3D Geometric Modelin and Processing (6 cfu) 23-24

Notes

University of Pisa

M.Sc. in Computer Science

Contents

Chapter 1

Representing Surfaces

We talk about modeling surfaces because most of the object we see are opaque, thus from the point of view of rendering them we are only interested in their external surface **since we never see what is inside them**.

Formally the main aim of Computer Graphics is representing the boundary surface between the objects and non-objects.

1.1 Defining Surfaces

There are two methods of defining surfaces: Analytic definitions and Approximated definitions.

1.1.1 Analytic Definitions

In the analytic definitions that try to define the surface by using a mathematical (exact) notation.

There are two way to represent a surface in a 3D space using mathematical notation:

- **Parametric representation:** it consists in a function $S : \mathbb{R}^2 \to \mathbb{R}^3$ that maps points on a 2D domain over a 3D surface.
- **Implicit representation:** I define the surface S as the zeros of a function $f : \mathbb{R}^3 \to \mathbb{R}$ s.t.:

$$S = \{p \in \mathbb{R}^3 : f(p) = 0\}$$

1.1.2 Drawbacks of Analytic Definitions

The main limitation of analytic definitions is that for realistic surfaces, meaning any surface that is not a simple object like a sphere or a cube, is not fesable to find an explicit

formulation with a single function that approximates a given shape with sufficient accuracy. To overcome this we define the surface **piecewise**, that is we identify it as being composed of smaller sub-region. By doing so we subdivide the function domain of the surface into smaller sub-regions and for each of them we define an individual function called **surface patch** that represent that segment. Each surface patch is only interested at approximating its sub-region locally, while the global approximation tolerance is controller by the size and number of segments.

1.2 Parametric Representation

A Parametric Surface is defined by a function $f : \omega \to S$ where:

- ω is a 2D parameter domain s.t. $\omega \subset \mathbb{R}^2$
- S is a 3D parameter domain obtained by applying ω to f s.t. $S = f(\omega) \subset \mathbb{R}^3$

1.3 Cellular Complex

A cellular complex, in mathematics and topology, refers to a structure composed of cells (convex polytope) of various dimensions, such as vertices (0-dimensional cells), edges (1-dimensional cells), faces (2-dimensional cells), and higher-dimensional cells. These cells are glued together according to certain rules or combinatorial data. After the second dimension we begin to consider solid 3D object (meaning their inside is not empty), usually used in simulation scenarios.

1.3.1 Properties of Cell Complex

- The order of a cell is the number of its sides. Sides are defined as boundaries of the higher-dimensional cells within the complex.
- a complex is a k-complex if the maximum of the order of its cells is k.

1.4 Definition of Meshes

Meshes are based upon the formalisation of cellular complex. Meshes are a set of polygons (usually they are triangles or four-sided polygons) that approximate my surface.

We say that a cellular complex is a mesh if and only if it satisfies the following rules:

- Every face of a cell belongs to the complex

- $\forall$ cells $C \wedge C^{'}$ their intersection is either empty or is a common face of both. This means that two cells either do not touch, or they must have at most edges/vertices/faces in their boundaries in common (I cannot have a faces defined inside another face).
- a cell is maximal if it is not a face of another cell.

1.4.1 Maximal Cell Complex

Given a cell complex we can say that it is a maximal cell complex if and only if all of its maximal cells have order k. In general, since we mostly will talk about 2 dimension cellular complex, it will amount in checking if there are no dangling edges.

1.5 Simplicial Complex

Simplicial Complex are a specific case of Cellular Complex where I limit the faces to be triangles or 2-simplex.

Triangles are really convenient because they have some convenient properties:

- The number of edges in a triangles is always three.
- A triangle has three vertices and **three points in a topological space always define a plane**. So every 2-simplex defines a plane. Consider for points in a space, these four points does not imply the existence of a plane that touches all of them (to do it I have to make assumption and defining some rules).

1.5.1 Face

Given a simplex σ, we say that the simplex $\sigma^{'}$ is a face (or sub-simplex) of σ if it is defined by a subset of vertices of σ. If $\sigma^{'}$ is a face of σ and $\sigma \neq \sigma^{'}$, then it is a proper face.

