull-Clark subdivision derivatives*

3.6.2 Analysis of a Catmull-Clark Extraordinary Vertex

The surface is only C^1 at extraordinary vertices. The results for the limit position V_0^∞ and the tangents T_0 and T_1 are shown in Equations 3.112 and 3.115.

$$V_0^\infty = \frac{n^2 V_0^g + 4\sum_{i=1}^n E_i^g + \sum_{i=1}^n F_i^g}{n\,(n+5)} \tag{3.112}$$

$$\alpha_n = 1 + \cos\left(\frac{2\pi}{n}\right) + \cos\left(\frac{\pi}{n}\right)\sqrt{2\left(9 + \cos\left(\frac{2\pi}{n}\right)\right)} \tag{3.113}$$

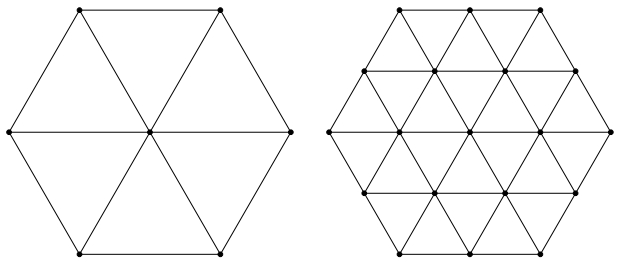
$$c\,(i) = \cos\left(\frac{2\pi i}{n}\right) \tag{3.114}$$

$$T_j = \sum_{i=1}^{n} \alpha_n c\,(i+j)\,E_i^g + (c\,(i+j) + c\,(i+j+1))\,F_i^{g+1} \tag{3.115}$$

$$N = T_0 \times T_1 \tag{3.116}$$

3.7 Loop Subdivision Surfaces

Loop subdivision surfaces [15] are a generalization of $N_{2,2,2}$ box splines for extraordinary vertices with valence $n \neq 6$. Loop subdivision is defined for triangulated control meshes only. The topological split creates new vertices at the edge midpoints and connects them by cutting off the corners to form four subtriangles in the next generation (Figure 3.11). The set of extraordinary vertices is the set defined in the original control mesh only. After one generation of subdivision each triangle will have at most one extraordinary vertex.



(a) *Generation* g (b) *Generation* $g+1$

Figure 3.11: *One generation of the Loop subdivision topological split*

There are different averaging rules for the two types of created vertices (Figure 3.12). An edge-midpoint vertex E_i^{g+1} gets the weighted average of the vertices on the two faces adject to the edge (Equation 3.117). A child vertex V_0^{g+1} gets a weighted average of its parent vertex and the centroid of the neighboring edge vertices (Equation 3.119).

$$E_i^{g+1} = \frac{1}{8}\left(3V_0^g + 3E_i^g + E_{i-1}^g + E_{i+1}^g\right) \tag{3.117}$$

$$\beta_n = \frac{5}{8} - \frac{\left(3 + 2\cos\left(\frac{2\pi}{n}\right)\right)^2}{64} \tag{3.118}$$

$$V_0^{g+1} = \left(1 - \beta_n\right) V_0^g + \beta_n \frac{1}{n} \sum_{i=1}^{n} E_i^g \tag{3.119}$$

Note that the edge rule and the vertex rule for valence $n = 6$ are the midpoint subdivision rules for the $N_{2,2,2}$ box spline.

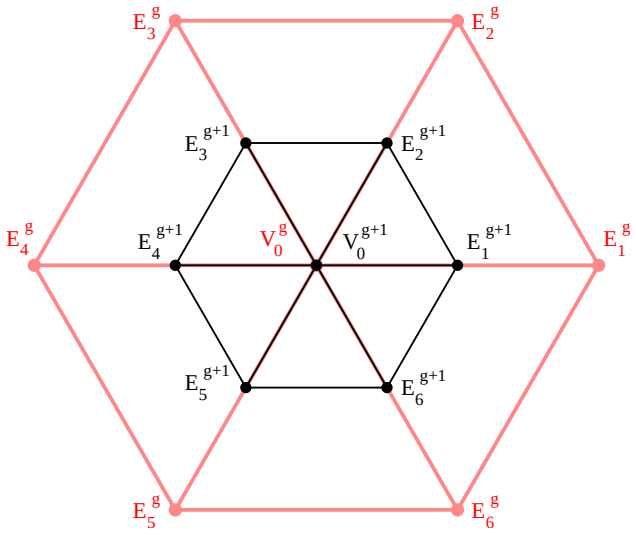


Figure 3.12: *The 1-ring mask for one step of Loop subdivision*

3.7.1 Analysis of a Loop Regular Vertex

The limit surface around a valence $n = 6$ regular vertex is an $N_{2,2,2}$ box spline, so our analysis should show it to be C^2 continuous. The 1-ring subdivision mask S is shown in Figure 3.12 and Equation 3.120.

$$S = \frac{1}{16}\begin{bmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 \\ 6 & 6 & 2 & 0 & 0 & 0 & 2 \\ 6 & 2 & 6 & 2 & 0 & 0 & 0 \\ 6 & 0 & 2 & 6 & 2 & 0 & 0 \\ 6 & 0 & 0 & 2 & 6 & 2 & 0 \\ 6 & 0 & 0 & 0 & 2 & 6 & 2 \\ 6 & 2 & 0 & 0 & 0 & 2 & 6 \end{bmatrix} \tag{3.120}$$

Equation 3.121 shows the eigen-decomposition of S. Note that the magnitude of the eigenvalues $\left\{1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right\}$ are consistent with the pattern for midpoint subdivision of B-spline curves $\mu_i = 2^{-i}$, and that the algebraic multiplicities are consistent with the derivatives for surfaces.

$$\begin{aligned} S =& V\Lambda V^{-1} \\ =& \begin{bmatrix} 1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\ 1 & -1 & 1 & \frac{3}{3} & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & \frac{3}{3} & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & \frac{3}{3} & -1 \\ 1 & 1 & -1 & \frac{3}{3} & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & \frac{3}{3} & 0 & -1 \\ 1 & 0 & 1 & 0 & 0 & \frac{3}{3} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} \end{bmatrix} \cdot \\ & \begin{bmatrix} \frac{6}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\ 0 & -\frac{1}{6} & -\frac{2}{6} & -\frac{1}{6} & \frac{1}{6} & \frac{2}{6} & \frac{1}{6} \\ 0 & \frac{1}{6} & -\frac{1}{6} & -\frac{2}{6} & -\frac{1}{6} & \frac{1}{6} & \frac{2}{6} \\ -\frac{6}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} \\ -\frac{6}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} \\ -\frac{6}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} \\ 0 & -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{6} \end{bmatrix} \end{aligned} \tag{3.121}$$

The weights for the limit position V_0^∞ are the elements of v_0^{-1} (Equation 3.122). By taking the limit of finite differences of rows of S^∞ in the patterns shown in Figure 3.13, it can be shown that the surface satisfies Equations 3.92 and 3.98, and it is thus C^2 continuous (Equations 3.123-3.127).

$$V_0^\infty = \frac{6V_0^g + \sum_{i=1}^{n} E_i^g}{12} \tag{3.122}$$

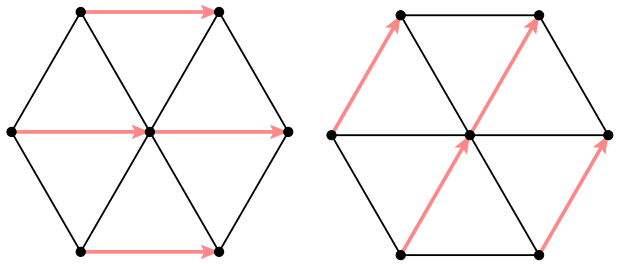
$$S_u = \frac{(E_6^g + 2E_1^g + E_2^g) - (E_3^g + 2E_4^g + E_5^g)}{2} \tag{3.123}$$

$$S_v = \frac{(E_1^g + 2E_2^g + E_3^g) - (E_4^g + 2E_5^g + E_6^g)}{2} \tag{3.124}$$

$$S_{uu} = E_1^g - 2V_0^g + E_4^g \tag{3.125}$$

$$S_{vv} = E_2^g - 2V_0^g + E_5^g \tag{3.126}$$

$$S_{uv} = \frac{(E_1^g + E_2^g + E_4^g + E_5^g) - (E_3^g + 2V_0^g + E_3^g)}{2} \tag{3.127}$$



(a) $\frac{\partial S}{\partial u}$ (b) $\frac{\partial S}{\partial v}$

Figure 3.13: *Finite differences for Loop subdivision derivatives*

3.7.2 Analysis of a Loop Extraordinary Vertex

The surface is only C^1 at extraordinary vertices. The results for the limit position V_0^∞, the position at any generation V_0^g, and the tangents T_0 and T_1 are shown in Equations 3.129, 3.133, and 3.134.

$$\delta_n = \frac{8\beta_n}{3 + 8\beta_n} \tag{3.128}$$

$$V_0^\infty = (1 - \delta_n)\, V_0^g + \delta_n \frac{1}{n} \sum_{i=1}^{n} E_i^g \tag{3.129}$$

$$\gamma_n = \frac{5 - 8\beta_n}{8} \tag{3.130}$$

$$\tau\,(n, g) = (1 - \delta_n) + \delta_n \gamma_n^g \tag{3.131}$$

$$V_0^g = \tau\,(n, g)\, V_0^0 + (1 - \tau\,(n, g))\, \frac{1}{n} \sum_{i=1}^{n} E_i^0 \tag{3.132}$$

$$= V_0^\infty + \gamma_n^g \delta_n \left(V_0^0 - \frac{1}{n} \sum_{i=1}^{n} E_i^0 \right) \tag{3.133}$$

$$T_j = \sum_{i=1}^{n} \cos\left(\frac{2\pi\,(i + j)}{n} \right) E_i^g \tag{3.134}$$

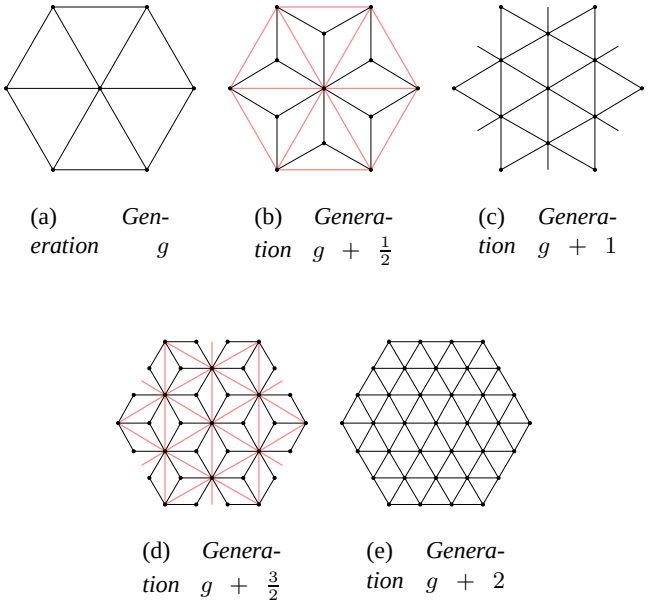
$$N = T_0 \times T_1 \tag{3.135}$$

3.8 $\sqrt{3}$ Subdivision Surfaces

$\sqrt{3}$ subdivision surfaces [12] also work on triangulated control meshes only. The limit surface at regular vertices with valence $n = 6$ is very similar to $N_{2,2,2}$ box splines. The topological split creates new vertices at the triangle centers. In the first half of the topological step, the triangle centers are connected to the original triangle corners (Figure 3.14(b)). For the moment, this doubles the valence at every vertex. In the second half of the topological step, all of the original edges are flipped with in their quadrilateral (Figure 3.14(c)). This restores the original valence to child vertices, and makes the valence of all triangle-center vertices equal to 6. The set of extraordinary vertices is the set defined in the original control mesh only. After one generation of subdivision each triangle will have at most one extraordinary vertex.

The topological split rotates the edges about a vertex by $\frac{\pi}{n}$. Another full topological split rotates the edges another $\frac{\pi}{n}$ making them coincide with their original orientation (Figure 3.14(e)). Two subdivision steps in effect trisect the original edges, which is like setting $b = 3$ in our uniform B-spline curve example. So one subdivision does a $\sqrt{3}$ split which is where the scheme gets its name.

There are different averaging rules for the two types of created vertices (Figure 3.15(a)). A triangle-center vertex F_i^{g+1} gets the centroid of the triangle (Equation 3.136). A child vertex



(a) *Generation* g (b) *Generation* $g + \frac{1}{2}$ (c) *Generation* $g + 1$

(d) *Generation* $g + \frac{3}{2}$ (e) *Generation* $g + 2$

Figure 3.14: *Two generations of the* $\sqrt{3}$ *subdivision topological split*

V_0^{g+1} gets a weighted average of its parent vertex and the centroid of the neighboring edge vertices (Equation 3.138).

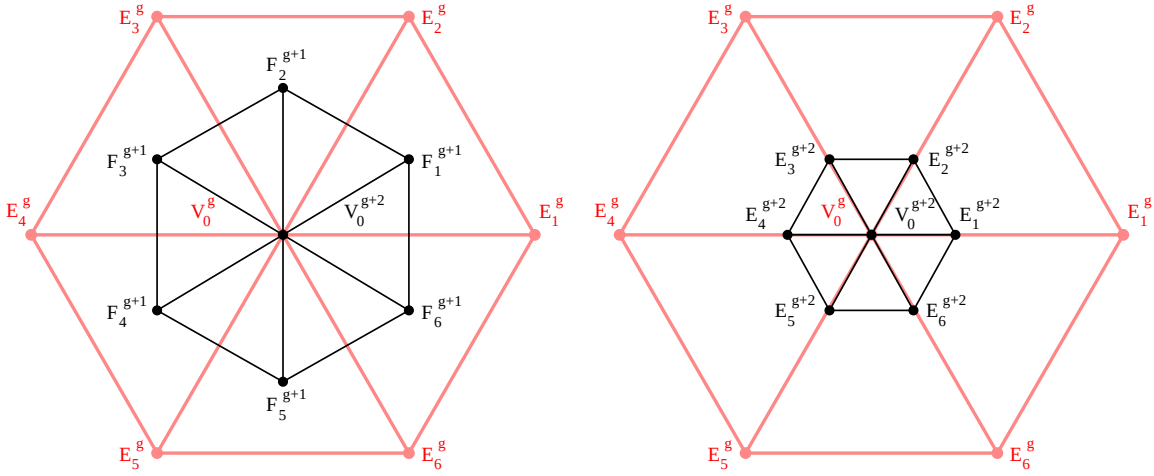
$$F_i^{g+1} = \frac{1}{3}\left(V_0^g + E_i^g + E_{i+1}^g\right) \tag{3.136}$$

$$\alpha_n = \frac{4 - 2\cos\left(\frac{2\pi}{n}\right)}{9} \tag{3.137}$$

$$V_0^{g+1} = (1 - \alpha_n)\, V_0^g + \alpha_n \frac{1}{n} \sum_{i=1}^{n} E_i^g \tag{3.138}$$

3.8.1 Analysis of a $\sqrt{3}$ Regular Vertex

The $\frac{\pi}{n}$ rotation makes it difficult to analyze the single step subdivision matrix. Instead, we will analyze the two step 1-ring subdivision mask S, shown in Figure 3.15(b) and Equation 3.139.



(a) *Step 1* (b) *Step 2*

Figure 3.15: *The 1-ring mask for one and two steps of* $\sqrt{3}$ *subdivision*

$$S = \frac{1}{54}\begin{bmatrix} 30 & 4 & 4 & 4 & 4 & 4 & 4 \\ 24 & 13 & 7 & 1 & 1 & 1 & 7 \\ 24 & 7 & 13 & 7 & 1 & 1 & 1 \\ 24 & 1 & 7 & 13 & 7 & 1 & 1 \\ 24 & 1 & 1 & 7 & 13 & 7 & 1 \\ 24 & 1 & 1 & 1 & 7 & 13 & 7 \\ 24 & 7 & 1 & 1 & 1 & 7 & 13 \end{bmatrix} \tag{3.139}$$

Equation 3.140 shows the eigen-decomposition of S. Note that the magnitude of the eigenvalues $\left\{1, \frac{1}{3}, \frac{1}{3}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0\right\}$ up until the final one are consistent with pattern for trisection of B-spline curves $\mu_i = 3^{-i}$. Also note that the eigenvectors in V are precisely the same as the ones for Loop valence $n = 6$ (Equation 3.121), i.e. $N_{2,2,2}$ box splines. There is no need to repeat the finite differencing analysis because it is exactly the same as Loop up to C^2 continuity. The two schemes only differ in the third order terms which have been truncated in the $\sqrt{3}$ scheme. The limit position and derivatives are the same as Loop (Equations 3.122-3.127).

$$
\begin{aligned}
S =& V\Lambda V^{-1} \\
=& \begin{bmatrix}
1 & 0 & 0 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \\
1 & -1 & 1 & \frac{3}{3} & 0 & 0 & -1 \\
1 & -1 & 0 & 0 & \frac{3}{3} & 0 & 1 \\
1 & 0 & -1 & 0 & 0 & \frac{3}{3} & -1 \\
1 & 1 & -1 & \frac{3}{3} & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & \frac{3}{3} & 0 & -1 \\
1 & 0 & 1 & 0 & 0 & \frac{3}{3} & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{9} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{9} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{9} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \cdot \\
& \begin{bmatrix}
\frac{6}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \\
0 & -\frac{1}{6} & -\frac{2}{6} & -\frac{1}{6} & \frac{1}{6} & \frac{2}{6} & \frac{1}{6} \\
0 & \frac{1}{6} & -\frac{1}{6} & -\frac{2}{6} & -\frac{1}{6} & \frac{1}{6} & \frac{2}{6} \\
-\frac{6}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} \\
-\frac{6}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} \\
-\frac{6}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} & -\frac{1}{12} & -\frac{1}{12} & \frac{5}{12} \\
0 & -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & \frac{1}{6}
\end{bmatrix}
\end{aligned}
\tag{3.140}
$$

3.8.2 Analysis of a $\sqrt{3}$ Extraordinary Vertex

The surface is only C^1 at extraordinary vertices. The results for the limit position V_0^∞, the position at any generation V_0^g, and the tangents T_0 and T_1 are shown in Equations 3.143, 3.145, and 3.146.

$$\beta(n,g) = \frac{3\alpha_n\left(1-\left(\frac{2}{3}-\alpha_n\right)^g\right)}{1+3\alpha_n} \tag{3.141}$$

$$\beta(n,\infty) = \frac{3\alpha_n}{1+3\alpha_n} \tag{3.142}$$

$$V_0^g = \left(1-\beta(n,g)\right)V_0^0 + \beta(n,g)\,\frac{1}{n}\sum_{i=1}^{n} E_i^0 \tag{3.143}$$

$$\gamma(n,g) = \left(\frac{2}{3}-\alpha_n\right)^g \tag{3.144}$$

$$V_0^g = \gamma(n,g)\,V_0^0 + \left(1-\gamma(n,g)\right)V_0^\infty \tag{3.145}$$

$$T_j = \sum_{i=1}^{n} \cos\left(\frac{2\pi(i+j)}{n}\right) E_i^g \tag{3.146}$$

$$N = T_0 \times T_1 \tag{3.147}$$

3.9 Improving Surface Behavior

We have seen that the Catmull-Clark, Loop, and $\sqrt{3}$ subdivision schemes are C^2 continuous in the regular areas where they act as box splines, and are only C^1 continuous near extraordinary vertices. In the standard schemes that we have described, the face and edge rules are same for all faces and edges. The edge rules adjacent to extraordinary vertices can be made a function of the valence of that vertex to change the spectrum of eigenvalues. To achieve bounded curvature at the extraordinary vertex the first six eigenvalues must be of the form $\left\{1, \lambda, \lambda, \lambda^2, \lambda^2, \lambda^2\right\}$ with $1 > \lambda$ and all other eigenvalues between λ^2 and zero. The Catmull-Clark [23] and Loop [16] schemes have been improved apon to achieve this bounded curvature condition.

The value λ of the first eigenvalue is important as well. As we saw with the B-spline curve example, the eigenvalue specifies how many knots per knot interval are being inserted. Since the topological splits of Catmull-Clark and Loop subdivision insert one vertex per edge, the optimal first eigenvalue would be $\lambda = \frac{1}{2}$. Hence given a regular arrangement of polygons in the parameter domain, the edge averaging rule must generate the edge midpoint. The relative sizes of polygons throughout all generations would then remain constant. This is not the case with the standard rules for Catmull-Clark and Loop which generate smaller than desired polygons for valences less than the regular case and larger than desired polygons for valence larger than the regular case.

Because subdivision schemes are based on different box splines, they have different prefered directions of averaging corresponding to the directions of convolutions of the box splines. There has been work to try to combine the Loop and Catmull-Clark subdivision schemes to make a hybrid triangle and quadrilateral scheme [30, 14, 20]. The newest results maintain C^2 continuity across the semi-regular vertex that has three consecutive triangles and two consecutive quadrilaterals around it.

It is not possible to define a stationary 1-ring averaging rule scheme that is C^2 everywhere. The second derivatives do not exist at the extraordinary vertices. It is possible to do a least squares fit of a quadric surface and use its second derivatives as an estimate. A new approach [32] performs a finite number of uniform subdivisions, and then uses an exponential mapping of parameter charts to generate a C^∞ arbitrary topology surface. This representation could then be used as a primitive in an optimization process to generate minimal energy surfaces that meet user constraints.

3.10 Conclusion

In this chapter, we have shown that subdivision surfaces are a generalization of box spline surfaces to meshes with general topology. The degree of continuity of box splines can be understood by the definition of box splines as a series of convolutions. Using our knowledge of box splines, we applied eigen-analysis to the regular subdivision matrix cases to show what criteria is necessary from the set of averaging rules to achieve C^2 continuity. We also briefly listed the limit positions, tangents, and normals at extraordinary vertices, which were derived using the same methods. It is not possible to achieve C^2 at extraordinary vertices of a 1-ring stationary subdivision scheme, but we mentioned schemes to improve the surface behavior there. In Chapters 4-5, we show how to derive the eigen-decomposition of the subdivision matrices of arbitrary valence piecewise smooth Loop and Catmull-Clark vertices. These decompositions can be used to study the limit behavior of the surfaces at these extraordinary vertices and provide derivations to the formulas given in this chapter.

Chapter 4

Signal Processing and Subdivision

4.1 Introduction

The pattern of edge and face subdivision averaging rules around a vertex can be viewed as repeated, shifted applications of a difference equation. In this chapter, we will use 1D signal processing to derive the eigen-decomposition of the subdivision matrix, which we will later use to eigen-analyze the continuity behavior of subdivision surfaces. The subdivision matrices S described here are 1-ring matrices of extraordinary vertices and have the form:

$$S = \left[\begin{array}{c|c} T_0 & T_{02} \\ \hline T_{11} & T_{12} \end{array}\right] \tag{4.1}$$

The T_0 subblock contains portions of the vertex and sharp edge averaging rules. The T_{12} subblock contains the regular face and edge averaging rules.

4.2 Difference Equations

A difference equation, like a differential equation, defines a family of functions by an implicit relationship between values of the function taken at consecutive samples. One classical example is the Fibonacci sequence $\{1, 1, 2, 3, 5, \ldots\}$ which is defined by the rule that the next value in the sequence is the sum of the previous two values.

$$f[t] = f[t-2] + f[t-1] \tag{4.2}$$

This relationship defines a family of functions, and to fully specify the Fibonacci sequence, it is necessary to give initial values for the function.

$$f[0] = 1, f[1] = 1 \tag{4.3}$$

For a general linear difference equation, the value at the current sample $f[t]$ is a linear combination of values of the function at earlier samples.

$$f[t] = \sum_{i=1}^{m} a_i f[t-i] \tag{4.4}$$

A common function that can be described as a difference equation is the cosine function. This relationship can be derived by starting with the quantity $\cos(\phi - 2\theta)$ and applying trigonometric identities to get Equation 4.9.

$$\begin{aligned}
\cos(\phi - 2\theta) &= \cos(\phi)\cos(2\theta) + \sin(\phi)\sin(2\theta) && (4.5)\\
&= \cos(\phi)\left(2\cos^2(\theta) - 1\right) + 2\sin(\phi)\sin(\theta)\cos(\theta) && (4.6)\\
&= 2\cos(\theta)\left(\cos(\phi)\cos(\theta) + \sin(\phi)\sin(\theta)\right) - \cos(\phi) && (4.7)\\
&= 2\cos(\theta)\cos(\phi - \theta) - \cos(\phi) && (4.8)\\
0 &= \cos(\phi - 2\theta) - 2\cos(\theta)\cos(\phi - \theta) + \cos(\phi) && (4.9)
\end{aligned}$$

This relationship shows that for a fixed angle step θ, the value of $\cos(\phi)$ can be computed from a combination of the values at the previous two steps. We can discretize Equation 4.9 to the difference equation in Equation 4.10. When setting an initial condition that $f[t]$ has even symmetry about the origin, the solution is then cosine sampled at intervals of θ.

$$f[t-2] - 2\cos(\theta) f[t-1] + f[t] = 0 \tag{4.10}$$

$$f[-1] = f[1] = \cos(-\theta) \tag{4.11}$$

$$f[t] = \cos[t\theta] \tag{4.12}$$

The sine function is just a phase shift of cosine, so it follows the same steady state governing rule. We derive the difference equation for sine by using the identity $\sin(\phi) = \cos\left(\frac{\pi}{2} - \phi\right)$ and substitute $\phi \rightarrow \left(\frac{\pi}{2} - \phi + 2\theta\right)$ into Equation 4.9.

$$0 = \cos\left(\frac{\pi}{2} - \phi + 2\theta - 2\theta\right) - 2\cos(\theta)\cos\left(\frac{\pi}{2} - \phi + 2\theta - \theta\right) + \cos\left(\frac{\pi}{2} - \phi + 2\theta\right) \tag{4.13}$$

$$0 = \cos\left(\frac{\pi}{2} - \phi\right) - 2\cos(\theta)\cos\left(\frac{\pi}{2} - (\phi - \theta)\right) + \cos\left(\frac{\pi}{2} - (\phi - 2\theta)\right) \tag{4.14}$$

$$0 = \sin(\phi) - 2\cos(\theta)\sin(\phi - \theta) + \sin(\phi - 2\theta) \tag{4.15}$$

The discrete difference equation is then the same as for cosine with the intial conditions modified to have odd symmetry about the origin.

$$f[t-2] - 2\cos(\theta) f[t-1] + f[t] = 0 \tag{4.16}$$

$$f[-1] = -f[1] = \sin(-\theta) \tag{4.17}$$

$$f[t] = \sin[t\theta] \tag{4.18}$$

The general solution to this difference equation can then be any linear combination of these sine and cosine functions.

4.3 Loop Eigenvectors

The cosine difference equation arises when solving for the eigenvector elements associated with the edge averaging rules of Loop subdivision. When solving for an eigenvector v_i associated with an eigenvalue λ_i of the subdivision matrix S, we solve the linear system:

$$0 = (S - \lambda_i I) v_i \tag{4.19}$$

In the Loop case, the edge rules have a repeated set of three coefficients along the diagonal as shown in Equation 4.20.

$$e = \frac{1}{8}, d = 3e - \lambda_i$$

$$0 = \begin{bmatrix} \ddots & & & & & \\ e & d & e & & & \\ & e & d & e & & \\ & & e & d & e & \\ & & & & \ddots & \end{bmatrix} \begin{bmatrix} \vdots \\ f[-1] \\ f[0] \\ f[1] \\ \vdots \end{bmatrix} \tag{4.20}$$

The elements $f[t]$ of the eigenvector can then be thought of as discrete samples from a 1D signal that is described by the difference equation:

$$0 = ef[t-2] + (3e - \lambda_i) f[t-1] + ef[t] \tag{4.21}$$

We normalize this equation by dividing by e and substituting $\alpha = \frac{3e-\lambda_i}{e}$.

$$0 = f[t-2] + \alpha f[t-1] + f[t] \tag{4.22}$$

The eigenvectors of S are a linearly independent set of functions from the family of functions that satisfy this difference equation. If we let $\alpha = -2\cos(\theta_i)$, then both $\cos[t\theta_i]$ and $\sin[t\theta_i]$ are candidate functions dependent on boundary conditions. For both sine and cosine, the relationship between the eigenvalue λ_i and the frequency θ_i of the signal in the associated eigenvector can be derived by examining α.

$$\alpha = \frac{3e - \lambda_i}{e} = -2\cos(\theta_i) \tag{4.23}$$

$$\lambda_i = e(3 + 2\cos(\theta_i)) \tag{4.24}$$

$$\cos(\theta_i) = -\frac{3e - \lambda_i}{2e} \tag{4.25}$$

Equation 4.25 shows that a sine or cosine based eigenvector is only possible if the eigenvalue is within certain limits because $-1 \leq \cos(\theta_i) \leq 1$.

$$-1 \leq -\frac{3e - \lambda_i}{2e} \leq 1 \tag{4.26}$$

$$-2e \leq \lambda_i - 3e \leq 2e \tag{4.27}$$

$$e \leq \lambda_i \leq 5e \tag{4.28}$$

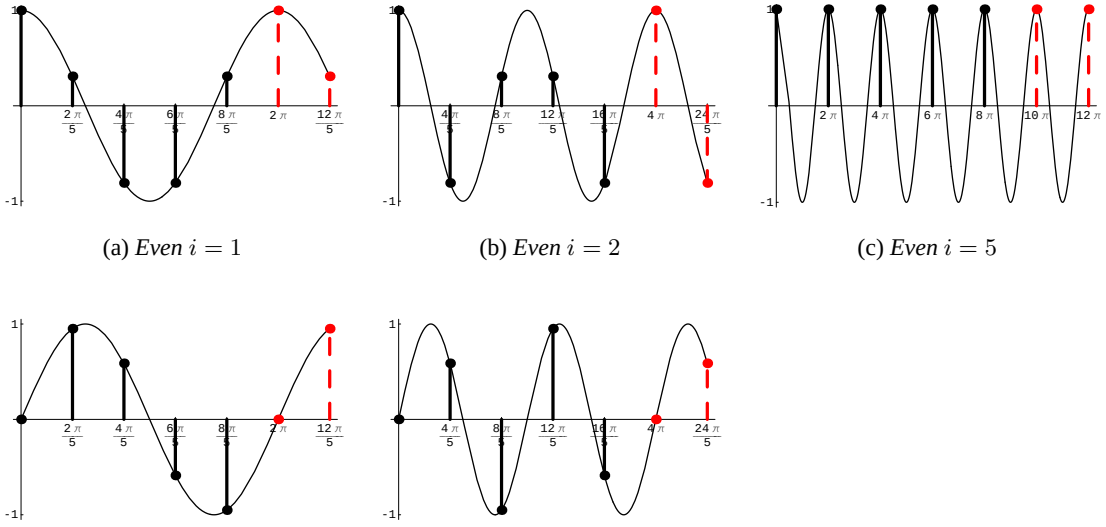
$$\frac{1}{8} \leq \lambda_i \leq \frac{5}{8} \tag{4.29}$$

The magnitude of the eigenvalues will be an issue in the dart extraordinary vertex case.

The boundary conditions for these eigenvectors is determined by the type of the extraordinary vertex in the subdivision matrix. In the smooth and spike cases, there is a cyclic boundary condition. In the crease and corner cases, a natural boundary condition is necessary as the averaging rules terminate at the sharp edges. The dart case is similar to the corner and crease cases, but for even symmetry cosine eigenvectors it is necessary to have the terminating value be non-zero.

4.3.1 Loop Smooth and Spike Eigenvectors

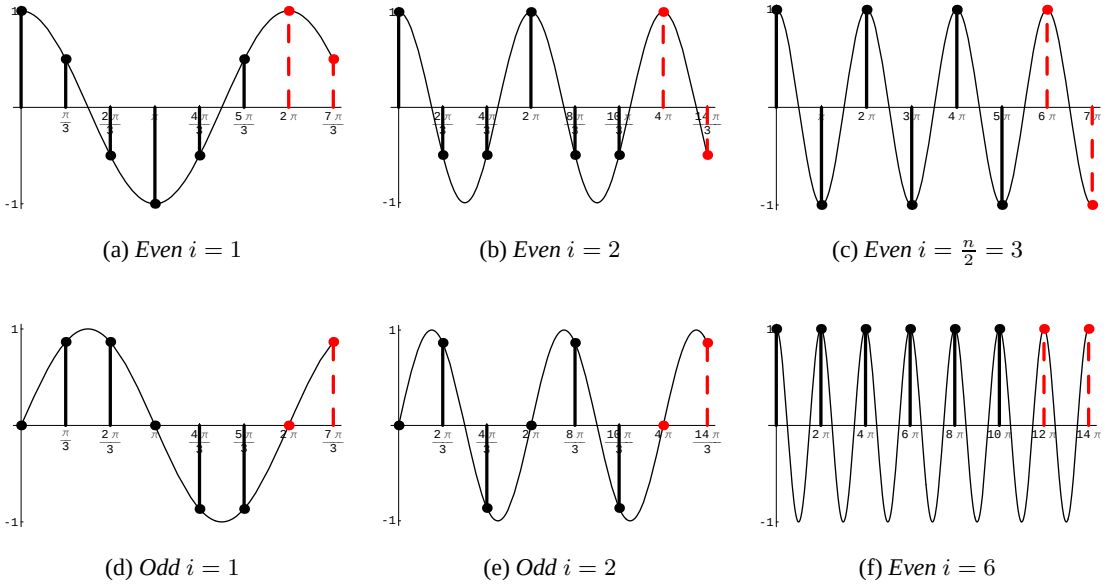
The Loop edge averaging rules in the T_{12} subblock for the smooth and spike vertex cases wrap around in a cyclic manner, Equation 4.30. The elements of the eigenvector, for a vertex of valence n, are $f[t]$ for $t \in [0, n-1]$. Equation 4.30 defines the values of the eigenvector for a



(a) *Even* $i = 1$ (b) *Even* $i = 2$ (c) *Even* $i = 5$

(d) *Odd* $i = 1$ (e) *Odd* $i = 2$

Figure 4.1: *Loop smooth and spike vertex eigenvectors for valence* $n = 5$*. The even functions are* $\cos\left[t\frac{2\pi i}{n}\right]$*, while the odd functions are* $\sin\left[t\frac{2\pi i}{n}\right]$*. The last two samples are not part of the eigenvector sequence, but instead demonstrate that the signal cyclically wraps around.*



(a) *Even* $i = 1$ (b) *Even* $i = 2$ (c) *Even* $i = \frac{n}{2} = 3$

(d) *Odd* $i = 1$ (e) *Odd* $i = 2$ (f) *Even* $i = 6$

Figure 4.2: *Loop smooth and spike vertex eigenvectors for valence* $n = 6$*. The even functions are* $\cos\left[t\frac{2\pi i}{n}\right]$*, while the odd functions are* $\sin\left[t\frac{2\pi i}{n}\right]$*. The last two samples are not part of the eigenvector sequence, but instead demonstrate that the signal cyclically wraps around.*

given eigenvalue λ_i.

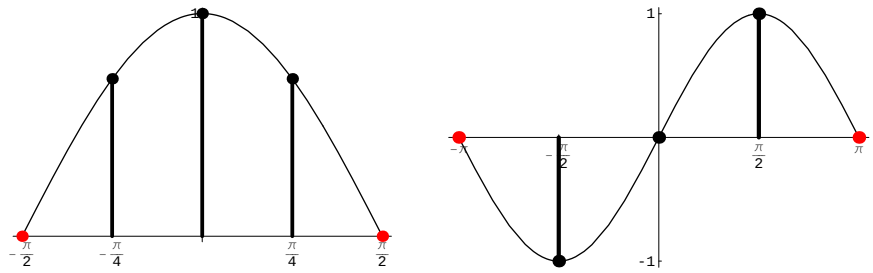
$$e = \frac{1}{8}, d = 3e - \lambda_i$$

$$0 = \begin{bmatrix} d & e & & & & & e \\ e & d & e & & & & \\ & e & d & e & & & \\ & & & \ddots & & & \\ & & & e & d & e & \\ & & & & e & d & e \\ e & & & & & e & d \end{bmatrix} \begin{bmatrix} \boxed{f[n-1]} \\ f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[n-3] \\ f[n-2] \\ f[n-1] \\ \boxed{f[0]} \end{bmatrix} \tag{4.30}$$

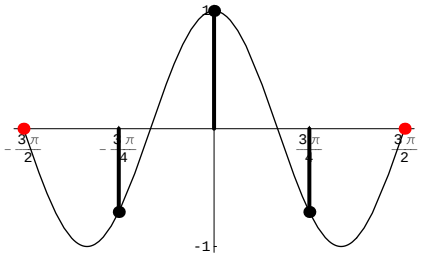
As indicated by the highlighted extra elements, the cyclic nature of the matrix shows that $f[n] = f[0]$ and $f[-1] = f[n-1]$. $f[t]$ must then be periodic in n, so $\theta_i = \frac{2\pi i}{n}$. The functions $f[t] = \cos\left[t\frac{2\pi i}{n}\right]$ and $f[t] = \sin\left[t\frac{2\pi i}{n}\right]$ for $i \in \left\{1, \left\lfloor\frac{n-1}{2}\right\rfloor\right\}$ define linearly independent eigenvectors, see Figures 4.1 and 4.2. If n is even, then an additional cosine eigenvector with $i = \frac{n}{2}$ exists, while the corresponding sine function is all zeros. The final eigenvector in both cases is all ones which corresponds to $i = n$.

4.3.2 Loop Crease and Corner Eigenvectors

The Loop edge averaging rules for a wedge in the crease and corner vertex cases get clipped at the sharp edge boundary indicated by the zeros in the upper right and lower left corners of the matrices in Equations 4.31 and 4.32. The elements of the eigenvector for a wedge of valence n are $f[t]$ for $t \in \left[-\frac{n-2}{2}, \frac{n-2}{2}\right]$. Equation 4.31 defines the values of the eigenvector for a given



(a) *i=1* (b) *i=2*



(c) *i=3*

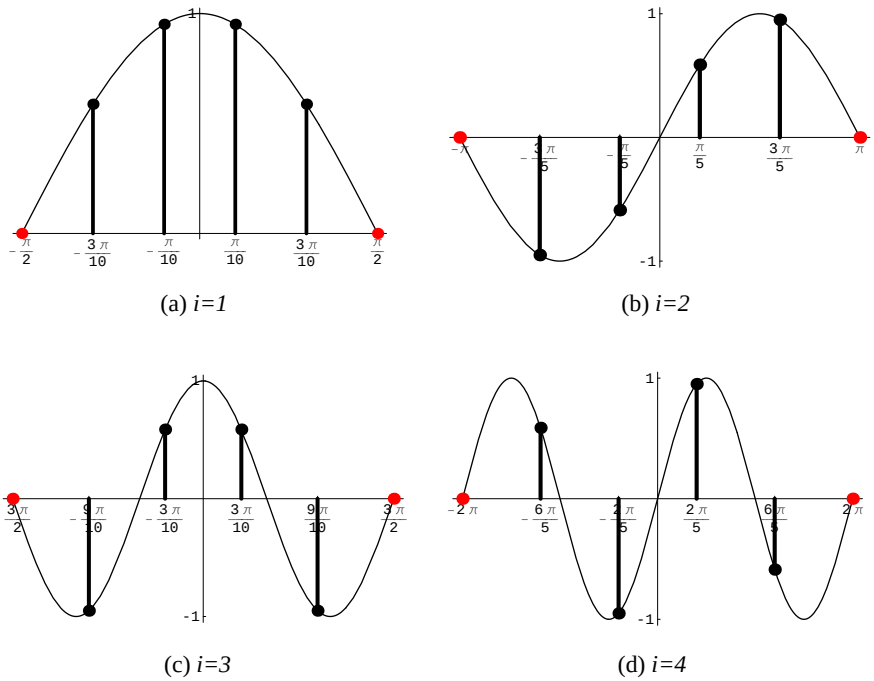
Figure 4.3: *Loop crease or corner vertex eigenvectors for wedge valence $n = 4$. The two end samples are not part of the eigenvector sequence, but instead show that the next sample outside the vector is zero.*

eigenvalue λ_i for a wedge with an even number n of triangles, i.e. an odd number of edges.

$$e = \frac{1}{8}, \bar{e} = \frac{1}{2} = 4e, d = 3e - \lambda_i$$

$$0 = \left[\begin{array}{ccc|ccccccccc} \bar{v}_0 - \lambda_i & \bar{v}_1 & \bar{v}_1 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 \\ \bar{e} & \bar{e} - \lambda_i & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \bar{e} & 0 & \bar{e} - \lambda_i & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 \\ \hline 3e & 0 & e & d & e & & & & & & & 0 \\ 3e & 0 & 0 & e & d & e & & & & & & \\ & \vdots & & & & \ddots & & & & & & \\ 3e & 0 & 0 & & & e & d & e & & & & \\ 3e & 0 & 0 & & & & e & d & e & & & \\ 3e & 0 & 0 & & & & & e & d & e & & \\ & \vdots & & & & & & & \ddots & & & \\ 3e & 0 & 0 & & & & & & e & d & e \\ 3e & e & 0 & 0 & & & & & & e & d \end{array}\right] \left[\begin{array}{c} c_0 \\ f\left[\frac{n}{2}\right] \\ f\left[-\frac{n}{2}\right] \\ \hline f\left[-\frac{n-2}{2}\right] \\ f\left[-\frac{n-4}{2}\right] \\ \vdots \\ f[-1] \\ f[0] \\ f[1] \\ \vdots \\ f\left[\frac{n-4}{2}\right] \\ f\left[\frac{n-2}{2}\right] \\ f\left[\frac{n}{2}\right] \end{array}\right] \tag{4.31}$$

Equation 4.32 defines the values of the eigenvector for a given eigenvalue λ_i for a wedge



(a) *i=1* (b) *i=2*

(c) *i=3* (d) *i=4*

Figure 4.4: *Loop crease or corner vertex eigenvectors for wedge valence $n = 5$. The two end samples are not part of the eigenvector sequence, but instead show that the next sample outside the vector is zero.*

with an odd number n of triangles, i.e. an even number of edges. Notice that the indices have been chosen on half integers, so that there is symmetry about the middle of the eigenvector.

$$e = \frac{1}{8}, \bar{e} = \frac{1}{2} = 4e, d = 3e - \lambda_i$$

$$0 = \left[\begin{array}{ccc|cccccccc} \bar{v}_0 - \lambda_i & \bar{v}_1 & \bar{v}_1 & 0 & 0 & & 0 & 0 & & 0 & 0 \\ \bar{e} & \bar{e} - \lambda_i & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \bar{e} & 0 & \bar{e} - \lambda_i & 0 & 0 & & 0 & 0 & & 0 & 0 \\ \hline 3e & 0 & e & d & e & & & & & & 0 \\ 3e & 0 & 0 & e & d & e & & & & & \\ & \vdots & & & & \ddots & & & & & \\ 3e & 0 & 0 & & & e & d & e & & & \\ 3e & 0 & 0 & & & & e & d & e & & \\ & \vdots & & & & & & & \ddots & & \\ 3e & 0 & 0 & & & & & & e & d & e \\ 3e & e & 0 & 0 & & & & & & e & d \end{array}\right] \left[\begin{array}{c} c_0 \\ f\left[\frac{n}{2}\right] \\ f\left[-\frac{n}{2}\right] \\ \hline f\left[-\frac{n-2}{2}\right] \\ f\left[-\frac{n-4}{2}\right] \\ \vdots \\ f\left[-\frac{1}{2}\right] \\ f\left[\frac{1}{2}\right] \\ \vdots \\ f\left[\frac{n-4}{2}\right] \\ f\left[\frac{n-2}{2}\right] \\ f\left[\frac{n}{2}\right] \end{array}\right] \tag{4.32}$$

The next sample at the front and end of the sequence $t = -\frac{n}{2}$ and $t = \frac{n}{2}$ are not present in

the first and last difference equations fo the T_{12} subblock. So functions that have these samples equal to zero, $f\left[-\frac{n}{2}\right] = f\left[\frac{n}{2}\right] = 0$, must be chosen from the family of solutions. $f\left[\frac{n}{2}\right] = \cos\left[\frac{n}{2}\theta_i\right] = 0$ when $\theta_i = \frac{i\pi}{n}$ for odd integers i, see the left hand columns of Figures 4.3 and 4.4. While $f\left[\frac{n}{2}\right] = \sin\left[\frac{n}{2}\theta_i\right] = 0$ when $\theta_i = \frac{i\pi}{n}$ for even integers i, see the right hand columns of Figures 4.3 and 4.4.

Because of the lower block triangular structure of the corner and crease subdivision matrices, all of the eigenvectors of the T_{12} subblock when zero extend on the top are also eigenvectors of the full subdivision matrix S. The last three eigenvectors are dependent on the subblock T_0. The eigenvalues of T_0 for the corner and crease cases take on values of $\lambda \in \left\{1, \frac{1}{2}, \frac{1}{4}\right\}$. These eigenvectors must have some of the top three elements nonzero which are all multiples of $f\left[\frac{n}{2}\right]$. Equation 4.25 shows the relationship between the eigenvalue λ and the frequency θ of the sine or cosine function. The following table shows the valences n which will have $f\left[\frac{n}{2}\right] = 0$ and can lead to a missing eigenvalue such that the matrix S is defective.

	$\theta = \cos^{-1}\left(-\frac{3e-\lambda}{2e}\right)$	$\cos\left(\frac{n}{2}\theta\right) = 0$	$\sin\left(\frac{n}{2}\theta\right) = 0$
$\lambda = \frac{1}{2} = 4e$	$\frac{\pi}{3}$	$n = 6k+3$	$n = 6k$
$\lambda = \frac{1}{4} = 2e$	$\frac{2\pi}{3}$	$n = 3k+\frac{3}{2}$	$n = 3k$

(4.33)

Both n and k must be integers, so k can be removed from these expressions by using modulo arithmetic.

	$\theta = \cos^{-1}\left(-\frac{3e-\lambda}{2e}\right)$	$\cos\left(\frac{n}{2}\theta\right) = 0$	$\sin\left(\frac{n}{2}\theta\right) = 0$
$\lambda = \frac{1}{2} = 4e$	$\frac{\pi}{3}$	$n\%6 = 3$	$n\%6 = 0$
$\lambda = \frac{1}{4} = 2e$	$\frac{2\pi}{3}$	$n\%3 = \frac{3}{2}$	$n\%3 = 0$

(4.34)

The bottom left entry demonstrates that there is never a problem with the cosine function for an eigenvalue $\lambda = \frac{1}{4}$. The defective matrix cases for corner and crease vertices will be discussed in more detail in Sections 4.7.1 and 4.7.2.

4.4 General z-Transform Solution n Even

The general difference equation corresponding to the Loop edge averaging rules is:

$$0 = f[t-2] + \alpha f[t-1] + f[t] \tag{4.35}$$

We showed that $f[t] = \cos[t\theta]$ and $f[t] = \sin[t\theta]$ were solutions if $\alpha = -2\cos[\theta]$. This is only possible if $-2 \leq \alpha \leq 2$. We can solve for a more general family of solutions using the

z-transform. The z-transform [1] is the discrete analog of the Laplace transform. The Laplace transform is useful for solving continuous differential equations, and the z-transform is useful for solving discrete difference equations.

The z-transform states that a discrete sampled time signal $f[t]$ can be represented in the z-domain as

$$f[t] \rightarrow f(z) = \sum_{t=0}^{\infty} f[t] z^{-t} \tag{4.36}$$

Using this definition, we can derive the z-transform for a shifted time signal by substituting $t \rightarrow t+m$ and manipulating the infinite sum.

$$f[t-m] \rightarrow \sum_{t=0}^{\infty} f[t-m] z^{-t} \tag{4.37}$$

$$\rightarrow \sum_{t=-m}^{\infty} f[t] z^{-t-m} \tag{4.38}$$

$$\rightarrow z^{-m} \left(\sum_{t=-m}^{-1} f[t] z^{-t} + \sum_{t=0}^{\infty} f[t] z^{-t} \right) \tag{4.39}$$

$$\rightarrow z^{-m} \left(\sum_{t=-m}^{-1} f[t] z^{-t} + f(z) \right) \tag{4.40}$$

We can convert our difference equation to the z-domain using this rule. The advantage of transforming our difference equation to the z-domain is that the many shifted versions are transformed into a single z-transformed function $f(z)$. The function $f(z)$ can be solved for as function of powers of the variable z and constant coefficient samples from the time signal $f[t]$.

In the case of our generic cosine type difference equation, the z-transformed function is formed by applying the time shift rule, and then solving for $f(z)$.

$$0 = \left(z^{-2} f(z) + z^{-1} f[-1] + f[-2] \right) + \alpha \left(z^{-1} f(z) + f[-1] \right) + f(z) \tag{4.41}$$

$$f(z) = -\frac{(f[-2] + \alpha f[-1]) + f[-1] z^{-1}}{1 + \alpha z^{-1} + z^{-2}} \tag{4.42}$$

By evaluating our difference equation at $t = 0$, we derive a useful relation between some of the samples of $f[t]$.

$$0 = f[-2] + \alpha f[-1] + f[0] \tag{4.43}$$

$$-f[0] = f[-2] + \alpha f[-1] \tag{4.44}$$

We can substitute this relation into the z-domain function to simplify it.

$$f(z) = \frac{f[0] - f[-1]z^{-1}}{1 + \alpha z^{-1} + z^{-2}} \tag{4.45}$$

We showed before that in the case where $\alpha = -2\cos(\theta)$ that both $f[t] = \cos[t\theta]$ and $f[t] = \sin[t\theta]$ are solutions to this difference equation. So we can use this formula to derive the z-transforms of both cosine and sine.

$$\cos(z\theta) = \frac{1 - \cos(\theta)z^{-1}}{1 - 2\cos(\theta)z^{-1} + z^{-2}} \tag{4.46}$$

$$\sin(z\theta) = \frac{\sin(\theta)z^{-1}}{1 - 2\cos(\theta)z^{-1} + z^{-2}} \tag{4.47}$$

The final step in using z-transforms to solve difference equations is to apply an inverse transform to $f(z)$ to get back the solution time signal $f[t]$, so it is useful to have a catalog of know functions and their z-transforms. Continuing to assume that $\alpha = -2\cos(\theta)$, we derive a general solution for any given initial conditions $f[-1]$ and $f[0]$ as a sum of cosines and sines.

$$f(z) = \frac{f[0] - f[-1]z^{-1}}{1 - 2\cos(\theta)z^{-1} + z^{-2}} \tag{4.48}$$

$$= f[0]\frac{1 - \cos(\theta)z^{-1}}{1 - 2\cos(\theta)z^{-1} + z^{-2}} + \left(\frac{\cos(\theta)f[0] - f[-1]}{\sin(\theta)}\right)\frac{\sin(\theta)z^{-1}}{1 - 2\cos(\theta)z^{-1} + z^{-2}} \tag{4.49}$$

We then apply the inverse z-transforms from Equations 4.46 and 4.46 to recover the time signal $f[t]$.

$$f[t] = f[0]\cos[t\theta] + \left(\frac{\cos(\theta)f[0] - f[-1]}{\sin(\theta)}\right)\sin[t\theta] \tag{4.50}$$

Defining different values for $f[-1]$ and $f[0]$ can give the cosine function the sine function or any linear combination of the two.

We saw that the general approach with z-transforms is to manipulate the expression for $f(z)$ into subterms that can be easily inverse transformed. For a more general solution to our difference equation, let α be any real or complex constant. We can simplify the expression for $f(z)$ by factoring the denominator using the quadratic equation.

$$f(z) = \frac{f[0] - f[-1]z^{-1}}{1 + \alpha z^{-1} + z^{-2}} \tag{4.51}$$

$$= \frac{f[0] - f[-1]z^{-1}}{\left(1 + \frac{\alpha + \sqrt{\alpha^2 - 4}}{2}z^{-1}\right)\left(1 + \frac{\alpha - \sqrt{\alpha^2 - 4}}{2}z^{-1}\right)} \tag{4.52}$$

Note that the difference of these factors has a simpler form:

$$\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}\right)-\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}\right)=\sqrt{\alpha^2-4}z^{-1} \tag{4.53}$$

The transformed function can be split into two simpler terms by multiplying by this quantity over itself.

$$f(z)=\frac{f[0]-f[-1]z^{-1}}{\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}\right)\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}\right)}\left(\frac{\sqrt{\alpha^2-4}z^{-1}}{\sqrt{\alpha^2-4}z^{-1}}\right) \tag{4.54}$$

$$=\frac{f[0]-f[-1]z^{-1}}{\sqrt{\alpha^2-4}z^{-1}}\left(\frac{\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}\right)-\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}\right)}{\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}\right)\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}\right)}\right) \tag{4.55}$$

$$=\frac{-f[-1]+f[0]z}{\sqrt{\alpha^2-4}}\left(\frac{1}{1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}}-\frac{1}{1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}}\right) \tag{4.56}$$

$$\begin{aligned}=&-\frac{f[-1]}{\sqrt{\alpha^2-4}}\left(\frac{1}{1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}}-\frac{1}{1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}}\right)\\&+\frac{f[0]}{\sqrt{\alpha^2-4}}\left(\frac{z}{1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-1}}-\frac{z}{1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-1}}\right)\end{aligned} \tag{4.57}$$

Each of these terms is a standard z-transform which corresponds to the following time signal:

$$f(z)=\frac{z^m}{1-\beta z^{-1}}\rightarrow \qquad f[t]=\beta^t u[t]\otimes\delta[t+m] \tag{4.58}$$

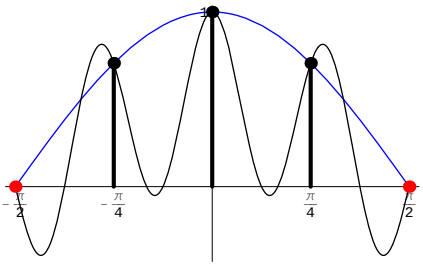
$$=\beta^{t+m}u[t+m] \tag{4.59}$$

The unit impulse $\delta[t-m]$ has a value of 1 at $t=m$ and 0 every where else. Convolving a function with an impulse shifts the other function by m. The unit step $u[t-m]$ is 0 for $t<m$ and 1 for $t\geq m$.

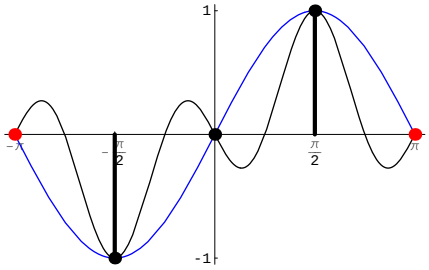
We can then perform the inverse z-transform to get the general form for the family of solutions. Note that the $u[t+1]$ can be changed to $u[t]$ because the value of the associated term at $t=-1$ is zero.

$$\begin{aligned}f[t]=&-\frac{f[-1]u[t]}{\sqrt{\alpha^2-4}}\left(\left(-\frac{\alpha-\sqrt{\alpha^2-4}}{2}\right)^n-\left(-\frac{\alpha+\sqrt{\alpha^2-4}}{2}\right)^n\right)\\&\frac{f[0]u[t+1]}{\sqrt{\alpha^2-4}}\left(\left(-\frac{\alpha-\sqrt{\alpha^2-4}}{2}\right)^{n+1}-\left(-\frac{\alpha+\sqrt{\alpha^2-4}}{2}\right)^{n+1}\right)\end{aligned} \tag{4.60}$$

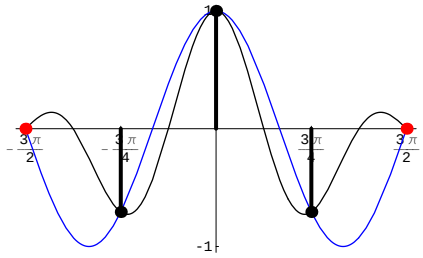
$$f[t]=\frac{(-1)^n u[t]}{\sqrt{\alpha^2-4}}\begin{pmatrix}-f[-1]\left(\left(\frac{\alpha-\sqrt{\alpha^2-4}}{2}\right)^n-\left(\frac{\alpha+\sqrt{\alpha^2-4}}{2}\right)^n\right)\\-f[0]\left(\left(\frac{\alpha-\sqrt{\alpha^2-4}}{2}\right)^{n+1}-\left(\frac{\alpha+\sqrt{\alpha^2-4}}{2}\right)^{n+1}\right)\end{pmatrix} \tag{4.61}$$



(a) *i=1*

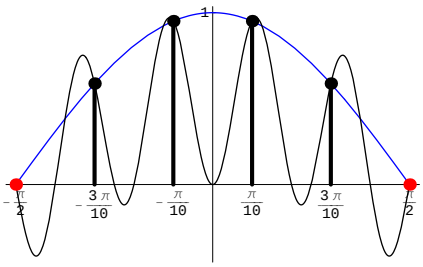


(b) *i=2*

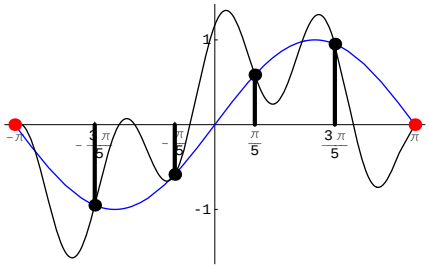


(c) *i=3*

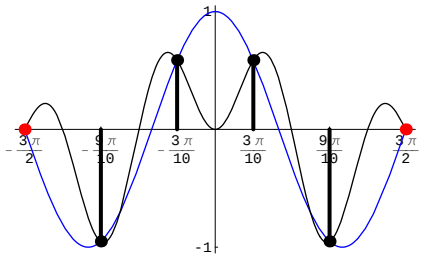
Figure 4.5: *Loop crease or corner vertex eigenvectors for valence* $n = 4$.



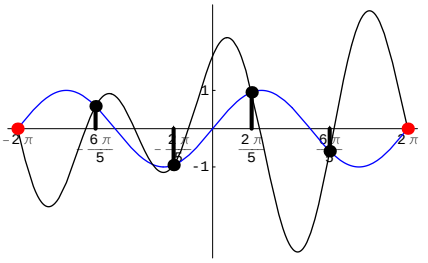
(a) *i=1*



(b) *i=2*



(c) *i=3*



(d) *i=4*

Figure 4.6: *Loop crease or corner vertex eigenvectors for valence* $n = 5$.

As with the cosine and sine functions, we are interested in both functions that have even or odd symmetry at the origin. Because the samples from the function will be used as an eigenvector, the scale of the function is not important, so we will keep $f[-1]$ as an unspecified scalar. We can then solve for $f[0]$ in terms of $f[-1]$ and $f[1]$ using the difference equation:

$$0 = f[-1] + \alpha f[0] + f[1] \tag{4.62}$$

$$f[0] = -\frac{1}{\alpha}(f[-1] + f[1]) \tag{4.63}$$

For even symmetry, we define $f[t] \equiv f[-t]$, so the general solution can be simplified.

$$f[1] \equiv f[-1] \tag{4.64}$$

$$f[0] = -\frac{2}{\alpha} f[-1] \tag{4.65}$$

$$f_{even}[t] = \frac{(-1)^n f[-1] u[t]}{2^n \alpha \sqrt{\alpha^2 - 4}} \left(\begin{array}{c} -\alpha \left(\left(\alpha - \sqrt{\alpha^2 - 4} \right)^n - \left(\alpha + \sqrt{\alpha^2 - 4} \right)^n \right) \\ + \left(\left(\alpha - \sqrt{\alpha^2 - 4} \right)^{n+1} - \left(\alpha + \sqrt{\alpha^2 - 4} \right)^{n+1} \right) \end{array} \right) \tag{4.66}$$

For odd symmetry, we define $f[t] \equiv -f[-t]$, so the general solution can be simplified.

$$f[1] = -f[-1] \tag{4.67}$$

$$f[0] = 0 \tag{4.68}$$

$$f_{odd}[t] = \frac{(-1)^{n+1} f[-1] u[t]}{2^n \sqrt{\alpha^2 - 4}} \left(\left(\alpha - \sqrt{\alpha^2 - 4} \right)^n - \left(\alpha + \sqrt{\alpha^2 - 4} \right)^n \right) \tag{4.69}$$

The eigenvectors for the crease and corner cases can be generated again using this new more general family of functions. We know from before that sine and cosine functions are solutions, so we will let $\alpha = -2\cos(\theta_i)$ where $\theta_i = \frac{i\pi}{n}$. In Figure 4.5, we graph the real projection of the complex general function in black superimposed over the real sine and cosine functions in blue. Even though the two analog solution functions are different, they coincide at the discrete sampling frequency. All of the discrete sample points are purely real values.

4.5 General z-Transform Solution n Odd

The sampling pattern for the general difference equation corresponding to the Loop edge averaging rules for valence n odd is slightly different than the even case. We want the samples to be symmetric about the origin on half integer values, so we must change the difference equation to

scale the parameter by 2 and only sample at odd integers.

$$0 = f[t-5] + \alpha f[t-3] + f[t-1] \tag{4.70}$$

We transform this equation to the z-domain using the z-transform.

$$\begin{aligned} 0 = &\left(f[-5] + f[-4]z^{-1} + f[-3]z^{-2} + f[-2]z^{-3} + f[-1]z^{-4} + z^{-5}f(z)\right) \\ &+ \alpha\left(f[-3] + f[-2]z^{-1} + f[-1]z^{-2} + z^{-3}f(z)\right) \\ &+ \left(f[-1] + z^{-1}f(z)\right) \end{aligned} \tag{4.71}$$

We then solve for the z-domain function $f(z)$.

$$f(z) = -\frac{\left(\begin{array}{c}(f[-5] + \alpha f[-3] + f[-1]) + (f[-4] + \alpha f[-2])z^{-1} \\ + (f[-3] + \alpha f[-1])z^{-2} + f[-2]z^{-3} + f[-1]z^{-4}\end{array}\right)}{z^{-1} + \alpha z^{-3} + z^{-5}} \tag{4.72}$$

To make it easier to specify our symmetry constraints, we solve for $f[-5]$ and $f[-3]$ in terms of $f[-1]$ and $f[1]$ using two instantiations of the difference equation.

$$0 = f[-5] + \alpha f[-3] + f[-1] \tag{4.73}$$

$$0 = f[-3] + \alpha f[-1] + f[1] \tag{4.74}$$

The expressions for $f[-5]$ and $f[-3]$ are then:

$$f[-5] = -(\alpha f[-3] + f[-1]) \tag{4.75}$$

$$f[-3] = -(\alpha f[-1] + f[1]) \tag{4.76}$$

Substituting these express into the z-domain function reduces the number of terms.

$$f(z) = -\frac{(f[-4] + \alpha f[-2])z^{-1} - f[1]z^{-2} + f[-2]z^{-3} + f[-1]z^{-4}}{z^{-1} + \alpha z^{-3} + z^{-5}} \tag{4.77}$$

We then factor the denominator using the quadratic equation.

$$f(z) = -\frac{(f[-4] + \alpha f[-2])z^{-1} - f[1]z^{-2} + f[-2]z^{-3} + f[-1]z^{-4}}{z^{-1} + \alpha z^{-3} + z^{-5}}\left(\frac{z}{z}\right) \tag{4.78}$$

$$= -\frac{(f[-4] + \alpha f[-2]) - f[1]z^{-1} + f[-2]z^{-2} + f[-1]z^{-3}}{1 + \alpha z^{-2} + z^{-4}} \tag{4.79}$$

$$= -\frac{(f[-4] + \alpha f[-2]) - f[1]z^{-1} + f[-2]z^{-2} + f[-1]z^{-3}}{\left(1 + \frac{\alpha + \sqrt{\alpha^2 - 4}}{2}z^{-2}\right)\left(1 + \frac{\alpha - \sqrt{\alpha^2 - 4}}{2}z^{-2}\right)} \tag{4.80}$$

Note that the difference of these factored terms is a simple expression.

$$\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-2}\right)-\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-2}\right)=\sqrt{\alpha^2-4}z^{-2} \tag{4.81}$$

Multiplying top and bottom by this difference of factors splits the fraction into two smaller fractions. We also manipulate the powers of the numerator z's to make the inverse z-transform easier.

$$f(z)=f(z)\frac{\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-2}\right)-\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-2}\right)}{\sqrt{\alpha^2-4}z^{-2}} \tag{4.82}$$

$$=-\frac{\begin{pmatrix}(f[-4]+\alpha f[-2])\\-f[1]z^{-1}\\+f[-2]z^{-2}\\+f[-1]z^{-3}\end{pmatrix}}{\sqrt{\alpha^2-4}z^{-2}}\left(\frac{1}{\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-2}\right)}-\frac{1}{\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-2}\right)}\right) \tag{4.83}$$

$$=-\frac{\begin{pmatrix}(f[-4]+\alpha f[-2])z^3\\-f[1]z^2\\+f[-2]z\\+f[-1]\end{pmatrix}}{\sqrt{\alpha^2-4}}\left(\frac{z^{-1}}{\left(1+\frac{\alpha-\sqrt{\alpha^2-4}}{2}z^{-2}\right)}-\frac{z^{-1}}{\left(1+\frac{\alpha+\sqrt{\alpha^2-4}}{2}z^{-2}\right)}\right) \tag{4.84}$$

$f(z)$ is the difference of two terms that are of the form $g(z)$. We can solve for the inverse z-transform $g[t]$ using a similar set of fraction manipulations.

$$g(z)=\frac{z^{-1}}{(1-\beta^2z^{-2})} \tag{4.85}$$

$$=\frac{z^{-1}}{(1+\beta z^{-1})(1-\beta z^{-1})} \tag{4.86}$$

$$=\frac{z^{-1}}{(1+\beta z^{-1})(1-\beta z^{-1})}\left(\frac{\left(1+\beta z^{-1}\right)-\left(1-\beta z^{-1}\right)}{2\beta z^{-1}}\right) \tag{4.87}$$

$$=\frac{1}{2\beta}\left(\frac{1}{1-\beta z^{-1}}-\frac{1}{1+\beta z^{-1}}\right) \tag{4.88}$$

$$g[t]=\frac{u[t]}{2\beta}\left(\beta^t-\beta^{-t}\right) \tag{4.89}$$

$$=\frac{u[t]}{2}\left(\beta^{t-1}-\beta^{-t-1}\right) \tag{4.90}$$

$$\tag{4.91}$$

We can now apply the inverse z-transform to find $f[t]$ where β takes on the imaginary values $\mathbf{i}\sqrt{\frac{\alpha\pm\sqrt{\alpha^2-4}}{2}}$ respectively.

$$f[t] = -\frac{1}{2\sqrt{\alpha^2-4}} \begin{pmatrix} (f[-4]+\alpha f[-2])\,\delta[t+3] \\ -f[1]\,\delta[t+2] \\ +f[-2]\,\delta[t+1] \\ +f[-1]\,\delta[t] \end{pmatrix} \otimes$$
$$u[t] \begin{pmatrix} \left(\left(\mathbf{i}\sqrt{\frac{\alpha-\sqrt{\alpha^2-4}}{2}}\right)^{t-1} - \left(\mathbf{i}\sqrt{\frac{\alpha-\sqrt{\alpha^2-4}}{2}}\right)^{-t-1}\right) \\ -\left(\left(\mathbf{i}\sqrt{\frac{\alpha+\sqrt{\alpha^2-4}}{2}}\right)^{t-1} - \left(\mathbf{i}\sqrt{\frac{\alpha+\sqrt{\alpha^2-4}}{2}}\right)^{-t-1}\right) \end{pmatrix} \tag{4.92}$$

For the even symmetry solutions, we solve for $f[-4]$ and $f[-2]$ using the following two equations:

$$f[-1] = f[1] \tag{4.93}$$
$$f[-3] = f[3] \tag{4.94}$$

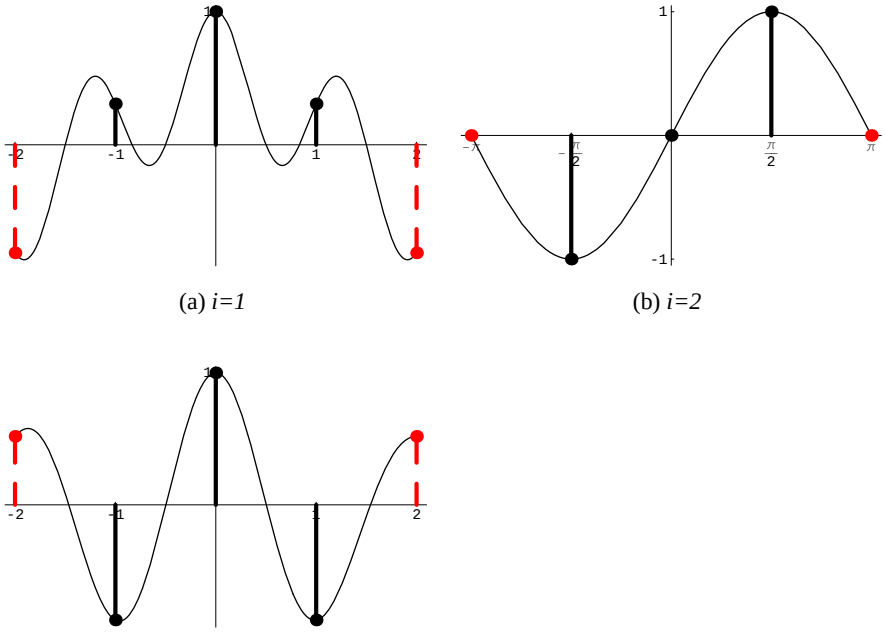
This yields the even symmetry function:

$$f[t] = \frac{\left(\mathbf{i}^{k+1}2^{-\frac{k-3}{2}} f[-1]\right)\left(\left(\sqrt{a-\sqrt{a^2-4}}\right)^k - \left(\sqrt{a+\sqrt{a^2-4}}\right)^k\right)}{\left(\sqrt{a+2}+\sqrt{a-2}\right)\sqrt{a-2}\sqrt{a-\sqrt{a^2-4}}} \tag{4.95}$$

As with the even valence case, the eigenvectors for the crease and corner cases can be generated again using this new more general family of functions for the odd valence case. We know from before that sine and cosine functions are solutions, so we will let $\alpha = -2\cos(\theta)$ where $\theta_i = \frac{i\pi}{n}$. In Figure 4.6, we graph the samples from the general function in black superimposed on the sine and cosine functions in blue. Even though the two analog solutions functions are different, they coincide at the discrete sampling frequency. The general function takes on imaginary values, so the graphs just show the projection onto the real plane. The function is purely real at the sample points, because it is consistent with the real valued sine and cosine for the chosen α.

