Machine Vision

Lecture 8. Part 2.

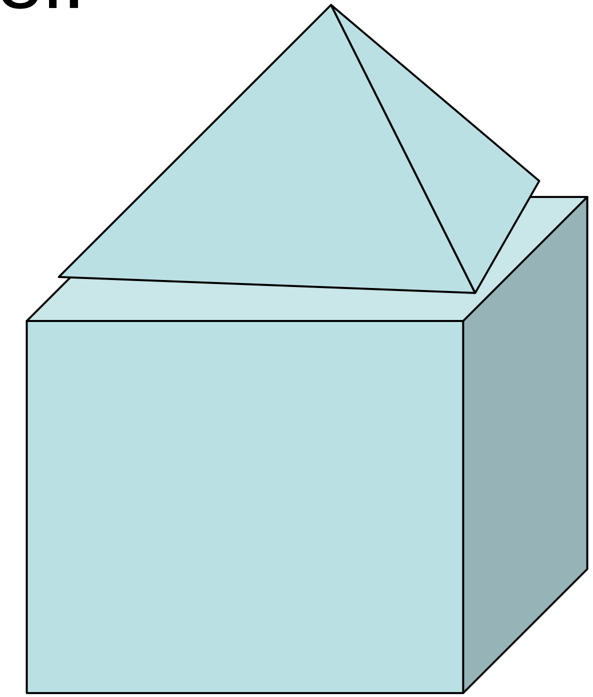
Representation of 3D images

Object Models

- We need to have a way of describing shapes of objects of interest (e.g., widgets), and also describing shapes in the scene.
- Need 3-d models, so we can recognize objects from different viewing angles.
- Base these on “volumetric primitives” (i.e. 3d shapes) (e.g., cube, cylinder).
- Now our first stage is to get from our surfaces, to possible shapes.
  - Reasonably easy, if surfaces are right and no obscuring objects.

Object Models

- Image of a house might end up with model:



Pyramid
+
Cube

3D models

- **3D model** is a mathematical representation of a three-dimensional object
- It can be displayed as a two-dimensional image through a process called *3D rendering* or used in a computer simulation of physical phenomena
- 3D models are most often created with special software applications called 3D modelers. Being a collection of data (points and other information)
- 3D models can be created by hand or algorithmically (procedural modeling) and usually are storing in separate files

3D models (cont.)

- A 3D model by itself is not visual. It can be rendered as a simple wireframe at varying levels of detail, or shaded in a variety of ways
- Many 3D models, however, are covered in a covering called a texture (the process of aligning the texture to coordinates on the 3D model is called texture mapping).
- A texture is nothing more than a graphic image, but gives the model more detail and makes it look more realistic. A 3D model of a person, for example, looks more realistic with a texture of skin and clothes, than a simple monochromatic model or wireframe of the same model

3D model (cont.)

- Other effects, beyond texturing, can be done to 3D models to add to their realism. For example, the surface normals can be tweaked to effect how they are lit, certain surfaces can have bump mapping applied and any other number of 3D rendering tricks can be applied.
- 3D models are often animated for some uses. They can be animated from within the 3D modeler that created them or externally. Often extra data is added to the model to make it easier to animate. For example, some 3D models of humans and animals have entire bone systems so they will look realistic when they move and can be manipulated via joints and bones

Primitives

- Model is based on primitives
- *Primitives* are basic shapes
- Most 3d packages have same primitives:
  - Polygon, Sphere, Cube, Cylinder, Plane
- Most used primitive is polygon

Problems with Polygons

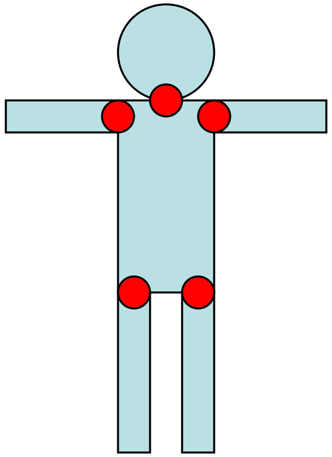
- Interaction is a problem
  - Dragging points around is time consuming
  - Maintaining things like smoothness is difficult
- They are inherently an approximation
  - Things like silhouettes can never be perfect without very large numbers of polygons, and corresponding expense
- Low level representation
  - Eg: Hard to increase, or decrease, the resolution
  - Hard to extract information like curvature

