Computer Graphics: Geometric Modeling

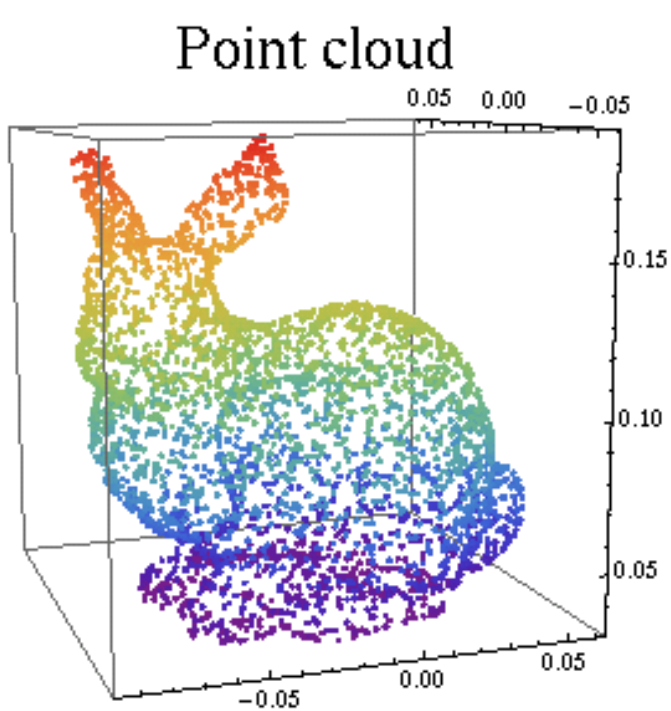
Table of contents

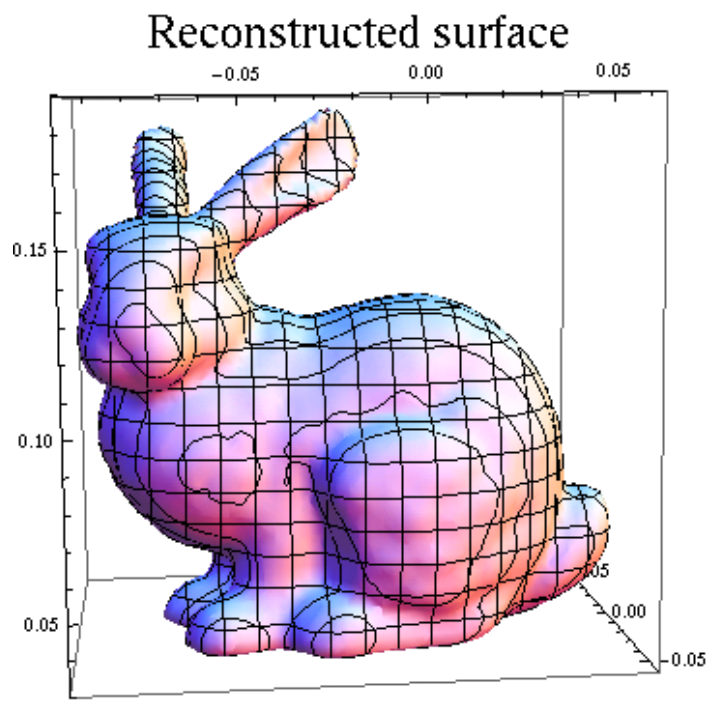
Range data

- Range image: image storing depth information at each pixel instead of color
- Acquisition from range scanner
- Registration: combination of several range images

Discrete point-set

- Unordered set of 3D point samples
- Usually converted to other representations: triangle mesh, implicit surface, for editing or processing
- Can be rendered directly with splatting





Discrete point-set

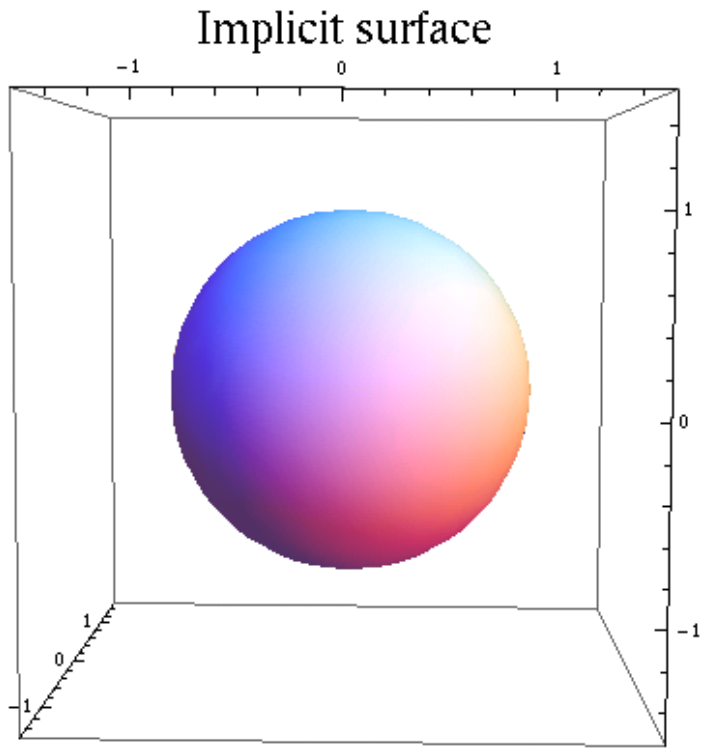
- Unordered set of 3D point samples
- Sources:
  - Often obtained from range scanner after registration of several images
  - Random sampling
  - Particle system
- Can associate information to each sample: normal vector, color, ...
- Potentially contains noise
- Usually converted to other representations: triangle mesh, implicit surface, for editing or processing

Polygon soup

- Unstructured set of polygons
- Created by combination of range images, by interactive system
- Sufficient for visualization but require repair if used for further editing or processing

Implicit Surfaces

- Definition: given a function $f : R^3 \rightarrow R$, an implicit surface S is obtained by considering (usually) the 0 isolevel of f: $S = f^{-1}(0) = \{(x, y, z) \in R^3 : f(x, y, z) = 0\}$
- Example: a sphere of radius r is defined by the set: $\{(x, y, z)^3 \in R : r^2 - (x^2 + y^2 + z^2) = 0\}$