1.5.2 Collection of simplexes

A collection of simplexes Σ is a simplicial k-complex if and only if:

- $\forall \sigma_1, \sigma_2 \in \Sigma . \sigma_1 \cap \sigma_2 \neq 0 \Rightarrow \sigma_1 \cap \sigma_2$ is a simplex of Σ.
- $\forall \sigma \in \Sigma$ all the faces of σ belong to Σ.
- k is the maximum degree (order) of simplexes in Σ.

There are the following properties:

- A simplex σ is maximal in a simplicial complex Σ if it is not a proper face of another simplex $\sigma^{'}$ of Σ.
- A simplician k-complex Σ is maximal if all its maximal simplex are of order k (there are no dangling lower dimensional pieces).

1.6 Triangle Meshes

When talking about triangle meshes the intended meaning is a maximal 2-simplicial complex, but in reality is not always true (in most cases I have some extra vertices in our space). Triangle meshes are considered a collection of triangles where each triangle define via its barycentic parametrization a segment of a piecewise linear surface representation.

A triangle is identified by its three vertex $[a, b, c]$. We can identify every point p in the interior of a triangles as a barycentric combination of the corner points:

$$p = \alpha a + \beta b + \gamma c$$

where:

$$\alpha + \beta + \gamma = 1, \quad \alpha, \beta, \gamma \geq 0$$

A triangle mesh is composed of a geometric and topological component.

1.6.1 Topological component

A triangle mesh can be represented topologically is a simplicial complex that ir represented by a graph structure with a set of vertices:

$$\mathcal{V} = \{v_1, ..., v_V\}$$

and a set of triangular faces connecting them:

$$\mathcal{F} = \{f_1, ..., f_F\} \quad s.t. \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$$

Each vertex is associated to a 3D position p_i s.t.:

$$\mathcal{P} = \{p_1, ..., p_V\}, p_i := p(v_i) = (x(v_i), y(v_i), z(v_i)) \in \mathbb{R}^3$$

1.7 Manifoldess

Manifoldess is one of the main property to check in a simplicial complex.

Given a surface S iff $\forall$ point in S if I take its neighborhood, then the neighborhood is homeomorphic to the Euclidean space in two dimension. In practice this means that the neighborhood of each point needs to be homeomorphic to a disk (or a semidisk if the surface has boundaries).

A surface is non-manifold if it has a non-manifold edge or a non-manifold vertex:

- **non-manifold edge**: meaning there is no edge that has more than two incident triangles.
- **non-manifold vertex:** is generated by pinching two surface sheets together at that vertex such that the vertex is incident to more that one fan of triangles. More intuitively a vertex is non-manifold it is in the middle of an hourglass shape

1.8 Orientability

A surface S is orientable if it is possible to make a consistent choice for the normal vector. In practice it means that starting from a face with a normal, I should be able to propagate this normal for all the faces near it and the faces near the faces near it.

1.9 Incidency

Given two simplexes $\sigma, \sigma^{'}$, if σ is a proper face of $\sigma^{'}$ then $\sigma, \sigma^{'}$ are incident.

1.10 Adjacency

Given two k-simplexes $\sigma, \sigma^{'}$ and $m < k$, if there exists a m-simplex that is a proper face of both $\sigma, \sigma^{'}$, then $\sigma, \sigma^{'}$ are m-adjacent.

Note:

- Two triangles sharing an edge are 1-adjacent.
- Two triangles sharing a vertex are 0-adjacent.

There are three classes of adjacency relations, identified by a pair of letters where each of them refers to one of the entity involved in the relation:

- **FF:** refers to an adjacency between triangular **F**aces (so they touch in an edge). It corresponds to the 1-adjacency.
- **EE:** it moves from one **E**dge to another following the vertices that connects them. It corresponds to the 0-adjacency.
- **FE:** it refers to the proper subfaces of a **F**ace that have dimension 1.
- **FV:** adjacency from **F**aces to **V**ertices (e.g. the vertices composing a face). It refers to the proper subfaces of a **F**ace with dimension 0.
- **EV:** given an **E**dge in returns its two **V**ertices. It refers to the propert subface of the **E**dge with dimension 0.
- **VF:** adjacency from a **V**ertex to a **F**ace (e.g. the triangles incident on a vertex). It refers to the $\{F \in Faces | V$ is a proper subface of $F\}$.
- **VE:** given a vertex it returns all the edges that utilize it. Its the set $\{E \in Edges :$ V is a proper subface of $E\}$
- **EF:** given an **E**dge it returns all the **F**aces that utilize it. If there are more than two **F**aces for an edge, then our mesh is not manifold. Its the set $\{F \in Faces | E$ is a proper subface of $F\}$.
- **VV:** given a vertex it returns all the vertices that have edges in common. its the set $\{V' \in Vertices | \exists E : (V, V')$.