More Object Representations

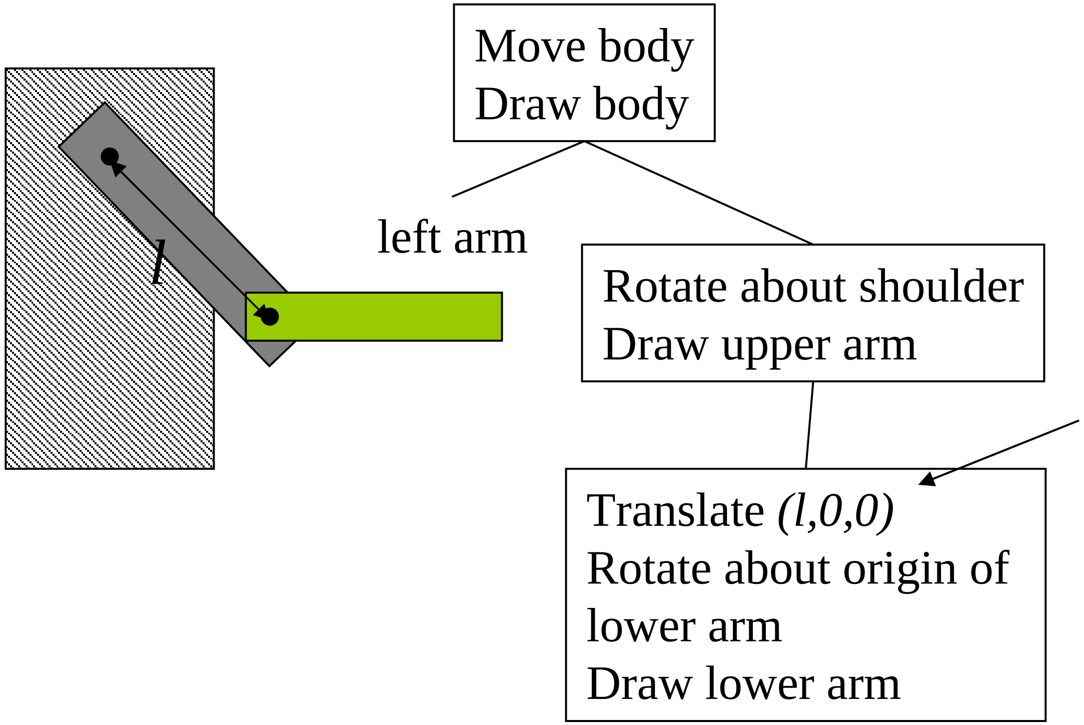
- Hierarchical modeling
- Instancing and Parametric Instancing
- Constructive Solid Geometry
- Sweep Objects
- Octrees
- Blobs and Metaballs and other such things

Hierarchical Modeling

- Hierarchical model: Group of meshes related by a tree (or graph) structure
  - Properties of children are derived from their parents
  - Most useful for animating polygonal meshes
- Consider a walking (humanoid, classic) robot:
  - How would you move the robot around?
  - Does the entire robot move in the same way?
  - Does the position of one part of the robot depend on other parts?



Hierarchical Model Example



Move body
Draw body

Rotate about shoulder
Draw upper arm

Translate *(l,0,0)*
Rotate about origin of lower arm
Draw lower arm

Important Point:

- Every node has its own local coordinate system.
- This makes specifying transformations much easier.
- What are we assuming about the "upper arm" coordinate system?

Hierarchical Details

- Generally represented as a tree, with transformations and instances at any node
  - Can use a general graph, but resolving inheritance conflicts is a problem
- Rendered by traversing the tree, applying the transformations, and rendering the instances
- Particularly useful for animation
  - Human is a hierarchy of body, head, upper arm, lower arm, etc…
  - Animate by changing the transformations at the nodes
- Other things can be inherited (colors, surface properties)

Instancing

- Sometimes you need many copies of the “same” object
  - Like chairs in a room
- Define one chair, the base or the prototype
- Create many *instances* (copies) of it, and apply a different transformation to each

