Solid Modeling

Michael Kazhdan

(601.457/657)

A Generalization of Algebraic Surface Drawing, Blinn 1982

Marching Cubes, Lorensen and Cline 1987

Solid Modeling

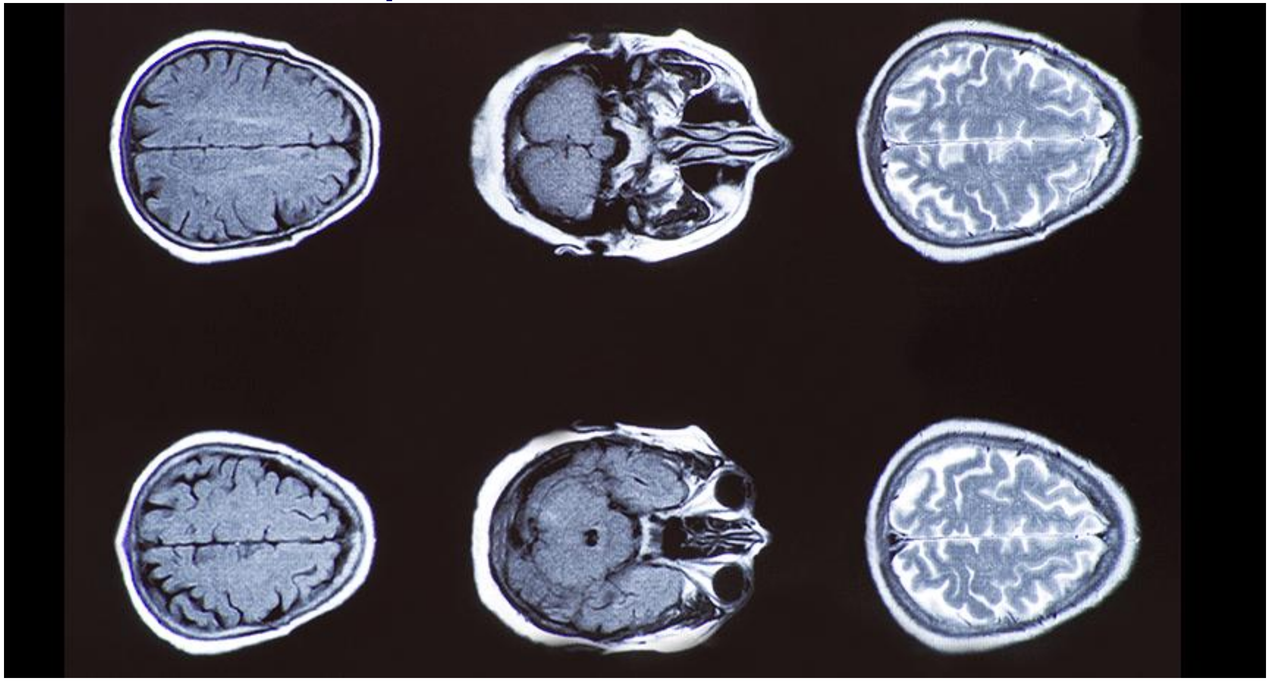
So far, we have focused on representing models with (triangular) meshes that approximate the surface/boundary of the model.

Advantages:

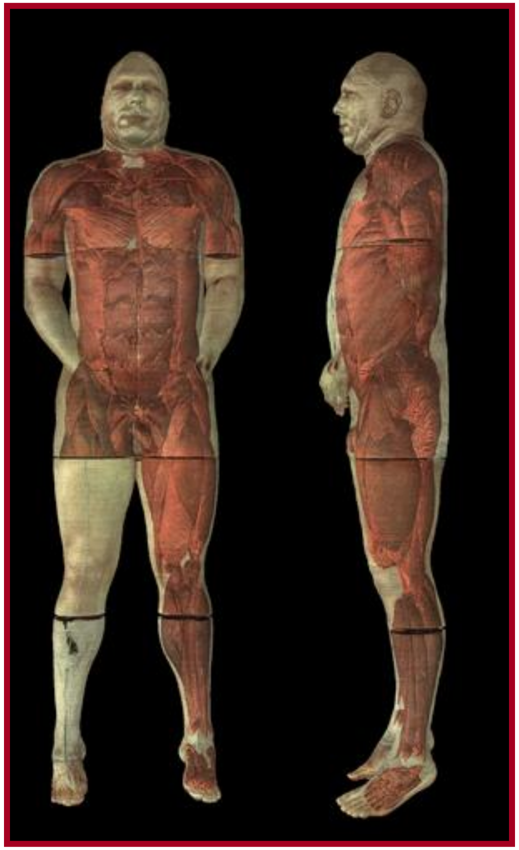
- Well-suited for animation
- Easy to visualize in graphics hardware

Motivation 1

- Some acquisition methods generate solids
  - Example: Medical visualizations



https://www.sciencenewsforstudents.org/

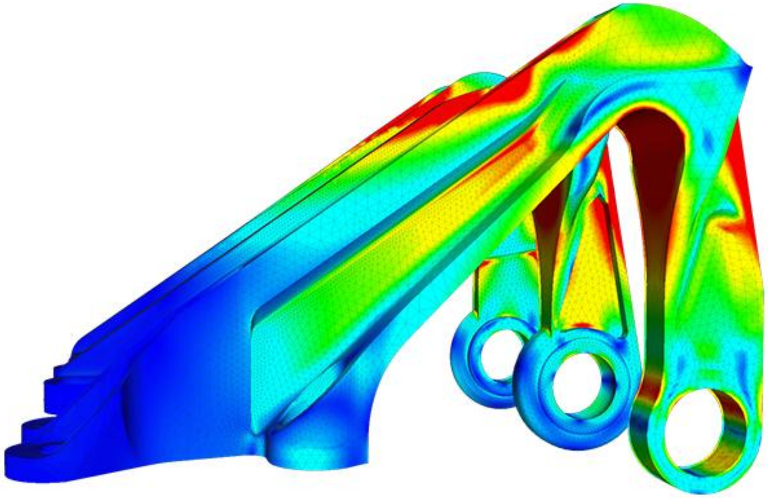


Visible Human

(National Library of Medicine)

Motivation 2

- Some representations require solids
  - Example: FEM simulations

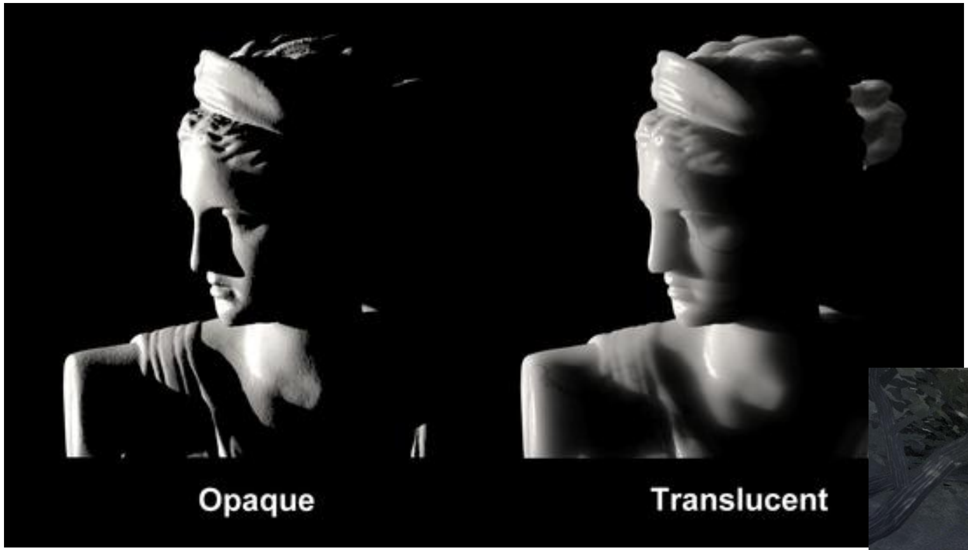


https://www.simscale.com/

[Irving *et al.*, 2007]

Motivation 3

- Some algorithms require solids
  - Example: ray tracing in participating media



http://graphics.stanford.edu/

http://casual-effects.com/

Overview

- Implicit Surfaces
- Voxels
- Quadtrees and Octrees

Implicit Surfaces

Given a real-valued function in 3D, $F(x, y, z)$, the implicit surface defined by F is the collection of points for which $F(x, y, z) = 0$.

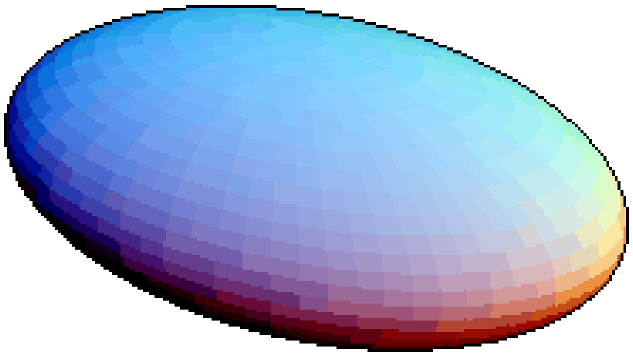
- Example: quadric
  - $F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k$

Implicit Surfaces

Given a real-valued function in 3D, $F(x, y, z)$, the implicit surface defined by F is the collection of points for which $F(x, y, z) = 0$.

- Example: quadric
  - $F(x,y,z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k$

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 - 1 = 0$$



Ellipsoids

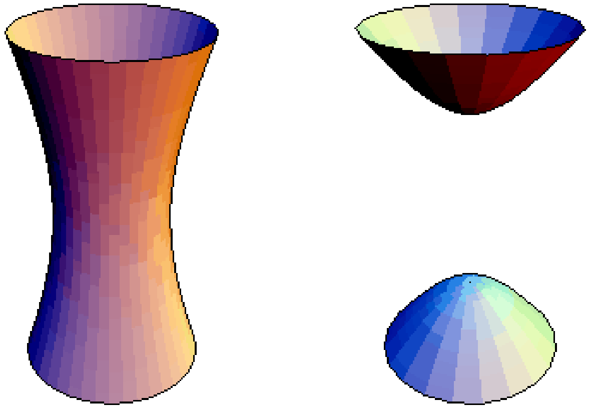
Image courtesy of http://www.geom.uiuc.edu/

Implicit Surfaces

Given a real-valued function in 3D, $F(x, y, z)$, the implicit surface defined by F is the collection of points for which $F(x, y, z) = 0$.

- Example: quadric
  - $F(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k$

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 - \left(\frac{z}{r_z}\right)^2 \pm 1 = 0$$



Hyperboloids

Image courtesy of http://www.geom.uiuc.edu/

Implicit Surfaces

Given a real-valued function in 3D, $F(x,y,z)$, the implicit surface defined by F is the collection of points for which $F(x,y,z)=0$.

