Representing, Storing and Visualizing 3D Data

MOD 4-CHAPTER 4

Introduction

- Examples of common 3D datasets range from the very small (molecules, microscopic tissue structures, 3D microstructures in materials science), to the human scale (3D human heart, 3D face, 3D body scans) to the large (3D modeling of buildings and landscapes) and beyond (3D modeling of astrophysical data).
- It is the scale, resolution and compression of this data that determines the volume of data stored. In turn, the challenges for storing, manipulating and visualizing this data grow as the volume increases.
- In this section, we summarize various 3D representations which we classify as:
- raw data (i.e. delivered by a 3D sensing device),
- surfaces (i.e. 2D manifolds embedded in 3D space) and
- solids (i.e. 3D objects with volume).

Raw Data

- The raw output of a 3D sensor can take a number of forms, such as points, a depth map and polygons.
- In its simplest form, 3D data exists as a set of unstructured 3-dimensional coordinates called a ***point cloud***, P, where $P = \{\mathbf{v}1, \ldots, \mathbf{v}n\}$ and $\mathbf{v}i \in R3$. Typically, a point cloud of n points is stored as an $n \times 3$ array of floating point numbers or a linked list of n vertex records. **Point clouds arise most commonly in vision as the output of multiview stereo.**
- **Structured Point Cloud :** A more constrained representation may be used when point cloud vertices adhere to an underlying structure, namely a grid with arbitrary sampling. In this case, vertices are stored in an ordered $m \times n \times 3$ array and, for each point $i = 1..m$, $j = 1..n$, there is a corresponding 3D vertex $[x(i, j)\ y(i, j)\ z(i, j)]T \in R3$.

Depth Maps and Range Images: A special case of structured point cloud arises when the sampling of points in the $x - y$ plane is viewer-centered. Although often used interchangeably, we define a *range image* as a structured point cloud which arises from a perspective projection and a *depth map* as an orthogonal projection and regular sampling of 3D vertices over a 2D image plane. Both representations have the advantage that they can be represented by a 2D function $z(x, y)$. Hence, these representations require less storage than those which allow variable spacing of points in the (x, y) plane and can effectively be stored (and compressed) as an image.

Raw Data

Needle map

- Photometric shape reconstruction methods often recover an intermediate representation comprising per-pixel estimates of the orientation of the underlying surface *z(x, y)*. In graphics this is known as a *bump map*.

Polygon Soup

- A *Polygon soup* is, in some senses, analogous to point cloud data, but comprises polygons rather than vertices. More precisely, it is a set of unstructured polygons, each of which connect vertices together but which are not themselves connected in a coherent structure such as a mesh.
- This sort of data may contain errors such as: inconsistently oriented polygons; intersecting, overlapping or missing polygons; cracks (shared edges not represented as such); or T-junctions.
- This causes problems for many applications including rendering, collision detection, finite element analysis and solid modeling operations.

Surface Representations

- The vast majority of geometric algorithms in computer vision and graphics operate on representations of 3D data based on surfaces. Of these representations, by far the most common is the triangular mesh.
- **Triangular Mesh**
- The most common surface representation comprises 3D vertices, connected together to form triangular faces, which in turn represent or approximate a 2D manifold embedded in 3D space.
- A number of categorizations are possible here. An important distinction is whether the mesh is closed (i.e. the surface completely encloses a volume) or open (i.e. the mesh contains “boundary” edges that are used by only one triangle).
- Often, mesh vertices are augmented with *texture coordinates*, also known as UV coordinates. These are most often 2D coordinates which describe a mapping from the surface to a 2D planar parameterization.

Surface Representations

Quadrilateral Mesh

One commonly used polygon mesh is based on quadrilateral polygons (sometimes known as a *quadmesh*). Quadmeshes can be easily converted to a triangular mesh by diagonally subdividing each quadrilateral. Quadrilateral meshes are preferred to triangular meshes in a number of circumstances

Subdivision Surfaces

Subdivision surfaces are used to represent a smooth surface using a low resolution base mesh and a subdivision rule (or *refinement scheme*). When applied recursively, the subdivided surface tends towards the smooth surface. The *limit* subdivision surface is the surface produced when the refinement scheme is iteratively applied infinitely many times.

Morphable Model

A space efficient representation, which can be used to approximate members of a class of surfaces (such as human faces or automobiles , is the *morphable model*. This is a compact statistical representation learnt from training samples.

