

MA 323 Geometric Modelling

Course Notes: Day 34

Polyhedral Surfaces

David L. Finn

Today, we want to start discussing the mathematical structure of polyhedral surfaces and polyhedra. This can be viewed as 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, but not necessarily), edges (line segments) and vertices. One of our principle goals today 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. For engineers, this can be used in essence to generate meshes on surface for which a finite element analysis can be done. 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.

34.1 Polyhedral Surfaces

We first want to spend a few today talking about methods to specify polyhedral surfaces, without considering any subdivision methods. As stated above polyhedral surfaces are three-dimensional versions of polygons such as cubes (six sided regular solid with square faces), tetrahedrons (4 sided regular solid with square faces), hexahedrons (six sided regular solid with triangular faces), octahedrons (8 sided solids with triangular faces), decahedrons, dodecahedrons, etc. These are the shapes of multi-sided dice typically used in role-playing games such as Dungeons and Dragons.

The study of polyhedra is an old field, and the natural extension of Euclidean geometry. Our interest at the moment is purely as a data type. We want to examine these objects as a natural class of objects to work with for modelling. As such, we want to discuss a few facts about polyhedral surfaces. How do we define a polyhedral surface? Are there any conditions on how we define a polyhedral surface, that is are there any relations between the number of faces, number of edges and the number of vertices in a polyhedral surface.

In the abstract definition, a polyhedral surface consists of a set of “polygons” called faces, a set of lines called edges and a set of points called vertices. In this abstract definition, there are relations imposed on the sets. Specifically, each edge belongs to two faces and each edge is defined by two vertices. This is much the same as the definition of a “graph in discrete mathematics” as a set of vertices and edges. In that sense, a polyhedral surface can be considered as a three dimensional analogue of a graph.

As a mathematical structure, we are interested in any conditions on the construction. In particular, are there any necessary conditions on the number of edges, the number of vertices and the number of faces in a polyhedral surface. The answer is yes. Consider a cube (six sided regular solid with square faces) and a tetrahedron (four sided regular solid with triangular faces). In the cube, there are 8 vertices, 6 faces, and 12 edges and in the tetrahedron there are 4 vertices, 4 faces, and 6 edges. One can easily check that the number of faces plus the number of vertices minus the number of edges equals two. More generally, if there are V vertices, F faces and E edges, a result of Euler states for surfaces homeomorphic to

spheres (cubes, tetrahedrons, and other regular platonic solids) that the number of edges + 2 = number of faces plus the number of vertices,

$$2 = F - E + V.$$

To verify that this formula is correct, we first note that each face is a polygon so the number of edges and vertices on each face are equal. The next observation is that each face lies in a plane therefore when two faces share an edge they will share one more vertex than edges. Therefore, when considering the number of edges and the number of vertices in two adjacent faces, we have one more edge than vertex. Therefore, two faces - number of edges in the two faces + number of vertices in the two faces equals 1. We can view the combination of the two faces as one face, and repeat the process. When we have used all the edges and vertices, we will still have one more face to close the solid. Thus, we have the formula.

There are analogous formulas to $2 = F - E + V$ for surfaces with different topology. For instance, a surface with the topology of a donut (i.e. a torus), see diagram below,

will necessarily satisfy

$$0 = F - E + V.$$

To see understand this formula is correct, consider the diagrams below.

We cut the torus along a sequence of edges that form a circle to transform the torus into a cylinder, diagram A transforms to diagram B. We then add a point to each new edge to reduce the torus to a surface homeomorphic to a sphere. Therefore, $2 = 2 + F - E + V$ or $0 = F - E + V$. By cutting along a sequence of edges that forms a circle, we do not change the quantity $F - E + V$ as the number of point and edges in this sequence are the same.

More generally, the topology of a compact closed surface can be described as a number g called the genus (which represents the number of holes in the surface). This number g is

related to the number of faces, edges and vertices by the formula

$$2 - 2g = F - E + V.$$

A sphere (and all surfaces homeomorphic to a sphere) has genus g , a torus (and all surfaces homeomorphic to a torus) has genus 1, a two-holed torus (pictured below and all surface homeomorphic to a two-holed torus) has genus 2. In the past few paragraphs, we have used the technical term homeomorphic loosely, without a definition. Two surfaces are homeomorphic if one can deform one surface to the other by continuous motions, this means that there is a continuous function with continuous inverse between the two surfaces.

