MA 323 Geometric Modelling
Course Notes: Day 36
Simple Subdivision Methods

David L. Finn

32 Polyhedral Surfaces and Subdivision Surfaces

In this chapter, we explore a different type of surface creation method. We start the chapter with a simple method based on a Hermite type interpolation for a triangular patch. This method is a extension of the methods in the previous chapters to a introduce a new structure for creating surfaces. In this structure, we start with a polyhedral surface, a collection of planar faces. We then define a new polyhedral surface from the old polyhedral surface, that is a subdivision algorithm.

More generally, in this chapter, we define a discrete approximation to a surface as a collection of points, edges, and faces. This is a direct generalization of a discrete curve as collection of line segments, that is a polyline or a for a closed curve a polygon. These surfaces are generalizations of the standard polyhedrons from Solid Euclidean geometry; tetrahedrons, cubes, hexahedrons, dodecahedrons, et cetera. For our purposes, we consider subdivision algorithms to produce refinements of a polyhedral surface, so that in the limit a smooth surface is created. A subdivision surface is a surface that arises by applying a subdivision algorithm (a recursive algorithm) to a polyhedral surface.

To describe our procedure more precisely, we recall that a polygon is an object in a plane that consists of a set of vertices (points) and edges where the edges are straight lines that join to points. More generally, a polygon is a special type of planar graph (from Discrete and Combinatorial Mathematics). For a polygon, the edges can be described as an ordered list of the edges, where edges are connected in order, and each pair of edges meets at a vertex. This implies since the list is closed the first vertex and the last vertex are the same that the number of vertices is equal to the number of edges in a polygon.

A polyhedral surface is a spatial object that consists of faces (typically a polygon in a plane), edges (line segments) and vertices. One of the principle goals of this chapter is to examine the structure of a polyhedral surface. This entails developing new data structures, as we will not be creating parametric surfaces. These data structures are especially nice, when one uses an object oriented programming language to describe the surfaces.

A second goal for this chapter is to introduce the notion of a subdivision surface. Recall, we talked about subdivision curves a little bit, when using de Casteljau's algorithm to create a Bezier curve. A subdivision surface is an extension of these methods. The advantage of subdivision surfaces over patches is that they are easy to implement on computers, as they are iterative to create a refinement of an object. Subdivision surfaces also are nice because you can define the object topologically, that is up to the number of holes in the

surface. The subdivision algorithms preserve topological structure, which is nice. To create a topological surface with patches, requires significant work in order to glue the patches together in a nice manner. Subdivision surfaces remove that annoyance. Basically, one only has to create a nice "framework" for the surface. The disadvantage is that it is hard to shape the surface, and that designing with subdivision surfaces is time consuming if you have to specify the information directly. However, subdivision surfaces are efficient given that the initial framework can be sampled directly from the object that is to be modelled then subdivision surfaces are easy to work with. There are no equations to solve to arrange the data, and we can work directly from the data provided.

33 A Hermite-Type Interpolation for Triangular Patches

Consider the following problem:

> Find a smooth surface $\mathcal{S}$ that passes through a given collection of points $\{p_i\}$ in space and having prescribed normal vectors $\{n_i\}$ at the given points.

There exist powerful methods for arranging a given collection of points into triangles, so that it suffices to consider the problem for three points and three normal vectors.

We therefore consider three points with corresponding normal vectors (p_a, n_a), (p_b, n_b), and (p_c, n_c). Rather than producing an expression for every point on the surface, we define a new collection of triangles with points and corresponding normal vectors. The method we describe with produces four triangles from the one triangle. We do this by defining a point and normal vector on each edge curve and then forming new triangles as per the diagram below.

We define the point on the edge curve p_{ij} and the normal vector n_{ij} as follows:

- For $i \neq j$, define $v_i^j = p_j - p_i$ and then the tangent vector for the edge curve t_i^j at p_i as the orthogonal complement of the vector v_i^j to the normal vector n_i, that is

$$t_i^j = v_i^j - \frac{v_i^j \cdot n_i}{n_i \cdot n_i}\, n_i.$$

- For $i \neq j$, define $p_i^j = p_i + \frac{1}{3}\, t_i^j$.

- Define $p_{ij} = \frac{1}{2}\, p_i^j + \frac{1}{2}\, p_j^i$.
- Define $n_{ij} = \frac{1}{2}\, n_i + \frac{1}{2}\, n_j$.

Notice that in this definition $p_{ij} = p_{ji}$ and $n_{ij} = n_{ji}$, but $p_i^j \neq p_j^i$ since $v_i^j = -v_j^i$ and n_i is not necessarily equal to n_j, see diagram below. Furthermore, notice the points p_i^j and p_j^i are basically the Bezier control points for the Hermite curve through p_i and p_j with tangent vectors v_i^j and $-v_j^i$.