4.6 Loop Dart Eigenvectors

In the dart vertex case, the eigenvectors associated with the edge averaging rules are coupled to the vertex averaging rules. The T_{12} subblock of edge averaging rules from the subdivision



(a) *i=1* (b) *i=2*

(c) *i=3*

Figure 4.7: *Loop dart vertex eigenvectors for valence* $n = 4$*. The two end samples are not part of the eigenvector sequence.*

matrix S is exactly the same as in the crease and corner cases. The difference is appearant in the T_{02} subblock because the v_1 averaging coefficients in the crease and corner case are all zeros. The

dart subdivision matrix when the valence n is even is:

$$e = \frac{1}{8}, v_0 = 1 - nv_1, d_0 = v_0 - \lambda_i, d_1 = 4e - \lambda_i, d = 3e - \lambda_i$$

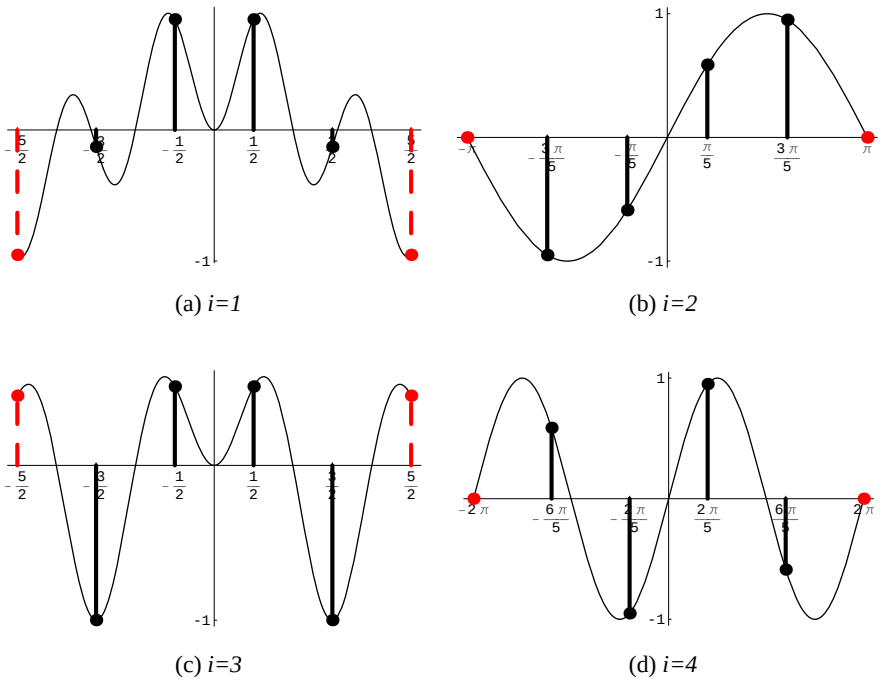
$$0 = \left[\begin{array}{cc|ccccccccc} d_0 & v_1 & v_1 & v_1 & & v_1 & v_1 & v_1 & & v_1 & v_1 \\ 4e & d_1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \hline 3e & e & d & e & & & & & & & 0 \\ e & 0 & e & d & e & & & & & & \\ & \vdots & & & \ddots & & & & & & \\ e & 0 & & & e & d & e & & & & \\ e & 0 & & & & e & d & e & & & \\ e & 0 & & & & & e & d & e & & \\ & \vdots & & & & & & & \ddots & & \\ e & 0 & & & & & & & e & d & e \\ 3e & e & 0 & & & & & & & e & d \end{array}\right] \left[\begin{array}{c} c_0 \\ b_{\frac{n}{2}} \\ \hline b_{-\frac{n-2}{2}} \\ b_{-\frac{n-4}{2}} \\ \vdots \\ b_{-1} \\ b_0 \\ b_1 \\ \vdots \\ b_{\frac{n-4}{2}} \\ b_{\frac{n-2}{2}} \\ b_{\frac{n}{2}} \end{array}\right] \tag{4.96}$$

The dart subdivision matrix when the valence n is odd is:

$$e = \frac{1}{8}, v_0 = 1 - nv_1, d_0 = v_0 - \lambda_i, d_1 = 4e - \lambda_i, d = 3e - \lambda_i$$

$$0 = \left[\begin{array}{cc|ccccccccc} d_0 & v_1 & v_1 & v_1 & & v_1 & v_1 & & v_1 & v_1 \\ 4e & d_1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \hline 3e & e & d & e & & & & & & 0 \\ e & 0 & e & d & e & & & & & \\ & \vdots & & & \ddots & & & & & \\ e & 0 & & & e & d & e & & & \\ e & 0 & & & & e & d & e & & \\ & \vdots & & & & & & \ddots & & \\ e & 0 & & & & & & e & d & e \\ 3e & e & 0 & & & & & & e & d \end{array}\right] \left[\begin{array}{c} c_0 \\ b_{\frac{n}{2}} \\ \hline b_{-\frac{n-2}{2}} \\ b_{-\frac{n-4}{2}} \\ \vdots \\ b_{-\frac{1}{2}} \\ b_{\frac{1}{2}} \\ \vdots \\ b_{\frac{n-4}{2}} \\ b_{\frac{n-2}{2}} \\ b_{\frac{n}{2}} \end{array}\right] \tag{4.97}$$

In the crease and corner cases, the sine and cosine based T_{12} subblock eigenvectors are also eigenvectors of the entire matrix when extended at the top with zeros because the matrix is lower block triangular. The vertex averaging rule from the dart case makes it more complicated.



(a) *i=1* (b) *i=2*

(c) *i=3* (d) *i=4*

Figure 4.8: *Loop dart vertex eigenvectors for valence* $n = 5$*. The two end samples are not part of the eigenvector sequence.*

The odd symmetry sine based eigenvectors are the same because they sum to zero, so when they are multiplied against the top row vector of all v_1's the result is zero. But this leaves $\left\lfloor \frac{n+4}{2} \right\rfloor$ even symmetry eigenvectors undetermined. The problem with eigenvectors from the crease and corner cases, which had cosine sequences that terminated with zeros on the ends, is that multiplication with the top vertex rule row will create a nonzero value. It is not possible to make c_0 nonzero while keeping $b_{\frac{n}{2}}$ zero because of the second row.

The same difference equation is still present, so a cosine-like sequence must be used. The difference is that $b_{\frac{n}{2}}$, which contains the value for both ends of the sequence, must be nonzero. Fortunately in the even symmetry case, $b_{\frac{n}{2}} = b_{-\frac{n}{2}}$, so we just need to adjust θ_i. Examining one of the middle rows, we see that c_0 which must be nonzero is multiplied by the nonzero value e, so the difference equation must sum to a nonzero quantity. This can be accomplished by adding a constant c_1 to every term in the the sequence.

$$b_t = c_1 + f\left[t\right] \tag{4.98}$$

Using the sharp edge rule from row 2 and any of the smooth edge rule rows, we can solve

for c_1 and c_0 in terms of the eigenvalue λ_i and $f\left[\frac{n}{2}\right]$.

$$0 = 4ec_0 + (4e - \lambda_i)\left(c_1 + f\left[\frac{n}{2}\right]\right) \tag{4.99}$$

$$0 = 3ec_0 + (5e - \lambda_i)\, c_1 \tag{4.100}$$

The expressions for c_1 and c_0 are:

$$c_1 = \frac{3\,(4e - \lambda_i)}{(8e - \lambda_i)} f\left[\frac{n}{2}\right] \tag{4.101}$$

$$c_0 = -\frac{5e - \lambda_i}{3e} c_1 = \frac{(5e - \lambda_i)\,(4e - \lambda_i)}{e\,(8e - \lambda_i)} f\left[\frac{n}{2}\right] \tag{4.102}$$

All of the elements of the eigenvector are now multiples of $f[-1]$ which can be divided out, so the top vertex averaging row can be used to create a lower degree characteristic polynomial with the sine based eigenvalue factors removed.

$$0 = (v_0 - \lambda_i)\, c_0 + v_1 \left(nc_1 + f\left[\frac{n}{2}\right] + f[0] + 2 \sum_{k=1}^{\left\lceil \frac{n-2}{2} \right\rceil} f[k] \right) \tag{4.103}$$

We do not currently have a closed form solution for these eigenvalues, so we use a numerical solver to get the eigenvalues and use them to generate the eigenvectors. Some of the eigenvalues are outside the range in which a cosine function is possible, so we use the general solution form instead. Figure 4.7 shows the eigenvectors for an even valence case, while Figure 4.8 shows the eigenvectors for an odd valence case. The left hand column has eigenvectors with even symmetry, while the right hand column has the same sine based eigenvectors from the crease and corner case.

4.7 Loop Crease and Corner Jordan Decompositions

We have seen how to use the z-transform as a general tool for solving difference equations, so let us return to the crease and corner cases where for certain valences n the subdivision matrix S is defective and use the z-transform to solve for the remaining vectors. When the matrix S is defective, there is an eigenvalue λ with multiplicity $m > 1$ where less than m linearly independent eigenvectors exist. In this case, a generalized eigenvector, also called a Jordan vector, must be used in the decomposition. A Jordan vector v_J for the eigenvalue λ satisfies the generalized eigenvalue equation where v_E is an eigenvector of S associated with the same eigenvalue λ.

$$(S - \lambda I)\, v_J = v_E \tag{4.104}$$

This affects the difference equation for an undetermined signal $g[t]$ by setting it equal to an existing signal $f[t]$ for the known eigenvector instead of zero.

$$g[t-2] + \alpha g[t-1] + g[t] = f[t-1] \tag{4.105}$$

We will now solve for these Jordan vectors for the defective corner and crease cases.

4.7.1 Loop Corner Jordan Decomposition

In the corner vertex case, the subdivision matrix S is lower block triangular, so the eigenvalues of the diagonal subblocks are also eigenvalues of the entire matrix. The T_0 subblock is of the form:

$$T_0 = \begin{bmatrix} 8e & 0 & 0 \\ 4e & 4e & 0 \\ 4e & 0 & 4e \end{bmatrix} \tag{4.106}$$

The eigen-decomposition of this block is:

$$T_0 V_0 = V_0 \Lambda_0 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & b_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & -b_{\frac{n}{2}} & b_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4e & 0 \\ 0 & 0 & 4e \end{bmatrix} \tag{4.107}$$

The first eigenvector, associated with the eigenvalue $\lambda = 1$, is all ones due to the sum to one property of the averaging rules. We need two eigenvectors where $b_{-\frac{n}{2}} \neq 0$ and $b_{\frac{n}{2}} \neq 0$, so that they will be linearly independent of the eigenvectors of subblock T_{12}. Equation 4.34 shows that when $\lambda = \frac{1}{2}$ and n is divisible by 3, either the sine or cosine sequence will terminate with a zero value. In this case, the matrix is defective, and a nontrivial Jordan block will have to be used.

When the matrix is defective, we solve the generalized eigenvalue equation which is similar to Equation 4.31 with eigenvector signal $f[t]$ replaced by $g[t]$ and the zero vector replaced by $f[t]$. Taking $\lambda = \frac{1}{2} = 4e$ from subblock T_0, we get the following difference equation.

$$eg[t-2] + (3e - \lambda) g[t-1] + eg[t] = f[t-1] \tag{4.108}$$

$$eg[t-2] - eg[t-1] + eg[t] = f[t-1] \tag{4.109}$$

$$g[t-2] - g[t-1] + g[t] = \frac{1}{e} f[t-1] \tag{4.110}$$

$$\tag{4.111}$$

Remember the cosine difference equation, the left hand side is like the cosine function with $\theta = \frac{\pi}{3}$.

$$g[t-2] - 2\cos\left(\frac{\pi}{3}\right) g[t-1] + g[t] = \frac{1}{e} f[t-1] \tag{4.112}$$

Substituting $t = 0$ into the difference gives a relationship for $g[-2]$.

$$g[-2] - g[-1] + g[0] = \frac{1}{e} f[-1] \tag{4.113}$$

$$g[-2] = \frac{1}{e} f[-1] + g[-1] - g[0] \tag{4.114}$$

We now move to the z-domain applying the z-transform to both sides of the equation.

$$\left(g[-2] + z^{-1} g[-1] + z^{-2} g(z)\right) - \left(g[-1] + z^{-1} g(z)\right) + g(z) = \frac{1}{e}\left(f[-1] + z^{-1} f(z)\right) \tag{4.115}$$

$$\left(1 - z^{-1} + z^{-2}\right) g(z) + \left(g[-2] - g[-1] + z^{-1} g[-1]\right) = \frac{1}{e} f[-1] + \frac{1}{e} z^{-1} f(z) \tag{4.116}$$

We solve for $g(z)$ and simplify by substituting for $g[-2]$.

$$\left(1 - z^{-1} + z^{-2}\right) g(z) = -\left(g[-2] - g[-1] - \frac{1}{e} f[-1] + z^{-1} g[-1]\right) + \frac{1}{e} z^{-1} f(z) \tag{4.117}$$

$$\left(1 - z^{-1} + z^{-2}\right) g(z) = -\left(-g[0] + z^{-1} g[-1]\right) + \frac{1}{e} z^{-1} f(z) \tag{4.118}$$

We find that $g(z)$ is made up of cosine and sine z-transform terms.

$$g(z) = \frac{g[0] - g[-1] z^{-1}}{1 - z^{-1} + z^{-2}} + \frac{z^{-1} f(z)}{e\left(1 - z^{-1} + z^{-2}\right)} \tag{4.119}$$

$$g(z) = \left(\begin{array}{c} g[0] \left(\frac{1 - \cos\left(\frac{\pi}{3}\right) z^{-1}}{1 - z^{-1} + z^{-2}} \right) + \frac{g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]}{\sin\left(\frac{\pi}{3}\right)} \left(\frac{\sin\left(\frac{\pi}{3}\right) z^{-1}}{1 - z^{-1} + z^{-2}} \right) \\ + \frac{1}{e\sin\left(\frac{\pi}{3}\right)} \left(\frac{\sin\left(\frac{\pi}{3}\right) z^{-1}}{1 - z^{-1} + z^{-2}} \right) f(z) \end{array} \right) \tag{4.120}$$

We can now apply the inverse z-transform. The third term is a sine function multiplied by the $f(z)$. We need to apply the property that multiplication in the z-domain is equivalent to convolution in the time domain and vice versa.

$$g[t] = \left(\begin{array}{c} g[0] \cos\left[t\frac{\pi}{3}\right] u[t] + \frac{g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]}{\sin\left(\frac{\pi}{3}\right)} \sin\left[t\frac{\pi}{3}\right] u[t] \\ + \frac{1}{e\sin\left(\frac{\pi}{3}\right)} \left(\sin\left[t\frac{\pi}{3}\right] u[t]\right) \otimes \left(f[t]\, u[t]\right) \end{array} \right) \tag{4.121}$$

$$\tag{4.122}$$

The discrete convolution of two functions $a[t]$ and $b[t]$ is defined as:

$$a[t] \otimes b[t] = \sum_{k=-\infty}^{\infty} a[k]\, b[t-k] \tag{4.123}$$

We substitute this definition into our equation and apply two simplifications based on zero values of the unit step function to make the sum finite.

$$g[t] = \frac{1}{e\sin\left(\frac{\pi}{3}\right)} \left(\begin{array}{c} e\sin\left(\frac{\pi}{3}\right) g[0] \cos\left[t\frac{\pi}{3}\right] u[t] + e\left(g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]\right) \sin\left[t\frac{\pi}{3}\right] u[t] \\ + \sum_{k=\infty}^{\infty} \sin\left[k\frac{\pi}{3}\right] u[k]\, f[t-k]\, u[t-k] \end{array} \right) \tag{4.124}$$

$$= \frac{1}{e\sin\left(\frac{\pi}{3}\right)} \left(\begin{array}{c} e\sin\left(\frac{\pi}{3}\right) g[0] \cos\left[t\frac{\pi}{3}\right] u[t] + e\left(g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]\right) \sin\left[t\frac{\pi}{3}\right] u[t] \\ + \sum_{k=\infty}^{t} \sin\left[k\frac{\pi}{3}\right] u[k]\, f[t-k] \end{array} \right) \tag{4.125}$$

$$= \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)} \left(\begin{array}{c} e\sin\left(\frac{\pi}{3}\right) g[0] \cos\left[t\frac{\pi}{3}\right] + e\left(g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]\right) \sin\left[t\frac{\pi}{3}\right] \\ + \sum_{k=0}^{t} \sin\left[k\frac{\pi}{3}\right] f[t-k] \end{array} \right) \tag{4.126}$$

Odd Symmetry

When the wedge valence n is even and divisible by 3, a Jordan vector coupled with the odd symmetry eigenvector solution $f[t] = \sin\left[t\frac{\pi}{3}\right]$ is required. Evaluating the difference equation gives some simplifying substitutions.

$$f[t] = \sin\left[t\frac{\pi}{3}\right] \tag{4.127}$$

$$f[0] = 0 \tag{4.128}$$

$$g[1] = -g[-1] \tag{4.129}$$

$$\frac{1}{e} f[0] = g[-1] - g[0] + g[1] \tag{4.130}$$

$$g[0] = 0 \tag{4.131}$$

$$\tag{4.132}$$

We substitute $f[t] = \sin\left[t\frac{\pi}{3}\right]$ into the difference equation and apply some trigonometric substitutions to simplify the summation.

$$g[t] = \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)}\left(\begin{array}{c} e\sin\left(\frac{\pi}{3}\right)g[0]\cos\left[t\frac{\pi}{3}\right] + e\left(g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]\right)\sin\left[t\frac{\pi}{3}\right] \\ +\sum_{k=0}^{t}\sin\left[k\frac{\pi}{3}\right]f[t-k] \end{array}\right) \tag{4.133}$$

$$g[t] = \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)}\left(eg[1]\sin\left[t\frac{\pi}{3}\right] + \sum_{k=0}^{t}\sin\left[k\frac{\pi}{3}\right]\sin\left[(t-k)\frac{\pi}{3}\right]\right) \tag{4.134}$$

$$g[t] = \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)}\left(eg[1]\sin\left[t\frac{\pi}{3}\right] + \sum_{k=0}^{t}\frac{1}{2}\left(\cos\left[(2k-t)\frac{\pi}{3}\right] - \cos\left[t\frac{\pi}{3}\right]\right)\right) \tag{4.135}$$

$$g[t] = \frac{u[t]}{2e\sin\left(\frac{\pi}{3}\right)}\left(2eg[1]\sin\left[t\frac{\pi}{3}\right] - (t+1)\cos\left[t\frac{\pi}{3}\right] + \sum_{k=0}^{t}\cos\left[(2k-t)\frac{\pi}{3}\right]\right) \tag{4.136}$$

$$g[t] = \frac{u[t]}{2e\sin\left(\frac{\pi}{3}\right)}\left(2eg[1]\sin\left[t\frac{\pi}{3}\right] - (t+1)\cos\left[t\frac{\pi}{3}\right] + \sum_{k=0}^{t}\cos\left[(2k-t)\frac{\pi}{3}\right]\right) \tag{4.137}$$

$$\tag{4.138}$$

The term $\cos\left[(2k-t)\frac{\pi}{3}\right]$ is periodic, and each of these cycles of three consecutive integer k's sum to zero. The sum can then be reduce to the remainder of $\frac{t}{3}$ terms which is at most two terms.

$$g[t] = \frac{u[t]}{2e\sin\left(\frac{\pi}{3}\right)}\left(2eg[1]\sin\left[t\frac{\pi}{3}\right] - (t+1)\cos\left[t\frac{\pi}{3}\right] + \sum_{k=3\left\lfloor\frac{t}{3}\right\rfloor}^{t}\cos\left[(2k-t)\frac{\pi}{3}\right]\right) \tag{4.139}$$

Even Symmetry

When the wedge valence n is odd and divisible by 3, a Jordan vector coupled with the even symmetry eigenvector solution $f[t] = \cos\left[t\frac{\pi}{3}\right]$ is required. Evaluating the difference equation gives some simplifying substitutions.

$$f[t] = \cos\left[t\frac{\pi}{3}\right] \tag{4.140}$$

$$f[0] = 1 \tag{4.141}$$

$$g[1] = g[-1] \tag{4.142}$$

$$\frac{1}{e}f[0] = g[-1] - g[0] + g[1] \tag{4.143}$$

$$g[0] = 2g[1] - \frac{1}{e} \tag{4.144}$$

$$\tag{4.145}$$

We substitute $f[t] = \cos\left[t\frac{\pi}{3}\right]$ into the difference equation and apply some trigonometric substitutions to simplify the summation.

$$g[t] = \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)}\left(\begin{array}{c} e\sin\left(\frac{\pi}{3}\right)g[0]\cos\left[t\frac{\pi}{3}\right] + e\left(g[0]\cos\left(\frac{\pi}{3}\right) - g[-1]\right)\sin\left[t\frac{\pi}{3}\right] \\ +\sum_{k=0}^{t}\sin\left[k\frac{\pi}{3}\right]f[t-k] \end{array}\right) \tag{4.146}$$

$$g[t] = \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)}\left(\begin{array}{c} 2e\sin\left(\frac{\pi}{3}\right)g[1]\cos\left[t\frac{\pi}{3}\right] - \sin\left(\frac{\pi}{3}\right)\cos\left[t\frac{\pi}{3}\right] - \cos\left(\frac{\pi}{3}\right)\sin\left[t\frac{\pi}{3}\right] \\ +\sum_{k=0}^{t}\sin\left[k\frac{\pi}{3}\right]\cos\left[(t-k)\frac{\pi}{3}\right] \end{array}\right) \tag{4.147}$$

$$g[t] = \frac{u[t]}{e\sin\left(\frac{\pi}{3}\right)}\left(\begin{array}{c} 2e\sin\left(\frac{\pi}{3}\right)g[1]\cos\left[t\frac{\pi}{3}\right] - \sin\left[(t+1)\frac{\pi}{3}\right] \\ +\sum_{k=0}^{t}\frac{1}{2}\left(\sin\left[(2k-t)\frac{\pi}{3}\right] + \sin\left[t\frac{\pi}{3}\right]\right) \end{array}\right) \tag{4.148}$$

$$g[t] = \frac{u[t]}{2e\sin\left(\frac{\pi}{3}\right)}\left(\begin{array}{c} 4e\sin\left(\frac{\pi}{3}\right)g[1]\cos\left[t\frac{\pi}{3}\right] - 2\sin\left[(t+1)\frac{\pi}{3}\right] + (t+1)\sin\left[t\frac{\pi}{3}\right] \\ +\sum_{k=0}^{t}\sin\left[(2k-t)\frac{\pi}{3}\right] \end{array}\right) \tag{4.149}$$

The term $\sin\left[(2k-t)\frac{\pi}{3}\right]$ is periodic, and each of these cycles of three consecutive integer k's sum to zero. The sum can then be reduce to the remainder of $\frac{t}{3}$ terms which is at most two terms.

$$g[t] = \frac{u[t]}{2e\sin\left(\frac{\pi}{3}\right)}\left(2\left(2eg[1]-1\right)\sin\left(\frac{\pi}{3}\right)\cos\left[t\frac{\pi}{3}\right] + t\sin\left[t\frac{\pi}{3}\right] + \sum_{k=3\left\lfloor\frac{t}{3}\right\rfloor}^{t}\sin\left[(2k-t)\frac{\pi}{3}\right]\right) \tag{4.150}$$

The solution we derived is for integer samples, but when the wedge valence n is odd, we choose to sample at half integer steps. For the smooth sine and cosine terms this time scaling works, but the summation term needs to be modified. The summation term is simply a periodic correction term, so we scale our half integers by two by substituting $t \to 2t$ into the summation only.

$$g[t] = \frac{u[t]}{2e\sin\left(\frac{\pi}{3}\right)}\left(2\left(2eg[1]-1\right)\sin\left(\frac{\pi}{3}\right)\cos\left[t\frac{\pi}{3}\right] + t\sin\left[t\frac{\pi}{3}\right] + \sum_{k=3\left\lfloor\frac{2t}{3}\right\rfloor}^{2t}\sin\left[(2k-2t)\frac{\pi}{3}\right]\right) \tag{4.151}$$

4.7.2 Loop Crease Jordan Decomposition

In the crease vertex case, the subdivision matrix S is lower block triangular, so the eigenvalues of the diagonal subblocks are also eigenvalues of the entire matrix. The T_0 subblock is of the form:

$$T_0 = \begin{bmatrix} 6e & e & e \\ 4e & 4e & 0 \\ 4e & 0 & 4e \end{bmatrix} \tag{4.152}$$

The eigen-decomposition of this block is:

$$T_0 V_0 = V_0 \Lambda_0 = \begin{bmatrix} 1 & 0 & -\frac{1}{2} b_{\frac{n}{2}} \\ 1 & b_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & -b_{\frac{n}{2}} & b_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4e & 0 \\ 0 & 0 & 2e \end{bmatrix} \tag{4.153}$$

The first eigenvector, associated with the eigenvalue $\lambda = 1$, is all ones due to the sum to one property of the averaging rules. When $\lambda = \frac{1}{2} = 4e$, the eigenvector must have odd symmetry, so the matrix is defective if $n\%6 = 3$, Equation 4.34. The Jordan vector in this case is the same as the corner case Equation 4.139. The eigenvalue $\lambda = \frac{1}{4} = 2e$ never causes the matrix to be defective.

4.8 Catmull-Clark Eigenvectors

The eigenvectors associated with the Catmull-Clark face and edge averaging rules follow a very similar set of difference equations to that of the edge rules of Loop subdivision. In the Catmull-Clark case, the face rules have a repeated set of three coefficients along the diagonal interleaved with the edge rules which have a repeated set of five coefficients as shown in Equation 4.154. We assume that the eigenvector contains samples from two different signals on alternating elements.

$$e = \frac{1}{16}, f = \frac{1}{4} = 4e, d_e = 6e - \lambda_i, d_f = f - \lambda_i = 4e - \lambda_i$$

$$0 = \begin{bmatrix} \ddots & & & & & & & \\ e & e & d_e & e & e & & & \\ & & f & d_f & f & & & \\ & & e & e & d_e & e & e & \\ & & & & f & d_f & f & \\ & & & & e & e & d_e & e \\ & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ g[-1] \\ h[0] \\ g[0] \\ h[1] \\ g[1] \\ \vdots \end{bmatrix} \tag{4.154}$$

Looking at a window of three consecutive rows, we have the following system of three difference equations over two different signals $g[t]$ and $h[t]$.

$$\begin{aligned} 0 = \quad & 4eg[t-2] + \quad (4e - \lambda_i) h[t-1] + \quad 4eg[t-1] \\ 0 = \quad & eg[t-2] + \quad eh[t-1] + \quad (6e - \lambda_i) g[t-1] \quad + \quad eh[t] + \quad eg[t] \\ 0 = \quad & \quad 4eg[t-1] \quad + \quad (4e - \lambda_i) h[t] + \quad 4eg[t] \end{aligned} \tag{4.155}$$

The bottom equation gives a formula for $h[t]$ in terms of $g[t]$:

$$h[t] = -\frac{4e}{4e-\lambda_i}\left(g[t-1]+g[t]\right) \tag{4.156}$$

We subtract the face rule equations multiplied by e from the edge rule equation multiplied by $(4e-\lambda_i)$ to form a difference equation for $g[t]$.

$$0 = -e\lambda_i g[t-2] + \left((4e-\lambda_i)(6e-\lambda_i) - 8e^2\right) g[t-1] - e\lambda_i g[t] \tag{4.157}$$

$$0 = g[t-2] - \frac{16e^2-10e\lambda_i+\lambda_i^2}{e\lambda_i} g[t-1] + g[t] \tag{4.158}$$

This is the cosine or sine difference equation that we saw before if the middle coefficient has the correct form. We can also solve for the relationship between the eigenvalue λ_i and the frequency of the sines and cosines θ_i.

$$-2\cos(\theta_i) = -\frac{16e^2-10e\lambda_i+\lambda_i^2}{e\lambda_i} \tag{4.159}$$

$$\cos(\theta_i) = \frac{16e^2-10e\lambda_i+\lambda_i^2}{2e\lambda_i} \tag{4.160}$$

$$0 = \lambda_i^2 - 2e(5+\cos(\theta_i))\lambda_i + 16e^2 \tag{4.161}$$

$$\lambda_i = e\left(5+\cos(\theta_i) \pm \cos\left(\frac{\theta_i}{2}\right)\sqrt{2(9+\cos(\theta_i))}\right) \tag{4.162}$$

So $g[t]$ is either a cosine or a sine function, and $h[t]$ is a scaled sum of two of the same type of function:

$$g[t] = \cos[t\theta_i], \qquad h[t] = -\frac{4e}{4e-\lambda_i}\left(\cos[(t-1)\theta_i] + \cos[t\theta_i]\right) \tag{4.163}$$

$$g[t] = \sin[t\theta_i], \qquad h[t] = -\frac{4e}{4e-\lambda_i}\left(\sin[(t-1)\theta_i] + \sin[t\theta_i]\right) \tag{4.164}$$

4.8.1 Catmull-Clark Smooth and Spike Eigenvectors

The Catmull-Clark face and edge averaging rules in the smooth and spike vertex cases wrap around in a cyclic manner, Equation 4.165. The edge and face rules are in alternating rows, so the elements of the eigenvector for a vertex of valence n are alternating pairs of $g[t]$ and $h[t]$ for

$t \in [0, n-1]$. Equation 4.165 defines the values of the eigenvector for a given eigenvalue λ_i.

$$e = \frac{1}{16}, f = \frac{1}{4} = 4e, d_e = 6e - \lambda_i, d_f = f - \lambda_i = 4e - \lambda_i$$

$$0 = \begin{bmatrix} d_e & e & e & & & & & e & e \\ f & d_f & f & & & & & & \\ e & e & d_e & e & e & & & & \\ & & f & d_f & f & & & & \\ & & & & \ddots & & & & \\ & & & e & e & d_e & e & e & \\ & & & & & f & d_f & f & \\ e & & & & & e & e & d_e & e \\ f & & & & & & & f & d_f \end{bmatrix} \begin{bmatrix} \boxed{g[n-1]} \\ \boxed{h[n-1]} \\ g[0] \\ h[0] \\ g[1] \\ h[1] \\ \vdots \\ g[n-2] \\ h[n-2] \\ g[n-1] \\ h[n-1] \\ \boxed{g[0]} \end{bmatrix} \tag{4.165}$$

As indicated by the highlighted extra elements, the cyclic nature of the matrix shows that $g[n] = g[0]$, $g[-1] = g[n-1]$, and $h[-1] = h[n-1]$. $g[t]$ must then be periodic in n, so $\theta_i = \frac{2\pi i}{n}$. The pairs of functions in Equations 4.166 and 4.167 for $i \in \left\{1, \left\lfloor \frac{n-1}{2} \right\rfloor\right\}$ define linearly independent eigenvectors for even and odd symmetries respectively. Note that we have applied a shift of $\delta[t+1]$ to the $h[t]$ function to make the indices in the eigenvector nicer.

$$g[t] = \cos\left[t\frac{2\pi i}{n}\right], \qquad h[t] = -\frac{4e}{4e-\lambda_i}\left(\cos\left[t\frac{2\pi i}{n}\right] + \cos\left[(t+1)\frac{2\pi i}{n}\right]\right) \tag{4.166}$$

$$g[t] = \sin\left[t\frac{2\pi i}{n}\right], \qquad h[t] = -\frac{4e}{4e-\lambda_i}\left(\sin\left[t\frac{2\pi i}{n}\right] + \sin\left[(t+1)\frac{2\pi i}{n}\right]\right) \tag{4.167}$$

If n is even, then an additional cosine eigenvector with $i = \frac{n}{2}$ exists, while the corresponding sine function is all zeros. The final eigenvector in both cases is all ones which corresponds to $i = n$.

4.8.2 Catmull-Clark Crease and Corner Eigenvectors

The CatmullClark edge averaging rules for a wedge in the crease and corner vertex cases get clipped at the sharp edge boundary, indicated by the zeros in the upper right and lower left corners of the T_{12} subblock in Equations 4.168 and 4.169. The elements of the eigenvector for a wedge of valence n are $g[t]$ and $h[t]$ for $t \in \left[-\frac{n-2}{2}, \frac{n-2}{2}\right]$.

Equation 4.168 defines the values of the eigenvector for a given eigenvalue λ_i for a wedge with an even number n of quads, i.e. an odd number of edges.

$$e = \frac{1}{16}, f = \frac{1}{4} = 4e, \bar{v}_0 = \frac{6}{8} = 12e, \bar{v}_1 = \frac{1}{8} = 2e, \bar{e} = \frac{1}{2} = 8e$$
$$d_e = 6e - \lambda_i, d_f = f - \lambda_i = 4e - \lambda_i, d_{\bar{v}} = \bar{v}_0 - \lambda_i, d_{\bar{e}} = \bar{e} - \lambda_i$$

$$0 = \left[\begin{array}{ccc|ccccccccccc}
d_{\bar{v}} & \bar{v}_1 & \bar{v}_1 & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 \\
\bar{e} & d_{\bar{e}} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\bar{e} & 0 & d_{\bar{e}} & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 \\
\hline
f & 0 & f & d_f & f & & & & & & & & & 0 \\
6e & 0 & e & e & d_e & e & e & & & & & & & 0 \\
f & 0 & 0 & & f & d_f & f & & & & & & & \\
 & \vdots & & & & & \ddots & & & & & & & \\
f & 0 & 0 & & & & f & d_f & f & & & & & \\
6e & 0 & 0 & & & & e & e & d_e & e & e & & & \\
f & 0 & 0 & & & & & & f & d_f & f & & & \\
 & \vdots & & & & & & & & & \ddots & & & \\
f & 0 & 0 & & & & & & & & f & d_f & f & \\
6e & e & 0 & 0 & & & & & & & e & e & d_e & e \\
f & f & 0 & 0 & & & & & & & & & f & d_f
\end{array}\right]
\left[\begin{array}{c}
c_0 \\
g\left[\frac{n}{2}\right] \\
g\left[-\frac{n}{2}\right] \\
\hline
h\left[-\frac{n-1}{2}\right] \\
g\left[-\frac{n-2}{2}\right] \\
h\left[-\frac{n-3}{2}\right] \\
\vdots \\
h\left[-\frac{1}{2}\right] \\
g\left[0\right] \\
h\left[\frac{1}{2}\right] \\
\vdots \\
h\left[\frac{n-3}{2}\right] \\
g\left[\frac{n-2}{2}\right] \\
h\left[\frac{n-1}{2}\right] \\
g\left[\frac{n}{2}\right]
\end{array}\right] \tag{4.168}$$

Equation 4.169 defines the values of the eigenvector for a given eigenvalue λ_i for a wedge

with an odd number n of quads, i.e. an even number of edges.

$$e = \frac{1}{16}, f = \frac{1}{4} = 4e, \bar{v}_0 = \frac{6}{8} = 12e, \bar{v}_1 = \frac{1}{8} = 2e, \bar{e} = \frac{1}{2} = 8e$$

$$d_e = 6e - \lambda_i, d_f = f - \lambda_i = 4e - \lambda_i, d_{\bar{v}} = \bar{v}_0 - \lambda_i, d_{\bar{e}} = \bar{e} - \lambda_i$$

$$0 = \left[\begin{array}{ccc|ccccccccccc}
d_{\bar{v}} & \bar{v}_1 & \bar{v}_1 & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 \\
\bar{e} & d_{\bar{e}} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
\bar{e} & 0 & d_{\bar{e}} & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 \\
\hline
f & 0 & f & d_f & f & & & & & & & & & 0 \\
6e & 0 & e & e & d_e & e & e & & & & & & & 0 \\
f & 0 & 0 & & f & d_f & f & & & & & & & \\
 & \vdots & & & & & \ddots & & & & & & & \\
6e & 0 & 0 & & & e & e & d_e & e & e & & & & \\
f & 0 & 0 & & & & & f & d_f & f & & & & \\
6e & 0 & 0 & & & & & e & e & d_e & e & e & & \\
 & \vdots & & & & & & & & & \ddots & & & \\
f & 0 & 0 & & & & & & & & f & d_f & f & \\
6e & e & 0 & 0 & & & & & & & e & e & d_e & e \\
f & f & 0 & 0 & & & & & & & & & f & d_f
\end{array}\right]
\left[\begin{array}{c}
c_0 \\
g\left[\frac{n}{2}\right] \\
g\left[-\frac{n}{2}\right] \\
\hline
h\left[-\frac{n-1}{2}\right] \\
g\left[-\frac{n-2}{2}\right] \\
h\left[-\frac{n-3}{2}\right] \\
\vdots \\
g\left[-\frac{1}{2}\right] \\
h\left[0\right] \\
g\left[\frac{1}{2}\right] \\
\vdots \\
h\left[\frac{n-3}{2}\right] \\
g\left[\frac{n-2}{2}\right] \\
h\left[\frac{n-1}{2}\right] \\
g\left[\frac{n}{2}\right]
\end{array}\right] \tag{4.169}$$

The next sample at the front and end of the sequence $t = -\frac{n}{2}$ and $t = \frac{n}{2}$ are not present in the first and last difference equations. So functions that have these samples equal to zero, $g\left[-\frac{n}{2}\right] = g\left[\frac{n}{2}\right] = 0$, must be chosen from the family of solutions. $g\left[\frac{n}{2}\right] = \cos\left[\frac{n}{2}\theta_i\right] = 0$ when $\theta_i = \frac{i\pi}{n}$ for odd integers i, see the left hand columns of Figures 4.3 and 4.4. While $g\left[\frac{n}{2}\right] = \sin\left[\frac{n}{2}\theta_i\right] = 0$ when $\theta_i = \frac{i\pi}{n}$ for even integers i, see the right hand columns of Figures 4.3 and 4.4. We can now write the formulas for $g\left[t\right]$ and $h\left[t\right]$ for odd and even columns of the eigenvector matrix. Note that we have applied a shift of $\delta\left[t + \frac{1}{2}\right]$ to the $h\left[t\right]$ function to make the indices in the eigenvector symmetric

about the middle element.

$$i \in [1, n-1] \qquad \theta_i = \frac{\pi i}{n} \qquad \lambda_i = e\left(5 + \cos(\theta_i) \pm \cos\left(\frac{\theta_i}{2}\right)\sqrt{2(9+\cos(\theta_i))}\right)$$

$$i\ Odd: \qquad g[t] = \cos\left[t\frac{\pi i}{n}\right], \quad h[t] = \frac{-4e}{4e-\lambda_i}\left(\cos\left[\left(t-\frac{1}{2}\right)\frac{\pi i}{n}\right] + \cos\left[\left(t+\frac{1}{2}\right)\frac{\pi i}{n}\right]\right) \tag{4.170}$$

$$i\ Even: \qquad g[t] = \sin\left[t\frac{\pi i}{n}\right], \quad h[t] = \frac{-4e}{4e-\lambda_i}\left(\sin\left[\left(t-\frac{1}{2}\right)\frac{\pi i}{n}\right] + \sin\left[\left(t+\frac{1}{2}\right)\frac{\pi i}{n}\right]\right) \tag{4.171}$$

These equations account for $2(n-1)$ of the eigenvectors because there are two different versions of $h[t]$ for a single i. The coefficient in $h[t]$ is based on λ_i which takes on two unique values. For all valences, the final eigenvector for the T_{12} subblock is for the eigenvalue $\lambda = \frac{1}{4} = 4e$ corresponding to $i = n$. For this eigenvalue, the previous eigenvector the functions $g[t]$ and $h[t]$ both reduce to zeros. Instead $g[t] = 0$ and $h[t] = \cos[t\pi]$ is used for valence n odd and $g[t] = 0$ and $h[t] = \sin[t\pi]$ is used for valence n even.

Similar to the Loop case, the final three eigenvectors of S are dependent on the subblock T_0. The eigenvalues of T_0 for the corner and crease cases take on values of $\lambda \in \left\{1, \frac{1}{2}, \frac{1}{4}\right\}$. These eigenvectors must have some of the top three elements nonzero which are all multiples of $g\left[\frac{n}{2}\right]$. Equation 4.162 shows the relationship between the eigenvalue λ and the frequency θ of the sine or cosine function. The following table shows the valences n which will have $g\left[\frac{n}{2}\right] = 0$ and can lead to a missing eigenvalue such that the matrix S is defective.

	$\theta = \cos^{-1}\left(\frac{16e^2-10e\lambda+\lambda^2}{2e\lambda}\right)$	$\cos\left(\frac{n}{2}\theta\right) = 0$	$\sin\left(\frac{n}{2}\theta\right) = 0$
$\lambda = \frac{1}{2} = 8e$	$\frac{\pi}{2}$	$n = 4k+2$	$n = 4k$
$\lambda = \frac{1}{4} = 4e$	π	$n = 2k+1$	$n = 2k$

(4.172)

Both n and k must be integers, so k can be removed from these expressions by using modulo arithmetic.

	$\theta = \cos^{-1}\left(\frac{16e^2-10e\lambda+\lambda^2}{2e\lambda}\right)$	$\cos\left(\frac{n}{2}\theta\right) = 0$	$\sin\left(\frac{n}{2}\theta\right) = 0$
$\lambda = \frac{1}{2} = 8e$	$\frac{\pi}{2}$	$n\%4 = 2$	$n\%4 = 0$
$\lambda = \frac{1}{4} = 4e$	π	$n\%2 = 1$	$n\%2 = 0$

(4.173)

4.8.3 Catmull-Clark Corner Jordan Vectors

In the corner vertex case, the subdivision matrix S is lower block triangular, so the eigenvalues of the diagonal subblocks are also eigenvalues of the entire matrix. The T_0 subblock is of the

form:

$$T_0 = \begin{bmatrix} 16e & 0 & 0 \\ 8e & 8e & 0 \\ 8e & 0 & 8e \end{bmatrix} \tag{4.174}$$

The eigen-decomposition of this block is:

$$T_0 V_0 = V_0 \Lambda_0 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & b_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & -b_{\frac{n}{2}} & b_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 8e & 0 \\ 0 & 0 & 8e \end{bmatrix} \tag{4.175}$$

The first eigenvector, associated with the eigenvalue $\lambda = 1$, is all ones due to the sum to one property of the averaging rules. We need two eigenvectors where $b_{-\frac{n}{2}} \neq 0$ and $b_{\frac{n}{2}} \neq 0$, so that they will be linearly independent of the eigenvectors of subblock T_{12}. Equation 4.173 shows that when $\lambda = \frac{1}{2}$ and n is divisible by 2, either the sine or cosine sequence will terminate with a zero value. In this case, the matrix is defective, and a nontrivial Jordan block will have to be used.

When the matrix is defective, we solve the generalized eigenvalue equation

$$(S - \lambda I)\, v_J = v_E \tag{4.176}$$

This is similar to Equation 4.168 with the zero vector replaced by an eigenvector from T_{12} composed of elements $a\,[t]$ and $b\,[t]$.

$$a\,[t] = \cos\left[t\frac{\pi i}{n}\right], \qquad b\,[t] = -\frac{4e}{4e-\lambda_i}\left(\cos\left[(t-1)\,\frac{\pi i}{n}\right] + \cos\left[t\frac{\pi i}{n}\right]\right) \tag{4.177}$$

$$a\,[t] = \sin\left[t\frac{\pi i}{n}\right], \qquad b\,[t] = -\frac{4e}{4e-\lambda_i}\left(\sin\left[(t-1)\,\frac{\pi i}{n}\right] + \sin\left[t\frac{\pi i}{n}\right]\right) \tag{4.178}$$

The relationship between $a\,[t]$ and $b\,[t]$ is

$$b\,[t] = -\frac{4e}{4e-\lambda_i}\left(a\,[t-1] + a\,[t]\right) \tag{4.179}$$

Taking $\lambda_i = \frac{1}{2} = 8e$ from subblock T_0, we get the following system of difference equations for the generalized eigenvector.

$$\begin{array}{rrrrrrrrr} b\,[t-1] = & 4eg\,[t-2] + & (4e-\lambda_i)\,h\,[t-1] + & 4eg\,[t-1] & & & & \\ a\,[t-1] = & eg\,[t-2] + & eh\,[t-1] + & (6e-\lambda_i)\,g\,[t-1] & + & eh\,[t] + & eg\,[t] \\ b\,[t] = & & & 4eg\,[t-1] & + & (4e-\lambda_i)\,h\,[t] + & 4eg\,[t] \end{array} \tag{4.180}$$

The bottom equation gives a formula for $h[t]$ in terms of $g[t]$ and $a[t]$:

$$h[t] = -\frac{4e}{4e-\lambda_i}(g[t-1]+g[t]) + \frac{1}{4e-\lambda_i}b[t] \tag{4.181}$$

$$= -\frac{4e}{4e-\lambda_i}\left((g[t-1]+g[t]) + \frac{1}{4e-\lambda_i}(a[t-1]+a[t])\right) \tag{4.182}$$

We subtract the face rule equations multiplied by e from the edge rule equation multiplied by $(4e-\lambda_i)$ to form a difference equation for $g[t]$.

$$-e\lambda_i g[t-2] + \left((4e-\lambda_i)(6e-\lambda_i) - 8e^2\right)g[t-1] - e\lambda_i g[t] = \begin{pmatrix} (4e-\lambda_i)\,a[t-1] \\ -e\,(b[t-1]+b[t]) \end{pmatrix} \tag{4.183}$$

$$g[t-2] - \tfrac{16e^2-10e\lambda_i+\lambda_i^2}{e\lambda_i}g[t-1] + g[t] = -\tfrac{1}{e\lambda_i(4e-\lambda_i)}\begin{pmatrix} (4e-\lambda_i)^2\,a[t-1] \\ +4e^2\,(a[t-2]+2a[t-1]+a[t]) \end{pmatrix} \tag{4.184}$$

$$g[t-2] - \tfrac{16e^2-10e\lambda_i+\lambda_i^2}{e\lambda_i}g[t-1] + g[t] = -\tfrac{1}{e\lambda_i(4e-\lambda_i)}\begin{pmatrix} \left(24e^2-8e\lambda_i+\lambda_i^2\right)a[t-1] \\ +4e^2\,(a[t-2]+a[t]) \end{pmatrix} \tag{4.185}$$

We will put this equation in a general form to simplify the work. The function $a[t]$ satisfies the same difference equation.

$$\alpha = -2\cos(\theta_i) = -\frac{16e^2-10e\lambda_i+\lambda_i^2}{e\lambda_i} \tag{4.186}$$

$$g[t-2] + \alpha g[t-1] + g[t] = f[t] \tag{4.187}$$

$$a[t-2] + \alpha a[t-1] + a[t] = 0 \tag{4.188}$$

$$\beta = \frac{\left(24e^2-8e\lambda_i+\lambda_i^2\right)}{4e^2} \tag{4.189}$$

$$\gamma = \frac{4e}{\lambda_i(4e-\lambda_i)} \tag{4.190}$$

$$f[t] = -\frac{1}{e\lambda_i(4e-\lambda_i)}\left(\left(24e^2-8e\lambda_i-\lambda_i^2\right)a[t-1] + 4e^2\,(a[t-2]+a[t])\right) \tag{4.191}$$

$$= -\frac{4e}{\lambda_i(4e-\lambda_i)}\left(\frac{\left(24e^2-8e\lambda_i-\lambda_i^2\right)}{4e^2}a[t-1] + (a[t-2]+a[t])\right) \tag{4.192}$$

$$= -\gamma\,(\beta a[t-1] + (a[t-2]+a[t])) \tag{4.193}$$

We move to the z-domain using the z-transform. First consider the function $f[t]$ alone.

$$f[0] = -\gamma(\beta a[-1] + (a[-2] + a[0])) \tag{4.194}$$

$$f(z) = -\gamma\left(\beta\left(a[-1] + z^{-1}a(z)\right) + \left(a[-2] + a[-1]z^{-1} + z^{-2}a(z) + a(z)\right)\right) \tag{4.195}$$

$$= -\gamma\left((a[-2] + \beta a[-1]) + a[-1]z^{-1} + \left(1 + \beta z^{-1} + z^{-2}\right)a(z)\right) \tag{4.196}$$

Next we apply the z-transform to the entire difference equation and solve for $g(z)$.

$$\left(g[-2] + g[-1]z^{-1} + z^{-2}g(z)\right) + \alpha\left(g[-1] + z^{-1}g(z)\right) + g(z) = f(z) \tag{4.197}$$

$$\left(1 + \alpha z^{-1} + z^{-2}\right)g(z) + \left(g[-2] + \alpha g[-1] + g[-1]z^{-1}\right) = f(z) \tag{4.198}$$

$$g(z) = -\frac{\left(g[-2] + \alpha g[-1] + g[-1]z^{-1}\right)}{1 + \alpha z^{-1} + z^{-2}} + \frac{f(z)}{1 + \alpha z^{-1} + z^{-2}} \tag{4.199}$$

We are only concerned with the case when $\lambda_i = \frac{1}{2}$ which greatly simplifies our formulas.

$$\lambda_i = \frac{1}{2} = 8e \tag{4.200}$$

$$\alpha = -2\cos(\theta_i) = -\frac{16e^2 - 80e^2 + 64e^2}{8e^2} = 0 \tag{4.201}$$

$$\theta_i = \frac{\pi}{2} \tag{4.202}$$

$$\beta = \frac{\left(24e^2 - 64e^2 + 64e^2\right)}{4e^2} = 6 \tag{4.203}$$

$$\gamma = \frac{4e}{8e(4e - 8e)} = -\frac{1}{8e} = -2 \tag{4.204}$$

$$\tag{4.205}$$

We solve for the constants $a[-2]$, $f[0]$, $g[-2]$ to help simplify the expression for $g(z)$.

$$a[-2] + \alpha a[-1] + a[0] = 0 \tag{4.206}$$

$$a[-2] = -a[0] \tag{4.207}$$

$$f[0] = 12a[-1] \tag{4.208}$$

$$g[-2] + \alpha g[-1] + g[0] = f[0] \tag{4.209}$$

$$g[-2] = f[0] - g[0] = 12a[-1] - g[0] \tag{4.210}$$

Next we simplify our z-domain functions.

$$f(z) = 2\left((a[-2] + 6a[-1]) + a[-1]z^{-1} + \left(1 + 6z^{-1} + z^{-2}\right)a(z)\right) \tag{4.211}$$

$$g(z) = -\frac{\left(g[-2] + g[-1]z^{-1}\right)}{1 + z^{-2}} + \frac{f(z)}{1 + z^{-2}} \tag{4.212}$$

$$= \frac{\left(-g[-2] - g[-1]z^{-1}\right)}{1 + z^{-2}} + \frac{2\left((a[-2] + 6a[-1]) + a[-1]z^{-1}\right)}{1 + z^{-2}} + \frac{2\left(1 + 6z^{-1} + z^{-2}\right)a(z)}{1 + z^{-2}} \tag{4.213}$$

$$= \frac{(-g[-2] + 2a[-2] + 12a[-1]) + (-g[-1] + 2a[-1])z^{-1}}{1 + z^{-2}} + 2\left(\frac{1 + z^{-2}}{1 + z^{-2}} + \frac{6z^{-1}}{1 + z^{-2}}\right)a(z) \tag{4.214}$$

$$= \frac{(g[0] - 2a[0])}{1 + z^{-2}} + \frac{(-g[-1] + 2a[-1])z^{-1}}{1 + z^{-2}} + 2a(z) + 12\left(\frac{z^{-1}}{1 + z^{-2}}\right)a(z) \tag{4.215}$$

We apply the inverse z-transform to return to the time domain, and we apply some simplifications to reduce the scope of the convolution summation.

$$g[t] = (g[0] - 2a[0])\cos\left[t\frac{\pi}{2}\right]u[t] + (-g[-1] + 2a[-1])\sin\left[t\frac{\pi}{2}\right]u[t] + 2a[t]u[t] + 12\left(\sin\left[t\frac{\pi}{2}\right]u[t]\right) \otimes (a[t]u[t]) \tag{4.216}$$

$$g[t] = (g[0] - 2a[0])\cos\left[t\frac{\pi}{2}\right]u[t] + (-g[-1] + 2a[-1])\sin\left[t\frac{\pi}{2}\right]u[t] + 2a[t]u[t] + 12\sum_{k=-\infty}^{\infty}\sin\left[k\frac{\pi}{2}\right]u[k]a[t-k]u[t-k] \tag{4.217}$$

$$g[t] = (g[0] - 2a[0])\cos\left[t\frac{\pi}{2}\right]u[t] + (-g[-1] + 2a[-1])\sin\left[t\frac{\pi}{2}\right]u[t] + 2a[t]u[t] + 12\sum_{k=-\infty}^{t}\sin\left[k\frac{\pi}{2}\right]u[k]a[t-k] \tag{4.218}$$

$$g[t] = u[t]\begin{pmatrix} (g[0] - 2a[0])\cos\left[t\frac{\pi}{2}\right] + (-g[-1] + 2a[-1])\sin\left[t\frac{\pi}{2}\right] + 2a[t] \\ +12\sum_{k=0}^{t}\sin\left[k\frac{\pi}{2}\right]a[t-k] \end{pmatrix} \tag{4.219}$$

Odd Symmetry

When the wedge valence n is divisible by 4, a Jordan vector coupled with the odd symmetry eigenvector solution $a[t] = \sin\left[t\frac{\pi}{2}\right]$ is required. Evaluating the difference equation gives

some simplifying substitutions.

$$a[t] = \sin\left[t\frac{\pi}{2}\right] \tag{4.220}$$

$$a[0] = 0 \tag{4.221}$$

$$a[-1] = -1 \tag{4.222}$$

$$g[1] = -g[-1] \tag{4.223}$$

We simplify $g[t]$ by substituting these values and applying some trigonometric identities.

$$g[t] = u[t]\left(\begin{array}{c}(g[0]-2a[0])\cos\left[t\frac{\pi}{2}\right]+(-g[-1]+2a[-1])\sin\left[t\frac{\pi}{2}\right]+2a[t]\\ +12\sum_{k=0}^{t}\sin\left[k\frac{\pi}{2}\right]a[t-k]\end{array}\right) \tag{4.224}$$

$$= u[t]\left(\begin{array}{c}(g[0])\cos\left[t\frac{\pi}{2}\right]+(-g[-1]-2)\sin\left[t\frac{\pi}{2}\right]+2\sin\left[t\frac{\pi}{2}\right]\\ +12\sum_{k=0}^{t}\sin\left[k\frac{\pi}{2}\right]\sin\left[(t-k)\frac{\pi}{2}\right]\end{array}\right) \tag{4.225}$$

$$= u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]+g[1]\sin\left[t\frac{\pi}{2}\right]+12\sum_{k=0}^{t}\frac{1}{2}\left(\cos\left[(2k-t)\frac{\pi}{2}\right]-\cos\left[t\frac{\pi}{2}\right]\right)\right) \tag{4.226}$$

$$= u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]+g[1]\sin\left[t\frac{\pi}{2}\right]-6(t+1)\cos\left[t\frac{\pi}{2}\right]+6\sum_{k=0}^{t}\cos\left[(2k-t)\frac{\pi}{2}\right]\right) \tag{4.227}$$

The term $\cos\left[(2k-t)\frac{\pi}{3}\right]$ is periodic, and each of these cycles of two consecutive integer k's sum to zero. The sum can then be reduce to the remainder of $\frac{t}{2}$ terms which is at most one term where $k = t$. The summation can be eliminated completely using the formula:

$$\sum_{k=0}^{t}\cos\left[(2k-t)\frac{\pi}{2}\right] = \frac{1}{2}(1+\cos[t\pi])\cos\left[t\frac{\pi}{2}\right] \tag{4.228}$$

This substitution simplifies $g[t]$ further.

$$g[t] = u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]+g[1]\sin\left[t\frac{\pi}{2}\right]-6(t+1)\cos\left[t\frac{\pi}{2}\right]+3(1+\cos[t\pi])\cos\left[t\frac{\pi}{2}\right]\right) \tag{4.229}$$

$$= u[t]\left((g[0]-6t-3+3\cos[t\pi])\cos\left[t\frac{\pi}{2}\right]+g[1]\sin\left[t\frac{\pi}{2}\right]\right) \tag{4.230}$$

When n is divisible by 4, we are searching for the odd symmetry solution, so we must set $g\left[\frac{n}{2}\right] = -g\left[-\frac{n}{2}\right]$.

$$0 = g\left[\frac{n}{2}\right] + g\left[-\frac{n}{2}\right] \tag{4.231}$$

$$0 = 2\left(g\left[0\right] - 3 + 3\cos\left[\frac{n}{4}2\pi\right]\right)\cos\left[\frac{n}{4}\pi\right] \tag{4.232}$$

$$g\left[0\right] = 3 - 3\cos\left[\frac{n}{4}2\pi\right] \tag{4.233}$$

$$= 0 \tag{4.234}$$

Substituting $g\left[0\right] = 0$ provides the final simplification to our signal $g\left[t\right]$.

$$g\left[t\right] = u\left[t\right]\left(-3\left(2t + 1 - \cos\left[t\pi\right]\right)\cos\left[t\frac{\pi}{2}\right] + g\left[1\right]\sin\left[t\frac{\pi}{2}\right]\right) \tag{4.235}$$

The value of $g\left[1\right]$ is unimportant as the sine function always sums to zero in the difference equation.

Even Symmetry

When the wedge valence n is even and not divisible by 4, a Jordan vector coupled with the even symmetry eigenvector solution $a\left[t\right] = \cos\left[t\frac{\pi}{2}\right]$ is required. Evaluating the difference equation gives some simplifying substitutions.

$$a\left[t\right] = \cos\left[t\frac{\pi}{2}\right] \tag{4.236}$$

$$a\left[0\right] = 1 \tag{4.237}$$

$$a\left[-1\right] = 0 \tag{4.238}$$

$$g\left[1\right] = g\left[-1\right] \tag{4.239}$$

We simplify $g[t]$ by substituting these values and applying some trigonometric identities.

$$g[t] = u[t]\left(\begin{array}{c}(g[0]-2a[0])\cos\left[t\frac{\pi}{2}\right]+(-g[-1]+2a[-1])\sin\left[t\frac{\pi}{2}\right]+2a[t]\\ +12\sum_{k=0}^{t}\sin\left[k\frac{\pi}{2}\right]a[t-k]\end{array}\right) \tag{4.240}$$

$$= u[t]\left(\begin{array}{c}(g[0]-2)\cos\left[t\frac{\pi}{2}\right]+(-g[1])\sin\left[t\frac{\pi}{2}\right]+2\cos\left[t\frac{\pi}{2}\right]\\ +12\sum_{k=0}^{t}\sin\left[k\frac{\pi}{2}\right]\cos\left[(t-k)\frac{\pi}{2}\right]\end{array}\right) \tag{4.241}$$

$$= u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]-g[1]\sin\left[t\frac{\pi}{2}\right]+12\sum_{k=0}^{t}\frac{1}{2}\left(\sin\left[(2k-t)\frac{\pi}{2}\right]+\sin\left[t\frac{\pi}{2}\right]\right)\right) \tag{4.242}$$

$$= u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]-g[1]\sin\left[t\frac{\pi}{2}\right]+6(t+1)\sin\left[t\frac{\pi}{2}\right]+6\sum_{k=0}^{t}\sin\left[(2k-t)\frac{\pi}{2}\right]\right) \tag{4.243}$$

The term $\sin\left[(2k-t)\frac{\pi}{2}\right]$ is periodic, and each of these cycles of two consecutive integer k's sum to zero. The sum can then be reduce to the remainder of $\frac{t}{2}$ terms which is at most one term where $k=t$. The summation can be eliminated completely using the formula:

$$\sum_{k=0}^{t}\sin\left[(2k-t)\frac{\pi}{2}\right]=\frac{1}{2}(1+\cos[t\pi])\sin\left[t\frac{\pi}{2}\right] \tag{4.244}$$

This substitution simplifies $g[t]$ further.

$$g[t] = u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]+(6t+6-g[1])\sin\left[t\frac{\pi}{2}\right]+3(1+\cos[t\pi])\sin\left[t\frac{\pi}{2}\right]\right) \tag{4.245}$$

$$= u[t]\left(g[0]\cos\left[t\frac{\pi}{2}\right]+(6t+9+3\cos[t\pi]-g[1])\sin\left[t\frac{\pi}{2}\right]\right) \tag{4.246}$$

When n is even and not divisible by 4, we are searching for the even symmetry solution, so we must set $g\left[\frac{n}{2}\right]=g\left[-\frac{n}{2}\right]$.

$$0 = g\left[\frac{n}{2}\right]-g\left[-\frac{n}{2}\right] \tag{4.247}$$

$$0 = 2\left(9+3\cos\left[\frac{n}{2}\pi\right]-g[1]\right)\sin\left[\frac{n}{2}\frac{\pi}{2}\right] \tag{4.248}$$

$$g[1] = 9+3\cos\left[\frac{n}{2}\pi\right] \tag{4.249}$$

$$g[1] = 6 \tag{4.250}$$

Substituting $g[1]=6$ provides the final simplification to our signal $g[t]$.

$$g[t] = u[t]\left(3(2t+1+\cos[t\pi])\sin\left[t\frac{\pi}{2}\right]+g[0]\cos\left[t\frac{\pi}{2}\right]\right) \tag{4.251}$$

The value of $g[0]$ is unimportant as the cosine function always sums to zero in the difference equation.