- Example: quadric
  - $F(x,y,z)=ax^2+by^2+cz^2+2dxy+2eyz+2fxz+2gx+2hy+2jz+k$

$$\left(\frac{x}{r_x}\right)^2 \pm \left(\frac{y}{r_y}\right)^2 + 2z = 0$$

Paraboloids

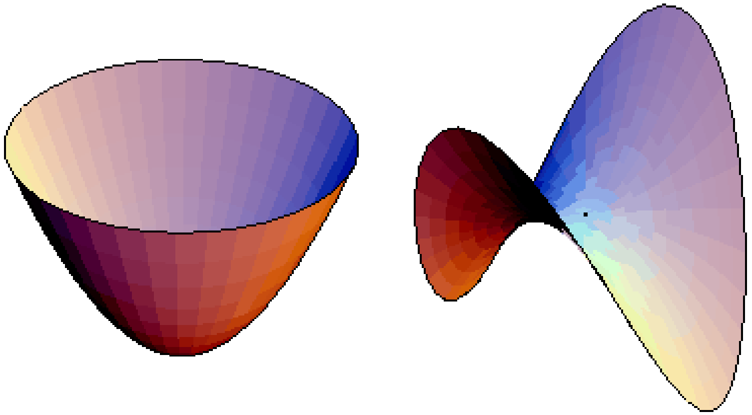


Image courtesy of http://www.geom.uiuc.edu/

Implicit Surfaces

Blobby Models [Blinn '82]

Express the implicit surface as a sum of (signed) Gaussians:

$$F(x,y,z) = \sum_i F_i(x,y,z)$$

$$F_i(x,y,z) = \alpha_i e^{-\left((x-x_i)^2+(y-y_i)^2+(z-z_i)^2\right)/2\sigma_i^2}$$

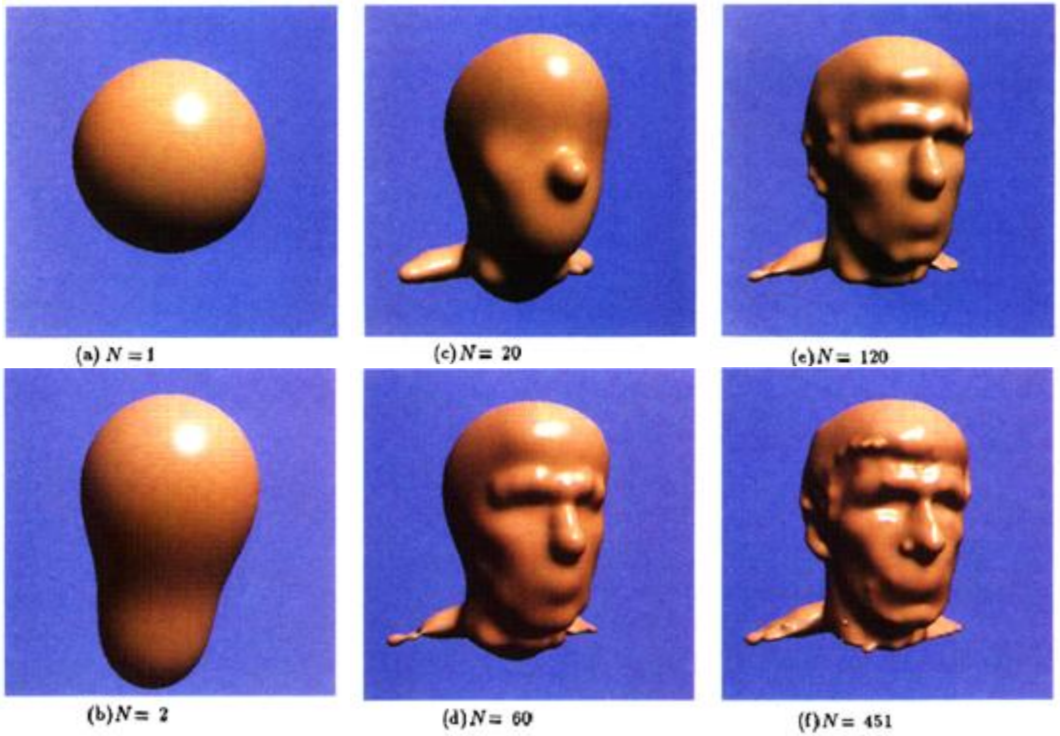
- (x_i, y_i, z_i)
  - Center of the Gaussian
- α_i controls the contribution of the Gaussian
  - How much the Gaussian contributes
  - Interior vs. exterior (sign)
- σ_i controls the width of the Gaussian
  - Extent of the contribution

Implicit Surfaces

Blobby Models [Blinn ‘82]

The more functions we use, the more accurate the reconstruction.

But this also makes the function more difficult to sample.



Muraki, 1991

Implicit Surfaces

$$F(x,y,z) = \sum_i F_i(x,y,z)$$

If the functions F_i are compactly supported, evaluation at a point can be done in sub-linear time.

Chen et al., SIGGRAPH 04

Implicit Surfaces

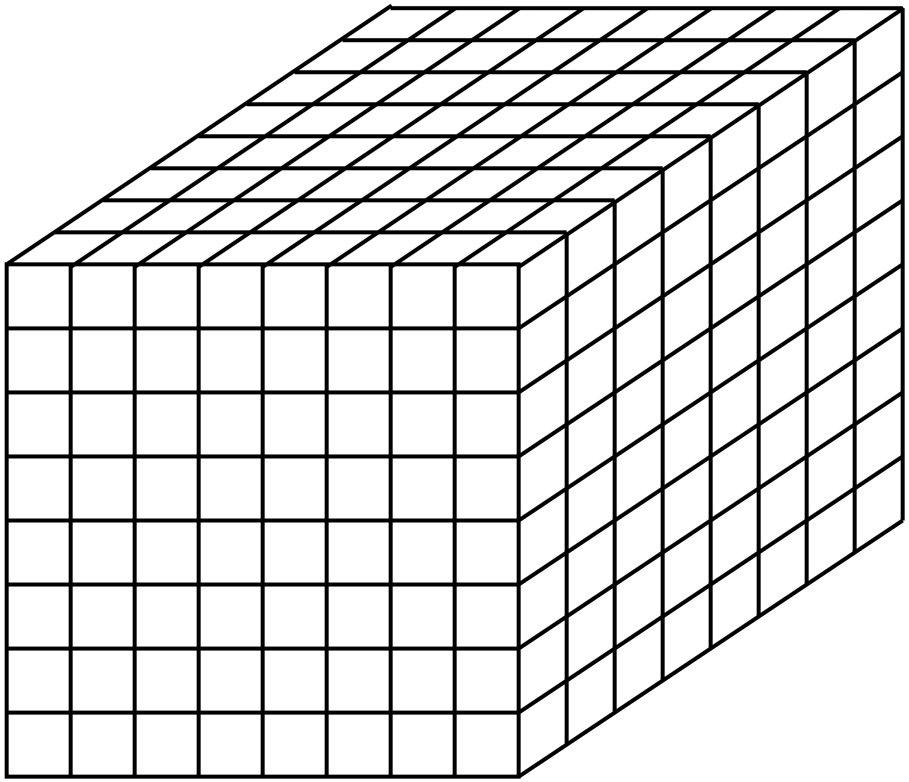
- Advantages:
  - Easy to test if a point is on the boundary of the solid
  - Easy to test if a point is inside the solid
  - Easy to intersect two solids (e.g. collision detection)
- Disadvantages:
  - Hard to describe complex shapes (concisely)
  - Hard to evaluate complex functions (efficiently)
  - Hard to enumerate points on surface

Overview

- Implicit Surfaces
- Voxels
- Quadtrees and Octrees

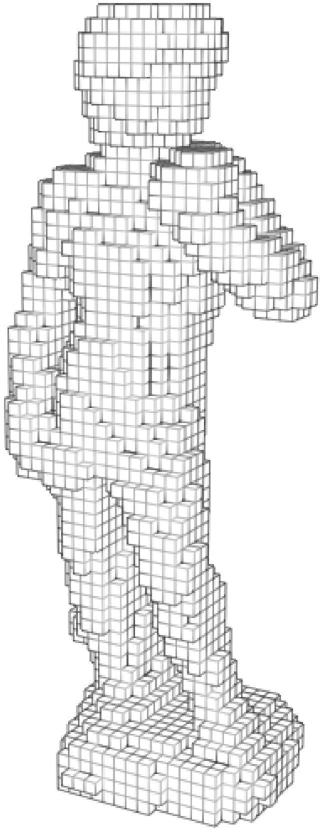
Voxels

- Partition space into uniform grid
  - Grid cells are called *voxels* (volumetric elements)
- Each voxel has a value associated to it.



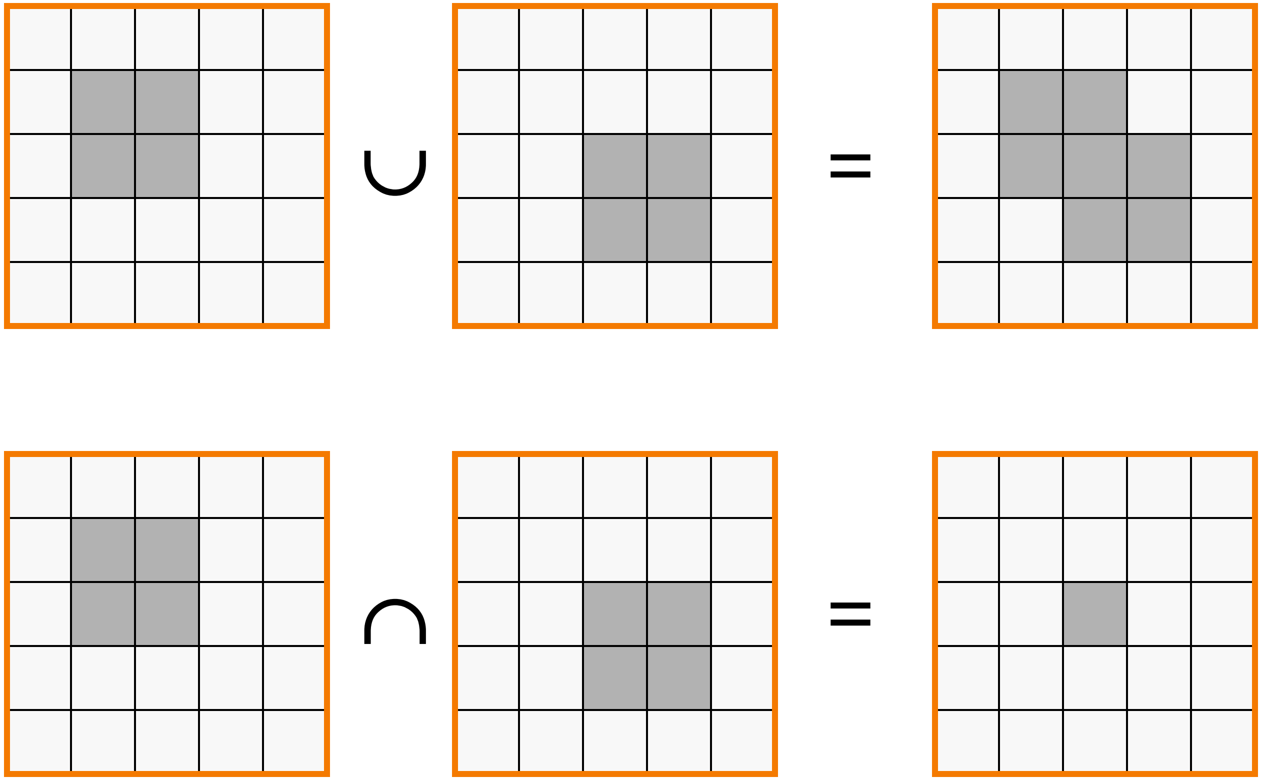
Voxels

- Partition space into uniform grid
  - Grid cells are called *voxels* (volumetric elements)
- Each voxel has a value associated to it.
  - Binary Voxel Grids:
    - » Value is 0 if the voxel is outside the model
    - » Value is 1 if the voxel is inside