With these new edge points p_{ab}, p_{bc} and p_{ca}, we define the four new triangles by $\{p_a, p_{ab}, p_{ac}\}$, $\{p_b, p_{ba}, p_{bc}\}$, $\{p_c, p_{ca}, p_{cb}\}$ and $\{p_{ab}, p_{bc}, p_{ca}\}$. This process is a subdivision algorithm, as for each triangle we produce four new triangles and we refine the approximation of the surface. Each of the original three points that was on the surface remains on the surface. After the first iteration, we have six points on the surface. After the second iteration, we have eighteen points on the surface. The six from the first iteration, plus 3 more for each triangle. Therefore, the number of points on the surface after n iterations is $p_n = 3\,4^{n-1} + p_{n-1}$ as there are 4^{n-1} triangles in the $n-1$st iteration.

The table below shows the number of points and the number of faces in the approximation to the surface. From the data in this table, one easily sees that the number of points on the surface after n iterations equals $4^n + 2$.

n	number of points	number of faces
0	3	1
1	6	4
2	18	16
3	66	64
4	258	256
5	1026	1024

33.1 Exercises

1. Given a set of four points and four normal vectors as described in the diagram below. How can you apply the subdivision procedure in this section to obtain a surface that

passes through the given points and has the given normal vectors? Is there only one such surface?

2. Apply your solution to the previous problem to the data

$$p_a = [0,0,0], \quad n_a = [0,0,1]$$
$$p_b = [1,2,1], \quad n_b = [1/2,1/2,2/3]$$
$$p_c = [1,3,0], \quad n_c = [-1/4,2/3,2/3]$$
$$p_d = [-1,2,-1], \quad n_d = [1/3,-1/3,9/10]$$

3. Construct a subdivision algorithm, given a set of four points and four normal vectors given that the four points all lie on the same plane that is the natural generalization of the subdivision algorithm in this section.

34 Some Simple Subdivision Surfaces

Polyhedral surfaces are nice but in general polyhedral surfaces are analogous to polylines for curves. They will not look smooth unless one uses a huge amount of data. Specifying the amount of data needed to obtain smoothness is not efficient. Therefore, it is convenient to specify a small amount of data to construct a rough approximate polyhedral surface and then apply a smoothing operation to obtain a nice surface that is close in some sense to the original polyhedral surface.

A simple method for defining a smoother operation is a subdivision algorithm. We discussed a subdivision algorithm for curves in Chapter 4, as an application of de Casteljau's algorithm for Bezier curves. In Chapter 4, our subdivision algorithm produced a new polyline that is a refinement of a polyline. A Bezier curve is the limiting curve that is produced by repeatedly applying the algorithm. There are other subdivision algorithm for curves other than the one derived from de Casteljau's algorithm that was presented in Chapter 4. For surfaces, subdivision algorithms are more complicated. For instance, the de Casteljau's algorithm for patches that we discussed in the previous two chapters do not easily generate subdivision type algorithms which will produce Bezier patches. This is mainly because the data structure for Bezier patches is not clearly a polyhedral surface and de Casteljau's algorithm is not defined in terms of the polyhedral structure of the control points.

In this section, we introduce some simple subdivision algorithms and the corresponding subdivision surfaces. The algorithms that we introduce in this section are basically corner cutting algorithms. Every vertex is replaced by a face. The difference is how new vertices and edges are defined.

34.1 General Facts about Subdivision Algorithms

Subdivision algorithms need to accomplish a few things in general. First, a subdivision algorithm must be applied to a polyhedral surface (a set of vertices, edges, and faces). Second, it must return a polyhedral surface. This means given a polyhedral surface, a subdivision algorithm must create a new set of points, a new set of edges and a new set of faces. In principal, the faces should be planar but they do not have to be. Normally, we want planar faces. To accomplish this requires that the algorithm produce planar faces, which is the hard part of a subdivision algorithm unless you restrict the type of polyhedral

surface that the algorithm can be applied. If you remove the restriction of planar faces, then one needs to use an interpolation algorithm to create a surface for each face.

To remove the restriction of planar faces, one standard technique is to apply the algorithm of triangulated surfaces, where all the faces are triangles. The advantage of triangulated surfaces is that three points (non collinear) always define a planar face. This means all the algorithm has to do is generate triangulated surfaces from triangulated surfaces. An advantage of triangulated surfaces is that they are easy to store in computers, and they have been used in computer graphics and CAD/CAM systems for quite a while. So, there are many algorithms for creating triangulated surfaces and for storing them and rendering them. Another advantage is that there are methods for creating patches on triangular faces.

34.2 The Midpoint Algorithm

A simple subdivision algorithm is given by defining a new point set by taking the midpoint of each edge in the surface, $e_{ij} = \frac{1}{2}p_i + \frac{1}{2}p_j$ if there is an edge between points p_i and p_j. We define new edges by connecting new edge points e_{ij} and e_{kl} if the edges p_ip_j and p_kp_l share a face and a vertex, (see diagram below). The new faces are given as follows. Each old face generates a new face by the edge points e_{ij} that lie on the edges of the face, and each old point p_i generates a new face based on all edges points e_{ij} which are generated by edges for which p_i is a vertex of the edge (see diagram below).