4.8.4 Catmull-Clark Crease Jordan Vectors

In the crease vertex case, the subdivision matrix S is lower block triangular, so the eigenvalues of the diagonal subblocks are also eigenvalues of the entire matrix. The T_0 subblock is of the form:

$$T_0 = \begin{bmatrix} 12e & 2e & 2e \\ 8e & 8e & 0 \\ 8e & 0 & 8e \end{bmatrix} \tag{4.252}$$

The eigen-decomposition of this block is:

$$T_0 V_0 = V_0 \Lambda_0 = \begin{bmatrix} 1 & 0 & -\frac{1}{2} b_{\frac{n}{2}} \\ 1 & b_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & -b_{\frac{n}{2}} & b_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 8e & 0 \\ 0 & 0 & 4e \end{bmatrix} \tag{4.253}$$

The first eigenvector, associated with the eigenvalue $\lambda = 1$, is all ones due to the sum to one property of the averaging rules. We need two eigenvectors where $b_{-\frac{n}{2}} \neq 0$ and $b_{\frac{n}{2}} \neq 0$, so that they will be linearly independent of the eigenvectors of subblock T_{12}. Equation 4.173 shows that when $\lambda = \frac{1}{2} = 8e$ and n is divisible by 4, the odd symmetry sine sequence will terminate with a zero value. In this case, the matrix is defective, and a nontrivial Jordan block with the Jordan vector from Equation 4.235 will have to be used. Further more when $\lambda = \frac{1}{4} = 4e$ and n is odd, the even symmetry cosine sequence will terminate with a zero value. In this case, the matrix is defective, and we must find the Jordan vector by solving the generalized eigenvalue equation

$$(S - \lambda I)\, v_J = v_E \tag{4.254}$$

The eigenvector from the block T_{12} associated with $\lambda = \frac{1}{4}$ has elements

$$a[t] = 0, \qquad b[t] = \cos[t\pi] \tag{4.255}$$

Because n is odd, we want to sample at half integer values. To handle this, we will substitute $s = 2t$ when forming our difference equations. The generalized eigenvector equations are:

$$a[s] = 0, b[s] = \cos\left[s\frac{\pi}{2}\right] \tag{4.256}$$

$$\begin{aligned} b[s-2] &= 4eg[s-3] + 0h[s-2] + 4eg[s-1] \\ a[s-1] &= eg[s-3] + eh[s-2] + 2eg[s-1] + eh[s] + eg[s+1] \\ b[s] &= 4eg[s-1] + 0h[s] + 4eg[s+1] \end{aligned} \tag{4.257}$$

The top face rule difference equation defines $g[s]$, and by multiplying the edge rule equation by 4 and subtracting the two face rules we generate a difference equation for $h[s]$. The difference equation for $h[s]$ is equal to zero because alternating pairs of values of $b[s]$ sum to zero.

$$g[s-3]+g[s-1]=\frac{1}{4e}b[s-2] \tag{4.258}$$

$$4eh[s-2]+4eh[s]=-(b[s-2]+b[s])=0 \tag{4.259}$$

$$h[s-2]+h[s]=0 \tag{4.260}$$

First we will solve for $h[s]$ using the z-transform. This difference equation is for sine or cosine where $\theta=\frac{\pi}{2}$.

$$h[s]=-h[-2]\cos\left[s\frac{\pi}{2}\right]-h[-1]\sin\left[s\frac{\pi}{2}\right] \tag{4.261}$$

Second we will solve for $g[s]$ using the z-transform. We substitute $s=0$ into the difference equation to derive a useful relationship between some of the samples.

$$g[s-3]+g[s-1]=\frac{1}{4e}b[s-2] \tag{4.262}$$

$$g[-3]+g[-1]=\frac{1}{4e}b[-2] \tag{4.263}$$

We apply the z-transform, and solve for the z-domain function $g(z)$.

$$\left(\begin{array}{c}\left(g[-3]+g[-2]z^{-1}+g[-1]z^{-2}+z^{-3}g(z)\right)\\+\left(g[-1]+z^{-1}g(z)\right)\end{array}\right)=\frac{1}{4e}\left(b[-2]+b[-1]z^{-1}+z^{-2}b(z)\right) \tag{4.264}$$

$$g(z)=\frac{\left(\frac{1}{4e}b[-2]-(g[-3]+g[-1])\right)+\left(\frac{1}{4e}b[-1]-g[-2]\right)z^{-1}-g[-1]z^{-2}+\frac{1}{4e}z^{-2}b(z)}{z^{-1}(1+z^{-2})} \tag{4.265}$$

Substituting Equation 4.263 and $b[-1]=0$, we simplify the expression for $g(z)$ and split it into terms that have simple inverse z-transforms.

$$g(z)=\frac{1}{4e}\frac{(b[-1]-4eg[-2])z^{-1}-4eg[-1]z^{-2}+z^{-2}b(z)}{z^{-1}(1+z^{-2})} \tag{4.266}$$

$$=\frac{1}{4e}\left(\frac{b[-1]-4eg[-2]}{1+z^{-2}}-\frac{4eg[-1]z^{-1}}{1+z^{-2}}+\frac{z^{-1}}{1+z^{-2}}b(z)\right) \tag{4.267}$$

We apply the inverse z-transform and perform simplifications similar to the corner vertex case to get a compact form for $g[s]$. Note that $16e = 1$.

$$g[s] = \frac{1}{4e}\begin{pmatrix} -4eg[-2]\cos\left[s\frac{\pi}{2}\right]u[s] - 4eg[-1]\sin\left[s\frac{\pi}{2}\right]u[s] \\ +\sin\left[s\frac{\pi}{2}\right]u[s] \otimes b[s]\,u[s] \end{pmatrix} \tag{4.268}$$

$$= \frac{1}{4e}\begin{pmatrix} -4eg[-2]\cos\left[s\frac{\pi}{2}\right]u[s] - 4eg[-1]\sin\left[s\frac{\pi}{2}\right]u[s] \\ +\sin\left[s\frac{\pi}{2}\right]u[s] \otimes \cos\left[s\frac{\pi}{2}\right]u[s] \end{pmatrix} \tag{4.269}$$

$$= \frac{u[s]}{4e}\begin{pmatrix} -4eg[-2]\cos\left[s\frac{\pi}{2}\right] - 4eg[-1]\sin\left[s\frac{\pi}{2}\right] \\ +\sum_{k=0}^{s}\sin\left[k\frac{\pi}{2}\right]\cos\left[(s-k)\frac{\pi}{2}\right] \end{pmatrix} \tag{4.270}$$

$$= \frac{u[s]}{4e}\begin{pmatrix} -4eg[-2]\cos\left[s\frac{\pi}{2}\right] - 4eg[-1]\sin\left[s\frac{\pi}{2}\right] \\ +\frac{1}{2}\left((s+1)\sin\left[s\frac{\pi}{2}\right] + \frac{1}{2}(1+\cos[s\pi])\sin\left[s\frac{\pi}{2}\right]\right) \end{pmatrix} \tag{4.271}$$

$$= \frac{u[s]}{16e}\begin{pmatrix} -16eg[-2]\cos\left[s\frac{\pi}{2}\right] - 16eg[-1]\sin\left[s\frac{\pi}{2}\right] \\ +\left(2(s+1)\sin\left[s\frac{\pi}{2}\right] + (1+\cos[s\pi])\sin\left[s\frac{\pi}{2}\right]\right) \end{pmatrix} \tag{4.272}$$

$$= u[s]\left(-g[-2]\cos\left[s\frac{\pi}{2}\right] + (2s+3-g[-1]+\cos[s\pi])\sin\left[s\frac{\pi}{2}\right]\right) \tag{4.273}$$

We now resubstitute $s = 2t$ to get a solution sampled at the desired half integer index frequency.

$$g[t] = u[t]\left(-g[-1]\cos[t\pi] + \left(4t+3-g\left[-\frac{1}{2}\right]+\cos[t2\pi]\right)\sin[t\pi]\right) \tag{4.274}$$

$$h[t] = -h[-1]\cos[t\pi] - h\left[-\frac{1}{2}\right]\sin[t\pi] \tag{4.275}$$

The undetermined constants in $h[t]$ are unimportant and can be set to zero. When the wedge valence n is odd, we are searching for the even symmetry solution, so we must set $g\left[\frac{n}{2}\right] = g\left[-\frac{n}{2}\right]$.

$$0 = g\left[\frac{n}{2}\right] - g\left[-\frac{n}{2}\right] \tag{4.276}$$

$$0 = 2\left(3 - g\left[-\frac{1}{2}\right] + \cos[n\pi]\right)\sin\left[\frac{n\pi}{2}\right] \tag{4.277}$$

$$g\left[-\frac{1}{2}\right] = 3 + \cos[n\pi] \tag{4.278}$$

$$g\left[-\frac{1}{2}\right] = 2 \tag{4.279}$$

Substituting $g\left[-\frac{1}{2}\right] = 2$ provides the final simplification to our signal $g[t]$.

$$g[t] = (4t+1+\cos[t2\pi])\sin[t\pi] - g[-1]\cos[t\pi] \tag{4.280}$$

$$h[t] = 0 \tag{4.281}$$

The value of $g[-1]$ is unimportant as the cosine function always sums to zero in the difference equation.

4.8.5 Catmull-Clark Dart Eigenvectors

In the dart vertex case, the eigenvectors associated with the face and edge averaging rules are coupled to the vertex averaging rules. The T_{12} subblock of edge averaging rules for the eigenvectors of the dart subdivision matrix is exactly the same as crease and corner case. The difference is appearant in the T_{02} subblock because the v_1 and v_2 averaging coefficients in the crease and corner case are all zeros. The dart subdivision matrix when the valence n is even is:

$$e = \frac{1}{16}, f = \frac{1}{4} = 4e, \bar{e} = \frac{1}{2} = 8e, v_0 = 1 - n\left(v_1 + v_2\right), v_1 = \frac{6}{4n^2}, v_2 = \frac{1}{4n^2}$$

$$d_e = 6e - \lambda_i, d_f = f - \lambda_i = 4e - \lambda_i, d_{\bar{e}} = \bar{e} - \lambda_i, d_v = v_0 - \lambda_i$$

$$0 = \left[\begin{array}{cc|ccccccccccc} d_v & v_1 & v_2 & v_1 & v_2 & \cdots & v_2 & v_1 & v_2 & \cdots & v_2 & v_1 & v_2 \\ \bar{e} & d_{\bar{e}} & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 \\ \hline f & f & d_f & f & & & & & & & & & \\ 6e & e & e & d_e & e & e & & & & & & & \\ f & 0 & & f & d_f & f & & & & & & & \\ \vdots & & & & & \ddots & & & & & & & \\ f & 0 & & & & f & d_f & f & & & & & \\ 6e & 0 & & & & e & e & d_e & e & e & & & \\ f & 0 & & & & & & f & d_f & f & & & \\ \vdots & & & & & & & & & \ddots & & & \\ f & 0 & & & & & & & & f & d_f & f & \\ 6e & e & & & & & & & & e & e & d_e & e \\ f & f & & & & & & & & & & f & d_f \end{array}\right] \left[\begin{array}{c} c_0 \\ a_{\frac{n}{2}} \\ \hline b_{-\frac{n-1}{2}} \\ a_{-\frac{n-2}{2}} \\ b_{-\frac{n-3}{2}} \\ \vdots \\ b_{-\frac{1}{2}} \\ a_0 \\ b_{\frac{1}{2}} \\ \vdots \\ b_{\frac{n-3}{2}} \\ a_{\frac{n-2}{2}} \\ b_{\frac{n-1}{2}} \\ a_{\frac{n}{2}} \end{array}\right] \tag{4.282}$$

The dart subdivision matrix when the valence n is odd is:

$$e = \frac{1}{16}, f = \frac{1}{4} = 4e, \bar{e} = \frac{1}{2} = 8e, v_0 = 1 - n\left(v_1 + v_2\right), v_1 = \frac{6}{4n^2}, v_2 = \frac{1}{4n^2}$$

$$d_e = 6e - \lambda_i, d_f = f - \lambda_i = 4e - \lambda_i, d_{\bar{e}} = \bar{e} - \lambda_i, d_v = v_0 - \lambda_i$$

$$0 = \left[\begin{array}{cc|ccccccccccc} dv & v_1 & v_2 & v_1 & v_2 & \cdots & v_1 & v_2 & v_1 & \cdots & v_2 & v_1 & v_2 \\ \bar{e} & d_{\bar{e}} & 0 & 0 & 0 & & 0 & 0 & 0 & & 0 & 0 & 0 \\ \hline f & f & d_f & f & & & & & & & & & \\ 6e & e & e & d_e & e & e & & & & & & & \\ f & 0 & & f & d_f & f & & & & & & & \\ \vdots & & & & & \ddots & & & & & & & \\ 6e & 0 & & & e & e & d_e & e & e & & & & \\ f & 0 & & & & & f & d_f & f & & & & \\ 6e & 0 & & & & & e & e & d_e & e & e & & \\ \vdots & & & & & & & & & \ddots & & & \\ f & 0 & & & & & & & & f & d_f & f & \\ 6e & e & & & & & & & & e & e & d_e & e \\ f & f & & & & & & & & & & f & d_f \end{array}\right] \left[\begin{array}{c} c_0 \\ a_{\frac{n}{2}} \\ \hline b_{-\frac{n-1}{2}} \\ a_{-\frac{n-2}{2}} \\ b_{-\frac{n-3}{2}} \\ \vdots \\ a_{-\frac{1}{2}} \\ b_0 \\ a_{\frac{1}{2}} \\ \vdots \\ b_{\frac{n-3}{2}} \\ a_{\frac{n-2}{2}} \\ b_{\frac{n-1}{2}} \\ a_{\frac{n}{2}} \end{array}\right] \tag{4.283}$$

In the crease and corner cases, the sine and cosine subblock eigenvectors are also eigenvectors of the entire matrix when extended at the top with zeros because it is lower block triangular. The vertex averaging rule from the dart case makes it more complicated. The odd symmetry sine based eigenvectors are the same because they sum to zero, so when they are dotted with the top row vector of alternating v_1 and v_2 the result is zero. But this leaves $n + 2$ even symmetry eigenvectors undetermined. The problem with eigenvectors from the crease and corner cases, which had cosine sequences that terminated with zeros on the ends, is that multiplication with the top vertex rule row will create a nonzero value. It is not possible to make c_0 nonzero while keeping $b_{\frac{n}{2}}$ zero because of the second row.

The same difference equation is still present, so a cosine-like sequence must be used. The difference is that $a_{\frac{n}{2}}$, which contains the value for both ends of the sequence, must be nonzero. Fortunately in the even symmetry case, $a_{\frac{n}{2}} = a_{-\frac{n}{2}}$, so we just need to adjust θ_i. Examining one of the middle rows, we see that c_0 which must be nonzero is multiplied by the nonzero value f or $6e$, so the difference equation must sum to a nonzero quantity. This can be accomplished by adding a

constant to every term in the the sequence.

$$a_t = c_1 + g[t] \tag{4.284}$$

$$b_t = c_2 + h[t] \tag{4.285}$$

Using the sharp edge rule from row 2 and any of the smooth face and edge rule rows, we can solve for c_0, c_1 and c_2 in terms of the eigenvalue λ_i and $g\left[\frac{n}{2}\right]$.

$$0 = 8ec_0 + (8e - \lambda_i)\left(c_1 + g\left[\frac{n}{2}\right]\right) \tag{4.286}$$

$$0 = 4ec_0 + 8ec_1 + (4e - \lambda_i)c_2 \tag{4.287}$$

$$0 = 6ec_0 + (8e - \lambda_i)c_1 + 2ec_2 \tag{4.288}$$

The expressions for c_0, c_1 and c_2 are:

$$c_0 = -\frac{(8e - \lambda_i)\left(16e^2 - 12e\lambda_i + \lambda_i^2\right)}{2e\lambda_i(16e - \lambda_i)} g\left[\frac{n}{2}\right] \tag{4.289}$$

$$c_1 = -\frac{64e^2 - 32e\lambda_i + 3\lambda_i^2}{\lambda_i(16e - \lambda_i)} g\left[\frac{n}{2}\right] \tag{4.290}$$

$$c_2 = \frac{2(8e - \lambda_i)(4e + \lambda_i)}{\lambda_i(16e - \lambda_i)} g\left[\frac{n}{2}\right] \tag{4.291}$$

All of the elements of the eigenvector are now multiples of $g[-1]$ which can be divided out, so the top vertex averaging row can be used to create a lower degree characteristic polynomial with the sine based eigenvalue factors removed.

$$0 = (v_0 - \lambda_i)c_0 + v_1\left(nc_1 + g\left[\frac{n}{2}\right] + g[0] + 2\sum_{k=1}^{\lceil\frac{n-2}{2}\rceil} g[k]\right) + v_2\left(nc_2 + 2\sum_{k=0}^{\lceil\frac{n-2}{2}\rceil} h\left[k + \frac{1}{2}\right]\right) \tag{4.292}$$

We do not currently have a closed form solution for these eigenvalues, so we use a numerical solver to get the eigenvalues and use them to generate the eigenvectors. Some of the eigenvalues are outside the range in which a cosine function is possible, so we use the general solution form instead.

4.9 Conclusion

We have shown how the eigen-structure of the 1-ring subdivision matrices of piecewise smooth Loop and Catmull-Clark subdivision can be understood using 1D signal processing techniques. All of these matrices contain a T_{12} subblock that is composed of a regular pattern of edge

or face averaging rules. When solving for the eigenvectors of the matrix, this pattern becomes the definition of a difference equation whose solution signal can be solved for using the technique of z-transforms. We derived formulas for the eigenvectors for the subdivision matrices for all vertex types of piecewise smooth Loop and Catmull-Clark subdivision. We also derived formulas for the generalized Jordan vectors in the defective cases for certain wedge valences in the corner and crease vertex cases.

In Chapter 5, we will reexamine the eigen-decomposition of smooth, corner, and crease vertices using properties of circulant and Toeplitz matrices which are related to the difference equations discussed in this chapter. This alternate form of analysis is used more frequently in the subdivision literature, and it provides an easy way to analyze more general schemes with modified edge rules. This matrix analysis is a short cut for the z-transform techniques presented here, but the underlying structure is the same.

Chapter 5

Circulant and Toeplitz Matrices

5.1 Introduction

The pattern of edge and face subdivision averaging rules around a vertex form either a circulant or Toeplitz matrix. The subdivision surface rules around a smooth interior vertex are topologically equivalent to a disc, and these rules generate a circulant matrix. The subdivision surface around a sharp boundary vertex is broken up by radiating sharp edges into sectors called wedges. The topology of a wedge is equalivalent to a half disc, and the subdivision averaging rules correspond to a Toeplitz matrix. In this chapter, we will review some of the properties of circulant and Toeplitz matrices, which we will later use to eigen-analyze the continuity behavior of subdivision surfaces.

5.2 Circulant Matrices

A circulant matrix [8] S has one unique row vector which is copied and shifted to form the other rows of the matrix. The rows of S can be thought of as averaging rules over a cyclic domain, e.g. the disc topology around a smooth vertex of a subdivision surface.

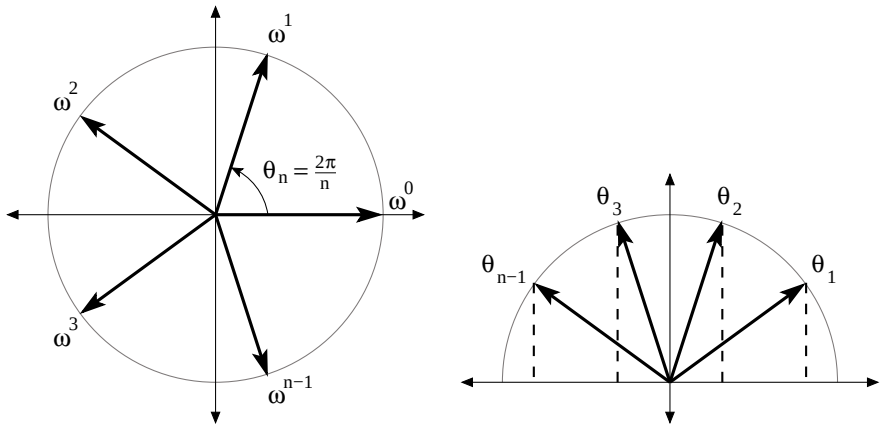
$$S = \begin{bmatrix} a_0 & a_1 & a_2 & & a_{n-1} \\ a_{n-1} & a_0 & a_1 & \cdots & a_{n-2} \\ a_{n-2} & a_{n-1} & a_0 & & a_{n-3} \\ & \vdots & & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{bmatrix} \tag{5.1}$$

A circulant matrix can always be diagonalized into its eigen-decomposition.

$$SF = F\Lambda \tag{5.2}$$

$$S = F\Lambda F^{-1} \tag{5.3}$$

The set of eigenvectors F of a circulant matrix correspond to the discrete Fourier transform (DFT).



(a) *Nth root of unity phasors* $\omega^k = e^{k\frac{2\pi}{n}\mathbf{i}}$ *on the imaginary plane for n=5*

(b) *Angles* $\theta_k = k\frac{\pi}{n}$ *over the real axis for n=5*

Figure 5.1: *Comparison of full disc topology of the Nth roots of unity to the half disc topology of sharp boundary edges.*

The terms of the DFT are unit phasors (Figure 5.1(a)) which are the nth roots of unity where n is the size of the matrix. These are the solutions to the equation:

$$x^n = 1 \tag{5.4}$$

The primitive root of unity is the phasor ω. It is a unit magnitude complex number represented in exponential form at a phase angle $\theta_n = \frac{2\pi}{n}$.

$$\omega = e^{\mathbf{i}\theta_n} = \cos\theta_n \ + \mathbf{i}\sin\theta_n \tag{5.5}$$

All other nth roots of unity x_i are integer powers of ω.

$$i \in [0, n-1] : x_i = \omega^i \tag{5.6}$$

Multiplying by ω corresponds to rotating by the angle θ_n. n multiplications of ω complete a full rotation, hence:

$$\omega^i = \omega^{i+n} \tag{5.7}$$

The inverse of a phasor is its complex conjugate:

$$\omega^{-i} = *\omega^{i} \tag{5.8}$$

The full set of n phasors sum to zero. It is obvious that the imaginary sine components sum to zero because for each phasor its complex conjugate is also present. The zero sum of the real cosine components is not as obvious, but the phasors can also be thought of as vertices of a regular n-gon centered at the origin. The sum of those vertex positions is then the zero vector.

$$0 = \sum_{i=0}^{n-1} \omega^{i} \tag{5.9}$$

The eigenvectors, the columns of F, correspond to stepping through the phasors at different frequencies.

$$F = \begin{bmatrix} 1 & 1 & 1 & & 1 \\ 1 & \omega^{1} & \omega^{2} & \cdots & \omega^{n-1} \\ 1 & \omega^{2} & \omega^{4} & & \omega^{2(n-1)} \\ & \vdots & & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \tag{5.10}$$

The inverse of the eigenvectors can also be written analytically.

$$F^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & & \omega^{-2(n-1)} \\ & \vdots & & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix} \tag{5.11}$$

Applying the DFT to S transforms it to the diagonal eigenvalue matrix Λ.

$$\Lambda = \begin{bmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{n-1} \end{bmatrix} \tag{5.12}$$

In general, the eigenvalues λ_i of a circulant matrix can be imaginary, but the eigenvalues of a symmetric circulant matrix are always real. The edge averaging rules for Loop subdivision form this type of matrix. For an n even sized matrix, a symmetric circulant matrix has following

form and eigenvalues:

$$S_{even} = \begin{bmatrix} a_0 & a_1 & a_2 & & a_{\frac{n}{2}} & & a_2 & a_1 \\ a_1 & a_0 & a_1 & \cdots & a_{\frac{n}{2}-1} & \cdots & a_3 & a_2 \\ a_2 & a_1 & a_0 & & a_{\frac{n}{2}-2} & & a_4 & a_3 \\ & \vdots & & \ddots & \vdots & & \vdots & \\ a_{\frac{n}{2}} & a_{\frac{n}{2}-1} & a_{\frac{n}{2}-2} & \cdots & a_0 & \cdots & a_{\frac{n}{2}-2} & a_{\frac{n}{2}-1} \\ & \vdots & & & \vdots & \ddots & \vdots & \\ a_2 & a_3 & a_4 & \cdots & a_{\frac{n}{2}-2} & \cdots & a_0 & a_1 \\ a_1 & a_2 & a_3 & & a_{\frac{n}{2}-1} & & a_1 & a_0 \end{bmatrix} \tag{5.13}$$

$$i \in [0, n-1] : \lambda_i = a_0 + 2 \sum_{j=1}^{\frac{n}{2}-1} \cos\left(ij\tfrac{2\pi}{n}\right) a_j + \cos(i\pi)\, a_{\frac{n}{2}} \tag{5.14}$$

For an n odd sized matrix, a symmetric circulant matrix has following form and eigenvalues:

$$S_{odd} = \begin{bmatrix} a_0 & a_1 & a_2 & & a_{\frac{n-1}{2}} & a_{\frac{n-1}{2}} & & a_2 & a_1 \\ a_1 & a_0 & a_1 & \cdots & a_{\frac{n-1}{2}-1} & a_{\frac{n-1}{2}} & \cdots & a_3 & a_2 \\ a_2 & a_1 & a_0 & & a_{\frac{n-1}{2}-2} & a_{\frac{n-1}{2}-1} & & a_4 & a_3 \\ & \vdots & & \ddots & \vdots & \vdots & & \vdots & \\ a_{\frac{n-1}{2}} & a_{\frac{n-1}{2}-1} & a_{\frac{n-1}{2}-2} & \cdots & a_0 & a_1 & \cdots & a_{\frac{n-1}{2}-1} & a_{\frac{n-1}{2}} \\ a_{\frac{n-1}{2}} & a_{\frac{n-1}{2}} & a_{\frac{n-1}{2}-1} & \cdots & a_1 & a_0 & \cdots & a_{\frac{n-1}{2}-2} & a_{\frac{n-1}{2}-1} \\ & \vdots & & & \vdots & \vdots & \ddots & \vdots & \\ a_2 & a_3 & a_4 & \cdots & a_{\frac{n-1}{2}-1} & a_{\frac{n-1}{2}-2} & \cdots & a_0 & a_1 \\ a_1 & a_2 & a_3 & & a_{\frac{n-1}{2}} & a_{\frac{n-1}{2}-1} & & a_1 & a_0 \end{bmatrix} \tag{5.15}$$

$$i \in [0, n-1] : \lambda_i = a_0 + 2 \sum_{j=1}^{\frac{n-1}{2}} \cos\left(ij\tfrac{2\pi}{n}\right) a_j \tag{5.16}$$

5.2.1 Interlaced Circulant Matrices

An interlaced circulant matrix has two different rows that are copied and shifted to form the other rows. Catmull-Clark subdivision has alternating rows of face and edge averaging rules, so the subdivision matrix is of this type. In this case, n is half the size of the matrix because pairs of

rows are repeated n times.

$$S = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & & a_{2n-2} & a_{2n-1} \\ b_{2n-1} & b_0 & b_1 & b_2 & & b_{2n-3} & b_{2n-2} \\ a_{2n-2} & a_{2n-1} & a_0 & a_1 & \cdots & a_{2n-4} & a_{2n-3} \\ b_{2n-3} & b_{2n-2} & b_{2n-1} & b_0 & & b_{2n-5} & b_{2n-4} \\ & & \vdots & & \ddots & \vdots & \\ a_2 & a_3 & a_4 & a_5 & \cdots & a_0 & a_1 \\ b_1 & b_2 & b_3 & b_4 & & b_{2n-1} & b_0 \end{bmatrix} \tag{5.17}$$

An interlaced circulant matrix can be transformed into a block diagonal matrix by applying an interlaced DFT matrix and its inverse.

$$\hat{S} = F^{-1}SF \tag{5.18}$$

The interlaced DFT is two versions of the DFT from Equation 5.10 interwoven in a checkerboard pattern as follows:

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & & 0 & 1 \\ 1 & 0 & \omega^1 & 0 & \omega^2 & 0 & & \omega^{n-1} & 0 \\ 0 & 1 & 0 & \omega^1 & 0 & \omega^2 & \cdots & 0 & \omega^{n-1} \\ 1 & 0 & \omega^2 & 0 & \omega^4 & 0 & & \omega^{2(n-1)} & 0 \\ 0 & 1 & 0 & \omega^2 & 0 & \omega^4 & & 0 & \omega^{2(n-1)} \\ & & & \vdots & & & \ddots & \vdots & \\ 1 & 0 & \omega^{n-1} & 0 & \omega^{2(n-1)} & 0 & \cdots & \omega^{(n-1)(n-1)} & 0 \\ 0 & 1 & 0 & \omega^{n-1} & 0 & \omega^{2(n-1)} & & 0 & \omega^{(n-1)(n-1)} \end{bmatrix} \tag{5.19}$$

The alternating pattern makes the two interlaced DFT matrices independent of each other, so the inverse can be trivially written as two interlaced versions of the inverse DFT from Equa-

tion 5.11:

$$F^{-1} = \frac{1}{n}\begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 0 & & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & & 0 & 1 \\
1 & 0 & \omega^{-1} & 0 & \omega^{-2} & 0 & & \omega^{-(n-1)} & 0 \\
0 & 1 & 0 & \omega^{-1} & 0 & \omega^{-2} & \cdots & 0 & \omega^{-(n-1)} \\
1 & 0 & \omega^{-2} & 0 & \omega^{-4} & 0 & & \omega^{-2(n-1)} & 0 \\
0 & 1 & 0 & \omega^{-2} & 0 & \omega^{-4} & & 0 & \omega^{-2(n-1)} \\
 & & & \vdots & & & \ddots & \vdots & \\
1 & 0 & \omega^{-(n-1)} & 0 & \omega^{-2(n-1)} & 0 & \cdots & \omega^{-(n-1)(n-1)} & 0 \\
0 & 1 & 0 & \omega^{-(n-1)} & 0 & \omega^{-2(n-1)} & & 0 & \omega^{-(n-1)(n-1)}
\end{bmatrix} \tag{5.20}$$

The block diagonal matrix $\hat{S}$ resulting from applying the DFT to S has n blocks that are 2×2 matrices.

$$\hat{S} = \begin{bmatrix}
B_0 & & & \\
 & B_1 & & \\
 & & \ddots & \\
 & & & B_{n-1}
\end{bmatrix}, B_i = \begin{bmatrix} \alpha_i & \beta_i \\ \gamma_i & \delta_i \end{bmatrix} \tag{5.21}$$

The eigenvalues $\lambda_{i\pm}$ for a block B_i have the following form:

$$\lambda_{i\pm} = \frac{\alpha_i + \delta_i \pm \sqrt{(\alpha_i - \delta_i)^2 + 4\beta_i\gamma_i}}{2} \tag{5.22}$$

When n is even, the entries of B_i are the following:

$$\alpha_i = a_0 + 2\sum_{j=1}^{\frac{n}{2}-1} \cos\left(ij\tfrac{2\pi}{n}\right) a_{2j} + \cos(i\pi)\, a_n \tag{5.23}$$

$$\delta_i = b_0 + 2\sum_{j=1}^{\frac{n}{2}-1} \cos\left(ij\tfrac{2\pi}{n}\right) b_{2j} + \cos(i\pi)\, b_n \tag{5.24}$$

$$\beta_i = \sum_{j=1}^{\frac{n}{2}} \left(\omega^{i(j-1)} + \omega^{-ij}\right) a_{2j-1} \tag{5.25}$$

$$\gamma_i = \sum_{j=1}^{\frac{n}{2}} \left(\omega^{-i(j-1)} + \omega^{ij}\right) b_{2j-1} \tag{5.26}$$

When n is odd, the entries of B_i are the following:

$$\alpha_i = a_0 + 2\sum_{j=1}^{\frac{n-1}{2}} \cos\left(ij\tfrac{2\pi}{n}\right) a_{2j} \tag{5.27}$$

$$\delta_i = b_0 + 2\sum_{j=1}^{\frac{n-1}{2}} \cos\left(ij\tfrac{2\pi}{n}\right) b_{2j} \tag{5.28}$$

$$\beta_i = \omega^i a_n + \sum_{j=1}^{\frac{n-1}{2}} \left(\omega^{i(j-1)} + \omega^{-ij}\right) a_{2j-1} \tag{5.29}$$

$$\gamma_i = \omega^{-i} b_n + \sum_{j=1}^{\frac{n-1}{2}} \left(\omega^{-i(j-1)} + \omega^{ij}\right) b_{2j-1} \tag{5.30}$$

Note that in both the even and odd cases, β_i and γ_i are imaginary quantities. Fortunately, we are only ever interested in the product of these two quantities, as in Equation 5.22, which is real in the even case or in the odd case when $a_n = b_n = 0$.

$$\beta_i\gamma_i = \left(\sum_{j=1}^{\frac{n}{2}} \left(\omega^{i(j-1)} + \omega^{-ij}\right) a_{2j-1}\right)\left(\sum_{k=1}^{\frac{n}{2}} \left(\omega^{-i(k-1)} + \omega^{ik}\right) b_{2k-1}\right) \tag{5.31}$$

$$= \sum_{j=1}^{\frac{n}{2}}\sum_{k=1}^{\frac{n}{2}} \left(\omega^{i(j-1)} + \omega^{-ij}\right)\left(\omega^{-i(k-1)} + \omega^{ik}\right) a_{2j-1}b_{2k-1} \tag{5.32}$$

$$= \sum_{j=1}^{\frac{n}{2}}\sum_{k=1}^{\frac{n}{2}} \left(\omega^{i((j-1)-(k-1))} + \omega^{-i(j-k)} + \omega^{i((j-1)+k)} + \omega^{-i(j+(k-1))}\right) a_{2j-1}b_{2k-1} \tag{5.33}$$

$$= \sum_{j=1}^{\frac{n}{2}}\sum_{k=1}^{\frac{n}{2}} \left(\left(\omega^{i(j-k)} + \omega^{-i(j-k)}\right) + \left(\omega^{i(j+k-1)} + \omega^{-i(j+k-1)}\right)\right) a_{2j-1}b_{2k-1} \tag{5.34}$$

$$= 2\sum_{j=1}^{\frac{n}{2}}\sum_{k=1}^{\frac{n}{2}} \left(\cos\left(i\,(j-k)\,\tfrac{2\pi}{n}\right) + \cos\left(i\,(j+k-1)\,\tfrac{2\pi}{n}\right)\right) a_{2j-1}b_{2k-1} \tag{5.35}$$

In general, an interlaced circulant matrix with r row types repeated n times is transformed by the interlaced DFT into a block diagonal matrix with n blocks that are $r \times r$ matrices.

5.3 Toeplitz Matrices

A Toeplitz matrix [8] has the same value for all elements along each diagonal parallel to the main diagonal.

$$S = \begin{bmatrix} a_0 & a_1 & a_2 & & a_{n-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{n-2} \\ a_{-2} & a_{-1} & a_0 & & a_{n-3} \\ & \vdots & & \ddots & \vdots \\ a_{-(n-1)} & a_{-(n-2)} & a_{-(n-3)} & \cdots & a_0 \end{bmatrix} \tag{5.36}$$

We are interested in a specific Toeplitz matrix which is a symmetric tridiagonal matrix. An example of this matrix type is the edge averaging rules for Loop subdivision in a wedge between two sharp boundary edges. The size of the matrix is $n-1$ where n is the number of triangles in the wedge.

$$S = \begin{bmatrix} a_0 & a_1 & 0 & 0 & & \\ a_1 & a_0 & a_1 & 0 & & \\ 0 & a_1 & a_0 & a_1 & & \\ & & & \ddots & & \\ & & & a_1 & a_0 & a_1 \\ & & & 0 & a_1 & a_0 \end{bmatrix} \tag{5.37}$$

The eigenvectors of S correspond to the discrete cosine transformation (DCT).

$$SV = V\Lambda \tag{5.38}$$

$$S = V\Lambda V^{-1} \tag{5.39}$$

The cosine terms are in increments of $\frac{\pi}{n}$ which are similar to the nth roots of unity but are instead on the real axis over a half disc domain (Figure 5.1(b)). This half disc is topologically equivalent to the wedge of triangles around a sharp Loop vertex and corresponds to these sharp boundary conditions.

When n is even the eigenvectors are as follows:

$$\alpha(r,c) = \cos r\tfrac{c\pi}{n} \tag{5.40}$$

$$\beta(r,c) = \sin r\tfrac{c\pi}{n} \tag{5.41}$$

$$V = \begin{bmatrix} \alpha\left(-\frac{n-2}{2},1\right) & \beta\left(-\frac{n-2}{2},2\right) & \cdots & \alpha\left(-\frac{n-2}{2},n-1\right) \\ \vdots & \vdots & & \vdots \\ \alpha(0,1) & \beta(0,2) & \cdots & \alpha(0,n-1) \\ \vdots & \vdots & & \vdots \\ \alpha\left(\frac{n-2}{2},1\right) & \beta\left(\frac{n-2}{2},2\right) & \cdots & \alpha\left(\frac{n-2}{2},n-1\right) \end{bmatrix} \tag{5.42}$$

$$V^{-1} = \frac{2}{n}V^t \tag{5.43}$$

When n is odd the eigenvectors are the same, but the row indexing would then be on half fractions instead of integers, so we rewrite it in the following way:

$$\alpha(r,c) = \cos\left(r+\tfrac{1}{2}\right)\tfrac{c\pi}{n} \tag{5.44}$$

$$\beta(r,c) = \sin\left(r+\tfrac{1}{2}\right)\tfrac{c\pi}{n} \tag{5.45}$$

$$V = \begin{bmatrix} \alpha\left(-\frac{n-1}{2},1\right) & \beta\left(-\frac{n-1}{2},2\right) & \cdots & \beta\left(-\frac{n-1}{2},n-1\right) \\ \vdots & \vdots & & \vdots \\ \alpha(-1,1) & \beta(-1,2) & \cdots & \beta(-1,n-1) \\ \alpha(0,1) & \beta(0,2) & \cdots & \beta(0,n-1) \\ \vdots & \vdots & & \vdots \\ \alpha\left(\frac{n-3}{2},1\right) & \beta\left(\frac{n-3}{2},2\right) & \cdots & \beta\left(\frac{n-3}{2},n-1\right) \end{bmatrix} \tag{5.46}$$

$$V^{-1} = \frac{2}{n}V^t \tag{5.47}$$

In either case, the eigenvalue associated with the cth column eigenvector is as follows:

$$c \in [1, n-1] : \lambda_c = a_0 + 2\cos\tfrac{c\pi}{n}\, a_1 \tag{5.48}$$

The following are some useful trigonometry identities for manipulating these eigenvec-

tors:

$$\cos(A+B) = \cos A \cos B - \sin A \sin B \tag{5.49}$$

$$\sin(A+B) = \sin A \cos B + \cos A \sin B \tag{5.50}$$

$$\cos A \cos B = \frac{\cos(A-B) + \cos(A+B)}{2} \tag{5.51}$$

$$\sin A \sin B = \frac{\cos(A-B) - \cos(A+B)}{2} \tag{5.52}$$

$$\cos A \sin B = \frac{\sin(A+B) - \sin(A-B)}{2} \tag{5.53}$$

5.3.1 Interlaced Toeplitz Matrices

We are also interested in the matrix of edge and face averaging rules around a sharp Catmull-Clark vertex. Similar to the generalization from the circulant matrix of smooth Loop to the interlaced matrix of smooth Catmull-Clark, these sharp Catmull-Clark rules can be represented by a special interlaced Toeplitz matrix S. S has two row types and $2n-1$ rows over all, where n is the number of quads in the wedge.

$$S = \begin{bmatrix} a_0 & a_1 & 0 & 0 & 0 & 0 & & \\ b_1 & b_0 & b_1 & b_2 & 0 & 0 & & \\ 0 & a_1 & a_0 & a_1 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{0}} \\ 0 & b_2 & b_1 & b_0 & b_1 & b_2 & & \\ 0 & 0 & 0 & a_1 & a_0 & a_1 & & \\ & & & & \ddots & & & \\ & \multicolumn{3}{c}{\multirow{2}{*}{0}} & b_2 & b_1 & b_0 & b_1 \\ & & & & 0 & 0 & a_1 & a_0 \end{bmatrix} \tag{5.54}$$

S can be transformed to a block diagonal matrix $\hat{S}$ using a transform matrix T which is very similar to the DCT based matrix used in the Toeplitz case.

$$ST = T\hat{S} \tag{5.55}$$

$$S = T\hat{S}T^{-1} \tag{5.56}$$

T is composed of two interlaced matrices. The smaller, embedded matrix is size $n-1 \times n-1$ and is the same as V from the Toeplitz case, Equations 5.42 and 5.46. The outer matrix is size $n \times n$ and is a combination of elements from V. The first column which is the eigenvector associated with

the eigenvalue a_0 is a series of cosine terms.

$$T = \left[\begin{array}{c|cc|c|cc} \cos 0\pi & 0 & v_{1,1} & & 0 & v_{1,n-1} \\ 0 & v_{1,1} & 0 & & v_{1,n-1} & 0 \\ \cos 1\pi & 0 & v_{1,1}+v_{2,1} & & 0 & v_{1,n-1}+v_{2,n-1} \\ 0 & v_{2,1} & 0 & & v_{2,n-1} & 0 \\ \cos 2\pi & 0 & v_{2,1}+v_{3,1} & & 0 & v_{2,n-1}+v_{3,n-1} \\ 0 & v_{3,1} & 0 & \cdots & v_{3,n-1} & 0 \\ & & \vdots & & \vdots & \\ \cos(n-2)\pi & 0 & v_{n-2,1}+v_{n-1,1} & & 0 & v_{n-2,n-1}+v_{n-1,n-1} \\ 0 & v_{n-1,1} & 0 & & v_{n-1,n-1} & 0 \\ \cos(n-1)\pi & 0 & v_{n-1,1} & & 0 & v_{n-1,n-1} \end{array}\right] \tag{5.57}$$

The inverse transform T^{-1} is the transpose of T scaled nonuniformly by the matrix Σ. The inner matrix is scaled by $\frac{2}{n}$, as in the Toeplitz case. The first vector of the outer matrix is scaled by $\frac{1}{n}$, while a vector from block B_i is scaled by $\frac{1}{n\left(1+\cos\frac{i\pi}{n}\right)}$.

$$T^{-1} = \Sigma T^t \tag{5.58}$$

$$\Sigma = \left[\begin{array}{c|cc|} \frac{1}{n} & & \\ \hline & \frac{2}{n} & 0 \\ & 0 & \frac{1}{n\left(1+\cos\frac{1\pi}{n}\right)} \\ \hline & & \ddots \\ & & \begin{array}{cc} \frac{2}{n} & 0 \\ 0 & \frac{1}{n\left(1+\cos\frac{(n-1)\pi}{n}\right)} \end{array} \end{array}\right] \tag{5.59}$$

$\hat{S}$ is a block diagonal matrix with one 1×1 block and $n-1$ blocks B_i of size 2×2 in the form:

$$\hat{S} = \left[\begin{array}{c|c|c} a_0 & & \\ \hline & B_1 & \\ \hline & \ddots & \\ \hline & & B_{n-1} \end{array}\right], B_i = \begin{bmatrix} b_0 + 2\cos\frac{i\pi}{n}\, b_2 & 2\left(1+\cos\frac{i\pi}{n}\right) b_1 \\ a_1 & a_0 \end{bmatrix} \tag{5.60}$$

The eigenvalues of S can be easily solved for within these 2×2 blocks:

$$\lambda_0 = a_0 \tag{5.61}$$

$$\lambda_{i\pm} = \frac{a_0 + b_0 + 2\cos\frac{i\pi}{n}\, b_2 \pm \sqrt{\left(-a_0 + b_0 + 2\cos\frac{i\pi}{n}\, b_2\right)^2 + 8\left(1+\cos\frac{i\pi}{n}\right) a_1 b_1}}{2} \tag{5.62}$$

The associated eigenvectors $\hat{v}_i$ of the transformed matrix $\hat{S}$ have zeros in rows outside of their block, and the two nonzero entries are as follows:

$$\hat{v}_{i\pm} = \begin{bmatrix} \frac{-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2 \pm \sqrt{\left(-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2\right)^2+8\left(1+\cos\frac{i\pi}{n}\right)a_1b_1}}{2a_1} \\ 1 \end{bmatrix} \tag{5.63}$$

These eigenvectors can be transformed back by T to give the eigenvectors v_i of S.

$$V = T\hat{V} \tag{5.64}$$

$$v_0 = \begin{bmatrix} \cos 0\pi \\ 0 \\ \cos 1\pi \\ 0 \\ \cos 2\pi \\ 0 \\ \vdots \\ \cos(n-2)\pi \\ 0 \\ \cos(n-1)\pi \end{bmatrix}, v_{i\pm} = \begin{bmatrix} v_{i,1} \\ v_{i,1}\frac{-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2 \pm \sqrt{\left(-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2\right)^2+8\left(1+\cos\frac{i\pi}{n}\right)a_1b_1}}{2a_1} \\ v_{i,1}+v_{i,2} \\ v_{i,2}\frac{-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2 \pm \sqrt{\left(-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2\right)^2+8\left(1+\cos\frac{i\pi}{n}\right)a_1b_1}}{2a_1} \\ v_{i,2}+v_{i,3} \\ \vdots \\ v_{i,n-2}+v_{i,n-1} \\ v_{i,n-1}\frac{-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2 \pm \sqrt{\left(-a_0+b_0+2\cos\frac{i\pi}{n}\ b_2\right)^2+8\left(1+\cos\frac{i\pi}{n}\right)a_1b_1}}{2a_1} \\ v_{i,n-1} \end{bmatrix} \tag{5.65}$$

5.4 Conclusion

We have shown the derivation of the eigen-decomposition of circulant and Toeplitz matrices that correspond to the edge and face averaging rules near smooth, spike, corner, and crease vertices in Loop and Catmull-Clark subdivision surfaces. This provides an easy way to analyze more general schemes with modified edge rules. In Chapters 6-8, we will use the eigen-decompositions presented here to perform exact evaluation of piecewise smooth Loop and Catmull-Clark surfaces.

Chapter 6

Subdivision Exact Evaluation

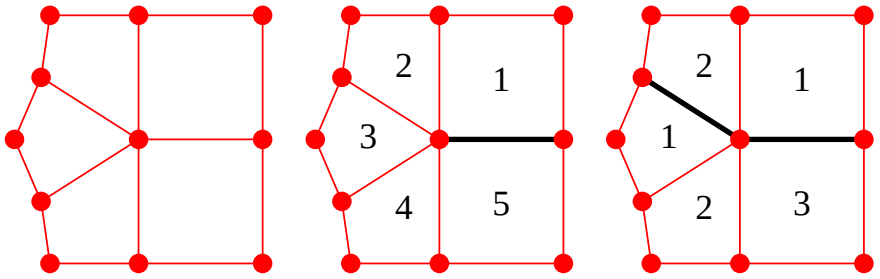
6.1 Introduction

In a Chapter 3, we described subdivision surfaces as a two step process: a topological split followed by local geometric averaging. This view is very straight forward to understand, but it does not provide a parameterization of the limit surface that can be evaluated at an arbitrary parametric location with a constant amount of work. In order to perform constant-time exact evaluation, a subdivision scheme must have the following two properties. First, the subdivision operator must at each iteration introduce new regions that can be described by known parameteric functions, and in the limit, the union of these regions must converge to cover the entire surface area. Second, it must be possible to perform an arbitrary number of applications of the subdivision operator in a constant amount of time.

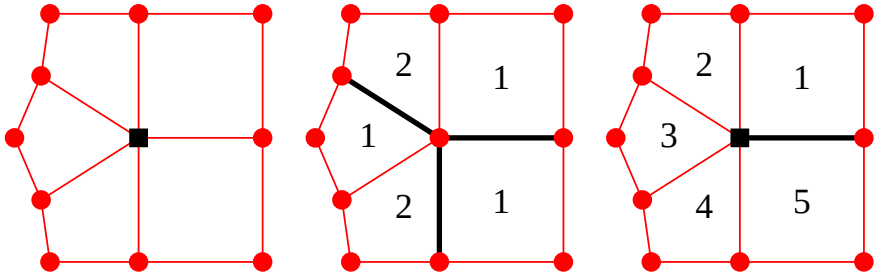
A previous method [29, 28] has shown that these properties hold for the smooth Loop and Catmull-Clark subdivision scheme. We now show how to extend this work to handle the remaining cases of patches near infinitely sharp features.

6.2 Matrix Formulation

The matrix formulation of the exact evaluation is very similar for the Catmull-Clark and Loop schemes. The the extraordinary vertex cases that we will consider are: smooth, spike, corner, crease, and dart patches [10, 6]. For example, Figure 6.1 shows the 1-ring neighborhood for each of these extraordinary vertex types for Catmull-Clark subdivision. The function $\mathbf{s}(u, v)$ evaluates a subdivision patch at a parametric location $(u, v) \in \{[0, 1], [0, 1]\}$ other than the origin. The origin



(a) *Smooth* (b) *Dart* (c) *Crease*



(d) *Spike* (e) *Corner* (f) *Corner*

Figure 6.1: *Examples of the five different extraordinary vertex types for piecewise smooth Catmull-Clark surfaces. Black thick lines and squares denote sharp edges and vertices. Light thin lines and circles represent smooth edges and vertices. The numbers indicate the patch number p within each wedge; the maximum number is the valence n of the wedge.*

of the patch is the extraordinary vertex (Figure 6.1), either a smooth vertex of extraordinary valence or a spike, corner, crease, or dart vertex of any valence. Conceptually, $g = f(\log_2 u, \log_2 v)$ subdivisions are applied such that the parametric point falls in one of the known spline patches ($\{U, UV, V\}$ in Figure 6.2). In matrix form, this is equivalent to applying A, the 1-patch-ring subdivision matrix from Figure 6.3, $g-1$ times, followed by one application of the extended subdivision matrix $\bar{A}$ to the set of control points C to generate all control points necessary for any of the regular spline subpatches. A picking matrix P_k, where k specifies the subpatch related to the given parametric location, then chooses the correct set of spline control points. Note that for a patch adjacent to a sharp edge, the picking matrix will include rules for generating the appropriate phantom vertices, see Figure 6.4(b). Once the spline control points have been generated and picked, they are weighted by a row vector containing the spline blending functions $B(u, v)$ evaluated at the correctly transformed subpatch parameters $(\bar{u}, \bar{v})$, as shown in the following equation:

$$\mathbf{s}(u, v) = B(\bar{u}, \bar{v}) P_k \bar{A} A^{g-1} C \tag{6.1}$$

For constant-time evaluation, the term A^{g-1} must be reduced to a constant number of matrix multiplications, which we accomplish using the Jordan decomposition, see Appendix B.3. Note that in the case of a diagonalizable matrix, the Jordan decomposition is the same as the eigen-decomposition. We also substitute a power basis vector $p_{uv}(\bar{u}, \bar{v})$ multiplied by a spline coefficient matrix B_{uv} for the spline blending function vector $B(\bar{u}, \bar{v})$. The vector $p_{uv}(\bar{u}, \bar{v})$ is very quick and easy to build and differentiate. Note that differentiating the patch reduces to differentiating $p_{uv}(\bar{u}, \bar{v})$. The expanded matrix formula is:

$$\mathbf{s}(u, v) = p_{uv}(\bar{u}, \bar{v}) B_{uv} P_k \bar{A} V J^{g-1} V^{-1} C \tag{6.2}$$

To reduce the constant number of matrix multiplications necessary at evaluation time, we introduce the three matrices $Q_k = B_{uv} P_k \bar{A} V$ dependent on the subpatch k, and also store the projected control points $\hat{C} = V^{-1} C$ for each patch in advance. This results in the simplified equation:

$$\mathbf{s}(u, v) = p_{uv}(\bar{u}, \bar{v}) Q_k J^{g-1} \hat{C} \tag{6.3}$$

Our algorithm is table driven, where we generate the matrices Q_k, J, and V^{-1} off-line for each subdivision scheme, patch type, valence n, and patch p in the case of the wedge based types of corners, creases, and darts.

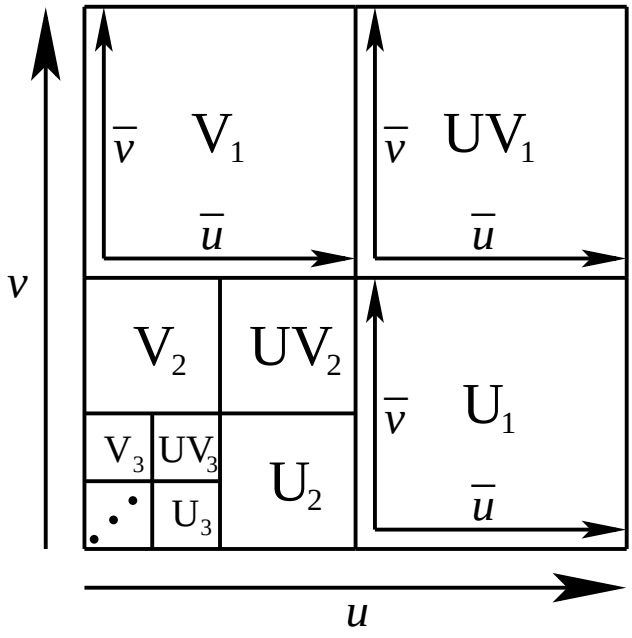


Figure 6.2: *Catmull-Clark parameterization*

6.2.1 Jordan Decomposition of A

The major hurdle we faced in implementing our algorithm was generating the Jordan decomposition of the 1-patch-ring subdivision matrix A for certain patch types. For all patch types, A is dependent on the valence n of the extraordinary vertex or wedge. For the corner, crease, and dart cases, A also depends on which patch p within the wedge is being evaluated, while for the smooth and spike cases p can be ignored. In all cases, $A_{n,p}$ is a block lower triangular matrix. The extended subdivision matrix $\bar{A}_{n,p}$ is simply $A_{n,p}$ extended to generate one additional layer of control points.

$$A_{n,p} = \begin{bmatrix} S_{0,n} & 0 \\ S_{11,n,p} & S_{12,n,p} \end{bmatrix} \tag{6.4}$$

$$\bar{A}_{n,p} = \begin{bmatrix} S_{0,n} & 0 \\ S_{11,n,p} & S_{12,n,p} \\ S_{21,n,p} & S_{22,n,p} \end{bmatrix} \tag{6.5}$$

The Jordan decomposition of A can be solved by combining the Jordan decompositions of $S_{0,n}$ (the 1-ring subdivision matrix) and the eigen-decomposition of $S_{12,n,p}$:

$$S_0 V_0 = V_0 J_0 \tag{6.6}$$

$$S_{12} V_{12} = V_{12} \Lambda_{12} \tag{6.7}$$

We know the block form of the generalized eigenvector matrix V of A:

$$AV = VJ \tag{6.8}$$

$$\begin{bmatrix} S_0 & 0 \\ S_{11} & S_{12} \end{bmatrix} \begin{bmatrix} V_0 & 0 \\ V_{11} & V_{12} \end{bmatrix} = \begin{bmatrix} V_0 & 0 \\ V_{11} & V_{12} \end{bmatrix} \begin{bmatrix} J_0 & 0 \\ 0 & \Lambda_{12} \end{bmatrix} \tag{6.9}$$

$$\begin{bmatrix} S_0 V_0 & 0 \\ S_{11} V_0 + S_{12} V_{11} & S_{12} V_{12} \end{bmatrix} = \begin{bmatrix} V_0 J_0 & 0 \\ V_{11} J_0 & V_{12} \Lambda_{12} \end{bmatrix} \tag{6.10}$$

The bottom left blocks of the matrices in Equation 6.10 defines V_{11}. We move V_{11} to the left side of the equation:

$$S_{11} V_0 + S_{12} V_{11} = V_{11} J_0 \tag{6.11}$$

$$V_{11} J_0 - S_{12} V_{11} = S_{11} V_0 \tag{6.12}$$

J_0 is mainly diagonal with some 1's (but in our cases at most one) on the subdiagonal. From Equation B.69, we can substitute $J_0 = \Lambda_0 + L_0$ which is a diagonal matrix plus a subdiagonal matrix.

$$V_{11}(\Lambda_0 + L_0) - S_{12}V_{11} = S_{11}V_0 \tag{6.13}$$

$$V_{11}\Lambda_0 - S_{12}V_{11} = S_{11}V_0 - V_{11}L_0 \tag{6.14}$$

The vectors $v_{11,c}$, which is column c of V_{11}, can be solved for from right to left, as a subdiagonal 1 can place a dependence on $v_{11,c+1}$. $L_0[c+1,c]$ is the entry of J_0 below the diagonal element λ_c, and it is either 1 or 0.

$$\lambda_c v_{11,c} - S_{12}v_{11,c} = S_{11}v_{0,c} - L_0[c+1,c]\, v_{11,c+1} \tag{6.15}$$

$$(\lambda_c I - S_{12})\, v_{11,c} = S_{11}v_{0,c} - L_0[c+1,c]\, v_{11,c+1} \tag{6.16}$$

$$v_{11,c} = (\lambda_c I - S_{12})^{-1}(S_{11}v_{0,c} - L_0[c+1,c]\, v_{11,c+1}) \tag{6.17}$$

Equation 6.17 is dependent on inverting $(\lambda_c I - S_{12})$. $(\lambda_c I - S_{12})$ is singular only if λ_c is an eigenvalue S_{12}. For the cases that we are considering, S_0 and S_{12} do not share any eigenvalues. $S_{12,n,p}$ is composed of averaging rules in regular spline regions of the subdivision surface, so its eigen-decomposition is trivial to calculate. So, if we can derive the Jordan decomposition of $S_{0,n}$ then we will have the decomposition for $A_{n,p}$.

6.2.2 Jordan Decomposition of S_0

For both Catmull-Clark and Loop subdivision, $S_{0,n}$ is non-defective for the smooth, spike, and dart cases, so it can be diagonalized into its eigen-decomposition. The difficult cases are the corner and crease patch cases, where for certain valences, $S_{0,n}$ is defective, and a nontrivial Jordan form must be used. Our experience using Mathematica [31] was that for small valences $n < 5$, it was able to produce the correct Jordan decompositions analytically, but as the valence increased, the computation became intractable. Mathematica's numerical solver gave quick results for all valences, but it returned a diagonal matrix $J = \Lambda$ with multiple eigenvectors that differed by ϵ, making the eigenvector matrix $V_{0,n}$ noninvertible and thus the decomposition invalid. Fortunately in these cases, the ordering of control vertices can be chosen to make $S_{0,n}$ become a block lower triangular matrix:

$$S_{0,n} = \begin{bmatrix} T & 0 \\ T_{11,n} & T_{12,n} \end{bmatrix} \tag{6.18}$$

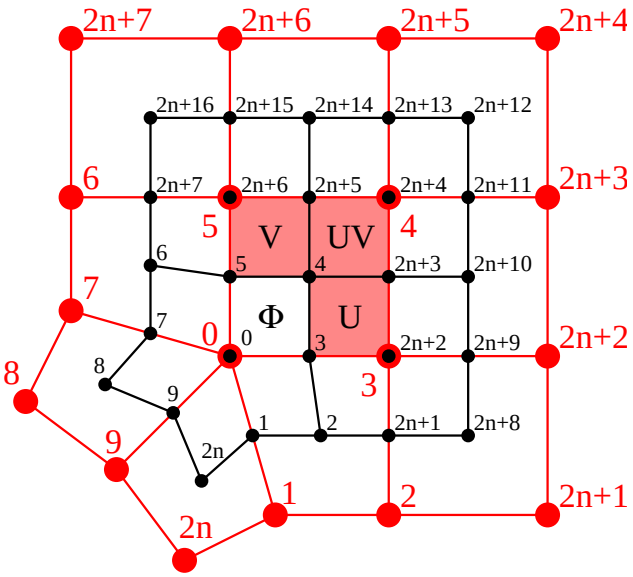


Figure 6.3: *Application of the $\bar{A}$ matrix to a quad patch next to a smooth extraordinary vertex produces a ribbon of three subpatches $\{U, UV, V\}$ that are regular B-spline patches and a fourth patch Φ which is analogous to its parent.*

The Jordan decomposition of $S_{0,n}$ can then be derived by combining the eigen-decompositions of T and $T_{12,n}$. Note that any eigenvector of $T_{12,n}$ can be prepended with zeros, and it will then also be an eigenvector of $S_{0,n}$. For the eigenvectors associated with the eigenvalues of T, the top elements $\overline{v_1}$ of v_1 are taken from the corresponding eigenvector from T. We can solve for the remaining lower elements $\underline{v_1}$ of the Jordan vector either using the eigenvector equation

$$S_{0,n}v_1 = \lambda v_1 \tag{6.19}$$

or if a Jordan block is necessary, the generalized eigenvector equation where for the cases we are considering v_2 is the single eigenvector of $T_{12,n}$ corresponding to λ:

$$S_{0,n}v_1 = \lambda v_1 + v_2 \tag{6.20}$$

When beginning to solve this equation for v_1, it is unknown whether the matrix is defective or not, so we combine the two equations by introducing a scalar β which will either be 0 in the eigenvector case or 1 in the generalized eigenvector case:

$$S_{0,n}v_1 = \lambda v_1 + \beta v_2 \tag{6.21}$$

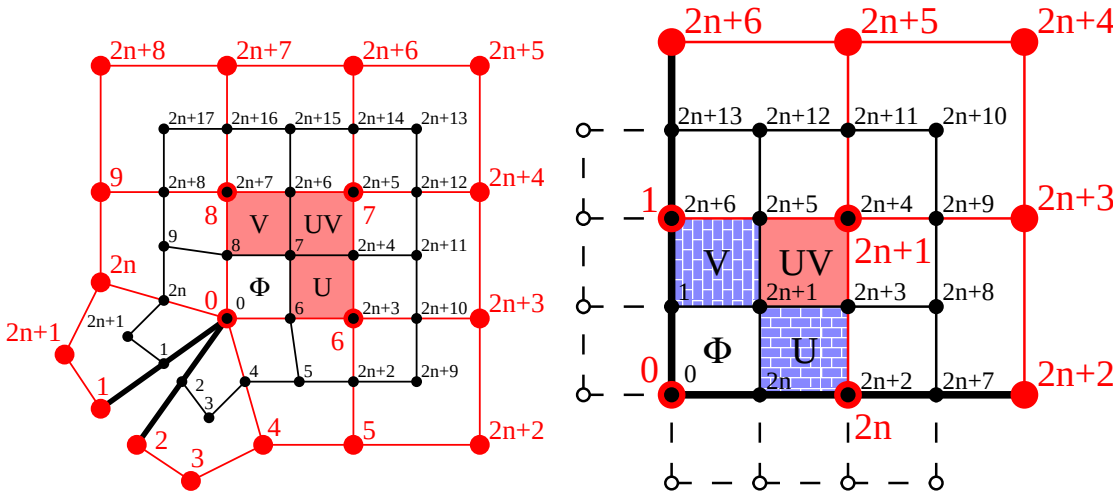


Figure 6.4: *Application of the $\bar{A}$ matrix to a quad patch next to an extraordinary vertex produces a ribbon of three subpatches $\{U, UV, V\}$ that are regular B-spline patches and a fourth patch Φ which is analogous to its parent. Patches not adjacent to any sharp edges (a) generate B-spline subpatches with a full set of 16 control points, while patches adjacent to sharp edges generate B-spline subpatches, e.g. $\{U, V\}$ in (b), that must calculate phantom vertices. Light vertices and edges represent the current generation, while black ones represent its child mesh.*

We now use the use the lower block triangle form of $S_{0,n}$ to derive an equation for $\underline{v_1}$:

$$S_{0,n} v_1 = \lambda v_1 + \beta v_2 \tag{6.22}$$

$$(S_{0,n} - \lambda I)\, v_1 = \beta v_2 \tag{6.23}$$

$$(T - \lambda I)\, \overline{v_1} = 0 \tag{6.24}$$

$$T_{11,n}\overline{v_1} + (T_{12,n} - \lambda I)\, \underline{v_1} = \beta \underline{v_2} \tag{6.25}$$

$$(T_{12,n} - \lambda I)\, \underline{v_1} = \beta \underline{v_2} - T_{11,n}\overline{v_1} \tag{6.26}$$

Solving for $\underline{v_1}$ requires the inverse of $(T_{12,n} - \lambda I)$ as shown in Equation B.57:

$$(T_{12,n} - \lambda I)^{-1} = V_{12,n}\, (\Lambda_{12,n} - \lambda I)^{-1}\, V_{12,n}^{-1} \tag{6.27}$$

If the eigenvalue λ of T is not an eigenvalue of $T_{12,n}$ then $(T_{12,n} - \lambda I)$ is invertible. In this case, the generalized eigenvector equation is not valid because an eigenvector v_2 of $T_{12,n}$ corresponding to the eigenvalue λ does not exist. Instead, the normal eigenvector equation is used, which can be

accomplished by letting $\beta = 0$.

$$\underline{v_1} = -\left(T_{12,n} - \lambda I\right)^{-1} T_{11,n}\overline{v_1} \tag{6.28}$$

$$\underline{v_1} = -V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)^{-1} V_{12,n}^{-1} T_{11,n}\overline{v_1} \tag{6.29}$$

Equation 6.29 defines the lower elements of the eigenvectors corresponding to λ in T if $S_{0,n}$ is not defective.

A generalized Jordan vector will only occur in cases when the eigenvalue λ of T is also an eigenvalue of $T_{12,n}$. For the cases we are considering, the multiplicity of λ in $T_{12,n}$ is at most 1, so $\underline{v_2}$ is the eigenvector of $T_{12,n}$ corresponding to λ. In this case, $(T_{12,n} - \lambda I)$ is a noninvertible matrix, so we use its pseudoinverse $(T_{12,n} - \lambda I)^{\dagger}$ instead to solve for $\underline{v_1}$.

$$\underline{v_1} = \left(T_{12,n} - \lambda I\right)^{\dagger}\left(\beta\underline{v_2} - T_{11,n}\overline{v_1}\right) \tag{6.30}$$

$$\underline{v_1} = V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)^{\dagger} V_{12,n}^{-1}\left(\beta\underline{v_2} - T_{11,n}\overline{v_1}\right) \tag{6.31}$$

$\underline{v_2}$ is one of the columns of $V_{12,n}$, so $V_{12,n}^{-1}\underline{v_2}$ is a column vector with a single entry of 1 and the rest 0. This row with the 1 corresponds to the diagonal entry of $(\Lambda_{12,n} - \lambda I)$ that is 0, hence:

$$\left(\Lambda_{12,n} - \lambda I\right)^{\dagger} V_{12,n}^{-1}\underline{v_2} = 0 \tag{6.32}$$

So the equation for $\underline{v_1}$ reduces to:

$$\underline{v_1} = -V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)^{\dagger} V_{12,n}^{-1} T_{11,n}\overline{v_1} \tag{6.33}$$

This is only a partial solution, so it is necessary to substitute $\underline{v_1}$ back into Equation 6.26. This creates a term which is $(T_{12,n} - \lambda I)$ right multiplied by its pseudoinverse, Equation 6.35. Looking at this term in its eigen-decomposition, we find $(\Lambda_{12,n} - \lambda I)$ right multiplied by its pseudoinverse which is the identity matrix with the diagonal element corresponding to $\underline{v_2}$ set to zero. This near identity matrix zeroes out $\underline{v_2}$ from $V_{12,n}$. The right multiplication by $V_{12,n}^{-1}$ results in the identity matrix minus the outer product of $\underline{v_2}$ and its corresponding row vector $\underline{v_2}^{-1}$ in $V_{12,n}^{-1}$. Subtracting

that quantity from the identity matrix simply yields the outer product $\underline{v_2}\left(\underline{v_2}^{-1}\right)^t$.

$$(T_{12,n} - \lambda I)\,\underline{v_1} = \beta\underline{v_2} - T_{11,n}\overline{v_1} \tag{6.34}$$

$$\left(V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)V_{12,n}^{-1}\right)\left(-V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)^{\dagger}V_{12,n}^{-1}T_{11,n}\overline{v_1}\right) = \beta\underline{v_2} - T_{11,n}\overline{v_1} \tag{6.35}$$

$$-\left(V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)\left(\Lambda_{12,n} - \lambda I\right)^{\dagger}V_{12,n}^{-1}\right)T_{11,n}\overline{v_1} = \beta\underline{v_2} - T_{11,n}\overline{v_1} \tag{6.36}$$

$$\left(I - \left(V_{12,n}\left(\Lambda_{12,n} - \lambda I\right)\left(\Lambda_{12,n} - \lambda I\right)^{\dagger}V_{12,n}^{-1}\right)\right)T_{11,n}\overline{v_1} = \beta\underline{v_2} \tag{6.37}$$

$$\underline{v_2}\left(\underline{v_2}^{-1}\right)^t T_{11,n}\overline{v_1} = \beta\underline{v_2} \tag{6.38}$$

$$\left(\underline{v_2}^{-1}\right)^t T_{11,n}\overline{v_1} = \beta \tag{6.39}$$

Equation 6.39 is a scalar equation that is the final constraint. If m, the multiplicity of λ in T, linearly independent eigenvectors of T represented by $\overline{v_1}$ can be constructed to satisfy the condition $\beta = 0$, then $S_{0,n}$ is not defective. Otherwise for the first generalized Jordan vector $\overline{v_1}$ must be constructed such that $\beta = 1$. For the cases we are considering, there is at most one generalized eigenvector necessary, so the largest Jordan block will be a single 2×2 block.