These relation are usually stored in the data structures of meshes. The idea is to store a subset of these relations (usually FF, FV and VF) and procedurally generate the rest.

1.11 Partial adiacency

For the sake of conciseness it can be useful to keep only partial sets of relations. This is done not only to save space, but also to have non-redundant sets that in complex situations are difficult to keep updating.

For example VF^* memorize only a reference from a vertex to a face and then surfs over the surface using FF to find the other faces incident on V.

1.12 Bounds of Adjacency Relation

Most of the adjacency relation we have seen have bounds well defined.

Given a two manifold simplicial 2-complex in $\mathbb{R}^3$:

- FV, FE, FF, EF, EV have bounded degree (meaning they are constants if there are no borders):
  - $|FV| = 3, |EV| = 2, |FE| = 3$
  - $|FF| \leq 2$
  - $|EF| \leq 2$
- VV, VE, VF, EE have some average estimations:
  - $|VV| \sim |VE| \sim |VF| \sim 6$
  - $|EE| \sim 10$
  - The number of **F**aces is usually double the number of **V**ertices.

1.13 Implicit Representation

In implicit representations, surfaces are represented by classifying each 3D point in space for being either inside, outside or exactly on the surface S that bounds a solid object (the contour).

In an implicit representation the function $f : \mathbb{R}^3 \to \mathbb{R}$ is defined in the implicit form $f(x, y, z) = c$ where c is a constant usually set to 0. For each point in the 3D space we sample its value according to f and then:

- If $f(x, y, z) > c$: then the point is outside of the surface.
- If $f(x, y, z) = c$: then the point is exactly on the contour.
- If $f(x, y, z) < c$: then the point is inside the surface.

If the function f is continuous, then the implicit surface defines by it has no holes

1.13.1 Queries in implicit representation

Identifying where a point p lies in the surface is trivial: check if the value returned by $f(x(p), y(p), z(p))$ is positive, negative or equal to zero.

1.13.2 Operations on implicit representations

Implicit surfaces can be enlarged by decreasing the function values of f locally and can be shrinked by increasing the function values of f locally.

1.13.3 Drawbacks of implicit representations

The implicit function f of a given surface S is not uniquely determined. As an example consider the function $f(x, y, z) = 0$ where each point of the contour of the surface S is defined by the point that return 0 in f. Now let us define the function $f^{'} = \lambda f$ and note that although they are scalarly different, they define the same point for the contour.

Generating sample points on an implicit surface, finding geodesic neighborhoods, and even just rendering the surface is relatively difficult. Moreover, implicit surfaces do not provide any means of parameterization, which is why it is very difficult to consistently paste textures onto evolving implicit surfaces.

1.13.4 Representing a implicit surface

To represent a implicit surface we will use a **signed distance function** which maps each 3D point x to its signed distance $d(x)$ from the surface S s.t.:

- if $d(x) > 0$: the point is outside S and distant $|d(x)|$ from the contour.
- if $d(x) < 0$: the point is inside S and distant $|d(x)|$ from the contour.
- if $d(x) = 0$: the point is in the contour.

1.13.5 Regular Grids

To process implicit representation the continuous scalar field f is discretized in some bounding box around the object by defining a sufficiently dense grid with nodes $g_{i,j,k} \in \mathbb{R}^3$

1.14 The Five Platonic Solids

A Platonic Solid defines a polyhedrons that is convex, regular polyhedron in three-dimensional Euclidean space. Being a regular polyhedron means that the faces are congruent (identical in shape and size) regular polygons (all angles congruent and all edges congruent), and the same number of faces meet at each vertex. There are only five such polyhedra.