Parametric Surface

Parametric Surface

A parametric surface is one which is defined by parametric equations with two parameters, as follows:

$x = fx(u, v),$

$y = fy(u, v),$

$z = fz(u, v).$

For example, the radius r sphere example given above can be described in terms of spherical coordinate parameters:

$x = r \sin \vartheta \cos \alpha,$

$y = r \sin \vartheta \sin \alpha,$

$z = r \cos \vartheta.$

A surface in such a form is easy to evaluate and, if the parametric equations are differentiable, it is straightforward to calculate differential properties of the surface.

Comparison of Surface Representations

- The best choice of surface representation is application dependent. Table 4.1 summarizes some of the broad strengths and weaknesses of the surface representations discussed above. Generally speaking, polygonal meshes are the most widely used representation and the standard rendering pipeline, such as OpenGL, is optimized for their use. Subdivision surfaces are popular in animation and interactive editing applications. Parametric surfaces are widely used in CAD, as are implicit surfaces, which also find use in mathematical or scientific applications.

Table 4.1 Comparison of surface representations

	Polygonal mesh	Subdivision surface	Morphable model	Implicit surface	Parametric surface
Accuracy	✗	✓	✗	✓	✓
Space efficiency	✗	✓	✓	✓	✓
Display efficiency	✓	✓	✓	✗	✓
InterSect. efficiency	✗	✗	✗	✓	✗
Intuitive specification	✗	✗	✓	✗	✓
Ease of editing	✗	✓	✓	✗	✗
Arbitrary topology	✓	✓	✗	✗	✗
Guaranteed continuity	✗	✓	✗	✓	✓

Solid-Based Representations

Common areas of application for solid-based representations include:

1. **Medical Imaging** [62]. Imaging modalities such as MRI and CT are volumetric in nature. Analyzing and visualizing such data necessitates volumetric representations.

2. **Engineering**. Components must be designed not only so that they fit together, but also so that their structural properties, determined by their 3D shape, meet

specifications.

3. **Scientific visualization** [46]. Volumetric data arises in fields ranging from oceanography to particle physics. In order for humans to interact with and draw

inferences from such data, it must be possible to visualize it in a meaningful way.

4. **Finite-element analysis** [79]. A numerical technique for finding approximate solution of partial differential equations and integrals.

5. **Stereolithography** [2]. A process in which an ultraviolet laser is used to trace out cross-sections through a 3D object, building up layers of resin until a complete object has been fabricated. Commonly known as “3D printing”.

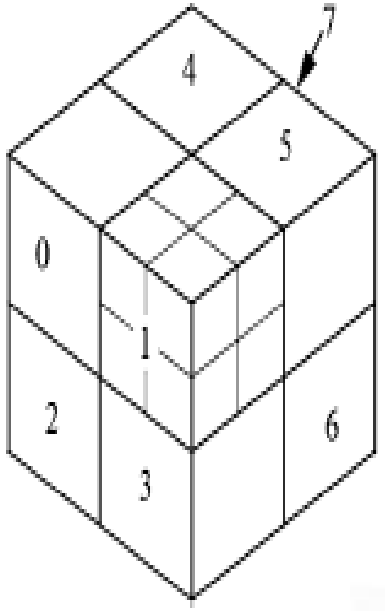
6. **Interference fit** [35]. Designing object shapes so that two parts can be held together by friction alone.

Voxels

- Voxels are the 3D analog of pixels (the name is a contraction of “volumetric pixel”).
- They are sometimes also referred to as a spatial occupancy enumeration. Each voxel corresponds to a small cubic volume within space. At each voxel, a boolean is stored which indicates the presence or absence of the volume at that point.
- **There are two common ways to visualize voxel data: direct volume rendering or conversion to polygonal surface for traditional mesh rendering.**
- In direct volume rendering, voxels are projected to a 2D viewplane and drawn from back to front using an appropriate color and opacity for every voxel.

A space efficient representation for voxel data is the *octree*. This uses adaptive resolution depending on the complexity of the surface throughout the volume. The key idea is that voxels which are fully occupied or fully empty are not subdivided while partially occupied voxels are. This continues until either all voxels are occupied or empty, or the smallest allowable voxel size is reached. Subdividing a voxel into smaller cubes results in 8 smaller voxels. This representation can be stored in a tree structure in which nodes have 8 children.