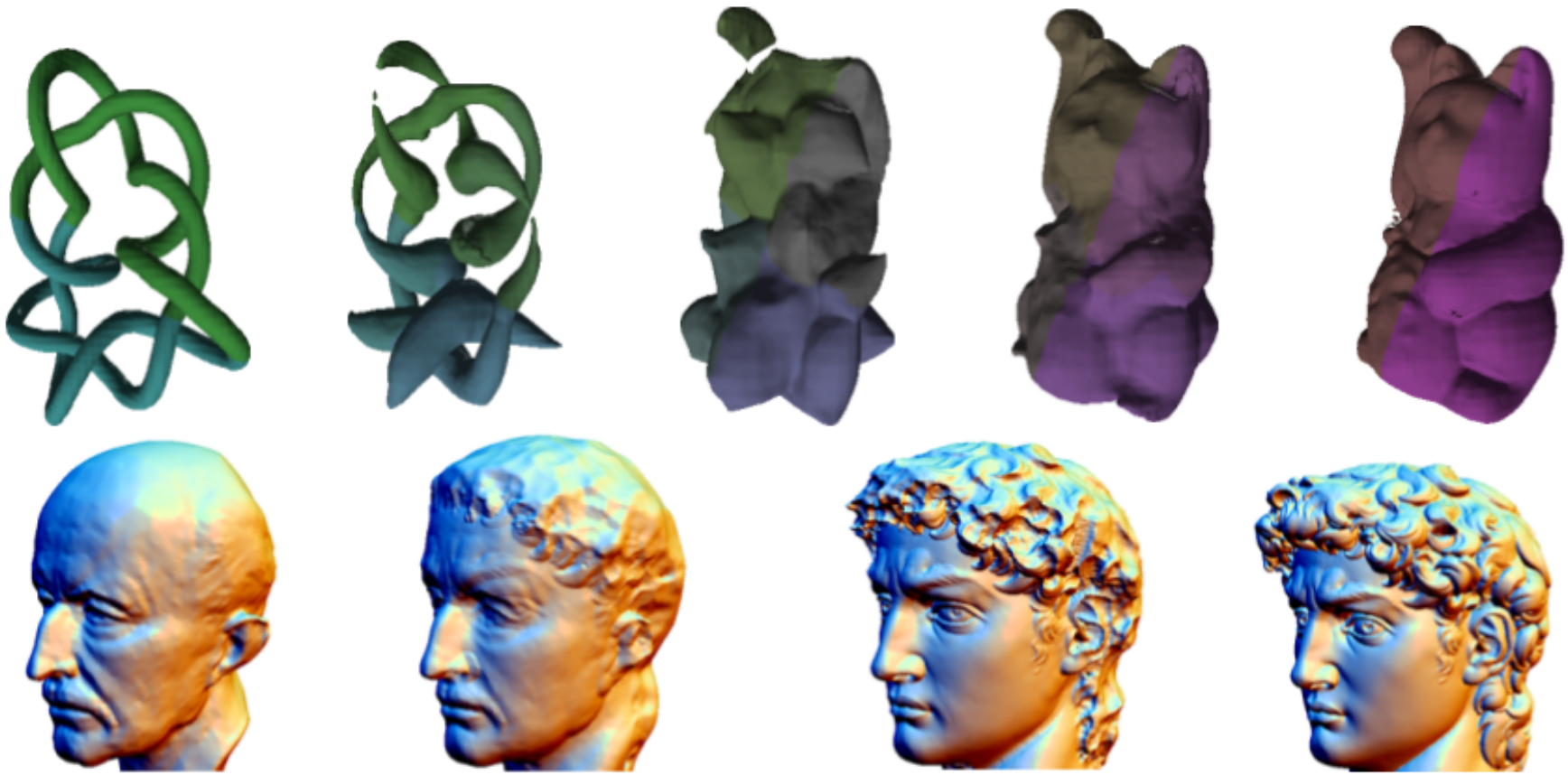
Properties

- A normal vector to the surface is given by the gradient of the function: $\nabla f(x, y, z) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$
- Example: a vector normal to the sphere $(f(x, y, z) = r^2 - (x^2 + y^2 + z^2))$ is given by: $\nabla f(x, y, z) = (-2x, -2y, -2z)$
- Other properties of the implicit surface, e.g. curvature information, can be obtained from the corresponding function

Animation of implicit

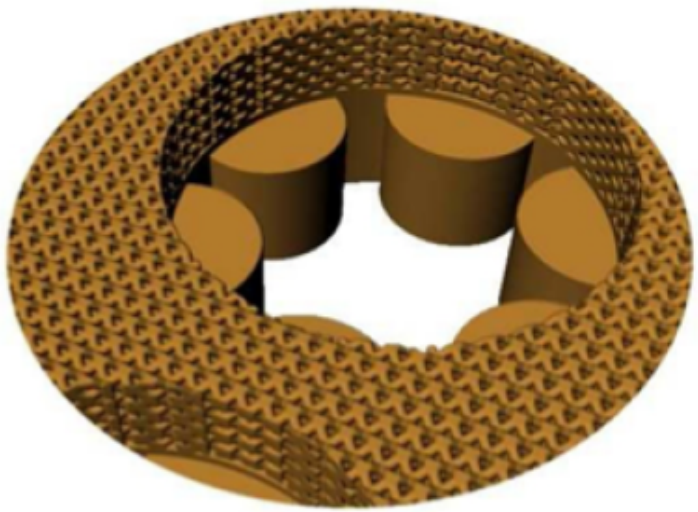
Metamorphosis from one implicit shape ($f_1 = 0$) to another ($f_2 = 0$) is easy

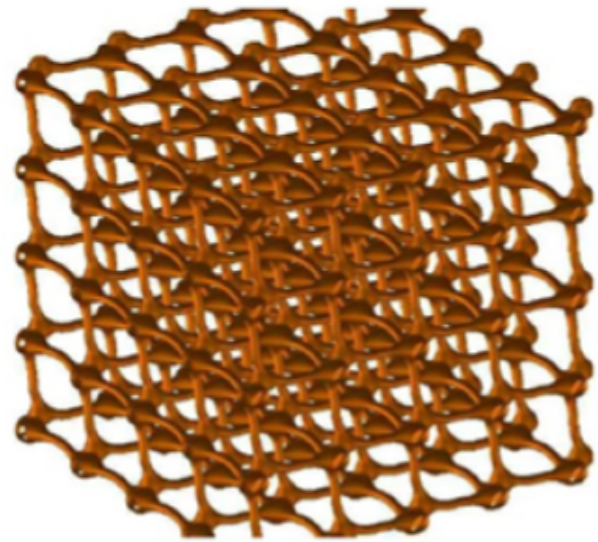
$$f(x, y, z, t) = (1 - t)f_1(x, y, z) + tf_2(x, y, z), t \in [0, 1]$$



Microstructure, tiling and repeating patterns

Implicit surfaces can be used to represent tiling and repeating patterns (microstructure)





Implicit Surfaces

- Pros
  - Easy to check if a point is on the surface
- Cons
  - Difficult to sample points on the surface
  - Rendering needs to be done with more complex algorithms (than polygon meshes): ray-tracing, polygonization (meshing) with Marching Cubes algorithm (or variants)

Distance fields

- A subset of the set of implicit surfaces where f represents the signed Euclidean distance to $f^{-1}(0)$;
- Example: for a sphere $r - \sqrt{x^2 + y^2 + z^2}$ represents the distance from $p(x, y, z)$ to the surface of a sphere of radius r;
- Practically, analytic forms exist only for simple expressions (mostly quadratic polynomials); otherwise a numerical procedure is required;
- Fast algorithm for computing the distance to a triangle mesh.

Distance fields: Properties

Distance fields are interesting because of their properties:

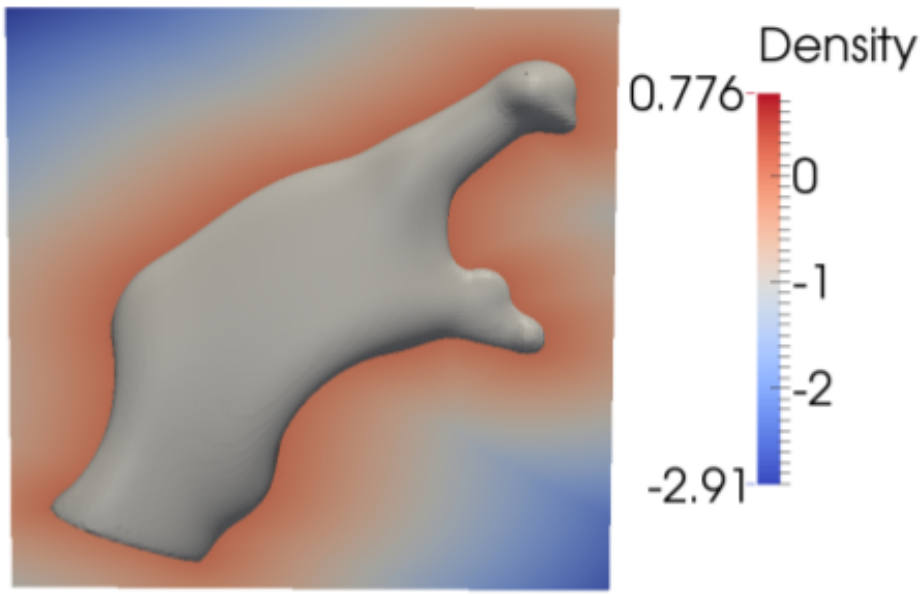
- Fast rendering by ray-marching (sphere tracing);
- Parameterization of material in heterogeneous objects and microstructures;
- Morphing and animation;
- Level-set methods