1.14.1 Tetrahedron

The simplest regular polygon is the equilateral triangle, thus the simplest regular polyhedron is obtained by composing four equilateral triangles used as faces, six straight edges, and four vertices.

1.14.2 Octahedron

We can construct an octahedron by placing e two tetraedron one below the other. What we obtain is a regular polyhedron composed of eight faces (that are equilateral triangles), 12 straight edges, and 6 vertices.

1.14.3 Icosahedron

The icosahedron is obtained by taking a uniform pentagonal antiprism and attaching two pentagonal pyramids in both of its pentagonal faces. The generated solid has 20 **F**aces, 30 **E**dges and 12 **V**ertices.

Pentagonal antiprism

A pentagonal antiprism is constructed by a sequence of even-numbered triangle sides closed by two polygon caps. It consists of two pentagons joined to each other by a ring of ten triangles for a total of twelve faces. The generated solid has 12 **F**aces, 20 **E**dges and 10 **V**ertices.

Pentagonal Pyramid

A pentagonal pyramid is a pyramid with a pentagonal base upon which are erected five triangular faces that meet at a point (the apex).

1.14.4 Cube

The cube is the only regular hexahedron (meaning a polygon composed of six faces). It is composed of 6 square **F**aces, facets, or sides, with three meeting at each vertex. It has 12 **E**dges and 8 **V**ertices.

1.14.5 Dodecahedron

A regular dodecahedron or pentagonal dodecahedron is a dodecahedron that is regular, which is composed of 12 regular pentagonal **F**aces, three meeting at each vertex. It has 12 **F**aces, 20 **V**ertices and 30 **E**dges.

Solid	Vertices	Edges	Faces
Tetrahedron	4	6	4
Cube	12	8	6
Octahedron	6	12	8
Dodecahedron	20	30	12
Icosahedron	12	30	20

Table 1.1: The five Platonic Solids

1.15 Euler Characteristic

As the above table shows, it seems that there is a rules regarding the relation of their number of edges, vertices and faces. The Euler Characteristisc explain this relationship in a form of a constant χ that has the same value for every simply connected polyhedron:

$$\chi = V - E + F$$

where:

- V is the number of vertices
- E is the number of edges
- F is the number of faces

Euler Characteristic work in every dimension.

The formula is constructed by alternating the signs of the variables, starting by the number of vertices $(+V)$, subtracting the number of edges $(-E)$, adding the faces $(+F)$ and so on if you are in a greater dimension.

Solid	Vertices	Edges	Faces	χ
Tetrahedron	4	6	4	2
Cube	8	12	6	2
Octahedron	6	12	8	2
Dodecahedron	20	30	12	2
Icosahedron	12	30	20	2

Table 1.2: The five Platonic Solids now with the Euler Characteristic χ

1.15.1 Intuition behind the Euler Characteristic

Let us give a small intuition behind the proof of the Euler Characteristic by construction:

assume we have a simple tetrahedron, then by computing its Euler Characteristic we have:

- 4 Vertices
- 6 Edges
- 4 Faces
- $\chi = V - E + F = 4 - 6 + 4 = 2$

Then suppose we cut the tetrahedron at the top, obtaining a solid that at the top has no more a vertex, but three forming a triangle at the top. Now let us reason about what we have done by cutting the vertex at the top:

- I have removed one vertex and added three by the cutting operation. So overall we have added two edges ($3added - 1removed = 2added$).
- Since the three new vertices form a triangle at the top, we have three new edges.
- I new face is added which is the triangle we have obtained

Thus the new Euler Characteristic will be:

$$\chi = (V + 2) - (E + 3) + (F + 1) = (4 + 2) - (6 + 3) + (4 + 1) = 6 - 9 + 5 = 2$$

This shows that for whatever kind of modification I do on the tetrahedron on its number of edges, vertices and faces, **the Euler Characteristic remains constant.**

Let us now consider an hallow cube where its hole has the shape of a parallelepiped.

Let us now count the faces of this solid:

- At the sides we have 4 square faces.
- at the top and bottom we have 8 trapezoid faces, 4 at the top and 4 at the bottom.
- inside the hole of the cube we have 4 parallelogram faces

In total we have 16 faces.

The vertices are 16, 8 at the top and 8 at the bottom.