Fig. 4.5 The 8 voxels produced by an octree voxel subdivision



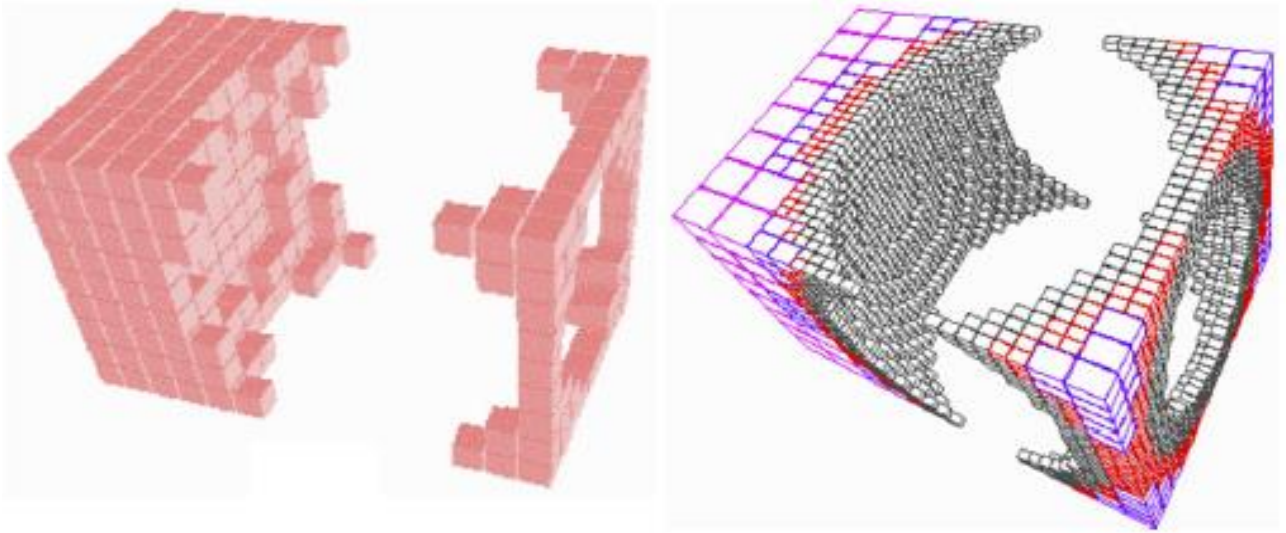
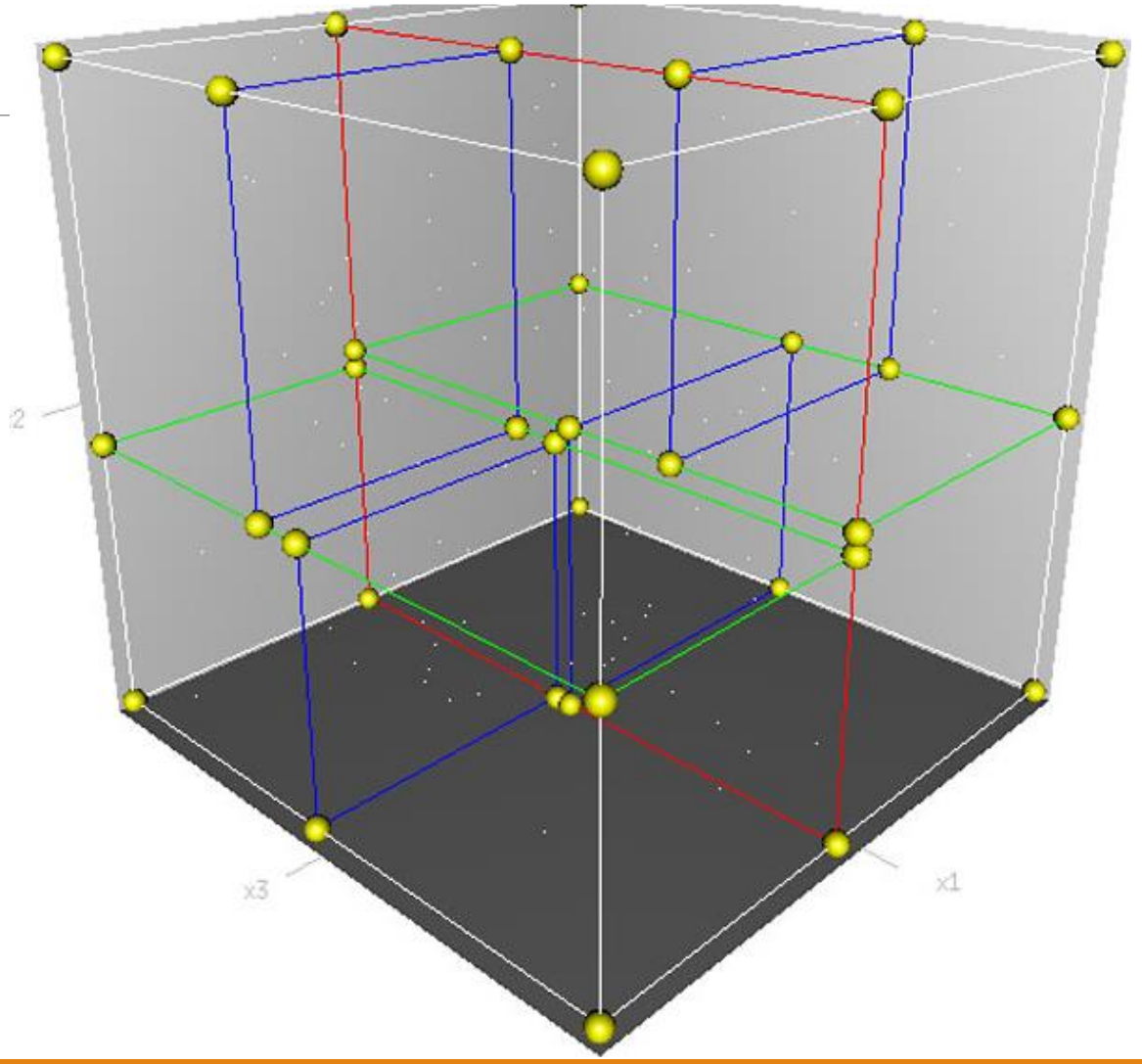