6.3 Darts

The real subdivison matrix $A_{n,q}$ is numerically diagonalizable for dart patches, but it has two complementary imaginary eigenvalues for some valences, which also causes the V and V^{-1} matrices to have imaginary values. Handling imaginary values increases the cost of doing a surface evaluation by a constant factor of 4.

$$\mathbf{s}\left(u,v\right) \tag{6.40}$$

$$= p_{uv}\left(u,v\right)B_{uv}P_k\bar{A}V\Lambda^{g-1}V^{-1}C \tag{6.41}$$

$$= p_{uv}\left(u,v\right)\left(Q_{k,a} + \mathbf{i}Q_{k,b}\right)\left(\Lambda_a + \mathbf{i}\Lambda_b\right)^{g-1}\left(\hat{C}_a + \mathbf{i}\hat{C}_b\right) \tag{6.42}$$

$$= \left(p_{uv}\left(u,v\right)_a + \mathbf{i}p_{uv}\left(u,v\right)_b\right)\left(\Lambda_a + \mathbf{i}\Lambda_b\right)^{g-1}\left(\hat{C}_a + \mathbf{i}\hat{C}_b\right) \tag{6.43}$$

$$= \left(a_1 + \mathbf{i}b_1\right)\left(a_2 + \mathbf{i}b_2\right)\left(\hat{C}_a + \mathbf{i}\hat{C}_b\right) \tag{6.44}$$

$$= \left(\left(a_1a_2 - b_1b_2\right) + \mathbf{i}\left(a_1b_2 + b_1a_2\right)\right)\left(\hat{C}_a + \mathbf{i}\hat{C}_b\right) \tag{6.45}$$

$$= \left(a_3 + \mathbf{i}b_3\right)\left(\hat{C}_a + \mathbf{i}\hat{C}_b\right) \tag{6.46}$$

$$= \left(a_3\hat{C}_a - b_3\hat{C}_b\right) + \mathbf{i}\underbrace{\left(a_3\hat{C}_b + b_3\hat{C}_a\right)}_{0} \tag{6.47}$$

Note that the value of $\mathbf{s}(u,v)$ is real, so the zero coefficient imaginary term in the final multiply does not need to be calculated. Raising an imaginary eigenvalue $\lambda = a + \mathbf{i}b$ to an integer power g is accomplished in $O(g)$ scalar operations with the sum:

$$\lambda^g = (a + \mathbf{i}b)^g = \sum_{j=0}^{g} \mathbf{i}^j \binom{g}{j} a^{g-j} b^j \tag{6.48}$$

6.4 Conclusion

This chapter has shown the general form for subdivision surface evaluation algorithms and the necessary matrix decompositions. Chapters 7 and 8 will derive the specific matrices for the Loop and Catmull-Clark subdivision schemes, respectively. Each chapter will provide all of the necessary matrices to perform evaluation on patches near smooth, spike, dart, corner, and crease extraordinary vertex cases.

Chapter 7

Loop Evaluation

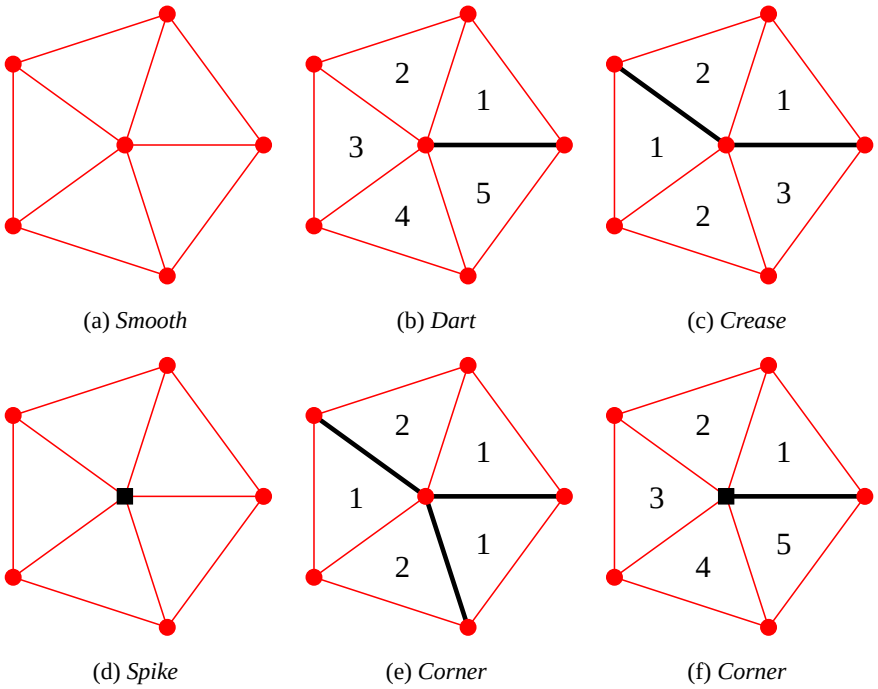
7.1 Introduction

Constant-time evaluation of Loop subdivision surfaces [15] near smooth extraordinary vertices [27] greatly increased the ease of use of this free-form surface primitive in graphics and CAD. The per triangle (u, v) parameterization has made subdivision surfaces compatible with standard CAD primitives such as NURBS. Sharp and semi-sharp edge and vertex averaging rules [10, 6, 4] make Loop surfaces an even more powerful and descriptive modeling primitive for CAD, allowing for intentional breaks in continuity to define sharp features. We present a new method for evaluation of the original piecewise smooth Loop subdivision scheme [10] based on the Jordan decomposition of the local subdivision operator.

7.2 Basic Approach

The smooth Loop evaluation [27] has shown that Loop subdivision meets the two necessary properties to enable constant-time exact evaluation. First, the subdivision operator at each iteration introduces new regions that can be described by the known parameteric functions of box splines, and in the limit, the union of these regions converge to cover the entire surface area. Second, it is possible to perform an arbitrary number of applications of the subdivision operator in a constant amount of time using its Jordan decomposition. We now show how to extend this work to handle the remaining cases of patches near infinitely sharp features.

The averaging rules of the Loop subdivision scheme generalize $N_{2,2,2}$ quartic box spline surfaces to meshes with vertices of arbitrary valence. In the regular regions of the mesh where the



(a) *Smooth* (b) *Dart* (c) *Crease*

(d) *Spike* (e) *Corner* (f) *Corner*

Figure 7.1: *Examples of the five different extraordinary vertex types for piecewise smooth Loop surfaces. Black thick lines and squares denote sharp edges and vertices. Light thin lines and circles represent smooth edges and vertices. The numbers indicate the patch number p within each wedge; the maximum number is the valence n of the wedge.*

vertices are all valence $n = 6$, the surface is a $N_{2,2,2}$ quartic box spline (Figure 7.4(a)). Smooth extraordinary vertices have valence $n \neq 6$ (Figure 7.1(a)), and they are averaged by a valence-specific smoothing rule. The subdivision operator near one of these vertices creates ever shrinking rings of regular box spline patches that converge to the limit position of that vertex [22]. The local averaging rules can be expressed by the extended subdivision matrix $\bar{A}$, and each application of it generates the control points for three regular box spline patches $\{U, UV, V\}$ and a fourth patch Φ that is a contracted topological copy of its parent and which then may get partitioned again in the same manner (Figure 7.5). Any barycentric location (u, v, w) with $u, v, w \in [0, 1]$ and $u+v+w = 1$ within the patch, other than the origin, can be evaluated by repeatedly subdividing a number of generations $g = \lceil -\log_2 (u + v) \rceil$, at which time the parametric point is contained in one of the three regular box spline subpatches (Figure 7.2). Then the transformed parametric values $(\bar{u}, \bar{v})$ can

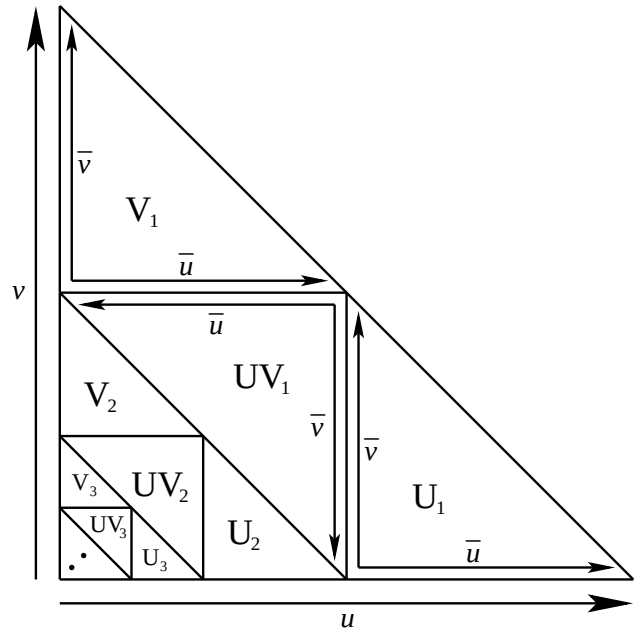


Figure 7.2: *Loop parameterization*

be substituted into the box spline polynomial to compute the result.

$$U_g : u > 2^{-g} : \bar{u} = 2^g u - 1$$
$$\bar{v} = 2^g v \tag{7.1}$$
$$V_g : v > 2^{-g} : \bar{u} = 2^g u$$
$$\bar{v} = 2^g v - 1 \tag{7.2}$$
$$UV_g : u, v \leq 2^{-g} : \bar{u} = 1 - 2^g u$$
$$\bar{v} = 1 - 2^g v \tag{7.3}$$

For a stationary scheme such as this, performing g subdivisions is equivalent to raising the constant 1-patch-ring subdivision matrix A to the power g, which can be achieved in constant-time by diagonalizing A into its eigen-decomposition.

For the smooth Loop subdivision rules, the A matrix is stationary for the regular neighborhood of a patch with a single extraordinary valence vertex. After at most two complete subdivisions of a closed, smooth Loop mesh, all of the patches have such a regular neighborhood, so this technique can be applied. The same technique with the same A matrix can be used to perform exact evaluation of Loop surfaces with semi-sharp edge tags [6]. Semi-sharp edges make the subdivision scheme non-stationary, but only for a finite number of generations, after which the entire mesh

becomes a smooth Loop surface. Infinitely sharp edges and vertices on the other hand are present throughout the subdivision generations, so a new method must be introduced.

7.2.1 Behavior Near Sharp Features

We extend the smooth method to the case of piecewise smooth Loop subdivision. Vertices along chains of sharp tagged edges are averaged by the uniform cubic B-spline curve rules rather than the surface averaging rules. The averaging of these crease vertices is only dependent on vertices along the curve, and this curve creates a sharp boundary across which no surface smoothing information is passed. These modified averaging rules lead to four additional extraordinary vertex cases, and we will address the evaluation of patches near vertices of these types.

A dart vertex (Figure 7.1(b)) has one sharp edge incident on it, but it is still averaged by the smooth Loop vertex rule, forming a transition from the feature curve to the smooth surface. A crease vertex (Figure 7.1(c)) has two sharp edges incident on it, and it is smoothed by the B-spline subdivision curve rule. A spike vertex (Figure 7.1(d)) is tagged not to move, while the rest of the surface around it follows the standard Loop rules, leading to geometry resembling the tip of a cone. A corner vertex is defined to be a vertex with 3 or more sharp edges incident on it (Figure 7.1(e)), or a vertex with 1 or more sharp edges where the vertex is also tagged to not move (Figure 7.1(f)). Similar to the smooth extraordinary vertex case, the behavior of the surface near these extraordinary vertices is a shrinking ribbon of box spline patches (Figure 7.3), (details in Section 7.4).

For any triangle patch p next to an extraordinary vertex, the behavior of the ribbon of box spline patches as they converge to the extraordinary vertex is defined by the subdivision rules of the local neighborhood, which can be written in a subdivision matrix A (Figure 7.3). For a patch adjacent to a smooth or spike vertex, A is dependent on the valence n of the vertex, but it is the same for any adjacent patch p, due to the structural symmetry around these types of vertices. Sharp edges prevent smoothing across them, so the behavior of the surface adjacent to dart, crease, or corner vertices is broken into isolated wedges. A wedge is a radially consecutive set of triangles starting and ending at a sharp edge. The number of triangles in a wedge is its valence n, and each triangular patch is identified by a number p counting counterclockwise around the extraordinary vertex. The subdivision matrix A is different for each patch p of a wedge and also depends on the valence n of the wedge. There is mirror symmetry about the central patch or edge, so for example A for patch $p = 1$ is the mirror of patch $p = n$ (Figure 7.1(f)). This symmetry reduces the number of cases.

For some valences, A is defective and cannot be diagonalized, so it is not possible to use

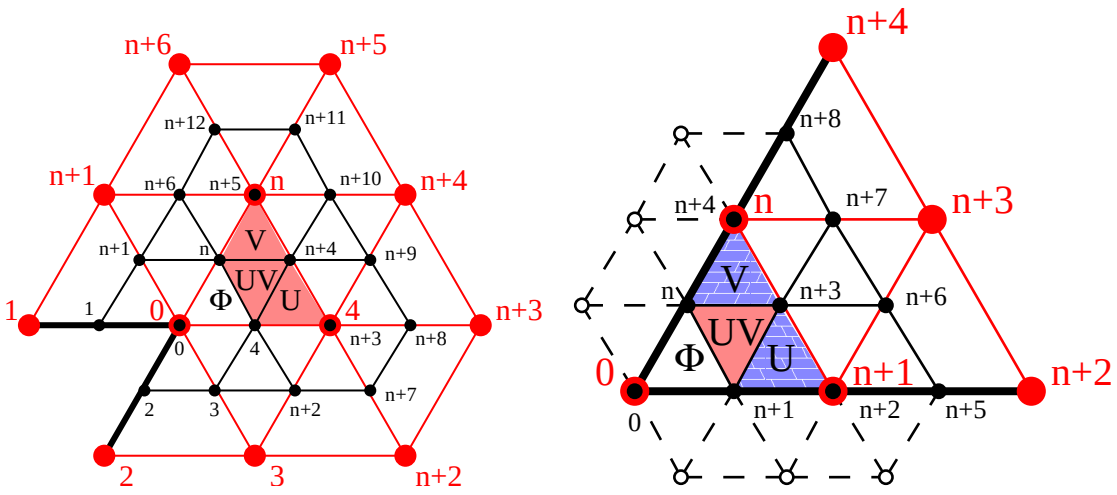


Figure 7.3: *Application of the $\bar{A}$ matrix to a triangle patch next to an extraordinary vertex produces a ribbon of three subpatches $\{U, UV, V\}$ that are regular box spline patches and a fourth patch Φ which is analogous to its parent. Patches not adjacent to any sharp edges (a) generate box spline subpatches with a full set of 12 control points, while patches adjacent to sharp edges generate box spline subpatches, e.g. $\{U, V\}$ in (b), that must calculate phantom vertices. Light vertices and edges represent the current generation, while black ones represent its child mesh.*

the eigen-decomposition to apply A^g. In these cases, we utilize the Jordan decomposition of A which always exists and is only slightly more expensive to raise to a power g.

We show that the subdivision rules near sharp features produce box spline patches and can be applied for an arbitrary number of generations by providing the Jordan decompositions of the subdivision matrices for the different extraordinary vertex types.

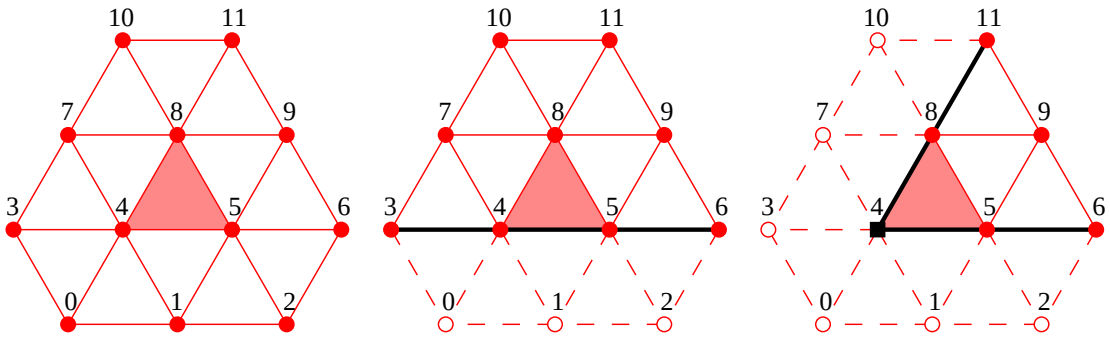
7.3 Related Work

Our method is a generalization of the exact evaluation of smooth Loop surfaces [27]. Infinitely sharp edge and vertex rules [10, 6] change the subdivision matrix A, and for some cases these matrices cannot be diagonalized. Parameterized expanded tagging rules with more control have also been defined [4], and follow up work [34] has shown how to evaluate Loop subdivision surfaces with these extended rules. Some of these subdivision matrices are defective, and the authors chose to use a novel matrix decomposition rather than the Jordan decompositions which correspond to singularities and are hard to parameterize over a family of matrices. Though we only handle the subset these averaging rules, we still do implement the full RendMan standard [21] which is a very

useful set of rules. The Jordan decomposition uses less matrix components because it has at most a single 2×2 block while the other method has many 6×6 blocks. Hence our method is faster at run-time.

7.4 Regular Box Spline Cases

To show that Loop surfaces with infinitely sharp edges and vertices can be exactly evaluated in constant-time, we show that the subdivision operator A produces uniform quartic box spline patches. In the limit, these box spline subpatches completely tile the extraordinary patch, so the entire subdivision surface is a cascading collection of box spline patches at different levels of subdivision.



(a) *Smooth* (b) *Crease* (c) *Corner*

Figure 7.4: *Loop patches that are uniform quartic box spline patches. Black thick lines and boxes are sharp edges and vertices. Light solid lines and circles are smooth edges and vertices. Light dashed lines and hollow circles are phantom edges and vertices.*

Figure 7.5 shows that subdividing a patch near a smooth extraordinary vertex creates shrinking rings of uniform quartic box spline patches $\{U, V, UV\}$, which are known polynomial functions, enabling exact evaluation. For many patches near sharp features, the same kind of box spline patches are generated during the subdivision process (Figure 7.3(a)). For patches adjacent to sharp edges, on the other hand, some of the subpatches will then be adjacent to sharp edges with an incomplete neighborhood of control points to define a box spline patch, e.g. subpatches U and V in Figure 7.3(b).

Fortunately, these subpatches are also uniform quartic box spline patches, and a set of phantom vertices can be easily constructed based on the existing control points to complete the control structure [33]. Further investigation has shown that there are three local cases of Loop patches

that reduce to uniform box splines if appropriate phantom vertices are introduced (Figure 7.4). Figure 7.4(a) shows a uniform quartic box spline control structure.

The 12 uniform quartic box spline basis functions [27] can be expressed as $B_4(u,v) = p_{uv}(u,v)\, B_{uv}$, where $p_{uv}(u,v)$ is the bi-quartic power basis vector:

$$p_u(u) = \begin{bmatrix} 1 & u & u^2 & u^3 & u^4 \end{bmatrix} \tag{7.4}$$

$$p_{uv}(u,v) = \begin{bmatrix} p_u(u) & vp_u(u) & v^2p_u(u) & v^3p_u(u) & v^4p_u(u) \end{bmatrix} \tag{7.5}$$

B_{uv} is the coefficient matrix on the power basis that constructs the uniform quartic box splines:

$$B_{uv} = \frac{1}{12}\begin{bmatrix}
1 & 1 & 0 & 1 & 6 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
-2 & 2 & 0 & -4 & 0 & 4 & 0 & -2 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & -12 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -4 & 2 & -4 & 8 & -4 & 0 & 2 & -4 & 2 & 0 & 0 \\
-1 & 2 & -1 & 1 & -1 & -1 & 1 & -1 & 2 & -1 & 0 & 0 \\
-4 & -2 & 0 & -2 & 0 & 2 & 0 & 2 & 4 & 0 & 0 & 0 \\
6 & -6 & 0 & 6 & -12 & 6 & 0 & -6 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & -6 & 12 & -6 & 0 & 6 & -12 & 6 & 0 & 0 \\
-2 & 4 & -2 & 2 & -2 & -2 & 2 & -2 & 4 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & -12 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
-6 & 6 & 0 & 0 & 12 & -12 & 0 & 0 & -6 & 6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-4 & 2 & 0 & 2 & 8 & -4 & 0 & -4 & -4 & 2 & 2 & 0 \\
2 & -2 & 0 & -2 & -2 & 4 & 0 & 4 & -2 & -2 & -2 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & -1 & -1 & 2 & 0 & 2 & -1 & -1 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{7.6}$$

The position of a phantom vertex is function of the other control vertex position to generate the control structure for a specific curve or surface. In the case of the bottom sharp edge B-spline

curve in Figure 7.4(c), the left control vertex V_3 is not specified by the user. This curve $C(u)$ should interpolate V_3 when $u = 0$, and it should be a uniform B-spline curve so that the normal subdivision averaging rules apply. Solving this equation yields an expression for the position of the curve phantom vertex V_3.

$$C(u) = \sum_{i=0}^{3} b_i(u) V_{3+i} \tag{7.7}$$

$$V_4 = C(0) \tag{7.8}$$

$$V_4 = \frac{1}{6}(V_3 + 4V_4 + V_5) \tag{7.9}$$

$$V_3 = 2V_4 - V_5 \tag{7.10}$$

Figure 7.4(b) illustrates the case of a patch adjacent to a regular crease edge. The necessary location for phantom vertices $0 \rightarrow 2$ are computed by mirroring their respective neighbor vertex across the sharp crease edge. For the example of vertex V_1, we must set:

$$V_1 = V_4 + V_5 - V_8 \tag{7.11}$$

Equation 7.11 is derived by equating the $v = 0$ isoparameter curve of the phantom patch to the sharp boundary B-spline curve of the patch.

Figure 7.4(c) shows a patch adjacent to a corner vertex and two regular crease edges. Identical to the single crease case, locations for phantom vertices $\{1, 2, 7, 10\}$ are computed by mirroring their associated neighbor vertex across the sharp crease edges. The location for the corner phantom vertices V_0 and V_3 are computed by mirroring the associated neighbor vertex across the corner vertex V_4. The rule for V_0 is as follows:

$$V_0 = 2V_4 - V_8 \tag{7.12}$$

Equation 7.12 is derived by equating the $u = 0$ and $v = 0$ isoparameter curves of the phantom patch to the sharp boundary curves of the patch. But first the sharp boundary curves must have their fourth phantom curve control point calculated for them using Equation 7.10.

These are the basic elements that build all uniform box spline patches, but combinations of the regular crease and corner situations are also uniform box spline patches. A triangle with all three corner vertices actually reduces to a bilinear patch. We have found that treating the regular corner case specially can greatly reduce the amount of subdivision that is necessary. For example, if a model is generated that texture maps each triangle with its own $\{[0, 1], [0, 1]\}$ texture space, then each texture domain is a bilinear patch and no subdivision is necessary. In principle though, only the smooth and regular crease cases are necessary to perform exact evaluation.

7.5 Loop Smooth and Spike Vertices

The smooth and spike vertex patch cases can be treated with the same matrix analysis. The only difference between the two cases is the averaging rule for the extraordinary vertex V_0 of Figure 7.5.

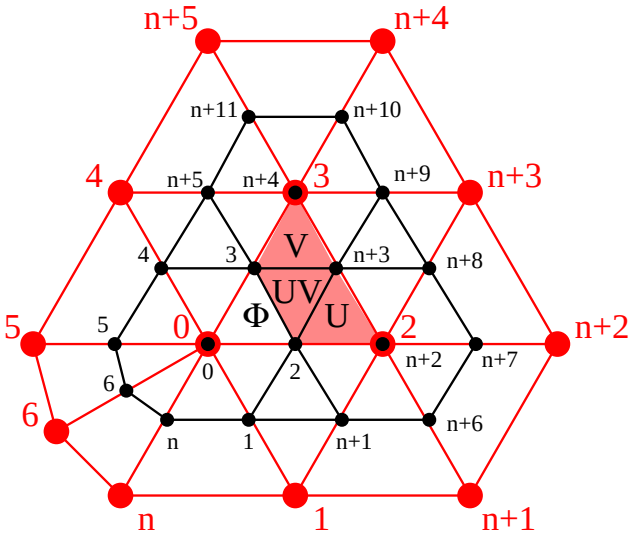


Figure 7.5: *Application of the $\bar{A}$ matrix to a triangle patch next to an extraordinary smooth or spike vertex produces a ribbon of three subpatches $\{U, UV, V\}$ that are regular box Spline patches and a fourth patch Φ which is analogous to its parent. Light vertices and edges represent the current generation, while black ones represent its child mesh.*

7.5.1 Loop Smooth and Spike $\bar{A}$

The extended 1-ring subdivision operator $\bar{A}$ generates all the necessary control vertices in the next generation that define the three regular box spline patches, see Figure 7.5. We choose to order the vertices in the 1-ring neighborhood as illustrated in Figure 7.5 to simplify our analysis.

$v = \frac{1}{12}, e = \frac{1}{8}$

$$\bar{A}_n = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \\ \hline\hline S_{21} & S_{22} \end{array}\right] = \left[\begin{array}{c|cccccc||ccccc} v_0 & v_1 & v_1 & v_1 & v_1 & \cdots & v_1 & 0 & 0 & 0 & 0 & 0 \\ \hline e_0 & e_1 & e_2 & 0 & 0 & \cdots & e_2 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_2 & e_1 & e_2 & 0 & & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & 0 & e_2 & e_1 & e_2 & & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & & & \ddots & & \vdots & & & \vdots & & \\ e_0 & 0 & 0 & & e_2 & e_1 & e_2 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_2 & 0 & \cdots & 0 & e_2 & e_1 & 0 & 0 & 0 & 0 & 0 \\ \hline\hline e & 3e & 3e & 0 & 0 & 0 & & e & 0 & 0 & 0 & 0 \\ v & v & 6v & v & 0 & 0 & & v & v & v & 0 & 0 \\ e & 0 & 3e & 3e & 0 & 0 & \cdots & 0 & 0 & e & 0 & 0 \\ v & 0 & v & 6v & v & 0 & & 0 & 0 & v & v & v \\ e & 0 & 0 & 3e & 3e & 0 & & 0 & 0 & 0 & 0 & e \\ \hline\hline 0 & e & 3e & 0 & 0 & 0 & & 3e & e & 0 & 0 & 0 \\ 0 & 0 & 3e & 0 & 0 & 0 & & e & 3e & e & 0 & 0 \\ 0 & 0 & 3e & e & 0 & 0 & \cdots & 0 & e & 3e & 0 & 0 \\ 0 & 0 & e & 3e & 0 & 0 & & 0 & 0 & 3e & e & 0 \\ 0 & 0 & 0 & 3e & 0 & 0 & & 0 & 0 & e & 3e & e \\ 0 & 0 & 0 & 3e & e & 0 & & 0 & 0 & 0 & e & 3e \end{array}\right] \tag{7.13}$$

When the valence $n = 3$, the shape of the S_0, S_{11}, and S_{21} matrices change slightly because V_1 is being used twice.

$$\bar{A}_3 = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \\ \hline\hline S_{21} & S_{22} \end{array}\right] = \left[\begin{array}{c|ccc||ccccc} v_0 & v_1 & v_1 & v_1 & 0 & 0 & 0 & 0 & 0 \\ \hline e_0 & e_1 & e_2 & e_2 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_2 & e_1 & e_2 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_2 & e_2 & e_1 & 0 & 0 & 0 & 0 & 0 \\ \hline\hline e & 3e & 3e & 0 & e & 0 & 0 & 0 & 0 \\ v & v & 6v & v & v & v & v & 0 & 0 \\ e & 0 & 3e & 3e & 0 & 0 & e & 0 & 0 \\ v & v & v & 6v & 0 & 0 & v & v & v \\ e & 3e & 0 & 3e & 0 & 0 & 0 & 0 & e \\ \hline\hline 0 & e & 3e & 0 & 3e & e & 0 & 0 & 0 \\ 0 & 0 & 3e & 0 & e & 3e & e & 0 & 0 \\ 0 & 0 & 3e & e & 0 & e & 3e & 0 & 0 \\ 0 & 0 & e & 3e & 0 & 0 & 3e & e & 0 \\ 0 & 0 & 0 & 3e & 0 & 0 & e & 3e & e \\ 0 & e & 0 & 3e & 0 & 0 & 0 & e & 3e \end{array}\right] \tag{7.14}$$

7.5.2 Loop Smooth and Spike A Jordan Decompositions

The subdivision operator A is the top square portion of $\bar{A}$.

$$A = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \end{array}\right] \tag{7.15}$$

For all valences $n > 3$, the subdivision operator A is not defective and can be diagonalized into its eigen-decomposition. For the single case of $n = 3$, A is defective, so the Jordan decomposition must used instead.

7.5.3 Loop Smooth and Spike S_{12}

The submatrix S_{12} is the same for all valences.

$$S_{12} = \begin{bmatrix} e & 0 & 0 & 0 & 0 \\ v & v & v & 0 & 0 \\ 0 & 0 & e & 0 & 0 \\ 0 & 0 & v & v & v \\ 0 & 0 & 0 & 0 & e \end{bmatrix} \tag{7.16}$$

S_{12} can be diagonalized into its eigen-decomposition. Due to the block lower triangular structure of A, the columns of V_{12} when zero extended on the top become eigenvectors of A.

$$\begin{aligned} S_{12} =& V_{12}\Lambda_{12}V_{12}^{-1} \\ =& \begin{bmatrix} 1 & 0 & \frac{e-v}{2} & 0 & -\frac{\sqrt{3}(e-v)}{2} \\ 0 & \frac{1}{2} & \frac{3v}{2} & -\frac{\sqrt{3}}{2} & -\frac{\sqrt{3}v}{2} \\ -1 & 0 & e-v & 0 & 0 \\ 0 & \frac{1}{2} & \frac{3v}{2} & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}v}{2} \\ 1 & 0 & \frac{e-v}{2} & 0 & \frac{\sqrt{3}(e-v)}{2} \end{bmatrix} \begin{bmatrix} e & 0 & 0 & 0 & 0 \\ 0 & v & 0 & 0 & 0 \\ 0 & 0 & e & 0 & 0 \\ 0 & 0 & 0 & v & 0 \\ 0 & 0 & 0 & 0 & e \end{bmatrix} \cdot \\ & \begin{bmatrix} \frac{1}{3} & 0 & -\frac{1}{3} & 0 & \frac{1}{3} \\ -\frac{v}{e-v} & 1 & -\frac{2v}{e-v} & 1 & -\frac{v}{e-v} \\ \frac{v}{3(e-v)} & 0 & \frac{2v}{3(e-v)} & 0 & \frac{v}{3(e-v)} \\ \frac{v}{\sqrt{3}(e-v)} & -\frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{3}} & -\frac{v}{\sqrt{3}(e-v)} \\ -\frac{v}{\sqrt{3}(e-v)} & 0 & 0 & 0 & \frac{v}{\sqrt{3}(e-v)} \end{bmatrix} \end{aligned} \tag{7.17}$$

7.5.4 Loop Smooth and Spike S_0 Eigen-decomposition

For the most general set of 1-ring averaging rules, the S_0 matrix has a symmetric circulant submatrix plus an additional row and column associated with control vertex V_0.

$$S_0 = \left[\begin{array}{c|cccccc} v_0 & v_1 & v_1 & v_1 & \cdots & v_1 & v_1 \\ \hline e_0 & e_1 & e_2 & e_3 & & e_3 & e_2 \\ e_0 & e_2 & e_1 & e_2 & \cdots & e_4 & e_3 \\ e_0 & e_3 & e_2 & e_1 & & e_5 & e_4 \\ \vdots & & \vdots & & \ddots & & \vdots \\ e_0 & e_3 & e_4 & e_5 & & e_1 & e_2 \\ e_0 & e_2 & e_3 & e_4 & \cdots & e_2 & e_1 \end{array}\right] \tag{7.18}$$

For a convergent subdivision scheme, the subdivision rules for the face, edge, and vertex rules must sum to one. If n is odd then $e_{\frac{n}{2}}$ does not exist.

$$v_0 = 1 - nv_1 \tag{7.19}$$

$$e_0 = 1 - \left(e_1 + 2 \sum_{k=2}^{\lfloor \frac{n-1}{2} \rfloor} e_k + e_{\frac{n}{2}} \right) \tag{7.20}$$

The S_0 matrix can be transformed into a block diagonal matrix $\hat{S}_0$ using a modified DFT.

$$\hat{S}_0 = F S_0 F^{-1} \tag{7.21}$$

$$= \frac{1}{n} \left[\begin{array}{c|cccc} n & 0 & 0 & & 0 \\ \hline 0 & 1 & 1 & & 1 \\ 0 & 1 & \omega^1 & \cdots & \omega^{n-1} \\ & & \vdots & \ddots & \vdots \\ 0 & 1 & \omega^{n-1} & \cdots & \omega^{(n-1)(n-1)} \end{array}\right] S_0 \left[\begin{array}{c|cccc} 1 & 0 & 0 & & 0 \\ \hline 0 & 1 & 1 & & 1 \\ 0 & 1 & \omega^{-1} & \cdots & \omega^{-(n-1)} \\ & & \vdots & \ddots & \vdots \\ 0 & 1 & \omega^{-(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{array}\right] \tag{7.22}$$

$\hat{S}_0$ has n blocks where B_0 is 2×2 and the other blocks are scalars and hence the eigenvalues λ_i for $i \in \{1, n-1\}$.

$$\hat{S}_0 = \left[\begin{array}{c|cc|c} B_0 & & & \\ \hline & \lambda_1 & & \\ & & \ddots & \\ \hline & & & \lambda_{n-1} \end{array}\right] \tag{7.23}$$

Block B_0 has the following form and eigenvalues $\lambda_{\pm 0}$:

$$B_0 = \begin{bmatrix} v_0 & nv_1 \\ e_0 & e_1 + 2\sum_{k=2}^{\lfloor \frac{n-1}{2} \rfloor} e_k + e_{\frac{n}{2}} \end{bmatrix} \tag{7.24}$$

$$\lambda_{\pm 0} = \left\{ 1, e_1 + 2 \sum_{k=2}^{\lfloor \frac{n-1}{2} \rfloor} e_k + e_{\frac{n}{2}} - nv_1 \right\} \tag{7.25}$$

The other eigenvalues λ_i for $i \in \{1, n-1\}$ from the scalar blocks are of the form:

$$\lambda_i = e_1 + 2 \sum_{k=2}^{\lfloor \frac{n-1}{2} \rfloor} \cos\left(\frac{2\pi i k}{n} \right) e_k + \cos(\pi i)\, e_{\frac{n}{2}} \tag{7.26}$$

These eigenvalues appear in pairs centered around $i = \frac{n}{2}$, so $\lambda_i = \lambda_{n-i}$ because they correspond to opposite multiples of the same angles and the cosines of each are the same.

The eigenvectors of S_0 can be found by first finding the eigenvectors of $\hat{S}_0$ and then applying the inverse DFT.

$$S_0 = F^{-1}\hat{S}_0 F \tag{7.27}$$

$$= F^{-1}\hat{V}_0\Lambda_0\hat{V}_0^{-1}F \tag{7.28}$$

$$= V_0\Lambda_0 V_0^{-1} \tag{7.29}$$

The eigenvectors of the $\hat{S}_0$ are:

$$\hat{V}_0 = \left[\begin{array}{cc|c} 1 & -\frac{1-v_0}{e_0} & \\ 1 & 1 & \\ \hline & & \boxed{1} \quad \ddots \\ & & \ddots \quad \boxed{1} \end{array}\right], \qquad \hat{V}_0^{-1} = \left[\begin{array}{cc|c} \frac{e_0}{1-v_0+e_0} & \frac{1-v_0}{1-v_0+e_0} & \\ -\frac{e_0}{1-v_0+e_0} & \frac{e_0}{1-v_0+e_0} & \\ \hline & & \boxed{1} \quad \ddots \\ & & \ddots \quad \boxed{1} \end{array}\right] \tag{7.30}$$

Transforming back by the inverse DFT, we find that the eigenvectors of S_0 are:

$$V_0 = F^{-1}\hat{V}_0$$

$$= \begin{bmatrix} 1 & -\frac{1-v_0}{e_0} & 0 & & 0 \\ 1 & 1 & 1 & & 1 \\ 1 & 1 & \omega^{-1} & \cdots & \omega^{-(n-1)} \\ & & \vdots & \ddots & \vdots \\ 1 & 1 & \omega^{-(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix} \tag{7.31}$$

$$V_0^{-1} = \hat{V}_0^{-1}F$$

$$= \frac{1}{n}\begin{bmatrix} \frac{ne_0}{1-v_0+e_0} & \frac{1-v_0}{1-v_0+e_0} & \frac{1-v_0}{1-v_0+e_0} & & \frac{1-v_0}{1-v_0+e_0} \\ -\frac{ne_0}{1-v_0+e_0} & \frac{e_0}{1-v_0+e_0} & \frac{e_0}{1-v_0+e_0} & & \frac{e_0}{1-v_0+e_0} \\ 0 & 1 & \omega^1 & \cdots & \omega^{n-1} \\ & & \vdots & \ddots & \vdots \\ 0 & 1 & \omega^{n-1} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \tag{7.32}$$

These eigenvectors have imaginary entries which is undesireable. Fortunately, the two eigenvectors v_i and v_{n-i} associated with the pairs of identical eigenvalues are complex congugates of each other, so it is possible to replace these eigenvectors with their sum and differences. If n is even, then the

eigenvector $v_{\frac{n}{2}}$ associated with $\lambda_{\frac{n}{2}}$ remains the same because it is real.

$$i \in \left\{1, \left\lfloor \tfrac{n}{2} \right\rfloor\right\}$$

$$u_i = \frac{1}{2}\left(v_i + v_{n-i}\right) = \begin{bmatrix} 0 \\ \cos\left(0\frac{2\pi i}{n}\right) \\ \cos\left(1\frac{2\pi i}{n}\right) \\ \cos\left(2\frac{2\pi i}{n}\right) \\ \vdots \\ \cos\left((n-1)\frac{2\pi i}{n}\right) \end{bmatrix}, u_{n-i} = \frac{\mathbf{i}}{2}\left(v_i - v_{n-i}\right) = \begin{bmatrix} 0 \\ \sin\left(0\frac{2\pi i}{n}\right) \\ \sin\left(1\frac{2\pi i}{n}\right) \\ \sin\left(2\frac{2\pi i}{n}\right) \\ \vdots \\ \sin\left((n-1)\frac{2\pi i}{n}\right) \end{bmatrix} \tag{7.33}$$

The corresponding row vectors from the inverse eigenvector matrix are then:

$$i \in \left\{1, \left\lfloor \tfrac{n}{2} \right\rfloor\right\}$$

$$u_i^{-1} = \left(v_i^{-1} + v_{n-i}^{-1}\right) = \frac{1}{n}\begin{bmatrix} 0 \\ 2\cos\left(0\frac{2\pi i}{n}\right) \\ 2\cos\left(1\frac{2\pi i}{n}\right) \\ 2\cos\left(2\frac{2\pi i}{n}\right) \\ \vdots \\ 2\cos\left((n-1)\frac{2\pi i}{n}\right) \end{bmatrix}, u_{n-i}^{-1} = -\mathbf{i}\left(v_i^{-1} - v_{n-i}^{-1}\right) = \frac{1}{n}\begin{bmatrix} 0 \\ 2\sin\left(0\frac{2\pi i}{n}\right) \\ 2\sin\left(1\frac{2\pi i}{n}\right) \\ 2\sin\left(2\frac{2\pi i}{n}\right) \\ \vdots \\ 2\sin\left((n-1)\frac{2\pi i}{n}\right) \end{bmatrix} \tag{7.34}$$

For the original Loop scheme, the S_0 matrix has many of the general face and edge averaging weights set to zero, so the matrix has the form:

$$S_0 = \left[\begin{array}{c|cccccc} v_0 & v_1 & v_1 & v_1 & v_1 & \cdots & v_1 \\ \hline e_0 & e_1 & e_2 & 0 & 0 & \cdots & e_2 \\ e_0 & e_2 & e_1 & e_2 & 0 & & 0 \\ e_0 & 0 & e_2 & e_1 & e_2 & & 0 \\ \vdots & \vdots & & & \ddots & & \vdots \\ e_0 & 0 & 0 & & e_2 & e_1 & e_2 \\ e_0 & e_2 & 0 & \cdots & 0 & e_2 & e_1 \end{array}\right] \tag{7.35}$$

For a convergent subdivision scheme, the subdivision rules for the face, edge, and vertex rules must sum to one.

$$v_0 = 1 - nv_1 \tag{7.36}$$

$$e_0 = 1 - (e_1 + 2e_2) \tag{7.37}$$

The values of the averaging rule weights adjacent to the extraordinary vertex can be modified to try to improve the surface behavior there, and it is convenient for us to leave these as variables so that we can combine the analysis for smooth and spike vertices. The standard edge rules are:

$$e_0 = 3e, e_1 = 3e, e_2 = e \tag{7.38}$$

The standard smooth vertex rules are:

$$v_0 = 1 - nv_1, v_1 = \frac{40 - \left(3 + 2\cos\left(\frac{2\pi}{n}\right)\right)^2}{64n} \tag{7.39}$$

The spike vertex rules are:

$$v_0 = 1, v_1 = 0 \tag{7.40}$$

The S_0 matrix can be transformed to a block diagonal matrix $\hat{S}_0$ using a modified DFT. The 2×2 block B_0 and its eigenvalues $\lambda_{\pm 0}$ are:

$$B_0 = \begin{bmatrix} v_0 & nv_1 \\ e_0 & e_1 + 2e_2 \end{bmatrix} \tag{7.41}$$

$$\lambda_{\pm 0} = \{1, e_1 + 2e_2 - nv_1\} \tag{7.42}$$

The other eigenvalues λ_i for $i \in \{1, n-1\}$ from the scalar blocks are of the form:

$$\lambda_i = e_1 + 2\cos\left(\frac{2\pi i}{n}\right) e_2 \tag{7.43}$$

The eigenvectors from the general case are already simplified.

7.5.5 Loop Smooth and Spike S_{11}

The eigenvector components V_{11} corresponding to S_{11} can be expressed with respect to the eigen-decompositions of S_0 and S_{12}. For valences $n > 3$, the columns of V_{11} can be computed using:

$$v_{11,c} = -V_{12}\left(\Lambda_{12} - \lambda_c I\right)^{-1} V_{12}^{-1} S_{11} v_{0,c} \tag{7.44}$$

In the smooth case when $n = 3$, the single eigenvalue $\lambda = \frac{1}{16}$ is also an eigenvalue of S_{12}, so the pseudoinverse must be used instead.

$$v_{11,c} = -V_{12}\left(\Lambda_{12} - \lambda_c I\right)^{\dagger} V_{12}^{-1} S_{11} v_{0,c} \tag{7.45}$$

7.5.6 Loop Smooth and Spike Picking Matrices

The picking matrices P_k where $k \in \{U, V, UV\}$ select out the 12 control points for the corresponding regular box spline patch from the $n + 11$ control points generated by the $\bar{A}$ matrix, see Figure 7.5. P_k is a $12 \times n+11$ matrix where each row has a single 1 in the column corresponding to control point listed in the sets below. Note that if the valence $n = 3$ then V_4 must be replaced by V_1.

$$P_{U,n} = \{V_1, V_{n+1}, V_{n+6}, V_0, V_2, V_{n+2}, V_{n+7}, V_3, V_{n+3}, V_{n+8}, V_{n+4}, V_{n+9}\} \tag{7.46}$$

$$P_{V,n} = \{V_0, V_2, V_{n+2}, V_4, V_3, V_{n+3}, V_{n+8}, V_{n+5}, V_{n+4}, V_{n+9}, V_{n+11}, V_{n+10}\} \tag{7.47}$$

$$P_{UV,n} = \{V_{n+9}, V_{n+4}, V_{n+5}, V_{n+8}, V_{n+3}, V_3, V_4, V_{n+2}, V_2, V_0, V_{n+1}, V_1\} \tag{7.48}$$

7.6 Loop Corner and Crease Vertices

The corner and crease vertex patch cases can be treated with the same matrix analysis. The only difference between the two cases is the averaging rule for the extraordinary vertex V_0 of Figure 7.3.

7.6.1 Loop Corner and Crease $\bar{A}$

The extended 1-ring subdivision operator $\bar{A}$ generates all the necessary wedge control vertices in the next generation that define the three regular box spline patches, see Figure 7.3. We choose to order the vertices in the 1-ring neighborhood as illustrated in Figure 7.3 to simplify our analysis. The first nonzero column of S_{11} and S_{21} is the one associated with vertex V_p.

$$\bar{v}_0 = \frac{6}{8}, \bar{v}_1 = \frac{1}{8}, \bar{e}_0 = \bar{e}_1 = \frac{1}{2}$$

$$\bar{A}_{n,p} = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \\ \hline\hline S_{21} & S_{22} \end{array}\right] = \left[\begin{array}{cc||c} T_0 & 0 & 0 \\ \hline T_{11} & T_{12} & \\ \hline\hline \multicolumn{2}{c||}{S_{11}} & S_{12} \\ \hline\hline \multicolumn{2}{c||}{S_{21}} & S_{22} \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccccccc||ccccc} \bar{v}_0 & \bar{v}_1 & \bar{v}_1 & 0 & 0 & 0 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{e}_0 & \bar{e}_1 & 0 & 0 & 0 & 0 & 0 & \cdots & & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{e}_0 & 0 & \bar{e}_1 & 0 & 0 & 0 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline e_0 & 0 & e_2 & e_1 & e_2 & 0 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & 0 & 0 & e_2 & e_1 & e_2 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & 0 & 0 & 0 & e_2 & e_1 & e_2 & & & 0 & 0 & 0 & 0 & 0 & 0 \\ & \vdots & & & & & \ddots & & & & & \vdots & & & \\ e_0 & 0 & 0 & & & & e_2 & e_1 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_2 & 0 & & & & 0 & e_2 & e_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline\hline e & 0 & 0 & & 3e & 3e & 0 & 0 & & e & 0 & 0 & 0 & 0 & 0 \\ v & 0 & 0 & & v & 6v & v & 0 & & v & v & v & 0 & 0 & 0 \\ e & 0 & 0 & \cdots & 0 & 3e & 3e & 0 & \cdots & 0 & 0 & e & 0 & 0 & 0 \\ v & 0 & 0 & & 0 & v & 6v & v & & 0 & 0 & v & v & v & 0 \\ e & 0 & 0 & & 0 & 0 & 3e & 3e & & 0 & 0 & 0 & 0 & e & 0 \\ \hline\hline 0 & 0 & 0 & & e & 3e & 0 & 0 & & 3e & e & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & & 0 & 3e & 0 & 0 & & e & 3e & e & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 3e & e & 0 & \cdots & 0 & e & 3e & 0 & 0 & 0 \\ 0 & 0 & 0 & & 0 & e & 3e & 0 & & 0 & 0 & 3e & e & 0 & 0 \\ 0 & 0 & 0 & & 0 & 0 & 3e & 0 & & 0 & 0 & e & 3e & e & 0 \\ 0 & 0 & 0 & & 0 & 0 & 3e & e & & 0 & 0 & 0 & e & 3e & 0 \end{array}\right] \tag{7.49}$$

When $p = 1$ or $p = n$, one side of the patch is adjacent to a crease edge, so the S_{22} submatrix is modified. Similar alterations are made to the S_{21}, S_{11}, and S_{12} submatrices. The eigen-decomposition of S_{12} will be discussed in greater detail in Section 7.6.3.

$$S_{22,n,1} = \begin{bmatrix} \bar{e} & 0 & 0 & 0 \\ e & 3e & 0 & 0 \\ 0 & 3e & e & 0 \\ 0 & e & 3e & e \\ 0 & 0 & e & 3e \end{bmatrix} \tag{7.50}$$

$$S_{22,n,n} = \begin{bmatrix} 3e & e & 0 & 0 \\ e & 3e & e & 0 \\ 0 & e & 3e & 0 \\ 0 & 0 & 3e & e \\ 0 & 0 & 0 & \bar{e} \end{bmatrix} \tag{7.51}$$

When the wedge valence $n = 1$ as in Figure 7.3(b), the shape of the S_0, S_{11}, and S_{21} matrices change slightly because the crease edge rules are present on both ends of the wedge in the second and third vertex rings.

$$\bar{A}_{1,1} = \left[\begin{array}{c|c} S_0 & 0 \\ \hline S_{11} & S_{12} \\ \hline S_{21} & S_{22} \end{array}\right] = \left[\begin{array}{ccc|ccc} \bar{v}_0 & \bar{v}_1 & \bar{v}_1 & 0 & 0 & 0 \\ \bar{e} & \bar{e} & 0 & 0 & 0 & 0 \\ \bar{e} & 0 & \bar{e} & 0 & 0 & 0 \\ \hline \bar{v}_1 & 0 & \bar{v}_0 & \bar{v}_1 & 0 & 0 \\ e & 3e & 3e & 0 & e & 0 \\ \bar{v}_1 & \bar{v}_0 & 0 & 0 & 0 & \bar{v}_1 \\ \hline 0 & 0 & \bar{e} & \bar{e} & 0 & 0 \\ 0 & e & 3e & e & 3e & 0 \\ 0 & 3e & e & 0 & 3e & e \\ 0 & \bar{e} & 0 & 0 & 0 & \bar{e} \end{array}\right] \tag{7.52}$$

7.6.2 Loop Corner and Crease A Jordan Decompositions

The Jordan decompositions of A for corner and crease patches are very similar. For certain valences, the submatrix S_0 is defective due to interactions of shared eigenvalues between its submatrices T_0 and T_{12}.

7.6.3 Loop Corner and Crease S_{12}

The submatrix S_{12}, n, p is the same as the smooth case from Equation 7.16 for most wedge valences n and subpatches p. However when $p = 1$ or $p = n$, the subpatch p of the wedge

is adjacent to a crease edge and a boundary rule is introduced into S_{12}, Figure 7.3(b). S_{12} can be diagonalized into its eigen-decomposition. Due to the block lower triangular structure of A, the columns of V_{12} when zero extended on the top become eigenvectors of A. The three special cases near crease curves are as follows:

$$S_{12,n,1} = \begin{bmatrix} \bar{v}_1 & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & v & v & v \\ 0 & 0 & 0 & e \end{bmatrix} \tag{7.53}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & e-v \\ 0 & 0 & 1 & 2v \\ 0 & -1 & 0 & e-v \end{bmatrix} \begin{bmatrix} \bar{v}_1 & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & e \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & -\frac{1}{2} \\ 0 & -\frac{v}{e-v} & 1 & -\frac{v}{e-v} \\ 0 & \frac{1}{2(e-v)} & 0 & \frac{1}{2(e-v)} \end{bmatrix} \tag{7.54}$$

$$S_{12,n,n} = \begin{bmatrix} e & 0 & 0 & 0 \\ v & v & v & 0 \\ 0 & 0 & e & 0 \\ 0 & 0 & 0 & \bar{v}_1 \end{bmatrix} \tag{7.55}$$

$$= \begin{bmatrix} 1 & 0 & e-v & 0 \\ 0 & 1 & 2v & 0 \\ -1 & 0 & e-v & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & e & 0 \\ 0 & 0 & 0 & \bar{v}_1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & 0 & -\frac{1}{2} & 0 \\ -\frac{v}{e-v} & 1 & -\frac{v}{e-v} & 0 \\ \frac{1}{2(e-v)} & 0 & \frac{1}{2(e-v)} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.56}$$

$$S_{12,1,1} = \begin{bmatrix} \bar{v}_1 & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & \bar{v}_1 \end{bmatrix} \tag{7.57}$$

7.6.4 Loop Corner and Crease S_0 Jordan decomposition

The S_0 matrix for corner and crease vertices is only dependent on the valence n and not on the subpatch p. S_0 is a lower block triangular matrix. The subblock T differs for the corner and crease cases because of the averaging rules for V_0. T_{11} and T_{12} are the same for both cases.

$$S_{0,n} = \left[\begin{array}{c|c} T & 0 \\ \hline T_{11,n} & T_{12,n} \end{array}\right] = \left[\begin{array}{ccc|ccccc} \multicolumn{3}{c|}{T} & \multicolumn{5}{c}{0} \\ \hline 3e & 0 & e & 3e & e & 0 & 0 & \multirow{3}{*}{0} \\ 3e & 0 & 0 & e & 3e & e & 0 & \\ 3e & 0 & 0 & 0 & e & 3e & e & \\ \vdots & & & & & & \ddots & & \\ 3e & 0 & 0 & \multirow{2}{*}{0} & & & e & 3e & e \\ 3e & e & 0 & & & & 0 & e & 3e \end{array}\right] \tag{7.58}$$

For certain valences, $S_{0,n}$ is defective due to a shared eigenvalue between the T and T_{12} subblocks, and a nontrivial Jordan form must be used. T_{12} is a tridiagonal Toeplitz matrix, so its eigenvectors are based on the DCT. When the valence n is even, the eigenvectors are:

$$\alpha\left(r,c\right) = \cos r\frac{c\pi}{n} \tag{7.59}$$

$$\beta\left(r,c\right) = \sin r\frac{c\pi}{n} \tag{7.60}$$

$$V_{12,n} = \left[\begin{array}{cccc} \alpha\left(-\frac{n-2}{2},1\right) & \beta\left(-\frac{n-2}{2},2\right) & \cdots & \alpha\left(-\frac{n-2}{2},n-1\right) \\ \vdots & \vdots & & \vdots \\ \alpha\left(0,1\right) & \beta\left(0,2\right) & \cdots & \alpha\left(0,n-1\right) \\ \vdots & \vdots & & \vdots \\ \alpha\left(\frac{n-2}{2},1\right) & \beta\left(\frac{n-2}{2},2\right) & \cdots & \alpha\left(\frac{n-2}{2},n-1\right) \end{array}\right] \tag{7.61}$$

$$V_{12,n}^{-1} = \frac{2}{n}V_{12,n}^{t} \tag{7.62}$$

When the valence n is odd, the eigenvectors are:

$$\alpha\left(r,c\right) = \cos r\frac{c\pi}{n} \tag{7.63}$$

$$\beta\left(r,c\right) = \sin r\frac{c\pi}{n} \tag{7.64}$$

$$V_{12,n} = \left[\begin{array}{cccc} \alpha\left(-\frac{n-2}{2},1\right) & \beta\left(-\frac{n-2}{2},2\right) & \cdots & \beta\left(-\frac{n-2}{2},n-1\right) \\ \vdots & \vdots & & \vdots \\ \alpha\left(-\frac{1}{2},1\right) & \beta\left(-\frac{1}{2},2\right) & \cdots & \beta\left(-\frac{1}{2},n-1\right) \\ \alpha\left(\frac{1}{2},1\right) & \beta\left(\frac{1}{2},2\right) & \cdots & \beta\left(\frac{1}{2},n-1\right) \\ \vdots & \vdots & & \vdots \\ \alpha\left(\frac{n-2}{2},1\right) & \beta\left(\frac{n-2}{2},2\right) & \cdots & \beta\left(\frac{n-2}{2},n-1\right) \end{array}\right] \tag{7.65}$$

$$V_{12,n}^{-1} = \frac{2}{n}V_{12,n}^{t} \tag{7.66}$$

The eigenvalues of $T_{12,n}$ are:

$$\left\{ \begin{array}{l} i \in [1, n-1] : \\ \lambda_i = e\left(3 + 2\cos\frac{i\pi}{n}\right) \end{array} \right\} \tag{7.67}$$

All of the eigenvalues of $T_{12,n}$ are unique.

7.6.5 Loop Corners

For the corner patch case, the submatrix T of $S_{0,n}$ (Equation 7.58) contains the averaging rules for the vertices $0 \rightarrow 2$ from Figure 7.3. Note that in the case of a corner vertex with one sharp edge, vertices 1 and 2 are two copies of the same point:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \tag{7.68}$$

T has the eigenvalues $\lambda = \left\{1, \frac{1}{2}, \frac{1}{2}\right\}$. $T_{12,n}$ also has an eigenvalue of $\lambda_{\frac{n}{3}} = \frac{1}{2}$ when the wedge valence n is a multiple of 3, so the associated eigenvectors of $S_{0,n}$ fall into two cases.

Loop Corner with $Mod\,(n, 3) \neq 0$

When $Mod\,(n, 3) \neq 0$, $S_{0,n}$ is diagonalizable, so there is a full set of eigenvectors.

$$\Lambda = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & \frac{1}{2} & 0 & \\ 0 & 0 & \frac{1}{2} & \\ \hline & & & \ddots \end{array}\right] \tag{7.69}$$

Equation 7.72 shows the first three eigenvectors $\{v_1, v_2, v_3\}$ for when n is odd.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.70}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.71}$$

$$\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 1 \end{array}\right], \left[\begin{array}{c} 0 \\ \alpha\left(\frac{n}{2},\frac{n}{3}\right) \\ \alpha\left(-\frac{n}{2},\frac{n}{3}\right) \\ \hline \alpha\left(-\frac{n-2}{2},\frac{n}{3}\right) \\ \vdots \\ \alpha\left(-\frac{1}{2},\frac{n}{3}\right) \\ \alpha\left(\frac{1}{2},\frac{n}{3}\right) \\ \vdots \\ \alpha\left(\frac{n-2}{2},\frac{n}{3}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ \beta\left(\frac{n}{2},\frac{n}{3}\right) \\ \beta\left(-\frac{n}{2},\frac{n}{3}\right) \\ \hline \beta\left(-\frac{n-1}{2},\frac{n}{3}\right) \\ \vdots \\ \beta\left(-\frac{1}{2},\frac{n}{3}\right) \\ \beta\left(\frac{1}{2},\frac{n}{3}\right) \\ \vdots \\ \beta\left(\frac{n-3}{2},\frac{n}{3}\right) \end{array}\right] \tag{7.72}$$

Equation 7.75 shows the first three eigenvectors $\{v_1, v_2, v_3\}$ for when n is even.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.73}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.74}$$

$$\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{array}\right], \left[\begin{array}{c} 0 \\ \alpha\left(\frac{n}{2},\frac{n}{3}\right) \\ \alpha\left(-\frac{n}{2},\frac{n}{3}\right) \\ \hline \alpha\left(-\frac{n-2}{2},\frac{n}{3}\right) \\ \vdots \\ \alpha\left(0,\frac{n}{3}\right) \\ \vdots \\ \alpha\left(\frac{n-2}{2},\frac{n}{3}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ \beta\left(\frac{n}{2},\frac{n}{3}\right) \\ \beta\left(-\frac{n}{2},\frac{n}{3}\right) \\ \hline \beta\left(-\frac{n-2}{2},\frac{n}{3}\right) \\ \vdots \\ \beta\left(0,\frac{n}{3}\right) \\ \vdots \\ \beta\left(\frac{n-2}{2},\frac{n}{3}\right) \end{array}\right] \tag{7.75}$$

Loop Corner with $Mod(n,3) = 0$

When n is divisible by 3, $S_{0,n}$ is defective due to the eigenvalue $\lambda = \frac{1}{2}$ which has algebraic multiplicity three but only two linearly independent eigenvectors.

$$J = \left[\begin{array}{ccc|cc} 1 & 0 & 0 & 0 & \\ 0 & \frac{1}{2} & 0 & 0 & \\ 0 & 0 & \frac{1}{2} & 0 & \\ \hline 0 & 0 & 1 & \frac{1}{2} & \\ & & & & \ddots \end{array}\right] \tag{7.76}$$

Equation 7.80 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$ when n is odd. Note that the scale of all vectors in a Jordan block are dependent on each other, so v_3 and v_4 must be scaled together.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.77}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.78}$$

$$\gamma(r) = \frac{1}{2e\sin\left(\frac{\pi}{3}\right)}\left(r\sin\left(r\frac{\pi}{3}\right) + \sum_{k=3\left\lfloor\frac{2r}{3}\right\rfloor}^{2r}\sin\left((2k-2r)\frac{\pi}{3}\right)\right) \tag{7.79}$$

$$\left[\begin{array}{c}1\\1\\1\\\hline 1\\\vdots\\1\\1\\\vdots\\1\end{array}\right], \left[\begin{array}{c}0\\\beta\left(\frac{n}{2},\frac{n}{3}\right)\\\beta\left(-\frac{n}{2},\frac{n}{3}\right)\\\hline \beta\left(-\frac{n-2}{2},\frac{n}{3}\right)\\\vdots\\\beta\left(-\frac{1}{2},\frac{n}{3}\right)\\\beta\left(\frac{1}{2},\frac{n}{3}\right)\\\vdots\\\beta\left(\frac{n-2}{2},\frac{n}{3}\right)\end{array}\right], \left[\begin{array}{c}0\\\gamma\left(\frac{n}{2}\right)\\\gamma\left(-\frac{n}{2}\right)\\\hline \gamma\left(-\frac{n-2}{2}\right)\\\vdots\\\gamma\left(-\frac{1}{2}\right)\\\gamma\left(\frac{1}{2}\right)\\\vdots\\\gamma\left(\frac{n-2}{2}\right)\end{array}\right], \left[\begin{array}{c}0\\0\\0\\\hline \alpha\left(-\frac{n-2}{2},\frac{n}{3}\right)\\\vdots\\\alpha\left(-\frac{1}{2},\frac{n}{3}\right)\\\alpha\left(\frac{1}{2},\frac{n}{3}\right)\\\vdots\\\alpha\left(\frac{n-2}{2},\frac{n}{3}\right)\end{array}\right] \tag{7.80}$$

Equation 7.84 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$ when n is even. Note that the scale of all vectors in a Jordan block are dependent on each other, so v_3 and v_4 must be

scaled together.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.81}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.82}$$

$$\gamma(r) = \frac{1}{2e\sin\left(\frac{\pi}{3}\right)}\left(-(r+1)\cos\left(r\frac{\pi}{3}\right) + \sum_{k=3\left\lfloor\frac{r}{3}\right\rfloor}^{r}\cos\left((2k-r)\frac{\pi}{3}\right)\right) \tag{7.83}$$

$$\left[\begin{array}{c}1\\1\\1\\\hline 1\\\vdots\\1\\\vdots\\1\end{array}\right], \left[\begin{array}{c}0\\\alpha\left(\frac{n}{2},\frac{n}{3}\right)\\\alpha\left(-\frac{n}{2},\frac{n}{3}\right)\\\hline \alpha\left(-\frac{n-2}{2},\frac{n}{3}\right)\\\vdots\\\alpha\left(0,\frac{n}{3}\right)\\\vdots\\\alpha\left(\frac{n-2}{2},\frac{n}{3}\right)\end{array}\right], \left[\begin{array}{c}0\\\gamma\left(\frac{n}{2}\right)\\\gamma\left(-\frac{n}{2}\right)\\\hline \gamma\left(-\frac{n-2}{2}\right)\\\vdots\\\gamma(0)\\\vdots\\\gamma\left(\frac{n-2}{2}\right)\end{array}\right], \left[\begin{array}{c}0\\0\\0\\\hline \beta\left(-\frac{n-2}{2},\frac{n}{3}\right)\\\vdots\\\beta\left(0,\frac{n}{3}\right)\\\vdots\\\beta\left(\frac{n-2}{2},\frac{n}{3}\right)\end{array}\right] \tag{7.84}$$

7.6.6 Loop Creases

For the crease patch case, the submatrix T of $S_{0,n}$ (Equation 7.58) contains the averaging rules for the vertices $0 \to 2$ from Figure 7.3. Note that in the case of a crease vertex with one sharp edge, vertices 1 and 2 are the same point:

$$T = \begin{bmatrix}\frac{6}{8} & \frac{1}{8} & \frac{1}{8}\\ \frac{1}{2} & \frac{1}{2} & 0\\ \frac{1}{2} & 0 & \frac{1}{2}\end{bmatrix} \tag{7.85}$$

T has the eigenvalues $\lambda = \left\{1, \frac{1}{2}, \frac{1}{4}\right\}$. $T_{12,n}$ also has an eigenvalue of $\frac{1}{2}$ when the wedge valence n is a multiple of 3, so the associated eigenvectors of $S_{0,n}$ fall into two cases.

Loop Crease with n Odd or $Mod(n,3) \neq 0$

When n is odd or $Mod(n,3) \neq 0$, $S_{0,n}$ is diagonalizable, so there is a full set of eigenvectors.

$$\Lambda = \left[\begin{array}{ccc|c}1 & 0 & 0 & \\ 0 & \frac{1}{2} & 0 & \\ 0 & 0 & \frac{1}{4} & \\\hline & & & \ddots\end{array}\right] \tag{7.86}$$

Equation 7.89 shows the first three eigenvectors $\{v_1, v_2, v_3\}$ for when n is odd.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.87}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.88}$$

$$\left[\begin{array}{c}1\\1\\1\\\hline 1\\\vdots\\1\\1\\\vdots\\1\end{array}\right], \left[\begin{array}{c}0\\\beta\left(\frac{n}{2},\frac{n}{3}\right)\\\beta\left(-\frac{n}{2},\frac{n}{3}\right)\\\hline \beta\left(-\frac{n-2}{2},\frac{n}{3}\right)\\\vdots\\\beta\left(-\frac{1}{2},\frac{n}{3}\right)\\\beta\left(\frac{1}{2},\frac{n}{3}\right)\\\vdots\\\beta\left(\frac{n-2}{2},\frac{n}{3}\right)\end{array}\right], \left[\begin{array}{c}-\cos\left(\frac{n\pi}{3}\right)\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(\frac{n}{2},\frac{2n}{3}\right)\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(-\frac{n}{2},\frac{2n}{3}\right)\\\hline \cos\left(\frac{n\pi}{3}\right)+\alpha\left(-\frac{n-2}{2},\frac{2n}{3}\right)\\\vdots\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(-\frac{1}{2},\frac{2n}{3}\right)\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(\frac{1}{2},\frac{2n}{3}\right)\\\vdots\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(\frac{n-2}{2},\frac{2n}{3}\right)\end{array}\right] \tag{7.89}$$

Equation 7.92 shows the first three eigenvectors $\{v_1, v_2, v_3\}$ for when n is even.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.90}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.91}$$

$$\left[\begin{array}{c}1\\1\\1\\\hline 1\\\vdots\\1\\\vdots\\1\end{array}\right], \left[\begin{array}{c}0\\\beta\left(\frac{n}{2},\frac{n}{3}\right)\\\beta\left(-\frac{n}{2},\frac{n}{3}\right)\\\hline \beta\left(-\frac{n-2}{2},\frac{n}{3}\right)\\\vdots\\\beta\left(0,\frac{n}{3}\right)\\\vdots\\\beta\left(\frac{n-2}{2},\frac{n}{3}\right)\end{array}\right], \left[\begin{array}{c}-\cos\left(\frac{n\pi}{3}\right)\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(\frac{n}{2},\frac{2n}{3}\right)\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(-\frac{n}{2},\frac{2n}{3}\right)\\\hline \cos\left(\frac{n\pi}{3}\right)+\alpha\left(-\frac{n-2}{2},\frac{2n}{3}\right)\\\vdots\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(0,\frac{2n}{3}\right)\\\vdots\\\cos\left(\frac{n\pi}{3}\right)+\alpha\left(\frac{n-2}{2},\frac{2n}{3}\right)\end{array}\right] \tag{7.92}$$

Loop Crease with n Even and $Mod(n,3) = 0$

When n is even and divisible by 3, $S_{0,n}$ is defective due to the eigenvalue $\lambda_{\frac{n}{3}} = \frac{1}{2}$ which has algebraic multiplicity two but only one linearly independent eigenvector.

$$J = \left[\begin{array}{ccc|cc}1 & 0 & 0 & 0 & \\0 & \frac{1}{4} & 0 & 0 & \\0 & 0 & \frac{1}{2} & 0 & \\\hline 0 & 0 & 1 & \frac{1}{2} & \\ & & & & \ddots\end{array}\right] \tag{7.93}$$

Equation 7.97 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$ when n is even. Note that the scale of all vectors in a Jordan block are dependent on each other, so v_3 and v_4 must be scaled together.

$$\alpha(r,c) = \cos r\frac{c\pi}{n} \tag{7.94}$$

$$\beta(r,c) = \sin r\frac{c\pi}{n} \tag{7.95}$$

$$\gamma(r) = \frac{1}{2e\sin\left(\frac{\pi}{3}\right)}\left(-(r+1)\cos\left(r\frac{\pi}{3}\right) + \sum_{k=3\left\lfloor\frac{r}{3}\right\rfloor}^{r}\cos\left((2k-r)\frac{\pi}{3}\right)\right) \tag{7.96}$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ \hline 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} -\cos\left(\frac{n\pi}{3}\right) \\ \cos\left(\frac{n\pi}{3}\right) + \alpha\left(\frac{n}{2}, \frac{2n}{3}\right) \\ \cos\left(\frac{n\pi}{3}\right) + \alpha\left(-\frac{n}{2}, \frac{2n}{3}\right) \\ \hline \cos\left(\frac{n\pi}{3}\right) + \alpha\left(-\frac{n-2}{2}, \frac{2n}{3}\right) \\ \vdots \\ \cos\left(\frac{n\pi}{3}\right) + \alpha\left(0, \frac{2n}{3}\right) \\ \vdots \\ \cos\left(\frac{n\pi}{3}\right) + \alpha\left(\frac{n-2}{2}, \frac{2n}{3}\right) \end{bmatrix}, \begin{bmatrix} 0 \\ \gamma\left(\frac{n}{2}\right) \\ \gamma\left(-\frac{n}{2}\right) \\ \hline \gamma\left(-\frac{n-2}{2}\right) \\ \vdots \\ \gamma(0) \\ \vdots \\ \gamma\left(\frac{n-2}{2}\right) \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline \beta\left(-\frac{n-2}{2}, \frac{n}{3}\right) \\ \vdots \\ \beta\left(0, \frac{n}{3}\right) \\ \vdots \\ \beta\left(\frac{n-2}{2}, \frac{n}{3}\right) \end{bmatrix} \tag{7.97}$$

7.6.7 Loop Corner and Crease S_{11}

The eigenvector components V_{11} corresponding to S_{11} can be expressed with respect to the eigen-decompositions of S_0 and S_{12}. For the diagonalizable cases, the columns of V_{11} can be computed using:

$$v_{11,c} = -V_{12}\left(\Lambda_{12} - \lambda_c I\right)^{-1} V_{12}^{-1} S_{11} v_{0,c} \tag{7.98}$$

For the defective cases, the single eigenvalue $\lambda = \frac{1}{2}$ is also an eigenvalue of S_{12}, so the pseudoinverse must be used instead.

$$v_{11,c} = -V_{12}\left(\Lambda_{12} - \lambda_c I\right)^{-1} V_{12}^{\dagger} S_{11} v_{0,c} \tag{7.99}$$

7.6.8 Loop Corner and Crease Picking Matrices

The picking matrices P_k where $k \in \{U, V, UV\}$ select out the 12 control points for the corresponding regular box spline patch from the $n+11$ control points generated by the $\bar{A}$ matrix, see Figure 7.3. P_k is a $12 \times n+11$ matrix where each row has a single 1 in the column corresponding to control point listed in the sets below.