The edges are more complex to compute:

- There are 8 vertical edges
- There are 16 parallel and perpendicular edges, 8 at the top and 8 at the bottom.
- We have 8 diagonal edges, 4 at the top and 4 at the bottom.

So overall we have 32 edges.
Let us now compute the Euler Characteristic:

$$\chi = V - E + F = 16 - 32 + 16 = 0$$

This seems strange, but it is correlated to another concept: **the genus**.

1.16 Genus

The Genus (in mathematics) is stricly connected to the Euler Characteristic. The Genus is a property of a solid that depends on its shape. The definition lies behind the idea that all the platonic solid can be transformed into a bullet by applying transformations that do not require splitting the surface of the solid in any parts.

Consider now a solid that have at least one hole inside it, like a torus. To transform a sphere, a cube o any Platonic Solid into a torus we require to carve from them a hole, **causing a split**. Solid that can be transformed into one another without requiring any extra hole to be carved have the same genus. **Intuitively we can define the genus as the number of handles in the solid.**

Formally the Genus of a closed, orientable and 2-manifold surface, is the maximum number of cuts we can make along non intersection closed curves without splitting the surface in two.

1.17 Theorem regarding the Euler Characteristic and the Genus

Given the genus of our surface g, we can compute the Euler Characteristic as follow:

$$\chi = 2 - 2g$$

If we consider the carved cube from the previous example, we now know that its genus $g = 1$ (intuitively note that it has a hole inside), thus:

$$\chi = V - E + F = 2 - 2g = 2 - 2 = 0$$

In this way I can compute the Genus by just knowing the number of Faces, Vertices and Edges.

1.18 Considering open surfaces

Until now we have only considered closed surfaces (meaning without any hole that is not a handle). Consider the cube we have seen previously, removing one of the faces in its side.

Considering its number of Vertices, Edges and Faces now we have:

- 16 Vertices
- 32 Edges
- 15 Faces

Note how the number of vertices and edges has remained the same, only the faces have decreased by one. Now computing the Euler Characteristic we get:

$$\chi = V - E + F = 16 - 32 + 15 = -1$$

The reason behind this result lies in the concept of borders of a surface.

Borders of the surface

With borders of a surface we intend the number of continuous borders in the surface, meaning that in the case of the cube we have seen there is only 1 made of 4 continuous edges.

1.18.1 Euler Characteristic for open surfaces

When a surface is open the Euler Characteristic depends from the Genus g and form the number of borders b of the surface:

$$\chi = V - E + F = 2 - 2g - b$$

Intuition behind the Euler Characteristic for open surfaces

Given an open surface with a border s.t. its Euler Characteristic is $\chi = V - E + F$. Suppose now to close the border by adding a new vertex and connecting all the k vertices on the border to it. Now we want to compute with $\chi^{'}$ the Euler Characteristic of the now closed surface, consider its added vertices $V^{'}$, edges $E^{'}$ and faces $F^{'}$:

- $V^{'} = 1$ since we added one vertex to close the surface.

- $E^{'} = k$ since for each of the k edges of the border we made an edge connecting it to the new vertex.
- $V^{'} = k$ since for each edge we constructed to close the border we define a face.

Thus the new Euler Characteristic is:

$$\chi^{'} = \chi + V^{'} - E^{'} + F" = X + 1 - k + k = \chi + 1$$

So by considering the equation from χ we obtain:

$$\chi = \chi^{'} - 1$$

Since we know that the number of borders of the open cube is $b = 1$. Knowing that $\chi^{'}$ is closed, then we can compute its Euler Characteristic as $\chi^{'} = 2 - 2g$. Now by substitution we obtain the equation of the Euler Characteristic for open surfaces:

$$\chi = \chi^{'} - b = 2 - 2g - b$$

1.19 Converting Representation

We have talked about the Parametric Representation and Implicit Representation for representing surfaces, let us now consider the problem of passing from one another.

1.19.1 Implicit to Parametric

Recall that a Implicit Representation consist in defining all points of mu surface as the zeroes of a function. The process to convert it into a parametric representation consists in applying an algorithm called Marching Cube. Before introducing it we explain its more simple 2D counterpart Marching Square.