Fig. 4.6 A volume represented using equally sized voxels (*left*) and the octree adaptive representation (*right*). Figure courtesy of lecture notes in computer graphics, Department of Computer Science, University of York

K-d Trees

- *K-dimensional trees*, or *K*-d trees for short, generalize the notion of space partitioning to arbitrary, axis-aligned planes. Whereas the octree uses a regular subdivision of space, a *K*-d tree is a data structure whose interior nodes represent a partition of space along a plane which is orthogonal to one of the axes.
- A *K*-d tree is stored in a binary tree where interior nodes represent a partition over an axis-aligned plane and leaf nodes represent a volume in space. As well as a volumetric representation for shape, *K*-d trees are widely used for organizing points to speed up multidimensional search [3], such as nearest-neighbor searches.
- Fig. 4.7 shows a *K*-d tree constructed over a 3-dimensional volume (initially comprising the white boxed region). The first subdivision is shown by the red plane. Each of the two subdivided regions are then split by the green planes and finally the blue planes yield 8 leaf volumes.



Binary Space Partitioning

- Binary Space Partitioning (BSP) further generalizes the notion of space partitioning to arbitrary subdivisions.
- Unlike the *K*-d tree, the cutting planes can be any orientation, thus a *K*-d tree is an axis-aligned BSP tree. A volume (or in 2D, an area) is represented by a series of binary tests determining which side of a plane (or in 2D, a line) a point lies on. This representation leads to a binary tree structure with the leaf nodes indicating whether a point lies inside or outside the volume and branching at nodes representing the outcome of the binary test. This is a simple and elegant representation on which boolean operations and point classifications are easy to compute, though it potentially results in a high memory overhead.

