MA 323 Geometric Modelling
Course Notes: Day 35
Simple Subdivision Methods

David L. Finn

Today, we want to introduce some simple subdivision methods. In particular, we introduce two methods, the midpoint method and the centroid method. These two methods can be viewed simply as cutting the corners and/or cutting edges, where a new polyhedron is obtained from the original polyhedron by cutting corners, see diagram below.

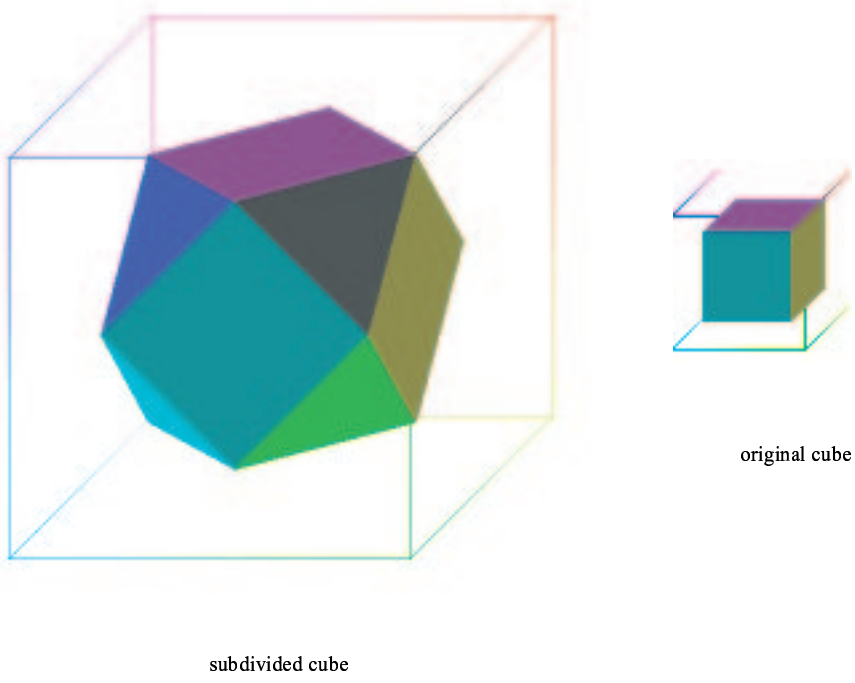

Figure 1: A subdivision method based on cutting corners

35.1 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.

35.2 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.

35.3 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 described below in pseudocode. Let p_i be the

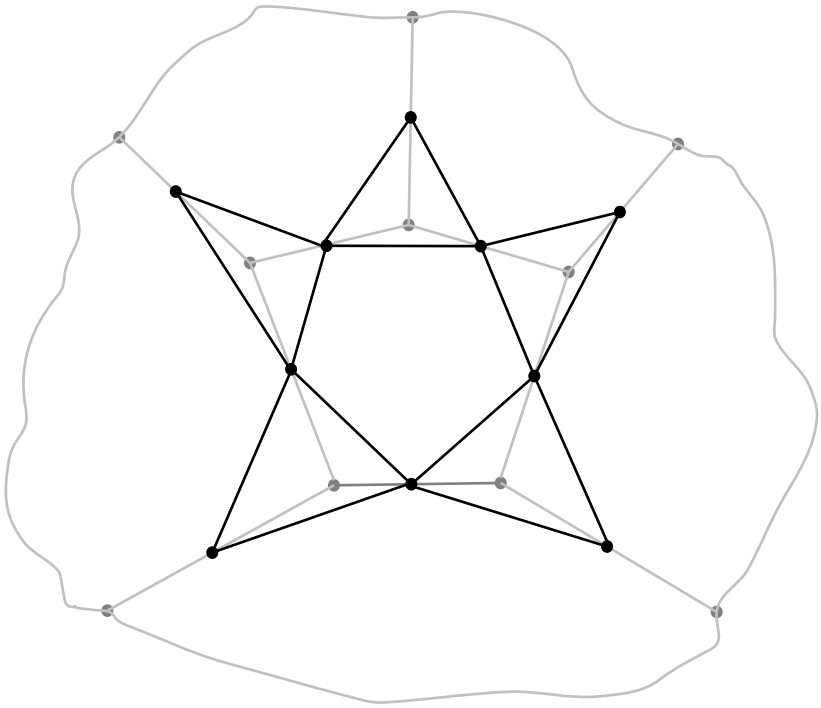

Figure 2: The Midpoint Algorithm

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.