Marching Square

Suppose I have an implicit function $f : \mathbb{R}^2 \to \mathbb{R}$ s.t. $f(x, y) = c$ where $c \in \mathbb{R}$ is a constant value (usually 0). We know that implicit functions can be used to represent surfaces in a space: the point that define the shape are those whose x and y values that return the constant c when given to f. Now, finding this point is a very complex problem that is impractical to solve using bruteforce. Instead we approximate the solution by subdividing the (2D) space into a grid and for each point of the grid we assign a binary value according to a given threshold t s.t.:

$$value(x,y) = \begin{cases} 1 & if \quad f(x,y) > t \\ 0 & otherwise \end{cases}$$

So the points whose value is 1 are inside the shape, while points with value 0 are outside the shape. Now our grid is made of cells, each cell has 4 vertices. Since each vertex is one of the points we have sampled, they can have at most 2 values (1 or 0), there are a total of $2^4 = 16$ possible configurations of a cell: So for all cells we check in which configuration in the lookup table the cell lies according to the values of its vertices and draw the contour passing through it using linear interpolation.

Marching Cubes

Marching Cubes is the same idea of Marching Squares only in a 3D setting. Given an implicit function $f : \mathbb{R}^3 \to \mathbb{R}$ in the form $f(x, y, z) = c$ where c is a constant. Thus the zeros of the function are the points of the solid defined by f. Like in the bidimensional case its impractical to try every possible values of x,y and z and check if is a point of the solid. Instead we approximate the solution by subdividing the (3D) space into a tridimensional grid and for each point of the grid we assign a binary value according to a given threshold t s.t.:

$$value(x,y,z) = \begin{cases} 1 & if \quad f(x,y,z) > t \\ 0 & otherwise \end{cases}$$

So the points whose value are 1 are inside the shape, while vertices with value 0 are outside the shape. Now our grid is made of cubes, each cube is made of 8 vertices. Since each vertex can have at most 2 values (1 or 0), there are a total of $2^8 = 256$ possible configurations of a cell. So for all cells we check in which configuration in the lookup table the cell lies and draw the contour passing through it accordingly. While in 2D the contour was made of lines, in 3D the contour is made of triangles. Although there are 256 possible configurations, they can be categorized in 15 cases, each having a group of configurations that are symmetrical from each other. Finally identified in which case its our cube we determine the triangles of the contour of our mesh through linear interpolation. A big drawback of Marching Cubes is that it could happen some ill-conditioned triangles will be generated.

1.19.2 Parametric to Implicit

Recall that an implicit representation consists in a function that defines if the points in our 3D space are part of the surface or not. So the idea is to construct a 3D grid that

will approximate the implicit representation of the surface. Then for each node of the grid we check its distance to the closest triangle of the surface. Notice that this is a signed distance field, so other than the absolute distance we need to know if its outside the surface (positive sign) or inside of it (negative sign).

Given the node of the grid g and the closest point of the surface c, then the sign can be derived from the angle between the vector $g - c$ and the outer normal $n(c)$:

- if $(g - c)^T n(c) < 0$ then the point g is inside the surface.
- if $(g - c)^T n(c) < 0$ then the point g is outside the surface.

1.19.3 Signed distance field

Consider a polygonal mesh of triangular faces as our parametric representation. The computation amounts to finding the distance of each point in the 3D mesh to a uniform and adaptive 3D grid. Computing the exact distance of a grid node to a given mesh requires to calculate the distance to the closest triangle.

Optimizing using the fast marching methods

Computing the distances on the entire grid can be accelerated by fast marching methods. In a first step, the exact signed distance values are computed for all grid nodes in the immediate vicinity of the triangle mesh. After this initialization, the fast marching method propagates distances to the remaining grid nodes with unknown distance value in a breadth-first manner.

1.20 Mesh Data Structures

As of today there is no standard regarding how to store meshes on disk. Different purposes requires different data structures, because they require to store different information of my (triangular) mesh. We will show some of the most common format.

1.20.1 Face Set (STL)

The simplest format. It consist of a collection of triangles t_i that is defined by its three vertices vertices v_i. Each vertex v_i is a triple containing its coordinate in a 3D space s.t. $v_i =< x_i, y_i, z_i >$.