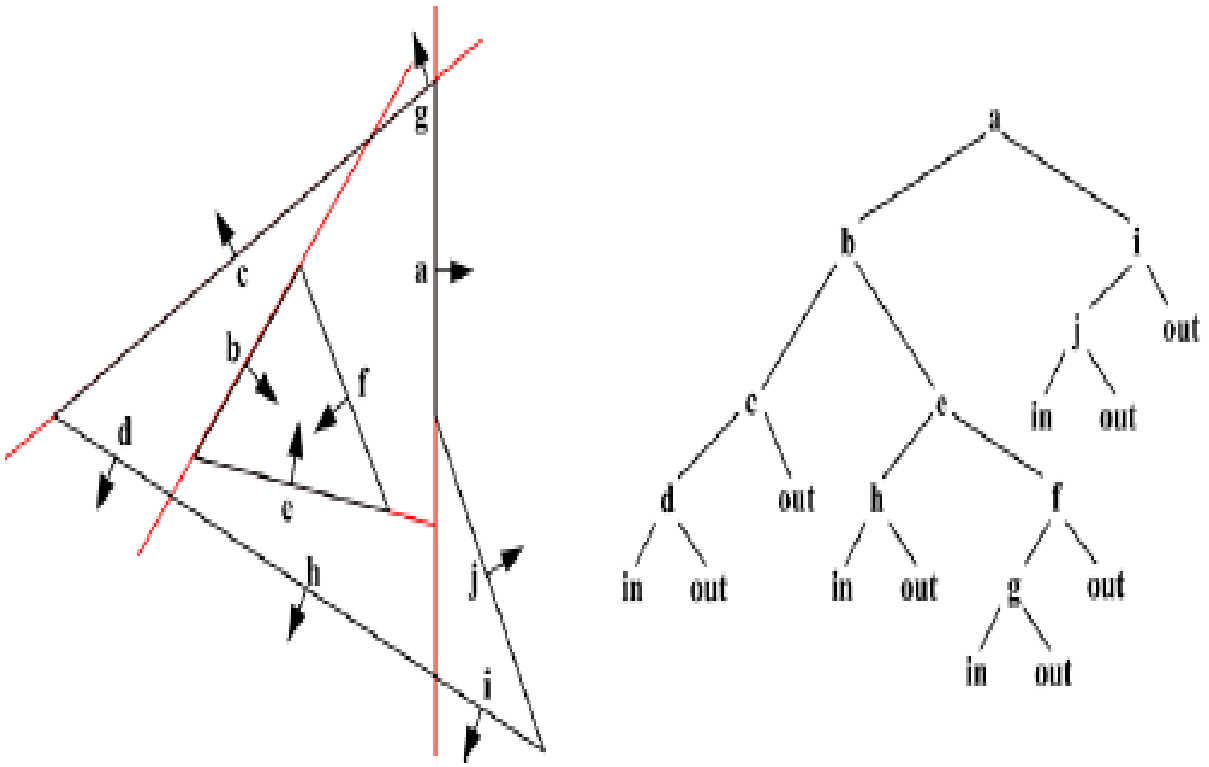
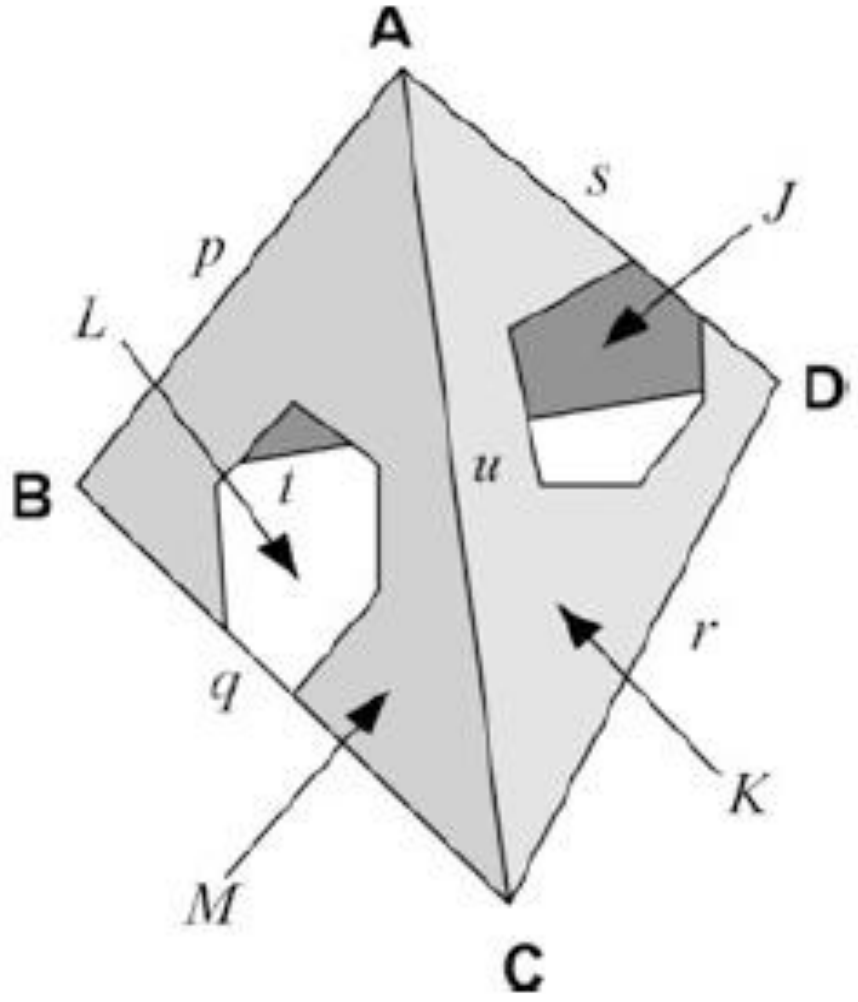


