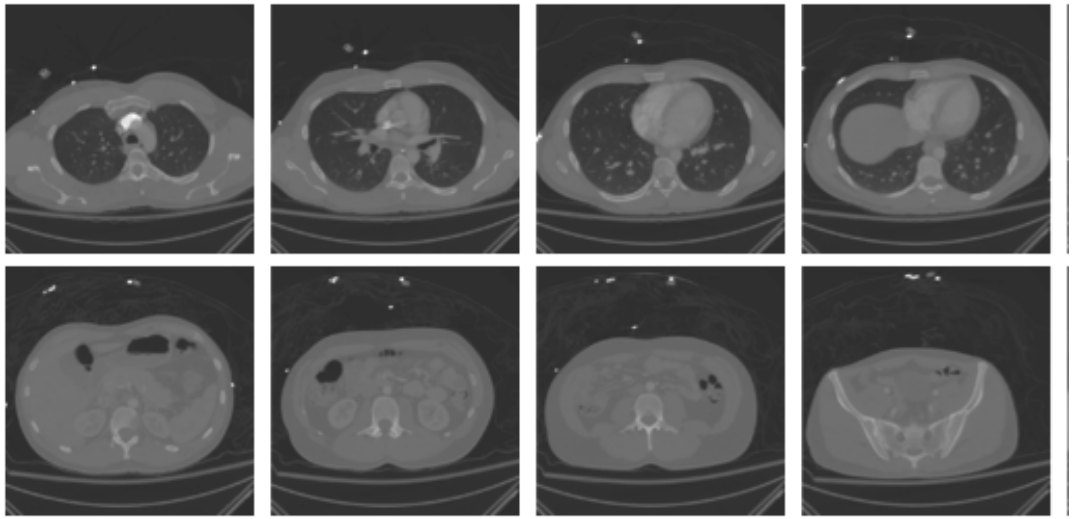
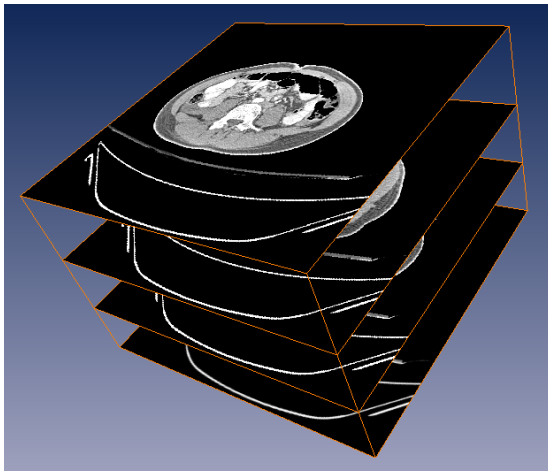
Module #13a:
Isosurface Visualization

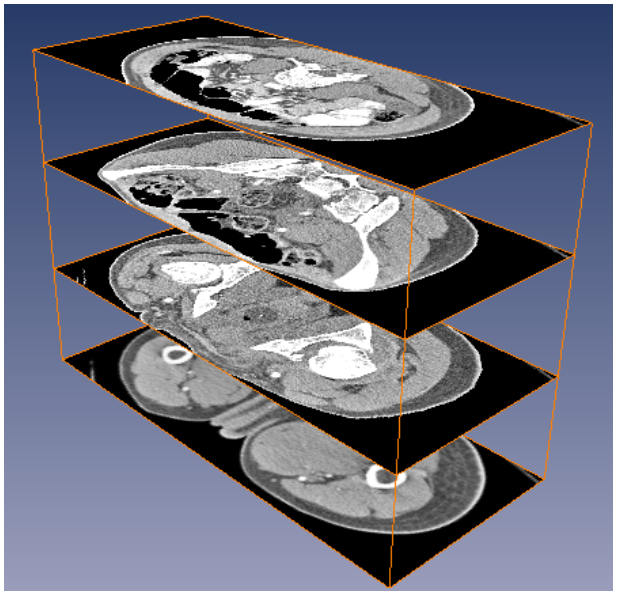
What's a volume?

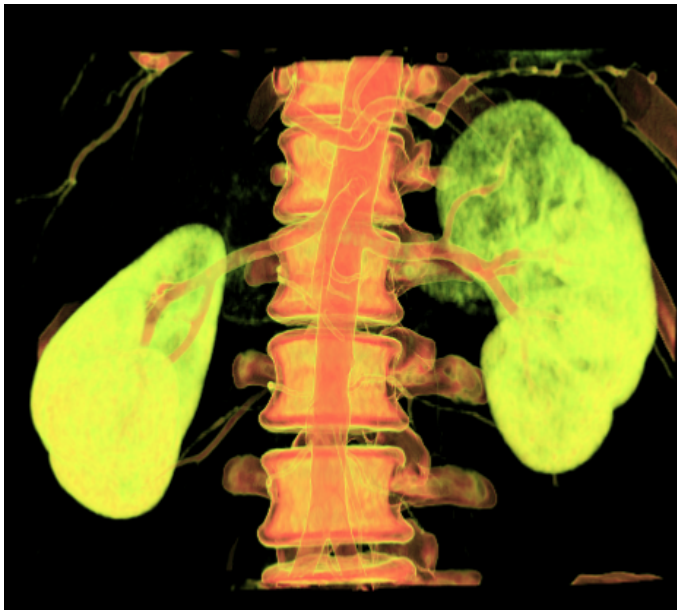




Direct Volume Rendering (DVR)