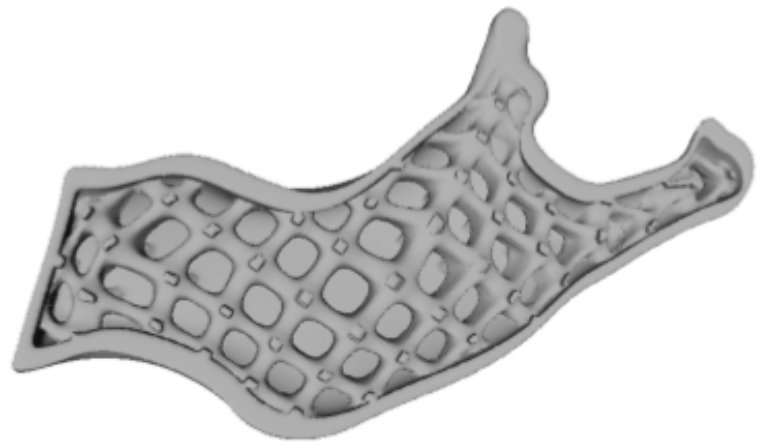
Distance fields: Operations

- Rigid body transformations maintain the distance property;
- Unfortunately, most of operations (as common as for example: non-uniform scaling) deform the field, and it needs to be re-initialized to a distance field by a numerical procedure.

Distance fields: examples

Example: signed distance to a scanned bone, and filling with microstructures, whose size is parameterized by the distance to the boundary.





Distance fields: examples

Complex SDF scenes by I. Quilez

Distance Fields

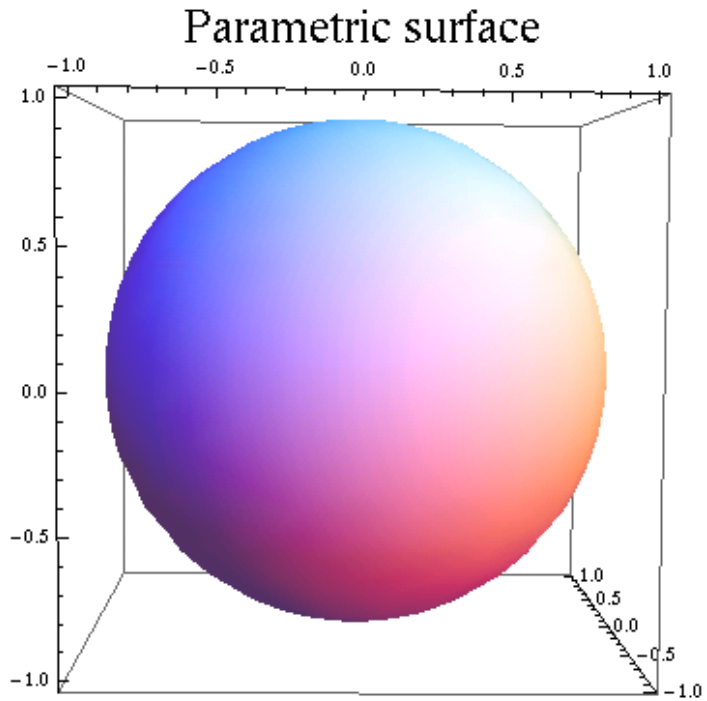
- Pros
  - Easy to check if a point is on the surface;
  - Efficient rendering algorithm (ray-marching);
  - Intuitive parameter for further modeling operations
- Cons
  - Difficult to sample points on the surface;
  - Limited number of primitives with analytical form; limited number of operations;
  - Often requires numerical procedure to convert a given field to a distance field;
  - Lack of smoothness on the medial axis

Parametric surfaces

- Surfaces are two-dimensional parameterizations: $p(u, v) = (x(u, v), y(u, v), z(u, v)$, where u, v are the parameters (often $u, v \in [0, 1]$) and $x, y, z : R^2 \rightarrow R$ are functions
- Example: a sphere of radius r and centre $(0, 0, 0)$ can be represented in parametric form by: $p(u, v) = r(cos(u)cos(v), sin(u)cos(v), sin(v))$ with $(u, v) \in [0, 2\pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}]$

Example

- Example: a sphere of radius r and centre $(0, 0, 0)$ can be represented in parametric form by:
  $p(u, v) = r(cos(u)cos(v), sin(u)cos(v), sin(v))$ with
  $(u, v) \in [0, 2\pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}]$



Properties

- Tangents to a parametric surface are obtained by taking the derivatives with respect to the parameters u and v: $T_u = \frac{\partial p}{\partial u}$ and $T_v = \frac{\partial p}{\partial v}$
- A vector normal to the surface is obtained by taking the cross product of the tangent vectors: $n(u, v) = T_u(u, v) \times T_v(u, v)$
- Example: using the previous parameterization for the sphere of radius r. $T_u(u, v) = r(-sin(u)cos(v), cos(u)cos(v), 0)$. $T_v(u, v) = r(-cos(u)sin(v), -sin(u)sin(v), cos(v))$. Finally, $n(u, v) = r^2 cos(v)(cos(u)cos(v), sin(u)cos(v), sin(v))$ i.e. the same direction as $\vec{OP}$ where O is the sphere centre and P is a point on the surface.

Tensor product surfaces

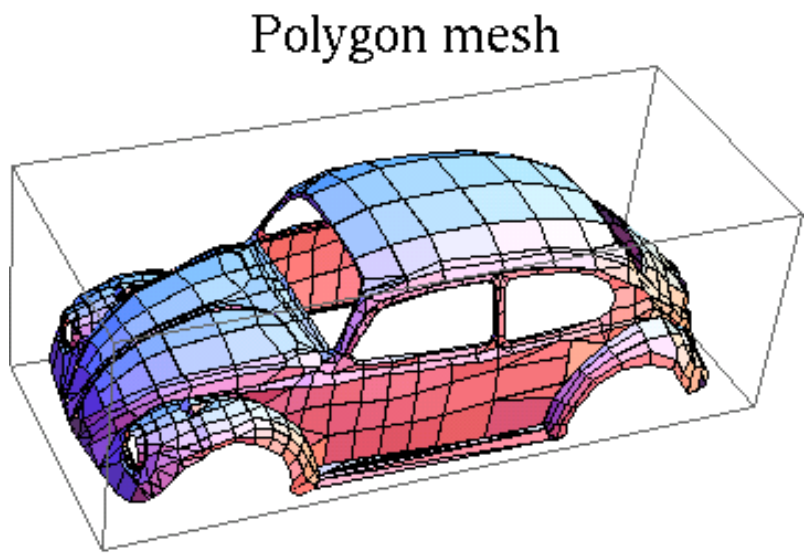
- Parametric surfaces are extremely popular in Computer Aided Geometric Design
- Most popular are tensor product surfaces: Bezier surfaces, B-spline patches, NURBS
- These are the topics of lecture 12