The following are the picking matrices for subpatches $1 < p < n$ (note that if $p = n - 1$ then V_{p+3} must be replaced by V_1):

$$P_{U,n,p} = \{V_p, V_{n+2}, V_{n+7}, V_0, V_{p+1}, V_{n+3}, V_{n+8}, V_{p+2}, V_{n+4}, V_{n+9}, V_{n+5}, V_{n+10}\} \tag{7.100}$$

$$P_{V,n,p} = \{V_0, V_{p+1}, V_{n+3}, V_{p+3}, V_{p+2}, V_{n+4}, V_{n+9}, V_{n+6}, V_{n+5}, V_{n+10}, V_{n+12}, V_{n+11}\} \tag{7.101}$$

$$P_{UV,n,p} = \{V_{n+10}, V_{n+5}, V_{n+6}, V_{n+9}, V_{n+4}, V_{p+2}, V_{p+3}, V_{n+3}, V_{p+1}, V_0, V_{n+2}, V_p\} \tag{7.102}$$

The following are the picking matrices for subpatch $p = 1$ (note that if $p = n - 1$ then V_{p+3} must be replaced with V_1):

$$\bar{V}_a = V_0 + V_{p+1} - V_{p+2} \tag{7.103}$$

$$\bar{V}_b = V_{p+1} + V_{n+2} - V_{n+3} \tag{7.104}$$

$$\bar{V}_c = V_{n+2} + V_{n+6} - V_{n+7} \tag{7.105}$$

$$P_{U,n,1} = \{\bar{V}_a, \bar{V}_b, \bar{V}_c, V_0, V_{p+1}, V_{n+2}, V_{n+6}, V_{p+2}, V_{n+3}, V_{n+7}, V_{n+4}, V_{n+9}\} \tag{7.106}$$

$$P_{V,n,1} = \{V_0, V_{p+1}, V_{n+2}, V_{p+3}, V_{p+2}, V_{n+3}, V_{n+7}, V_{n+5}, V_{n+4}, V_{n+8}, V_{n+10}, V_{n+9}\} \tag{7.107}$$

$$P_{UV,n,1} = \{V_{n+8}, V_{n+4}, V_{n+5}, V_{n+7}, V_{n+3}, V_{p+2}, V_{p+3}, V_{n+2}, V_{p+1}, V_0, \bar{V}_b, \bar{V}_a\} \tag{7.108}$$

The following are the picking matrices for subpatch $p = n$:

$$\bar{V}_d = V_0 + V_1 - V_{p+1} \tag{7.109}$$

$$\bar{V}_e = V_1 + V_{n+5} - V_{n+4} \tag{7.110}$$

$$\bar{V}_f = V_{n+5} + V_{n+10} - V_{n+9} \tag{7.111}$$

$$P_{U,n,n} = \{V_p, V_{n+2}, V_{n+6}, V_0, V_{p+1}, V_{n+3}, V_{n+7}, V_1, V_{n+4}, V_{n+8}, V_{n+5}, V_{n+9}\} \tag{7.112}$$

$$P_{V,n,n} = \{V_0, V_{p+1}, V_{n+3}, \bar{V}_d, V_1, V_{n+4}, V_{n+8}, \bar{V}_e, V_{n+5}, V_{n+9}, \bar{V}_f, V_{n+10}\} \tag{7.113}$$

$$P_{UV,n,n} = \{V_{n+9}, V_{n+5}, \bar{V}_e, V_{n+8}, V_{n+4}, V_1, \bar{V}_d, V_{n+3}, V_{p+1}, V_0, V_{n+2}, V_p\} \tag{7.114}$$

For a wedge with a single patch $n = 1$, the picking matrices are:

$$\bar{V}_a = V_0 + V_2 - V_1 \tag{7.115}$$

$$\bar{V}_b = V_2 + V_{n+2} - V_{n+3} \tag{7.116}$$

$$\bar{V}_c = V_{n+2} + V_{n+5} - V_{n+6} \tag{7.117}$$

$$\bar{V}_d = V_0 + V_1 - V_2 \tag{7.118}$$

$$\bar{V}_e = V_1 + V_{n+4} - V_{n+3} \tag{7.119}$$

$$\bar{V}_f = V_{n+4} + V_{n+8} - V_{n+7} \tag{7.120}$$

$$P_{U,1,1} = \{\bar{V}_a, \bar{V}_b, \bar{V}_c, V_0, V_2, V_{n+2}, V_{n+5}, V_1, V_{n+3}, V_{n+6}, V_{n+4}, V_{n+7}\} \tag{7.121}$$

$$P_{V,1,1} = \{V_0, V_2, V_{n+2}, \bar{V}_d, V_1, V_{n+3}, V_{n+6}, \bar{V}_e, V_{n+4}, V_{n+7}, \bar{V}_f, V_{n+8}\} \tag{7.122}$$

$$P_{UV,1,1} = \{V_{n+7}, V_{n+4}, \bar{V}_e, V_{n+6}, V_{n+3}, V_1, \bar{V}_d, V_{n+2}, V_2, V_0, \bar{V}_b, \bar{V}_a\} \tag{7.123}$$

7.7 Conclusion

We have implemented a new exact evaluation algorithm for piecewise smooth Loop subdivision surfaces. We have developed new techniques for dart, crease, corner, and spike patches, and we have provided the set of matrices necessary for each of these cases.

Chapter 8

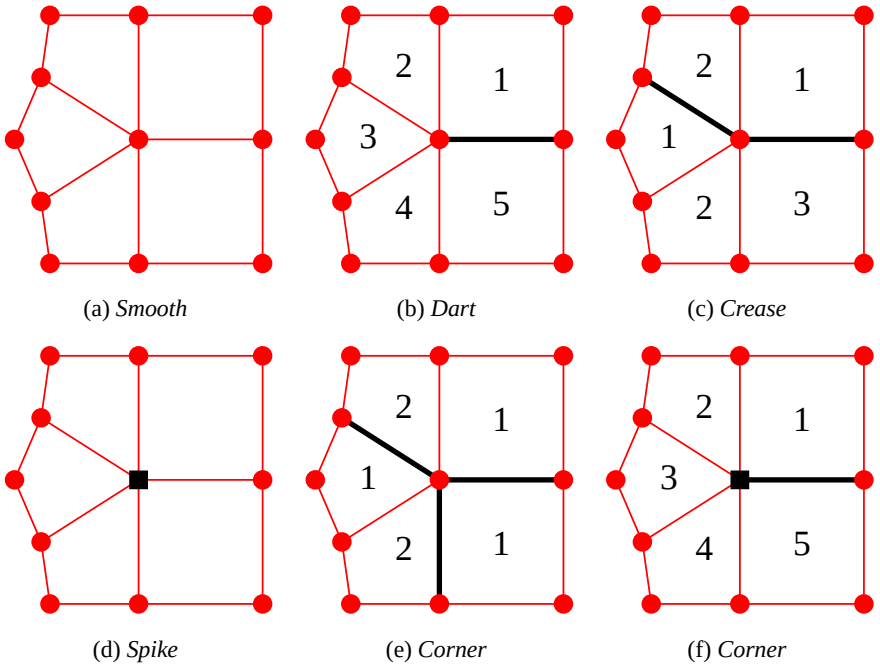
Catmull-Clark Evaluation

8.1 Introduction

Constant-time evaluation of Catmull-Clark surfaces [5] near smooth extraordinary vertices [29] greatly increased the ease of use of this free-form surface primitive in graphics and CAD. The per quadrilateral (u, v) parameterization has made subdivision surfaces compatible with standard CAD primitives such as NURBS. Sharp and semi-sharp edge and vertex averaging rules [10, 6, 4] make Catmull-Clark surfaces an even more powerful and descriptive modeling primitive for CAD, allowing for intentional breaks in continuity to define sharp features. Current methods for evaluating surfaces near these features rely on performing repeated subdivisions until the control structure approximates the limit surface within some tolerance. It is then possible to push the control points to their limit positions and bilinearly interpolate values across an inexact surface patch. In some applications, the exact solution near these sharp features is critical, e.g. mechanical modeling where these features may correspond to areas of particularly tight tolerances. We present a new approach for the sharp edge and vertex rules [6] supported in the RenderMan interface [21] that enables the constant-time exact evaluation of patches adjacent to corner, crease, dart, and spike vertices (Figure 8.1) by employing the Jordan decomposition [7] of the local subdivision operator.

8.2 Basic Approach

The smooth Catmull-Clark evaluation [29] has shown that Catmull-Clark subdivision meets the two necessary properties to enable contant-time exact evaluation. First, the subdivision operator at each iteration introduces new regions that can be described by known parameteric func-



(a) *Smooth* (b) *Dart* (c) *Crease*

(d) *Spike* (e) *Corner* (f) *Corner*

Figure 8.1: *Examples of the five different extraordinary vertex types for piecewise smooth Catmull-Clark surfaces. Black thick lines and squares denote sharp edges and vertices. Light thin lines and circles represent smooth edges and vertices. The numbers indicate the quad patch number p within each wedge; the maximum number is the valence n of the wedge.*

tions of uniform B-splines, and in the limit, the union of these regions converge to cover the entire surface area. Second, it is possible to perform an arbitrary number of applications of the subdivision operator in a constant amount of time using its eigen-decomposition. We now show how to extend this work to handle the remaining cases of patches near infinitely sharp features.

The averaging rules of the Catmull-Clark subdivision scheme generalize uniform bi-cubic B-spline surfaces to meshes with vertices of arbitrary valence. In the regular regions of the mesh where the vertices are all valence $n = 4$, the surface is a uniform bi-cubic B-spline. Smooth extraordinary vertices have valence $n \neq 4$ (Figure 8.1(a)), and they are averaged by a valence-specific smoothing rule. The subdivision operator near one of these vertices creates ever shrinking rings of regular B-spline patches that converge to the limit position of that vertex [22]. The local averaging rules can be expressed by the extended subdivision matrix $\bar{A}$, and each application of it generates the control points for three regular B-spline patches $\{U, UV, V\}$ and a fourth patch Φ that is a contracted topological copy of its parent and which then may get partitioned again in the same manner (Figure 8.3(a)). Any parametric location $(u, v) \in ([0, 1], [0, 1])$ within the patch, other

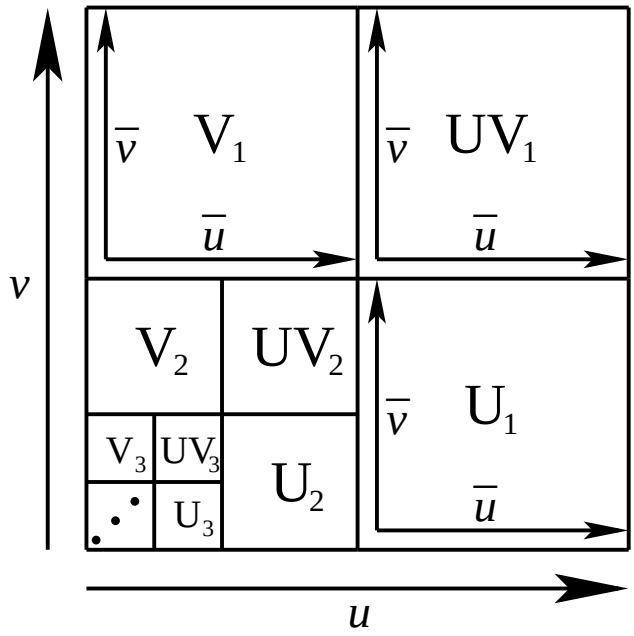


Figure 8.2: *Catmull-Clark parameterization*

than the origin, can be evaluated by repeatedly subdividing a number of generations g such that the parametric point is contained in one of the three regular B-spline subpatches (Figure 8.2).

$$g_u = max\left(\lceil -\log_2 u \rceil, 1\right) \tag{8.1}$$

$$g_v = max\left(\lceil -\log_2 v \rceil, 1\right) \tag{8.2}$$

$$g = min\left(g_u, g_v\right) \tag{8.3}$$

Then the transformed parametric values $(\bar{u}, \bar{v})$ can be substituted into the B-spline polynomial to compute the result.

$$\begin{aligned} U_g : g_u < g_v : \bar{u} &= 2^g u - 1 \\ \bar{v} &= 2^g v \end{aligned} \tag{8.4}$$

$$\begin{aligned} V_g : g_u > g_v : \bar{u} &= 2^g u \\ \bar{v} &= 2^g v - 1 \end{aligned} \tag{8.5}$$

$$\begin{aligned} UV_g : g_u = g_v : \bar{u} &= 2^g u - 1 \\ \bar{v} &= 2^g v - 1 \end{aligned} \tag{8.6}$$

For a stationary scheme such as this, performing g subdivisions is equivalent to raising the constant 1-patch-ring subdivision matrix A to the power g, which can be achieved in constant-time by

diagonalizing A into its eigen-decomposition.

For the smooth Catmull-Clark subdivision rules, the A matrix is stationary for the regular neighborhood of a quad patch with a single extraordinary valence vertex and all the surrounding facets also being quads. After at most two complete subdivisions of a closed, smooth Catmull-Clark mesh, all of the quads have such a regular neighborhood, so this technique can be applied. The same technique with the same A matrix can be used to perform exact evaluation of Catmull-Clark surfaces with semi-sharp edge tags [6]. Semi-sharp edges make the subdivision scheme non-stationary, but only for a finite number of generations, after which the entire mesh becomes a smooth Catmull-Clark surface. Infinitely sharp edges and vertices are present throughout the subdivision generations so a new method must be introduced.

8.2.1 Behavior Near Sharp Features

We extend the smooth method to the case of piecewise smooth Catmull-Clark subdivision. Vertices along chains of sharp tagged edges are averaged by the uniform cubic B-spline curve rules rather than the surface averaging rules. The averaging of these crease vertices is only dependent on vertices along the curve, and this curve creates a sharp boundary across which no surface smoothing information is passed. These modified averaging rules lead to four additional extraordinary vertex cases, and we will address the evaluation of patches near vertices of these types.

A dart vertex (Figure 8.1(b)) has one sharp edge incident on it, but it is still averaged by the smooth Catmull-Clark vertex rule, forming a transition from the feature curve to the smooth surface. A crease vertex (Figure 8.1(c)) has two sharp edges incident on it, and it is smoothed by the B-spline subdivision curve rule. A spike vertex (Figure 8.1(d)) is tagged not to move, while the rest of the surface around it follows the standard Catmull-Clark rules, leading to geometry resembling the tip of a cone. A corner vertex is defined to be a vertex with 3 or more sharp edges incident on it (Figure 8.1(e)), or a vertex with 1 or more sharp edges where the vertex is also tagged to not move (Figure 8.1(f)). Similar to the smooth extraordinary vertex case, the behavior of the surface near these extraordinary vertices is a shrinking ribbon of B-spline patches (Figure 8.3), (details in Section 8.4).

For any quad patch p next to an extraordinary vertex, the behavior of the ribbon of B-spline patches as they converge to the extraordinary vertex is defined by the subdivision rules of the local neighborhood, which can be written in a subdivision matrix A (Figure 8.3). For a patch adjacent to a smooth or spike vertex, A is dependent on the valence n of the vertex, but it is the same

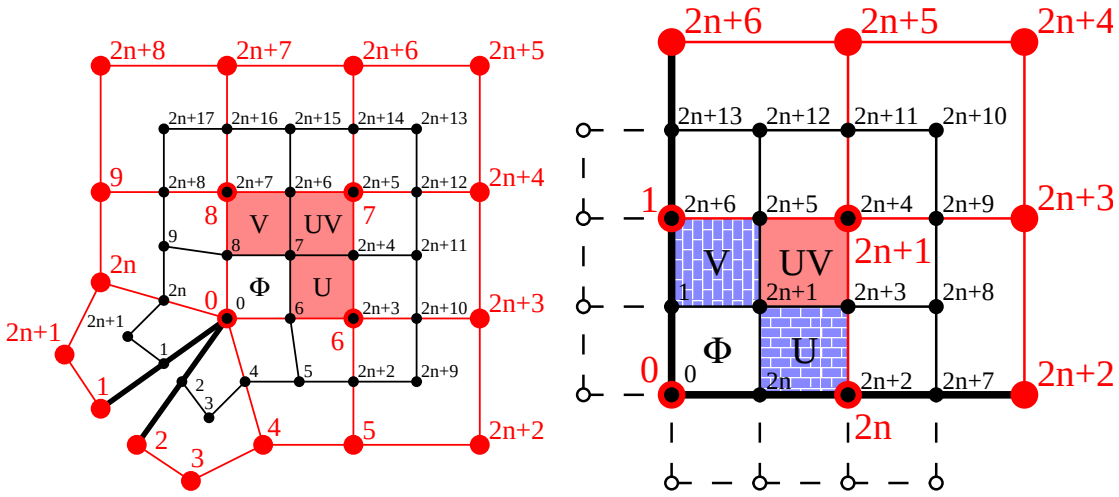


Figure 8.3: *Application of the $\bar{A}$ matrix to a quad patch next to an extraordinary vertex produces a ribbon of three subpatches $\{U, UV, V\}$ that are regular B-spline patches and a fourth patch Φ which is analogous to its parent. Patches not adjacent to any sharp edges (a) generate B-spline subpatches with a full set of 16 control points, while patches adjacent to sharp edges generate B-spline subpatches, e.g. $\{U, V\}$ in (b), that must calculate phantom vertices. Light vertices and edges represent the current generation, while black ones represent its child mesh.*

for any adjacent patch p, due to the structural symmetry around these types of vertices. Sharp edges prevent smoothing across them, so the behavior of the surface adjacent to dart, crease, or corner vertices is broken into isolated wedges. A wedge is a radially consecutive set of quads starting and ending at a sharp edge. The number of quads in a wedge is its valence n, and each quad is identified by a number p counting counterclockwise around the extraordinary vertex. The subdivision matrix A is different for each patch p of a wedge and also depends on the valence n of the wedge. There is mirror symmetry about the central quad or edge, so for example A for patch $p = 1$ is the mirror of patch $p = n$ (Figure 8.1(f)). This symmetry reduces the number of cases.

For some valences, A is defective and cannot be diagonalized, so it is not possible to use the eigen-decomposition to apply A^g. In these cases, we utilize the Jordan decomposition of A which always exists and is only slightly more expensive to raise to a power g.

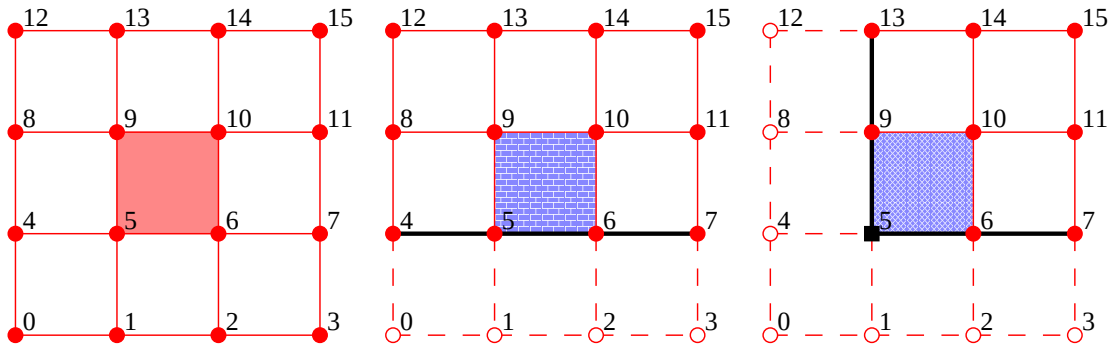
We show that the subdivision rules near sharp features produce B-spline patches and can be applied for an arbitrary number of generations by providing the Jordan decompositions of the subdivsion matrices for the different extraordinary vertex types.

8.3 Related Work

Our method is a generalization of the exact evaluation of smooth Catmull-Clark surfaces [29]. Infinitely sharp edge and vertex rules [10, 6] change the subdivision matrix A, and for some cases these matrices cannot be diagonalized. Parameterized expanded tagging rules with more control have also been defined [4], and follow up work [34] has shown how to evaluate Loop subdivision surfaces with these extended rules. Some of these subdivision matrices are defective, and the authors chose to use a novel matrix decomposition rather than the Jordan decompositions which correspond to singularities and are hard to parameterize over a family of matrices. In addition to the work on sharp Loop surfaces, the authors of [34] mention additional unpublished work about piecewise smooth Catmull-Clark surfaces as well as unpublished results by J. Stam that have been implemented in Alias|Wavefront's Maya [2]. Though we only handle the subset these averaging rules, we still do implement the full RendMan standard [21] which is a very useful set of rules. The Jordan decomposition uses less matrix components because it has at most a single 2×2 block while the other method has many 6×6 blocks. Hence our method is faster at run-time.

8.4 Regular B-spline Cases

To show that Catmull-Clark surfaces with infinitely sharp edges and vertices can be exactly evaluated in constant-time, we show that the subdivision operator produces uniform bi-cubic B-spline patches.



(a) *Smooth* (b) *Crease* (c) *Corner*

Figure 8.4: *Catmull-Clark patches that are uniform bi-cubic B-spline patches. Black thick lines and boxes are sharp edges and vertices. Light solid lines and circles are smooth edges and vertices. Light dashed lines and hollow circles are phantom edges and vertices.*

Figure 8.5 shows that subdividing a patch near a smooth extraordinary vertex creates shrinking rings of uniform bi-cubic B-spline patches $\{U, V, UV\}$, which are known polynomial functions, enabling exact evaluation. For many patches near sharp features, the same kind of B-spline patches are generated during the subdivision process (Figure 8.3(a)). For patches adjacent to sharp edges, on the other hand, some of the subpatches will then be adjacent to sharp edges with an incomplete neighborhood of control points to define a B-spline patch, e.g. subpatches U and V in Figure 8.3(b).

Fortunately, these subpatches are also uniform bi-cubic B-spline patches, and a set of phantom vertices can be easily constructed based on the existing control points to complete the control structure [33]. Further investigation has shown that there are three local cases of Catmull-Clark patches that reduce to uniform B-splines if appropriate phantom vertices are introduced (Figure 8.4). Figure 8.4(a) shows a uniform bi-cubic B-spline control structure.

The 16 uniform bi-cubic B-splines basis functions can be expressed as $B_3(u,v) = p_{uv}(u,v)\, B_{uv}$, where $p_{uv}(u,v)$ is the bi-cubic power basis vector:

$$p_u(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \tag{8.7}$$

$$p_{uv}(u,v) = \begin{bmatrix} p_u(u) & v p_u(u) & v^2 p_u(u) & v^3 p_u(u) \end{bmatrix} \tag{8.8}$$

B_{uv} is the coefficient matrix on the power basis that constructs the uniform bi-cubic B-splines:

$$B_u = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{8.9}$$

$$B_{uv} = \frac{1}{6}\begin{bmatrix} B_u & 4B_u & B_u & 0 \\ -3B_u & 0 & 3B_u & 0 \\ 3B_u & -6B_u & 3B_u & 0 \\ -B_u & 3B_u & -3B_u & B_u \end{bmatrix} \tag{8.10}$$

Figure 8.4(b) illustrates the case of a patch adjacent to a regular crease edge. The necessary location for phantom vertices $0 \rightarrow 3$ are computed by mirroring their respective neighbor vertex across the sharp crease edge. For the example of vertex V_1, we must set:

$$V_1 = 2V_5 - V_9 \tag{8.11}$$

Equation 8.11 is derived by equating the $v = 0$ isoparameter curve of the phantom patch to the sharp boundary B-spline curve of the patch.

$$B(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\, b_j(u)\, V_{4j+i} \tag{8.12}$$

$$B(u,0) = \sum_{i=0}^{3} b_i(u)\, \frac{1}{6}\left(V_i + 4V_{4+i} + V_{8+i}\right) \tag{8.13}$$

$$C(u) = \sum_{i=0}^{3} b_i(u)\, V_{4+i} \tag{8.14}$$

$$C(u) = B(u,0) \tag{8.15}$$

$$V_{4+i} = \frac{1}{6}\left(V_i + 4V_{4+i} + V_{8+i}\right) \tag{8.16}$$

$$V_i = 2V_{4+i} - V_{8+i} \tag{8.17}$$

$$V_1 = 2V_5 - V_9 \tag{8.18}$$

Figure 8.4(c) shows a patch adjacent to a corner vertex and two regular crease edges. Identical to the single crease case, locations for phantom vertices $\{1,2,3,4,8,12\}$ are computed by mirroring their associated neighbor vertex across the sharp crease edges. The location for the corner phantom vertex V_0 is computed by:

$$V_0 = 4V_5 + V_{10} - 2\left(V_6 + V_9\right) \tag{8.19}$$

Equation 8.19 is derived by equating the $u = 0$ and $v = 0$ isoparameter curves of the phantom patch to the sharp boundary curves of the patch. But first, sharp boundary curves must have a fourth phantom curve control point calculated for them in the same way, see Equation 8.22.

$$B(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\, b_j(u)\, V_{4j+i} \tag{8.20}$$

$$B(u,0) = \sum_{i=0}^{3} b_i(u)\, \frac{1}{6}\left(V_i + 4V_{4+i} + V_{8+i}\right) \qquad B(0,v) = \sum_{j=0}^{3} b_j(v)\, \frac{1}{6}\left(V_{4j} + 4V_{4j+1} + V_{4j+2}\right) \tag{8.21}$$

$$C(u) = b_0(u)\left(2V_5 - V_6\right) + \sum_{i=1}^{3} b_i(u)\, V_{4+i} \qquad C(v) = b_0(v)\left(2V_5 - V_9\right) + \sum_{i=1}^{3} b_i(u)\, V_{4j+1} \tag{8.22}$$

$$C(u) = B(u, 0) \qquad C(v) = B(0, v) \tag{8.23}$$

$$2V_5 - V_6 = \tfrac{1}{6}(V_0 + 4V_4 + V_8) \qquad 2V_5 - V_9 = \tfrac{1}{6}(V_0 + 4V_1 + V_2) \tag{8.24}$$

$$2V_5 - V_6 = \tfrac{1}{6}(V_0 + 4(2V_5 - V_6) + (2V_9 - V_{10})) \quad 2V_5 - V_9 = \tfrac{1}{6}(V_0 + 4(2V_5 - V_9) + (2V_6 - V_{10})) \tag{8.25}$$

$$V_0 = 2(2V_5 - V_6) - (2V_9 - V_{10}) \qquad V_0 = 2(2V_5 - V_9) - (2V_6 - V_{10}) \tag{8.26}$$

$$V_0 = 4V_5 - 2(V_6 + V_9) + V_{10} \qquad V_0 = 4V_5 - 2(V_6 + V_9) + V_{10} \tag{8.27}$$

$$V_0 = 4V_5 - 2(V_6 + V_9) + V_{10} \tag{8.28}$$

These are the basic elements that build all uniform B-spline patches, but combinations of the regular crease and corner situations are also uniform B-spline patches. A quad with two parallel regular crease edges and a quad with all four corner vertices are both uniform bi-cubic B-splines; the latter actually reduces to a bilinear patch. We have found that treating the regular corner case specially can greatly reduce the amount of subdivision that is necessary. For example, if a model is generated that texture maps each quad with its own $\{[0, 1], [0, 1]\}$ texture space, then each texture domain is a bilinear patch and no subdivision is necessary. In principle though, only the smooth and regular crease cases are necessary to perform exact evaluation.

8.5 Catmull-Clark Smooth and Spike Vertices

The smooth and spike vertex patch cases can be treated with the same matrix analysis. The only difference between the two cases is the averaging rule for the extraordinary vertex V_0 of Figure 8.5.

8.5.1 Catmull-Clark Smooth and Spike $\bar{A}$

The extended 1-ring subdivision operator $\bar{A}$ generates all the necessary control vertices in the next generation that define the three regular bi-cubic B-Spline patches, see Figure 8.5. We choose to order the vertices in the 1-ring neighborhood as illustrated in Figure 8.5 to simplify our analysis.

$$v = \frac{1}{64}, e = \frac{1}{16}, f = \frac{1}{4}$$

$$\bar{A}_n = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \\ \hline\hline S_{21} & S_{22} \end{array}\right]$$

$$= \left[\begin{array}{c|c|c|c|c|c|c|c|c||ccccccc} v_0 & v_1 & v_2 & v_1 & v_2 & v_1 & \cdots & v_1 & v_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline e_0 & e_1 & e_2 & e_3 & 0 & 0 & & e_3 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & f_2 & f_1 & f_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_3 & e_2 & e_1 & e_2 & e_3 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & 0 & 0 & f_2 & f_1 & f_2 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & \vdots & & & \ddots & & \vdots & & & & & \vdots & & & \\ f_0 & 0 & 0 & 0 & & f_2 & f_1 & f_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_3 & 0 & 0 & \cdots & e_3 & e_2 & e_1 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & f_2 & 0 & 0 & & 0 & 0 & f_2 & f_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline\hline e & e & 6e & 6e & 0 & 0 & 0 & 0 & & e & e & 0 & 0 & 0 & 0 & 0 \\ 6v & v & 6v & 36v & 6v & v & 0 & 0 & & v & 6v & v & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 6e & 6e & e & 0 & 0 & & 0 & e & e & 0 & 0 & 0 & 0 \\ v & 0 & 0 & 6v & 36v & 6v & 0 & 0 & \cdots & 0 & v & 6v & v & 6v & v & 0 \\ e & 0 & 0 & e & 6e & 6e & 0 & 0 & & 0 & 0 & 0 & 0 & e & e & 0 \\ 6v & 0 & 0 & v & 6v & 36v & 6v & v & & 0 & 0 & 0 & 0 & v & 6v & v \\ e & 0 & 0 & 0 & 0 & 6e & 6e & e & & 0 & 0 & 0 & 0 & 0 & e & e \\ \hline\hline 0 & 0 & f & f & 0 & 0 & 0 & 0 & & f & f & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e & 6e & e & 0 & 0 & 0 & & e & 6e & e & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f & f & 0 & 0 & 0 & & 0 & f & f & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e & 6e & 0 & 0 & 0 & & 0 & e & 6e & e & e & 0 & 0 \\ 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & \cdots & 0 & 0 & f & f & f & 0 & 0 \\ 0 & 0 & 0 & 0 & 6e & e & 0 & 0 & & 0 & 0 & e & e & 6e & e & 0 \\ 0 & 0 & 0 & 0 & f & f & 0 & 0 & & 0 & 0 & 0 & 0 & f & f & 0 \\ 0 & 0 & 0 & 0 & e & 6e & e & 0 & & 0 & 0 & 0 & 0 & e & 6e & e \\ 0 & 0 & 0 & 0 & 0 & f & f & 0 & & 0 & 0 & 0 & 0 & 0 & f & f \end{array}\right] \tag{8.29}$$

When the valence $n = 3$, the shape of the S_0, S_{11}, and S_{21} matrices change slightly

because V_1 is being used twice.

$$\bar{A}_3 = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \\ \hline\hline S_{21} & S_{22} \end{array}\right]$$

$$= \left[\begin{array}{c|c|c|c|c|c|c||c|c|c|c|c|c|c} v_0 & v_1 & v_2 & v_1 & v_2 & v_1 & v_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline e_0 & e_1 & e_2 & e_3 & 0 & e_3 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & f_2 & f_1 & f_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_3 & e_2 & e_1 & e_2 & e_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & 0 & 0 & f_2 & f_1 & f_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_3 & 0 & e_3 & e_2 & e_1 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & f_2 & 0 & 0 & 0 & f_2 & f_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline\hline e & e & 6e & 6e & 0 & 0 & 0 & e & e & 0 & 0 & 0 & 0 & 0 \\ 6v & v & 6v & 36v & 6v & v & 0 & v & 6v & v & 0 & 0 & 0 & 0 \\ e & 0 & 0 & 6e & 6e & e & 0 & 0 & e & e & 0 & 0 & 0 & 0 \\ v & 0 & 0 & 6v & 36v & 6v & 0 & 0 & v & 6v & v & 6v & v & 0 \\ e & 0 & 0 & e & 6e & 6e & 0 & 0 & 0 & 0 & 0 & e & e & 0 \\ 6v & v & 0 & v & 6v & 36v & 6v & 0 & 0 & 0 & 0 & v & 6v & v \\ e & e & 0 & 0 & 0 & 6e & 6e & 0 & 0 & 0 & 0 & 0 & e & e \\ \hline\hline 0 & 0 & f & f & 0 & 0 & 0 & f & f & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e & 6e & e & 0 & 0 & e & 6e & e & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & f & f & 0 & 0 & 0 & f & f & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e & 6e & 0 & 0 & 0 & e & 6e & e & e & 0 & 0 \\ 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & f & f & f & 0 & 0 \\ 0 & 0 & 0 & 0 & 6e & e & 0 & 0 & 0 & e & e & 6e & e & 0 \\ 0 & 0 & 0 & 0 & f & f & 0 & 0 & 0 & 0 & 0 & f & f & 0 \\ 0 & 0 & 0 & 0 & e & 6e & e & 0 & 0 & 0 & 0 & e & 6e & e \\ 0 & 0 & 0 & 0 & 0 & f & f & 0 & 0 & 0 & 0 & 0 & f & f \end{array}\right] \tag{8.30}$$

8.5.2 Catmull-Clark Smooth and Spike A Eigen-decomposition

The subdivision operator A is the top square portion of $\bar{A}$.

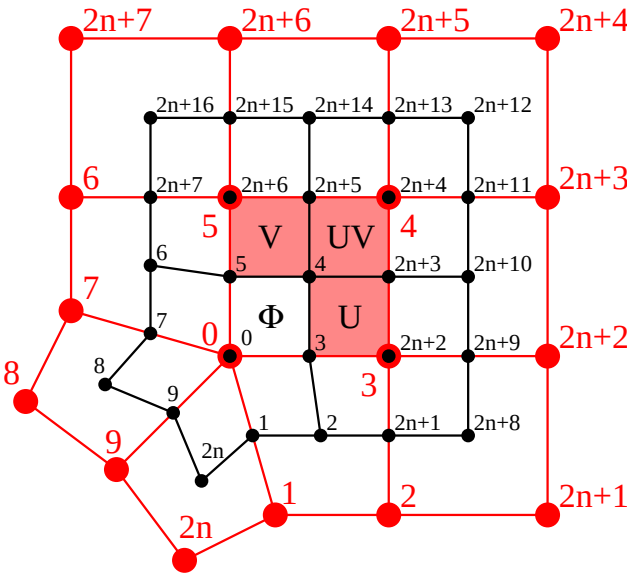


Figure 8.5: *Application of the $\bar{A}$ matrix to a quad patch next to an extraordinary smooth or spike vertex produces a ribbon of three subpatches $\{U, UV, V\}$ that are regular bi-cubic B-Spline patches and a fourth patch Φ which is analogous to its parent. Light vertices and edges represent the current generation, while black ones represent its child mesh.*

$$A = \left[\begin{array}{c|c} S_0 & 0 \\ \hline S_{11} & S_{12} \end{array}\right] \tag{8.31}$$

For all valences, the subdivision operator A is not defective and can be diagonalized into its eigen-decomposition.

8.5.3 Catmull-Clark Smooth and Spike S_{12}

The submatrix S_{12} is the same for all valences.

$$S_{12} = \begin{bmatrix} e & e & 0 & 0 & 0 & 0 & 0 \\ v & 6v & v & 0 & 0 & 0 & 0 \\ 0 & e & e & 0 & 0 & 0 & 0 \\ 0 & v & 6v & v & 6v & v & 0 \\ 0 & 0 & 0 & 0 & e & e & 0 \\ 0 & 0 & 0 & 0 & v & 6v & v \\ 0 & 0 & 0 & 0 & 0 & e & e \end{bmatrix} \tag{8.32}$$

We substitute $e = 4v$ to simplify the expressions in the eigen-decomposition.

$$S_{12} = \begin{bmatrix} 4v & 4v & 0 & 0 & 0 & 0 & 0 \\ v & 6v & v & 0 & 0 & 0 & 0 \\ 0 & 4v & 4v & 0 & 0 & 0 & 0 \\ 0 & v & 6v & v & 6v & v & 0 \\ 0 & 0 & 0 & 0 & 4v & 4v & 0 \\ 0 & 0 & 0 & 0 & v & 6v & v \\ 0 & 0 & 0 & 0 & 0 & 4v & 4v \end{bmatrix} \tag{8.33}$$

S_{12} can be diagonalized into its eigen-decomposition. Due to the block lower triangular structure of A, the columns of V_{12} when zero extended on the top become eigenvectors of A.

$$\begin{aligned} S_{12} =& V_{12}\Lambda_{12}V_{12}^{-1} \\ =& \begin{bmatrix} 1 & -1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 2 & 11 & 1 & 11 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} 8v & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4v & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2v & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2v & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4v & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8v \end{bmatrix} \cdot \\ & \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ -3 & 0 & 3 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ -6 & 18 & -18 & 6 & -18 & 18 & -6 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 3 & 0 & -3 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 \end{bmatrix} \end{aligned} \tag{8.34}$$

8.5.4 Catmull-Clark Smooth and Spike S_0 Eigen-decomposition

For the most general set of 1-ring averaging rules, the S_0 matrix has an interlaced circulant submatrix plus an additional row and column associated with the central control vertex V_0.

$$S_0 = \left[\begin{array}{c|cccccccc} v_0 & v_1 & v_2 & v_1 & v_2 & v_1 & \cdots & v_1 & v_2 \\ \hline e_0 & e_1 & e_2 & e_3 & e_4 & e_5 & & e_3 & e_2 \\ f_0 & f_2 & f_1 & f_2 & f_3 & f_4 & & f_4 & f_3 \\ e_0 & e_3 & e_2 & e_1 & e_2 & e_3 & & e_5 & e_4 \\ f_0 & f_4 & f_3 & f_2 & f_1 & f_2 & & f_6 & f_5 \\ \vdots & & \vdots & & & \ddots & & \vdots & \\ f_0 & f_4 & f_5 & f_6 & f_7 & f_8 & & f_2 & f_3 \\ e_0 & e_3 & e_4 & e_5 & e_6 & e_7 & & e_1 & e_2 \\ f_0 & f_2 & f_3 & f_4 & f_5 & f_6 & & f_2 & f_1 \end{array}\right] \tag{8.35}$$

For a convergent subdivision scheme, the subdivision rules for the face, edge, and vertex rules must each sum to one.

$$v_0 = 1 - n\left(v_1 + v_2\right) \tag{8.36}$$

$$e_0 = 1 - \left(e_1 + 2\sum_{k=2}^{n} e_k + e_{n+1}\right) \tag{8.37}$$

$$f_0 = 1 - \left(f_1 + 2\sum_{k=2}^{n} f_k + f_{n+1}\right) \tag{8.38}$$

The S_0 matrix can be transformed into a block diagonal matrix $\hat{S}_0$ using a modified DFT.

$$\hat{S}_0 = F S_0 F^{-1} \tag{8.39}$$

$$F = \frac{1}{n}\left[\begin{array}{c|cccccccc} n & 0 & 0 & 0 & 0 & & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \omega^1 & 0 & & \omega^{n-1} & 0 \\ 0 & 0 & 1 & 0 & \omega^1 & & 0 & \omega^{n-1} \\ & & \vdots & & & \ddots & \vdots & \\ 0 & 1 & 0 & \omega^{n-1} & 0 & \cdots & \omega^{(n-1)(n-1)} & 0 \\ 0 & 0 & 1 & 0 & \omega^{n-1} & & 0 & \omega^{(n-1)(n-1)} \end{array}\right] \tag{8.40}$$

$$F^{-1} = \left[\begin{array}{c|cccccccc} 1 & 0 & 0 & 0 & 0 & & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \omega^{-1} & 0 & & \omega^{-(n-1)} & 0 \\ 0 & 0 & 1 & 0 & \omega^{-1} & & 0 & \omega^{-(n-1)} \\ & & \vdots & & & \ddots & \vdots & \\ 0 & 1 & 0 & \omega^{-(n-1)} & 0 & \cdots & \omega^{-(n-1)(n-1)} & 0 \\ 0 & 0 & 1 & 0 & \omega^{-(n-1)} & & 0 & \omega^{-(n-1)(n-1)} \end{array}\right] \tag{8.41}$$

$\hat{S}_0$ has n blocks where B_0 is 3×3 while each B_i for $i \in \{1, n-1\}$ is 2×2.

$$\hat{S}_0 = \left[\begin{array}{c|ccc} B_0 & & & \\ \hline & \boxed{B_1} & & \\ & & \ddots & \\ & & & \boxed{B_{n-1}} \end{array}\right] \tag{8.42}$$

When the valence n is even, the blocks have the following form:

$$B_0 = \begin{bmatrix} v_0 & n v_1 & n v_2 \\ e_0 & e_1 + 2\sum_{k=1}^{\frac{n-2}{2}} e_{1+2k} + e_{n+1} & 2\sum_{k=1}^{\frac{n}{2}} e_{2k} \\ f_0 & 2\sum_{k=1}^{\frac{n}{2}} f_{2k} & f_1 + 2\sum_{k=1}^{\frac{n-2}{2}} f_{1+2k} + f_{n+1} \end{bmatrix} \tag{8.43}$$

$$B_i = \begin{bmatrix} e_1 + 2\sum_{k=1}^{\frac{n-2}{2}} \cos\left(\frac{2\pi i k}{n}\right) e_{1+2k} + \cos(i\pi)\, e_{n+1} & \sum_{k=1}^{\frac{n}{2}} e_{2k} \left(\omega^{-i(k-1)} + \omega^{ik}\right) \\ \sum_{k=1}^{\frac{n}{2}} f_{2k} \left(\omega^{i(k-1)} + \omega^{-ik}\right) & f_1 + 2\sum_{k=1}^{\frac{n-2}{2}} \cos\left(\frac{2\pi i k}{n}\right) f_{1+2k} + \cos(i\pi)\, f_{n+1} \end{bmatrix} \tag{8.44}$$

When the valence n is odd, the blocks have the following form:

$$B_0 = \begin{bmatrix} v_0 & nv_1 & nv_2 \\ e_0 & e_1 + 2\sum_{k=1}^{\frac{n-1}{2}} e_{1+2k} & 2\sum_{k=1}^{\frac{n-1}{2}} e_{2k} + e_{n+1} \\ f_0 & 2\sum_{k=1}^{\frac{n-1}{2}} f_{2k} + f_{n+1} & f_1 + 2\sum_{k=1}^{\frac{n-1}{2}} f_{1+2k} \end{bmatrix} \tag{8.45}$$

$$B_i = \begin{bmatrix} e_1 + 2\sum_{k=1}^{\frac{n-1}{2}} \cos\left(\frac{2\pi ik}{n}\right) e_{1+2k} & \sum_{k=1}^{\frac{n-1}{2}} e_{2k}\left(\omega^{-i(k-1)} + \omega^{ik}\right) + e_{n+1}\omega^{-i\frac{n-1}{2}} \\ \sum_{k=1}^{\frac{n-1}{2}} f_{2k}\left(\omega^{i(k-1)} + \omega^{-ik}\right) + f_{n+1}\omega^{i\frac{n-1}{2}} & f_1 + 2\sum_{k=1}^{\frac{n-1}{2}} \cos\left(\frac{2\pi ik}{n}\right) f_{1+2k} \end{bmatrix} \tag{8.46}$$

All of the values in the B_0 block are real, but the off diagonal elements of the B_i blocks are imaginary. The product of these two elements appears in the eigenvalue formula, and if we set $e_{n+1} = 0$ and $f_{n+1} = 0$ in the n odd case then this product will be real.

$$e_{n+1} = 0, f_{n+1} = 0$$

$$B_i = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \lambda_{\pm i} = \frac{a + d \pm \sqrt{(a-d)^2 + 4bc}}{2} \tag{8.47}$$

$$bc = 2\sum_{j=1}^{\lfloor\frac{n-1}{2}\rfloor}\sum_{k=1}^{\lfloor\frac{n-1}{2}\rfloor} f_{1+2j}e_{1+2k}\left(\cos\left(\tfrac{2\pi i(j-k)}{n}\right) + \cos\left(\tfrac{2\pi i(j+k-1)}{n}\right)\right) \tag{8.48}$$

For the original Catmull-Clark scheme, the S_0 matrix has many of the general face and edge averaging weights set to zero, so the matrix has the form:

$$S_0 = \left[\begin{array}{c|cccccccc} v_0 & v_1 & v_2 & v_1 & v_2 & v_1 & \cdots & v_1 & v_2 \\ \hline e_0 & e_1 & e_2 & e_3 & 0 & 0 & & e_3 & e_2 \\ f_0 & f_2 & f_1 & f_2 & 0 & 0 & \cdots & 0 & 0 \\ e_0 & e_3 & e_2 & e_1 & e_2 & e_3 & & 0 & 0 \\ f_0 & 0 & 0 & f_2 & f_1 & f_2 & & 0 & 0 \\ \vdots & & \vdots & & & \ddots & & \vdots & \\ f_0 & 0 & 0 & 0 & & f_2 & f_1 & f_2 & 0 \\ e_0 & e_3 & 0 & 0 & \cdots & e_3 & e_2 & e_1 & e_2 \\ f_0 & f_2 & 0 & 0 & & 0 & 0 & f_2 & f_1 \end{array}\right] \tag{8.49}$$

For a convergent subdivision scheme, the subdivision rules for the face, edge, and vertex rules must

each sum to one.

$$v_0 = 1 - n\left(v_1 + v_2\right) \tag{8.50}$$

$$e_0 = 1 - \left(e_1 + 2e_2 + 2e_3\right) \tag{8.51}$$

$$f_0 = 1 - \left(f_1 + 2f_2\right) \tag{8.52}$$

The values of the averaging rule weights adjacent to the extraordinary vertex can be modified to try to improve the surface behavior there, and it is convenient for us to leave these as variables so that we can combine the analysis for smooth and spike vertices. The standard face rules are:

$$f_0 = f, f_1 = f, f_2 = f \tag{8.53}$$

The standard edge rules are:

$$e_0 = 6e, e_1 = 6e, e_2 = e, e_3 = e \tag{8.54}$$

The standard smooth vertex rules are:

$$v_0 = 1 - \frac{7n}{4n^2}, v_1 = \frac{6}{4n^2}, v_2 = \frac{1}{4n^2} \tag{8.55}$$

The spike vertex rules are:

$$v_0 = 1, v_1 = 0, v_2 = 0 \tag{8.56}$$

The S_0 matrix can be transformed into a block diagonal matrix $\hat{S}_0$ using a modified DFT, and the blocks B_i for $i \in \{1, n-1\}$ have the following form:

$$B_0 = \begin{bmatrix} v_0 & nv_1 & nv_2 \\ e_0 & e_1 + 2e_3 & 2e_2 \\ f_0 & 2f_2 & f_1 \end{bmatrix} \tag{8.57}$$

$$B_i = \begin{bmatrix} e_1 + 2\cos\left(\frac{2\pi i}{n}\right)e_3 & e_2\left(1 + \omega^i\right) \\ f_2\left(1 + \omega^{-i}\right) & f_1 \end{bmatrix} \tag{8.58}$$

In the smooth case, the eigenvalues of B_0 are:

$$\lambda_0 = 1 \tag{8.59}$$

$$\lambda_{\pm 0} = \frac{1}{2}\left(\begin{array}{l} e_1 + 2e_3 + f_1 - nv_1 - nv_2 \pm \\ \sqrt{\begin{array}{l} \left(e_1 + 2e_3 + f_1 - nv_1 - nv_2\right)^2 + \\ 4\left(nv_1 f_1 + \left(nv_2 - f_1\right)\left(e_1 + 2e_3\right) + 4e_2 f_2 - 2nv_1 e_2 - 2nv_2 f_2\right) \end{array}} \end{array} \right) \tag{8.60}$$

$$= \frac{-7 + 3n \pm \sqrt{49 - 30n + 5n^2}}{8n} \tag{8.61}$$

In the spike case, the eigenvalues of B_0 have the simpler form:

$$\lambda_{\pm 0} = \frac{e_1 + 2e_3 + f_1 \pm \sqrt{(e_1 + 2e_3 + f_1)^2 + 4\left(4e_2 f_2 - f_1\left(e_1 + 2e_3\right)\right)}}{2} \tag{8.62}$$

$$= \frac{3 \pm \sqrt{5}}{8} \tag{8.63}$$

The eigenvectors $\hat{V}_0$ of B_0 can be expressed with respect to the eigenvalues of B_0, so the same expressions can be used for both the smooth and spike cases.

$$\hat{V}_0 = \begin{bmatrix} \hat{v}_0 & \hat{v}_{+0} & \hat{v}_{-0} \end{bmatrix} \tag{8.64}$$

$$\hat{v}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \hat{v}_{\pm 0} = \begin{bmatrix} \lambda_{\pm 0}^2 - \left(e_1 + 2e_3 + f_1\right)\lambda_{\pm 0} + \left(\left(e_1 + 2e_3\right) f_1 - 4e_2 f_2\right) \\ e_0 \lambda_{\pm 0} + \left(2e_2 - f_2 + \left(e_1 + 2e_3\right) f_1 - 4e_2 f_2\right) \\ f_0 \lambda_{\pm 0} + \left(-\left(e_1 + 2e_3\right) + \left(e_1 + 2e_3\right) f_1 + 2f_2 - 4e_2 f_2\right) \end{bmatrix} \tag{8.65}$$

The inverse DFT can be applied to $\hat{S}_0$ to restore it to S_0. Substituting the eigendecompositions of both of these matrices into Equation 8.66 shows that the eigenvectors V of S_0 can be derived by applying the inverse DFT to the eigenvectors $\hat{V}$ of S_0.

$$S_0 = F^{-1} \hat{S}_0 F \tag{8.66}$$

$$= F^{-1} \hat{V} \Lambda \hat{V}^{-1} F \tag{8.67}$$

$$= V \Lambda V^{-1} \tag{8.68}$$

$$V = F^{-1} \hat{V} \tag{8.69}$$

$$V^{-1} = \hat{V}^{-1} F \tag{8.70}$$

The three eigenvectors associated with the eigenvalues of block B_0 are:

$$v_0 = \begin{bmatrix} 1 \\ \hline 1 \\ 1 \\ \hline \vdots \\ \hline 1 \\ 1 \end{bmatrix}, v_{\pm 0} = \begin{bmatrix} \lambda_{\pm 0}^2 - \left(e_1 + 2e_3 + f_1\right)\lambda_{\pm 0} + \left(\left(e_1 + 2e_3\right) f_1 - 4e_2 f_2\right) \\ \hline e_0 \lambda_{\pm 0} + \left(2e_2 - f_2 + \left(e_1 + 2e_3\right) f_1 - 4e_2 f_2\right) \\ f_0 \lambda_{\pm 0} + \left(-\left(e_1 + 2e_3\right) + \left(e_1 + 2e_3\right) f_1 + 2f_2 - 4e_2 f_2\right) \\ \hline \vdots \\ \hline e_0 \lambda_{\pm 0} + \left(2e_2 - f_2 + \left(e_1 + 2e_3\right) f_1 - 4e_2 f_2\right) \\ f_0 \lambda_{\pm 0} + \left(-\left(e_1 + 2e_3\right) + \left(e_1 + 2e_3\right) f_1 + 2f_2 - 4e_2 f_2\right) \end{bmatrix} \tag{8.71}$$

The 2×2 blocks B_i for $i \in \{1, n-1\}$ each have the two eigenvalues $\lambda_{\pm i}$.

$$\lambda_{\pm i} = \frac{e_1 + 2\cos\left(\frac{2\pi i}{n}\right) e_3 + f_1 \pm \sqrt{\left(e_1 + 2\cos\left(\frac{2\pi i}{n}\right) e_3 - f_1\right)^2 + 8\left(1 + \cos\left(\frac{2\pi i}{n}\right)\right) e_2 f_2}}{2} \tag{8.72}$$

$$= \frac{5 + \cos\left(\frac{2\pi i}{n}\right) \pm \cos\left(\frac{\pi i}{n}\right) \sqrt{2\left(9 + \cos\left(\frac{2\pi i}{n}\right)\right)}}{16} \tag{8.73}$$

Note that $\lambda_{+i} = \lambda_{+(n-i)}$ and $\lambda_{-i} = \lambda_{-(n-i)}$ because the indices i and $n-i$ correspond to opposite multiples of the angle $\frac{2\pi}{n}$ and the eigenvalues only depend on the cosine of these angles which are then the same value. Also, if n is even, then $\lambda_{+\frac{n}{2}} = \lambda_{-\frac{n}{2}}$ because the discriminant is zero. So all eigenvalues from the 2×2 blocks occur in pairs. An eigenvector v_i is associated with the eigenvalue λ_i where i can be $+i$ or $-i$. The eigenvectors $\hat{v}_i$ and $\hat{v}_{n-i}$ are complex congugates of each other.

$$\hat{V}_i = \begin{bmatrix} \hat{v}_{+i} & \hat{v}_{-i} \end{bmatrix} = \begin{bmatrix} \frac{\lambda_{+i} - f_1}{f_2} & \frac{\lambda_{-i} - f_1}{f_2} \\ 1 + \omega^i & 1 + \omega^i \end{bmatrix} \tag{8.74}$$

$$\hat{V}_i^{-1} = \begin{bmatrix} \hat{v}_{+i}^{-1} \\ \hat{v}_{-i}^{-1} \end{bmatrix} = \begin{bmatrix} \frac{f_2}{\lambda_{+i} - \lambda_{-i}} & -\frac{\lambda_{-i} - f_1}{(\lambda_{+i} - \lambda_{-i})(1+\omega^i)} \\ -\frac{f_2}{\lambda_{+i} - \lambda_{-i}} & \frac{\lambda_{+i} - f_1}{(\lambda_{+i} - \lambda_{-i})(1+\omega^i)} \end{bmatrix} \tag{8.75}$$

$$= \begin{bmatrix} \frac{f_2}{\lambda_{+i} - \lambda_{-i}} & -\frac{(\lambda_{-i} - f_1)\left(1+\omega^{-i}\right)}{2\left(1+\cos\left(\frac{2\pi i}{n}\right)\right)(\lambda_{+i} - \lambda_{-i})} \\ -\frac{f_2}{\lambda_{+i} - \lambda_{-i}} & \frac{(\lambda_{+i} - f_1)\left(1+\omega^{-i}\right)}{2\left(1+\cos\left(\frac{2\pi i}{n}\right)\right)(\lambda_{+i} - \lambda_{-i})} \end{bmatrix} \tag{8.76}$$

The pairs of eigenvectors v_{+i} and v_{-i} are treated analogously, so we will drop the $\pm$ distinction. The inverse DFT transformed eigenvectors v_i for $i \in \pm\{1, n-1\}$ are of the form:

$$v_i = F^{-1}\hat{v}_i = \begin{bmatrix} 0 \\ \frac{\lambda_i - f_1}{f_2}\omega^{-i0} \\ \omega^{-i0} + \omega^{-i(0-1)} \\ \vdots \\ \frac{\lambda_i - f_1}{f_2}\omega^{-ik} \\ \omega^{-ik} + \omega^{-i(k-1)} \\ \vdots \\ \frac{\lambda_i - f_1}{f_2}\omega^{-i(n-1)} \\ \omega^{-i(n-1)} + \omega^{-i((n-1)-1)} \end{bmatrix} \tag{8.77}$$

These eigenvectors have imaginary entries which is undesirable. Fortunately, the complex congugate eigenvectors v_i and v_{n-i} associated with the paired eigenvalue $\lambda_i = \lambda_{n-i}$ can be combined to

form two real eigenvectors u_i and u_{n-i}.

$$\alpha(r) = \cos\left(\tfrac{2\pi r}{n}\right)$$
$$\beta(r) = \sin\left(\tfrac{2\pi r}{n}\right)$$

$$u_i = \frac{1}{2}\left(v_i + v_{n-i}\right) = \begin{bmatrix} 0 \\ \frac{\lambda_i - f_1}{f_2}\left(\alpha(i0)\right) \\ \alpha(i0) + \alpha(i(0+1)) \\ \vdots \\ \frac{\lambda_i - f_1}{f_2}\left(\alpha(ik)\right) \\ \alpha(ik) + \alpha(i(k+1)) \\ \vdots \\ \frac{\lambda_i - f_1}{f_2}\left(\alpha(i(n-1))\right) \\ \alpha(i(n-1)) + \alpha(i((n-1)+1)) \end{bmatrix} \tag{8.78}$$

$$u_{n-i} = \frac{\mathbf{i}}{2}\left(v_i - v_{n-i}\right) = \begin{bmatrix} 0 \\ \frac{\lambda_i - f_1}{f_2}\left(\beta(i0)\right) \\ \beta(i0) + \beta(i(0+1)) \\ \vdots \\ \frac{\lambda_i - f_1}{f_2}\left(\beta(ik)\right) \\ \beta(ik) + \beta(i(k+1)) \\ \vdots \\ \frac{\lambda_i - f_1}{f_2}\left(\beta(i(n-1))\right) \\ \beta(i(n-1)) + \beta(i((n-1)+1)) \end{bmatrix} \tag{8.79}$$

The corresponding row vectors from the inverse eigenvector matrix are then:

$$u_i^{-1} = \left(v_i^{-1} + v_{n-i}^{-1}\right), u_{n-i}^{-1} = -\mathbf{i}\left(v_i^{-1} - v_{n-i}^{-1}\right) \tag{8.80}$$

If the valence n is even, then block $B_{\frac{n}{2}}$ of $\hat{S}_0$ has two identical eigenvalues $\lambda_{\frac{n}{2}}$. The expression for the eigenvectors $\hat{V}_{\frac{n}{2}}$ in Equation 8.74 produce degenerate eigenvectors because $\omega^{\frac{n}{2}} = \omega^{-\frac{n}{2}} = \cos(\pi) = -1$ and $\lambda_{+\frac{n}{2}} = \lambda_{-\frac{n}{2}} = \frac{1}{4}$. In this case, the eigenvectors $\hat{V}_{\frac{n}{2}}$ of the 2×2 block are simply the identity matrix, which has a trivial inverse.

$$\hat{V}_{\frac{n}{2}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{8.81}$$

The corresponding eigenvectors of S_0 are derived by applying the inverse DFT:

$$v_{\pm\frac{n}{2}} = \begin{bmatrix} 0 \\ \omega^{-0\frac{n}{2}} \\ 0 \\ \vdots \\ \omega^{-k\frac{n}{2}} \\ 0 \\ \vdots \\ \omega^{-(n-1)\frac{n}{2}} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \omega^{-0\frac{n}{2}} \\ \vdots \\ 0 \\ \omega^{-k\frac{n}{2}} \\ \vdots \\ 0 \\ \omega^{-(n-1)\frac{n}{2}} \end{bmatrix} = \begin{bmatrix} 0 \\ \cos(0\pi) \\ 0 \\ \vdots \\ \cos(k\pi) \\ 0 \\ \vdots \\ \cos((n-1)\pi) \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ \cos(0\pi) \\ \vdots \\ 0 \\ \cos(k\pi) \\ \vdots \\ 0 \\ \cos((n-1)\pi) \end{bmatrix} \tag{8.82}$$

8.5.5 Catmull-Clark Smooth and Spike S_{11}

The eigenvector components V_{11} corresponding to S_{11} can be expressed with respect to the eigen-decompositions of S_0 and S_{12}. The columns of V_{11} can be computed using:

$$v_{11,c} = -V_{12}\left(\Lambda_{12} - \lambda_c I\right)^{-1} V_{12}^{-1} S_{11} v_{0,c} \tag{8.83}$$

8.5.6 Catmull-Clark Smooth and Spike Picking Matrices

The picking matrices P_k where $k \in \{U, V, UV\}$ select out the 16 control points for the corresponding uniform bi-cubic B-Spline patch from the $2n+16$ control points generated by the $\bar{A}$ matrix, see Figure 8.5. P_k is a $16 \times 2n+16$ matrix where each row has a single 1 in the column corresponding to control point listed in the sets below. Note that if the valence $n = 3$ then V_7 must be replaced by V_1.

$$\begin{aligned} P_{U,n} =& \{V_1, V_2, V_{2n+1}, V_{2n+8}, V_0, V_3, V_{2n+2}, V_{2n+9}, \\ & V_5, V_4, V_{2n+3}, V_{2n+10}, V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+11}\} \end{aligned} \tag{8.84}$$

$$\begin{aligned} P_{V,n} =& \{V_7, V_0, V_3, V_{2n+2}, V_6, V_5, V_4, V_{2n+3}, \\ & V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+16}, V_{2n+15}, V_{2n+14}, V_{2n+13}\} \end{aligned} \tag{8.85}$$

$$\begin{aligned} P_{UV,n} =& \{V_0, V_3, V_{2n+2}, V_{2n+9}, V_5, V_4, V_{2n+3}, V_{2n+10}, \\ & V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+11}, V_{2n+15}, V_{2n+14}, V_{2n+13}, V_{2n+12}\} \end{aligned} \tag{8.86}$$

8.6 Catmull-Clark Corner and Crease Vertices

The corner and crease vertex patch cases can be treated with the same matrix analysis. The only difference between the two cases is the averaging rule for the extraordinary vertex V_0 of Figure 8.3.