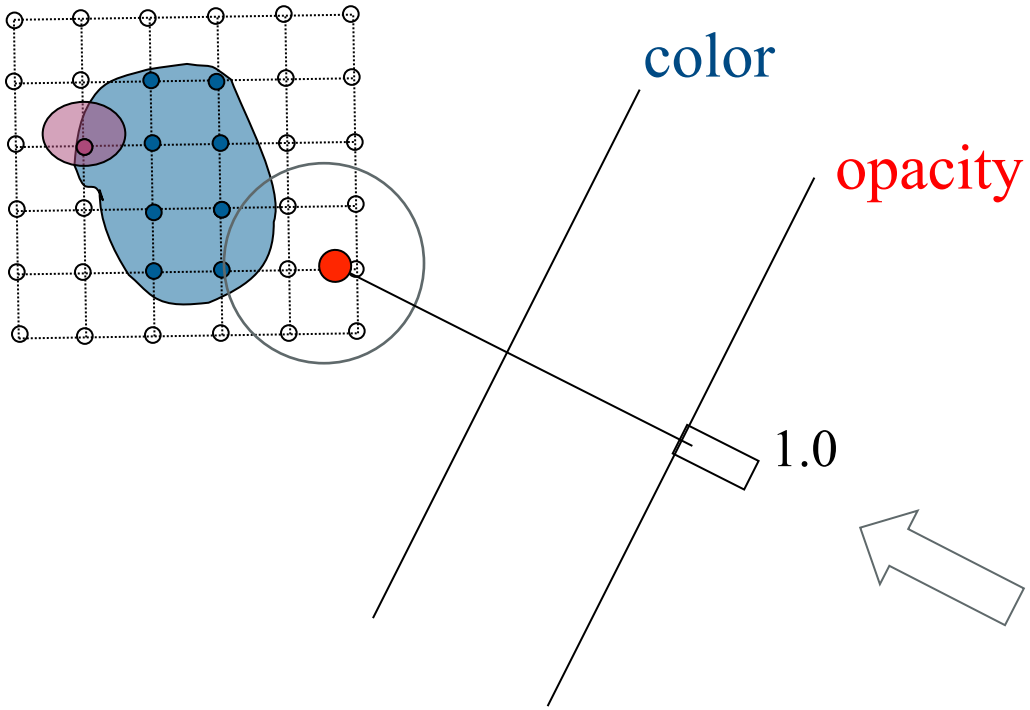
- Direct Volume Rendering
  - Technique used to display a 2D projection/image of a 3D image
  - Effective method to render different materials