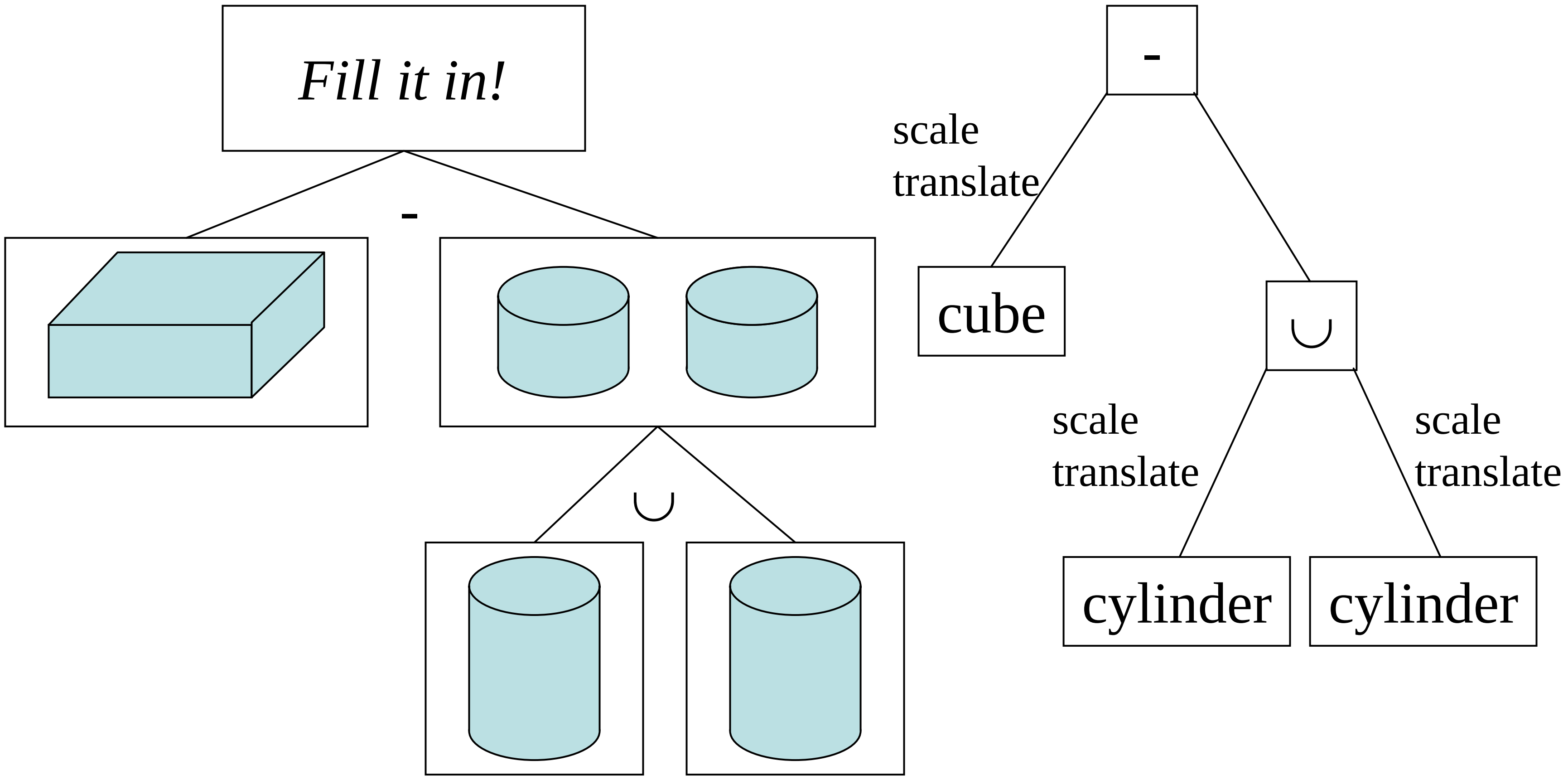
Parametric Instancing

- Many things, called primitives, are conveniently described by a label and a few parameters
  - Cylinder: Radius, length, does it have end-caps, …
  - Bolts: length, diameter, thread pitch, …
- This is a modeling format:
  - Provide software that knows how to draw the object given the parameters, or knows how to produce a polygonal mesh
  - How you manage the model depends on the rendering style
  - Can be an exact representation

Constructive Solid Geometry (CSG)