Parametric surfaces

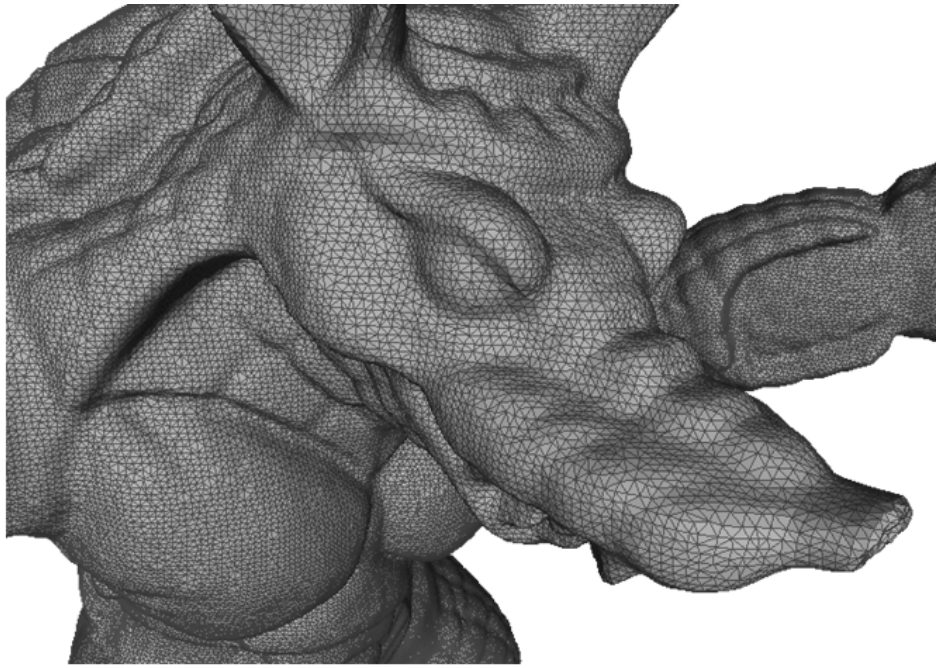
- Pros
  - Easy to sample points on the surface
- Cons
  - Difficult to determine if a given point is on the surface or not
  - If the surface is closed, difficult to check if a point is inside or outside

Polygon meshes

- A collection of vertices, edges, and faces defining a polyhedral object (i.e. a solid with straight edges and flat faces)
- Faces consist usually in triangles, quadrilaterals or simple convex polygons (because it simplifies rendering)
- Polygon meshes are widely used in computer graphics
- Typical operations on polygon meshes include: deformation, simplification, smoothing, combination by set operations



Polygon meshes



Mesh representations

There are various methods for representing polygon meshes:

- Face-vertex lists: Vertices coordinates (and attributes such as normal, texture coordinates, color) are stored in a list; faces of the mesh point to the vertices that make the face
- Winged-edge datastructure: Each edge points to two vertices, two faces and the four edges that touch it
- Halfedge datastructure: each halfedge has pointers to its opposite halfedge, the next halfedge of the face, its adjacent face, the vertex it points to

Each representation has its own advantages and inconvenients. e.g. Winged-edge and halfedge representations allow for neighborhood queries in constant time at the expense of memory usage. Face-vertex list is simple; it is often used for saving meshes on disk and rendering.

Normals to triangle mesh

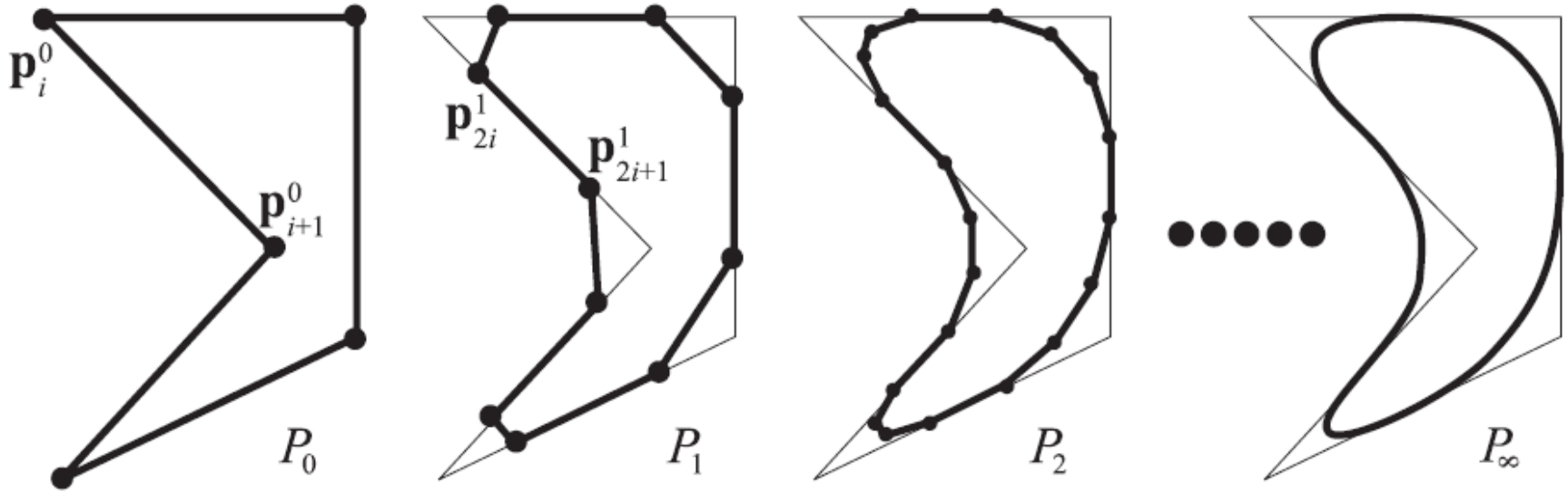
Most common polygon meshes in computer graphics consist of triangles. We distinguish between face (triangle) normals and vertex normals.

- Face normal: consider the triangle made of vertices (p_0, p_1, p_2). Assuming the vertices are enumerated in counter-clockwise order, a unit normal to the triangle is obtained by computing a normal: $\vec{n} = p_0\vec{p}_1 \times p_0\vec{p}_2$ and normalizing it: $\vec{n_u} = \frac{\vec{n}}{|\vec{n}|}$
- Vertex normal: there are several methods to compute a vector normal at a vertex. One method consists in:
  - Identify the set of triangles adjacent to a vertex
  - Compute a normal to each adjacent triangle
  - Take the mean vector and normalize it

  Variant of this method weights each normal by the area of the triangle.

Subdivision curves

Idea: represent a smooth curve by recursive subdivision and refinement of a coarse one



Example: Chaikin's subdivision (corner cutting)

Subdivision surfaces

The same idea extends to surfaces:

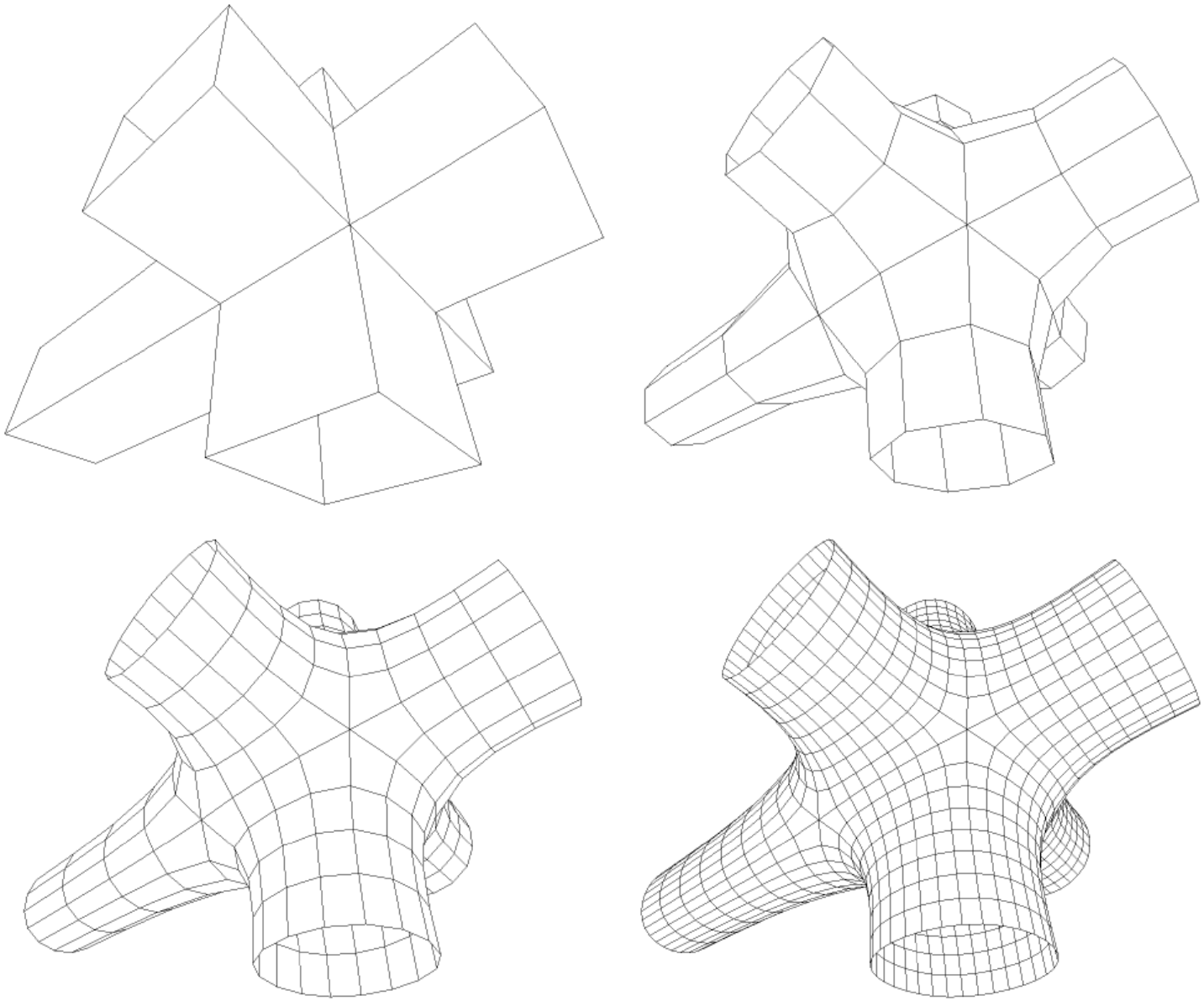
- Starting from a coarse polygon mesh
- Recursively subdivide each face of the polygon to create smaller faces, and update the vertices positions

Subdivision schemes (or refinement schemes) are the set of rules used when subdividing a surface. They are divided in two categories (similar to splines):

- Interpolating schemes: Butterfly, Midedge, ...
- Approximation schemes: Catmull-Clark, Doo-Sabin, Loop ...

Subdivision surfaces are essential in editing and modeling in the game and animated movie industries

Example: Catmull-Clark subdivision



Solid modeling

- Closed surfaces partition the space in two connected components: the bounded interior and the unbounded exterior
- Solid modeling representations try to capture this volumetric information (interior with surface)
- While representing an object surface is sufficient for rendering purpose, solid representations are needed for further modeling and simulations (e.g. finite element methods)

Implicit surfaces

- Implicit surfaces can be naturally extended to represent volumes by considering instead sets of the form: $V = \{(x, y, z) \in R^3 : f(x, y, z) \geq 0\}$. Note the $=$ replaced by $\geq$ in the set definition.
- All properties discussed earlier are valids.
- Rendering can be done by extracting the surface (with ray-tracing or Marching Cubes type algorithms), by volume rendering (volumetric ray-casting) or by extracting a tetrahedral approximation of the interior (Marching Tetahedra, Delaunay based algorithms)
- This representation is quite general and allows for physically impossible objects

CSG operations

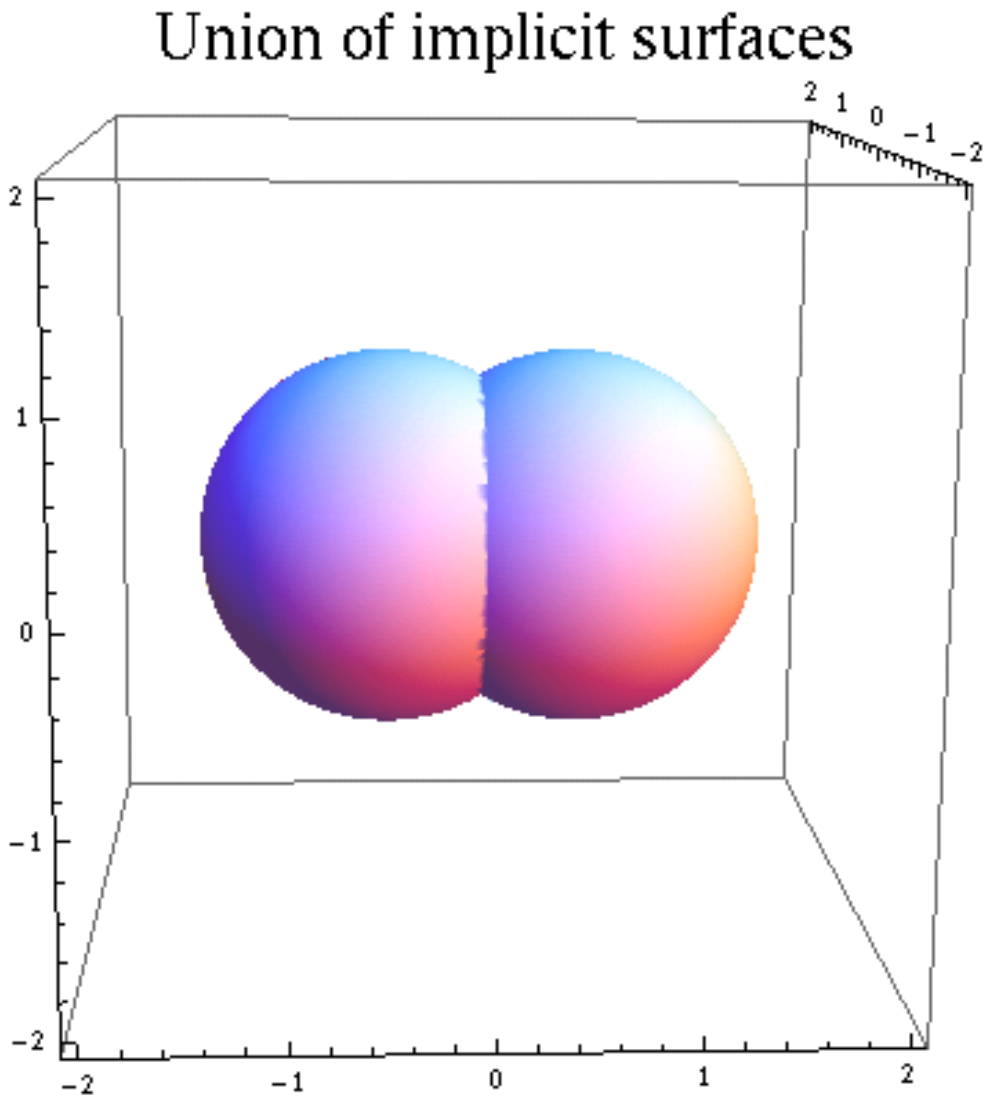
Standard set (also called Boolean or CSG) operations: union, intersection, difference, are easy to implement