8.6.1 Catmull-Clark Corner and Crease $\bar{A}$

The extended 1-ring subdivision operator $\bar{A}$ generates all the necessary control vertices in the next generation that define the three bi-cubic B-Spline patches, see Figure 8.3. We choose to order the vertices in the 1-ring neighborhood as illustrated in Figure 8.3 to simplify our analysis. The first nonzero column of S_{11} and S_{21} is the one associated with vertex V_{2p-1}.

$$\bar{v}_0 = \frac{6}{8}, \bar{v}_1 = \frac{1}{8}, \bar{e}_0 = \bar{e}_1 = \frac{1}{2}$$

$$\bar{A}_{n,p} = \left[\begin{array}{c|c} S_0 & 0 \\ \hline S_{11} & S_{12} \\ \hline S_{21} & S_{22} \end{array}\right] \tag{8.87}$$

$$= \left[\begin{array}{ccc|cccccc|ccccccc} \bar{v}_0 & \bar{v}_1 & \bar{v}_1 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{e} & \bar{e} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{e} & 0 & \bar{e} & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline f_0 & 0 & f_2 & f_1 & f_2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & 0 & e_3 & e_2 & e_1 & e_2 & e_3 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & 0 & 0 & 0 & f_2 & f_1 & f_2 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & \vdots & & \vdots & & & \ddots & & \vdots & & & & \vdots & & & \\ f_0 & 0 & 0 & 0 & 0 & & f_2 & f_1 & f_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ e_0 & e_3 & 0 & 0 & 0 & \cdots & e_3 & e_2 & e_1 & e_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ f_0 & f_2 & 0 & 0 & 0 & & 0 & 0 & f_2 & f_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline e & & e & 6e & 6e & 0 & 0 & 0 & 0 & & e & e & 0 & 0 & 0 & 0 & 0 \\ 6v & & v & 6v & 36v & 6v & v & 0 & 0 & & v & 6v & v & 0 & 0 & 0 & 0 \\ e & & 0 & 0 & 6e & 6e & e & 0 & 0 & & 0 & e & e & 0 & 0 & 0 & 0 \\ v & \cdots & 0 & 0 & 6v & 36v & 6v & 0 & 0 & \cdots & 0 & v & 6v & v & 6v & v & 0 \\ e & & 0 & 0 & e & 6e & 6e & 0 & 0 & & 0 & 0 & 0 & 0 & e & e & 0 \\ 6v & & 0 & 0 & v & 6v & 36v & 6v & v & & 0 & 0 & 0 & 0 & v & 6v & v \\ e & & 0 & 0 & 0 & 0 & 6e & 6e & e & & 0 & 0 & 0 & 0 & 0 & e & e \\ \hline 0 & & 0 & f & f & 0 & 0 & 0 & 0 & & f & f & 0 & 0 & 0 & 0 & 0 \\ 0 & & 0 & e & 6e & e & 0 & 0 & 0 & & e & 6e & e & 0 & 0 & 0 & 0 \\ 0 & & 0 & 0 & f & f & 0 & 0 & 0 & & 0 & f & f & 0 & 0 & 0 & 0 \\ 0 & & 0 & 0 & e & 6e & 0 & 0 & 0 & & 0 & e & 6e & e & e & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & f & 0 & 0 & 0 & \cdots & 0 & 0 & f & f & f & 0 & 0 \\ 0 & & 0 & 0 & 0 & 6e & e & 0 & 0 & & 0 & 0 & e & e & 6e & e & 0 \\ 0 & & 0 & 0 & 0 & f & f & 0 & 0 & & 0 & 0 & 0 & 0 & f & f & 0 \\ 0 & & 0 & 0 & 0 & e & 6e & e & 0 & & 0 & 0 & 0 & 0 & e & 6e & e \\ 0 & & 0 & 0 & 0 & 0 & f & f & 0 & & 0 & 0 & 0 & 0 & 0 & f & f \end{array}\right] \tag{8.88}$$

When $p = 1$ or $p = n$, one side of the patch is adjacent to a crease edge, so the S_{22} submatrix is modified. Similar alterations are made to the S_{21}, S_{11}, and S_{12} submatrices. The eigen-decomposition of S_{12} will be discussed in greater detail in Section 8.6.3.

$$S_{22,n,1} = \begin{bmatrix} \bar{e} & 0 & 0 & 0 & 0 & 0 \\ f & f & 0 & 0 & 0 & 0 \\ e & 6e & e & e & 0 & 0 \\ 0 & f & f & f & 0 & 0 \\ 0 & e & e & 6e & e & 0 \\ 0 & 0 & 0 & f & f & 0 \\ 0 & 0 & 0 & e & 6e & e \\ 0 & 0 & 0 & 0 & f & f \end{bmatrix}, \qquad S_{22,n,n} = \begin{bmatrix} f & f & 0 & 0 & 0 & 0 \\ e & 6e & e & 0 & 0 & 0 \\ 0 & f & f & 0 & 0 & 0 \\ 0 & e & 6e & e & e & 0 \\ 0 & 0 & f & f & f & 0 \\ 0 & 0 & e & e & 6e & e \\ 0 & 0 & 0 & 0 & f & f \\ 0 & 0 & 0 & 0 & 0 & \bar{e} \end{bmatrix} \tag{8.89}$$

When the wedge valence $n = 1$ as in Figure 8.3(b), the shape of the S_0, S_{11}, and S_{21} matrices change slightly because the crease edge rules are present on both ends of the wedge in the second and third vertex rings.

$$\bar{A}_{1,1} = \left[\begin{array}{c||c} S_0 & 0 \\ \hline\hline S_{11} & S_{12} \\ \hline\hline S_{21} & S_{22} \end{array}\right] = \left[\begin{array}{ccc|c||ccccc} \bar{v}_0 & \bar{v}_1 & \bar{v}_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{e} & \bar{e} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{e} & 0 & \bar{e} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline f_0 & f_2 & f_2 & f_1 & 0 & 0 & 0 & 0 & 0 \\ \hline\hline \bar{v}_1 & 0 & \bar{v}_0 & 0 & \bar{v}_1 & 0 & 0 & 0 & 0 \\ e & e & 6e & 6e & e & e & 0 & 0 & 0 \\ v & 6v & 6v & 36v & v & 6v & v & 6v & v \\ e & 6e & e & 6e & 0 & 0 & 0 & e & e \\ \bar{v}_1 & \bar{v}_0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{v}_1 \\ \hline\hline 0 & 0 & \bar{e} & 0 & \bar{e} & 0 & 0 & 0 & 0 \\ 0 & 0 & f & f & f & f & 0 & 0 & 0 \\ 0 & 0 & e & 6e & e & 6e & e & e & 0 \\ 0 & 0 & 0 & f & 0 & f & f & f & 0 \\ 0 & e & 0 & 6e & 0 & e & e & 6e & e \\ 0 & f & 0 & f & 0 & 0 & 0 & f & f \\ 0 & \bar{e} & 0 & 0 & 0 & 0 & 0 & 0 & \bar{e} \end{array}\right] \tag{8.90}$$

8.6.2 Catmull-Clark Corner and Crease A Jordan Decompositions

The Jordan decompositions of A for corner and crease patches are very similar. For certain valences, the submatrix S_0 is defective due to interactions of shared eigenvalues between its submatrices T_0 and T_{12}.

8.6.3 Catmull-Clark Corner and Crease S_{12}

The submatrix S_{12}, n, p is the same as the smooth case from Equation 8.33 for most wedge valences n and subpatches p. However when $p = 1$ or $p = n$, the subpatch p of the wedge is adjacent to a crease edge and a boundary rule is introduced into S_{12}, Figure 8.3(b). S_{12} can be diagonalized into its eigen-decomposition. Due to the block lower triangular structure of A, the columns of V_{12} when zero extended on the top become eigenvectors of A. The three special cases near crease curves are as follows:

$$
\begin{aligned}
S_{12,n,1} &= \begin{bmatrix} \bar{v}_1 & 0 & 0 & 0 & 0 & 0 \\ e & e & 0 & 0 & 0 & 0 \\ v & 6v & v & 6v & v & 0 \\ 0 & 0 & 0 & e & e & 0 \\ 0 & 0 & 0 & v & 6v & v \\ 0 & 0 & 0 & 0 & e & e \end{bmatrix} = \begin{bmatrix} 8v & 0 & 0 & 0 & 0 & 0 \\ 4v & 4v & 0 & 0 & 0 & 0 \\ v & 6v & v & 6v & v & 0 \\ 0 & 0 & 0 & 4v & 4v & 0 \\ 0 & 0 & 0 & v & 6v & v \\ 0 & 0 & 0 & 0 & 4v & 4v \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 11 & 2 & 1 \\ 0 & 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 2 & -1 & 1 \end{bmatrix} \begin{bmatrix} 8v & 0 & 0 & 0 & 0 & 0 \\ 0 & 4v & 0 & 0 & 0 & 0 \\ 0 & 0 & v & 0 & 0 & 0 \\ 0 & 0 & 0 & 2v & 0 & 0 \\ 0 & 0 & 0 & 0 & 4v & 0 \\ 0 & 0 & 0 & 0 & 0 & 8v \end{bmatrix} \cdot \\
&\quad \frac{1}{6} \begin{bmatrix} 6 & 0 & 0 & 0 & 0 & 0 \\ -6 & 6 & 0 & 0 & 0 & 0 \\ 6 & -12 & 6 & -18 & 18 & -6 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 3 & 0 & -3 \\ 0 & 0 & 0 & 1 & 4 & 1 \end{bmatrix}
\end{aligned}
\tag{8.91}
$$

$$
\begin{aligned}
S_{12,n,n} &= \begin{bmatrix} e & e & 0 & 0 & 0 & 0 \\ v & 6v & v & 0 & 0 & 0 \\ 0 & e & e & 0 & 0 & 0 \\ 0 & v & 6v & v & 6v & v \\ 0 & 0 & 0 & 0 & e & e \\ 0 & 0 & 0 & 0 & 0 & \bar{v}_1 \end{bmatrix} = \begin{bmatrix} 4v & 4v & 0 & 0 & 0 & 0 \\ v & 6v & v & 0 & 0 & 0 \\ 0 & 4v & 4v & 0 & 0 & 0 \\ 0 & v & 6v & v & 6v & v \\ 0 & 0 & 0 & 0 & 4v & 4v \\ 0 & 0 & 0 & 0 & 0 & 8v \end{bmatrix} \\
&= \begin{bmatrix} 1 & -1 & 2 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 11 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8v & 0 & 0 & 0 & 0 & 0 \\ 0 & 4v & 0 & 0 & 0 & 0 \\ 0 & 0 & 2v & 0 & 0 & 0 \\ 0 & 0 & 0 & v & 0 & 0 \\ 0 & 0 & 0 & 0 & 4v & 0 \\ 0 & 0 & 0 & 0 & 0 & 8v \end{bmatrix} \cdot \\
&\quad \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 & 0 & 0 \\ -3 & 0 & 3 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ -6 & 18 & -18 & 6 & -12 & 6 \\ 0 & 0 & 0 & 0 & 6 & -6 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{bmatrix}
\end{aligned} \tag{8.92}
$$

$$
\begin{aligned}
S_{12,1,1} &= \begin{bmatrix} \bar{v}_1 & 0 & 0 & 0 & 0 \\ e & e & 0 & 0 & 0 \\ v & 6v & v & 6v & v \\ 0 & 0 & 0 & e & e \\ 0 & 0 & 0 & 0 & \bar{v}_1 \end{bmatrix} = \begin{bmatrix} 8v & 0 & 0 & 0 & 0 \\ 4v & 4v & 0 & 0 & 0 \\ v & 6v & v & 6v & v \\ 0 & 0 & 0 & 4v & 4v \\ 0 & 0 & 0 & 0 & 8v \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8v & 0 & 0 & 0 & 0 \\ 0 & 4v & 0 & 0 & 0 \\ 0 & 0 & v & 0 & 0 \\ 0 & 0 & 0 & 4v & 0 \\ 0 & 0 & 0 & 0 & 8v \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{8.93}
$$

8.6.4 Catmull-Clark Corner and Crease S_0 Jordan decomposition

The S_0 matrix for corner and crease vertices is only dependent on the valence n and not on the subpatch p. S_0 is a lower block triangular matrix. The subblock T differs for the corner and crease cases because of the averaging rules for the control vertex V_0. T_{11} and T_{12} are the same for both cases.

$$f = \frac{1}{4}, e = \frac{1}{16}$$

$$S_{0,n} = \left[\begin{array}{c|c} T & 0 \\ \hline T_{11,n} & T_{12,n} \end{array}\right] = \left[\begin{array}{ccc|cccccc} \multicolumn{3}{c|}{T} & \multicolumn{6}{c}{0} \\ \hline f & 0 & f & f & f & 0 & 0 & 0 & 0 & \\ 6e & 0 & e & e & 6e & e & e & 0 & 0 & \\ f & 0 & 0 & 0 & f & f & f & 0 & 0 & 0 \\ 6e & 0 & 0 & 0 & e & e & 6e & e & e & \\ f & 0 & 0 & 0 & 0 & 0 & f & f & f & \\ & \vdots & & & & & & \ddots & & \\ 6e & e & 0 & & & 0 & & e & e & 6e & e \\ f & f & 0 & & & & & 0 & 0 & f & f \end{array}\right] \tag{8.94}$$

For certain valences, $S_{0,n}$ is defective due to a shared eigenvalue between the T and T_{12} subblocks, and a nontrivial Jordan form must be used. It is intuitive to think of the eigenvectors of T_{12} as solutions to a difference equation as we showed in Chapter 4. The elements of these eigenvectors will be associated with edge and face averaging rule rows. There are two families of solution functions: even symmetry α's and odd symmetry β's. The functions for these elements are parameterized by the row r and column c of the eigenvector matrix.

$$c \in [1, n-1] \qquad \lambda_c = e\left(5 + \cos\left(\frac{c\pi}{n}\right) \pm \cos\left(\frac{c\pi}{2n}\right)\sqrt{2\left(9 + \cos\left(\frac{c\pi}{n}\right)\right)}\right)$$

$$\alpha_e(r,c) = \frac{\lambda_c - 4e}{4e}\cos\left(r\tfrac{c\pi}{n}\right), \quad \alpha_f(r,c) = \cos\left(\left(r-\tfrac{1}{2}\right)\tfrac{c\pi}{n}\right) + \cos\left(\left(r+\tfrac{1}{2}\right)\tfrac{c\pi}{n}\right) \tag{8.95}$$

$$\beta_e(r,c) = \frac{\lambda_c - 4e}{4e}\sin\left(r\tfrac{c\pi}{n}\right), \quad \beta_f(r,c) = \sin\left(\left(r-\tfrac{1}{2}\right)\tfrac{c\pi}{n}\right) + \sin\left(\left(r+\tfrac{1}{2}\right)\tfrac{c\pi}{n}\right) \tag{8.96}$$

When the wedge valence n is even, the eigenvectors of $V_{12,n}$ have an edge rule as their center row. The first column is the eigenvector associated with the eigenvalue $\lambda_n = \frac{1}{4}$. The other columns, which consist of α's and β's, each represent two eigenvectors corresponding to the two eigenvalues λ_c^- and λ_c^+. The positive or negative sign on the eigenvalue specifies whether the

discriminant is added or subtracted in the eigenvalue formula.

$$V_{12,n} = \begin{bmatrix} \sin\left(-\frac{n-1}{2}\pi\right) & \alpha_f\left(-\frac{n-1}{2},1\right) & \beta_f\left(-\frac{n-1}{2},2\right) & \cdots & \beta_f\left(-\frac{n-1}{2},n-1\right) \\ 0 & \alpha_e\left(-\frac{n-2}{2},1\right) & \beta_e\left(-\frac{n-2}{2},2\right) & \cdots & \beta_e\left(-\frac{n-2}{2},n-1\right) \\ \vdots & \vdots & \vdots & & \vdots \\ \sin\left(-\frac{1}{2}\pi\right) & \alpha_f\left(-\frac{1}{2},1\right) & \beta_f\left(-\frac{1}{2},2\right) & \cdots & \beta_f\left(-\frac{1}{2},n-1\right) \\ 0 & \alpha_e\left(0,1\right) & \beta_e\left(0,2\right) & \cdots & \beta_e\left(0,n-1\right) \\ \sin\left(\frac{1}{2}\pi\right) & \alpha_f\left(\frac{1}{2},1\right) & \beta_f\left(\frac{1}{2},2\right) & \cdots & \beta_f\left(\frac{1}{2},n-1\right) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \alpha_e\left(\frac{n-2}{2},1\right) & \beta_e\left(\frac{n-2}{2},2\right) & \cdots & \beta_e\left(\frac{n-2}{2},n-1\right) \\ \sin\left(\frac{n-1}{2}\pi\right) & \alpha_f\left(\frac{n-1}{2},1\right) & \beta_f\left(\frac{n-1}{2},2\right) & \cdots & \beta_f\left(\frac{n-1}{2},n-1\right) \end{bmatrix} \tag{8.97}$$

When the wedge valence n is odd, the eigenvectors of $V_{12,n}$ have a face rule as their center row. The first column is the eigenvector associated with the eigenvalue $\lambda_n = \frac{1}{4}$. The other columns, which consist of α's and β's, each represent two eigenvectors corresponding to the two eigenvalues λ_c^- and λ_c^+. The positive or negative sign on the eigenvalue specifies the discriminant is added or subtracted.

$$V_{12,n} = \begin{bmatrix} \cos\left(-\frac{n-1}{2}\pi\right) & \alpha_f\left(-\frac{n-1}{2},1\right) & \beta_f\left(-\frac{n-1}{2},2\right) & \cdots & \beta_f\left(-\frac{n-1}{2},n-1\right) \\ 0 & \alpha_e\left(-\frac{n-2}{2},1\right) & \beta_e\left(-\frac{n-2}{2},2\right) & \cdots & \beta_e\left(-\frac{n-2}{2},n-1\right) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \alpha_e\left(-\frac{1}{2},1\right) & \beta_e\left(-\frac{1}{2},2\right) & \cdots & \beta_e\left(-\frac{1}{2},n-1\right) \\ \cos\left(-\frac{1}{2}\pi\right) & \alpha_f\left(0,1\right) & \beta_f\left(0,2\right) & \cdots & \beta_f\left(0,n-1\right) \\ 0 & \alpha_e\left(\frac{1}{2},1\right) & \beta_e\left(\frac{1}{2},2\right) & \cdots & \beta_e\left(\frac{1}{2},n-1\right) \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \alpha_e\left(\frac{n-2}{2},1\right) & \beta_e\left(\frac{n-2}{2},2\right) & \cdots & \beta_e\left(\frac{n-2}{2},n-1\right) \\ \cos\left(\frac{n-1}{2}\pi\right) & \alpha_f\left(\frac{n-1}{2},1\right) & \beta_f\left(\frac{n-1}{2},2\right) & \cdots & \beta_f\left(\frac{n-1}{2},n-1\right) \end{bmatrix} \tag{8.98}$$

8.6.5 Catmull-Clark Corners

For the corner patch case, the submatrix T of $S_{0,n}$ (Equation 8.94) contains the averaging rules for the vertices $0 \rightarrow 2$ from Figure 8.3. Note that in the case of a corner vertex with one sharp edge, vertices 1 and 2 are the same point:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \tag{8.99}$$

T alone has an eigen-decomposition of the form:

$$TV = V\Lambda \tag{8.100}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & a_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & a_{\frac{n}{2}} & -b_{\frac{n}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & a_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & a_{\frac{n}{2}} & -b_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{8.101}$$

$T_{12,n}$ also has an eigenvalue of $\frac{1}{2}$ when the wedge valence n is even, so the associated eigenvectors of $S_{0,n}$ fall into odd and even valence cases.

Corner with Odd n

When n is odd, $S_{0,n}$ is diagonalizable, so there is a full set of eigenvectors.

$$\Lambda = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & \frac{1}{2} & 0 & \\ 0 & 0 & \frac{1}{2} & \\ \hline & & & \ddots \end{array}\right] \tag{8.102}$$

Equation 8.103 shows the first three eigenvectors $\{v_1, v_2, v_3\}$.

$$\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{array}\right], \left[\begin{array}{c} 0 \\ \alpha_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \alpha_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \alpha_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \alpha_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \alpha_e\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \alpha_f\left(0, \frac{n}{2}\right) \\ \alpha_e\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \alpha_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \alpha_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ \beta_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \beta_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \beta_f\left(0, \frac{n}{2}\right) \\ \beta_e\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \beta_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right] \tag{8.103}$$

Corner with Even n

When n is even, $S_{0,n}$ is defective due to the eigenvalue $\lambda = \frac{1}{2}$ which has algebraic multiplicity three but only two linearly independent eigenvectors.

$$J = \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & \\ 0 & \frac{1}{2} & 0 & 0 & \\ 0 & 0 & \frac{1}{2} & 0 & \\ \hline 0 & 0 & 1 & \frac{1}{2} & \\ & & & & \ddots \end{array}\right] \tag{8.104}$$

When n is divisible by 4, the sine based eigenvector is missing. Equation 8.105 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$.

$$\begin{aligned} \gamma_e(r) &= -3(2t + 1 - \cos[t\pi])\cos\left[t\frac{\pi}{2}\right] \\ \gamma_f(r) &= \gamma_e\left(r - \frac{1}{2}\right) + \gamma_e\left(r + \frac{1}{2}\right) \end{aligned}$$

$$\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{array}\right], \left[\begin{array}{c} 0 \\ \alpha_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \alpha_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \alpha_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \alpha_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \alpha_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \alpha_e\left(0, \frac{n}{2}\right) \\ \alpha_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \alpha_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \alpha_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ \gamma_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \gamma_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(0, \frac{n}{2}\right) \\ \gamma_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \gamma_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \beta_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \beta_e\left(0, \frac{n}{2}\right) \\ \beta_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \beta_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right] \tag{8.105}$$

When n is only divisible by 2 and not 4, the cosine based eigenvector is missing. Equation 8.106 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$.

$$\gamma_e(r) = 3(2t + 1 + \cos[t\pi]) \sin\left[t\frac{\pi}{2}\right]$$
$$\gamma_f(r) = \gamma_e\left(r - \frac{1}{2}\right) + \gamma_e\left(r + \frac{1}{2}\right)$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ \beta_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \beta_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \beta_e\left(0, \frac{n}{2}\right) \\ \beta_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \beta_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{bmatrix}, \begin{bmatrix} 0 \\ \gamma_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \gamma_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(0, \frac{n}{2}\right) \\ \gamma_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \gamma_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline \alpha_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \alpha_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \alpha_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \alpha_e\left(0, \frac{n}{2}\right) \\ \alpha_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \alpha_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \alpha_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{bmatrix} \tag{8.106}$$

8.6.6 Catmull-Clark Creases

For the crease patch case, the submatrix T of $S_{0,n}$ (Equation 8.94) contains the averaging rules for the vertices $0 \rightarrow 2$ from Figure 8.3:

$$T = \begin{bmatrix} \frac{6}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \tag{8.107}$$

T alone has an eigen-decomposition of the form:

$$TV = V\Lambda \tag{8.108}$$

$$\begin{bmatrix} \frac{6}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 & -\frac{1}{2}b_{\frac{n}{2}} \\ 1 & a_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & -a_{\frac{n}{2}} & b_{\frac{n}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\frac{1}{2}b_{\frac{n}{2}} \\ 1 & a_{\frac{n}{2}} & b_{\frac{n}{2}} \\ 1 & -a_{\frac{n}{2}} & b_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{bmatrix} \tag{8.109}$$

$T_{12,n}$ always has an eigenvalue of $\frac{1}{4}$ and it has an eigenvalue of $\frac{1}{2}$ when the wedge valence n is even. The associated eigenvectors of $S_{0,n}$ fall into odd, divisible by four, and even valence cases.

Catmull-Clark Crease with $Mod\,(n,4)=2$

When $Mod\,(n,4)=2$, n is even but not divisible by 4, and $S_{0,n}$ is diagonalizable.

$$\Lambda = \left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & \frac{1}{2} & 0 & \\ 0 & 0 & \frac{1}{4} & \\ \hline & & & \ddots \end{array}\right] \tag{8.110}$$

Equation 8.111 shows the first three eigenvectors $\{v_1, v_2, v_3\}$. Note that if $Mod\,(n,4)=2$ then $\frac{n}{2}$ is odd, so $\cos\left(\frac{n}{2}\pi\right) = -1$. Also note that $\cos\left(t\pi\right) = -\cos\left(\left(t+1\right)\pi\right)$.

$$\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{array}\right], \left[\begin{array}{c} 0 \\ \beta_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \beta_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \beta_e\left(0, \frac{n}{2}\right) \\ \beta_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \beta_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} -\frac{1}{2}\left(\frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(\frac{n}{2}\pi\right)\right) \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(\frac{n}{2}\pi\right) \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(-\frac{n}{2}\pi\right) \\ \hline \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(-\frac{n-2}{2}\pi\right) \\ \vdots \\ \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(0\pi\right) \\ \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \\ \vdots \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(\frac{n-2}{2}\pi\right) \\ \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \end{array}\right] \tag{8.111}$$

Crease with n Divisible by Four

When n is divisible by four, $S_{0,n}$ is defective due to the eigenvalue $\lambda = \frac{1}{2}$ which has algebraic multiplicity two but only one linearly independent eigenvector.

$$J = \left[\begin{array}{ccc|cc} 1 & 0 & 0 & 0 & \\ 0 & \frac{1}{4} & 0 & 0 & \\ 0 & 0 & \frac{1}{2} & 0 & \\ \hline 0 & 0 & 1 & \frac{1}{2} & \\ & & & & \ddots \end{array}\right] \tag{8.112}$$

Equation 8.113 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$ where $i \in \left[1, \frac{n}{2} - 1\right]$.

$$\gamma_e(r) = -3\left(2t - 1 + \cos[t\pi]\right)\cos\left[t\frac{\pi}{2}\right]$$
$$\gamma_f(r) = \gamma_e\left(r - \frac{1}{2}\right) + \gamma_e\left(r + \frac{1}{2}\right)$$

$$\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{array}\right], \left[\begin{array}{c} -\frac{1}{2}\left(\frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(\frac{n}{2}\pi\right)\right) \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(\frac{n}{2}\pi\right) \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(-\frac{n}{2}\pi\right) \\ \hline \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(-\frac{n-2}{2}\pi\right) \\ \vdots \\ \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos(0\pi) \\ \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \\ \vdots \\ \frac{1}{3}\cos\left(\frac{n}{2}\pi\right) + \cos\left(\frac{n-2}{2}\pi\right) \\ \frac{4}{3}\cos\left(\frac{n}{2}\pi\right) + 0 \end{array}\right], \left[\begin{array}{c} 0 \\ \gamma_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \gamma_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(0, \frac{n}{2}\right) \\ \gamma_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \gamma_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \beta_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_f\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \beta_e\left(0, \frac{n}{2}\right) \\ \beta_f\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \beta_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right] \tag{8.113}$$

Catmull-Clark Crease with Odd n

When n is odd the eigenvalue $\lambda = \frac{1}{4}$ is defective having algebraic multiplicity two but only one linearly independent eigenvector.

$$J = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ \hline 0 & 0 & 1 & \frac{1}{4} \\ & & & & \ddots \end{array}\right] \tag{8.114}$$

Equation 8.115 shows the first four generalized eigenvectors $\{v_1, v_2, v_3, v_4\}$.

$$\begin{aligned}
&\gamma_e(r) = (4r + 1 + \cos[r2\pi])\sin[r\pi] \\
&\gamma_f(r) = 0 \\
&\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{array}\right], \left[\begin{array}{c} -\frac{1}{2}\beta_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \beta_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \beta_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \beta_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \beta_f\left(0, \frac{n}{2}\right) \\ \beta_e\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \beta_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \beta_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ \gamma_e\left(\frac{n}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n}{2}, \frac{n}{2}\right) \\ \hline \gamma_f\left(-\frac{n-1}{2}, \frac{n}{2}\right) \\ \gamma_e\left(-\frac{n-2}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_e\left(-\frac{1}{2}, \frac{n}{2}\right) \\ \gamma_f\left(0, \frac{n}{2}\right) \\ \gamma_e\left(\frac{1}{2}, \frac{n}{2}\right) \\ \vdots \\ \gamma_e\left(\frac{n-2}{2}, \frac{n}{2}\right) \\ \gamma_f\left(\frac{n-1}{2}, \frac{n}{2}\right) \end{array}\right], \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline \cos\left(-\frac{n-1}{2}\pi\right) \\ 0 \\ \vdots \\ 0 \\ \cos(0\pi) \\ 0 \\ \vdots \\ 0 \\ \cos\left(\frac{n-1}{2}\pi\right) \end{array}\right]
\end{aligned} \tag{8.115}$$

8.6.7 Catmull-Clark Corner and Crease Picking Matrices

The picking matrices P_k where $k \in \{U, V, UV\}$ select out the 16 control points for the corresponding uniform bi-cubic B-spline patch from the $2n + 17$ control points generated by the $\bar{A}$ matrix, see Figure 8.3. P_k is a $16 \times 2n + 17$ matrix where each row has a single 1 in the column corresponding to control point listed in the sets below.

The following are the picking matrices for subpatches $1 < p < n$ (note that if $p = n - 1$ then V_{2p+4} must be replaced by V_1):

$$\begin{aligned}
P_{U,n,p} = &\{V_{2p-2}, V_{2p-1}, V_{2n+2}, V_{2n+9}, V_0, V_{2p}, V_{2n+3}, V_{2n+10}, \\
&V_{2p+2}, V_{2p+1}, V_{2n+4}, V_{2n+11}, V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+12}\} \qquad (8.116) \\
P_{V,n,p} = &\{V_{2p+4}, V_0, V_{2p}, V_{2n+3}, V_{2p+3}, V_{2p+2}, V_{2p+1}, V_{2n+4}, \\
&V_{2n+8}, V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+17}, V_{2n+16}, V_{2n+15}, V_{2n+14}\} \qquad (8.117) \\
P_{UV,n,p} = &\{V_0, V_{2p}, V_{2n+3}, V_{2n+10}, V_{2p+2}, V_{2p+1}, V_{2n+4}, V_{2n+11}, \\
&V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+12}, V_{2n+16}, V_{2n+15}, V_{2n+14}, V_{2n+13}\} \qquad (8.118)
\end{aligned}$$

The following are the picking matrices for subpatch $p = 1$ (note that if $p = n - 1$ then

V_{2p+4} must be replaced with V_1):

$$\bar{V}_a = 2V_0 - V_{2p+2} \tag{8.119}$$

$$\bar{V}_b = 2V_{2p} - V_{2p+1} \tag{8.120}$$

$$\bar{V}_c = 2V_{2n+2} - V_{2n+3} \tag{8.121}$$

$$\bar{V}_d = 2V_{2n+8} - V_{2n+9} \tag{8.122}$$

$$\begin{aligned} P_{U,n,1} = \{&\bar{V}_a, \bar{V}_b, \bar{V}_c, \bar{V}_d, V_0, V_{2p}, V_{2n+2}, V_{2n+8}, \\ &V_{2p+2}, V_{2p+1}, V_{2n+3}, V_{2n+9}, V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+10}\} \end{aligned} \tag{8.123}$$

$$\begin{aligned} P_{V,n,1} = \{&V_{2p+4}, V_0, V_{2p}, V_{2n+2}, V_{2p+3}, V_{2p+2}, V_{2p+1}, V_{2n+3}, \\ &V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+15}, V_{2n+14}, V_{2n+13}, V_{2n+12}\} \end{aligned} \tag{8.124}$$

$$\begin{aligned} P_{UV,n,1} = \{&V_0, V_{2p}, V_{2n+2}, V_{2n+8}, V_{2p+2}, V_{2p+1}, V_{2n+3}, V_{2n+9}, \\ &V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+10}, V_{2n+14}, V_{2n+13}, V_{2n+12}, V_{2n+11}\} \end{aligned} \tag{8.125}$$

The following are the picking matrices for subpatch $p = n$:

$$\bar{V}_e = 2V_0 - V_{2p} \tag{8.126}$$

$$\bar{V}_f = 2V_1 - V_{2p+1} \tag{8.127}$$

$$\bar{V}_g = 2V_{2n+7} - V_{2n+6} \tag{8.128}$$

$$\bar{V}_h = 2V_{2n+15} - V_{2n+14} \tag{8.129}$$

$$\begin{aligned} P_{U,n,n} = \{&V_{2p-2}, V_{2p-1}, V_{2n+2}, V_{2n+8}, V_0, V_{2p}, V_{2n+3}, V_{2n+9}, \\ &V_1, V_{2p+1}, V_{2n+4}, V_{2n+10}, V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+11}\} \end{aligned} \tag{8.130}$$

$$\begin{aligned} P_{V,n,n} = \{&\bar{V}_e, V_0, V_{2p}, V_{2n+3}, \bar{V}_f, V_1, V_{2p+1}, V_{2n+4}, \\ &\bar{V}_g, V_{2n+7}, V_{2n+6}, V_{2n+5}, \bar{V}_h, V_{2n+15}, V_{2n+14}, V_{2n+13}\} \end{aligned} \tag{8.131}$$

$$\begin{aligned} P_{UV,n,n} = \{&V_0, V_{2p}, V_{2n+3}, V_{2n+9}, V_1, V_{2p+1}, V_{2n+4}, V_{2n+10}, \\ &V_{2n+7}, V_{2n+6}, V_{2n+5}, V_{2n+11}, V_{2n+15}, V_{2n+14}, V_{2n+13}, V_{2n+12}\} \end{aligned} \tag{8.132}$$

For a wedge with a single patch $n = 1$, the picking matrices are:

$$\bar{V}_a = 2V_0 - V_1 \tag{8.133}$$

$$\bar{V}_b = 2V_{2p} - V_{2p+1} \tag{8.134}$$

$$\bar{V}_c = 2V_{2n+2} - V_{2n+3} \tag{8.135}$$

$$\bar{V}_d = 2V_{2n+7} - V_{2n+8} \tag{8.136}$$

$$\bar{V}_e = 2V_0 - V_{2p} \tag{8.137}$$

$$\bar{V}_f = 2V_1 - V_{2p+1} \tag{8.138}$$

$$\bar{V}_g = 2V_{2n+6} - V_{2n+5} \tag{8.139}$$

$$\bar{V}_h = 2V_{2n+13} - V_{2n+12} \tag{8.140}$$

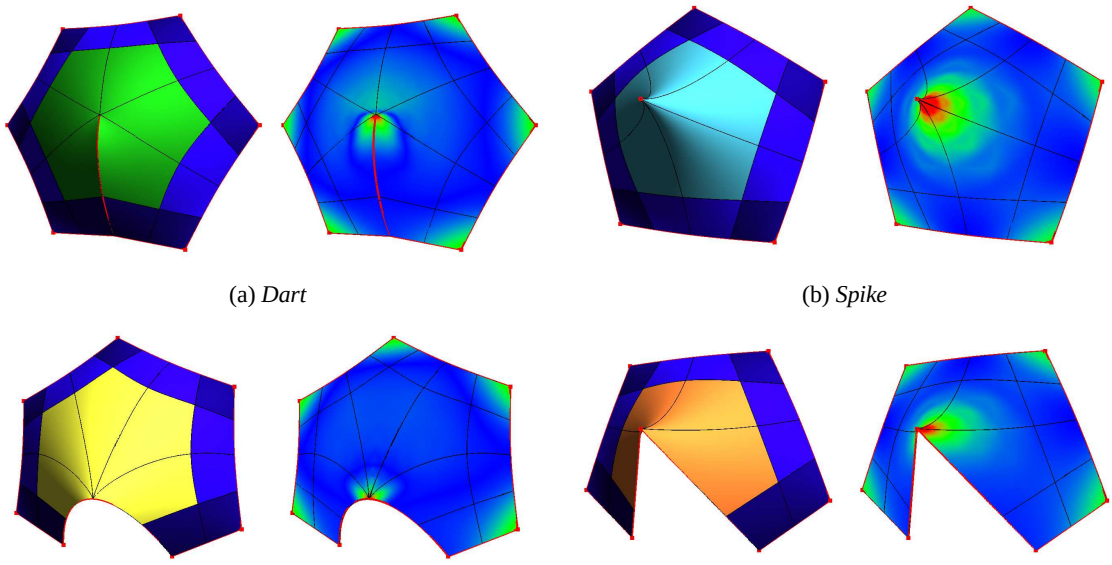
$$\begin{aligned} P_{U,1,1} = \{&\bar{V}_a, \bar{V}_b, \bar{V}_c, \bar{V}_d, V_0, V_{2p}, V_{2n+2}, V_{2n+7}, \\ &V_1, V_{2p+1}, V_{2n+3}, V_{2n+8}, V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+9}\} \end{aligned} \tag{8.141}$$

$$\begin{aligned} P_{V,1,1} = \{&\bar{V}_e, V_0, V_{2p}, V_{2n+2}, \bar{V}_f, V_1, V_{2p+1}, V_{2n+3}, \\ &\bar{V}_g, V_{2n+6}, V_{2n+5}, V_{2n+4}, \bar{V}_h, V_{2n+13}, V_{2n+12}, V_{2n+11}\} \end{aligned} \tag{8.142}$$

$$\begin{aligned} P_{UV,1,1} = \{&V_0, V_{2p}, V_{2n+2}, V_{2n+7}, V_1, V_{2p+1}, V_{2n+3}, V_{2n+8}, \\ &V_{2n+6}, V_{2n+5}, V_{2n+4}, V_{2n+9}, V_{2n+13}, V_{2n+12}, V_{2n+11}, V_{2n+10}\} \end{aligned} \tag{8.143}$$

8.7 Conclusion

We have implemented a new exact evaluation algorithm for piecewise smooth Catmull-Clark subdivision surfaces. We have developed new techniques for dart, crease, corner, and spike patches. Figure 8.6 shows examples of the 1-ring neighborhoods around vertices of these four types. The right hand image of each of these pairs shows the absolute magnitude Gaussian curvature map which was calculated by using our algorithm to evaluate the derivatives of the surface.



(a) *Dart* (b) *Spike*

(c) *Crease* (d) *Corner*

Figure 8.6: *1-ring vertex wedge for each of the four different sharp extraordinary vertex types for piecewise smooth Catmull-Clark surfaces. The left image of each pair is a shaded rendering of the surface with a color coding for patch type. The right image is a map of absolute Gaussian curvature. The surface position and derivatives were evaluated using our technique. Sharp edges and vertices are drawn in red.*

Chapter 9

Subdivision Implementation

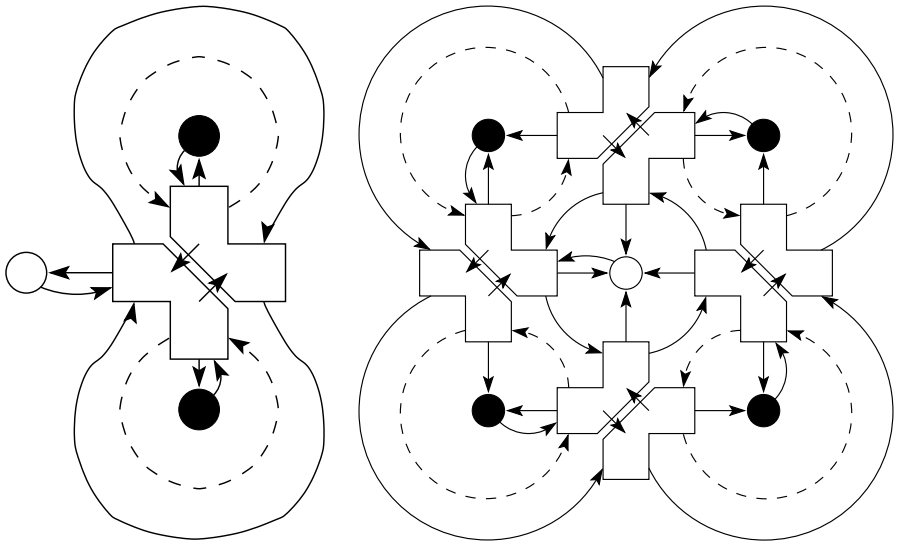
9.1 Introduction

The data structure necessary for implementing a subdivision surface algorithm is a 2-manifold mesh structure. To perform the averaging rules, it is necessary to quickly access all vertices in the 1-ring neighborhood of vertices, edges, and faces. This chapter describes the full connected mesh structure used in our implementation and a number of operators on the mesh to perform uniform subdivision, adaptive subdivision, and exact evaluation of the limit surface.

9.2 Half-Edge Data Structure

Our implementation uses a 2-manifold half-edge data structure which was inspired by the quad-edge data structure [9]. There are three data types that make up a half-edge mesh: half-edge, vertex, and facet (Figure 9.1). Each of these types is of a fixed sized despite the fact that there can be a variable number of edges around each vertex or facet. In our half-edge structure, every edge is represented by a pair of half-edges which each point in opposite directions along the edge. Each half-edge has a data pointer to a facet and another to its vertex of origin in a counterclockwise ordering around the facet. Each vertex and facet points to any one half-edge in the counterclockwise ring around them. Each half-edge has a pointer to its partner half-edge, a next pointer around its facet, and a next pointer around its vertex. Half-edges are always allocated in pairs, and their topological half-edge pointers are initialized as shown in Figure 9.1(a). These topological pointers make it possible to start at any half-edge and traverse to any other entity in the mesh. It is also possible to start from a vertex or facet, but there is an ambiguity as to which half-edge is the first

in its ring that must be tested for first. The similarity of representation for vertices and facets in the half-edge structure emphasizes the duality between these two entities in a 2-manifold.



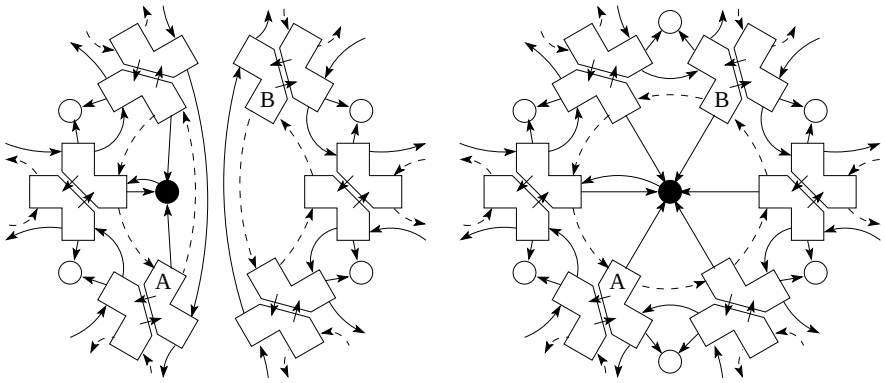
(a) *Single pair of half-edges* (b) *Square facet*

Figure 9.1: *The left figure shows the representation of a single edge. The elbow shapes are the half-edges. The dark circles are vertices. The empty circles are facets. A half-edge has a data pointer to a facet and another to its vertex of origin in a counterclockwise ordering around the facet. Each vertex and facet points to a single half-edge in the ring around it. Each half-edge has a pointer to its partner half-edge, a solid next pointer around its facet, and a dashed next pointer around its vertex. The right figure shows the representation of a square.*

9.2.1 Splice Operator

The fundamental topological editing operation on a half-edge mesh is the splice operation, which is identical to the splice operation introduced with the quad-edge mesh. The splice operation is the 2-manifold analog of both inserting into and removing from a doubly linked list (Figure 9.2). The operands to splice are two half-edges A and B. Splice swaps four next pointers, the two vertex next pointers of A and B, and the two facet next pointers of the half-edges pointed to by the vertex next pointers of A and B. The symmetry of the swaps shows that the ordering of the operands A and B is unimportant. The effect of these swaps is different based on whether A and B start off in the same vertex ring or not. If they are not in the same vertex ring, then splice has the effect of merging the two rings into a single ring, i.e. going from Figure 9.2(a) to 9.2(b). The vertex

ring with A is split between A and the next half-edge around the vertex, and the same is true of the vertex ring with B. If A and B are in the same ring, then splice has the exact opposite effect of splitting the ring just after A and B into two separate rings. Splice is the only operation necessary for modifying the edge connectivity of a mesh, but it is necessary to be careful about updating all pointers to and from vertices and facets to match any changes. A single pair of half-edges on their own are always configured to have independent vertex rings of size one at each end. Hence adding or removing a single half-edge from a vertex ring reduces to performing a splice on that half-edge with the half-edge previous to it in the vertex ring.



(a) *Two separate vertex rings* (b) *Spliced into one vertex ring*

Figure 9.2: *The splice operator when called on half-edges A and B, in either order, updates the four middle half-edge pointers to connect the two vertex rings, and when called again on the same A and B it will undo the operation to split the vertex ring into two.*

9.2.2 Boundary Edges

Most boundary representation file formats give the topology of the mesh as a list of facets which in turn each have a counterclockwise ordered list of vertices. The half-edge mesh is then built by adding a facet at a time and splicing its edges together with the current mesh. This is accomplished by comparing the edges of the new facet to see if they match up with any edges in the current mesh. Because the mesh can only represent 2-manifolds, each edge is used by at most two facets, so the internal edges need not be considered when searching to attach a new facet. A boundary edge has one half-edge pointing to a facet and the other half-edge pointing to the null facet. These boundary edges are the only ones that need to be considered while adding a new facet, so we store a hash table of boundary edges indexed by the ordered pair of end points for fast access.

As soon as a facet is created that matches one of the boundary edges, the facet is correctly connected there to the rest of the mesh, and the edge is removed from the hash table. The boundary edge hash table is also helpful for merging two meshes together that share a seam of edges. In this case, the matching process can be accelerated after an initial edge pair has been matched using the half-edge mesh structure to walk along the seam in both meshes.

9.2.3 Vertex Data

The data associated per vertex may be different for different applications. For most applications, a position vector is stored. For subdivision surfaces, a limit position vector is also stored for rendering purposes. In general, the vertex position and limit position need not just be a 3D vector. The number of elements in the vector to be averaged by the subdivision rules can be specified at run-time. Examples of this type of *vertex data* are the vertex positions, color values, or possibly normals. The user can also specify a sharpness value which ranges from 0 for smooth to ∞ for infinitely sharp or any value in between for semi-sharp. A derived enumerated vertex type which is a function of the vertex sharpness and the number of sharp edges incident to the vertex helps to simplify the implementation of the subdivision algorithms. The vertex types are: *smooth, dart, crease, corner, spike, semi-dart, semi-crease, semi-corner, semi-spike,* or *smooth boundary*. Smooth and dart vertices follow the same averaging rules. Corner and spike vertices both remain stationary. All the semi-sharp vertex types have the possibility of being a blend between the smooth and sharp averaging rules if the sharpness value is between 0 and 1. A smooth boundary vertex does not have a full 1-ring neighborhood, so it cannot be subdivided. The final data field is the derived valence of the vertex which is only useful in the smooth and spike cases. In the vertex structure, as well as the other entity structures, bit fields are used to pack enumerated type variables and counters, like the valence, into a single integer variable to save memory.

9.2.4 Facet Data

For most applications, the valence, the centroid position, and the best fit normal are stored per facet. The centroid and normal are used for rendering. The valence is the number of edges or vertices. For subdivision surfaces, it is also useful to store a derived enumerated patch type. The different patch types are: *regular B-spline, smooth extraordinary, dart, crease, corner, spike, incomplete,* or *non-patch*. A regular B-spline patch can be directly evaluated with the uniform bi-cubic B-spline polynomials. The smooth extraordinary, dart, crease, corner, and spike cases

all have the proper 1-ring neighborhoods to be evaluated using our Jordan decomposition method. An incomplete patch does not have the proper 1-ring neighborhood to do evaluation on, so some adaptive subdivision must be performed before it can be evaluated. A non-patch is similar to a smooth boundary vertex. It is either a facet that was tagged by the user as a hole, or a facet next to a smooth boundary edge. In either case, the facet will not produce a limit surface patch, so further subdivision is not require by this facet. A non-patch facet may have more subdivision performed on it to support a neighboring incomplete patch. The final piece of data associated with a facet is a possible *patch evaluator* object. If a patch has a suitable 1-ring neighborhood, a patch evaluator is allocated and pointed to. A patch evaluator copies and eigen-projects all of the data necessary to evaluate the type of patch, so that evaluation queries to the patch at run-time are streamlined.

9.2.5 Edge Data

Some data is associated once per edge instead of per half-edge, so each pair of half-edges points to an edge and the edge points back to one of the half-edges. For subdivision surfaces, the only data that is assigned at the edge is the sharpness value which is identical to the one described in Section 9.2.3 for vertices.

9.2.6 Half-Edge Data

A half-edge is the relationship between a facet and one of its vertices, so it is the perfect place to assign per facet per vertex data. One example of this type of data is the vertex normal. Spike, crease, and corner vertices do not have a unique normal. For many of these types, there is one normal per wedge of facets around the vertex, but for the sake of simplicity, we redundantly store one normal per half-edge for rendering. The other type of data that is stored per half-edge is grouped into the category of *parameter data*. Examples of parameter data are texture map coordinates or bump map coordinates. For the discussion of the subdivision process, we will first address vertex data alone, and then in Section 9.7 we will describe how to generalize the subdivision algorithms to include the parameter data as well.

9.2.7 Parent and Child Pointers

A uniform subdivision step takes a parent mesh and performs a topological split to all of its facets generating a child mesh, which in the case of Loop and Catmull-Clark subdivision is four times larger. One possible approach would be to modify the parent mesh by splitting all edges

and adding new vertices and facets, but there are a number of reasons that this is not the ideal way to implement subdivision. The first reason corresponds to the averaging phase of a subdivision step. The vertex positions in the child generation are derived from the parent vertex positions. The difficulty with computing these in place are that conceptually all of the vertex positions must be calculated in parallel from the parent positions, but we are implementing our algorithm on a sequential machine. One solution would be to keep copies of the parent positions, but this would double the data size of our representation.

A solution to this problem is to keep around the parent mesh which is a factor of four smaller than the child mesh. After the subdivision averaging is finished, the parent mesh could be deleted, but it is not very costly to keep all generations of subdivision mesh around. Each subdivision step splits each facet into four, so the history of subdivisions is effectively a tree structure where each node (facet) has four children. The sum of all nodes in such a tree is less than double the number of leaf nodes, i.e. the newest generation.

There are a number of advantages to retaining all of the generations of subdivision. We have already described how it aids in calculating the averaging rules. It also helps with the topological split operation. A naive approach would be to split each facet into four and use our boundary edge hash table to merge the facets together in the child generation. This approach is extremely inefficient because the parent mesh contains all of the connectivity information that is necessary. The parent mesh can be thought of as a sparse scaffold that the child mesh is hung off during the topological construction. Parent and child pointers are added to the data structures for half edges, vertices, and facets. When an edge is split at its midpoint, the child pointers of the parent half-edges are assigned to corresponding half-edges in the child mesh, and those child half-edges have their parent half-edges set to point back into the parent mesh. Edges internal to parent facets have their parent pointers set to null. Vertices point to corresponding child vertices in the child mesh and vice versa, while edge midpoint and facet centroid vertices have their parent pointers set to null. All facets in the child mesh have their parent pointers set to the facet of which they are a subpatch. Facet child pointers are different for Loop vs. Catmull-Clark subdivision because of the different topological split patterns of their facets. In Loop subdivision, the facet child pointer is assigned to the central triangle of the four. In the Catmull-Clark case, the facet child pointer is assigned to the child vertex corresponding to the facet centroid. With these parent and child pointers assigned as the child mesh is being constructed, the parent mesh can be used very efficiently in lieu of the hash table of boundary edges to find preexisting edges in the child mesh to link to.

The tree of generations created by the parent and child pointers is also useful for adaptive

subdivision, patch evaluation, and editing of control vertex positions. For adaptive subdivision, the coarse control mesh can be augmented with finer scale facets in local areas by locally subdividing there and hanging the partial child meshes off of their parent meshes. This adaptive subdivision tree can be used to build a minimal set of patch evaluators. As early in terms of generations as a facet has the correct 1-ring neighborhood, it can be converted to a patch evaluator. This might occur in different generations for different patches based on the spacing of extraordinary vertices and the existence of semi-sharp edge and vertex tags. Once the adaptive tree of patch evaluators is built, the (u, v) parameters given to the facet at the control mesh level direct the traversal of nodes, i.e. facets, of the tree to arrive at the corresponding patch evaluator. During the traversal, the parameters are rotated and scaled to align with local $(\bar{u}, \bar{v})$ parameters of the leaf patch evaluator. The final operation that benefits from this tree structure is editing of the control mesh vertex positions. The naive approach would make it necessary to repeat all of the subdivision work whenever a value was changed, but with the child pointers, none of the topology needs to be changed and only the pyramid of child vertices whose positions are derived from that control vertex need to be updated.

9.2.8 Indexed Arrays vs. Pointers

In our discussion of the topological structure of the half-edge mesh, we have described references to half-edges, vertices, and facets as pointers, but there are actually many reasons to use array indices instead of C pointers. The half-edge mesh contains sets of arrays proportional in length to the number of vertices, facets, and half edges. A vertex, for instance, is assigned to an array index, and data for the vertex may be spread over multiple arrays but always at the same index location. This provides a simple structure for subclassing different types of mesh structures in a C++ class hierarchy. The base mesh type only contains topological information, and a subdivision mesh subclass extends the vertex, facet, and half edge definitions to have subdivision data.

Edges and half-edges can be implemented efficiently using the indexing scheme as well. Every edge is associated with a pair of half-edges. When an edge is allocated, we allocate two consecutive half-edges. This eliminates the need for the partner half-edge pointers, because the partner half-edge index can be calculated from the other index by performing a bitwise exclusive-or with 1. The half-edge to edge pointers can also be eliminated. The number of edge entities is exactly half the number of half-edges, so we allocate edges and half edges such that shifting the half-edge index to the right by one generates the edge index. Conversely, the even half-edge index can be calculated by shifting the edge index to the left by one. The sizes of the edge and half-edge arrays

are interdependent and controlled by the same variable.

Array based allocation has the problem of choosing the correct size, which may not be known at the mesh creation time. When more vertices are added to the mesh, eventually the array size will be met and the array will have to be reallocated. Fortunately, the relative nature of array indexing makes resizing very easy and efficient. An array of twice the size is allocated, and the previous array is block-copied into the new array. All of the references to vertices are still valid, they are just relative to a new array base pointer. This is not the case if C pointers are used, where a deep copy that forwards all references would have to be used. This same property makes it very easy and fast to make a copy of a the mesh. Another mesh of the same size is allocated and all of the arrays of data are block copied into the new mesh.

Subdivision algorithms tend to only grow the size of the mesh without ever deleting vertices, facets or edges, but in general it is useful to be able to delete entities from a mesh. We allow for the deletion of an entity at any index in the array. One approach would be to shift the other entities in the array to keep all of the allocated entities contiguous, but this is very costly and difficult because all index references would have to be updated for every deletion. Instead, we simply maintain a free-list of entities in the vertex, facet, and edge arrays. These free-lists are implemented by adding a next pointer within the data structure for each vertex, facet, and edge. Note that the next pointer is only added per edge and not per half-edge to save memory. The indices of half-edges and edges are correlated, so managing a list for edges effectively does the same for half-edges. These free-lists are implemented in a very clean way using the concept of a *sentinel node*. The 0 index of all arrays is a reserved sentinel node. This has a number of nice properties. First, the concept of a null pointer is then represented by the 0 index. Some extra liberties can be taken with these null indices that cannot with the C null pointer. These null indices can be safely written to because there is memory allocated to them, but the program should never rely on reading from them. One example of this is reflexively assigning the half-edge pointer of a vertex or facet when a half-edge sets its vertex or facet pointer, in order to maintain the property that a vertex or facet points to a half-edge in its ring. The sentinel node makes it safe to do this assignment without having to check if the vertex or facet index is null.

With these sentinel nodes and free-list next pointers in place, the free-list is managed by a simple singly linked list. Operations on the free-list are fast because we only need to access the head element of the list. A new array is initialized by setting all next pointers to the subsequent array index, with the 0 index sentinel node as the head of the list. An entity allocation pops the first entity pointed to by the sentinel out of the free-list. When an entity is deallocated, it is pushed

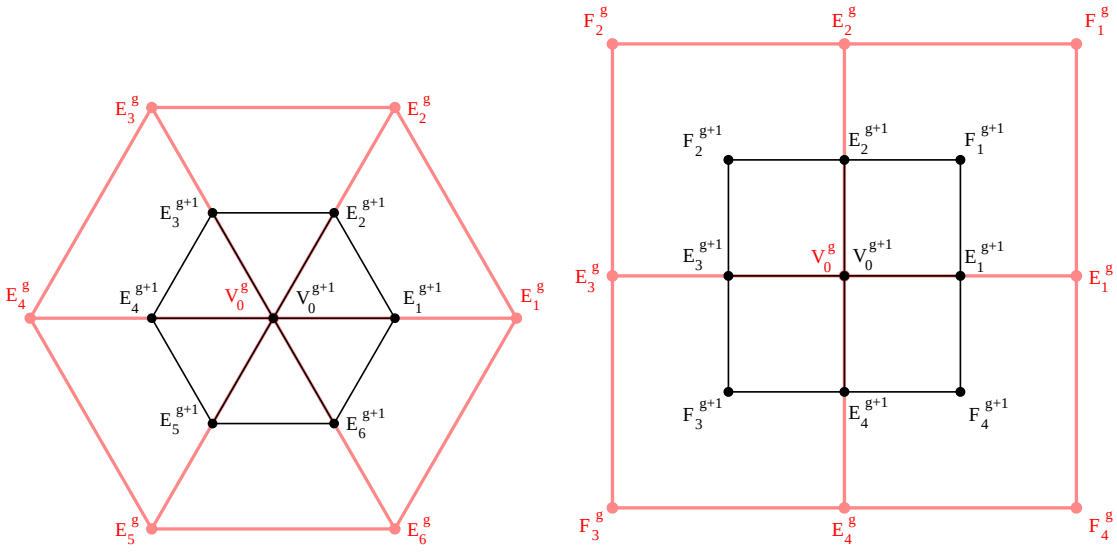
onto the front of the free-list, so that if another entity is allocated it will reuse that array index. This heuristic is used to try to prevent fragmentation of the array.

Other routines can be written to coalesce the arrays when a mesh is copied, but we have not needed this functionality with our subdivision routines. The only major differnce from this array index based interface to using C pointers is that the mesh pointer must be passed with each entity index access. This emphasizes the fact that vertices, facets, and edges belong to a given mesh and are not self sufficient objects. For child and parent pointers, the mesh must contain child and parent mesh pointers, and all of the child and parent entity indices are relative to these two meshes.

9.3 Subdivide-Vertex Operator

Once the basic half-edge mesh is in place, we can begin to build operators for these meshes. The main goal is to implement a subdivision algorithm for this mesh type. There are two ways to think about the topological split of a subdivision step, a facet-centered way and a vertex-centered way. The easier method to describe conceptually is the facet centered way. In the case of Loop subdivision, each triangle is divided into four subtriangles by constructing edges between the three pairings of edge midpoints. For Catmull-Clark subdivision, each facet is divided into quadrilaterals by connecting an edge from the facet centroid to each of the edge midpoints. The limitation of this topological operator is that the application of it to a single triangle, in the Loop case, does not create any regions in the partial child mesh that can be further subdivided. To calculate the position of a vertex in the following generation, the complete 1-ring neighborhood of the vertex must be present in the same generation. All of the vertices created by splitting a single triangle lie on the triangle border, and thus do not have the proper 1-ring neighborhood.

The alternate approach is a vertex-centered topological operator. We will describe how variants of this operator can be implemented for the Loop and Catmull-Clark subdivision schemes (Figure 9.3). The operation of locally subdividing around a vertex generates the vertices, facets, and edges which make up the 1-ring neighborhood of the image vertex in the child mesh. It is then possible to apply the same subdivide-vertex operator on the child vertex without any other subdivision to the parent mesh. This makes the subdivide-vertex operator the ideal atomic operation for building an adaptive subdivision scheme. It should not be surprising that generating the 1-ring neighborhood of the child vertex allows for further subdivision. This notion is the key principle in the matrix analysis of stationary subdivision schemes. When neighboring vertices in the parent mesh are operated on, they must check which neighbor vertices have been previously subdivided, so



(a) *Loop subdivide-vertex* (b) *Catmull-Clark subdivide-vertex*

Figure 9.3: *The subdivide-vertex operator is dependent on the 1-ring neighborhood of a vertex and generates the complete 1-ring neighborhood for the child vertex in the next generation.*

that they properly share edge midpoints and connecting edges in the child mesh. Once these issues are accounted for, the vertices of a mesh can be operated on in any random order and will always produce the proper child generation subdivision mesh. This operator also works in the presence of sharp tagging.

For Loop subdivision, the 1-ring neighborhood of a vertex is a fan of triangles (Figure 9.3(a)). The child vertex is generated. For each half-edge radiating out of the vertex, a vertex for the edge midpoint, an edge to connect the child vertex to the midpoint, another edge to cut off the triangular corner, and a facet to represent that triangle are allocated. The child pointer of the half-edge is assigned to the corresponding half-edge in the child mesh. Before the midpoint vertex is allocated, the partner half-edge's child pointer is checked. If that neighboring vertex has already been processed, then the edge midpoint already exists in the child mesh, so the new edges are spliced into midpoint vertex's ring. If this is the final vertex of the parent triangle to be subdivided, then a facet for the central triangle is allocated and all of the child edges are assigned to it.

For Catmull-Clark subdivision, the 1-ring neighborhood of a vertex is a fan of quadrilaterals (Figure 9.3(b)). The child vertex is generated. For each half-edge radiating out of the vertex, a T-shaped arrangement of three edges and two vertices is generated. A vertex for the edge midpoint is generated, and an edge from the child vertex to that midpoint. A half-edge of this edge is pointed

to by the child pointer of the parent half-edge. To the right of the edge, a perpendicular edge is constructed which will link up to the facet centroid. To the left, a similar edge, a vertex for the facet centroid, and a facet for the corner quadrilateral are constructed. If the neighbor vertex has already been subdivided, then the T-bar is already present so only the edge that connects the child vertex to the edge midpoint is constructed. When generating or connecting the facet centroid, it is necessary to search around the parent facet to check if any of its other vertices have been subdivided.

This subdivide-vertex operator is also responsible for updating the sharp and semi-sharp edge and vertex tags. This includes updating the enumerated types for vertex and patch types. Most of these flags can be trivially copied from the parent mesh, but in areas near semi-sharp tags these flags will change at some point in the subdivision hierarchy.

9.4 Uniform Subdivision

Once the subdivide-vertex operator has been implemented, performing a full uniform subdivision is quite simple. A child mesh is allocated. Then the subdivide-vertex operator is applied to each of the parent vertices in any order. The result is a child mesh that is the next generation of subdivision.

9.5 Adaptive Subdivision

We have seen that the subdivide-vertex operator is the means of performing adaptive subdivision, but it still remains to decide which portions of the mesh to adaptively subdivide. The two applications that we have implemented are adaptive tessellation for reducing discretization error when outputting a triangle model for layered manufacturing and adaptive subdivision to create a minimal set of patches that can be evaluated using the Jordan decomposition method.

9.5.1 Reduction of Discretization Error

Adaptive tessellation applies more subdivision in areas of greater interest, producing smaller triangles that reduce the discretization error there. In the case of layered manufacturing, the measure of importance is the estimate of curvature at a control vertex. Our estimate is the maximum absolute distance of the limit positions of the vertices in the 1-ring neighborhood from the limit tangent plane of the central vertex. All of the vertices are inserted into a priority queue based on this error metric. The vertex with the maximum error from the priority queue is considered first.

If it has a complete 1-ring subdivision neighborhood, it is removed from the priority queue and subdivided. The newly generated vertices are added to the priority queue, and the error estimate for all neighboring vertices is updated as well as their positions in the priority queue. If the candidate vertex does not have the proper 1-ring neighborhood, then a neighboring vertex in a previous generation that it is dependent on is subdivided to help fill out the proper neighborhood. Two adjacent vertices in this adaptive mesh never differ by more that one generation. The priority is again updated with new error metric values. This process continues until the maximum error value falls below a user-defined threshold, or until a user specified maximum number of vertices have been generated.

The final subdivision mesh structure is a series of multiple generations of partial meshes. Each control mesh level facet is tessellated separately. When tessellating a control mesh level facet, it is necessary to follow the child pointers down to the finest generation. Complete facets at this finest level are output. Facets that span between two levels of subdivision can always be converted into a fan of triangles while preserving the boundary edges, so that they will be compatible with neighboring patches.

9.5.2 Exact Evaluation

The subdivide-vertex operator is also used to minimize the number of patch evaluators that are constructed. Each patch evaluator stores the eigen-projected control points of its 1-ring neighborhood which is a sizeable amount of data. We apply an adaptive subdivision hierarchy that only subdivides near a patch until the neighborhood is regular enough to apply our evaluation technique. We will describe the situation for Catmull-Clark subdivision; Loop subdivision is very similar.

For Catmull-Clark subdivision, a quad patch p adjacent to an extraordinary vertex can be evaluated if its opposite corner vertex is a regular smooth vertex ($n = 4$), and its other two vertices are either regular smooth vertices or crease vertices with regular wedges. A regular crease wedge has two quads in between the infinitely sharp tagged edges. In addition, the 1-ring of patches must be all quads. If a patch does not meet this criteria, then the patch will be marked as incomplete. We perform a vertex subdivision to each of the vertices of all incomplete patches throughout the generations until all patches have the correct 1-ring neighborhood. A patch evaluator is constructed for all leaf patches throughout the generations. The hanging layers of partial subdivision meshes form a quadtree of patches over the generations.

A patch evaluator stores the eigen-projected control points for the 1-ring neighborhood of

the patch. Then, based on the patch type and valence of the vertex or wedge, the evaluator raises the subdivision matrix to the necessary power using the Jordan decomposition and multiplies by the evaluated spline basis functions. The current implementation stores the tables of eigenvalues and eigenvectors statically for a set number of valences for each patch type. With our new analytic eigenvector functions based on sines and cosines, it is possible to allocate and generate these tables on demand. This will greatly reduce the compiled size of the executable. It is still advisable to place a limit on the maximum valence supported, or else the tables will grow very large and cause the program to run out of memory.

The adaptive subdivision must locally subdivide near semi-sharp features until they revert to the smooth rules, and it must subdivide near infinitely sharp extraordinary features until they are sufficiently separated, which takes at most two subdivisions. To simplify our parameter mappings per patch, we uniformly subdivide the top-level mesh if it has any non-quadrilateral patches.

We thus define that the top-level mesh be all quads. Our parameterization of the entire subdivision is such that each top-level quad has its own continuous $\{[0,1],[0,1]\}$ parameter space. We do not attempt to align these parameter orientations globally across the 2-manifold because extraordinary vertices will cause singularities that cannot be avoided. Evaluating a (u,v) parameter value of a top-level patch may involve descending the adaptive quadtree hierarchy until a leaf facet with a patch evaluator is encountered. While descending the quadtree, the parameters are scaled and rotated to generate the transformed local parameter coordinates $(\bar{u},\bar{v})$ for the patch evaluator. Derivatives must be scaled proportional to the depth of traversal of the quadtree to account for the scaling by a power of two in the change of variables.

Figures 9.6(a), 9.6(b), and 9.6(c) illustrate our adaptive hierarchy. The size of each quad shows at which generation of subdivision it was able to be evaluated. The color of the patches correspond to the type of extraordinary vertex to which they are adjacent. The red edges and vertices correspond to infinitely sharp features, while the yellow edges and vertices correspond to semi-sharp features.

9.6 Editing of Control Vertices

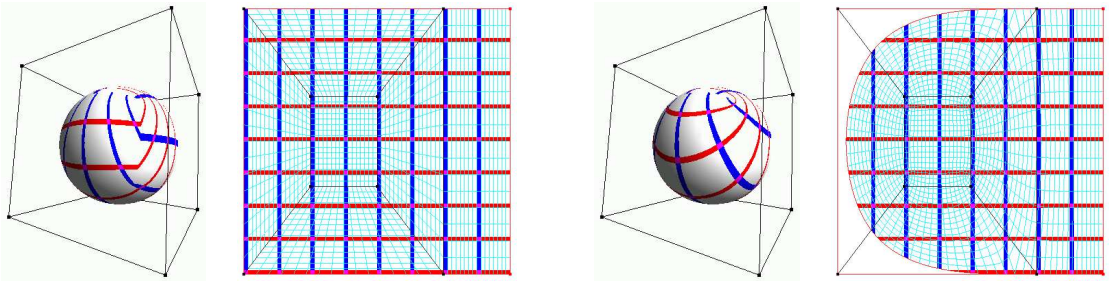
In our current subdivision libraray, the position data of the control vertices can be modified by a user interactively. Modifying vertex data does not affect the topology of the adaptive evaluation mesh structure. Edits to the sharp and semi-sharp tags can cause large scale topological changes, so we currently limit the interface to only allow the user to edit this positional vertex data. When

the position of a single control vertex is edited, it affects the positions of the descendant vertices in a pyramid region. The edit algorithm proceeds by first updating the positions of the vertices in the current generation, and then records a set of vertices in the next generation that are dependent on the altered vertices in the current generation. This set of vertices is passed down to the next generation, and the process is repeated until the leaves of the mesh hierarchy are reached. At the leaf level, a facet has a patch evaluator associated with it. The evaluator must be reinitialized with the modified set of control positions, and the new set of eigen-project control points must be calculated. The set-based interface of edited vertices in the algorithm allows the user to edit a set of vertices at the same time instead of just a single vertex. In the future, this facility could be augmented to make local topology changes as well, but currently all child meshes are deleted and the adaptive hierarchy must be rebuilt.