Since each triangle is independent from the others, then if two or more triangles share some vertex, it is stored multiple time in each triangle. So even though it is the

same vertex, in the data structure is identified as totally different vertex that shares the same coordinates. It does not store any additional information.

1.20.2 Operation I want to do on meshes

All the other formats have been made to help us execute these mesh operations:

- Access individual vertices, edges, and faces. Thus we need to be able to enumerate all elements of the mesh in an unspecified order.
- Oriented traversal of the edges of a face, which refers to finding the next (or previous) edge in a face.
- Access to the incident faces of an edge. Depending on the orientation, this is either the left or right face assuming the mash is manifold.
- Given an edge, access to its two endpoint vertices.
- Given a vertex, at least one incident face or edge must be accessible. Then for manifold meshes all other elements in the so-called one-ring neighborhood of vertex can be enumerated.

1.20.3 Shared Vertex (OBJ, OFF)

The OBJ and OFF format overcome the limitation of STL by unifying the vertices of the mesh. To do that the format define two collection:

- **Vertices:** containing all the vertices. Each vertex is a triple containing its coordinate in the 3D space.
- **Triangles:** where each triangle is a triple of vertices from the Vertices collection.

These formats do not contain any adjacent information (e.g. knowing the triangle near the current one).

1.20.4 Face-Based Connectivity

With Face-Based Connectivity we identify a set of data structures that have a vertex object and face object s.t. in each of them there are stored the following information:

- **Vertex**
  - position of the vertex in the 3D space.

  - a pointer to a face that has that vertex.

- **Face**
  - the three vertices that define the face.
  - three pointers for each triangle in its neighbors.

Note that in a Face-Based Connectivity data structures there is no collection dedicated to storing edges information. The reason for missing the edges its because for the majority of 3D algorithms the edges are not necessary for their computation.

1.20.5 Edge-Based Connectivity

With Edge-Based Connectivity we identify a set of data structure that have a vertex object, an edge object and a face object s.t. in each of them there are stored the following information:

- **Vertex**
  - the 3D coordinates of the position of the vertex
  - a pointer to an edge that has that vertex.

- **Edge**
  - the pointers to the two vertex defining it.
  - the pointers to the two faces that have in common that edge.
  - the pointers to the four edges in its neighborhood.

- **Face**
  - the pointer to an edge that uses.

Resolving the query of identifying the vertex composing a face in an Edge-Based Connectivity data structure can be done in linear time by getting the pointer to the face, then getting the pointer to one of the edges of that face, from there I get the vertex composing that edge and repeating the process to the edge next to it stored as a pointer.

Halfedge

Given an edge e that connects two vertices v_1 and v_2, it can be decomposed in two halfedges h_1 and h_2 where:

- h_1 connects from v_1 to v_2
- h_2 connects from v_2 to v_1

So e is undirected, instead h_1 and h_2 are directed.

This is done in order to sort the edges of each face and sort them in order to give all the faces of the mesh a consistent orientation: either counter-clockwise or clockwise. This works only to orientable meshes. For each halfedge we store:

- A pointer to the next edge that shares the vertex it arrives to.
- A pointer to the face that contains it.

Operations on halfedges

Halfedges permits us to get the neighborhood of a vertex in constant time. This permits us to move a face more easily, since it requires to set a new position to all the vertex composing it.

1.20.6 Halfedge-Based Connectivity

Halfedge-based connectivity its a type of data structure where we store vertices, halfedges and faces s.t.:

- Vertex
  - contains its 3D coordinates.
  - a pointer to the halfedge that contains it.
- Halfedge
  - contains the pointer to the starting vertex.
  - contains the pointer to the face that contains it.
  - contains the pointer to the halfedge that shares its vertex of arrival. It can also contains the next two halfedges in the path.
- Face
  - a pointer to one of the halfedges that compose it.

Appendix A

Elemental Geometry Concepts

A.1 Topological Space

In mathematics, a topological space is, roughly speaking, a geometrical space in which closeness is defined but cannot necessarily be measured by a numeric distance. More specifically, a topological space is a set whose elements are called points, along with an additional structure called a topology, which can be defined as a set of neighbourhoods for each point that satisfy some axioms formalizing the concept of closeness. It is the most general type of a mathematical space that allows for the definition of limits, continuity, and connectedness. Common types of topological spaces include Euclidean spaces, metric spaces and manifolds.