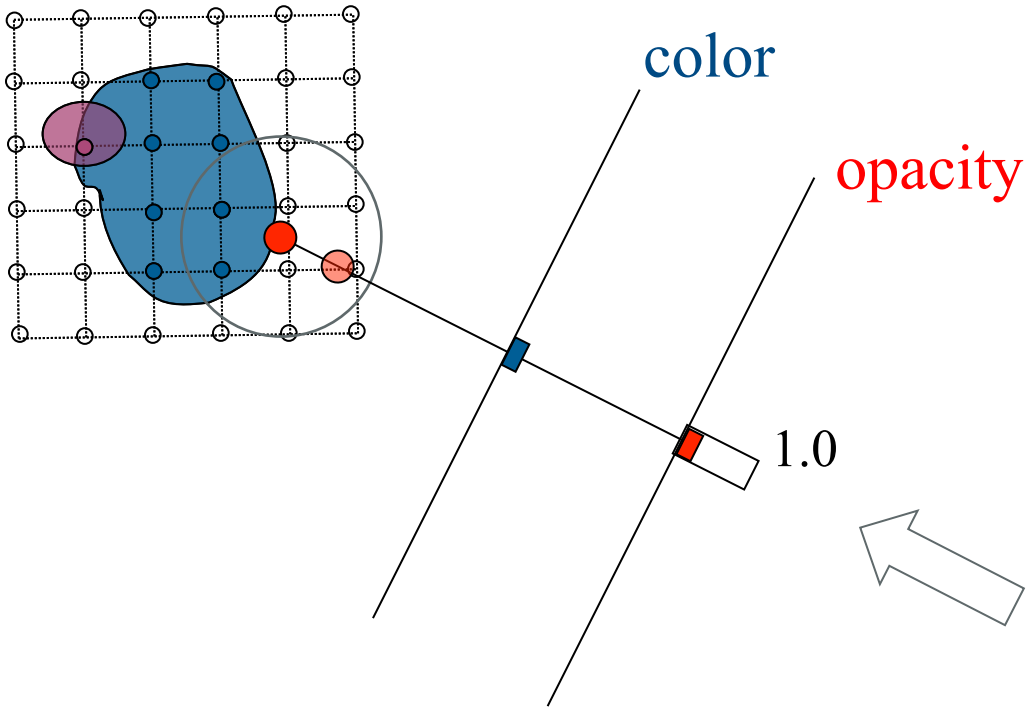
Raycasting

volumetric compositing



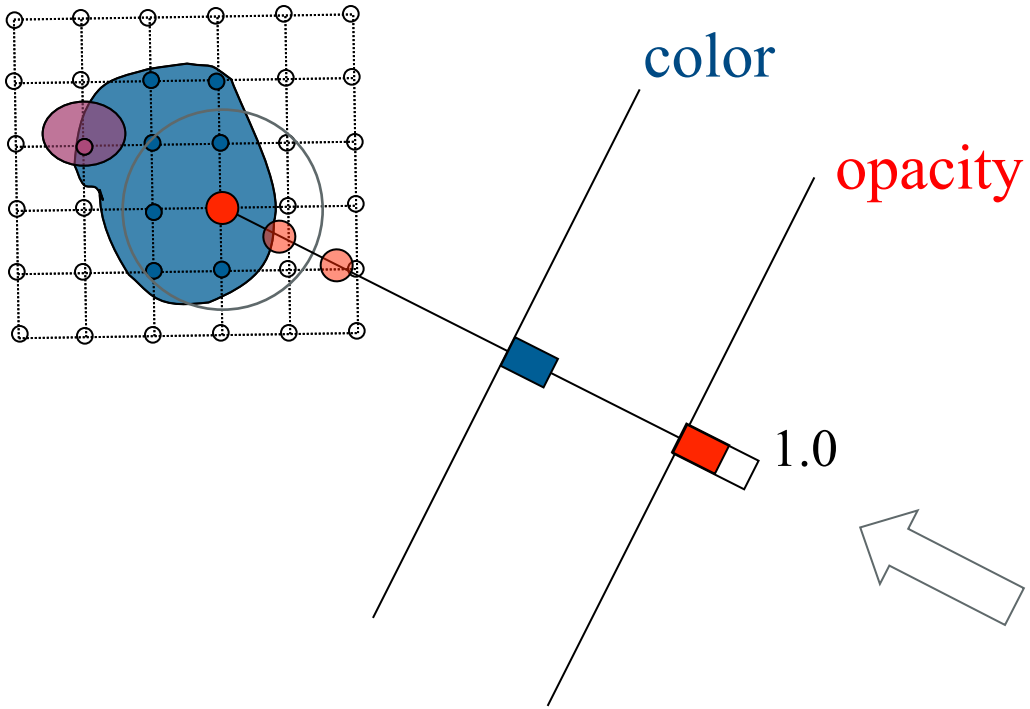
Raycasting

volumetric compositing



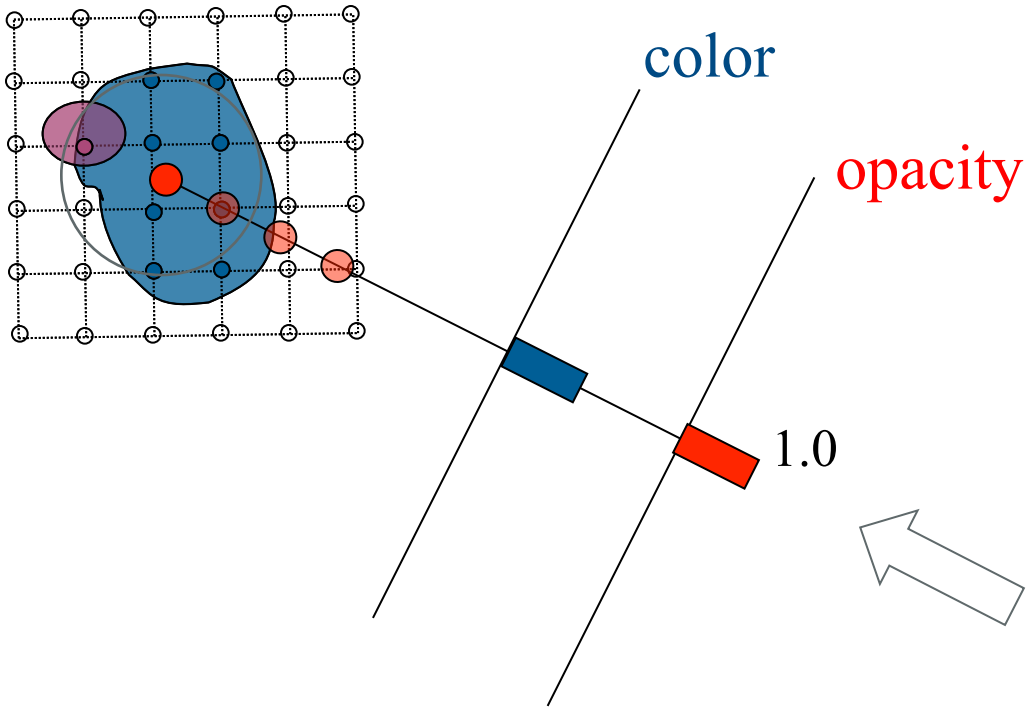
Raycasting

volumetric compositing



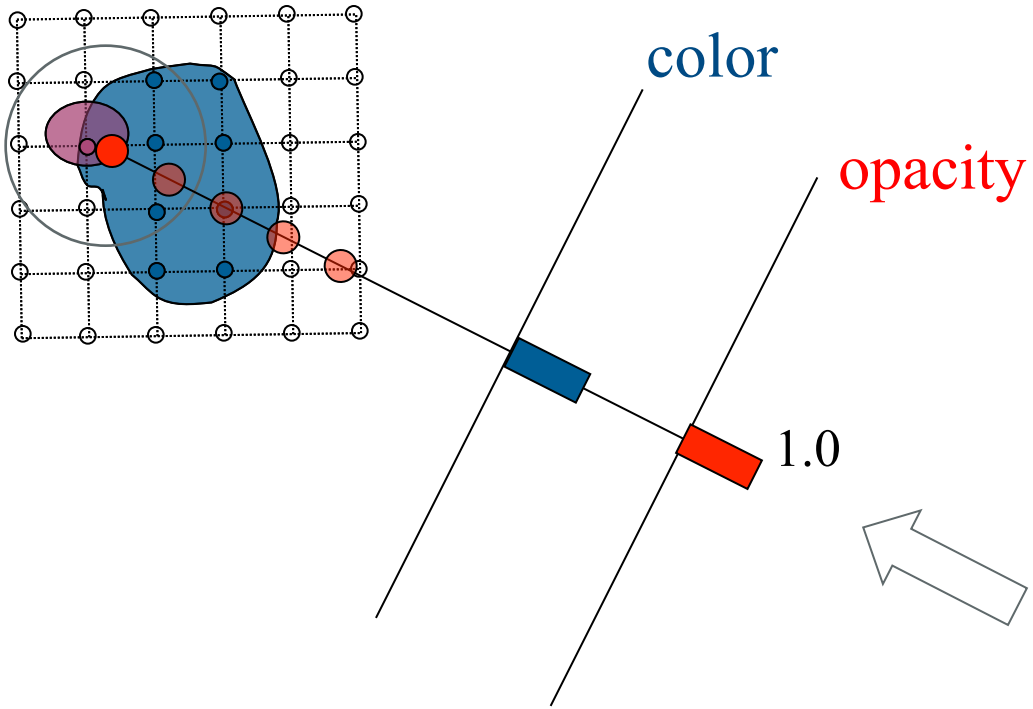
Raycasting

volumetric compositing