9.7 Parameter Space Subdivision

Our implementation allows for two different schemes of averaging the parameter data stored in the half-edges, i.e. texture coordinates. The simpler scheme is bi-linear interpolation over the quadrilateral patches. The only differences to the algorithms described above are that the parameter data for newly created half-edges must be set according to the bi-linear averaging rules. The neighborhood of support is limited to a single facet, so the topological operators for the vertex data are overly conservative for this parameter data. In other words, the parameter data does not place any more constraints on the adaptive subdivision algorithms. The problem with bi-linear interpolation of texture coordinates is that discontinuities in the texture coordinate domain can lead to noticeable artifacts in the final texture mapping of otherwise smooth subdivision surfaces. One solution to this problem is to apply a relaxation algorithm to the 2D parameter space mesh, or specially construct the texture map to hide these discontinuities. Another scheme to deal with this problem is to apply the same set of averaging rules used on the vertex data to the parameter data [6], which is part of the RenderMan interface [21]. Figure 9.4 compares bilinear interpolation to Catmull-Clark smoothing of texture coordinates. Both schemes start with the texture coordinates of the smooth as possible parameter mapping from Figure 9.5(b) with the cube rotated 90° downward. One of the diagonal parameter space edges is visible as a vertical discontinuity in the bilinear cube, but it is smoothed out in the Catmull-Clark case. The horizontal discontinuity at the top of both cubes is due to the outer boundary edges of the parameter space mesh. Parameter space boundary edge discontinuities are unavoidable for 3D meshes that are not topologically equivalent to a mesh

in the 2D plane.

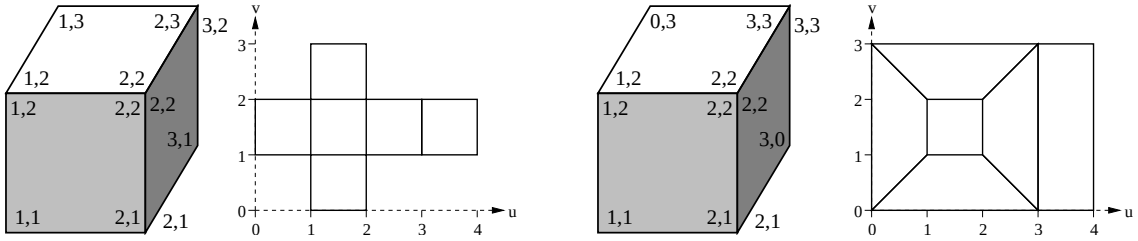


(a) *Bilinear parameter smoothing* (b) *Catmull-Clark parameter smoothing*

Figure 9.4: *A cube with smooth as possible texture coordinates rendered using bilinear parameter smoothing on the left and Catmull-Clark parameter smoothing on the right. Each shows the 3D texture mapped result as well as the 2D parameter space mesh subdivided four times.*

Applying subdivision to the parameter data stored in the half-edges is equivalent to specifying a set of parameter data sharp tags to the subdivision mesh in addition to the vertex data sharp tags. With the two sets of sharp tags, one subdivision mesh actually describes two separate topologies. The parameter space sharp tags are not assigned directly by the user, but instead are derived from the specified parameter data at the half-edges. Parameter space edge tags are either smooth or infinitely sharp. Every half-edge is potentially an edge in the parameter space topology. For instance, if each quadrilateral patch is assigned its own separate disjoint square of parameter space, then the result is that all of the half-edges are marked as sharp parameter edges, so it is equivalent to bi-linear averaging. Smooth subdivision averaging of parameter space data can only be performed across an edge if the two pairs of half-edge corner information at the vertices of both end points agree, respectively. A vertex can only be smoothly averaged if all of its incident edges are smooth parameter edges. Otherwise, the vertex is either a parameter space crease or corner vertex. A parameter space dart vertex can never occur because if a single edge is a parameter space sharp edge, then both its incoming and outgoing half-edges are parameter space sharp edges. Hence it is a parameter space crease vertex. Also by definition, any crease wedge with a single patch is tagged to be a parameter space corner vertex. An intuitive way of thinking about the topology described by the parameter space sharp tags is to think about trying to flatten the mesh of facets into the 2D parameter plane. For example, a cube must be cut along some subset of its edges, so that it can be flattened. These cuts correspond to sharp parameter edges and discontinuities in the texture mapping coordinates. Figure 9.5 shows two possible parameter space mappings of a completely smooth

cube control mesh. In both cases, the topology of the parameter space mesh is different from the vertex data mesh. Demands of the parameter space tagging can sometimes cause more adaptive subdivision for evaluation purposes than is obvious from the vertex data sharp tagging.

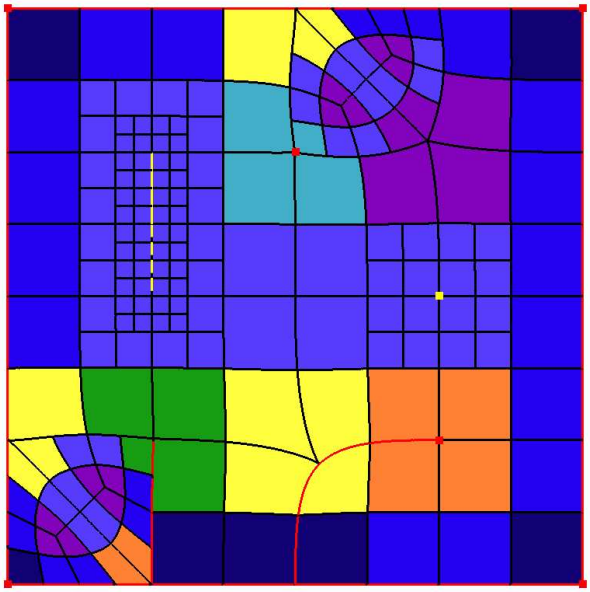


(a) *T-bar shaped parameter mapping* (b) *Smooth as possible parameter mapping*

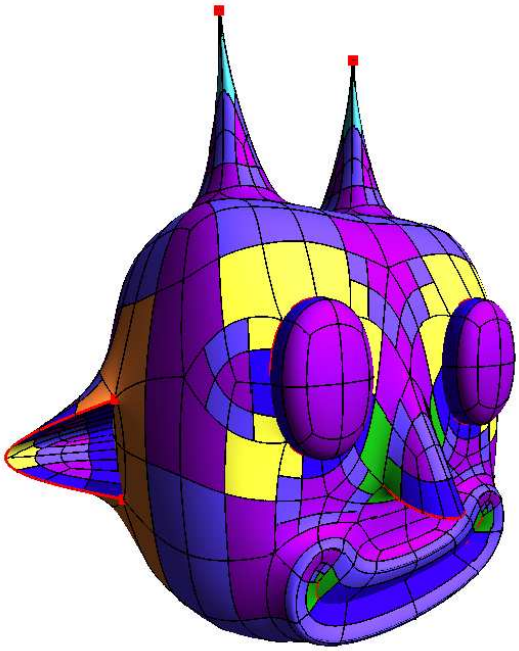
Figure 9.5: *Two parameter space mappings of a cube. The left mapping has no smooth vertices, only corners and creases. In the right mapping, the four vertices of the front face are smooth, while the rest are corners or creases.*

9.8 Conclusion

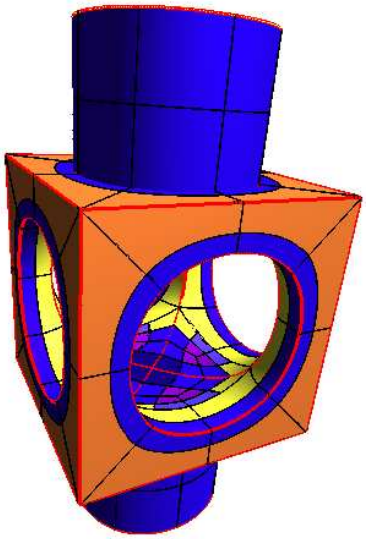
We have completely implemented the algorithms described above for Catmull-Clark subdivision. We have also implemented Loop and many other subdivision algorithms to certain degrees. In the near future, we plan to implement the Loop patch evaluation. We also plan to investigate implementing evaluation for the new C^2 hybrid scheme of triangles and quadrilaterals [14, 20]. Our implementation has been used to design objects which are then adaptively tessellated and sent on to the layered manufacturing process planning stage. Chapter 10 will discuss some of the geometric processing that is done on the output watertight triangle mesh.



(a) *2D adaptive subdivision hierarchy*



(b) *Alien head*



(c) *Mechanical part*

Figure 9.6: *Adaptive subdivision hierarchies. The size of the quads indicates the subdivision generation. The color specifies the patch type: blue is a regular B-spline, medium blue is a regular B-spline crease patch, dark blue is a regular B-spline corner patch, purple is a smooth extraordinary vertex, cyan is a spike, green is a dart, yellow is a crease, and orange is a corner. Sharp edges and vertices are drawn in red, while semi-sharp ones are drawn in yellow. The yellow semi-sharp edge has sharpnes 3, while the semi-sharp vertex has sharpness 2.*

Chapter 10

Robust Voronoi Diagrams of Sets of Polygonal Contours

10.1 Introduction

An algorithm for robustly constructing the Voronoi diagram (VD) of a set of 2D oriented polygonal contours is useful in many domains including geometric modeling, manufacturing analysis, and obstacle avoidance in path planning. The work presented in this chapter is motivated by the application of geometric processing for layered manufacturing. The goal is to speed up the build time of fused deposition modeling (FDM) parts by reducing the amount of material used in their production [18]. This goal is achieved by building a semi-hollow thin-walled version of the part. The thin-walled geometry is generated by making a conservative $2\frac{1}{2}$D approximation to the true 3D offset surface. An important step in this process is the construction of 2D contour offsets from a set of oriented piecewise linear input contours. The Voronoi diagram of the input contours is used to make the offsetting process fast and robust.

The input contours are created by slicing a closed 3D polyhedral boundary representation (B-Rep) with a build layer plane. Given that the B-Rep's are closed 2-manifold polyhedrons, the set of slice contours on a single layer will be piecewise linear, closed, oriented, non-intersecting, and possibly nested. While walking around an oriented contour, the inside of the part is on the left hand side and the outside is on the right hand side (Figure 10.10(a)). Counterclockwise (CCW) contours specify the boundary of positive regions while nested clockwise (CW) contours specify negative holes. The nesting of contours can be arbitrarily deep with the restrictions that the orientations

alternate between CCW and CW, and that the top level contours are CCW.

The Voronoi diagram plays an important role in the creation of the 2D offset contours [17]. We represent the VD as a Voronoi mesh using our half-edge variant of the quad-edge data structure [9]. Once created, the Voronoi mesh is a compact representation of the signed distance field defined by the input set of contours. We define the inside of the contours to have positive distances and the outside to have negative distances. The Voronoi mesh can be viewed in 3D as a height field by raising all of the points of the plane by their signed distance from the input contours (Figure 10.11(a)). Offset contours can be quickly and robustly generated by running an algorithm that effectively slices this Voronoi height field mesh with a plane $z = d$ where d is the offset distance; more details are in a previous publication [19]. Other offsetting algorithms offset individual contour elements separately and then run a clean up pass which tries to eliminate erroneous loops created by self interestions [17]. These algorithms tend to be slow because global interactions arise between contour elements that are spatially close but topologically distant (Figure 10.11(b)). These algorithms are also less robust because they generate intermediate geometric entities, and then perform numeric intersection tests using them. It is always possible to construct a set of input to this kind of algorithm that will either crash the program or generate invalid Voronoi meshes where some faces overlap each other. Our approach addresses these robustness issues during the construction of the Voronoi diagram.

In this chapter, we describe a divide-and-conquer algorithm that builds the Voronoi diagram of a set of 2D oriented, non-intersecting polygonal contours in a numerically stable way. Numerical robustness issues are addressed by using geometric predicates that depend only on the original contour elements and not on any generated geometric entities. As a result, we avoid the issue of floating point round-off error in irrational values that cannot be represented in a finite amount of space on a computer. Our geometric predicates are in a class that can benefit from adaptive exact arithmetic techniques [26].

10.1.1 Background

The algorithm presented in this paper is designed to work for input sets of closed piecewise linear contours, but a more general treatment of Voronoi diagrams is useful for discussing the properties of this mathematical entity. The treatment in this section is fairly abstract. For a concrete example, consider the Voronoi diagram of a set of 2D points where the sites are the input vertices (Figure 10.1(a)).

The definition of a Voronoi diagram (VD) of a set of n general input sites $S_0, S_1, ...S_{n-1}$ in $\Re^2$ is a partitioning of the plane into n respective Voronoi faces $VF_0, VF_1, ...VF_{n-1}$ such that every point that lies in the interior of VF_i is closer to S_i than to any other input site. The area of a VF may be infinite. Consider the simple example of a single input site S_0. Its associated Voronoi face, VF_0, which is the only one, covers the entire plane extending out to infinity in all directions. With $n>1$, two adjacent Voronoi faces VF_i and VF_j meet along a boundary curve called a Voronoi edge $VE_{i,j}$. All of the points that lie on the interior of $VE_{i,j}$ are equidistant from the two sites S_i and S_j that are associated with VF_i and VF_j respectively and are closer to those two sites than to any other input sites. Hence the Voronoi edge $VE_{i,j}$ is the bisector curve between the two input sites S_i and S_j. Like VF's, VE's can extend to infinity in one or both directions. When two VE's meet, they form an endpoint called a Voronoi vertex (VV).

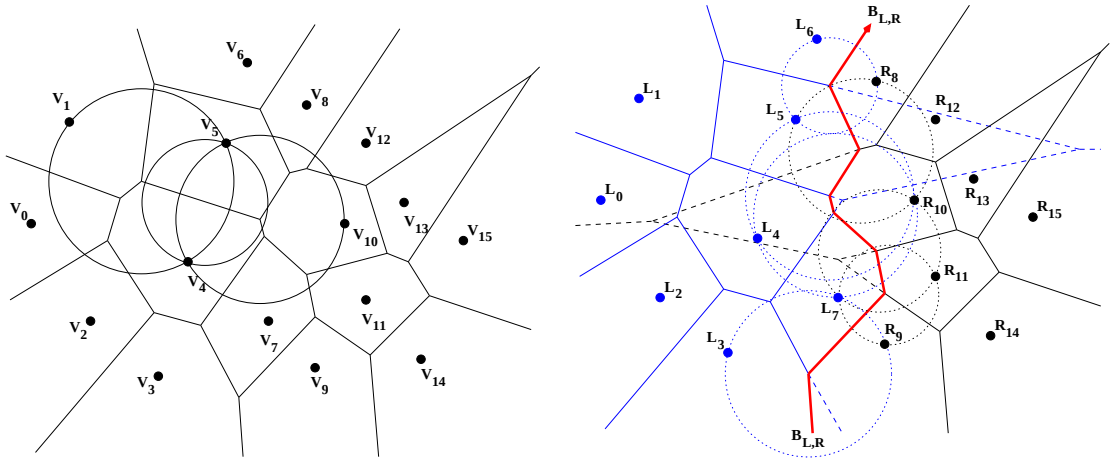
For a given Voronoi face VF_i, two consecutive boundary edges $VE_{i,j}$ and $VE_{i,j+1}$ around the face will meet at $VV_{i,j,j+1}$. VF_i, VF_j, and VF_{j+1} will be three distinct Voronoi faces that correspond to the three distinct input sites S_i, S_j, and S_{j+1} respectively. Since $VV_{i,j,j+1}$ lies on both the bisector edges $VE_{i,j}$ and $VE_{i,j+1}$, $VV_{i,j,j+1}$ is a point that is equidistant from the three input sites S_i, S_j, and S_{j+1}, and no other site is closer to $VV_{i,j,j+1}$. Let r be the distance from $VV_{i,j,j+1}$ to any of these three sites. Let C be a circle centered at $VV_{i,j,j+1}$ with a radius of r. The input sites S_i, S_j, and S_{j+1} lie on the boundary of C, and since no input site passes closer to the center $VV_{i,j,j+1}$, the interior of C must be free of any of the input sites. A general position assumption that disallows more than three co-circular input sites is often used to simplify the analysis of the complexity or performance of geometrical algorithms. No such assumption will be used here, so the input may have three or more co-circular sites that touch the circle C. As a final mental exercise of this general treatment of Voronoi diagrams, imagine sliding C's center point off of $VV_{i,j,j+1}$ along one of the incident bisector edges, for example $VE_{i,j}$, and constraining the radius r to take on the value equal to the common distance from the new center position to the two closest input sites, S_i and S_j. At all times, the boundary of the circle will touch at least two input sites, and the interior of the circle will remain empty. The property of the empty inscribed circles is important to understanding the divide-and-conquer algorithm.

10.2 Related Work

Many algorithms to construct Voronoi diagrams have been described. We will limit our discussion to the divide-and-conquer algorithms that form the foundation for the algorithm pre-

sented. Divide-and-conquer Voronoi algorithms recursively split the set of n input sites in half, forming a top-down binary tree, until a base case set of size 1, 2, or 3 is reached at the leaves. The Voronoi diagram of these simple sets can be trivially constructed. Then the splitting process is reversed by a bottom-up merge process. At each node of the tree, the Voronoi diagram of the left child (VD_L) and the right child (VD_R) are merged together to form the combined Voronoi diagram of all of the children sites. The VD of the entire input set of sites is constructed when the final merge operations is completed at the root of the tree. There are $O(\log(n))$ levels in the tree. For all of the algorithms discussed in this paper, the merge step over all nodes per level will total $O(n)$, so the running times will be $O(n\log(n))$.

10.2.1 Voronoi Algorithm for a 2D Point Set



(a) *Voronoi diagram of a point set with inscribed circles*

(b) *Final merge of Voronoi diagram of a point set.* $SV_{l_0} = L_3$ *and* $SV_{r_0} = R_9$.

Figure 10.1: *Voronoi algorithm for a set of 2D points*

The first description of a divide-and-conquer Voronoi algorithm took a set of n 2D vertex sites $SV_0, SV_1, ...SV_{n-1}$ as input. This algorithm was first described by Shamos and Hoey [25] and later refined by Lee and Drysdale [13]. The techniques developed in this algorithm are the basis for our algorithm for contours. The bisector between two input vertices SV_i and SV_j is the straight line perpendicular bisector to the line segment defined by SV_i and SV_j. These bisector lines intersect to form the VV's. The VE curves are straight line segments and rays. The VF's are convex polygons. If a Voronoi face VF_i is associated with the input vertex SV_i that lies on the convex hull of the input

vertices, then VF_i extends to the point at infinity.

The most important contribution of this algorithm is the description of the merge process (Figure 10.1(b)). The vertex set is split into a left half set (L) and right half set (R), and VD_L and VD_R are constructed independently. When taken together, VD_L and VD_R will overlap each other. The task of the merge is to construct the bisector polyline $B_{L,R}$ between L and R and to trim off the extra portions of VD_L and VD_R that are no longer valid. An individual segment of $B_{L,R}$ will be a bisector Voronoi edge $B_{l,r}$ between a site SV_l from L and a site SV_r from R. The key insight is that all portions of VD_L that lie to the right of the not yet created $B_{L,R}$ are closer to some site in R than any of the sites in L, so they are not part of the composite VD of L and R. Similarly, all portions of VD_R that lie to the left of $B_{L,R}$ are closer to some site in R than any of the sites in L, so they are not part of the composite VD of L and R.

Each step in the merge process searches for the terminating endpoint of a bisector ray $B_{l,r}$ between SV_l and SV_r that has been started from the below. To start, SV_{l_0} and SV_{r_0} are the bottom two vertices that define a convex hull edge that connects L and R. Their bisector starts at infinity in the $-y$-direction. A merge step walks around the VE's of VF_l and VF_r, associated with SV_l and SV_r respectively, looking for the intersection points with $B_{l,r}$. Every edge $VE_{l,i}$ of VF_l that is on the right side of $B_{l,r}$ is closer to SV_r than to SV_l; hence $VE_{l,i}$ will not be in the final Voronoi diagram. To prevent back tracking in the merge step, the edges of VF_l are traversed in CCW order starting from the starting point of $B_{l,r}$; every edge $VE_{l,i}$ up until the edge that intersects $B_{l,r}$ lies to the right of $B_{l,r}$ and can be collapsed never to be considered again. The same argument holds for a CW traversal of the edges of VF_r where the edges $VE_{r,j}$ on the left side of $B_{l,r}$ are collapsed. $B_{l,r}$ is terminated at the closer of the two intersections points. If $VE_{l,i}$ from VF_l is closer, then VF_i will be broken open at the intersection point $VV_{i,l,r}$, and the scan process will be repeated for a new bisector segment $B_{i,r}$ between SV_i and SV_r. The opposite is true if the intersection point $VV_{l,r,j}$ is closer. The merge process continues until no more intersections exist, at which point $B_{L,R}$ is completed and VD_L and VD_R are merged. There is no back tracking in the scan of a VF, so an edge is traversed at most once. Since the number of edges in L and R combined is linearly proportional to the number of input sites in L and R by the Euler relation, the running time of a single merge process is $O(n)$.

10.2.2 Rising Bubble Method for 2D Point Sets

The calculations to find the intersection point of $B_{l,r}$ and $VE_{l,i}$ work on derived geometric objects, so they are prone to numeric round-off error. The search for an intersection point can be

made more numerically stable by reinterpreting it as the search for the center of an empty circle inscribe by three ordered sites. This formulation of the merge process was first described by Guibas and Stolfi [9] in their work on Delaunay triangulations. Relating this work to the previous algorithm, $B_{L,R}$ is the path of the center of a rising empty circular bubble. The radius of the circle shrinks and grows to stay in contact with two or three vertex sites while it moves along the VE's that comprise $B_{L,R}$.

In this new phrasing of the intersection test, the CCW merge scan around VF_l will advance from $VE_{l,i}$ to $VE_{l,i+1}$ if a CCW oriented circle C passes through SV_{i+1}, SV_l, and SV_r in CCW order, and C is empty of SV_i. This test is equivalent to treating $VE_{l,i}$ and $VE_{l,i+1}$ as infinite lines and picking the one that the ray $B_{l,r}$ intersects first. The counterclockwise test of the circle ($VCCW$) ensures that $VE_{l,i+1}$ intersects with the forward direction of the ray $B_{l,r}$.

$VCCW\,(V_0,\ V_1,\ V_2)$ is equivalent to testing whether V_2 lies on the left hand side of the oriented edge from V_0 to V_1. $VInCircle\,(V_0,\ V_1,\ V_2,\ V_3)$ tests to see if V_3 is inside the oriented circle inscribed by V_0, V_1, and V_2. Equations 10.1 and 10.2 show that the $VCCW$ and $VInCircle$ tests depend directly on the input coordinate values [9].

$$VCCW\,(V_0, V_1, V_2) = \begin{vmatrix} V_{0_x} & V_{0_y} & 1 \\ V_{1_x} & V_{1_y} & 1 \\ V_{2_x} & V_{2_y} & 1 \end{vmatrix} > 0 \tag{10.1}$$

$$VInCircle\,(V_0, V_1, V_2, V_3) = \begin{vmatrix} V_{0_x} & V_{0_y} & \left(V_{0_x}^2 + V_{0_y}^2\right) & 1 \\ V_{1_x} & V_{1_y} & \left(V_{1_x}^2 + V_{1_y}^2\right) & 1 \\ V_{2_x} & V_{2_y} & \left(V_{2_x}^2 + V_{2_y}^2\right) & 1 \\ V_{3_x} & V_{3_y} & \left(V_{3_x}^2 + V_{3_y}^2\right) & 1 \end{vmatrix} > 0 \tag{10.2}$$

The redesigned algorithm uses only these predicates when making numeric decisions in the merge process. Hence any algorithmic problems due to round-off error are limited to the calculation of these predicates. No numeric decisions are made using constructed geometric objects.

10.2.3 Adaptive Exact Arithmatic

The work of Shewchuck [26] demonstrates that the $VCCW$ and $VInCircle$ predicates can be made free of round-off error using adaptive exact arithmetic. With an adaptive technique the accumulated error in the calculation is dynamically tested against error bounds to decide whether

to increase the resolution of the calculation. The approach checks against error bounds at a discrete number of resolutions and defaults to a more costly exact evaluation if necessary. In order to take advantage of the speed of the adaptive arithmetic, the equations for the predicates are restructured so that the values are more likely to be in a range were floating point round-off error will have less of an effect. The technique is to use relative position vectors from a local origin rather than absolute world coordinates. Assuming exact arithmetic, the relative positions of the input vertices decide the predicate tests regardless of the coordinate system. Without exact arithmetic, these important difference values can be lost to truncation and rounding in floating point operations if multiplications of large absolute positions are performed before the subtractions. In Equations 10.3 and 10.4, the $VCCW$ and $VInCircle$ tests have been modified so that V_0 is the local origin.

$$\begin{aligned} VCCW\left(V_0, V_1, V_2\right) = & \begin{vmatrix} V_{1_x} - V_{0_x} & V_{1_y} - V_{0_y} \\ V_{2_x} - V_{0_x} & V_{2_y} - V_{0_y} \end{vmatrix} > 0 \\ = & \| \left(V_1 - V_0\right) \times \left(V_2 - V_0\right) \| > 0 \end{aligned} \tag{10.3}$$

$$VInCircle\left(V_0, V_1, V_2, V_3\right) = \begin{vmatrix} V_{1_x} - V_{0_x} & V_{1_y} - V_{0_y} & \| \left(V_1 - V_0\right) \|^2 \\ V_{2_x} - V_{0_x} & V_{2_y} - V_{0_y} & \| \left(V_2 - V_0\right) \|^2 \\ V_{3_x} - V_{0_x} & V_{3_y} - V_{0_y} & \| \left(V_3 - V_0\right) \|^2 \end{vmatrix} > 0 \tag{10.4}$$

With exact arithmetic versions of addition, subtraction, and multiplication, it is possible to determine the result of these two predicates precisely. Because the predicates compare an expression with zero, division can be avoided by cross multiplying. Some square roots can be avoided by comparing the squared distance instead of the Euclidean distance, as in the $VInCircle$ predicate of Equation 10.4. For the expanded generalized set of predicates that are necessary with line segment input sites, it is sometimes necessary to compute square roots. We use a fast interval arithmetic floating point filter for all of our new predicates. If the ambiguity of the resulting computation contains zero, we depend on a slow exact arithmetic version using the MPFUN numeric library [3].

The techniques of using relative positions, avoiding divisions, and avoiding square roots are useful in avoiding floating point round-off error in geometric constructors as well. Geometric constructors, such as generating intersection points or circle centers, cannot be stored exactly because the answers might involve irrational numbers that cannot be represented in a computer in a finite amount of space. Examples of relatively accurate constructors are the equations for the center and radius of the circle inscribed by three vertices [26], as shown in Figure A.2 and Equation A.3.

10.2.4 Voronoi Diagram of Polygonal Contours

So far, the discussion of Voronoi diagrams has been limited to input sets of 2D vertex sites only. Our algorithm constructs the Voronoi diagram of piecewise linear contours, so the definition of the Voronoi diagram must be generalized to handle vertex sites (SV) and oriented edge sites (SE). Lee and Drysdale [13] presented an algorithm for computing the VD of general arrangements of finite line segments and their endpoints. Then based on that work, Held [17] developed an algorithm for the VD of the inside of a contour made up of straight line segments and circular arcs. Our algorithm is a modification of Held's algorithm where the issue of robustness is the key concern. In Held's algorithm, the interactions between the vertex sites, edge sites, and circular arc sites are abstracted by dealing with a general form for the bisector curves. The intersections of these bisector curves guide the decision making in the merge process. Like the first description of the point set algorithm, these intersection tests are prone to round-off errors. Our work modifies this Voronoi algorithm so that it is more robust against these round-off errors. To keep this effort manageable, we disallow circle arcs in the input contours to limit the number of cases in the implementation. The number of cases is proportional to the number of combinations of input site types. In the future, we plan to extend our robust algorithm to support circular segments; in the mean time we can approximate circular arcs with a suitable number of line segments. Reducing the set of input site types to vertices and line segments still serves many real applications including the offsetting of polyhedral slice contours described in the introduction [19].

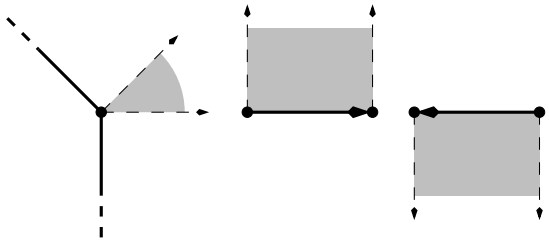


Figure 10.2: *Vertex zone, oriented edge zone, opposite oriented edge zone*

We use a definition of the Voronoi diagram based on the zones of influence of the input sites [17], but modify it to work for the inside and the outside of the contour. The zones of influence address the fact that the vertices and edges are connected to neighboring sites around the contour (Figure 10.2). The zone of influence of an oriented edge site lies to the left of the oriented edge in between two lines perpendicular to the edge at its two endpoints. By definition, any point to the

right side of the oriented edge is closer to the opposite oriented edge site, and any point beyond one of the oriented edge's endpoints is closer to the corresponding vertex site. Vertex sites have a zone of influence in the angle between the two normals to its two incident edges at that point. This region only exists on the side of the contour where the angle between the edges is greater than $180°$. On the side of the contour where the zone exists, the vertex is called a *reflex* vertex. On the other side of the contour, the same vertex is called *non-reflex*. By definition, the common vertex between two consecutive collinear edge segments of the contour will have no zone of influence, because the two edge zones will meet along the perpendicular at the vertex causing the vertex zone to vanish.

The VF's are associated with a vertex site or an oriented edge site. In the plane, the bisector curves between two SE's or two SV's are straight lines. The bisector curve between an SE and a SV is a parabola. In the 3D height function (Figure 10.11(a)), the VF of a SE is a portion of a $45°$ angled plane, and the VF of a SV is a portion of a cone with $45°$ angled sides. Also note that the distance function along the bisector of two SV's is a hyperbola.

Our representation of the Voronoi mesh is built on our half-edge data structure with added pointers from a VF to the site it is associated with. The representatoin of the VE's is purely topological. The type of bisector curve is implicity specified by the types of the two sites associated with the two adjacent VF's.

10.3 Overview

The input to our Voronoi algorithm is a set of 2D oriented, nonintersecting polygonal contours (Figure 10.10). The output is the Voronoi mesh of those contours where the sites are the vertices and the oriented line segment edges of the contours. The approach of the algorithm is to construct the VD's of the contours independently and merge each one into a composite VD. Once all of the contours have been processed, the composite VD will be the VD of the set of contours. The Voronoi vertices are assigned a z-coordinate equal to the signed distance to the input contours where points on the inside are positive and points on the outside are negative. Finally, a clean up routine is run over the Voronoi mesh to make it better suited to down stream applications like offsetting.

10.4 Single Contour Voronoi Diagram

The first step in creating the VD of a set of contours is constructing the VD of a single contour using a divide-and-conquer approach. The input to this routine is a CCW oriented, simple polygonal contour (Figure 10.4(a)). The result is the Voronoi mesh where the VV are assign a z-coordinate equal to their signed distance to the contour. The contour separates the plane into two areas, inside and outside. If a VV lies inside the contour, it is given a positive distance, and if it lies outside the contour, it is given a negative distance. The VD's of the inside and outside are independent of each other and meet along the contour. Taking advantage of this, the VD's of the inside and outside are constructed separately and then trivially merged at the contour (Figure 10.4). The processes for creating the inside and outside VD are very similar. On the outside the edges are traversed in the opposite direction, so in both cases the VD will be constructed on the left hand side walking around the contour.

The divide-and-conquer method recursively breaks the contour in half in a tree fashion until the pseudo-Voronoi diagram of the partial contour path at each leaf can be easily constructed. These pseudo-Voronoi diagrams are correct in a region local to the path, but in areas further away from the path, the VF's can overlap each other. Then reversing the recursion, the two pseudo-Voronoi diagrams of the children subpaths of each node of the tree are merged together starting at a common endpoint to form a larger pseudo-Voronoi diagram. The final merge at the root of the tree will create the true Voronoi diagram of the inside or outside of the contour. These two VD's are trivially glued together along the contour and the VD of the contour is formed.

10.4.1 Maximal Reflex Paths

The algorithm for a single side of the oriented contour begins by breaking the contour up into a set of maximal reflex paths (MRP) [17]. An MRP is a consecutive set of edges connected by reflex vertices where the oriented path turns to the right or continues straight ahead (Figure 10.3). An MRP begins at a non-reflex vertex and ends at another non-reflex vertex, so it is maximally reflex. It is apparent that the set of MRP's for the inside and outside invocations of the algorithm will be different (Figure 10.10).

The algorithm walks around the oriented contour searching for a non-reflex vertex V_i, i.e. such that $VCCW\,(V_{i-1},\ V_i,\ V_{i+1})$ is true. On the inside of the contour, a non-reflex vertex will always exist, but on the outside of the contour, a non-reflex vertex may not exist. If the input contour is convex then all of the outside vertices are reflex. This simple example illustrates the efficiency

benefit of dividing the contour into MRP's. The outside VD of a convex contour is trivial to construct. It simply is the contour plus rays perpendicular to the edges at the end points. These rays separate the reflex vertex zones from the edge zones. If two consecutive edges are collinear, then the VF for the common endpoint does not exist and only a single perpendicular ray is constructed to separate the two edge zones. If one or more non-reflex vertices exists, the MRP's for a contour are traced out by starting at the first non-reflex vertex and walking around the oriented contour marking the end of each MRP at each following non-reflex vertex until the first convex vertex is visited again. The pseudo-Voronoi diagram of each MRP is trivially constructed in the same way that the VD of the outside of a convex contour is constructed. The only difference is that the endpoints of the MRP are not treated as sites because they are non-reflex and will not have associated VF's on this side of the contour. Note that the simplest MRP is a single edge without either of its endpoints, so its pseudo-Voronoi diagram will not have any VE's.

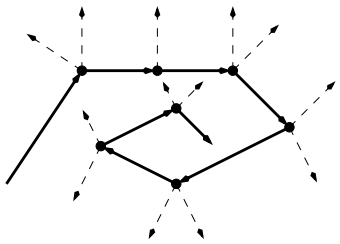
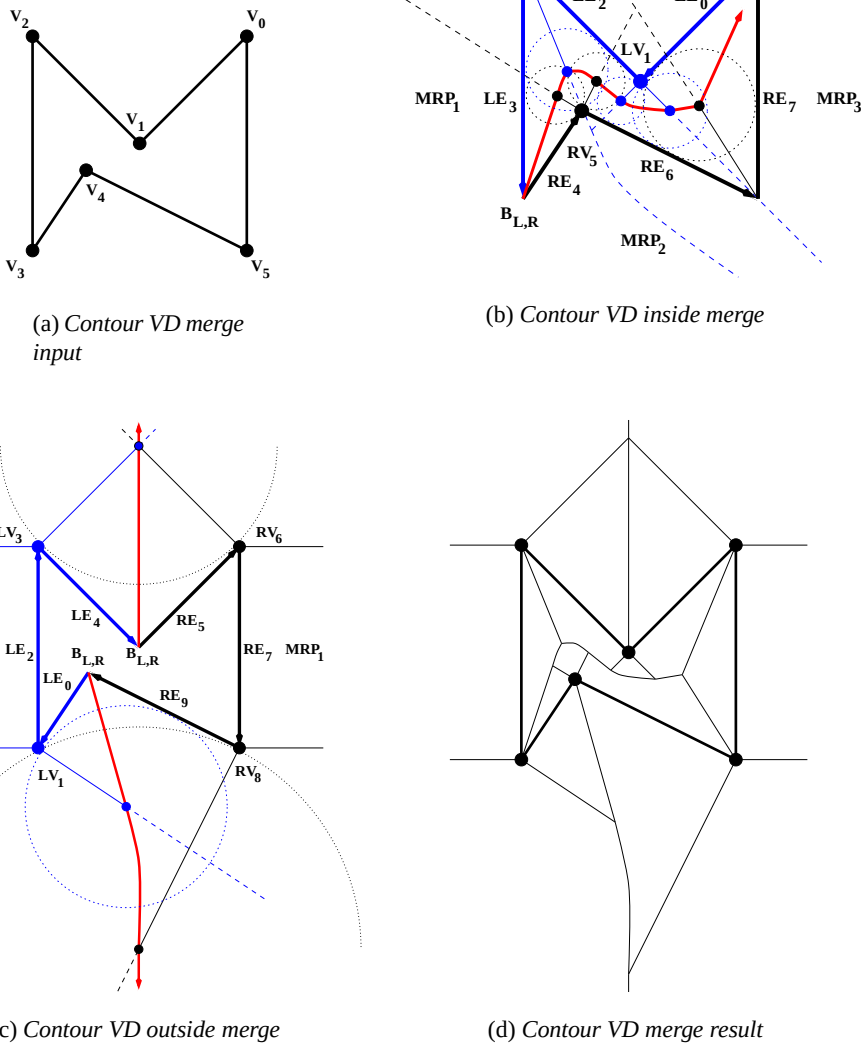


Figure 10.3: *Maximal reflex path with its pseudo-Voronoi diagram*

10.4.2 Divide and Conquer

Once the contour is split up into MRP's, the binary divide-and-conquer algorithm can be applied with the MRP's as the leaves of the tree. Then the pseudo-Voronoi diagrams of two consecutive subpaths are merged together starting at their common non-reflex endpoint. The merge constructs the bisector path $B_{L,R}$ and deletes away the erroneous portions of VD_L and VD_R. The merge continues until it can no longer generate a VV, i.e. no more intersections between $B_{L,R}$ and one of the VE's of VD_L or VD_R exist. The merge algorithm can end before a globally valid VD is created because the endpoints of the subpaths are not included in these pseudo-Voronoi diagrams. Nevertheless the resulting pseudo-Voronoi diagram will have enough local information to guide the later merges.



(a) *Contour VD merge input*

(b) *Contour VD inside merge*

(c) *Contour VD outside merge*

(d) *Contour VD merge result*

Figure 10.4: *Voronoi algorithm for an individual contour*

10.4.3 Merge

The construction of the merge bisector is similar to the contrustion of the one for the VD of a point set. The bisector path $B_{L,R}$ is created by connecting the centers of the empty inscribed circles that mark the intersection points of $B_{L,R}$ with VD_L and VD_R. All portions of VD_L that lie to the right of $B_{L,R}$ will not be in the final VD, and all portions of VD_R that lie to the left of $B_{L,R}$

will not be in the final VD. The procedure starts by fusing the rightmost edge site S_l of MRP_L with leftmost edge site S_r of MRP_R at their shared non-reflex endpoint. The first bisector curve segment $B_{l,r}$ of $B_{L,R}$ is attached to the common endpoint as the angle bisector. The Voronoi face VF_l associated with S_l is broken open at infinity, and a CCW scan of its VE's is started from there at $VE_{l,i}$. Similarly, the Voronoi face VF_r associated with S_r is broken open at infinity, and a CW scan of its VE's is started from there at $VE_{r,j}$. $VE_{l,i+1}$ is the next VE after $VE_{l,i}$ in CCW order around VF_l, and $VE_{r,j+1}$ is the next VE after $VE_{r,j}$ in CW order around VF_r.

The scan of the two VF's VF_l and VF_r is done in two phases. The first phase attempts to reject VV's created by three sites from the same subpath that were created in isolation ignoring the sites from the other subpath. At this point in the merge, S_l and S_r are known to have neighboring VF's separated by $B_{l,r}$, and the merge is looking to find an empty inscribed circle touching S_l and S_r and a third site. VV's of VF_l that are closer to S_r than to S_l will contain some part of S_r inside their associated inscribed circle, and vice versa. This check is accomplished using a generalized version of the in-circle-test called $SInCircle$ (Section 10.7). $SInCircle$ takes any combination of four ordered vertex and edge sites. For the left side, the test is $SInCircle(S_l,\ S_i,\ S_{i+1},\ S_r)$. The first three parameters are the sites that inscribe a circle C, and the fourth parameter is tested against C. If the test is true, then $VE_{l,i}$ is collapsed and the pointer is advanced to $VE_{l,i+1}$. This process is continued until an inscribed circle is found that does not contain S_r or until a boundary condition is reached. The boundary conditions prevent certain VE's from ever being collapsed and prevents $B_{L,R}$ from reversing direction midway through the merge. These protected VE's are the ones that make up the contour and the ones that are incident to the contour because some portion of these edges will be in the final VD. This condition prevents the scan of the edges of a VF from iterating past its associated edge or vertex site, which keeps the generation of the bisector headed in the correct direction. Similarly for the right hand side, the test $SInCircle(S_r,\ S_{j+1},\ S_j,\ S_l)$ is used and $VE_{r,j}$ is collapsed and the pointer is advanced to $VE_{r,j+1}$, where $j+1$ is the next edge in CW order.

The second phase of the merge scan of a VF searches for new empty circles inscribed by the three sites S_i, S_l, and S_r on the left or S_l, S_r, and S_j on the right. To prevent the bisector from reversing directions, a check is made to see if a properly oriented circle can be inscribed within the three ordered sites. The test is more complicated than $VCCW$ (Equation 10.3) from the 2D point set algorithm because the sites can be vertices or oriented line segments. The test is broken into two steps. The first step, a generalized version of $VCCW$ called $SCCW$ (Section 10.5) is used to trivially reject some cases of input geometry. $SCCW$ treats all edge sites as infinite oriented lines

and checks to see if a CCW oriented circle can be inscribed within the arrangement of vertices and oriented lines. $SCCW\,(S_i,\ S_l,\ S_r)$ is called on the left hand side. The boundary conditions are also enforced here. The $SCCW$ filtering step simplifies the complexity of the equations used in the second step. If necessary, the second step attempts to compute the center of the inscribed circle within the three ordered sites, $SCenter\,(S_i,\ S_l,\ S_r)$ (Section 10.6). Like $SCCW$, the equations used in $SCenter$ model finite edge sites as infinite oriented lines. $SCenter$ computes possible solutions for the center point and checks to see if they lie within all the three input sites' Voronoi zones (Figure 10.2) to determine if a solution exists. There is at most one valid solution. $SCenter$ also computes an approximate floating point value of the radius of the circle which it assigns to the z-coordinate of the solution. Similarly on the right hand side, the merge process tests for the existence of an inscribed circle center by calling $SCCW\,(S_l,\ S_r,\ S_j)$ and $SCenter\,(S_l,\ S_r,\ S_j)$.

The scan continues to search for the optimal circle center. The optimal circle center is the VV at the nearest intersection of $B_{l,r}$ with VD_L and VD_R. On the left, if $SCenter\,(S_{i+1},\ S_l,\ S_r)$ exists, then $SInCircle\,(S_{i+1},\ S_l,\ S_r,\ S_i)$ is called to test if its associated circle is empty. If so $VE_{l,i}$ is collapsed. In parallel on the right, if $SCenter\,(S_l,\ S_r,\ S_{j+1})$ exists, then $SInCircle\,(S_l,\ S_r,\ S_{j+1},\ S_j)$ is called to test if its associated circle is empty. If so $VE_{r,j}$ is collapsed. This process continues until no more progress is made on the left or the right. Then if there are two possible answers, a final $SInCircle\,(S_i,\ S_l,\ S_r,\ S_j)$ test is performed to determine which center is the next VV. If the empty circle is on the left hand side, then VF_i is broken open and $B_{l,r}$ is attached to $VE_{l,i}$ at $VV_{i,l,r}$. The left hand edge pointer is advanced to the next VE to $VE_{l,i}$ around VF_i in CCW order. Similarly, if the empty circle is on the right hand side, then VF_j is broken open and $B_{l,r}$ is attached to $VE_{r,j}$ at $VV_{l,r,j}$. The left hand edge pointer is advanced to the next VE to $VE_{r,j}$ around VF_j in CW order.

The merge scan of the VF's is repeated until no further empty circles can be formed. At this point, $B_{L,R}$ has been constructed and VD_L and VD_R have been merged.

10.4.4 The Final Merge

Merges between adjacent pseudo-Voronoi diagrams continue until the final merge at the root of the recursion tree. On the final merge step, two non-reflex vertices, the starting vertex and the one on the other opposite side of the oriented contour, must be fused by the merge process to form the VD of the one side of the contour. When merging the inside of the contour, the merge will complete at the non-reflex junction of the first and last edges of the contour. In this case, the

terminating end of $B_{L,R}$ can be easily attached to the contour (Figure 10.4(b)). When merging the outside of the contour, there are two different cases. The first case is where $B_{L,R}$ spans across a concavity enclosed on the outside of the contour and meets up with the second junction vertex. This case is exactly the same as the inside case. The second case is where $B_{L,R}$ extends out to infinity (Figure 10.4(c)). In this case, one additional merge scan is run starting from the final unjoined non-reflex vertex. It is also possible that the outside orientation of the contour will have exactly one non-reflex vertex. In this case, the single merge process will extend out to infinity.

The VD's of the inside and outside orientations of the contour have been created by separate divide-and-conquer merges. The VV's of the inside VD are assigned positive distances, and The VV's of the outside VD are assigned negative distances. Then the two meshes are trivially stitched together along the contour itself, thus constructing the signed Voronoi mesh of the input CCW contour.

10.4.5 Clockwise Oriented Contours

The CW oriented contours from the input set are handled by the same algorithm. A contour can be classified as CCW or CW using a predicate that checks its signed area. CW oriented contours are reversed and treated like CCW contours. Since the VD's of individual contours are combined later in the process, it is important that the sign of the distance function in the merge area matches. This is achieved by negating the distance values in the VV's of the CW oriented contours.

10.5 Generalized Counterclockwise Test

Our generalized counterclockwise test $SCCW$ is a trivial rejection test that filters out bad input to the $SCenter$ inscribed circle center constructor. The $SCCW$ predicate takes any ordered triple of vertex or oriented edge sites and decides if a CCW oriented circle can be inscribed within the vertex sites and the left hand side of the oriented infinite lines defined by the edge sites. This test is used within the merge process to prevent the bisector $B_{L,R}$ from reversing its direction.

There are four different combinations of input site triples: three vertices (VVV), two vertices and one edge (VVE), one vertex and two edges (VEE), or three edges (EEE). Figure 10.6 shows the four combinations. The VVV case is trivial because $SCCW$ is the same as $VCCW$. We use Shewchuk's adaptive exact arithmetic vertex counterclockwise test $VCCW$ (Equation 10.3) here, and we also use it as the basic computation in the other cases as well. The remaining combi-

nations of input sites all involve at least one edge and will be treated together. $S_{i,0}$ is the starting endpoint, and $S_{i,1}$ is the terminating endpoint. If S_i is a vertex site, then the starting and terminating endpoints are simply the vertex, i.e. $S_{i,0} = S_{i,1} = S_i$. We define a point $S_{i,j}$ to be **on** an edge E if it is collinear with the edge, i.e. $VCCW\,(E_0,\ E_1,\ S_{i,j}) = 0$. And we define $S_{i,j}$ to be **to the left of** the edge E if it is to the left of the oriented infinite line, i.e. $VCCW\,(E_0,\ E_1,\ S_{i,j}) > 0$.

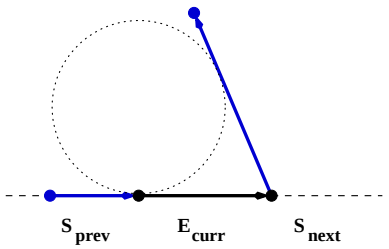


Figure 10.5: *Generalized counterclockwise rejection test*

Each edge site E_{curr} is tested against the other two sites S_{prev} and S_{next} (Figure 10.5). There are four topological cases of how the endpoints of E_{curr} can be shared with the surrounding sites.

Case 1 E_{curr} shares its endpoints with both the previous and next sites. The test is successful if one of the following conditions is satisfied.

- $S_{prev,0}$ is **on** or **to the left of** E_{curr} **and** $S_{next,1}$ is **to the left of** E_{curr}
- $S_{prev,0}$ is **to the left of** E_{curr} **and** $S_{next,1}$ is **on** or **to the left of** E_{curr}

Case 2 E_{curr} shares $E_{curr,0}$ with the previous site. This test is successful if $S_{prev,0}$ is **on** or **to the left of** E_{curr} **and** $S_{next,0}$ or $S_{next,1}$ is **to the left of** E_{curr}.

Case 3 E_{curr} shares $E_{curr,1}$ with the next site. This test is successful if $S_{next,1}$ is **on** or **to the left of** E_{curr} **and** $S_{prev,0}$ or $S_{prev,1}$ is **to the left of** E_{curr}.

Case 4 E_{curr} does not share either of its endpoints. This test is successful if $S_{prev,0}$ or $S_{prev,1}$ is **to the left of** E_{curr} **and** $S_{next,0}$ or $S_{next,1}$ is **to the left of** E_{curr}.

If a test fails for any edge, then it is not possible to inscribe a circle within the three sites.

10.6 Generalized Inscribed Circle Center

The $SCCW$ predicate is not a sufficient test of whether an inscribed circle exists. The inscribed circle center constructor routine ($SCenter$) doubles as a predicate for the existence of such a circle. $SCenter$ takes an ordered triple of input sites ($S_0,\ S_1,\ S_2$) and attempts to compute the center and radius of the circle inscribed within them. The equations used by $SCenter$ treat the finite edge segments as infinite oriented lines. The $SCCW$ trivial rejection test guarantees that the circle will always lie to the left hand side of the oriented lines. This reduces the solution space to at most two possible answers. Without the $SCCW$ filter, there could be as many as four possible solutions.

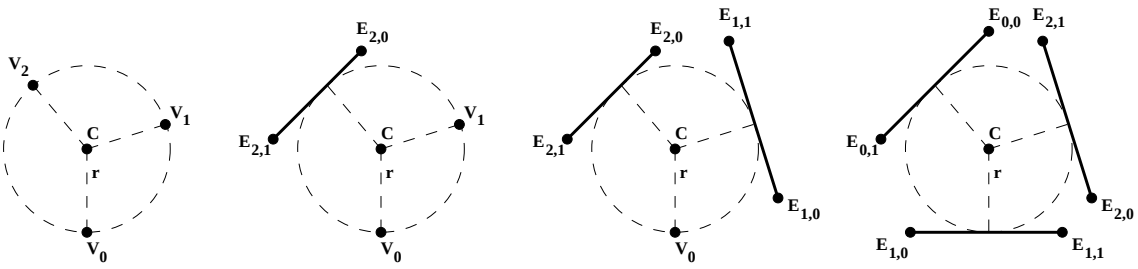


Figure 10.6: *VVV, VVE, VEE, and EEE*

As with $SCCW$, there are four major categories of combinations of input sites (Figure 10.6). These four categories are expanded into twelve cases to deal with degeneracies, see Appendix A for details. The equations for the VVE and VEE categories can have two possible answers. Some topological situations make it is possible to limit the solution space to a single answer. These situations arise when two edges share a common endpoint or when a vertex site is the endpoint of one of the edge sites. Recognizing these cases does not involve any numeric predicates. Other degeneracies arise due to geometric events like two edges being parallel to each other. These cases can be diagnosed using the $VCCW$ test. These degeneracies must be handled by special case equations because they cause the denominators of the more general equations to vanish.

The equations have been formed so that they depend on the original input values. This formulation is necessary so that exact algebra can be applied to predicates that depend on them. The equations have also been translated so that they have a local origin O. Using relative position vectors makes the computations less susceptible to floating point round-off error which allows the cheaper computions of the adaptive scheme to succeed more frequently.

There can be at most one valid center. The valid circle center, if it exists, lies in the

intersection of the Voronoi zones of the three input sites (Figure 10.2). There is no closer site to the center of a valid circle center than the three sites that inscribe it. If a candidate center lies outside of the Voronoi zones of the input sites, then it is closer to some other input site. To avoid round-off error when making this determination, the generating equation for the candidate center is substituted into a Voronoi zone test to form a predicate for each of sites. At most one center will be valid. If no answer is valid, then no circle can be inscribed within the three input sites.

10.7 Generalized In-Circle Test

The equations from $SCenter$ are also used in our generalized in-circle-test, $SInCircle$. $SInCircle$ takes three ordered sites S_0, S_1, and S_2 that define an inscribed circle and tests to see if that circle is penetrated by a fourth site S_3. $SInCircle$ compares the radius of the circle to the distance between the center of the circle and the fourth site. The distance values are never negative, so an equivalent test is to compare the squared distances. This technique will in some cases eliminate square roots from the equations. In order to use exact algebra, the equations for the local origin O, the center position C, and the radius r from Appendix A for the case specified by S_0, S_1, and S_2 are substituted into the equations below. The test can be split into two cases: when the fourth site is a vertex or when it is an edge.

10.7.1 Vertex Site

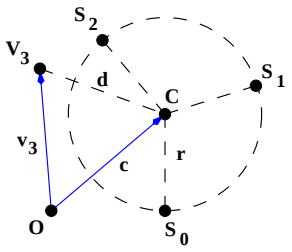


Figure 10.7: *The test that checks if the vertex V_3 lies inside the circle defined by S_0, S_1, and S_2.*

In this case, the vertex site V_3 is tested to see if it is inside the circle (Figure 10.7). The calculations are made local by subtracting the origin O of the $SCenter$ calculation from all the vertices.

$$c = C - O \tag{10.5}$$

$$v_3 = V_3 - O \tag{10.6}$$

V_3 is in the circle if d^2, the squared distance from V_3 to C, is less than r^2, the radius squared.

$$\|v_3\|^2 - 2\,(c \cdot v_3) + \|c\|^2 - r^2 < 0 \tag{10.7}$$

In all $SCenter$ cases except the EEE case (see appendix), the origin O is on the circle, so the magnitude of the position vector of the center $\|c\|$ is equal to the radius r. In these cases, the expression can be simplified by cancelling the last two terms.

$$\|v_3\|^2 - 2\,(c \cdot v_3) < 0 \tag{10.8}$$

10.7.2 Edge Site

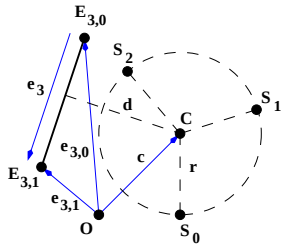


Figure 10.8: *The test that checks if the infinite line containing the edge with endpoints E_{3_0} and E_{3_1} penetrates the inside of the circle defined by S_0, S_1, and S_2.*

In this case, the edge site E_3 is tested to see if any point on it is inside the circle (Figure 10.8). The calculations are made local by subtracting the origin O of the $SCenter$ calculation from all the vertices.

$$\begin{aligned} c &= C - O \\ e_{3,0} &= E_{3,0} - O \\ e_{3,1} &= E_{3,1} - O \end{aligned} \tag{10.9}$$

The distance function of a finite edge can be broken up into 3 regions depending on whether the center of the circle is closest to the starting endpoint, the middle of the edge, or the terminating endpoint (Figure 10.9).

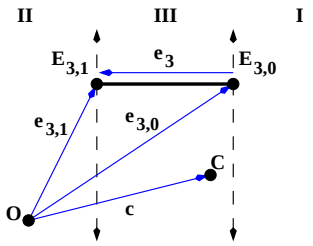


Figure 10.9: *The Voronoi zones of an edge and its two endpoints* $E_{3,0}$ *and* $E_{3,1}$.

Case I The center point C projects onto the infinite line defined by the edge before the starting vertex of the edge. The distance from the edge to C is equal to the distance from $E_{3,0}$ to C. The test for this case is as follows:

$$-e_3 \cdot (c - e_{3,0}) \geq 0 \tag{10.10}$$

If this equation holds then the vertex site in-circle-test (Equation 10.7) is performed with $V_3 = E_{3,0}$.

Case II The center point C projects onto the infinite line defined by the edge after the terminating vertex of the edge. The distance from the edge to C is equal to the distance from $E_{3,1}$ to C. The test for this case is as follows:

$$e_3 \cdot (c - e_{3,1}) \geq 0 \tag{10.11}$$

If this equation holds then the vertex site in-circle-test (Equation 10.7) is performed with $V_3 = E_{3,1}$.

Case III The center point C projects onto the middle of the edge. The distance to the edge is equal to the distance to the infinite line defined by the edge. The edge passes inside the circle if d^2, the squared distance from C to the line, is less than r^2, the radius squared. We take advantage of the comparison against zero and multiply the equation by $\|e_3\|^2$ to remove any divisions.

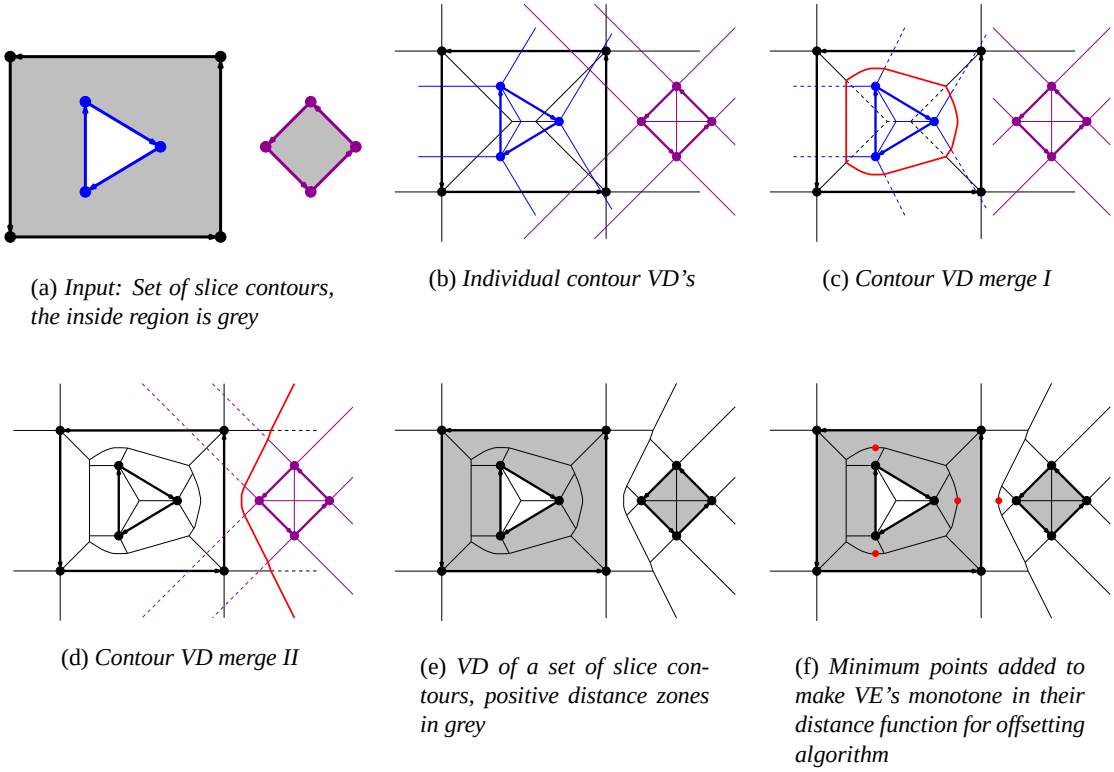
$$\left(-e_{3_y}c_x + e_{3_x}c_y + \|e_{3,0} \times e_{3,1}\|\right)^2 - \|e_3\|^2 r^2 < 0 \tag{10.12}$$

10.8 Voronoi Diagram of a Contour Set

The VD of a set of contours can be constructed by merging the individual contour VD's (Figure 10.10). Contours must be processed in order so that the sign of the overlapping distance functions match up. The ordering criterion is that a contour C_i's immediate enclosing contour must be merged into the composite VD before C_i is processed. Given that input contours are properly nested and do not intersect, contours are processed in order by their minimum y-coordinate, with the x-coordinate used to break ties, using a priority queue. The first contour taken from the priority queue will be a CCW contour and its VD will become the initial composite VD. Each of the rest of the contours will be removed from the head of the queue in turn and have their VD constructed and merged into the composite VD.

Most of the machinery needed to perform the merge of the VD of a single contour with the composite VD of a set of contours is the same as the merge algorithm used to construct the VD of a single contour. The only difference is that it is more difficult to find a common point within the two VD's to begin the merge. In the divide-and-conquer algorithm, the merge is between two paths that join at a shared non-reflex vertex. The joining vertex is a natural and easy place to begin that merge process. There are no shared vertices between the new contour and any of the contours in the composite VD, because by definition they do not intersect. The starting point is found by finding two adjacent VF's, one from the composite VD and one from the new VD. We can limit the set of VF's that are considered from the new VD to the ones that extend to infinity and are associated with a vertex site. We find the VF from the composite VD that contains the corresponding vertex site, i.e. the closest site in the composite VD. The candidate VF's of the new VD are searched until one is found that contains a point from the closest site in the composite VD.

Once two such sites are found, their VF's are scanned until a VE from each of them is intersected. At this point the mesh is in a state where the earlier merge can be applied. When all of the contours have been processed, the signed Voronoi mesh is constructed (Figure 10.10(e)).



(a) *Input: Set of slice contours, the inside region is grey*

(b) *Individual contour VD's*

(c) *Contour VD merge I*

(d) *Contour VD merge II*

(e) *VD of a set of slice contours, positive distance zones in grey*

(f) *Minimum points added to make VE's monotone in their distance function for offsetting algorithm*

Figure 10.10: *Voronoi algorithm for a set of contours*

10.9 Voronoi Diagram Clean Up

The Voronoi mesh that results from the prescribed algorithm may need to be cleaned up before they are used in downstream applications. The clean up procedure removes zero length edges that are caused by four or more co-circular vertices. It also splits any VE's that are not monotone in their z-coordinate at their minimum distance point (Figure 10.10(f)). A VE between two input edges is a portion of a straight line, so it will never need to be split. A VE between an input vertex and edge will be a parabola in 2D and in 3D. A VE edge between two input vertices looks like a straight line in 2D, but the 3D curve is actually a hyperbola. Both parabolic and hyperbolic VE's may need to be split. The minimum point is the midpoint of the line segment connecting the two vertex sites or the midpoint of the line projecting the vertex on the infinite line along the edge site. If this minimum point lies within the Voronoi region of both of the generating sites then the VE is split. In this case, a new vertex is created at the minimum point with an appropriate signed z-coordinate, and the VE is split into two z-monotone edges.

10.10 Results

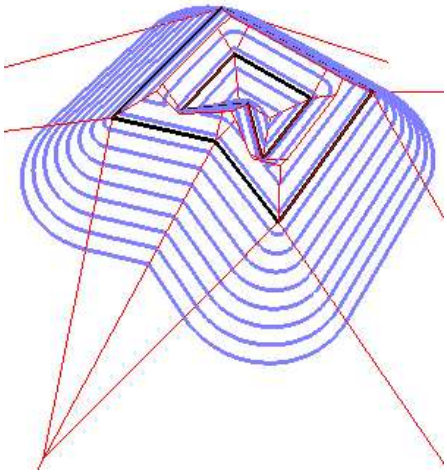
We have implemented the divide-and-conquer Voronoi diagram using our half-edge data structure. We have tested our implementation on contours created by slicing 3D polyhedrons and on a number of fractal contours (Figure 10.11). The fractal contours, including the third generation Koch island with about 2900 vertices depicted in Figure 10.11(b), exercised our implementation in a number of different ways. The recursive structure of the fractals make it possible to generate large input data sets. The regular, repetitive pattern of a fractal might suggest that after a few generations all situtations will have been exhausted making them poor test data for uncovering unexpected corner cases. On the contrary, we found that the global interactions between contour elements were extremely efficient in exposing unhandled cases. In addition, the symmetries of the fractals created many topological coincidences such as parallel edges and four or more co-circular sites which may not arise with more random input. These are the sort of coincidences that cause algorithms that assume general position to fail.

We have used the distance information in the Voronoi mesh to generate offset contours. The offsetting algorithm uses the z-monotone mesh to create offset contours by performing a graph traversal around peaks of the height field [19]. The Voronoi height field is a mesh made up of 45° angled plane and cone pieces. Figure 10.11(a) shows a 3D rendering of the Voronoi distance mesh. The VV's have been lifted along the z-axis by their signed distances from the input contours. A series of offset contours at ten incremental offset distances show the level sets of the distance function. Figure 10.11(b) shows the the Koch island with the set of offset contours at a specified distance. For this example, the construction of the VD takes a couple of seconds. Once the VD has been constructed, the user can interactively adjust the offset distance and see the offset contours generated in real time (less than $\frac{1}{30}$ of a second). Figure 10.11(c) shows the application of our offsetting algorithm to creating thin-walled FDM models. The thin-walled build used only 27% as much material as a solid build, and it took 61% of the time. The speed up was not as dramatic as the reduction in material usage because 42% of the build time was devoted to creating the support structure which is not optimized by our algorithm.

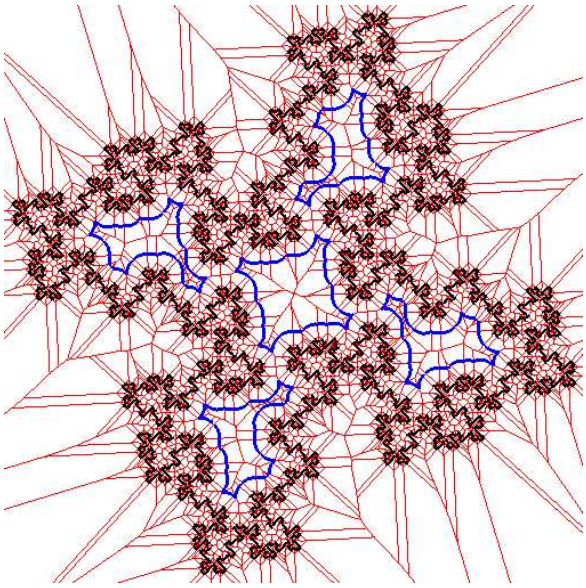
The Voronoi mesh is very useful in tool-path planning applications such as milling. After the cost of constructing the Voronoi mesh has been incurred once, many offset contours can be generated at interactive speeds. This is useful when looking for an optimal combination of cutting tools of different diameters that will mill out a pocket in a minimal time.

10.11 Conclusion

We have described a robust divide-and-conquer algorithm for constructing the Voronoi diagram of a set of oriented polygonal contours. We have used the resulting Voronoi mesh to create offset contours. The 2D offsetting routine can be used in applications to speed up layered manufacturing [19], generate contour parallel tool paths, and to minimize machining time in pocket milling.



(a) *Voronoi mesh height fi eld of nested contours, offset contours act as topographical contours*



(b) *Voronoi diagram of the third order Koch island fractal contour and the offset contours with* $d = 0.05$

(c) *Thin-walled cow model being built on an FDM machine*

Figure 10.11: *Results from the Voronoi algorithm*

Chapter 11

Conclusion

In this thesis, we have shown that new advances in subdivision surfaces are making them an ideal surface representation for CAD and manufacturing. Subdivision surfaces of arbitrary mesh topology and genus can be defined by watertight 2-manifold meshes, and sharp tagging schemes allow the designer to intentionally break the continuity and introduce crease features. This gives designers the freedom to define interesting geometries that fully enclose physical volumes. The control meshes stay watertight throughout the subdivision process, and can also be adaptively tessellated in a watertight manner. New exact evaluation methods provide a global per face (u, v) parameterization, that makes the interface to subdivision surfaces analogous to patch based representations such as NURBS. We have shown a new algorithm to perform this exact evaluation near sharp features of piecewise smooth Loop and Catmull-Clark surfaces, that allows the surface to be queried for position, tangents and normal, and principle curvature directions and values at arbitrary parameter values. The differential geometry of the surface can be used by manufacturing process planning algorithms to generate and optimize tool paths for automatically building physical representations of computer models. These automated process planning algorithms must be numerically robust. We have described our robust implementation of the generalized Voronoi diagram of sets of polygonal contours, which has been used to generate thin-walled representations of objects for layered manufacturing [19]. The Voronoi diagram is used to generate 2D offset contours that are combined between layers to make a conservative approximation to the 3D offset surface. Creating robust geometric algorithms and CAD software is a difficult and interesting field of study which still requires much future investigation.