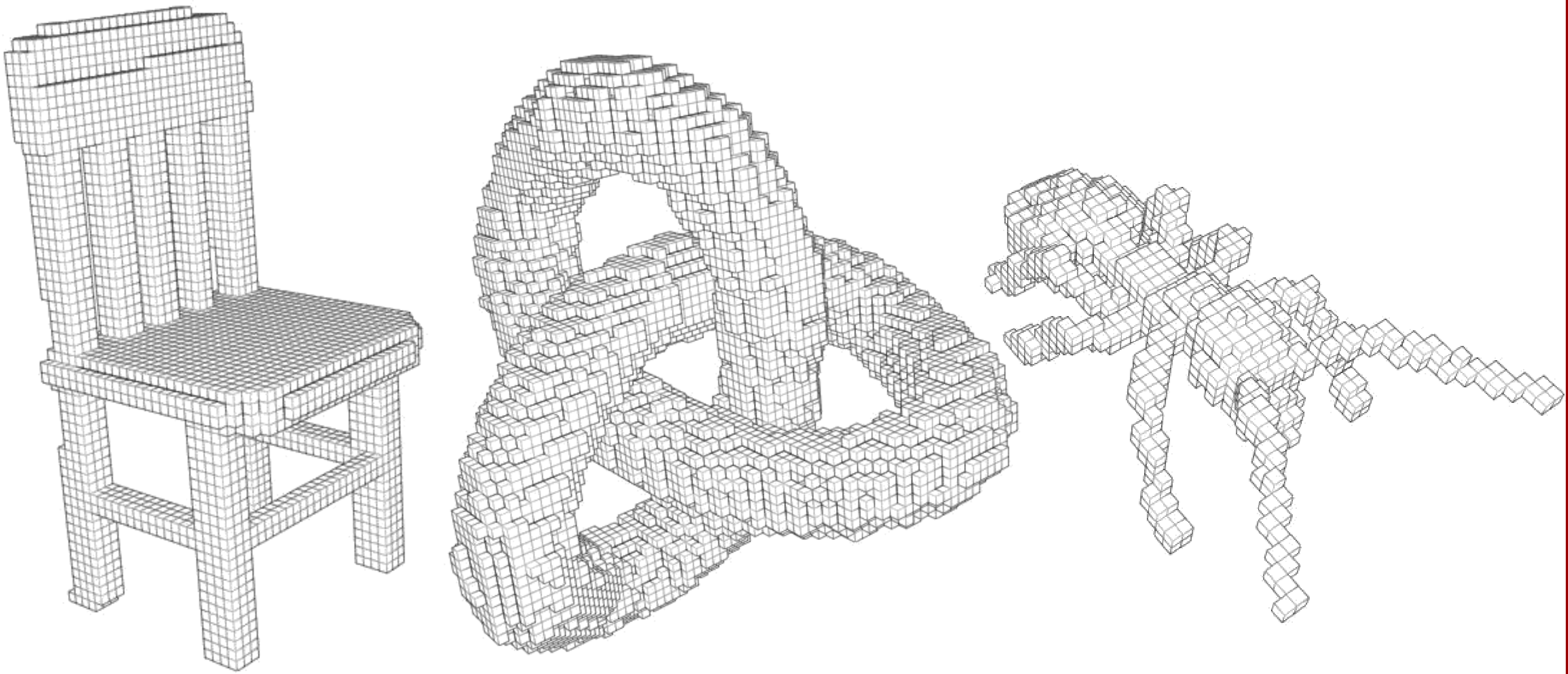
Binary Voxel Boolean Operations

- Compare objects voxel by voxel
  - Trivial



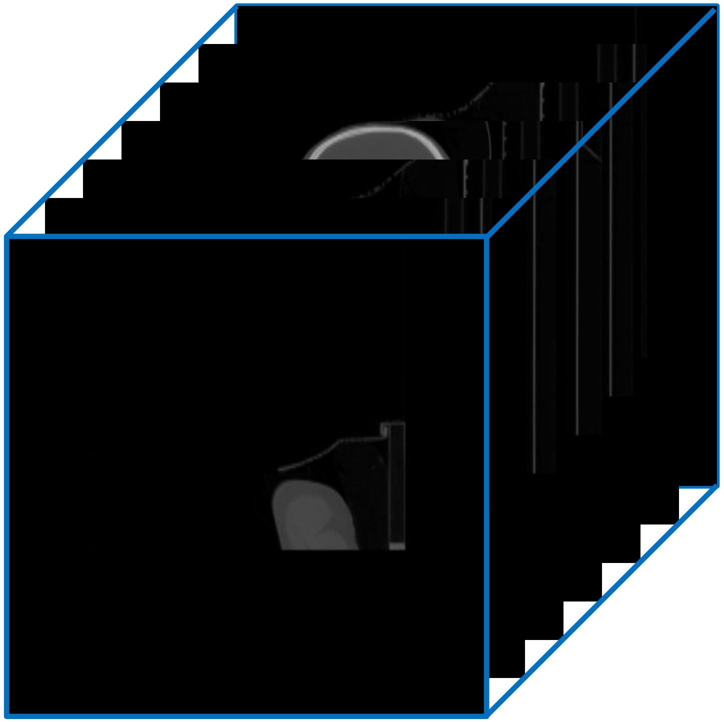
Binary Voxel Visualization

- Draw the faces between on and off voxels.



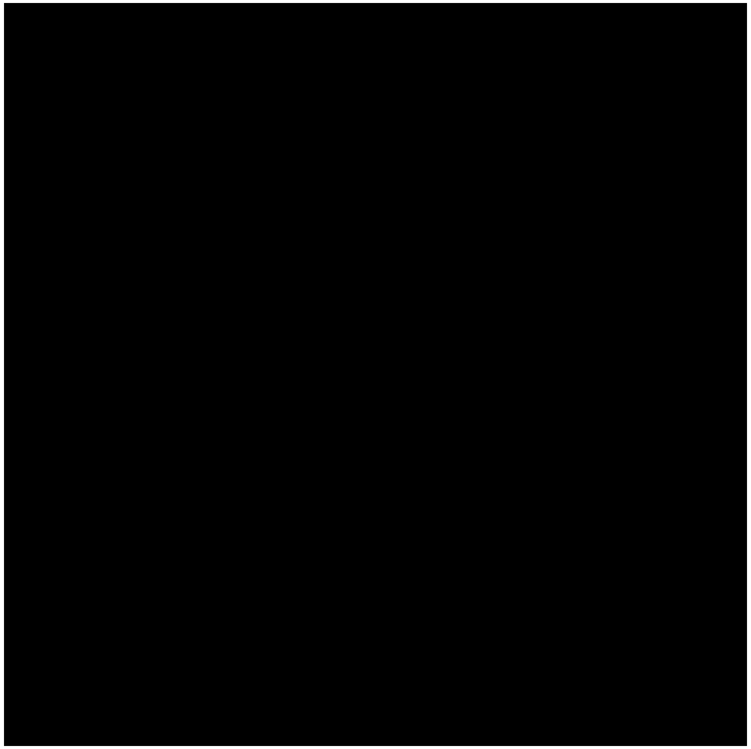
Voxels

- Partition space into uniform grid
  - Grid cells are called *voxels* (volumetric elements)
- Each voxel has a value associated to it.
  - Binary Voxel Grids:
  - Continuous Voxel Grids:
    - » Each voxel stores a continuous value (e.g. density, temperature, color, etc.)



Voxels

- Partition space into uniform grid
  - Grid cells are called *voxels* (like pixels)
- Each voxel has a value associated to it.
  - Binary Voxel Grids:
  - Continuous Voxel Grids:
    - » Each voxel stores a continuous value (e.g. density, temperature, color, etc.)

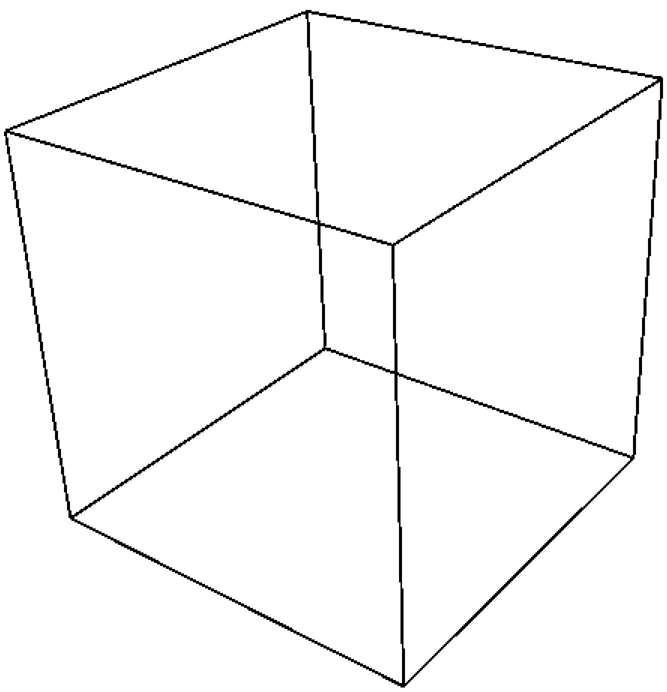


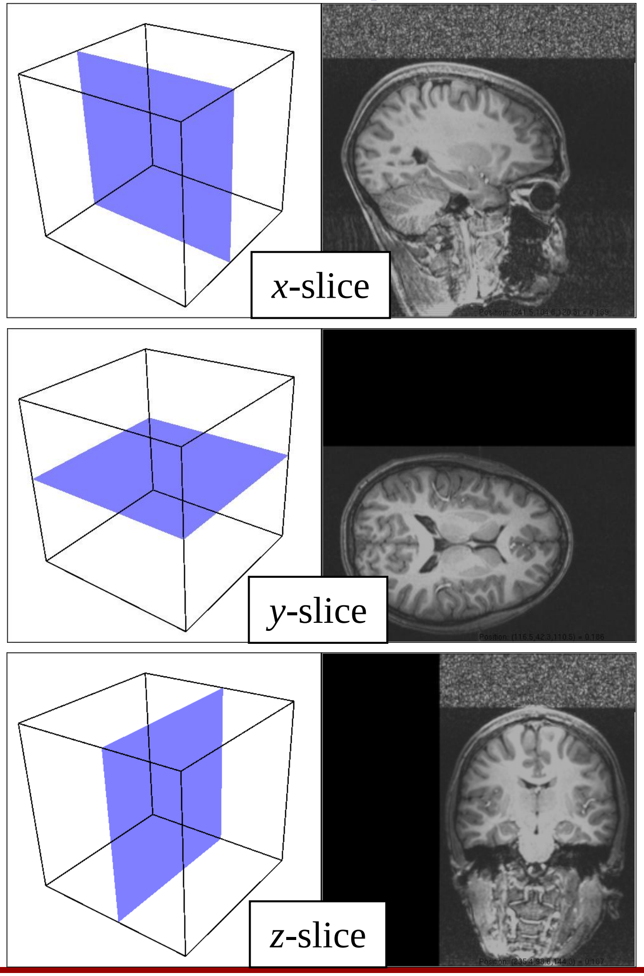
Continuous Voxel Visualization

- Slicing
- Ray-Casting
- Iso-Surface Extraction

Voxel Display

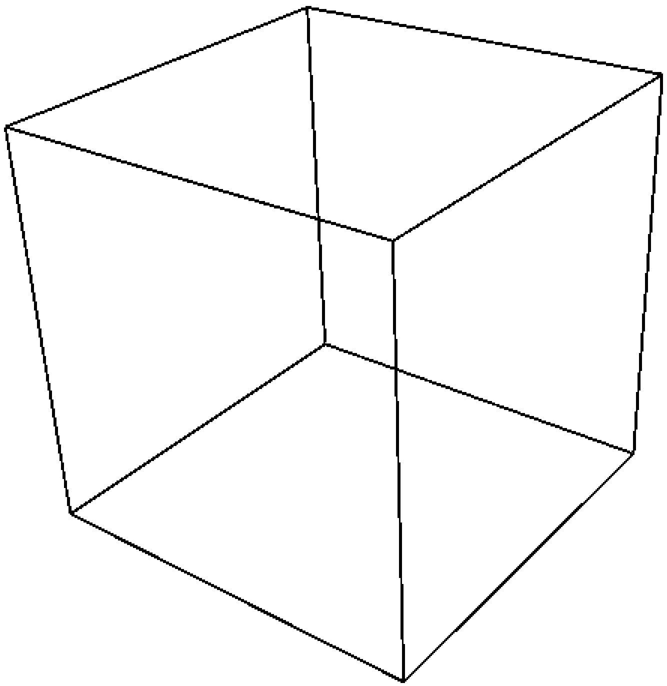
- Slicing
  - Draw 2D image by intersecting voxels with a plane
    - » Supported by 3D texturing on the GPU

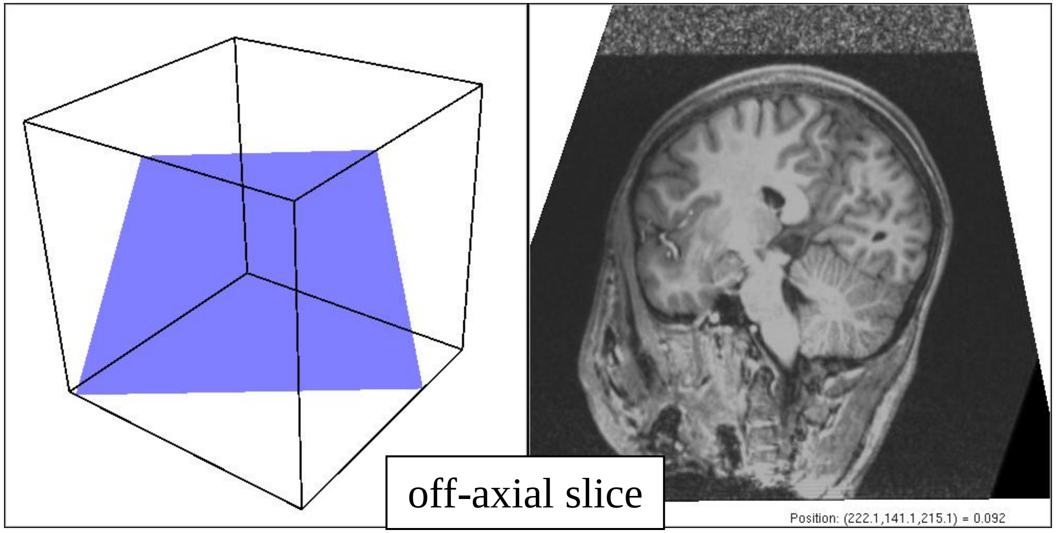




Voxel Display

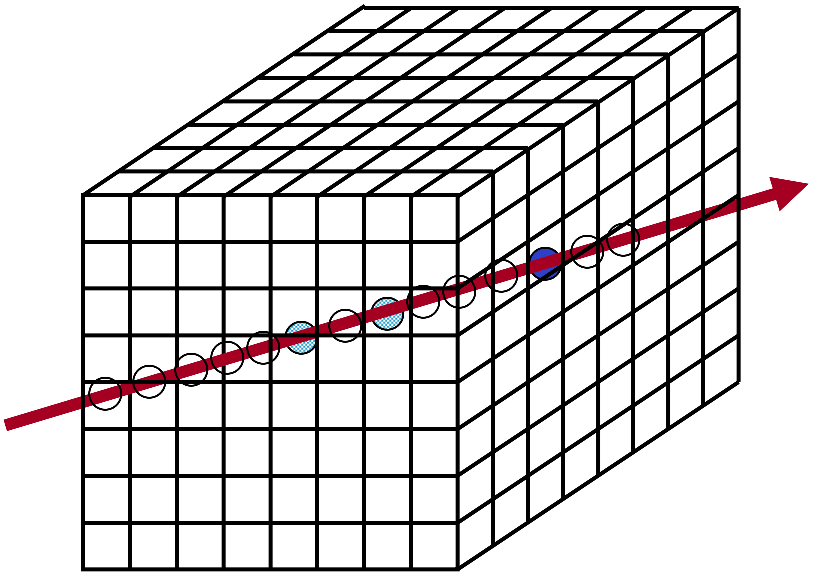
- Slicing
  - Draw 2D image by intersecting voxels with a plane
    - » Supported by 3D texturing on the GPU