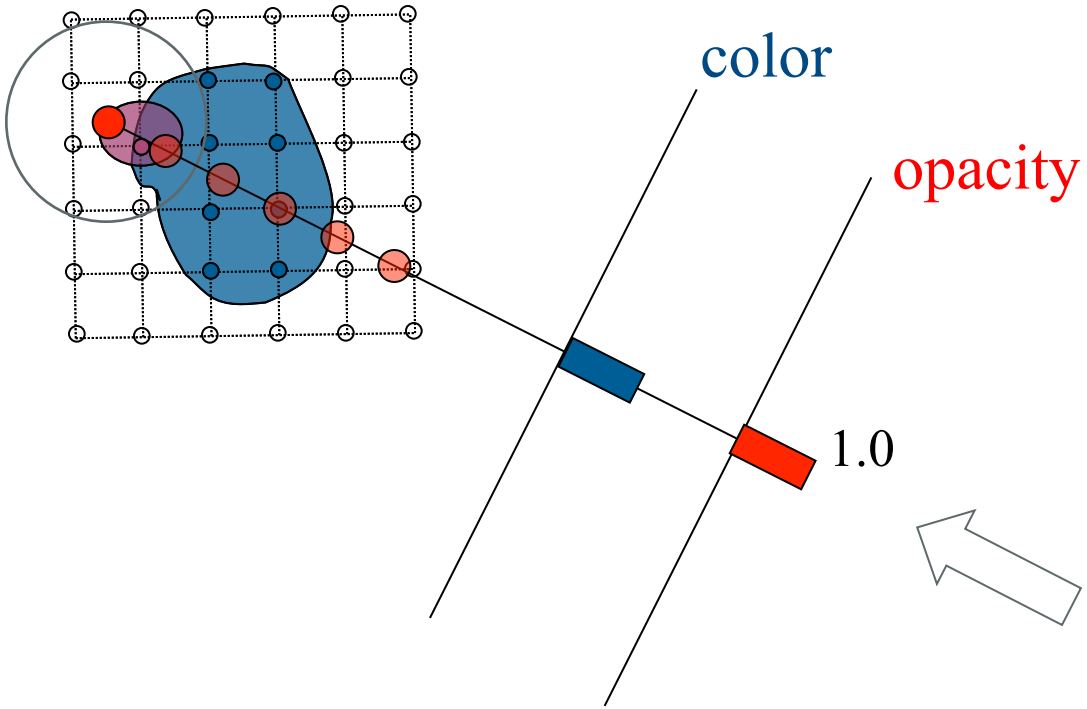
Raycasting

volumetric compositing



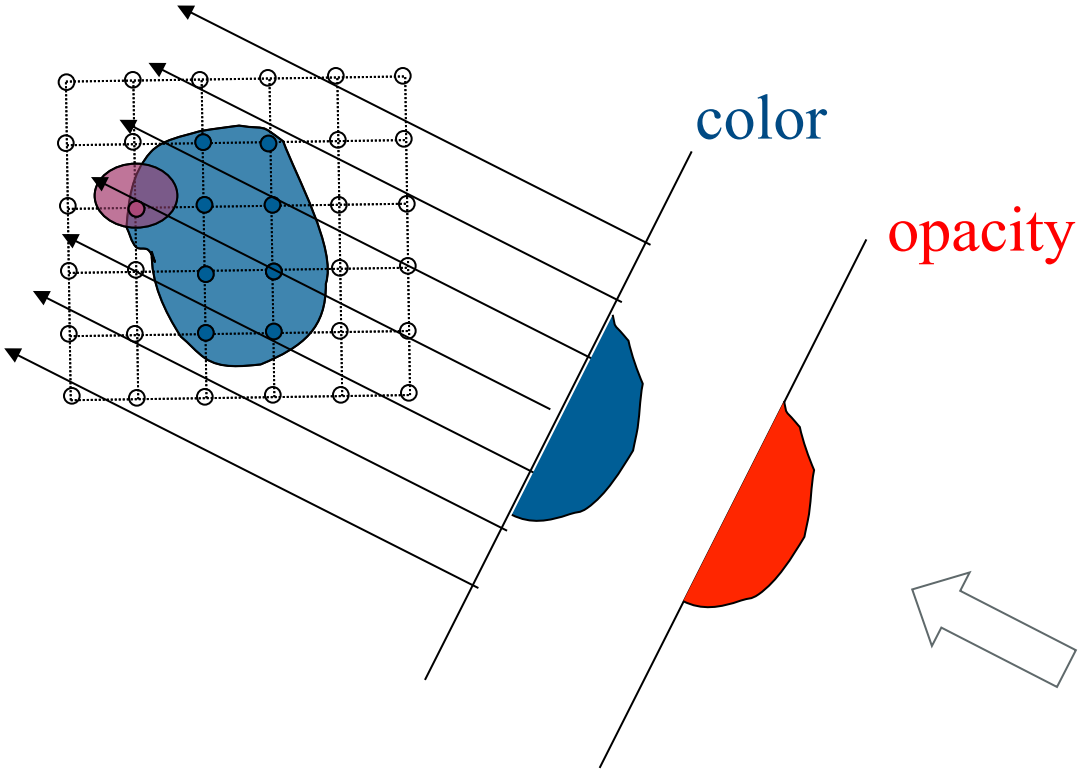
Raycasting

volumetric compositing

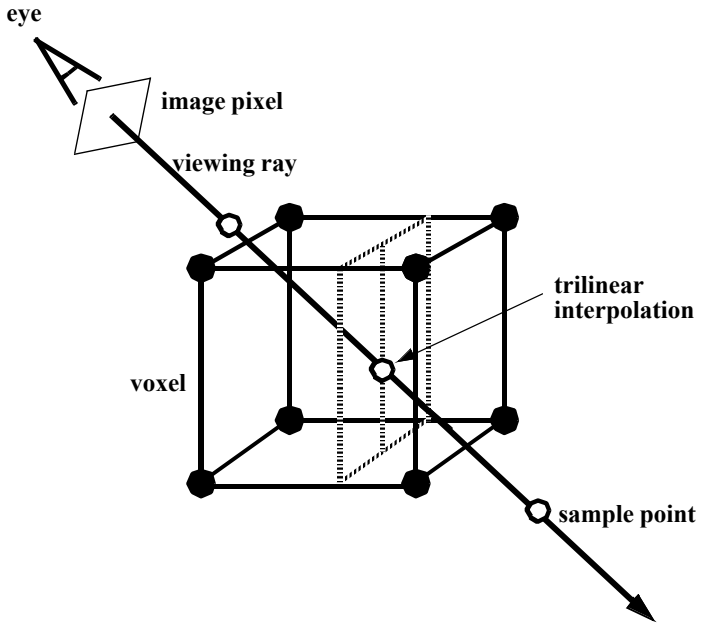


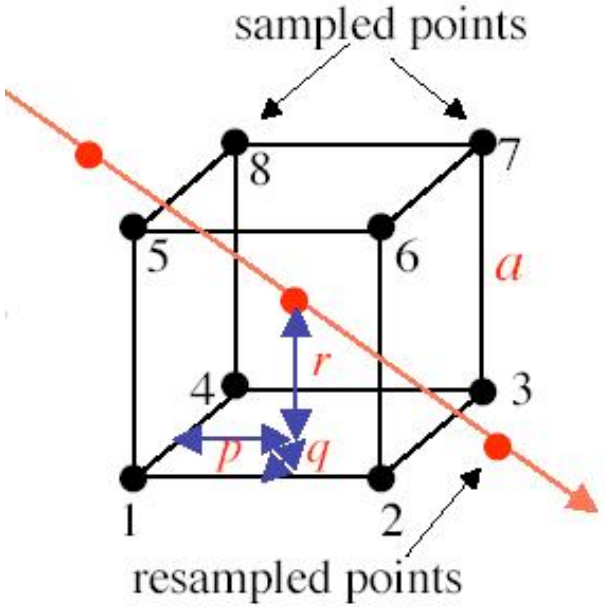
Raycasting

volumetric compositing