- Union: $S_f \cup S_g = \{(x, y, z) : h(x, y, z) \geq 0\}$, where $h(x, y, z) = max(f(x, y, z), g(x, y, z))$
- Intersection: $S_f \cap S_g = \{(x, y, z) : h(x, y, z) \geq 0\}$, where $h(x, y, z) = min(f(x, y, z), g(x, y, z))$
- Difference: $S_f \setminus S_g = \{(x, y, z) : h(x, y, z) \geq 0\}$, where $h(x, y, z) = min(f(x, y, z), -g(x, y, z))$

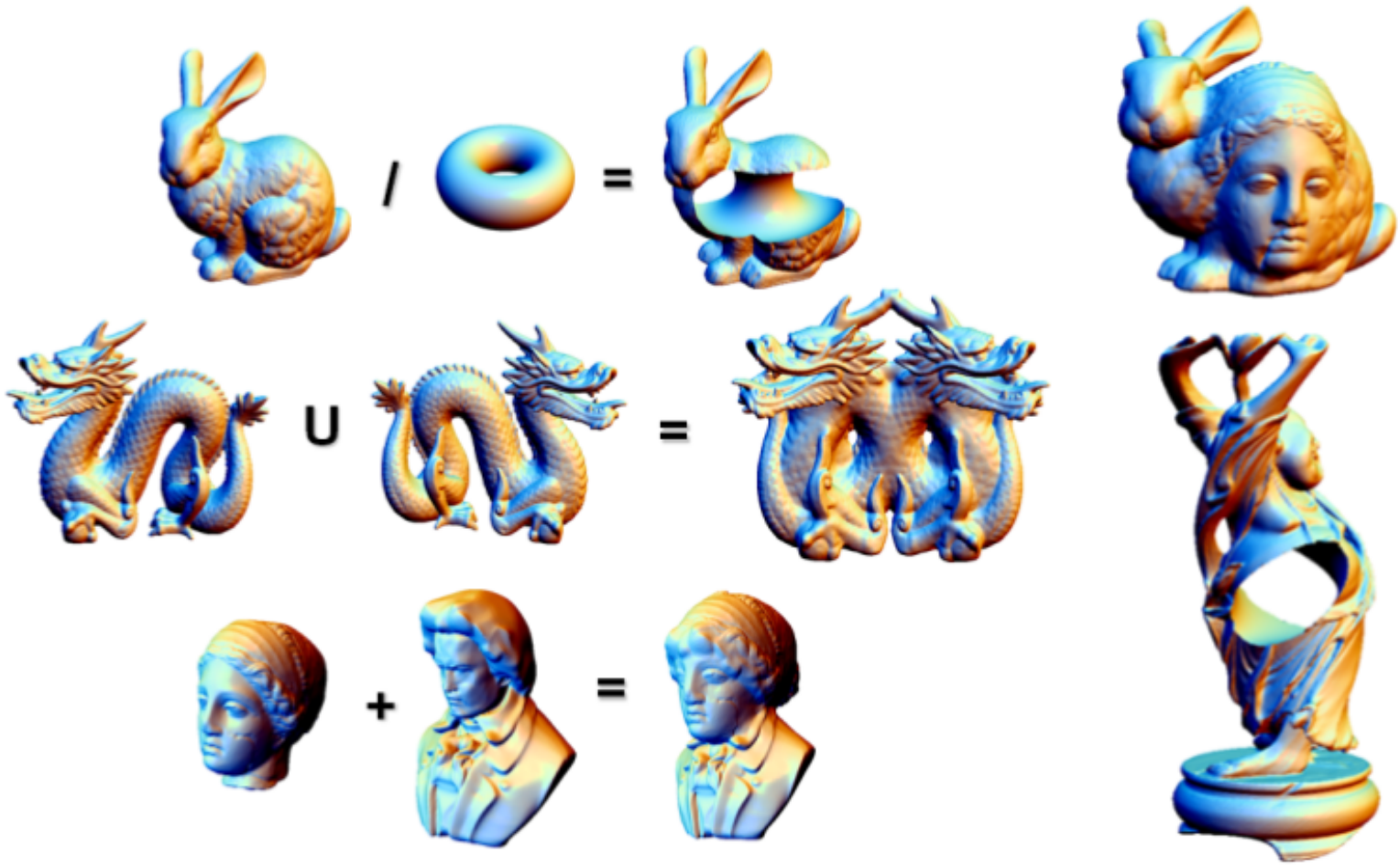
where S_f and S_g are the solids corresponding to the functions f and g.

Example: the union of two spheres of radius 1 and centered in $(-0.5, 0, 0)$ and $0.5, 0, 0$ is defined by $S = \{(x, y, z) \in R^3 : h(x, y, z) \geq 0\}$, where $h(x, y, z) = max(1.0 - ((x + 0.5)^2 + y^2 + z^2), 1.0 - ((x - 0.5)^2 + y^2 + z^2))$

Example: union of two spheres



Example: CSG and other operations on implicits

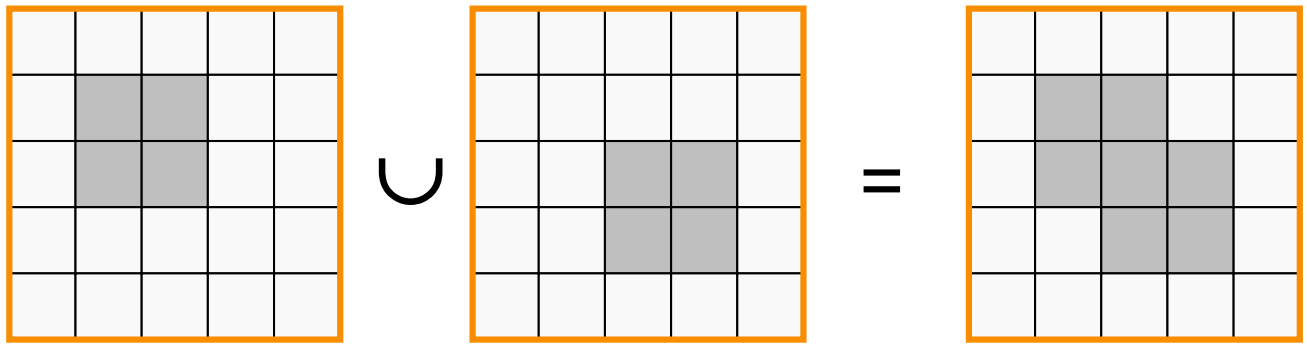


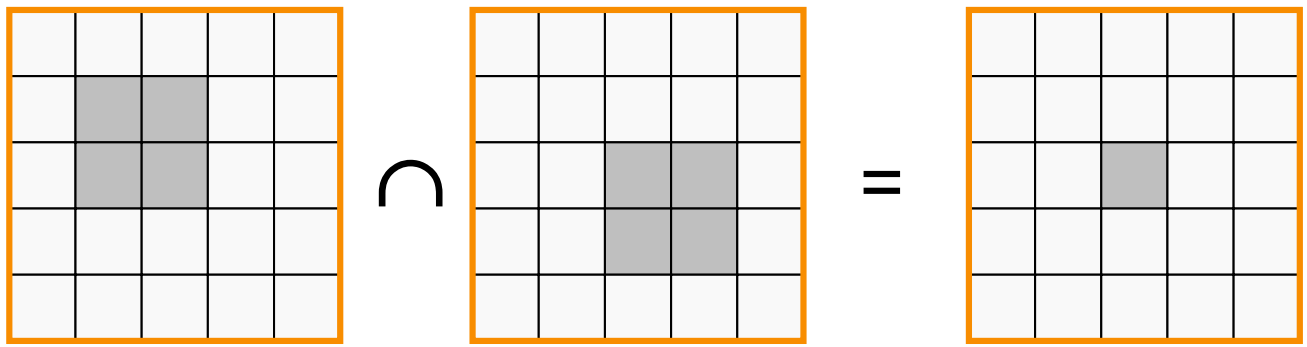
Voxels

- Partition space in uniform grid
- Each grid cell is called a voxel (like pixel)
- Each voxel has a value associated ot it:
  - Binary voxel: 0, if the voxel is outside the object; 1, if the voxel is inside the object
  - Continuous voxel: each voxel stores a continuous value (e.g. temperature, color, density, . . .)