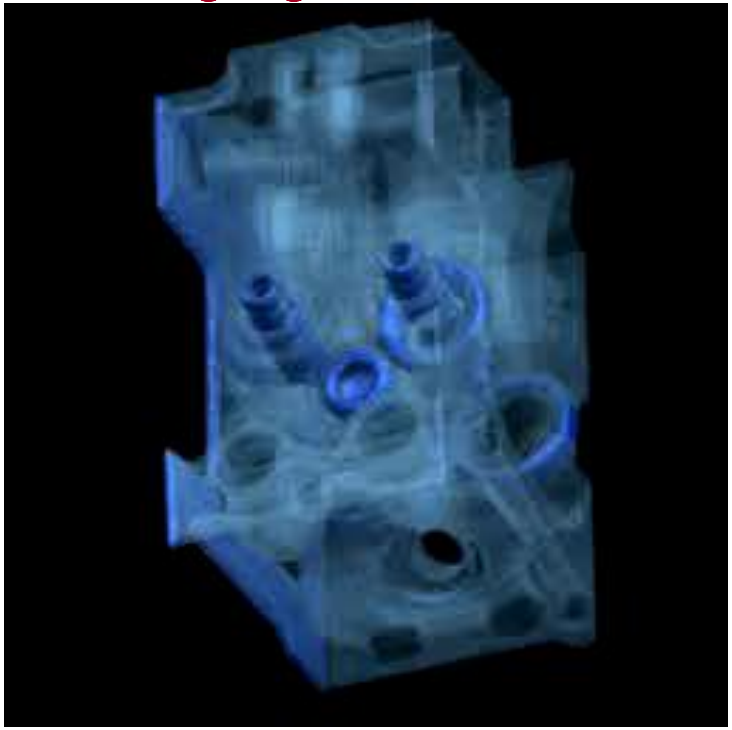




Voxel Display

- Ray casting
  - Integrate density along rays through pixels
    - » Doing this interactively is challenging





Engine Block
Stanford University

Voxel Display

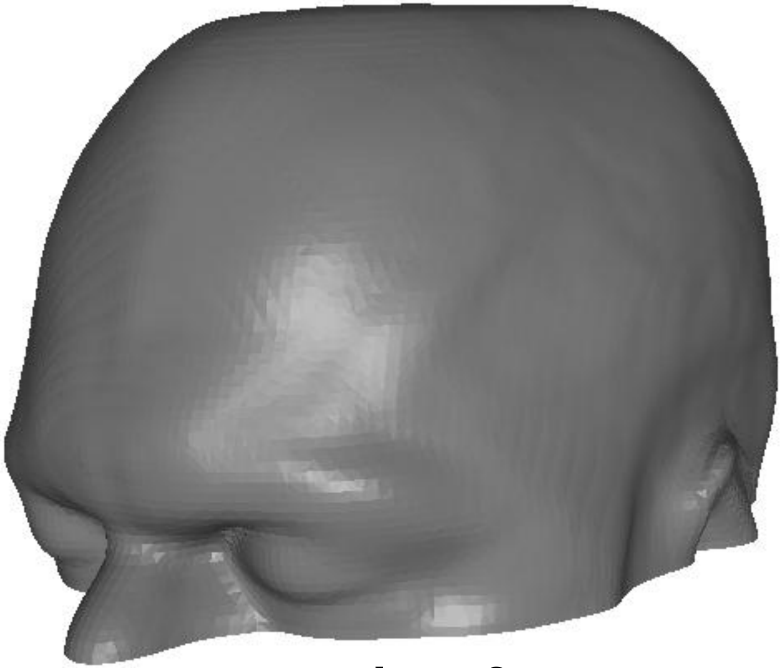
- Ray casting
  - Integrate density along rays through pixels

https://innoarea.com/files/ejemplo_gaussian_splatting.png

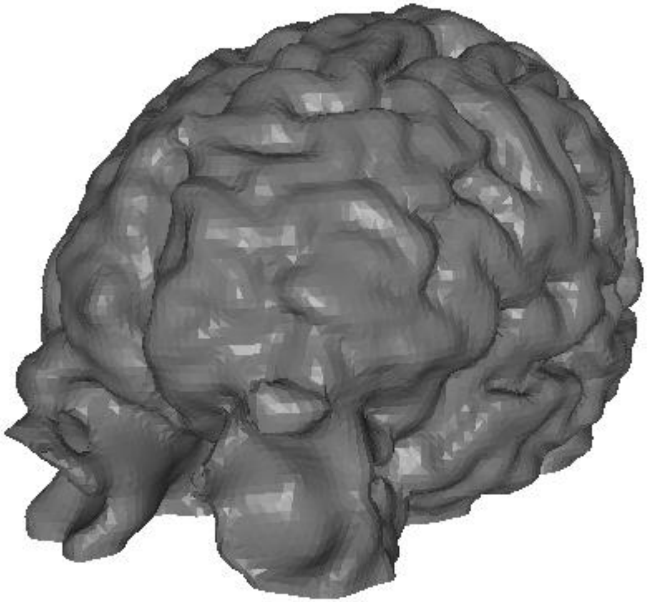
Similar to what's used in NeRF [Mildenhall, 2020] and Gaussian Splats [Kerbl, 2023] do.

Voxel Display

- Iso-Surface Extraction
  - Treat the voxel grid as a regular sampling of a function $F(x, y, z)$, and extract the iso-surface with $F(x, y, z) = \delta$.



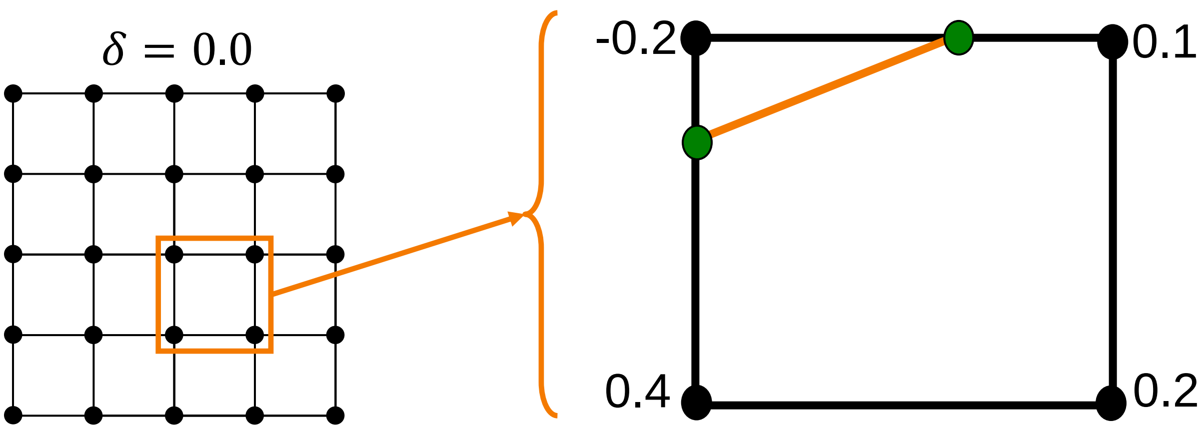
Iso-Value = δ_1



Iso-Value = δ_2

Marching Cubes Algorithm [Lorensen and Cline, '87]

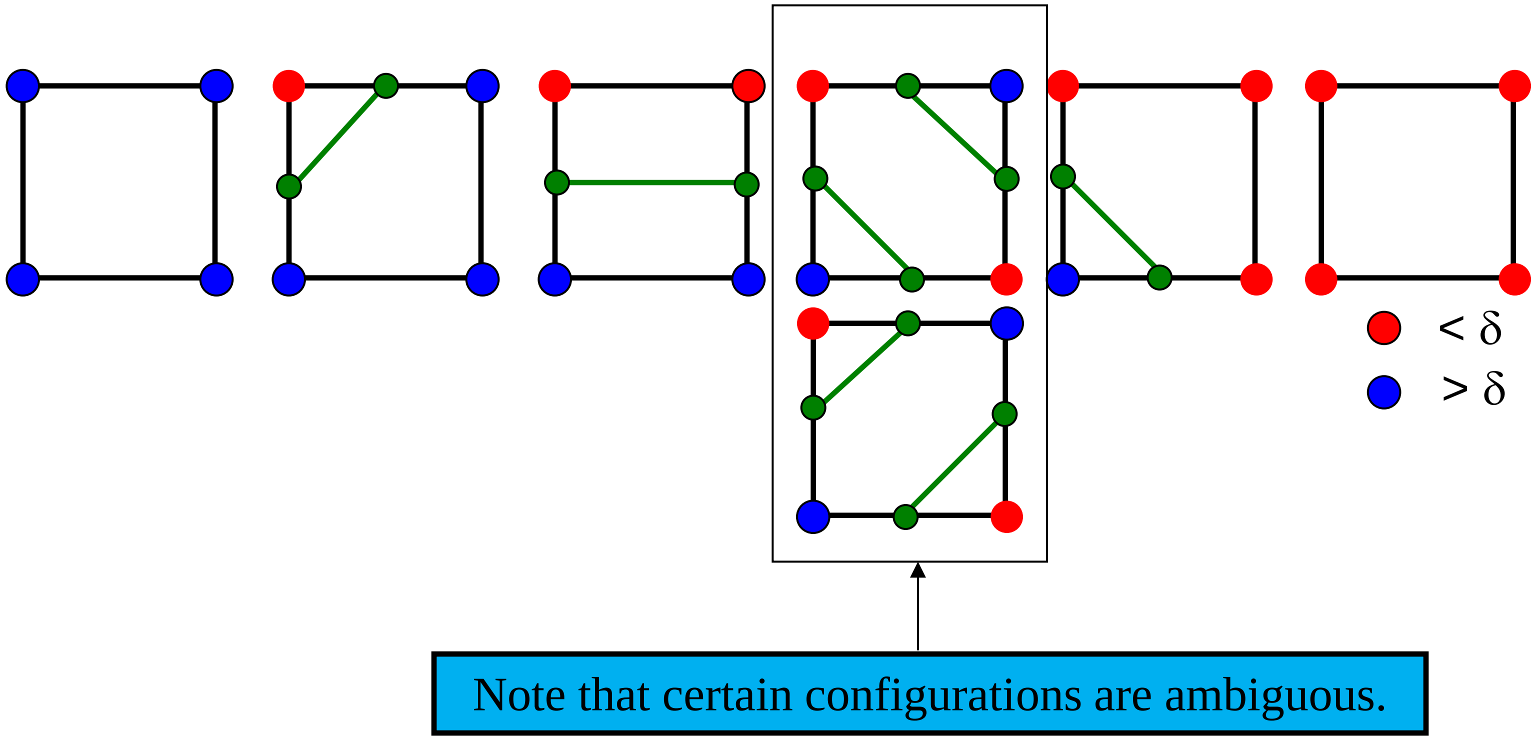
- Iso-Surfaces analog with 2D grid
  - Assume each grid vertex has scalar value
    - » **Iso-Vertices**: If one of the edge vertices has value larger than δ and the other has value less than δ, find the point on the edge whose linear interpolation equals δ.
    - » **Iso-Edge**: Per cell, connect iso-vertices with line segments.