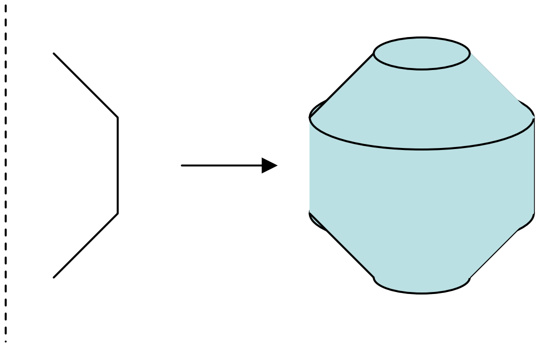
- Based on a tree structure, like hierarchical modeling, but now:
  - The internal nodes are set operations: union, intersection or difference (sometimes complement)
  - The edges of the tree have transformations associated with them
  - The leaves contain only geometry
- Allows complex shapes with only a few primitives
  - Common primitives are cylinders, cubes, etc, or quadric surfaces
- Motivated by computer aided design and manufacture
  - *Difference* is like drilling or milling
  - A common format in CAD products

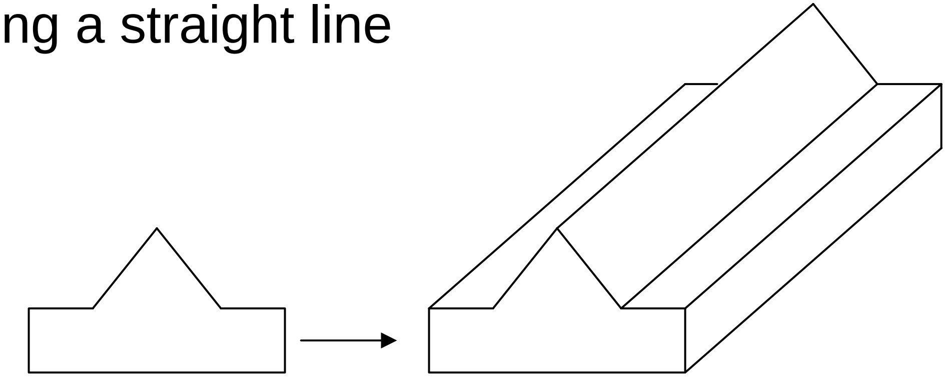
CSG Example


Fill it in!
-
∪
-
scale
translate
cube
∪
scale
translate
scale
translate
cylinder
cylinder

Sweep Objects

- Define a polygon by its edges
- Sweep it along a path
- The path taken by the edges form a surface - the sweep surface
- Special cases
  - Surface of revolution: Rotate edges about an axis
  - Extrusion: Sweep along a straight line





Rendering Sweeps

- Convert to polygons
  - Break path into short segments
  - Create a copy of the sweep polygon at each segment
  - Join the corresponding vertices between the polygons
  - May need things like end-caps on surfaces of revolution and extrusions
- Normals come from sweep polygon and path orientation
- Sweep polygon defines one texture parameter, sweep path defines the other

General Sweeps

- The path maybe any curve
- The polygon that is swept may be transformed as it is moved along the path
  - Scale, rotate with respect to path orientation, …
- One common way to specify is:
  - Give a poly-line (sequence of line segments) as the path
  - Give a poly-line as the shape to sweep
  - Give a transformation to apply at the vertex of each path segment
- Difficult to avoid self-intersection