A.2 Polytope

In elementary geometry, a polytope is a geometric object with flat sides (faces). Polytopes are the generalization of three-dimensional polyhedra to any number of dimensions. Polytopes may exist in any general number of dimensions n as an n-dimensional polytope or n-polytope. For example, a two-dimensional polygon is a 2-polytope and a three-dimensional polyhedron is a 3-polytope.

A.3 Curve

In mathematics, a curve (also called a curved line in older texts) is an object similar to a line, but that does not have to be straight.

Formally a curve is the image of an interval to a topological space by a continuous function. In some contexts, the function that defines the curve is called a parametriza-

tion, and the curve is a parametric curve. In this article, these curves are sometimes called topological curves to distinguish them from more constrained curves such as differentiable curves. This definition encompasses most curves that are studied in mathematics; notable exceptions are level curves (which are unions of curves and isolated points), and algebraic curves (see below). Level curves and algebraic curves are sometimes called implicit curves, since they are generally defined by implicit equations.

A.4 Convex Hull

A convex hull (also known as convex envelope or convex closure) of a shape is the smallest convex set that contains it. The convex hull may be defined either as the intersection of all convex sets containing a given subset of a Euclidean space, or equivalently as the set of all convex combinations of points in the subset. For a bounded subset of the plane, the convex hull may be visualized as the shape enclosed by a rubber band stretched around the subset.

A.5 D-simplex

A d-simplex is the convex hull of d+1 point that are linearly dependent in d dimensions.

A.6 Plane

In mathematics, a plane is a two-dimensional space or flat surface that extends indefinitely. A plane is the two-dimensional analogue of a point (zero dimensions), a line (one dimension) and three-dimensional space.

A.7 Topological Characterization

How the elements are combinatorially connected.

A.8 Geometric Characteriation

Where the vertices are actually placed in space.

A.9 Open Set

An open set is a set that, along with every point P, contains all points that are sufficiently near to P (that is, all points whose distance to P is less than some value depending on P).

More generally, an open set is a member of a given collection of subsets of a given set, a collection that has the property of containing every union of its members, every finite intersection of its members, the empty set, and the whole set itself. A set in which such a collection is given is called a topological space, and the collection is called a topology. These conditions are very loose, and allow enormous flexibility in the choice of open sets. For example, every subset can be open (the discrete topology), or no subset can be open except the space itself and the empty set (the indiscrete topology).

In practice, however, open sets are usually chosen to provide a notion of nearness that is similar to that of metric spaces, without having a notion of distance defined. In particular, a topology allows defining properties such as continuity, connectedness, and compactness, which were originally defined by means of a distance.

A.10 Neighborhood of a point

The neighborhood of a point is a concept of topological space.

Given a topological space X and a point $p \in X$, then a neighbourhood of p is a subset V of X that includes an open set U containing p s.t.:

$$p \in U \subseteq V \subseteq X$$

A.11 Contour

A contour line of a function of two variables is a curve along which the function has a constant value, so that the curve joins points of equal value. More generally, a contour line for a function of two variables is a curve connecting points where the function has the same particular value.

A.11.1 Ill-conditioned triangles

A triangle is said to be ill-conditioned when one of its angle is greater than 120 deg or inferior than 30 deg. Small perturbations in the sides or angles in an ill-conditioned triangle result in large variations in its area, angles, or side lengths.

A.11.2 Non-Degenerate 3D Solid

a non-degenerate 3D solid is one that does not have any infinitely thin parts or feature such that the surface properly sperated the interior and exterior of the solid.

Appendix A

Elementar Equations

A.1 2D Shapes

A.1.1 Cicle Equation

Given the center (x_0, y_0) and the radius r^2 we define the circle by the following implicit function:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

A.1.2 Ellipse Equation

Given the center (x_0, y_0) and the minor axis (think it as the shortest radius) b and major axis (think it as the greater radius) a, we can define the ellipse by the following implicit function:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1$$

A.2 3D Solids

A.2.1 Sphere Equation

Given the radius of the sphere r, the area of the sphere is defined by the following implicit equation:

$$x^2 + y^2 + z^2 = r^2$$

Bibliography