Fig. 4.8 A 2D example of a Binary Space Partitioning tree

Boundary Representations

- Boundary representations (known as *B-reps*) describe solids by defining the boundary between solid and non-solid. They are widely used in Computer Aided Design.
- B-reps are composed of two parts. The first is the *topology*. This describes how the surface elements are connected and is specified in terms of faces, edges and vertices. The second part is the *geometry*, which specifies the shape of a face, edge or vertex in terms of surfaces, curves and points.
- A face is associated with a bounded portion of a surface and an edge with a bounded piece of a curve.
- The topology of a B-rep is stored in a data structure, most commonly the *winged-edge* which stores a face, edge and vertex table. Each edge stores pointers to its two vertices, the two faces adjacent to the edge and the four edges adjacent to both the edge and the adjacent faces. Each vertex and face stores a pointer to one of its adjacent edges. Adjacency relationship can therefore be computed in constant time.



Vertex table

A	x_A	y_A	z_A
B	x_B	y_B	z_B
C	x_C	y_C	z_C
D	x_D	y_D	z_D

Edge table

p	**A**	**B**
q	**B**	**C**
r	**C**	**D**
s	**D**	**A**
t	**B**	**D**
u	**A**	**C**

Face table

J	p	s	t
K	r	s	u
L	r	t	q
M	p	q	u

Fig. 4.10 An example of the topology of a solid described by a B-rep

Mesh Data Structures

- To apply any processing to a mesh, such as rendering, manipulation or editing, we must be able to retrieve elements of the mesh and discover adjacency relationships.
- The most common such queries include: finding the faces/edges which are incident on a given vertex, finding the faces which border an edge, finding the edges which border a face, and finding the faces which are adjacent to a face.
- Mesh data structures can be classified according to how efficiently these queries can be answered. This is often traded off against storage overhead and representational power. There are a large number of data structures available for the purpose of representing meshes. Some of the most common are summarized below.

Face list. A list of faces, each of which stores vertex positions. There is no redundancy in this representation, but connectivity between faces with shared vertices is not stored. Adjacency queries or transformations are inefficient and awkward.

Vertex-face list. A commonly used representation which is space efficient. It comprises a list of shared vertices and a list of faces, each of which stores pointers intothe shared vertex list for each of its vertices. Since this is the representation used in most 3D file formats, such as OBJ, it is straightforward to load archival data into this structure.

Vertex-vertex list. A list of vertices, each containing a list to the vertices to which it is adjacent. Face and edge information is implicit and hence rendering is inefficient since it is necessary to traverse the structure to build lists of polygons.

Edge list. An edge list can be built from a vertex/face list in *O(M)* time for a mesh of *M* faces. An edge list is useful for a number of geometric computer graphics

algorithms such as computing stencil shadows.