Boolean operations

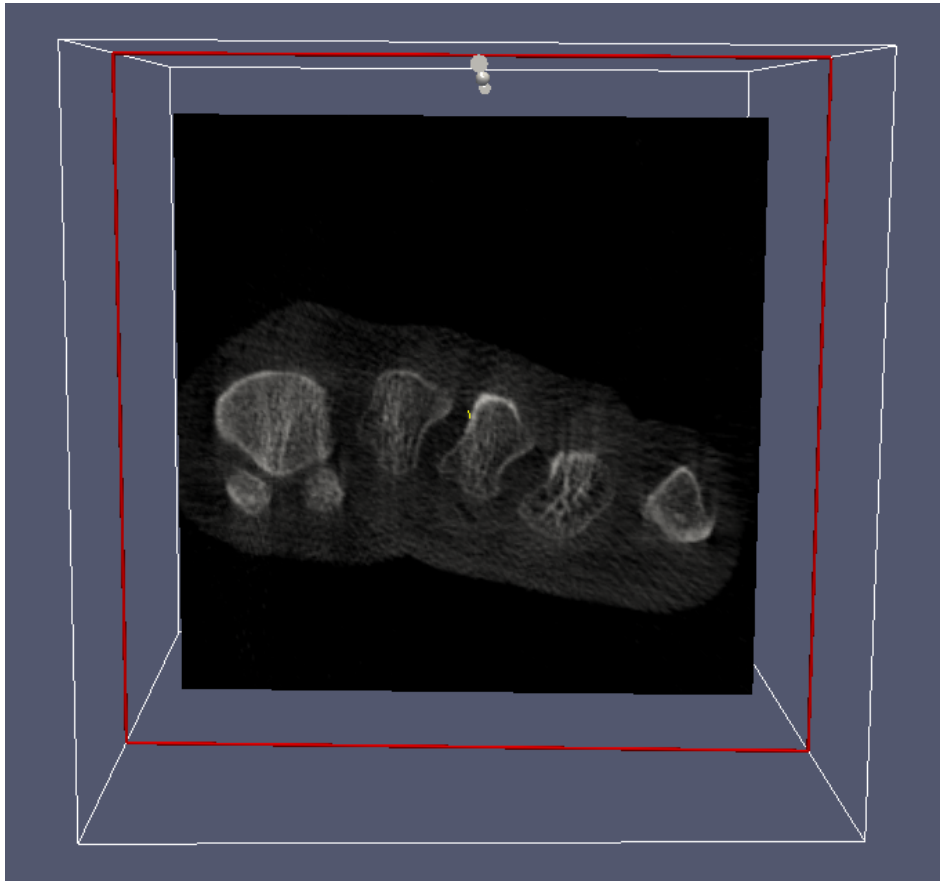
Easy: by comparing objects voxel by voxel.





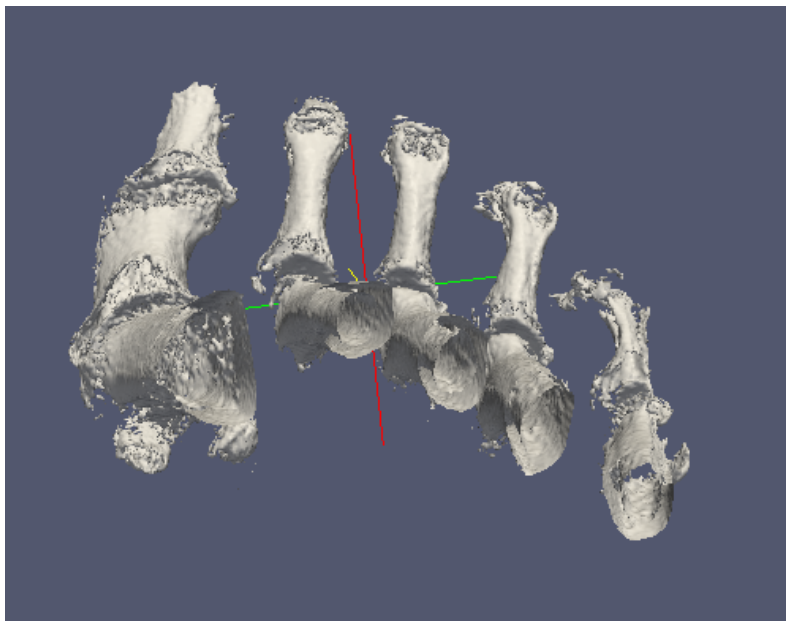
Visualization - slicing

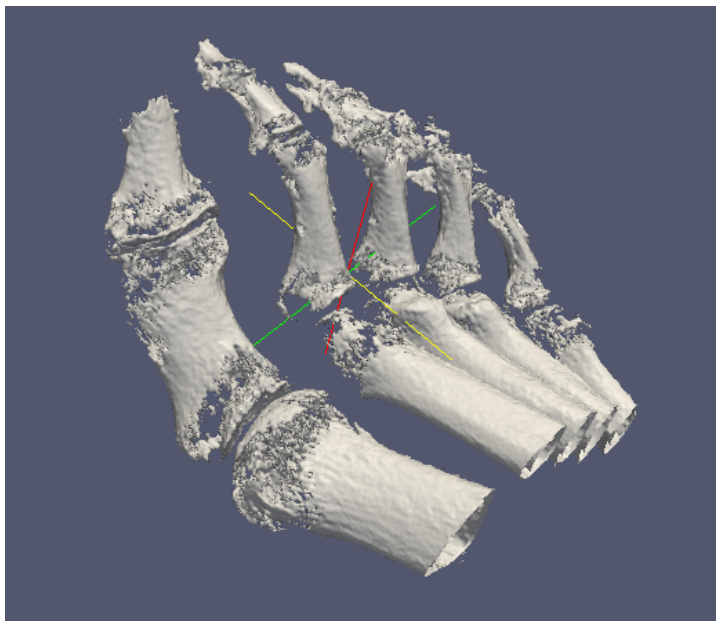
Slicing: draw a two-dimensional image resulting from the intersection of voxels with a user specified plane