Note: The number of cell edges at which we insert vertices must be even

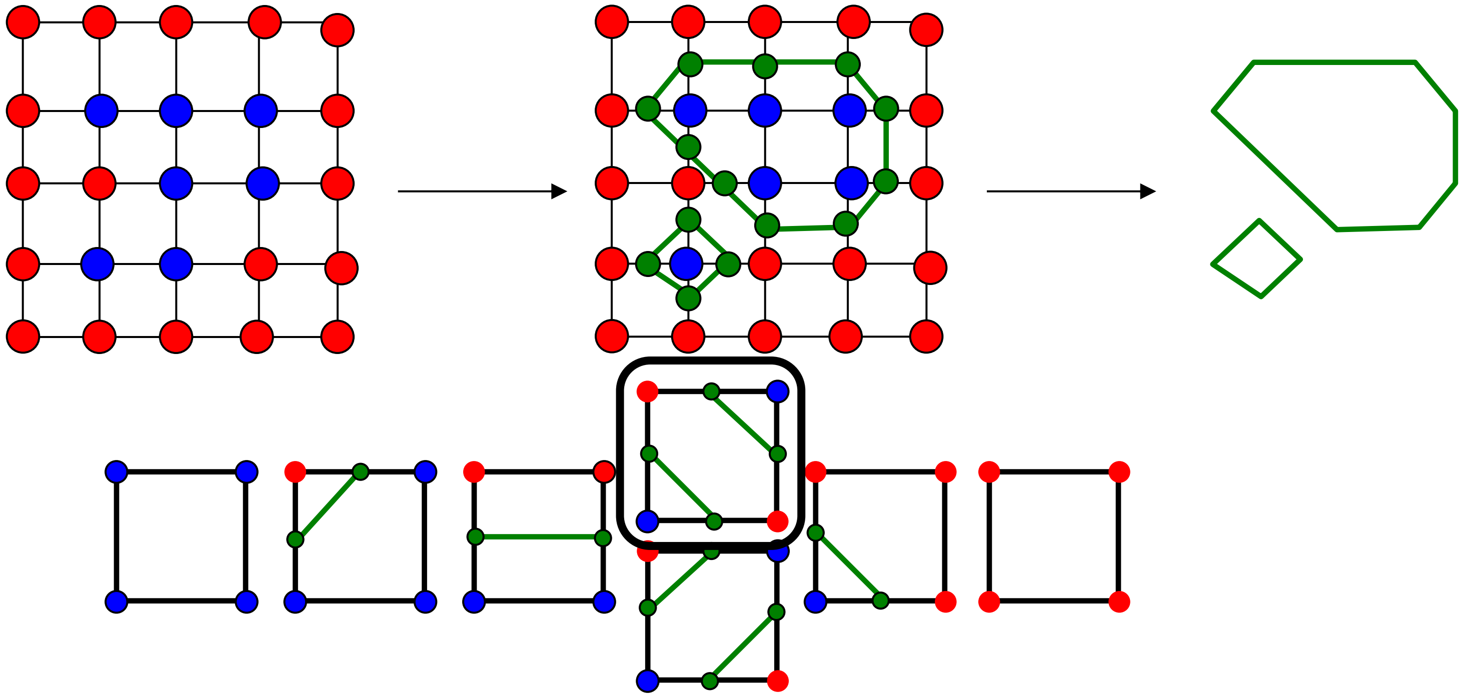
Marching Cubes Algorithm [Lorensen and Cline, '87]

- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case



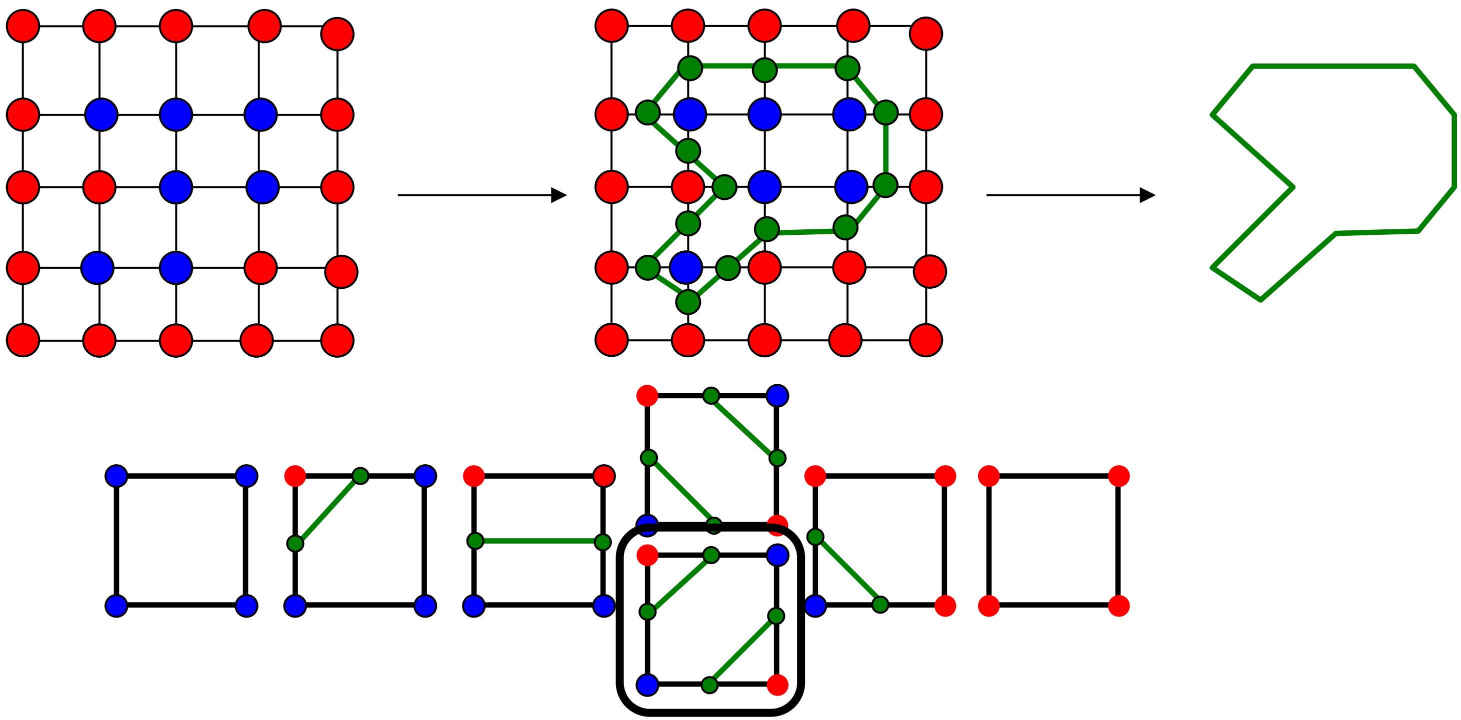
Marching Cubes Algorithm [Lorensen and Cline, ’87]

- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case
  - Combine the iso-vertices from the different cells



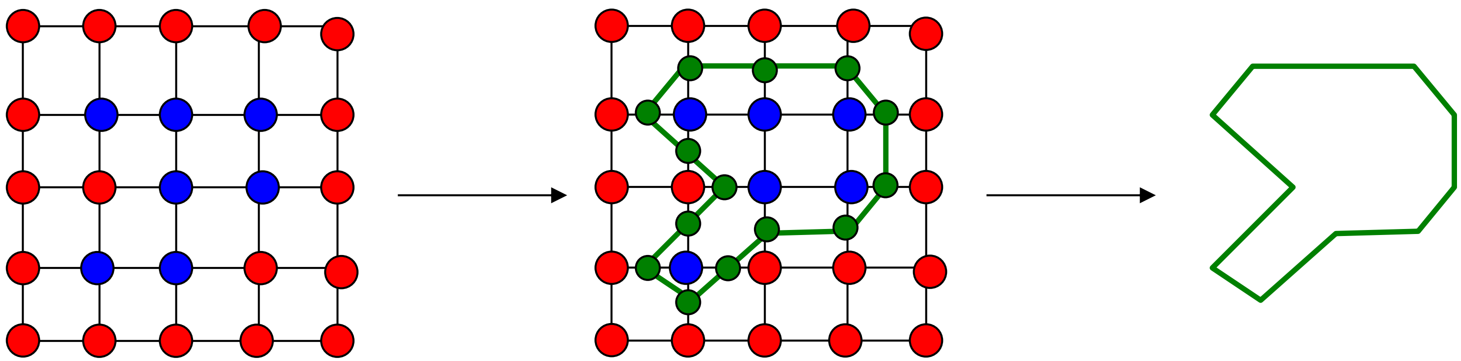
Marching Cubes Algorithm [Lorensen and Cline, '87]

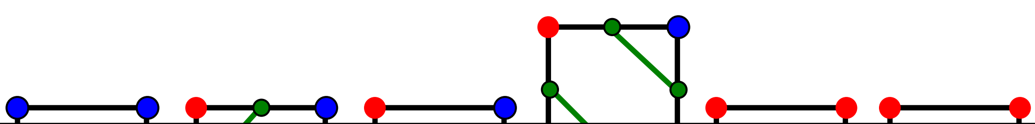
- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case
  - Combine the iso-vertices from the different cells



Marching Cubes Algorithm [Lorensen and Cline, ’87]

- Iso-Surfaces analog with 2D grid
  - Break up into the $2^4 = 16$ different possible cases
  - Assign a rule for curve extraction in each case
  - Combine the iso-vertices from the different cells





As long as the geometry shared by adjacent cells (e.g. position of iso-vertices) is defined by values along the shared edge, adjacent cells will define consistent (connected) segments.

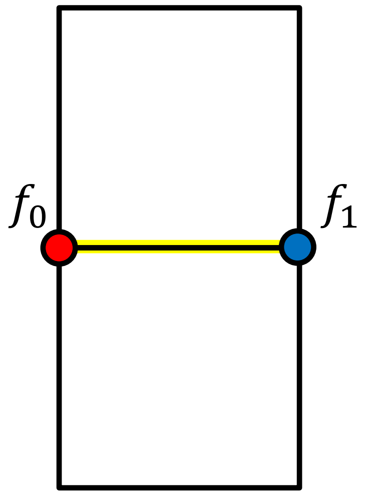
Marching Cubes Algorithm [Lorensen and Cline, '87]

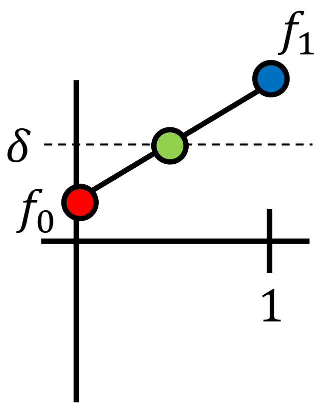
Assigning iso-vertex positions (linear):

Given values f_0 and f_1 at the endpoints of an edge we can fit a linear interpolant:

$$\boldsymbol{F}(t) = f_0 \cdot (1-t) + f_1 \cdot t$$

$\Rightarrow$ The function has value $\boldsymbol{F}(t) = \delta$ at $t = \frac{\delta - f_0}{f_1 - f_0}$.





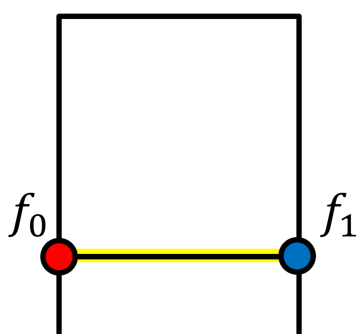
Marching Cubes Algorithm [Lorensen and Cline, '87]

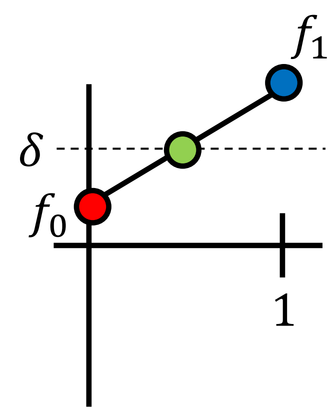
Assigning iso-vertex positions (linear):

Given values f_0 and f_1 at the endpoints of an edge we can fit a linear interpolant:

$$\boldsymbol{F}(t) = f_0 \cdot (1 - t) + f_1 \cdot t$$

$\Rightarrow$ The function has value $\boldsymbol{F}(t) = \delta$ at $t = \frac{\delta - f_0}{f_1 - f_0}$.