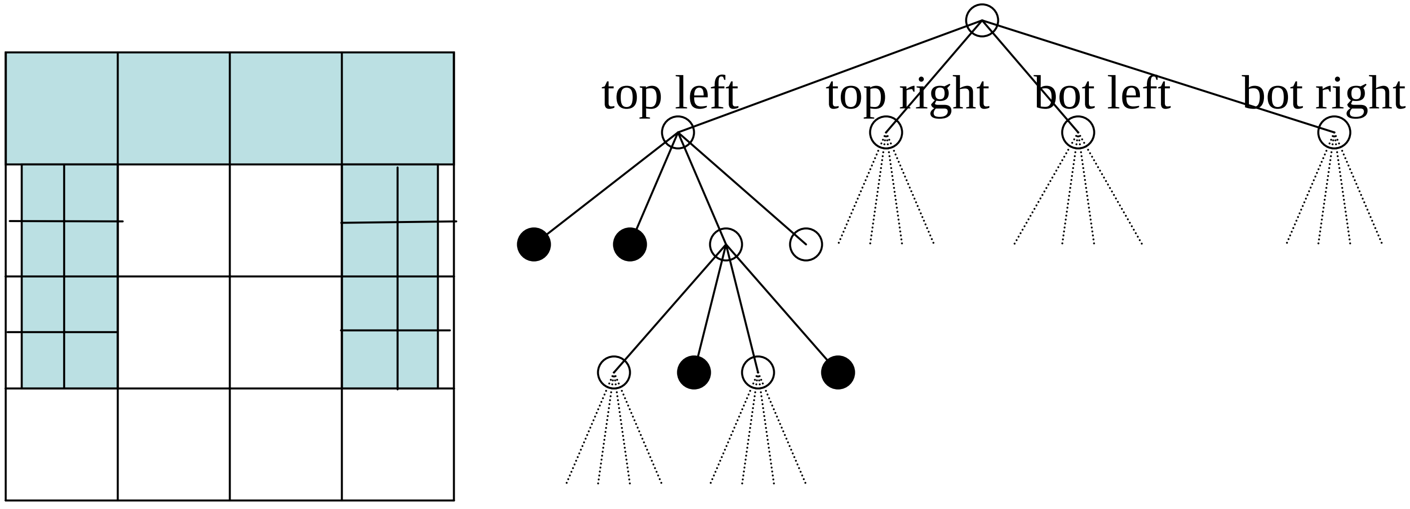
Spatial Enumeration

- Basic idea: Describe something by the space it occupies
  - For example, break the volume of interest into lots of tiny cubes, and say which cubes are inside the object
  - Works well for things like medical data
    - The process itself, like MRI or CAT scans, enumerates the volume
    - Data is associated with each voxel (volume element)
- Problem to overcome:
  - For anything other than small volumes or low resolutions, the number of voxels explodes
  - Note that the number of voxels grows with the *cube* of linear dimension

Octrees (and Quadtrees)

- Build a tree where successive levels represent better resolution (smaller voxels)
- Large uniform spaces result in shallow trees
- Quadtree is for 2D (four children for each node)
- Octree is for 3D (eight children for each node)

Quadtree Example



Octree principle is the same, but there are 8 children

Rendering Octrees

- Volume rendering renders octrees and associated data directly
  - A special area of graphics, visualization, not covered in this class
- Can convert to polygons by a few methods:
  - Just take faces of voxels that are on the boundary
  - Find iso-surfaces within the volume and render those
  - Typically do some interpolation (smoothing) to get rid of the artifacts from the voxelization
- Typically render with colors that indicate something about the data, but other methods exist

Spatial Data Structures

- Octrees are an example of a spatial data structure
  - A data structure specifically designed for storing information of a spatial nature
    - E.g. Storing the location of fire hydrants in a city
- In graphics, octrees are frequently used to store information about where polygons, or other primitives, are located in a scene
  - Speeds up many computations by making it fast to determine when something is relevant or not
  - Just like BSP trees speed up visibility
- Other spatial data structures include BSP trees, KD-Trees, Interval trees, …

Implicit Functions

- Some surfaces can be represented as the vanishing points of functions (defined over 3D space)
  - Places where a function *f(x,y,z)=0*
- Some objects are easy represent this way
  - Spheres, ellipses, and similar
  - More generally, quadratic surfaces:

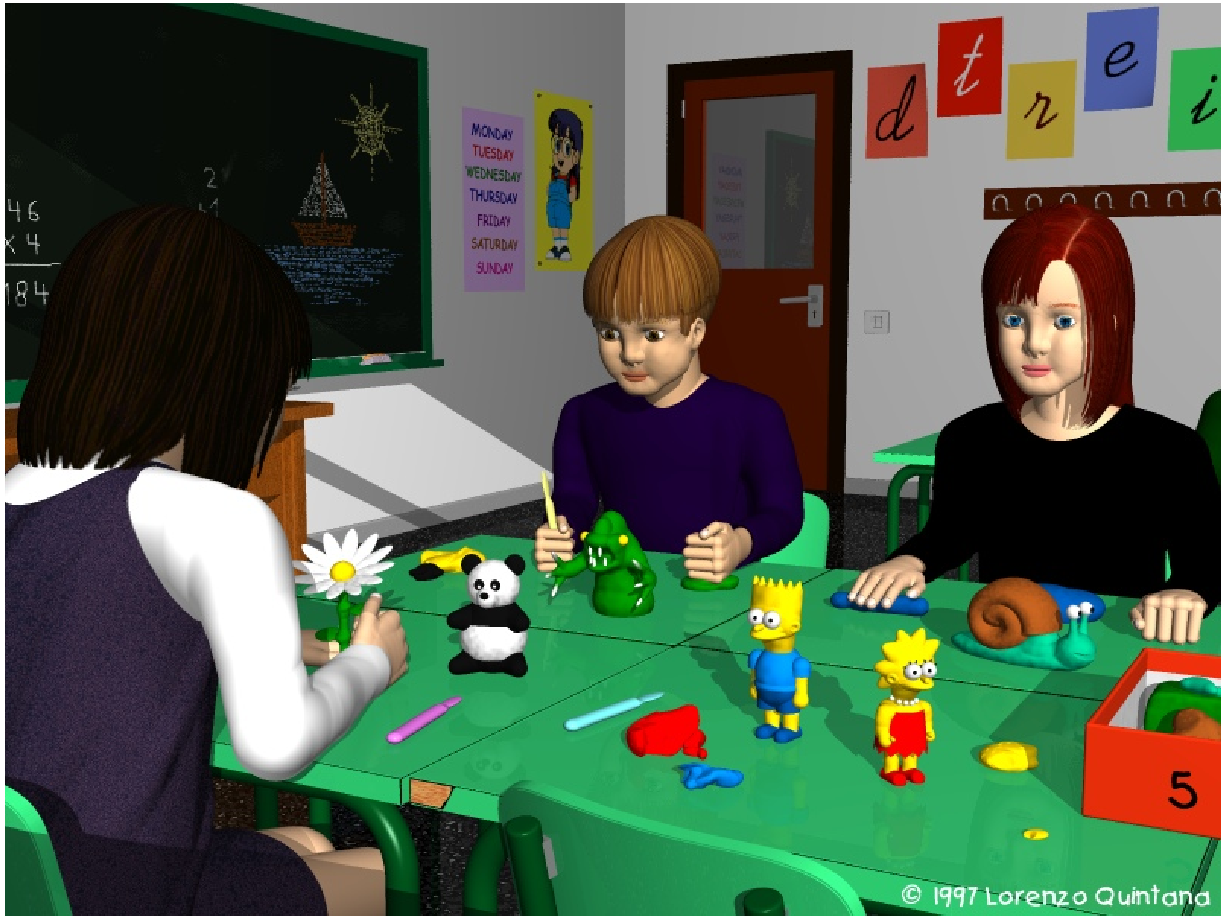
$$ax^2 + bx + cy^2 + dy + ez^2 + fz + g = 0$$

  - Shapes depends on all the parameters *a,b,c,d,e,f,g*

Blobs and Metaballs

- Define the location of some points
- For each point, define a function on the distance to a given point, *(x,y,z)*
- Sum these functions up, and use them as an implicit function
- Question: If I have two special points, in 2D, and my function is just the distance, what shape results?
- More generally, use Gaussian functions of distance, or other forms
  - Various results are called blobs or metaballs

Example with Blobs



© 1997 Lorenzo Quintana

Rendered with POVray. Not everything is a blob, but the characters are.

Blob Math

- Implicit equation: $$f(x,y,z)=\sum_{i=1}^{n_{blobs}} w_i g_i(x,y,z)=0$$
- The *wi* are weights – just numbers
- The *gi* are functions, one common choice is:

$$g_i(\mathbf{x})=e^{\frac{-(\mathbf{x}-c_i)^2}{\sigma_i}}$$

  - *ci* and σ*i* are parameters

Rendering Implicit Surfaces

- Some methods can render then directly
  - Raytracing - find intersections with Newton's method
- For polygonal renderer, must convert to polygons
- Advantages:
  - Good for organic looking shapes e.g. human body
  - Reasonable interfaces for design
- Disadvantages:
  - Difficult to render and control when animating
  - Being replaced with subdivision surfaces, it appears