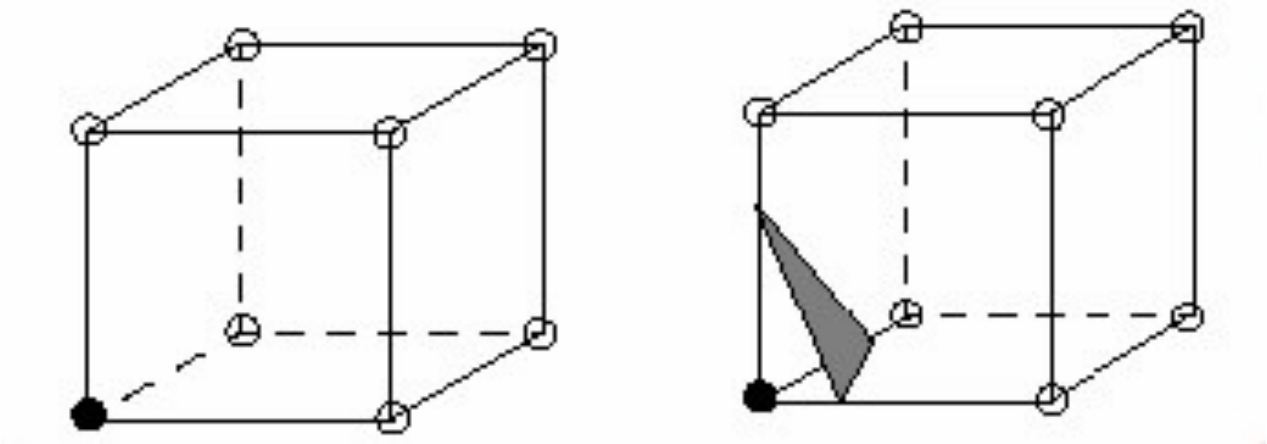
Trilinear - Interpolation





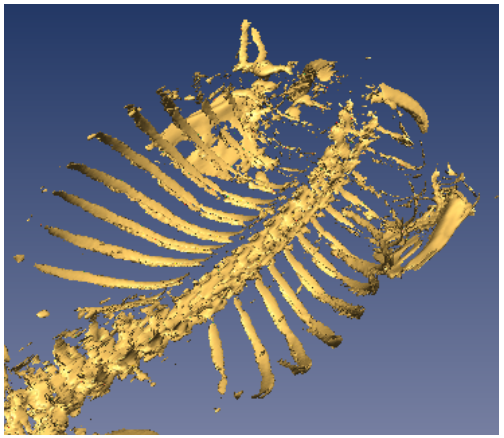
$$f_v = f_1(1-p/a)(1-q/a)(1-r/a) + f_2(p/a)(1-q/a)(1-r/a)$$
$$+ f_3(p/a)(q/a)(1-r/a) + f_4(1-p/a)(q/a)(1-r/a)$$
$$+ f_5(1-p/a)(1-q/a)(r/a) + f_6(p/a)(1-q/a)(r/a)$$
$$+ f_7(p/a)(q/a)(r/a) + f_8(1-p/a)(q/a)(r/a)$$