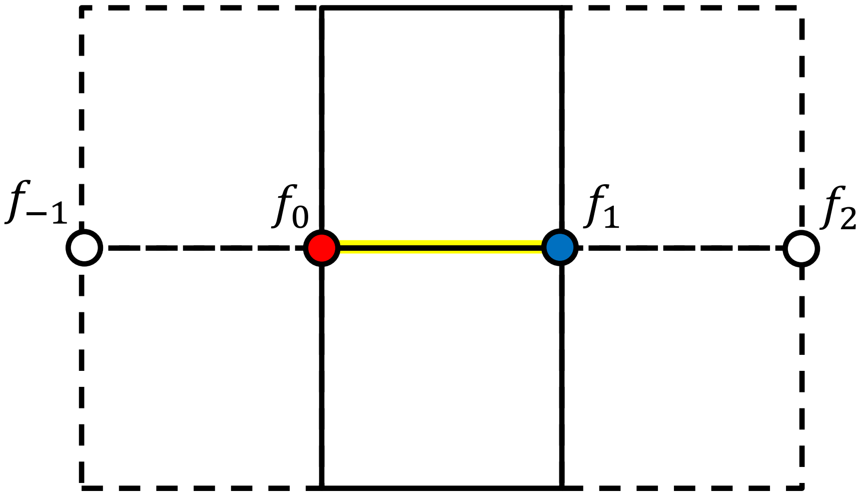
Note:
Since both cells see the same values along the shared edge, they both define the same iso-vertex.

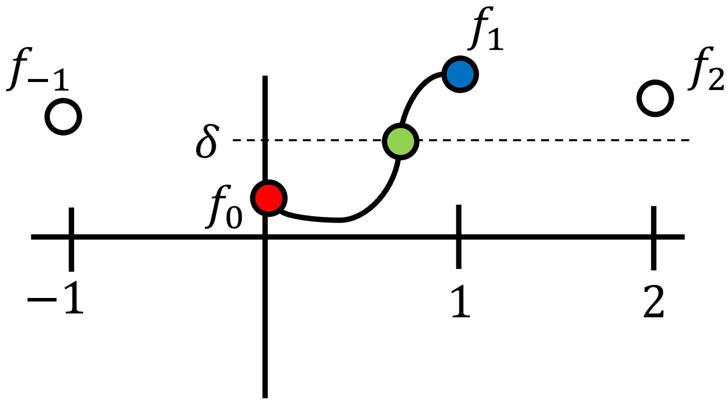
Marching Cubes Algorithm [Lorensen and Cline, ’87]

<u>Assigning iso-vertex positions (cubic)</u>:

If we also know f_{-1} and f_2 we can fit a Cardinal B-spline to the four values and find the root(s) of the cubic polynomial in the range $[0,1]$:

$$\boldsymbol{F}(t) = f_{-1} \cdot BF_0(t) + f_0 \cdot BF_1(t) + f_1 \cdot BF_2(t) + f_2 \cdot BF_3(t)$$





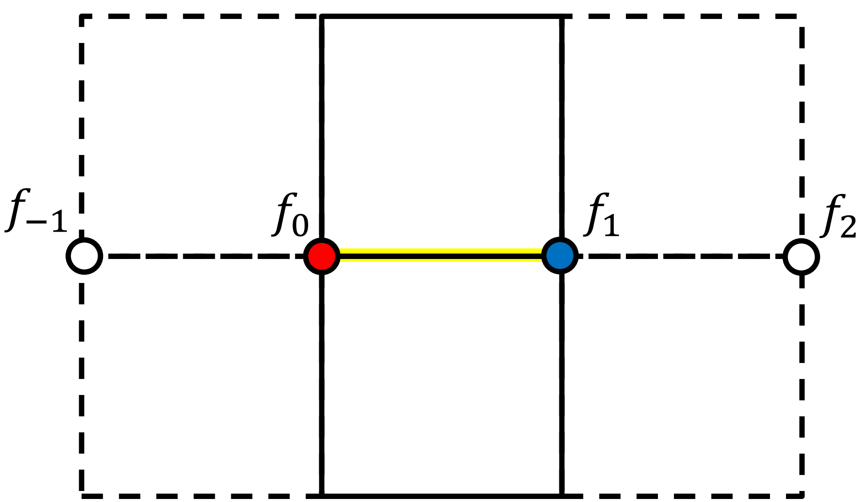
Marching Cubes Algorithm [Lorensen and Cline, ’87]

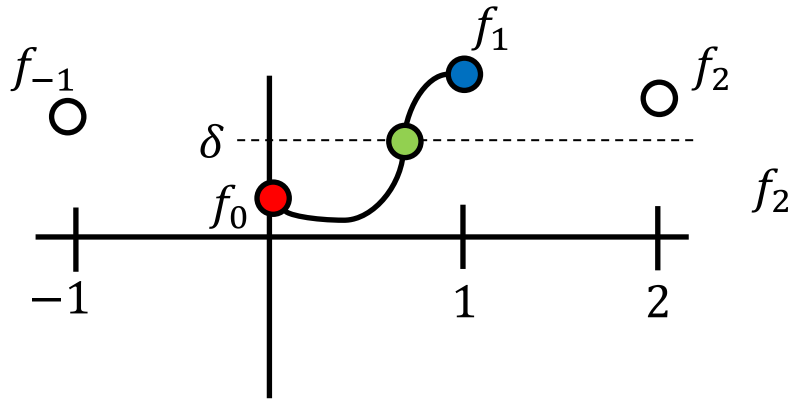
Note:

Since the spline is interpolating, if $f_0 < \delta$ and $f_1 > \delta$ (or vice-versa) $\boldsymbol{F}(t)$ will have an odd number of roots in [0,1].

Because it is cubic, $\boldsymbol{F}(t)$ may have 3 roots in [0,1].
⇒ We need to choose the zero-crossing consistently.

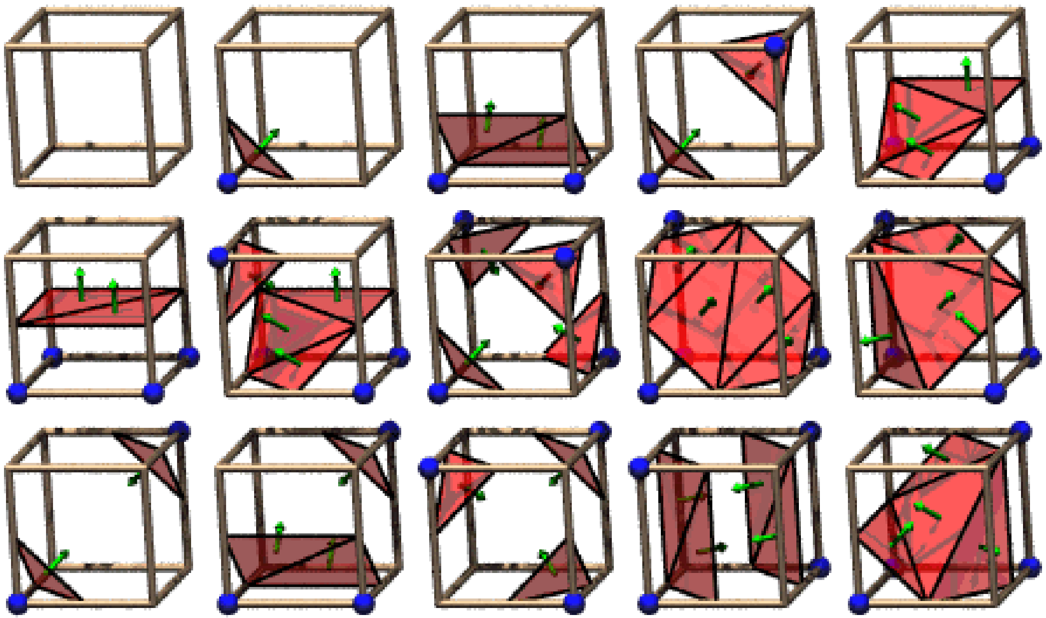
⇒ Same number of iso-vertices as linear interpolation.





Marching Cubes Algorithm [Lorensen and Cline, ’87]

- Iso-Surface with 3D grid
  - Break up into the $2^8 = 256$ different possible cases
  - Assign a rule for surface extraction in each case
  - Combine the iso-triangles from the different cells

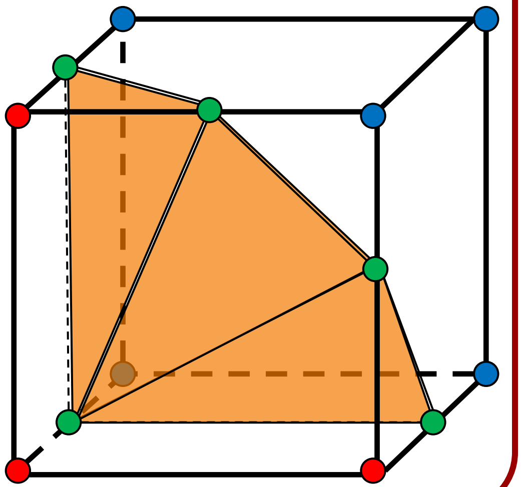


The 15 Cube Combinations

Back in the day, a table of 2^8 configurations was too much to store in memory. Leveraging symmetry, [Lorensen and Cline, 1987] reduced it to 15 cases.

Marching Cubes Algorithm (Informally)

- <u>Inductively</u>:
  - Per edge (1D)
    - » Iso-vertices: Get edge δ-crossings
  - Per face (2D)
    - » Iso-edges: Connect iso-vertices in each face
  - Per cell (3D)
    - » Iso-triangles: Merge iso-edges and triangulate

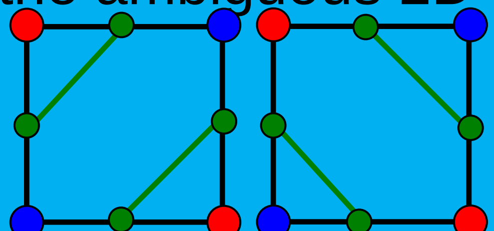


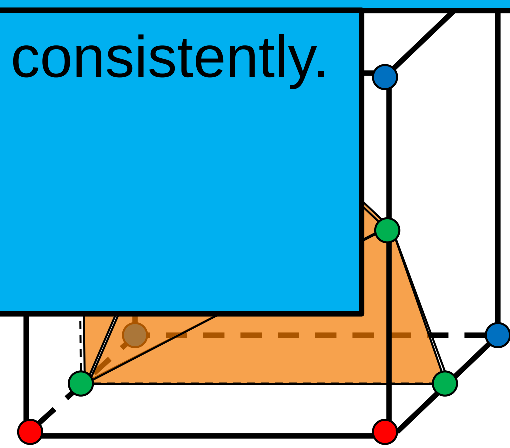
Marching Cubes Algorithm (Informally)

- Inductively:
  - Per edge (1D)
    - » Iso-vertices: Get edge δ-crossings
  - Per face (2D)
    - » Iso-edges: Connect iso-vertices in each face
  - Per cell (3D)

2D: Adjacent cells need to match iso-vertices across the shared edge.
3D: Adjacent cells need to match iso-edges across the shared face.

⇒ Have to resolve the ambiguous 2D cases consistently.