Visualization - Isosurfaces extraction

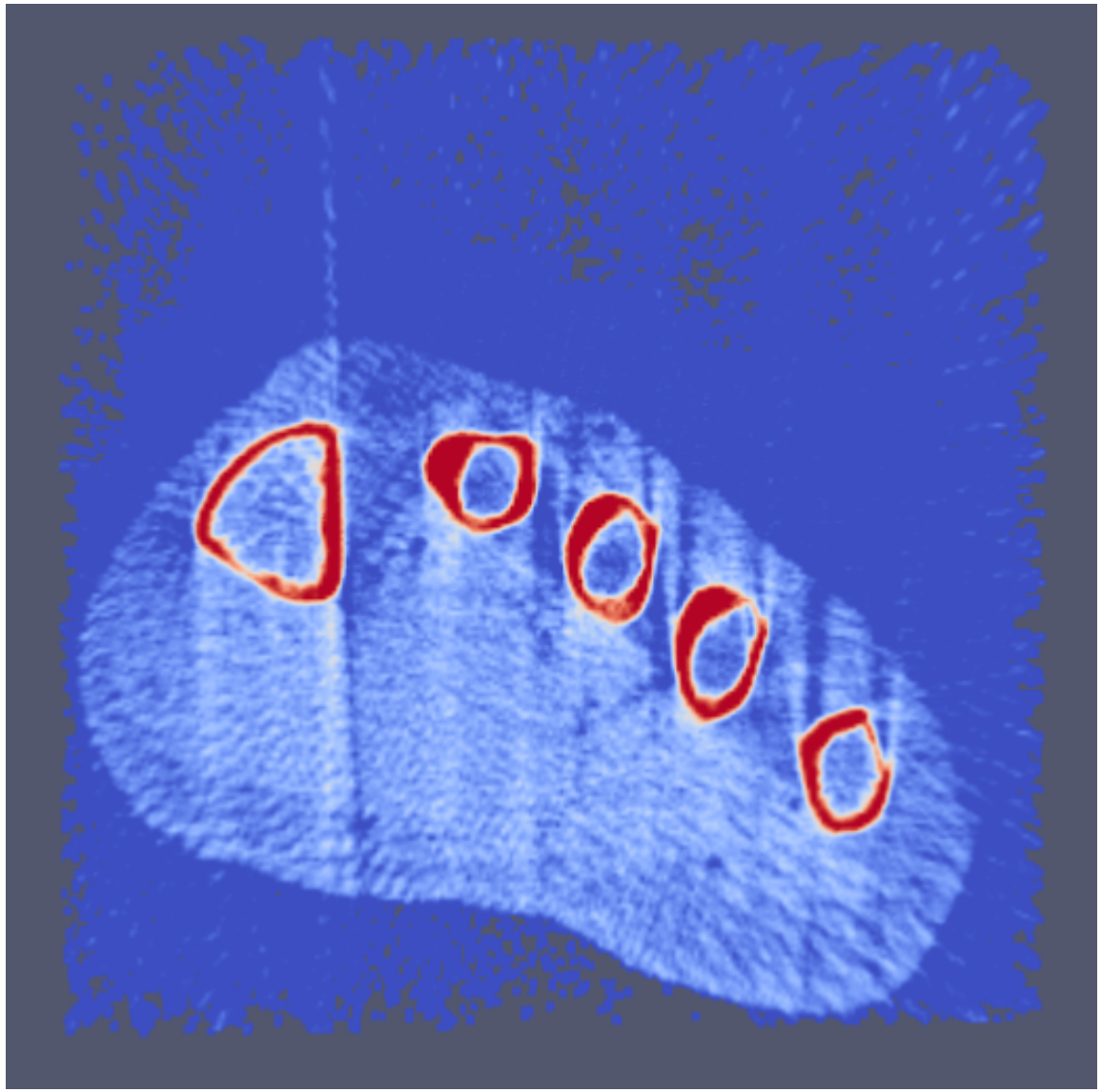
Treat the voxel grid as a sampling of some function $f(x, y, z)$ on a regular grid and extract isosurfaces satisfying $\{(x, y, z) \in R^3 : f(x, y, z) = \lambda\}$ for a user specified λ by algorithms like Marching Cubes and its variants





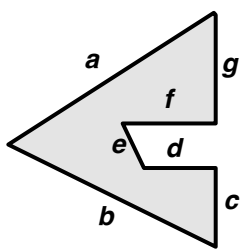
Visualization - volumetric ray-casting

Integrate density along rays through voxels

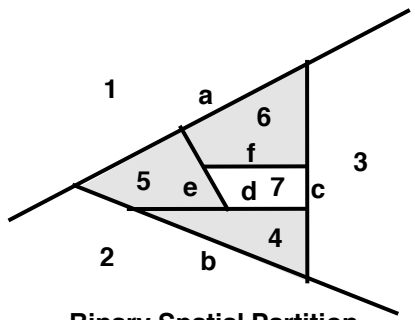


BSP trees

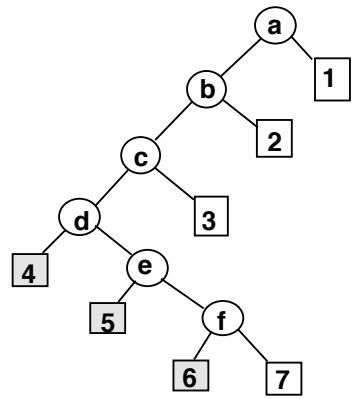
- BSP: Binary Space Partition
- Recursive partition of space by planes of arbitrary orientation
- Mark leaves of the corresponding binary tree as inside or outside of the object
- BSP trees are constructed from polygonal representations (polygon meshes)
- Point classification (i.e. determine if a point is inside or outside a solid) is done by traversing the tree and assigning to the point the status (inside or outside) of the leaf cell hit by the traversal



Object



Binary Spatial Partition



Binary Tree

Boundary representation

- Often abbreviated as B-rep or B-Rep or BREP
- Represents a solid as a collection of connected surface elements that define the boundary between the solid interior and its exterior
- B-rep models are composed of two parts: topology and geometry
  - topology: describes the different entities, e.g. vertices, (curved) edges, and (curved) faces and how these entities are connected together.
  - geometry: describes the vertices coordinates and the equations used in describing edges and faces
- Operations: set operations (union, intersection, complement), extrusion (or sweeping), chamfer, blending ...
- Point membership queries are done by computing the number of intersections between the surface and a ray starting from the point: if this number is odd, the point is inside, otherwise it is outside

Constructive Solid Geometry

- Often abbreviated as CSG (Constructive Solid Geometry)
- A technique for modeling complex objects by combining simple primitives with set operations
- Typical primitives: cubes, cylinders, spheres, pyramids
- Set operations used in CSG are regularized: i.e. the resulting solid is a regularized set
- Point membership queries are done by classifying a point against each primitive and then evaluating the resulting boolean expression

Constructive Solid Geometry

- A technique for modeling complex objects by combining simple primitives with set operations
- Typical primitives: cubes, cylinders, spheres, pyramids

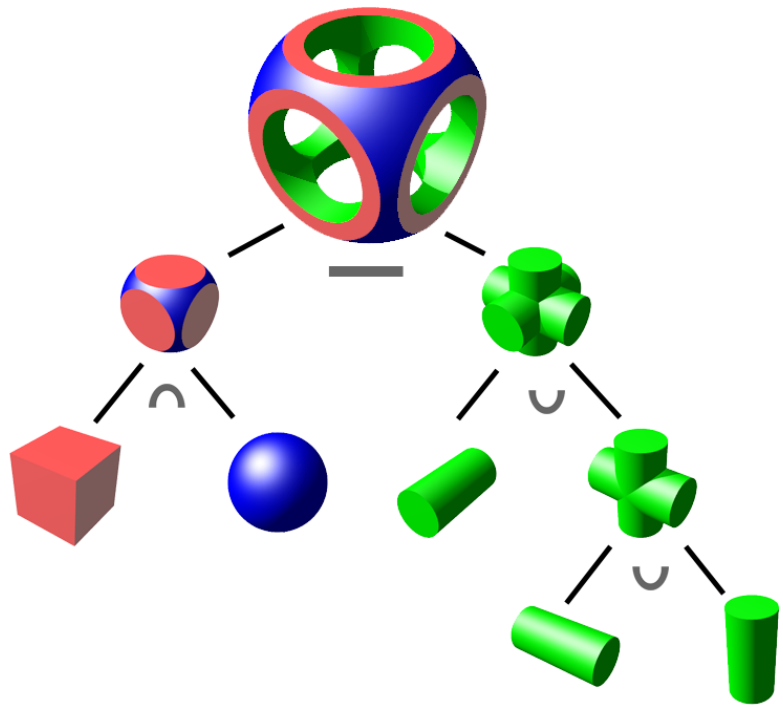


Image credit: Wikipedia