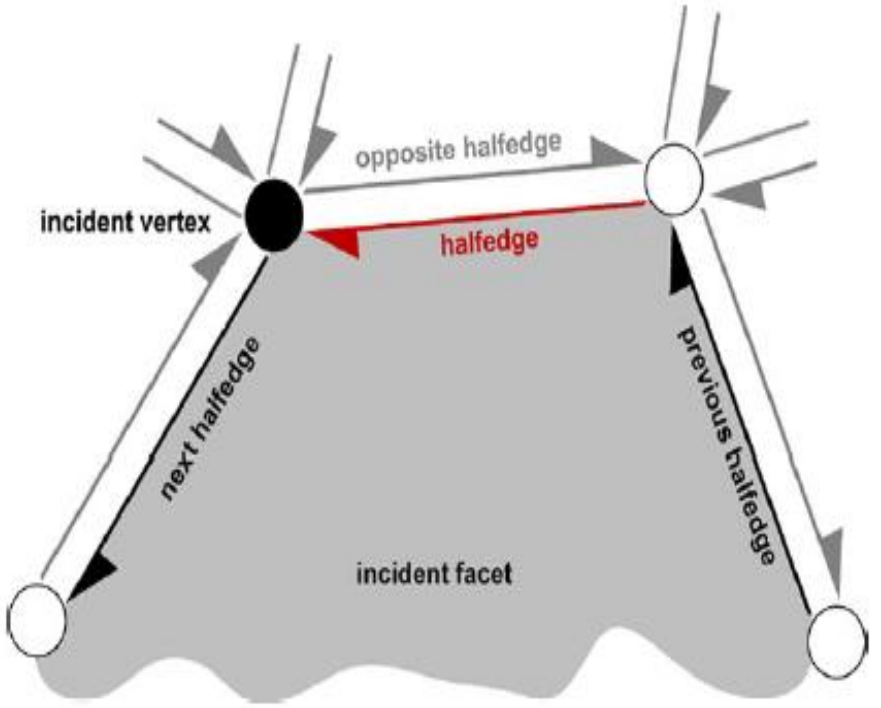
Winged-edge. The best known boundary representation (B-rep). Each edge stores pointers to the two vertices at their ends, the two faces bordering them, and pointers to four of the edges connected to the end points. This structure allows edge-vertex and edge-face queries to be answered in constant time, though other adjacency queries can require more processing.

Halfedge. Also known as the FE-structure [78] or as a doubly-connected edge list (DCEL) [4], although note that the originally proposed DCEL [43] described a different data structure. The halfedge is a B-rep structure which makes explicit the notion that an edge is shared by two faces by splitting an edge into two entities. It is restricted to representing manifold surfaces.

Adjacency matrix. A symmetric matrix of size $N \times N$, which contains 1 if there is an edge between the corresponding vertices. Highly space inefficient for meshes but allows some operations to be performed by applying linear algebra to the adjacency matrix.

The halfedge structure allows all adjacency queries to be computed in constant time, while requiring only a modest overhead in storage requirements. For this reason, it is a good choice as a general purpose data structure for mesh processing.

Halfedge Structure



- The halfedge data structure comprises vertices, faces and “halfedges”. Each edge in the mesh is represented by two halfedges. Conceptually, a halfedge is obtained by dividing an edge down its length. Figure 4.11 shows a small section of a mesh represented using the halfedge structure.
- Halfedges store pointers to the following:
- 1. The next halfedge in the facet (and so they form a circularly linked list around the face).
- 2. Its companion “opposite” halfedge.
- 3. The vertex at the end of the halfedge.
- 4. The face that the halfedge borders.
- Note that halfedges can be linked in clockwise or counterclockwise direction about a face, but this must be consistent over the mesh.

Subdivision Surfaces

- Subdivision surfaces are based on an iterative refinement of a base mesh according to a refinement scheme.
- Each iteration of the subdivision results in a mesh which is smoother than the previous.
- Refinement schemes are divided into two classes: *approximating* and *interpolating*.
- Interpolating schemes retain the positions of the original base mesh as part of the subdivided surfaces.
- In contrast, approximating schemes are free to adjust the positions of these vertices.
- Approximating schemes generally lead to smoother surfaces but allow less precise control for designers who wish to specify the exact position of control vertices. In the context of 3D imaging, subdivision surfaces provide an ideal representation for storing and interacting with 3D data, due to their space efficiency and ease of editing.

Compression and Levels of Detail

- storage requirements for 3D data can be reduced in two ways: storing a 3D representation in less space (compression) or deriving a lower resolution representation that approximates the original data.
- Techniques for 3D data compression can be categorized into three classes of approach:
- **Mesh-based methods.** These methods involve traversal of a polygonal mesh and encoding of the mesh structure in a manner that can be compressed. An example of such an approach is *topological surgery*
- **Progressive and hierarchical methods.** We have already seen subdivision surfaces that fall into this category. Another important approach is the *compressed progressive mesh* [24]. In the original progressive mesh, a mesh is encoded as a form of reverse simplification.
- **Imaged-based methods.** A 3D surface may be represented in image space. This can either be native to the representation (in the case of a range image or bump map) or via a process known as surface parameterization or surface flattening