Voxels

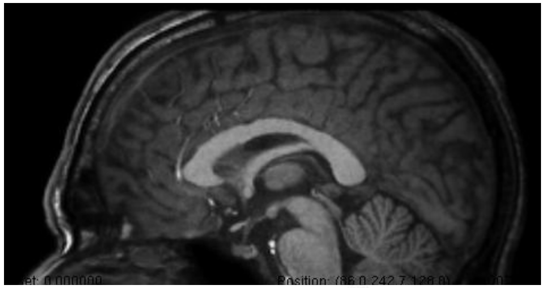
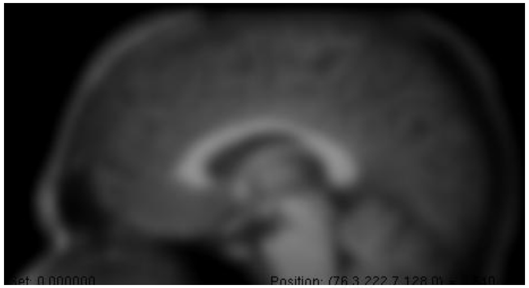
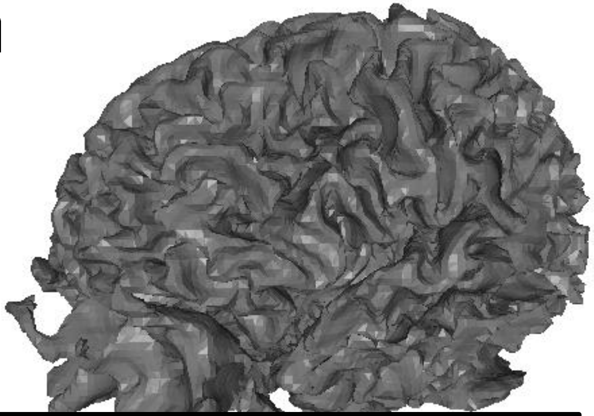
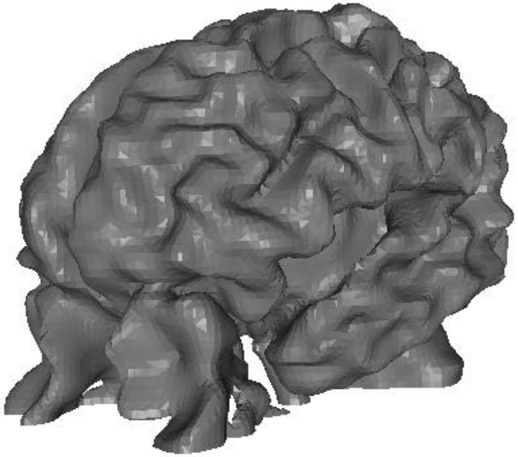
Continuous voxel grids are 3D images. Operations that we applied to 2D images can also be applied to voxel grids:

- Sampling
- Contrast
- Edge detection
- Smoothing

Voxels

Continuous voxel grids are 3D images. Operations that we applied to 2D images can also be applied to voxel grids:

- Sampling
- Contrast
- Edge detection
- **Smoothing**

Demo

Voxels

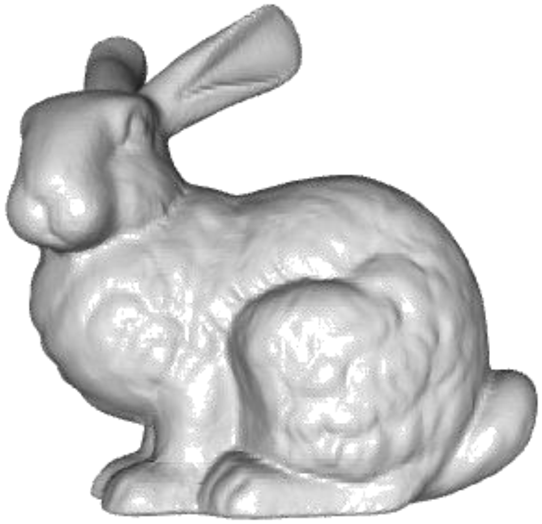
- Advantages
  - Simple
  - Same complexity for all objects
  - Natural acquisition for some applications
  - Trivial boolean operations
- Disadvantages
  - Approximate
  - Not affine invariant
  - Large storage requirements
  - Expensive display

Solid Modeling Representations

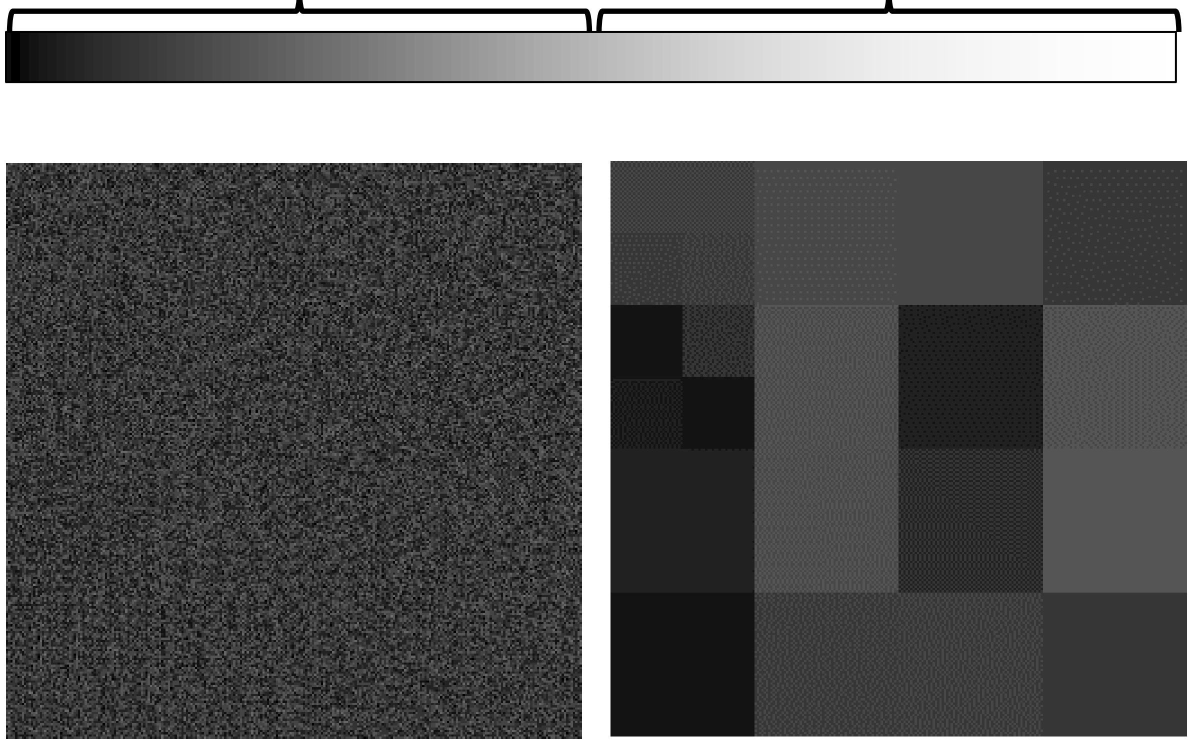
- Implicit Surfaces
- Voxels
- Quadtrees & Octrees

Quadtrees(2D) & Octrees(3D)

- Refine resolution of voxels hierarchically



Outside Inside

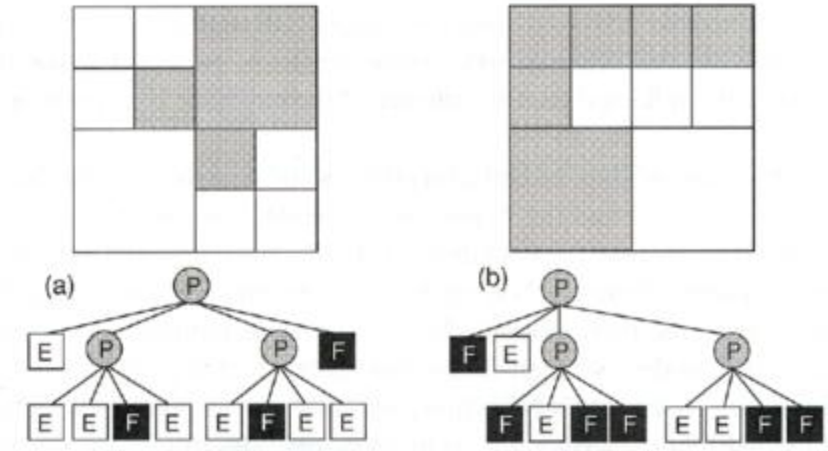


Uniform Voxels Octree

Quadtrees(2D) and Octrees(3D)