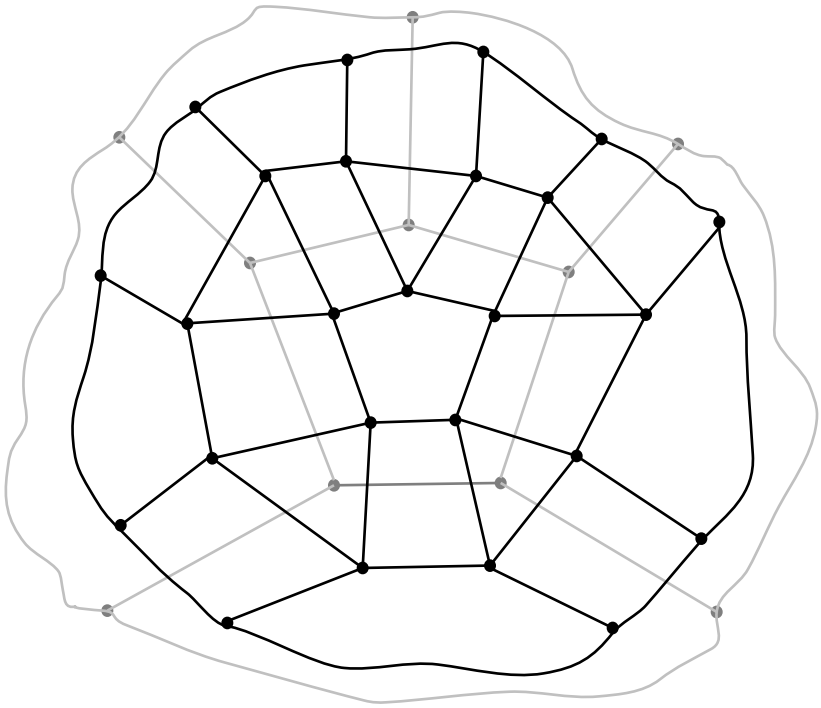

Figure 3: The Centroid Method

35.4 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 pseudocode 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. This is demonstrated pictorially in the figure above

- 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.

35.5 EXERCISES

1. Apply both the midpoint algorithm and the centroid algorithm once to the figure below. Construct the edge graph and the face graph for the polyhedra, and the edge and the face graph after the subdivision and then sketch the polyhedra.

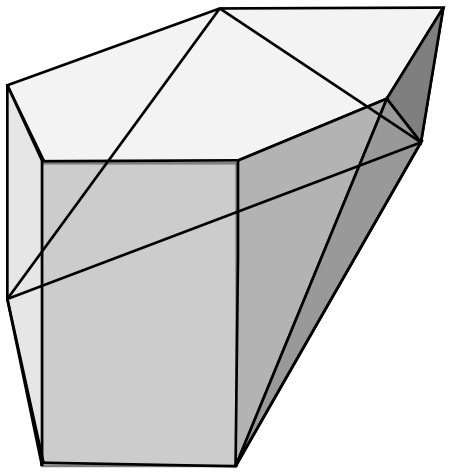

Figure 4: Apply the subdivision methods to this surface

2. Use the Applets to play with the algorithms and the methods to see what happens to different polyhedra under the algorithms.
   (a) What is similar between the surfaces generated by the two methods?
   (b) What is different between the surfaces generated by the two methods?
   (c) For surfaces with the same "topology" (same Euler characteristic $V - F + E$) what is the same and what is different between the surfaces generated by each method?

3. Verify that the midpoint and centroid algorithm will preserve Euler's formula $V-F+E$ for a polyhedra. Therefore, the algorithms preserve the "topology" of the surface.

4. Do the midpoint and centroid algorithm satisfy any convex hull type properties? This is a inclusion type properties meaning if P^0 is a polyhedron and P^1 is the polyhedron generated by the subdivision method, is P^1 contained in convex hull of P^0. More specifically, is P^1 contained in P^0 as a solid object? Justify your conclusions.