Appendix A

Generalized Voronoi Predicates

A.1 Introduction

This appendix provides formulas to calculate the center C and radius r of the inscribed circle to three ordered input sites for the twelve cases need for generalized Voronoi diagram of polygonal contours. These expressions are then plugged into the predicates described in Section 10.7 for testing whether a fourth site penetrates this inscribed circle. One of the input vertices is selected as a local origin O in order to make the calculations less prone to floating point round-off error.

A.2 Edge-Edge Intersection

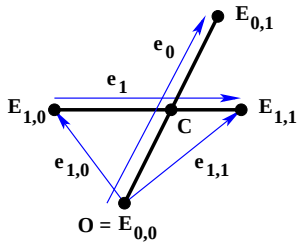


Figure A.1: *Edge-Edge intersection*

The following computes the intersection point of two edges, if it exists. This could be used to build a more general Voronoi algorithm that allows for intersecting line segments. The solution to the arrangement of intersecting line segments in the plane is a numerically challenging problem on its own. We have avoided addressing this problem by assuming that the input contours

do not intersect.

$$\begin{aligned} O =& E_{0,0} \\ C_x =& O_x + \frac{\|e_{1,0} \times e_{1,1}\| e_{0_x}}{\|e_0 \times e_1\|} \\ C_y =& O_y + \frac{\|e_{1,0} \times e_{1,1}\| e_{0_y}}{\|e_0 \times e_1\|} \end{aligned} \tag{A.1}$$

The existence of the intersection of two line segments reduces to four applications of the $VCCW$ test (Equation 10.3). This test determines which side of a line segment a third vertex is on. Two line segments intersect only if for each line segment the two end points of the other segment lie on different sides of it.

A.3 Site Closest to a Vertex

This predicate tests which of two other sites is closest to a vertex V_0. V_0 is chosen as the local origin O to improve the floating point accuracy. This predicate is used to perform point location in a Voronoi mesh which is necessary when searching for a seed point to begin the merge between two disconnected contour VD's.

$$\begin{aligned} O =& V_0 \\ 0 \leq& (e_1 \cdot e_{1,0}) \Rightarrow v_1 \leftarrow e_{1,0} \\ 0 \geq& (e_1 \cdot e_{1,1}) \Rightarrow v_1 \leftarrow e_{1,1} \\ VV : 0 >& \|v_1\|^2 - \|v_2\|^2 \\ VE : 0 >& \frac{\|v_1\|^2 \|e_2\|^2 - \|e_{2,0} \times e_{2,1}\|^2}{\|e_2\|^2} \\ EE : 0 >& \frac{\|e_2\|^2 \|e_{1,0} \times e_{1,1}\|^2 - \|e_1\|^2 \|e_{2,0} \times e_{2,1}\|^2}{\|e_1\|^2 \|e_2\|^2} \end{aligned} \tag{A.2}$$

A.4 VVV

In the VVV case, all three sites are vertices (Figure A.2). The equations for this case are taken from Shewchuk [26]. We chose the origin to be V_0. The following are the formulas for the

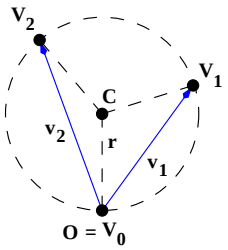


Figure A.2: *VVV*

circle center C and its radius r.

$$\begin{aligned} O =& V_0 \\ C_x =& O_x - \frac{\|v_2\|^2 v_{1_y} - \|v_1\|^2 v_{2_y}}{2\|v_1 \times v_2\|} \\ C_y =& O_y + \frac{\|v_2\|^2 v_{1_x} - \|v_1\|^2 v_{2_x}}{2\|v_1 \times v_2\|} \\ r =& \frac{\|v_1\|\|v_2\|\|v_2 - v_1\|}{2\|v_1 \times v_2\|} \end{aligned} \tag{A.3}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the circle.

$$\begin{aligned} 0 \leq& \frac{2\|v_1 \times v_2\| \left(e_3 \cdot e_{3,0}\right) - \left(\|v_1\|^2 \|e_3 \times v_2\| - \|v_2\|^2 \|e_3 \times v_1\|\right)}{2\|v_1 \times v_2\|} \\ 0 \geq& \frac{2\|v_1 \times v_2\| \left(e_3 \cdot e_{3,1}\right) - \left(\|v_1\|^2 \|e_3 \times v_2\| - \|v_2\|^2 \|e_3 \times v_1\|\right)}{2\|v_1 \times v_2\|} \\ 0 <& \frac{\|v_1\|^2 \|v_2 \times v_3\| + \|v_2\|^2 \|v_3 \times v_1\| + \|v_3\|^2 \|v_1 \times v_2\|}{\|v_1 \times v_2\|} \\ 0 <& \frac{\left(\begin{array}{c} \left(\|v_1\|^4 \left(e_3 \cdot v_2\right)^2 + \|v_2\|^4 \left(e_3 \cdot v_1\right)^2\right) \\ - \left(\|v_1\|^2 \|v_2\|^2 \left(\|v_2 - v_1\|^2 \|e_3\|^2 + 2\left(e_3 \cdot v_1\right)\left(e_3 \cdot v_2\right)\right)\right) \\ + \left(4\|v_1 \times v_2\|^2 \|e_{3,0} \times e_{3,1}\|^2\right) \\ + \left(4\|v_1 \times v_2\| \|e_{3,0} \times e_{3,1}\| \left(\|v_2\|^2 \left(e_3 \cdot v_1\right) - \|v_1\|^2 \left(e_3 \cdot v_2\right)\right)\right) \end{array}\right)}{4\|e_3\|^2 \|v_1 \times v_2\|^2} \end{aligned} \tag{A.4}$$

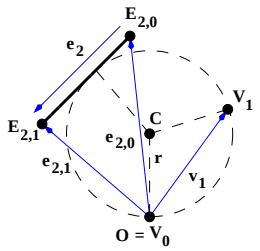


Figure A.3: *VVE*

A.5 VVE

In the VVE case, there are two vertex sites and one edge site (Figure A.3). This is the general form, and it can have two candidate solutions. This can be reduced to one candidate equation because the center must lie to the left of the line connecting V_0 to V_1 for the circle to touch the sites in the correct order. We chose the origin to be V_0. The following are the formulas for the candidate circle center C and its radius r.

$$\begin{aligned}
O =& V_0 \\
disc =& \|v_1\|^2 \|e_2\|^2 \|e_{2,0} \times e_{2,1}\| \left(\|e_{2,0} \times e_{2,1}\| + \|e_2 \times v_1\|\right) \\
den =& 2\|e_2 \times v_1\|^2 \\
C_x =& O_x - \frac{\left(\|v_1\|^2 \|e_2 \times v_1\| e_{2_y} + 2\|e_{2,0} \times e_{2,1}\| \left(e_2 \cdot v_1\right) v_{1_y}\right) - 2v_{1_y}\sqrt{disc}}{den} \\
C_y =& O_y + \frac{\left(\|v_1\|^2 \|e_2 \times v_1\| e_{2_x} + 2\|e_{2,0} \times e_{2,1}\| \left(e_2 \cdot v_1\right) v_{1_x}\right) + 2v_{1_x}\sqrt{disc}}{den} \\
disc_r =& \|v_1\|^2 \|e_{2,0} \times e_{2,1}\| \left(\|e_{2,0} \times e_{2,1}\| + \|e_2 \times v_1\|\right) \\
r =& \frac{\|v_1\|^2 \|e_2\| \left(2\|e_{2,0} \times e_{2,1}\| + \|e_2 \times v_1\|\right) + 2\left(e_2 \cdot v_1\right)\sqrt{disc_r}}{den}
\end{aligned} \tag{A.5}$$

The candidate center must satisfy the following two predicates for it to project into the body of the edge segment.

$$0 \geq \frac{\left(\|e_2 \times v_1\| \left(e_2 \cdot e_{2,0}\right) + \|e_{2,0} \times e_{2,1}\| \left(e_2 \cdot v_1\right) + \sqrt{disc}\right)}{\|e_2 \times v_1\|} \tag{A.6}$$

$$0 \leq \frac{\left(\|e_2 \times v_1\| \left(e_2 \cdot e_{2,1}\right) + \|e_{2,0} \times e_{2,1}\| \left(e_2 \cdot v_1\right) + \sqrt{disc}\right)}{\|e_2 \times v_1\|} \tag{A.7}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex

V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the circle.

$$
\begin{aligned}
0 \leq & \frac{\left(\begin{array}{c} 2\|e_2 \times v_1\|^2 (e_3 \cdot e_{3,0}) - \|e_2 \times v_1\|\|e_2 \times e_3\|\|v_1\|^2 \\ +2\|e_3 \times v_1\|\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) + 2\|e_3 \times v_1\|\sqrt{disc} \end{array}\right)}{2\|e_2 \times v_1\|^2} \\
0 \geq & \frac{\left(\begin{array}{c} 2\|e_2 \times v_1\|^2 (e_3 \cdot e_{3,1}) - \|e_2 \times v_1\|\|e_2 \times e_3\|\|v_1\|^2 \\ +2\|e_3 \times v_1\|\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) + 2\|e_3 \times v_1\|\sqrt{disc} \end{array}\right)}{2\|e_2 \times v_1\|^2} \\
0 < & \frac{\left(\begin{array}{c} \|e_2 \times v_1\| \left(\|v_3\|^2\|e_2 \times v_1\| - \|v_1\|^2\|e_2 \times v_3\|\right) \\ -2\|v_1 \times v_3\| \left(\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) + \sqrt{disc}\right) \end{array}\right)}{\|e_2 \times v_1\|^2} \\
a = & \|v_1\|^2\|e_2 \times v_1\| (e_2 \cdot e_3) + 2 (v_1 \cdot e_3) \left(\|e_{2,0} \times e_{2,1}\| (v_1 \cdot e_2) + \sqrt{disc}\right) \\
b = & \|v_1\|^2\|e_2\| (2\|e_{2,0} \times e_{2,1}\| + \|e_2 \times v_1\|) + 2 (v_1 \cdot e_2) \sqrt{disc_r} \\
0 < & \frac{a^2 + 4\|e_{3,0} \times e_{3,1}\|\|e_2 \times v_1\|^2 a + 4\|e_{3,0} \times e_{3,1}\|^2\|e_2 \times v_1\|^4 - \|e_3\|^2 b^2}{4\|e_2 \times v_1\|^4\|e_3\|^2}
\end{aligned}
\tag{A.8}
$$

A.5.1 VVE Antiparallel

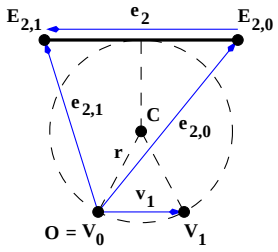


Figure A.4: *VVE Antiparallel*

The VVE antiparallel case arises when the two vertex sites lie on a line parallel to the edge site (Figure A.4). This case is diagnosed using a numerical test. There is exactly one candidate circle center in this case. We chose the origin to be V_0. The following are the formulas for the

candidate circle center C and its radius r.

$$\begin{aligned} O =& V_0 \\ C_x =& O_x + \frac{v_{1_x}}{2} - \frac{\left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) v_{1_y}}{-8\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1)} \\ C_y =& O_y + \frac{v_{1_y}}{2} + \frac{\left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) v_{1_x}}{-8\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1)} \\ r =& \frac{4\|e_{2,0} \times e_{2,1}\|^2 + \|e_2\|^2\|v_1\|^2}{8\|e_{2,0} \times e_{2,1}\|\|e_2\|} \end{aligned} \tag{A.9}$$

The candidate center must satisfy the following two predicates for it to project into the body of the edge segment.

$$0 \geq 2(e_2 \cdot e_{2,0}) - (e_2 \cdot v_1) \tag{A.10}$$

$$0 \leq 2(e_2 \cdot e_{2,1}) - (e_2 \cdot v_1) \tag{A.11}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the circle.

$$\begin{aligned} 0 \leq& \frac{-4\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) (2(e_3 \cdot e_{3,0}) - (e_3 \cdot v_1)) + \left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) \|e_3 \times v_1\|}{-8\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1)} \\ 0 \geq& \frac{-4\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) (2(e_3 \cdot e_{3,1}) - (e_3 \cdot v_1)) + \left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) \|e_3 \times v_1\|}{-8\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1)} \\ 0 <& \frac{-4\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) \left(\|v_3\|^2 - (v_1 \cdot v_3)\right) - \left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) \|v_1 \times v_3\|}{-4\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1)} \\ 0 <& \frac{\begin{pmatrix} 64\|e_{2,0} \times e_{2,1}\|^2 (e_2 \cdot v_1)^2 \|e_{3,0} \times e_{3,1}\| (\|e_{3,0} \times e_{3,1}\| + \|e_3 \times v_1\|) \\ -16\|e_{2,0} \times e_{2,1}\|^2 (e_2 \cdot v_1)^2 (e_3 \cdot v_1)^2 \\ -16\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) (e_3 \cdot v_1) \left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) \|e_{3,0} \times e_{3,1}\| \\ -8\|e_{2,0} \times e_{2,1}\| (e_2 \cdot v_1) (e_3 \cdot v_1) \left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right) \|e_3 \times v_1\| \\ -\|e_3 \times v_1\|^2 \left(4\|e_{2,0} \times e_{2,1}\|^2 - (e_2 \cdot v_1)^2\right)^2 \end{pmatrix}}{64\|e_3\|^2\|e_{2,0} \times e_{2,1}\|^2 (e_2 \cdot v_1)^2} \end{aligned} \tag{A.12}$$

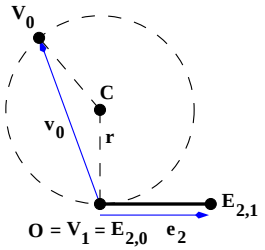


Figure A.5: *VvE*

A.5.2 VvE

The VvE case arises when a vertex site is the starting endpoint of the edge site (Figure A.5). This case is diagnosed using a topological test. There is only one candidate solution, and it is always in the zone of influence of the edge site. We chose the origin to be $E_{2,0}$ which is the shared end point of the edge site. The following are the formulas for the candidate circle center C and its radius r.

$$\begin{aligned} O =& E_{2,0} \\ C_x =& O_x - \frac{\|v_0\|^2 e_{2_y}}{2\|e_2 \times v_0\|} \\ C_y =& O_y + \frac{\|v_0\|^2 e_{2_x}}{2\|e_2 \times v_0\|} \\ r =& \frac{\|v_0\|^2 \|e_2\|}{2\|e_2 \times v_0\|} \end{aligned} \tag{A.13}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the circle.

$$\begin{aligned} 0 \leq& \frac{2\|e_2 \times v_0\| \, (e_3 \cdot e_{3,0}) - \|v_0\|^2 \|e_2 \times e_3\|}{2\|e_2 \times v_0\|} \\ 0 \geq& \frac{2\|e_2 \times v_0\| \, (e_3 \cdot e_{3,1}) - \|v_0\|^2 \|e_2 \times e_3\|}{2\|e_2 \times v_0\|} \\ 0 <& \frac{\|v_3\|^2 \|e_2 \times v_0\| - \|v_0\|^2 \|e_2 \times v_3\|}{\|e_2 \times v_0\|} \\ 0 <& \frac{4\|e_2 \times v_0\|^2 \|e_{3,0} \times e_{3,1}\|^2 + 4\|e_2 \times v_0\| \|e_{3,0} \times e_{3,1}\| \|v_0\|^2 \, (e_2 \cdot e_3) - \|v_0\|^4 \|e_2 \times e_3\|^2}{4\|e_3\|^2 \|e_2 \times v_0\|^2} \end{aligned} \tag{A.14}$$

A.5.3 EvV

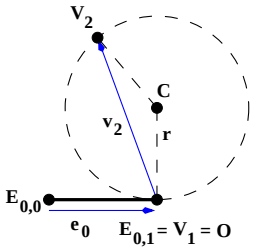


Figure A.6: *EvV*

The EvV case arises when a vertex site is the terminating endpoint of the edge site (Figure A.6). The equations are closely related to the VvE case. This case is diagnosed using a topological test. There is only one candidate solution, and it is always in the zone of influence of the edge site. We chose the origin to be $E_{2,0}$ which is the shared end point of the edge site. The following are the formulas for the candidate circle center C and its radius r.

$$\begin{aligned} O =& E_{0,1} \\ C_x =& O_x - \frac{\|v_2\|^2 e_{0_y}}{2\|e_0 \times v_2\|} \\ C_y =& O_y + \frac{\|v_2\|^2 e_{0_x}}{2\|e_0 \times v_2\|} \\ r =& \frac{\|v_2\|^2 \|e_0\|}{2\|e_0 \times v_2\|} \end{aligned} \tag{A.15}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the circle.

$$\begin{aligned} 0 \leq& \frac{2\|e_0 \times v_2\| (e_3 \cdot e_{3,0}) - \|v_2\|^2 \|e_0 \times e_3\|}{2\|e_0 \times v_2\|} \\ 0 \geq& \frac{2\|e_0 \times v_2\| (e_3 \cdot e_{3,1}) - \|v_2\|^2 \|e_0 \times e_3\|}{2\|e_0 \times v_2\|} \\ 0 <& \frac{\|v_3\|^2 \|e_0 \times v_2\| - \|v_2\|^2 \|e_0 \times v_3\|}{\|e_0 \times v_2\|} \\ 0 <& \frac{4\|e_0 \times v_2\|^2 \|e_{3,0} \times e_{3,1}\|^2 + 4\|e_0 \times v_2\| \|e_{3,0} \times e_{3,1}\| \|v_2\|^2 (e_0 \cdot e_3) - \|v_2\|^4 \|e_0 \times e_3\|^2}{4\|e_3\|^2 \|e_0 \times v_2\|^2} \end{aligned} \tag{A.16}$$

A.6 VEE

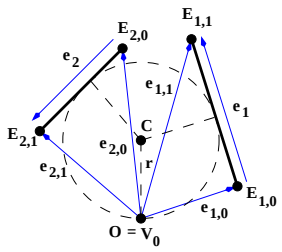


Figure A.7: *VEE*

There is one vertex site and two edge sites (Figure A.7). This the general form, and it can have two candidate solutions. We chose the origin to be V_0. The following are the formulas for the candidate circle center C and its radius r.

$$
\begin{aligned}
O =& V_0 \\
disc =& 2\|e_{1,0} \times e_{1,1}\|\|e_{2,0} \times e_{2,1}\| \left(\|e_1\|\|e_2\| + (e_1 \cdot e_2)\right) \\
den =& \|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right) \\
s =& \left\{ \begin{array}{ccc} 1 & if & \|e_1 \times e_2\| < 0 \\ -1 & if & \|e_1 \times e_2\| > 0 \end{array} \right\} \\
C_x =& O_x + \frac{\left(\begin{array}{c} \left(\|e_1\|^2 e_{2_x} - (e_1 \cdot e_2)\, e_{1_x}\right) \|e_{2,0} \times e_{2,1}\| \\ + \left((e_1 \cdot e_2)\, e_{2_x} - \|e_2\|^2 e_{1_x}\right) \|e_{1,0} \times e_{1,1}\| \\ + s \left(\|e_1\| e_{2_x} - \|e_2\| e_{1_x}\right) \sqrt{disc} \end{array} \right)}{den} \\
C_y =& O_y + \frac{\left(\begin{array}{c} \left(\|e_1\|^2 e_{2_y} - (e_1 \cdot e_2)\, e_{1_y}\right) \|e_{2,0} \times e_{2,1}\| \\ + \left((e_1 \cdot e_2)\, e_{2_y} - \|e_2\|^2 e_{1_y}\right) \|e_{1,0} \times e_{1,1}\| \\ + s \left(\|e_1\| e_{2_y} - \|e_2\| e_{1_y}\right) \sqrt{disc} \end{array} \right)}{den} \qquad \text{(A.17)} \\
r =& \frac{\|e_1\|\|e_{2,0} \times e_{2,1}\| + \|e_2\|\|e_{1,0} \times e_{1,1}\| + s\sqrt{disc}}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \qquad \text{(A.18)}
\end{aligned}
$$

The candidate center must satisfy the following four predicates for it to project into the body of both

of the edge segments.

$$0 \geq \frac{\|e_1 \times e_2\| (e_1 \cdot e_{1,0}) + \|e_{1,0} \times e_{1,1}\| (\|e_1\|\|e_2\| + (e_1 \cdot e_2)) + s\|e_1\|\sqrt{disc}}{\|e_1 \times e_2\|} \tag{A.19}$$

$$0 \leq \frac{\|e_1 \times e_2\| (e_1 \cdot e_{1,1}) + \|e_{1,0} \times e_{1,1}\| (\|e_1\|\|e_2\| + (e_1 \cdot e_2)) + s\|e_1\|\sqrt{disc}}{\|e_1 \times e_2\|} \tag{A.20}$$

$$0 \geq \frac{\|e_1 \times e_2\| (e_2 \cdot e_{2,0}) - \|e_{2,0} \times e_{2,1}\| (\|e_1\|\|e_2\| + (e_1 \cdot e_2)) - s\|e_2\|\sqrt{disc}}{\|e_1 \times e_2\|} \tag{A.21}$$

$$0 \leq \frac{\|e_1 \times e_2\| (e_2 \cdot e_{2,1}) - \|e_{2,0} \times e_{2,1}\| (\|e_1\|\|e_2\| + (e_1 \cdot e_2)) - s\|e_2\|\sqrt{disc}}{\|e_1 \times e_2\|} \tag{A.22}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the

circle.

$$
\begin{aligned}
0 \leq & \frac{\left(\begin{array}{c}
\|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)(e_3 \cdot e_{3,0}) \\
+ \left(\|e_2\|^2 (e_1 \cdot e_3) - (e_1 \cdot e_2)(e_2 \cdot e_3)\right) \|e_{1,0} \times e_{1,1}\| \\
- \left(\|e_1\|^2 (e_2 \cdot e_3) - (e_1 \cdot e_2)(e_1 \cdot e_3)\right) \|e_{2,0} \times e_{2,1}\| \\
+ s \left(\|e_2\| (e_1 \cdot e_3) - \|e_1\| (e_2 \cdot e_3)\right) \sqrt{disc}
\end{array}\right)}{\|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)} \\
0 \geq & \frac{\left(\begin{array}{c}
\|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)(e_3 \cdot e_{3,1}) \\
+ \left(\|e_2\|^2 (e_1 \cdot e_3) - (e_1 \cdot e_2)(e_2 \cdot e_3)\right) \|e_{1,0} \times e_{1,1}\| \\
- \left(\|e_1\|^2 (e_2 \cdot e_3) - (e_1 \cdot e_2)(e_1 \cdot e_3)\right) \|e_{2,0} \times e_{2,1}\| \\
+ s \left(\|e_2\| (e_1 \cdot e_3) - \|e_1\| (e_2 \cdot e_3)\right) \sqrt{disc}
\end{array}\right)}{\|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)} \\
0 < & \frac{\left(\begin{array}{c}
\|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right) \|v_3\|^2 \\
+2 \left(\|e_2\|^2 (e_1 \cdot v_3) - (e_1 \cdot e_2)(e_2 \cdot v_3)\right) \|e_{1,0} \times e_{1,1}\| \\
-2 \left(\|e_1\|^2 (e_2 \cdot v_3) - (e_1 \cdot e_2)(e_1 \cdot v_3)\right) \|e_{2,0} \times e_{2,1}\| \\
+ s2 \left(\|e_2\| (e_1 \cdot v_3) - \|e_1\| (e_2 \cdot v_3)\right) \sqrt{disc}
\end{array}\right)}{\|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)} \\
a = & \left(\begin{array}{c}
\left((e_1 \cdot e_2) \|e_1 \times e_3\| - \|e_1\|^2 \|e_2 \times e_3\|\right) \|e_{2,0} \times e_{2,1}\| \\
+ \left(\left(\|e_2\|^2 \|e_1 \times e_3\| - e_1 \cdot e_2\right) \|e_2 \times e_3\|\right) \|e_{1,0} \times e_{1,1}\| \\
+ s \left(\|e_2\| \|e_1 \times e_3\| - \|e_1\| \|e_2 \times e_3\|\right) \sqrt{disc}
\end{array}\right) \\
b = & \|e_1\| \|e_{2,0} \times e_{2,1}\| + \|e_2\| \|e_{1,0} \times e_{1,1}\| + s\sqrt{disc} \\
0 < & \frac{\left(\begin{array}{c}
a^2 \\
+2 \|e_{3,0} \times e_{3,1}\| \|e_1 \times e_2\| \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right) a \\
+ \|e_{3,0} \times e_{3,1}\|^2 \|e_1 \times e_2\|^2 \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)^2 \\
- \|e_3\|^2 \|e_1 \times e_2\|^2 b
\end{array}\right)}{\|e_3\|^2 \|e_1 \times e_2\|^2 \left(\|e_1\|\|e_2\| - (e_1 \cdot e_2)\right)^2}
\end{aligned}
\tag{A.23}
$$

A.6.1 VEE Antiparallel

The VEE antiparallel case arises when the edge sites are antiparallel to each other (Figure A.8). This case is diagnosed using a numerical test. We chose the origin to be V_0. The following

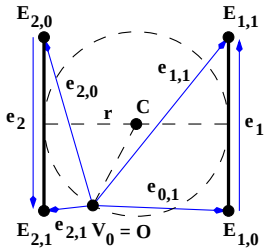


Figure A.8: *VEE Antiparallel*

are the formulas for the candidate circle center C and its radius r.

$$\begin{aligned} O =& V_0 \\ disc =& \|e_1 \times e_{2,0}\| \|e_{1,0} \times e_{1,1}\| \\ C_x =& O_x + \frac{-\left(\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|\right) e_{1_y} + 2e_{1_x}\sqrt{disc}}{2\|e_1\|^2} \\ C_y =& O_y + \frac{\left(\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|\right) e_{1_x} + 2e_{1_y}\sqrt{disc}}{2\|e_1\|^2} \\ r =& \frac{\|e_1 \times e_{2,0}\| + \|e_{1,0} \times e_{1,1}\|}{2\|e_1\|} \end{aligned} \tag{A.24}$$

The candidate center must satisfy the following two predicates for it to project into the body of both of the edge segments.

$$0 \geq (e_1 \cdot e_{1,0}) - \sqrt{\|e_1 \times e_{2,0}\| \|e_{1,0} \times e_{1,1}\|} \tag{A.25}$$

$$0 \leq (e_1 \cdot e_{1,1}) - \sqrt{\|e_1 \times e_{2,0}\| \|e_{1,0} \times e_{1,1}\|} \tag{A.26}$$

$$0 \geq (e_2 \cdot e_{2,0}) + \sqrt{\|e_2 \times e_{1,0}\| \|e_{2,0} \times e_{2,1}\|} \tag{A.27}$$

$$0 \leq (e_2 \cdot e_{2,1}) + \sqrt{\|e_2 \times e_{1,0}\| \|e_{2,0} \times e_{2,1}\|} \tag{A.28}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the

circle.

$$
\begin{aligned}
0 \leq & \frac{2\|e_1\|^2 (e_3 \cdot e_{3,0}) - (\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|) \|e_1 \times e_3\| - 2 (e_1 \cdot e_3) \sqrt{disc}}{2\|e_1\|^2} \\
0 \geq & \frac{2\|e_1\|^2 (e_3 \cdot e_{3,1}) - (\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|) \|e_1 \times e_3\| - 2 (e_1 \cdot e_3) \sqrt{disc}}{2\|e_1\|^2} \\
0 < & \frac{\|e_1\|^2\|v_3\|^2 - (\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|) \|e_1 \times v_3\| - 2 (e_1 \cdot v_3) \sqrt{disc}}{\|e_1\|^2} \\
0 < & \frac{\left(\begin{array}{c} 4\|e_1\|^2\|e_{3,0} \times e_{3,1}\| \left(\|e_1\|^2\|e_{3,0} \times e_{3,1}\| + (e_1 \cdot e_3) (\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|) \right) \\ -4 (e_1 \cdot e_3)^2 \|e_1 \times e_{2,0}\|\|e_{1,0} \times e_{1,1}\| \\ -\|e_1 \times e_3\|^2 (\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|)^2 \\ +4\|e_1 \times e_3\|\sqrt{disc} \left(2\|e_1\|^2\|e_{3,0} \times e_{3,1}\| + (e_1 \cdot e_3) (\|e_1 \times e_{2,0}\| - \|e_{1,0} \times e_{1,1}\|) \right) \end{array} \right)}{4\|e_1\|^4 (\|e_1\|\|e_2\| - (e_1 \cdot e_2))^2}
\end{aligned}
\tag{A.29}
$$

A.6.2 VEvE Parallel

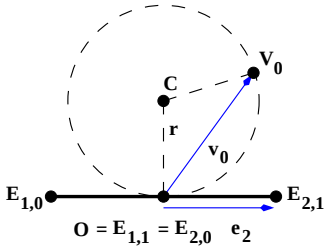


Figure A.9: *VEvE Parallel*

The $VEvE$ parallel case arises when the two edge sites share a common vertex and are collinear (Figure A.9). This case is diagnosed using topological and numerical tests. It is equivalent to VvE with $V_1 = E_{1,1}$.

A.6.3 vEE

The vEE case arises when the vertex site is the starting endpoint of the subsequent edge site (Figure A.10). This case is diagnosed using a topological test. We chose the origin to be the shared vertex $E_{1,0}$. The following are the formulas for the candidate circle center C and its radius

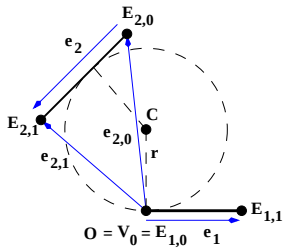


Figure A.10: *vEE*

r.

$$\begin{aligned} O =& E_{1,0} \\ C_x =& O_x - \frac{\|e_{2,0} \times e_{2,1}\| e_{1_y}}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \\ C_y =& O_y + \frac{\|e_{2,0} \times e_{2,1}\| e_{1_x}}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \\ r =& \frac{\|e_1\|\|e_{2,0} \times e_{2,1}\|}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \end{aligned} \tag{A.30}$$

The candidate center must satisfy the following two predicates for it to project into the body of the edge segment E_2.

$$0 \geq \frac{(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,(e_2 \cdot e_{2,0}) - \|e_{2,0} \times e_{2,1}\|\|e_1 \times e_2\|}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \tag{A.31}$$

$$0 \leq \frac{(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,(e_2 \cdot e_{2,1}) - \|e_{2,0} \times e_{2,1}\|\|e_1 \times e_2\|}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \tag{A.32}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the

circle.

$$
\begin{aligned}
0 \leq & \frac{(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,(e_3 \cdot e_{3,0}) - \|e_{2,0} \times e_{2,1}\|\|e_1 \times e_3\|}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \\
0 \geq & \frac{(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,(e_3 \cdot e_{3,1}) - \|e_{2,0} \times e_{2,1}\|\|e_1 \times e_3\|}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \\
0 < & \frac{(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,\|v_3\|^2 - 2\|e_{2,0} \times e_{2,1}\|\|e_1 \times v_3\|}{\|e_1\|\|e_2\| - (e_1 \cdot e_2)} \\
0 < & \frac{\left(\begin{array}{c} (\|e_1\|\|e_2\| - (e_1 \cdot e_2))^2\,\|e_{3,0} \times e_{3,1}\|^2 \\ +2\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,\|e_{3,0} \times e_{3,1}\|\|e_{2,0} \times e_{2,1}\|\,(e_1 \cdot e_3) \\ -\|e_{2,0} \times e_{2,1}\|^2\|e_1 \times e_3\|^2 \end{array}\right)}{\|e_3\|^2\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))^2}
\end{aligned}
\tag{A.33}
$$

A.6.4 EEv

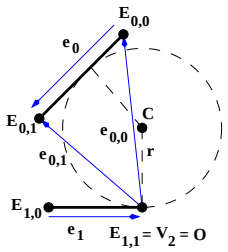


Figure A.11: *EEv*

The EEv case arises when the vertex site is the terminating endpoint of the previous edge site (Figure A.11). The equations are closely related to the vEE case. This case is diagnosed using a topological test. We chose the origin to be the shared vertex $E_{1,1}$. The following are the formulas for the candidate circle center C and its radius r.

$$
\begin{aligned}
O =& E_{1,1} \\
C_x =& O_x - \frac{\|e_{0,0} \times e_{0,1}\|e_{1_y}}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \\
C_y =& O_y + \frac{\|e_{0,0} \times e_{0,1}\|e_{1_x}}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \\
r =& \frac{\|e_1\|\|e_{0,0} \times e_{0,1}\|}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)}
\end{aligned}
\tag{A.34}
$$

The candidate center must satisfy the following two predicates for it to project into the body of the

edge segment E_0.

$$0 \geq \frac{(\|e_0\|\|e_1\| - (e_0 \cdot e_1))(e_0 \cdot e_{0,0}) + \|e_{0,0} \times e_{0,1}\|\|e_0 \times e_1\|}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \tag{A.35}$$

$$0 \leq \frac{(\|e_0\|\|e_1\| - (e_0 \cdot e_1))(e_0 \cdot e_{0,1}) + \|e_{0,0} \times e_{0,1}\|\|e_0 \times e_1\|}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \tag{A.36}$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the circle.

$$\begin{aligned}
0 \leq & \frac{(\|e_0\|\|e_1\| - (e_0 \cdot e_1))(e_3 \cdot e_{3,0}) - \|e_{0,0} \times e_{0,1}\|\|e_1 \times e_3\|}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \\
0 \geq & \frac{(\|e_0\|\|e_1\| - (e_0 \cdot e_1))(e_3 \cdot e_{3,1}) - \|e_{0,0} \times e_{0,1}\|\|e_1 \times e_3\|}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \\
0 < & \frac{(\|e_0\|\|e_1\| - (e_0 \cdot e_1))\|v_3\|^2 - 2\|e_{0,0} \times e_{0,1}\|\|e_1 \times v_3\|}{\|e_0\|\|e_1\| - (e_0 \cdot e_1)} \\
0 < & \frac{\left(\begin{array}{c} (\|e_0\|\|e_1\| - (e_0 \cdot e_1))^2 \|e_{3,0} \times e_{3,1}\|^2 \\ +2(\|e_0\|\|e_1\| - (e_0 \cdot e_1))\|e_{3,0} \times e_{3,1}\|\|e_{0,0} \times e_{0,1}\|(e_1 \cdot e_3) \\ -\|e_{0,0} \times e_{0,1}\|^2\|e_1 \times e_3\|^2 \end{array} \right)}{\|e_3\|^2(\|e_0\|\|e_1\| - (e_0 \cdot e_1))^2}
\end{aligned} \tag{A.37}$$

A.7 EEE

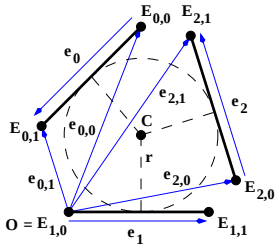


Figure A.12: *EEE*

In the EEE case, there are three edge sites (Figure A.12). This the general form.

$$
\begin{aligned}
O =& E_{1,0} \\
den =& \|e_1 \times e_2\| \|e_0\| + \|e_2 \times e_0\| \|e_1\| + \|e_0 \times e_1\| \|e_2\| \\
C_x =& O_x + \frac{\left(\|e_1\| e_{2_x} - \|e_2\| e_{1_x}\right) \|e_{0,0} \times e_{0,1}\| + \left(\|e_0\| e_{1_x} - \|e_1\| e_{0_x}\right) \|e_{2,0} \times e_{2,1}\|}{den} \\
C_y =& O_y + \frac{\left(\|e_1\| e_{2_y} - \|e_2\| e_{1_y}\right) \|e_{0,0} \times e_{0,1}\| + \left(\|e_0\| e_{1_y} - \|e_1\| e_{0_y}\right) \|e_{2,0} \times e_{2,1}\|}{den} \\
r =& \frac{\|e_0 \times e_1\| \|e_{2,0} \times e_{2,1}\| + \|e_1 \times e_2\| \|e_{0,0} \times e_{0,1}\|}{den}
\end{aligned} \tag{A.38}
$$

The candidate center must satisfy the following six predicates for it to project into the body of the three edge segments.

$$
0 \geq \frac{\left(\begin{array}{c} den\,(e_0 \cdot e_{0,0}) - (\|e_1\|\,(e_0 \cdot e_2) - \|e_2\|\,(e_0 \cdot e_1))\,\|e_{0,0} \times e_{0,1}\| \\ +\|e_0\|\,(\|e_0\|\|e_1\| - (e_0 \cdot e_1))\,\|e_{2,0} \times e_{2,1}\| \end{array} \right)}{den} \tag{A.39}
$$

$$
0 \leq \frac{\left(\begin{array}{c} den\,(e_0 \cdot e_{0,1}) - (\|e_1\|\,(e_0 \cdot e_2) - \|e_2\|\,(e_0 \cdot e_1))\,\|e_{0,0} \times e_{0,1}\| \\ +\|e_0\|\,(\|e_0\|\|e_1\| - (e_0 \cdot e_1))\,\|e_{2,0} \times e_{2,1}\| \end{array} \right)}{den} \tag{A.40}
$$

$$
0 \geq \frac{\left(\begin{array}{c} den\,(e_1 \cdot e_{1,0}) + \|e_1\|\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,\|e_{0,0} \times e_{0,1}\| \\ -\|e_1\|\,(\|e_0\|\|e_1\| - (e_0 \cdot e_1))\,\|e_{2,0} \times e_{2,1}\| \end{array} \right)}{den} \tag{A.41}
$$

$$
0 \leq \frac{\left(\begin{array}{c} den\,(e_1 \cdot e_{1,1}) + \|e_1\|\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,\|e_{0,0} \times e_{0,1}\| \\ -\|e_1\|\,(\|e_0\|\|e_1\| - (e_0 \cdot e_1))\,\|e_{2,0} \times e_{2,1}\| \end{array} \right)}{den} \tag{A.42}
$$

$$
0 \geq \frac{\left(\begin{array}{c} den\,(e_2 \cdot e_{2,0}) - \|e_2\|\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,\|e_{0,0} \times e_{0,1}\| \\ -\,(\|e_0\|\,(e_1 \cdot e_2) - \|e_1\|\,(e_0 \cdot e_2))\,\|e_{2,0} \times e_{2,1}\| \end{array} \right)}{den} \tag{A.43}
$$

$$
0 \leq \frac{\left(\begin{array}{c} den\,(e_2 \cdot e_{2,1}) - \|e_2\|\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))\,\|e_{0,0} \times e_{0,1}\| \\ -\,(\|e_0\|\,(e_1 \cdot e_2) - \|e_1\|\,(e_0 \cdot e_2))\,\|e_{2,0} \times e_{2,1}\| \end{array} \right)}{den} \tag{A.44}
$$

The following first two predicates check if the circle center is closer to one of the two end points of a fourth edge segment E_3 rather than the body of the segment. The third predicate checks if a vertex V_3 lies inside the circle. Note that V_3 can be set to be one of the end points of an edge segment. The final predicate tests whether the body of an edge E_3, represented as an infinite line, penetrates the

circle.

$$
\begin{aligned}
0 \leq & \frac{\left(\begin{array}{c} den\,(e_3 \cdot e_{3,0}) - (\|e_1\|\,(e_2 \cdot e_3) - \|e_2\|\,(e_1 \cdot e_3))\,\|e_{0,0} \times e_{0,1}\| \\ -\,(\|e_0\|\,(e_1 \cdot e_3) - \|e_1\|\,(e_0 \cdot e_3))\,\|e_{2,0} \times e_{2,1}\| \end{array}\right)}{den} \\
0 \geq & \frac{\left(\begin{array}{c} den\,(e_3 \cdot e_{3,1}) - (\|e_1\|\,(e_2 \cdot e_3) - \|e_2\|\,(e_1 \cdot e_3))\,\|e_{0,0} \times e_{0,1}\| \\ -\,(\|e_0\|\,(e_1 \cdot e_3) - \|e_1\|\,(e_0 \cdot e_3))\,\|e_{2,0} \times e_{2,1}\| \end{array}\right)}{den} \\
0 < & \frac{\left(\begin{array}{c} den^2\|v_3\|^2 + 2den\,(e_1 \cdot v_3)\,(\|e_2\|\|e_{0,0} \times e_{0,1}\| - \|e_0\|\|e_{2,0} \times e_{2,1}\|) \\ +2den\,(e_0 \cdot v_3)\,\|e_1\|\|e_{2,0} \times e_{2,1}\| - 2den\,(e_2 \cdot v_3)\,\|e_1\|\|e_{0,0} \times e_{0,1}\| \\ +\|e_{0,0} \times e_{0,1}\|^2\,(\|e_1\|\|e_2\| - (e_1 \cdot e_2))^2 \\ +2\|e_{0,0} \times e_{0,1}\|\|e_{2,0} \times e_{2,1}\|\|e_1\|\,(\|e_0\|\,(e_1 \cdot e_2) + \|e_2\|\,(e_0 \cdot e_1)) \\ -2\|e_{0,0} \times e_{0,1}\|\|e_{2,0} \times e_{2,1}\|\|e_1\|^2\,(\|e_0\|\|e_2\| + (e_0 \cdot e_2)) \\ -2\|e_{0,0} \times e_{0,1}\|\|e_{2,0} \times e_{2,1}\|\|e_0 \times e_1\|\|e_1 \times e_2\| \\ +\|e_{2,0} \times e_{2,1}\|^2\,(\|e_0\|\|e_1\| - (e_0 \cdot e_1))^2 \end{array}\right)}{den} \\
a = & \left(\begin{array}{c} (\|e_2\|\|e_1 \times e_3\| - \|e_1\|\|e_2 \times e_3\|)\,\|e_{0,0} \times e_{0,1}\| \\ +\,(\|e_1\|\|e_0 \times e_3\| - \|e_0\|\|e_1 \times e_3\|)\,\|e_{2,0} \times e_{2,1}\| \end{array}\right) \\
b = & \|e_0 \times e_1\|\|e_{2,0} \times e_{2,1}\| + \|e_1 \times e_2\|\|e_{0,0} \times e_{0,1}\| \\
0 < & \frac{a^2 + 2den\|e_{3,0} \times e_{3,1}\|a + \|e_{3,0} \times e_{3,1}\|^2 den^2 - \|e_3\|^2 b^2}{\|e_3\|^2 den^2}
\end{aligned}
\tag{A.45}
$$

A.7.1 EvEE Parallel

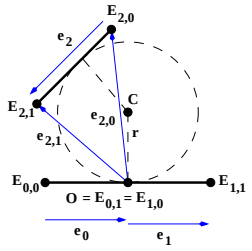


Figure A.13: *EvEE Parallel*

The $EvEE$ parallel case arises when the vertex site is the starting endpoint of the subsequent edge site (Figure A.13). This case is diagnosed using topological and numerical tests. It is equivalent to vEE with $V_0 = E_{0,1}$.

Appendix B

Linear Algebra

B.1 Introduction

This appendix describes some basic linear algebra concepts about eigen-decompostions and more general Jordan decompositions of matrices. It shows how to derive these decompositions and how to use them to quickly raise a matrix to an integer power.

B.2 Eigen-decomposition

Constant time evaluation of subdivision schemes depends on raising the stationary matrix A to a variable power $g-1$ in a constant number of matrix operations. Diagonalizing A into its eigen-decomposition makes this possible. In this section, we will review the eigen-decomposition and use a general 2×2 matrix as an example.

B.2.1 General 2×2 Matrices

Matrix A_2 in Equation B.1 is a general 2×2 matrix. The inverse, A_2^{-1}, can be computed in closed form as shown in Equation B.2, if the determinant of A_2, $Det\,(A_2) = \frac{(ad-bc)}{\alpha}$, does not equal zero.

$$A_2 = \frac{1}{\alpha}\begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{B.1}$$

$$A_2^{-1} = \frac{\alpha}{ad - bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{B.2}$$

B.2.2 Eigenvalues

For a general $n \times n$ square matrix A, the eigenvalues of A are n special scalar values $\lambda_i \{i \in [1, n]\}$ such that for special corresponding eigenvectors $v_i \{i \in [1, n]\}$, multiplication of A with v_i is the same as scaling v_i by λ_i, Equation B.3. By substituting v_i with itself multiplied by the identity matrix I in Equation B.4, we can then combine terms in Equation B.5.

$$\lambda_i v_i = A v_i \tag{B.3}$$

$$\lambda_i (I v_i) = A v_i \tag{B.4}$$

$$\lambda_i I v_i = A v_i$$

$$0 = (A - \lambda_i I) v_i \tag{B.5}$$

v_i is in the nullspace of the matrix $(A - \lambda_i I)$, but the vector $v_i = 0$ always satisfies Equation B.5. We are interested in nonzero eigenvectors v_i only, so $(A - \lambda_i I)$ must be singular so that its nullspace will be nonempty. $(A - \lambda_i I)$ is singular if its determinant is zero. Equation B.6 is the characteristic equation of A, and the n roots of this equation define the eigenvalues $\lambda_i \{i \in [1, n]\}$.

$$0 = Det (A - \lambda_i I) \tag{B.6}$$

For the case of the general 2×2 matrix A_2, the characteristic equation, Equation B.7 can be expanded analytically to yield an equation that is quadratic in λ_i, Equation B.11.

$$0 = Det\left(A_2 - \lambda_i I\right) \tag{B.7}$$

$$0 = Det\left(\frac{1}{\alpha}\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} \lambda_i & 0 \\ 0 & \lambda_i \end{bmatrix}\right) \tag{B.8}$$

$$0 = \frac{1}{\alpha} Det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} \alpha\lambda_i & 0 \\ 0 & \alpha\lambda_i \end{bmatrix}\right) \tag{B.9}$$

$$0 = \frac{1}{\alpha}\begin{vmatrix} a - \alpha\lambda_i & b \\ c & d - \alpha\lambda_i \end{vmatrix} \tag{B.10}$$

$$0 = \frac{1}{\alpha}\left(\alpha^2\lambda_i^2 - \alpha\left(a + d\right)\lambda_i + \left(ad - bc\right)\right) \tag{B.11}$$

Equation B.11 can be solved analytically using the quadratic equation to yield the 2 eigenvalues λ_1 and λ_2 in Equation B.12.

$$\lambda_{1,2} = \frac{a + d \pm \sqrt{\left(a - d\right)^2 + 4bc}}{2\alpha} \tag{B.12}$$

B.2.3 Eigenvectors

For a general $n \times n$ square matrix A, once the eigenvalues $\lambda_i \{i \in [1, n]\}$ have been computed, the corresponding eigenvectors $v_i \{i \in [1, n]\}$ are computed by finding the nullspace of the matrix $(A - \lambda_i I)$. The null space is computed for each λ_i by substituting its value into the matrix $(A - \lambda_i I)$ and then performing Gaussian elimination. The matrix will be singular, so at least one row will become all zeroes. It is then possible to solve for the components of v_i as parametric equations of a subset of free components.

The parametric nature of the v_i's show that the eigenvectors are not unique. In Equation B.3, v_i can be replaced with itself multiplied by an abitrary scalar. This arbitrary scaling of the eigenvectors is the only degree of freedom if the eigenvalues are unique, i.e. their algebraic multiplicities are all one. In this case, the nullspaces are all 1-dimensional subspaces, lines through the origin.

If there are multiple roots to the characteristic equation, then the algebraic multiplicity m of some of the eigenvalues will be greater than one. The m eigenvectors corresponding to such an eigenvalue will be a set of linearly independent vectors that span the m-dimensional subspace. The

choice of these vectors has more degrees of freedom than the scaling in the 1-dimension case. If $m = 2$, then the subspace is a plane through the origin, and there is a rotational degree of freedom for choosing the eigenvectors as well as the scaling degree of freedom.

In general, there are some matrices with eigenvalues of algebraic multiplicity $m > 1$ that do not have a complete set of m associated eigenvectors, i.e. the geometric multiplicity is less than the algebraic multiplicity. These matrices are defective, and the more general Jordan decomposition must be used, see Section B.3.

If the eigenvalues are all unique, then the eigenvectors will all be orthogonal.

For the case of a non-defective 2×2 matrix A_2, we will try to solve for the eigenvectors analytically by substituting the eigenvalues from Equation B.12 into the matrix in Equation B.10 and then performing Gaussian elimination.

$$0 = \frac{1}{\alpha} \begin{bmatrix} a - \alpha\lambda_i & b \\ c & d - \alpha\lambda_i \end{bmatrix} v_i \tag{B.13}$$

$$0 = \begin{bmatrix} a - \alpha\lambda_i & b \\ c & d - \alpha\lambda_i \end{bmatrix} \begin{bmatrix} v_{i,x} \\ v_{i,y} \end{bmatrix} \tag{B.14}$$

$$0 = \begin{bmatrix} a - \frac{a+d \pm \sqrt{(a-d)^2+4bc}}{2} & b \\ c & d - \frac{a+d \pm \sqrt{(a-d)^2+4bc}}{2} \end{bmatrix} \begin{bmatrix} v_{i,x} \\ v_{i,y} \end{bmatrix} \tag{B.15}$$

$$0 = \begin{bmatrix} \frac{(a-d) \mp \sqrt{(a-d)^2+4bc}}{2} & b \\ c & \frac{-(a-d) \mp \sqrt{(a-d)^2+4bc}}{2} \end{bmatrix} \begin{bmatrix} v_{i,x} \\ v_{i,y} \end{bmatrix} \tag{B.16}$$

$$0 = \begin{bmatrix} 0 & 0 \\ 1 & \frac{-(a-d) \mp \sqrt{(a-d)^2+4bc}}{2c} \end{bmatrix} \begin{bmatrix} v_{i,x} \\ v_{i,y} \end{bmatrix} \tag{B.17}$$

Equation B.17 shows that the matrix $(A - \lambda_i I)$ is singular. We will let $v_{i,y} = 1$ then we can solve for $v_{i,x}$, Equation B.18.

$$v_{1,2} = \begin{bmatrix} \frac{(a-d) \pm \sqrt{(a-d)^2+4bc}}{2c} \\ 1 \end{bmatrix} \tag{B.18}$$

In Equation B.18, if $c = 0$, then the eigenvector is not well defined. We must handle the case where $c = 0$ or $b = 0$ separately. In this case the eigenvalues simplify to $\lambda_1 = \frac{a}{\alpha}$ and $\lambda_2 = \frac{d}{\alpha}$, and Gaussian elimination will produce a matrix with a single non-zero element. Hence, $v_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^t$ and $v_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^t$, respectively.

B.2.4 Eigen-Decomposition

For a non-defective $n \times n$ square matrix A, the geometric multiplicity equals the algebraic multiplicity for all of the eigenvalues $\lambda_i \{i \in [1, n]\}$, so there is a corresponding set of n linearly independent eigenvectors $v_i \{i \in [1, n]\}$. It is then possible to diagonalize A by transforming A by the eigenvectors and their inverse. Remembering Equation B.19 that defines each eigenvector, we can write all n equations as columns next to each other in Equation B.20.

$$Av_i = v_i\lambda_i \tag{B.19}$$

$$A \begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} v_1\lambda_1 & v_2\lambda_2 & \cdots & v_n\lambda_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \tag{B.20}$$

We will define V as the matrix of column eigenvectors. We can rewrite the right side of Equation B.20 as the eigenvector matrix V multiplied by a diagonal matrix Λ that has the n eigenvalues along its diagonal, Equation B.21. After right multiplying by V^{-1}, we derive the eigen-decomposition of A, Equation B.24.

$$AV = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{B.21}$$

$$AV = V\Lambda \tag{B.22}$$

$$AVV^{-1} = V\Lambda V^{-1} \tag{B.23}$$

$$A = V\Lambda V^{-1} \tag{B.24}$$

The eigen-decomposition of A, Equation B.24, makes it very efficient to raise A to an integer exponent g. Consider first squaring A, Equation B.25.

$$\begin{aligned} A^2 &= AA \\ &= V\Lambda V^{-1}V\Lambda V^{-1} \\ &= V\Lambda\Lambda V^{-1} \\ &= V\Lambda^2 V^{-1} \end{aligned} \tag{B.25}$$

Equation B.26 for A to an integer exponent g is then derived by induction. This is very efficient to compute because raising Λ to the exponent g can be computed by simply raising the n diagonal elements to g, instead of performing g matrix multiplications.

$$A^g = V\Lambda^g V^{-1} \tag{B.26}$$

For the 2×2 matrix A_2, the eigen-decomposition is the following:

$$A_2 = V_2\Lambda_2 V_2^{-1} \tag{B.27}$$

$$V_2 = \begin{bmatrix} \frac{a-d-\sqrt{(a-d)^2+4bc}}{2c} & \frac{a-d+\sqrt{(a-d)^2+4bc}}{2c} \\ 1 & 1 \end{bmatrix} \tag{B.28}$$

$$\Lambda_2 = \begin{bmatrix} \frac{a+d-\sqrt{(a-d)^2+4bc}}{2\alpha} & 0 \\ 0 & \frac{a+d+\sqrt{(a-d)^2+4bc}}{2\alpha} \end{bmatrix} \tag{B.29}$$

$$V_2^{-1} = \frac{-c}{\sqrt{(a-d)^2+4bc}} \begin{bmatrix} 1 & -\frac{a-d+\sqrt{(a-d)^2+4bc}}{2c} \\ -1 & \frac{a-d-\sqrt{(a-d)^2+4bc}}{2c} \end{bmatrix} \tag{B.30}$$

B.2.5 Product of eigenvalues

We remember that the determinant of the multiplication of two $n \times n$ matrices A and B is the product of their determinants.

$$Det\,(AB) = Det\,(A)\,Det\,(B) \tag{B.31}$$

We can then prove that the determinant of a matrix is equal to the product of its eigenvalues, using the eigen-decomposition.

$$A = V\Lambda V^{-1} \tag{B.32}$$

$$Det\,(A) = Det\left(V\Lambda V^{-1}\right) \tag{B.33}$$

$$= Det\,(V)\,Det\,(\Lambda)\,Det\left(V^{-1}\right) \tag{B.34}$$

$$= Det\left(V^{-1}V\Lambda\right) \tag{B.35}$$

$$= Det\,(\Lambda) \tag{B.36}$$

$$= \prod \lambda_i \tag{B.37}$$

For the 2×2 matrix A_2 the analytic solution is as follows:

$$\lambda_1 \lambda_2 = \frac{ad - bc}{\alpha} = Det\,(A_2) \tag{B.38}$$

B.2.6 Sum of eigenvalues

We can prove that the trace of a matrix is equal to the sum of its eigenvalues, using the eigen-decomposition and the Einstein tensor notation. Equation B.39 defines matrix multiplication using tensor notation. Note that the order of terms in tensor notation does not matter, but the order of the indices does. Equation B.40 defines the Kronecker delta or identity tensor δ_{ij}.

$$AB = A_{ik} B_{kj} = B_{kj} A_{ik} \tag{B.39}$$

$$AA^{-1} = I = A_{ik} A^{-1}_{kj} = \delta_{ij} \tag{B.40}$$

Using these tensor relations and the eigen-decomposition of A, we can derive the trace of A.

$$A = V \Lambda V^{-1} \tag{B.41}$$

$$A_{ij} = V_{ik} \Lambda_{km} V^{-1}_{mj} \tag{B.42}$$

$$Trace\,(A) = A_{ii} \tag{B.43}$$

$$= V_{ik} \Lambda_{km} V^{-1}_{mi} \tag{B.44}$$

$$= V^{-1}_{mi} V_{ik} \Lambda_{km} \tag{B.45}$$

$$= \delta_{mk} \Lambda_{km} \tag{B.46}$$

$$= \Lambda_{mm} \tag{B.47}$$

$$= Trace\,(\Lambda) \tag{B.48}$$

$$= \sum \lambda_i \tag{B.49}$$

For the 2×2 matrix A_2 the analytic solution is as follows:

$$\frac{\lambda_1 + \lambda_2}{2} = \frac{a + d}{2\alpha} = \frac{Trace\,(A_2)}{2} \tag{B.50}$$

B.2.7 Eigen-decomposition of $A' = A - \lambda I$

Given a matrix A and its eigen-decomposition, we will show that a matrix A' equal to the matrix A minus a matrix which is a uniform scaling by the factor λ has a very similar eigen-decomposition.

$$AV = V\Lambda \tag{B.51}$$

$$A' = A - \lambda I \tag{B.52}$$

$$A'V = (A - \lambda I)\, V \tag{B.53}$$

$$= AV - (\lambda I)\, V \tag{B.54}$$

$$= V\Lambda - V\,(\lambda I) \tag{B.55}$$

$$= V\,(\Lambda - \lambda I) \tag{B.56}$$

$$A'V = V\Lambda' \tag{B.57}$$

Equation B.57 shows that A' has the same set of eigenvectors V as A. The eigenvalues λ'_i of A' differ from the eigenvalues λ_i of A by the scalar λ.

$$\Lambda' = \Lambda - \lambda I \tag{B.58}$$

$$\lambda'_i = \lambda_i - \lambda : i \in [1, n] \tag{B.59}$$

B.2.8 Inverses and Pseudoinverses

Given a matrix A and its eigen-decomposition, it is easy to form its inverse A^{-1}, if it exists.

$$A = V\Lambda V^{-1} \tag{B.60}$$

$$A^{-1} = \left(V\Lambda V^{-1}\right)^{-1} \tag{B.61}$$

$$= \left(V^{-1}\right)^{-1} \Lambda^{-1} V^{-1} \tag{B.62}$$

$$A^{-1} = V\Lambda^{-1}V^{-1} \tag{B.63}$$

Inverting A then reduces to inverting the eigenvalue matrix Λ which is a non-uniform scaling matrix. Λ^{-1} is also a diagonal matrix where the entries on the diagonal are:

$$\lambda_i^{-1} = \frac{1}{\lambda_i} : i \in [1, n] \tag{B.64}$$

A is invertible if none of its eigenvalues equal zero.

If A is non-invertible, then its pseudoinverse can be formed by inverting all non-zero eigenvalues while keeping all zero eigenvalues as zero. The pseudoinverse will then ignore projections of the matrix onto the subspace spanned by the eigenvectors corresponding to zero eigenvalues.

$$A^\dagger = V\Lambda^\dagger V^{-1} \tag{B.65}$$

B.3 Jordan Decomposition

Exact evaluation of stationary subdivision schemes depends on the ability to raise the stationary subdivision matrix A to an arbitrary power g in constant-time. The subdivision matrices for smooth extraordinary patches [29] are all diagonalizable by their eigen-decomposition, so raising them to a power simplifies to raising the scalar diagonal eigenvalues to that power and then performing a constant number of matrix operations.

$$A = V\Lambda V^{-1} \tag{B.66}$$

$$A^g = V\Lambda^g V^{-1} \tag{B.67}$$

On the other hand, the subdivision matrices for patches near crease and corner vertices of certain valences are defective, i.e. the algebraic multiplicity of certain eigenvalues is higher than the geometric multiplicity, so it is impossible to find a full rank set of eigenvectors. In the case of a defective matrix, it is still always possible to find its Jordan decomposition, which is block diagonal and easy to raise to an arbitrary power g in constant-time.

$$A = VJV^{-1} \tag{B.68}$$

A Jordan block J_b of J will be of size n_b equal to the number of missing eigenvectors plus one for an associated eigenvalue λ_b with algebraic multiplicity $\geq n_b$. J_b has λ_b along the diagonal, and under our definition, ones below the diagonal. J_b can be thought of as a diagonal matrix Λ_b plus a

nilpotent subdiagonal matrix L_b.

$$J_b = \Lambda_b + L_b$$

$$\begin{bmatrix} \lambda_b & 0 & 0 & 0 \\ 1 & \lambda_b & 0 & 0 \\ 0 & 1 & \lambda_b & 0 \\ 0 & 0 & 1 & \lambda_b \end{bmatrix} = \begin{bmatrix} \lambda_b & 0 & 0 & 0 \\ 0 & \lambda_b & 0 & 0 \\ 0 & 0 & \lambda_b & 0 \\ 0 & 0 & 0 & \lambda_b \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{B.69}$$

The vectors $\{v_i : i = 1 \rightarrow n_b - 1\}$ of V associated with a Jordan block J_b satisfy the equation $Av_i = \lambda_b v_i + v_{i+1}$, while the final vector v_{n_b} satisfies the eigenvector equation $Av_{n_b} = \lambda_b v_{n_b}$.

We can then express J_b^g as the binomial expansion of $(\Lambda_b + L_b)^g$.

$$J_b^g = (\Lambda_b + L_b)^g = \sum_{i=0}^{g} \binom{g}{i} \Lambda_b^{g-i} L_b^i \tag{B.70}$$

L_b^0 is defined to be the identity matrix I_b of size n_b. Λ_b^{g-i} is simply a uniform scaling matrix, so it can be replaced by the scalar λ_b^{g-i}. If $g < i$, then $\binom{g}{i} \equiv 0$. Each multiplication of L_b to itself moves the line of ones diagonally down to the left, so if $i \geq n_b$ then $L_b^i \equiv 0$. This truncates the sum to the first n_b terms.

$$J_b^g = \sum_{i=0}^{n_b-1} \binom{g}{i} \lambda_b^{g-i} L_b^i \tag{B.71}$$

$$= \begin{bmatrix} \binom{g}{0} \lambda_b^g & 0 & 0 & 0 \\ \binom{g}{1} \lambda_b^{g-1} & \binom{g}{0} \lambda_b^g & 0 & 0 \\ \binom{g}{2} \lambda_b^{g-2} & \binom{g}{1} \lambda_b^{g-1} & \binom{g}{0} \lambda_b^g & 0 \\ \binom{g}{n_b-1} \lambda_b^{g-(n_b-1)} & \binom{g}{2} \lambda_b^{g-2} & \binom{g}{1} \lambda_b^{g-1} & \binom{g}{0} \lambda_b^g \end{bmatrix} \tag{B.72}$$

The necessary set of binomial coefficients $\binom{g}{i}$ can be computed in $O(n_b)$ time by starting with $\binom{g}{0} \equiv 1$ and iteratively applying the formula:

$$\binom{g}{i+1} = \binom{g}{i} \frac{g-i}{i+1} \tag{B.73}$$

Left multiplying J_b^g by a row vector $x = \{x_0, x_1, \cdots x_{n_b-1}\}$ reduces to the sum:

$$i \in [0, n_b - 1] : y_i = \sum_{j=0}^{min(n_b-1-i,g)} \binom{g}{j} \lambda^{g-j} x_{i+j} \tag{B.74}$$

The Jordan decomposition is easy and robust to use, once it has been derived. For the subdivision matrices that we are considering, J will have at most one Jordan block of size $n_b = 2$, and all other blocks will be eigen blocks of size 1. Unfortunately numerical methods to find the Jordan decomposition, such as in Mathematica [31], are not stable and are likely to return a diagonal matrix where some of the eigenvectors differ by ϵ. The eigenvector matrix V is then noninvertible, so the decomposition is invalid. Fortunately, for the subdivision matrices we are interested in, we are able to derive analytic forms for their Jordan decompositions (see Chapter 4 for details).

Bibliography

[1] A. Oppenheim and A. Willsky with I. Young. *Signals and Systems*. Prentice-Hall, United States of America, first edition, 1983.

[2] Alias|Wavefront. Maya 3.0, 2000.

[3] David H. Bailey. Mpfun arbitrary-precision arithmetic package, 1995.

[4] H. Biermann, A. Levin, and D. Zorin. Piecewise Smooth Subdivision Surfaces with Normal Control. *Proceedings of SIGGRAPH 2000*, pages 113–120, August 2000.

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

[6] T. DeRose, M. Kass, and T. Troung. Subdivision Surfaces in Character Animation. *Proceedings of SIGGRAPH 1998*, pages 85–94, July 1998.

[7] G. Strang. *Linear Algebra and its Applications, Appendix B*. Thomas Learning, United States of America, third edition, 1988.

[8] R. Gray. Toeplitz and Circulant Matrices: A review. Technical report, Stanford, Stanford University, Stanford, CA 94305, 1971. http://ee.stanford.edu/g̃ray/toeplitz.pdf.

[9] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[10] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise Smooth Suface Reconstruction. *Proceedings of SIGGRAPH 1994*, pages 295–302, July 1994.

[11] John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*. Springer, New York, first edition, 1997.

[12] L. Kobbelt. $\sqrt{3}$-Subdivision. *Proceedings of SIGGRAPH 2000*, pages 103–112, July 2000.

[13] D. T. Lee and R. L. Drysdale. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10(1):73–78, February 1981.

[14] A. Levin and D. Levin. Analysis of quasi uniform subdivision. *Applied and Computational Harmonic Analysis*, 15(1):18–32, 2003.

[15] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.

[16] C. Loop. Bounded Curvature Triangle Mesh Subdivision with the Convex Hull Property. *The Visual Computer*, 18:316–325, 2002.

[17] M. Held. *On the Computational Geometry of Pocket Milling*. Springer, Berlin, lncs edition, 1991.

[18] S. McMains, J. Smith, J. Wang, and C. Séquin. Layered Manufacturing of Thin-Walled Parts. *ASME Design Engineering Technical Conferences 2000, 26th Design Automation Conference*, September 2000.

[19] Sara McMains, Jordan Smith, and Carlo Séquin. Thin-Wall Calculation for Layered Manufacturing. *Transactions of the ASME Journal of Computing and Information Science in Engineering*, 3(3):210–218, September 2003.

[20] Jörg Peters and Le-Jeng Shiue. 4-3 Directionally Ripple-free Subdivision. *ACM Transactions on Graphics*, pages 1–22, MONTH 20YY.

[21] Pixar. *The RenderMan Interface Version 3.2*. https://renderman.pixar.com, July 2000.

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

[23] M. Sabin. Cubic Recursive Division with Bounded Curvature. *Curves and surfaces*, pages 411–414, 1991.

[24] S. Sarma and P. Wright. Reference Free Part Encapsulation: A New Universal Fixturing Concept. *Journal of Manufacturing Systems*, 16(1), 1997.

[25] M. I. Shamos and D. Hoey. Closest Point Problems. *Proceedings 16th Annual IEEE Symposium on Foundation of Computer Science*, October 1975.

[26] J. R. Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–363, October 1997.

[27] J. Stam. Evaluation Of Loop Subdivision Surfaces. *Proceedings of SIGGRAPH 1998*, pages 395–404, July 1998.

[28] J. Stam. Evaluation of Loop Subdivision Surfaces. Technical report, Alias|Wavefront, 1218 Third Ave, 8th Floor, Seattle, WA 98101, 1998. http://www.dgp.toronto.edu/people/stam/reality/Research/pdf/loop.pdf.

[29] J. Stam. Exact Evaluation Of Catmull-Clark Subdivision Surfaces At Arbitrary Parameter Values. *Proceedings of SIGGRAPH 1998*, pages 395–404, July 1998.

[30] J. Stam and C. Loop. Quad/Triangle Subdivision. *Computer Graphics Forum*, 22(1):1–7, 2003.

[31] Wolfram, S. *The Mathematica Book*. Wolfram Media, Champaign, IL, fourth edition, 1999.

[32] L. Ying and D. Zorin. A Simple Manifold-Based Construction of Surfaces of Arbitrary Smoothness. *ACM Transactions on Graphics*, 23(3):XX, July 2004.

[33] D. Zorin. Personal correspondence, December 2003.

[34] D. Zorin and D. Kristjansson. Evaluation of Piecewise Smooth Subdivision Surfaces. *The Visual Computer, 18*, pages 299–315, 2002.