34.2 Data Structures for Polyhedral Surfaces

To create a polyhedral surface as described above, we need to give a list of vertices, edges and faces. The vertices or points $\{p_i\}$ can be given as an $n \times 3$ matrix, where each row of the matrix represent a point in xyz -space. The edges can be described with an $n \times n$ matrix, an adjacency matrix $A = [a_{ij}]$. Entries of the matrix are either 0 or 1, where we interpret the i, j entry equal 1 as there exists an edge between the points p_i and p_j . On the other hand if the entry is 0, there is no edge between p_i and p_j . With this set-up, the matrix will be symmetric $a_{ij} = a_{ji}$ and the diagonal entries are zero $a_{ii} = 0$. The faces are represented as set of points, representing the polygon that forms the face. If you order the points so that there exists an edge between each consecutive pair of points, you can use the order to replace the adjacency matrix. An alternative representation for the faces is to say which faces an edge is adjacent on. Each edge is adjacent to only two faces, thus we can create a face adjacency matrix where each entry in the matrix is a set of faces that an edge is adjacent to.

One can also use other data structures, depending on the computer language used, and the sophistication of the programmer. For instance, if one desires to use object oriented programming, you can define a classes for a polyhedral surface, a class for a face, and inside that there must be a class for a point and an edge et cetera, see below.

An Example: Consider a unit cube. The points are

$$\begin{aligned} p_1 &= [0, 0, 0], \quad p_2 = [1, 0, 0], \quad p_3 = [1, 1, 0], \quad p_4 = [0, 1, 0], \\ p_5 &= [0, 0, 1], \quad p_6 = [1, 0, 1], \quad p_7 = [1, 1, 1], \quad p_8 = [0, 1, 1] \end{aligned}$$

The adjacency matrix is

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The faces are given by (each number represents the index of a point)

$$\begin{aligned} F_1 &= \{1, 2, 3, 4\}, \quad F_2 = \{1, 2, 5, 6\}, \quad F_3 = \{2, 3, 6, 7\}, \\ F_4 &= \{3, 4, 7, 8\}, \quad F_5 = \{1, 4, 5, 8\} \quad F_6 = \{5, 6, 7, 8\} \end{aligned}$$

The edge adjacency matrix (the entries are sets telling which face as numbers an edge is adjacent to)

$$B = \begin{bmatrix} \{\} & \{1, 2\} & \{\} & \{1, 4\} & \{2, 4\} & \{\} & \{\} & \{\} \\ \{1, 2\} & \{\} & \{1, 3\} & \{\} & \{\} & \{2, 3\} & \{\} & \{\} \\ \{\} & \{1, 3\} & \{\} & \{1, 5\} & \{\} & \{\} & \{3, 5\} & \{\} \\ \{1, 4\} & \{\} & \{1, 5\} & \{\} & \{\} & \{\} & \{\} & \{4, 5\} \\ \{2, 4\} & \{\} & \{\} & \{\} & \{\} & \{2, 6\} & \{\} & \{4, 6\} \\ \{\} & \{2, 3\} & \{\} & \{\} & \{2, 6\} & \{\} & \{3, 6\} & \{\} \\ \{\} & \{\} & \{3, 5\} & \{\} & \{\} & \{3, 6\} & \{\} & \{5, 6\} \\ \{\} & \{\} & \{\} & \{4, 5\} & \{4, 6\} & \{\} & \{5, 6\} & \{\} \end{bmatrix}$$

There is a non-empty set in the ij entry only if there is an edge between the point p_i and p_j .

The type of data structure used will be dependent on the exact subdivision algorithm that is employed. The algorithm will dictate some of the data structure necessities in the complete data structure.

34.3 Graph Structures for Polyhedral Surfaces

An alternative data structure for a polyhedral is a by given a graph structure to give describes the incidence structure. There will be a edge incidence graph to describe the existence of edges between vertices. This is simple viewing the surface as a wireframe, i.e. viewing the surface as a graph by ignoring the faces. In addition, there is a face-edge graph obtained by viewing the faces are the vertices and joining two faces if they share an edge. In general these graphs are “globally” nonplanar, there will be intersections between edges where there are no vertices. However, they are locally planar, if viewed in part they can be drawn on a piece of paper without intersections between edges when there are no vertices.

EXAMPLE OF POLYHEDRON and GRAPH STRUCTURE

It is worth noting that in programming algorithms for polyhedral surfaces, one uses object oriented programming and defines a polyhedral surface by given a list of vertices, edges and faces. Each vertex in the polyhedral surface has associated to it a list of edges and faces to which it is incident. Each edge in the polyhedral surface has associated to it its two end points as vertices and the two faces to which it is incident. Each faces in the polyhedral surface has associated to it the list of vertices and edge that are incident to the face.

34.4 Exercises

1. For the figure below, with the points labelled. Create
 - (a) An adjacency matrix.
 - (b) A face adjacency matrix.
 - (c) A list of the faces.

2. For the figure below, with the points labelled. Create the graph structures associated to the polyhedron.