The algorithm for this subdivision surface is attached in Maple code. Below, the algorithm is described in pseudocode. Let p_i be the vertices in the surface, a_{ij} be the adjacency matrix for the surface, f_{ij} be the face adjacency matrix, m be the number of faces. Let q_k, b_{kl}, g_{kl} and n be the vertices, adjacency matrix, face adjacency matrix, and number of faces for the new subdivision surface.

- Define the new points. If $a_{ij} = 1$ define $q_k = \frac{1}{2}p_i + \frac{1}{2}p_j$, and set $\text{prec}_k = \{m+i, m+j\}$ the predecessor points of q_k also set $\text{face}_k = f_{ij}$ the faces q_k is adjacent to. We store prec_k as $m + i$ as renumbering of the point indices.
- Define the new edges. Set $b_{kl} = 1$ if $\text{prec}_k \cap \text{prec}_l \neq \{\}$ and $\text{face}_k \cap \text{face}_l \neq \{\}$. That is if the points share a predecessor point and a face.

- Define the new faces. If $b_{kl} = 1$ set $g_{kl} = (\text{prec}_k \cap \text{prec}_l) \cup (\text{face}_k \cap \text{face}_l)$. The edge kl is adjacent to the point $\text{prec}_k \cap \text{prec}_l$ and the edge kl is adjacent to $\text{face}_k \cap \text{face}_l$

This type of algorithm involves cutting the corners of the polyhedral surface. In the limit each original face will generate generically one point on the surface.

This midpoint algorithm can easily be generalized by considering a trisection algorithm by having each edge generate two points. Define new points $e^1_{ij} = \frac{2}{3}p_i + \frac{1}{3}p_j$ and $e^2_{ij} = \frac{1}{3}p_i + \frac{2}{3}p_j$. New edges are defined in the same ways as in the manner as in the midpoint algorithm. With the addition of an edge between e^1_{ij} and e^2_{ij}. New faces are generated in the same manner as in the midpoint algorithm.

34.3 The Centroid Algorithm

In this algorithm, we define a different corner cutting algorithm based on using the centroid of each face. The centroid of a face is the point given by the affine combination

$$f_F = \frac{1}{n} \sum_{v_i \in F} v_i$$

where v_1, v_2, $\cdots$, v_n are the vertices of the face F. In this algorithm, we define new points by *contracting* each face, that is defining the new vertex set by

$$v^1_{F,v} = \frac{1}{2}\, f_F + \frac{1}{2}\, p$$

if $v \in F$. New faces are generated by each old face (containing the new vertex that are generated by that face), each old vertex (containing the new vertices that are generated by that face), and each old edge (containing the vertices that are generated by the endpoints of the edge). The new edges are generated by the old edges. Two vertices have an edge between them if they are on the same face and there was an edge between the vertices that generated them or if they were generated by the same vertex and the faces that generated them were adjacent, see diagram below.

The algorithm is given in the attached Maple code, and a pseudocode version is given below. Let p_i be the vertices of the original surface, a_{ij} be the adjacency matrix of the original

surface, f_{ij} the face adjacency matrix, F_i be the faces of the original surface, and m the number of faces in the original surface. Let q_k, b_{kl}, g_{kl}, G_k, and n be the vertices, adjacency matrix, face adjacency matrix, and number of faces for the new subdivision surface.

- Define the new points: Define the centroid of each face, $f_k = \frac{1}{|F_k|}\sum p_{i_j}$ where $|F_k|$ is the number of vertices in face F_k and p_{i_j} are the vertices in F_k. Then define $q_{k,i_j} = \frac{1}{2}f_k + \frac{1}{2}p_{i_j}$.
- Define the new faces:
  - face faces: For each face, F_k define a new face consisting of the points q_{k,i_j} generated by the face F_k.
  - point faces: For each point p_i define a new face consisting of all the points $q_{k,i}$ generated by the point p_i.
  - edge faces: For each edge e_{ij} between vertices p_i and p_j generate a new face consisting of the four vertices $q_{k,i}$, $q_{k,j}$, $q_{l,i}$, and $q_{l,j}$ where e_{ij} is adjacent to the faces F_k and F_l.
- Define the new edges:
  - edge edges: There is an edge between points $q_{k,i}$ and $q_{k,j}$ if there was an edge between the points p_i and p_j which both belong to the face F_k.
  - face edges: There is an edge between points q_{ki} and q_{li} if there is an edge e_{ij} in the original polyhedral surface with the faces F_k and F_l generating q_{ki} and q_{li} respectively.

34.4 EXERCISES

1. Apply both the midpoint algorithm and the centroid algorithm once to the figure below. Sketch the surface, by hand. Do not implement the code in this problem.

2. Apply the algorithms provided (the Maple code) to the figure below. Hint: The data structures are the same as in the exercise in the previous section.