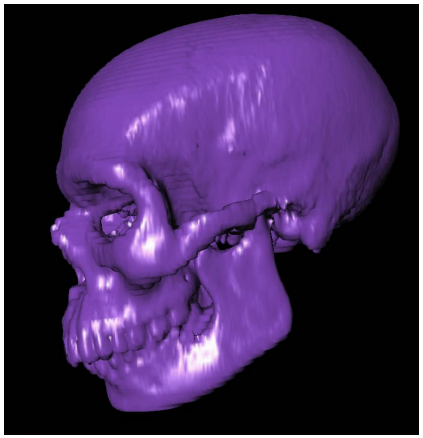
What about if we render triangles?



Iso-surfaces

- Approximate the data to polygonal primitives
- Use contours/boundaries of the volume to generate polygonal structures/“surfaces”
- Goal: create a constant density surface from a 3D array of data





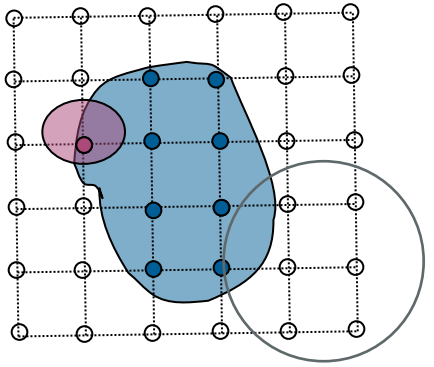
Introduction – Marching Cubes

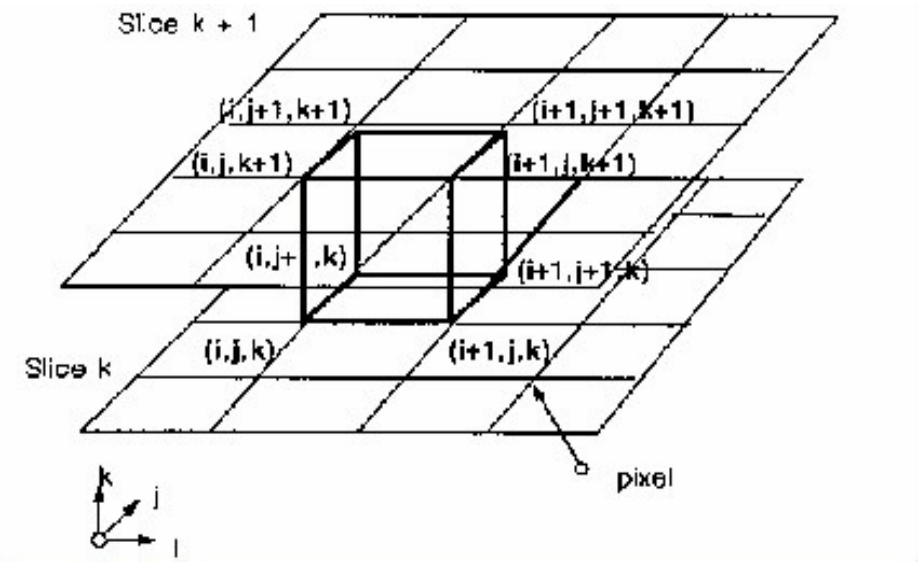
- Idea
  - create a triangular mesh that will approximate the iso-surface
  - calculate the normals to the surface at each vertex of the triangle

- Algorithm:
  1. locate the surface in a cube of eight pixels
  2. calculate normals
  3. march to the next cube

Surface intersection in a cube

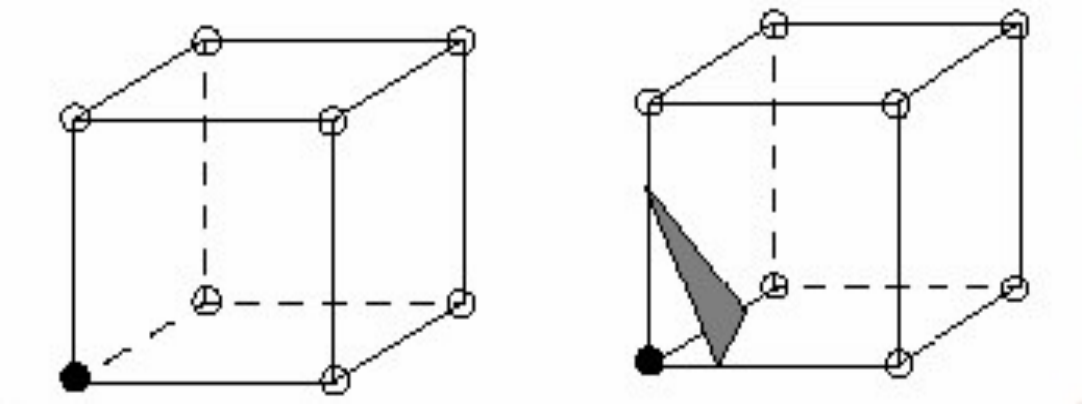
- Assign ZERO to vertex outside the surface
- Assign ONE to vertex inside the surface
- Surface intersects those cube edges where one vertex is outside and the other inside the surface