<u>Cell ordering</u>:
1. bottom left
2. bottom right
3. top left
4. top right

A



B

- Expected complexity:
  number of nodes ≅ to perimeter or surface area

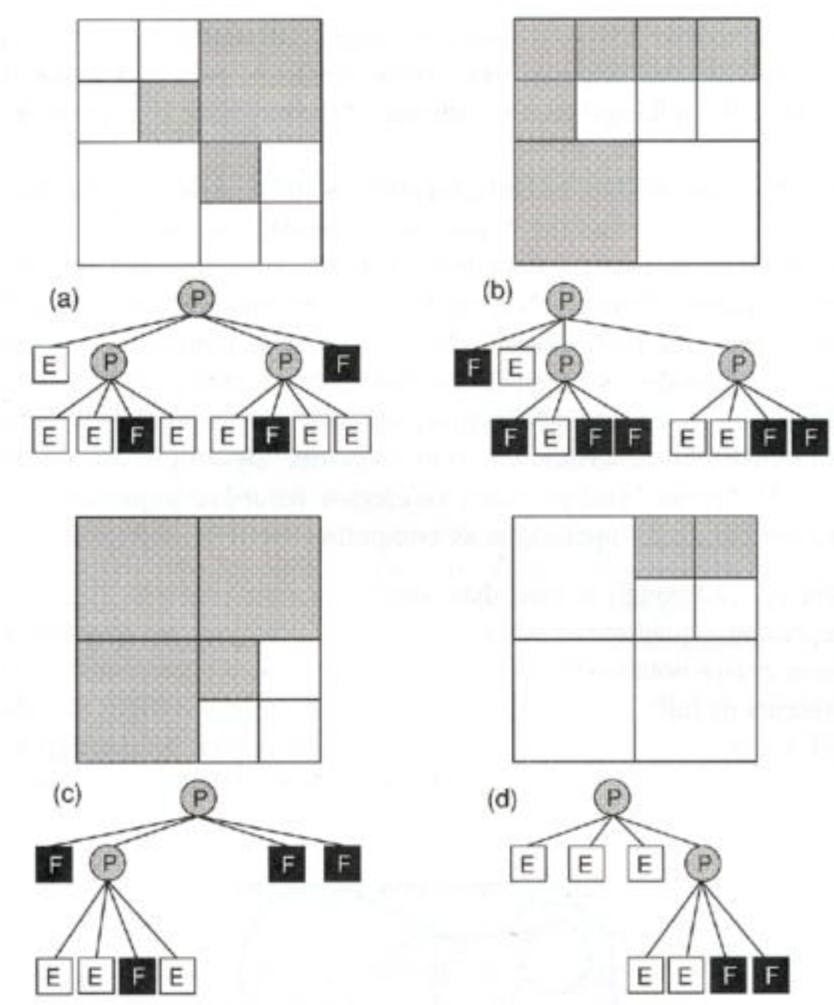
FvDFH Figure 12.24

Binary Quadtree Boolean Operations

<u>Cell ordering</u>:
1. bottom left
2. bottom right
3. top left
4. top right

A

B



$A \cup B$

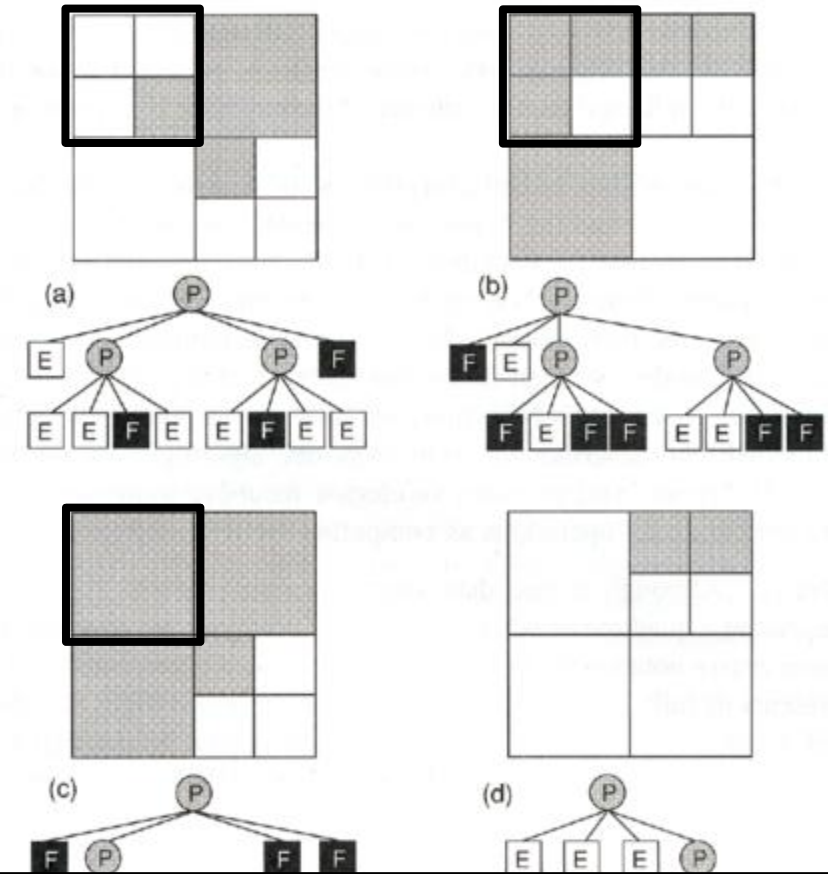
$A \cap B$

FvDFH Figure 12.24

Binary Quadtree Boolean Operations

<u>Cell ordering</u>:
1. bottom left
2. bottom right
3. top left
4. top right

A ***B***

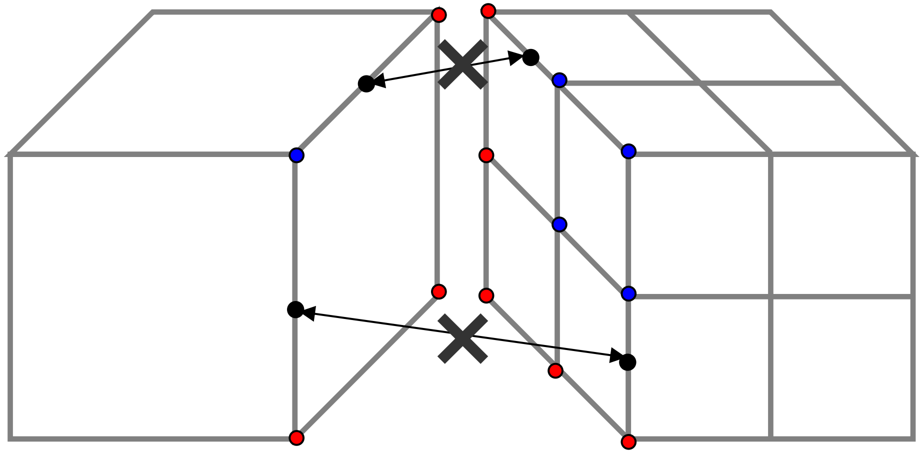


A* ∪ *B ***A* ∩ *B***

If the operation results in <u>all</u> child nodes being marked empty/full
⇓
Remove the children and mark the parent

Octree Display

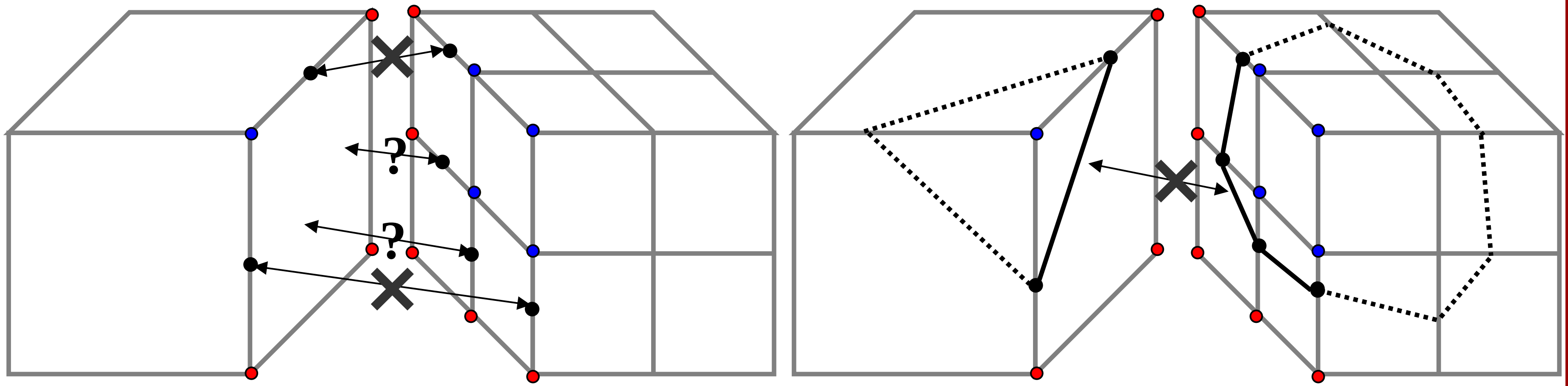
- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction



How to define positions of δ-crossings along edges shared by cells at different resolutions?

Octree Display

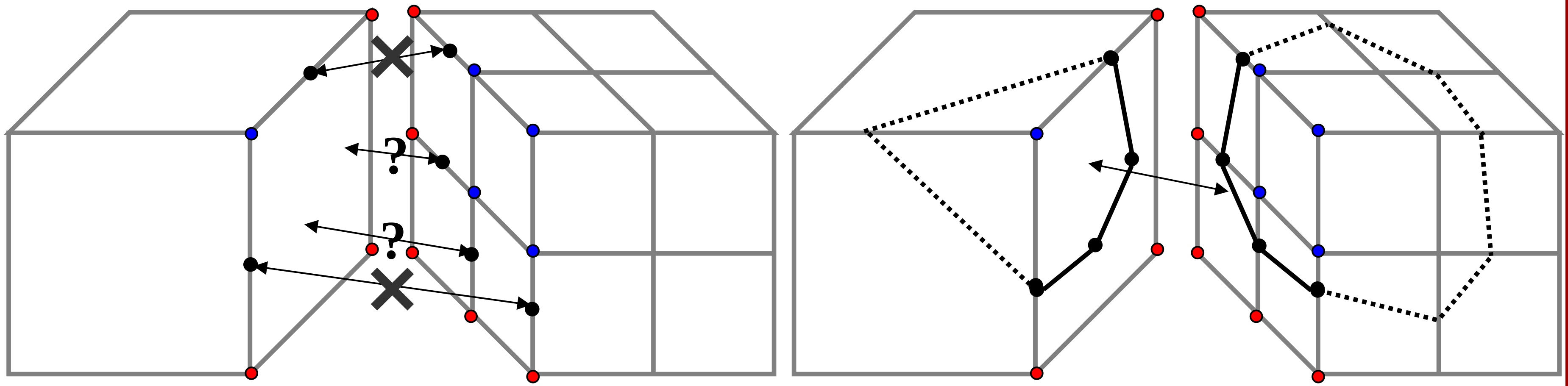
- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction



How to handle the situation when vertices on one side of a face do not exist on the other?

Octree Display

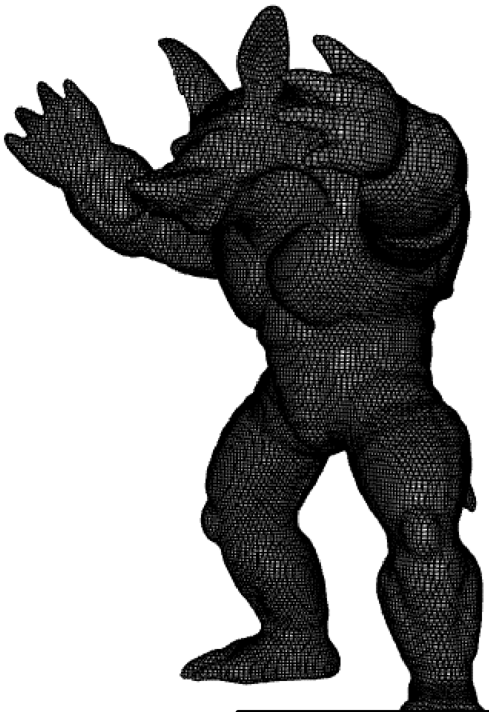
- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction

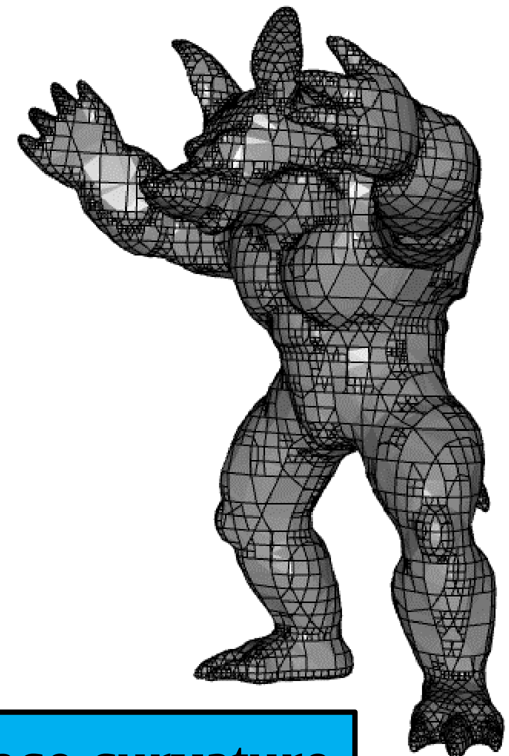


General Approach:
Copy from the finer faces to the coarser one.

Octree Display

- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction

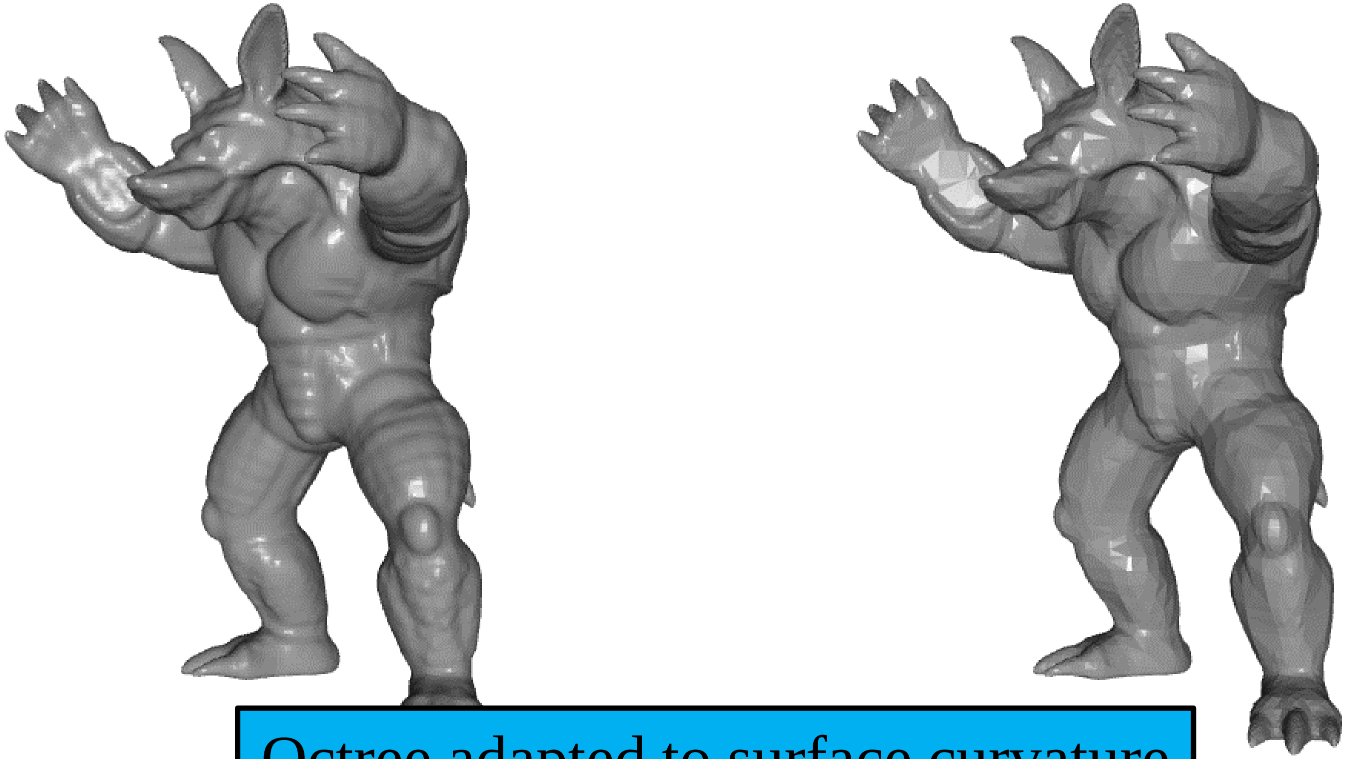




Octree adapted to surface curvature

Octree Display

- Extend voxel methods
  - Slicing
  - Ray casting
  - Iso-surface extraction



Octree adapted to surface curvature