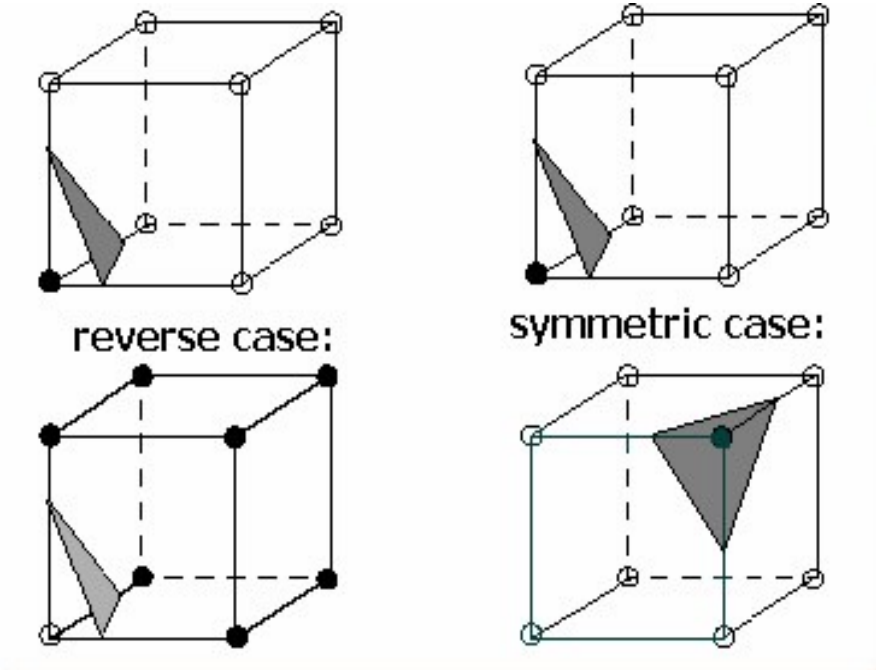
Surface intersection in a cube

- There are 28=256 ways the surface may intersect the cube
- Triangulate each case

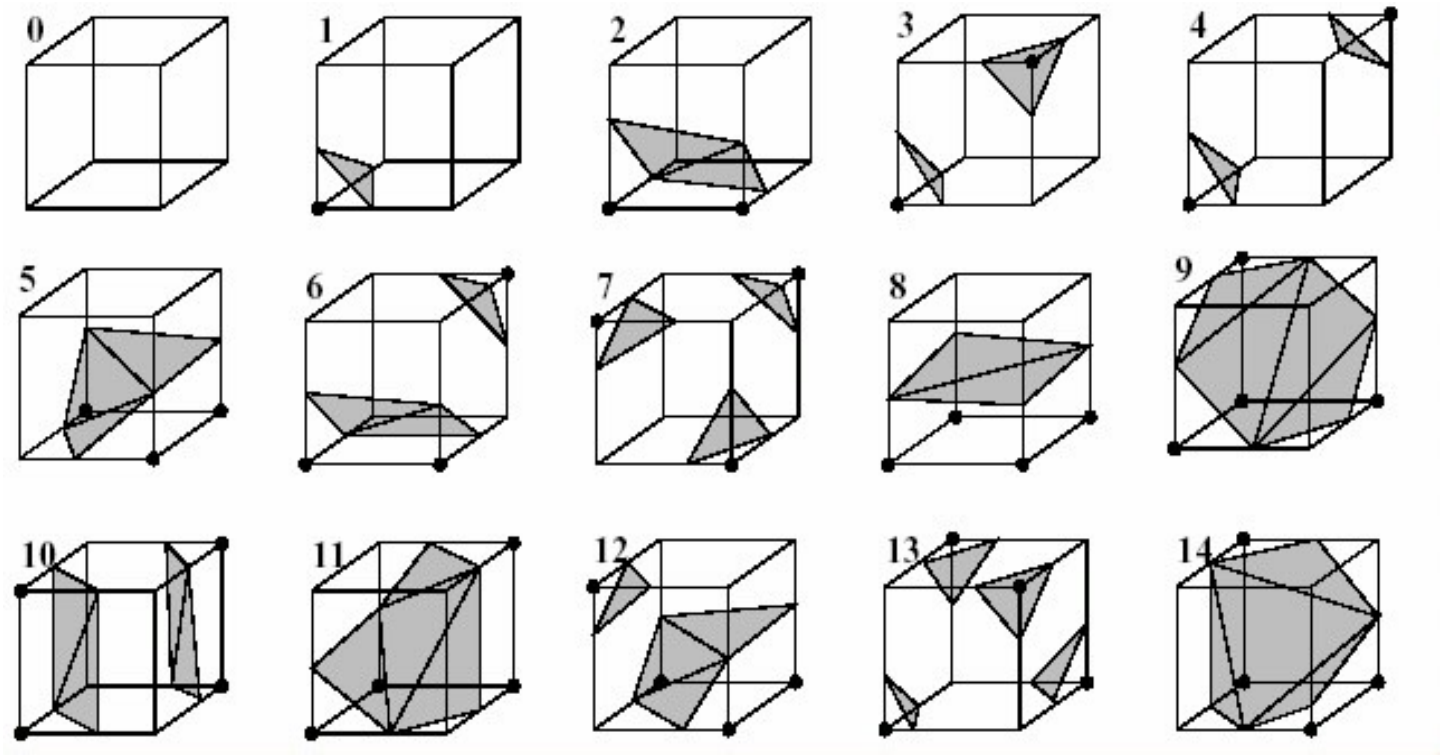


Patterns

Using the symmetries reduces those 256 cases to 15 patterns

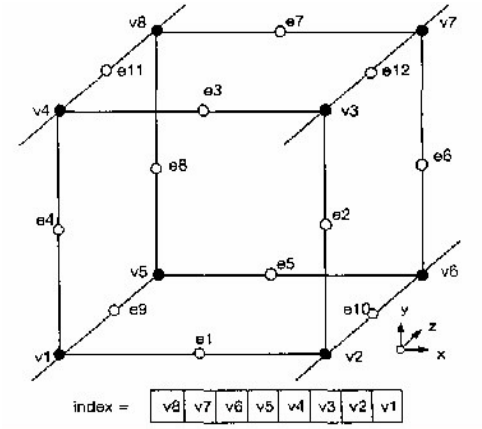


Patterns



Surface intersection in a cube

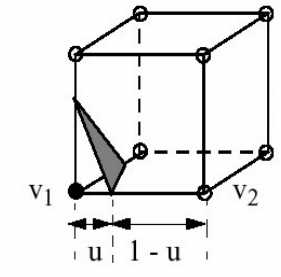
- Create an index for each case:



- Interpolate surface intersection along each edge

$$v_i = v_1 * (1 - u) + v_2 * u$$

$$u = \frac{v_1 - v_i}{v_1 - v_2}$$



Calculating normals

- Calculate normal for each cube vertex:

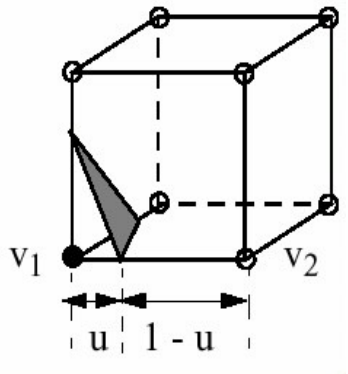
$$G_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x}$$

$$G_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y}$$

$$G_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z}$$

- Interpolate the normals at the vertices of the triangles:

$$\overrightarrow{n_1} = u\overrightarrow{g_2} + (1-u)\overrightarrow{g_1}$$

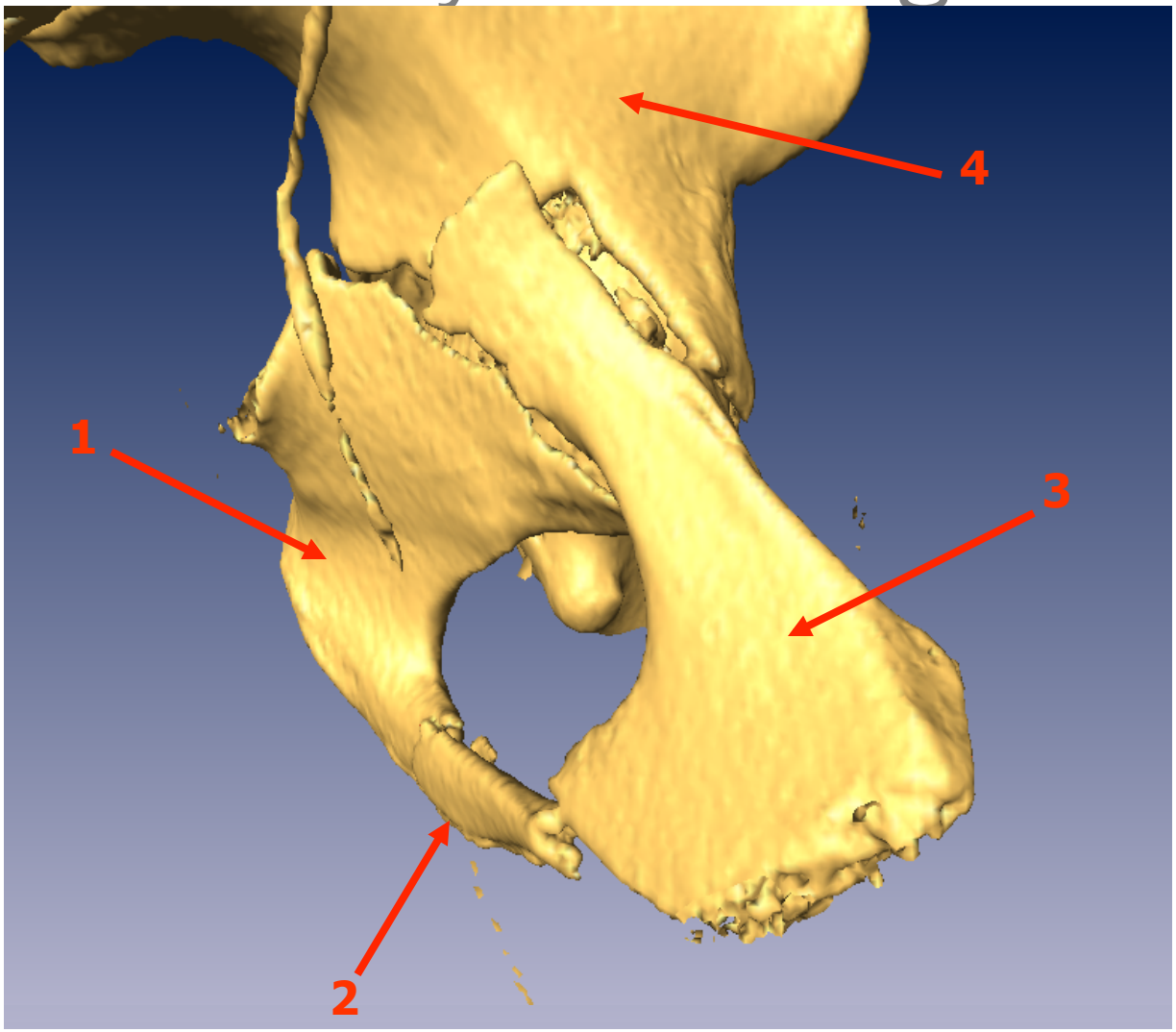


Pros and Cons

- Pros:
  - Simple rendering and manipulation
  - High resolution
- Cons:
  - Possible holes in the model
  - Model complexity

Iso-surfaces - Example

X-Ray Rendering



Conclusion