# Solid Modeling

Sai-Kit Yeung

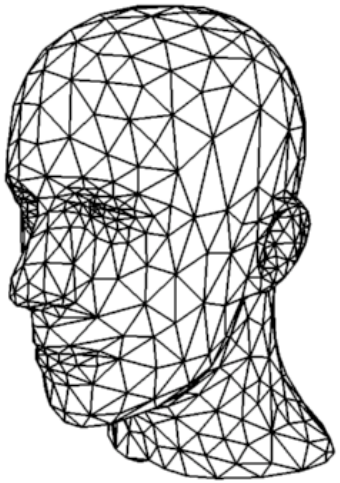
ISTD, SUTD

Notes courtesy by Wojciech Matusik

# Boundary Representations (B-Reps)

- Only boundary of an object is specified
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- This is what we have seen in 50.017

Martin Newell



Leif Kobbelt

# Solid Modeling

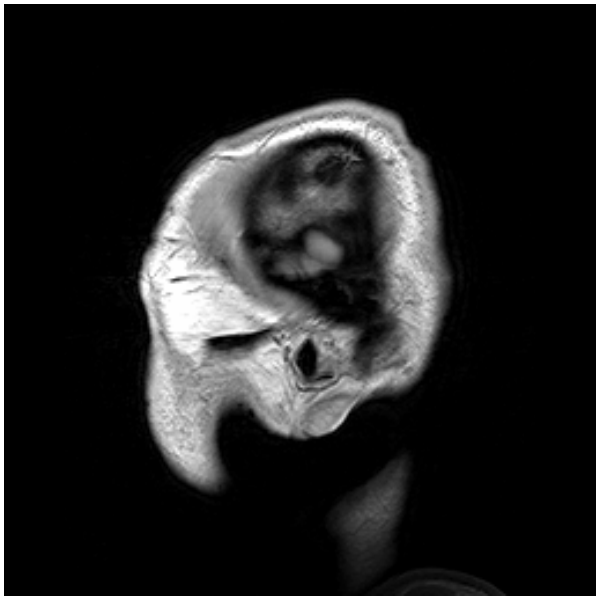
- Represent solid interiors of objects

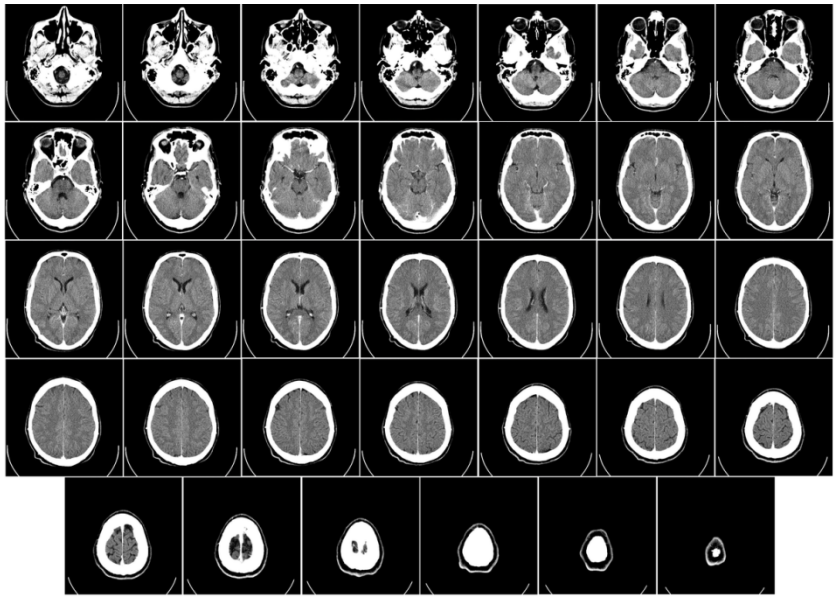
Visible Human
(National Library of Medicine)

SUNY Stony Brook

# Why Volumetric Representations?

- Some acquisition methods generate solids
  - Magnetic Resonance Imaging (MRI)
  - Computed Tomography (CT/ CAT)



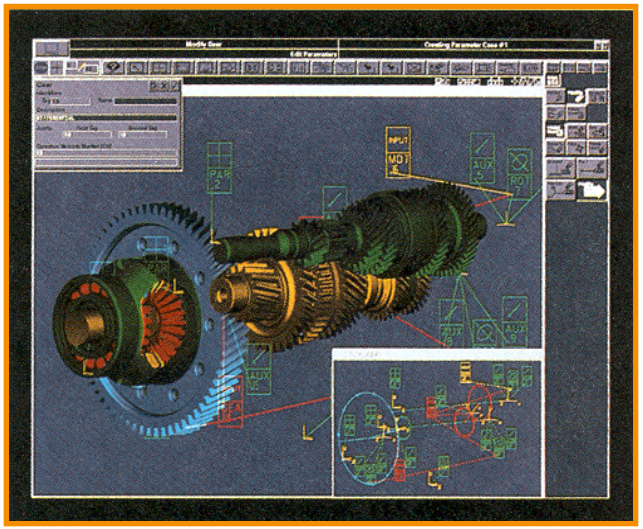


Source: Wikipedia

# Why Volumetric Representations?

- Some applications require solids
  - CAD/CAM
  - material(s) need to be specified inside the object

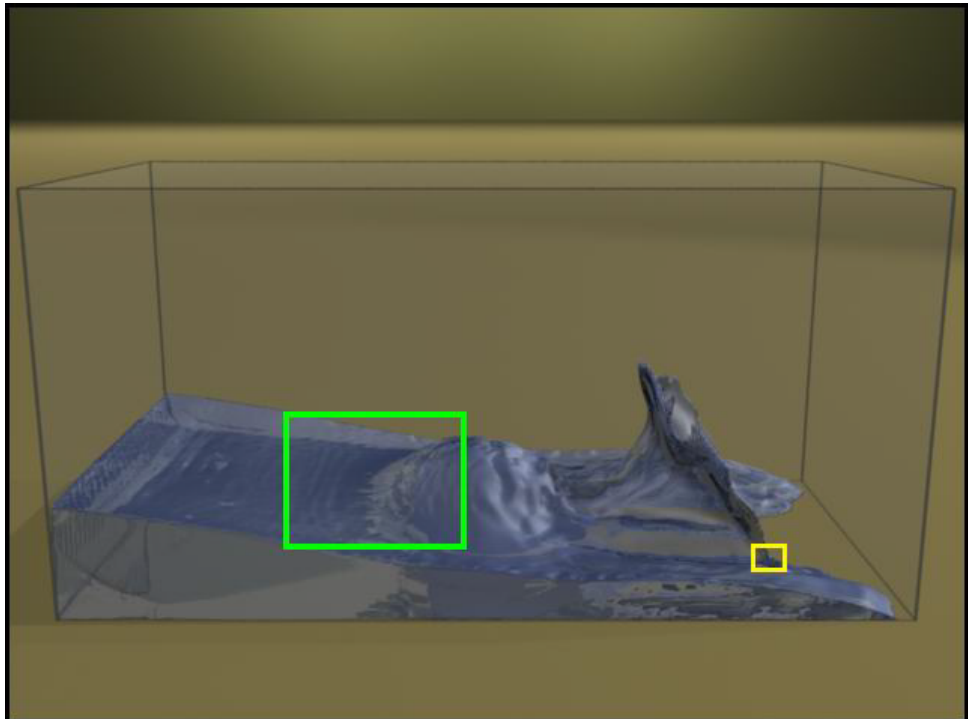
Multi-material 3D Printing

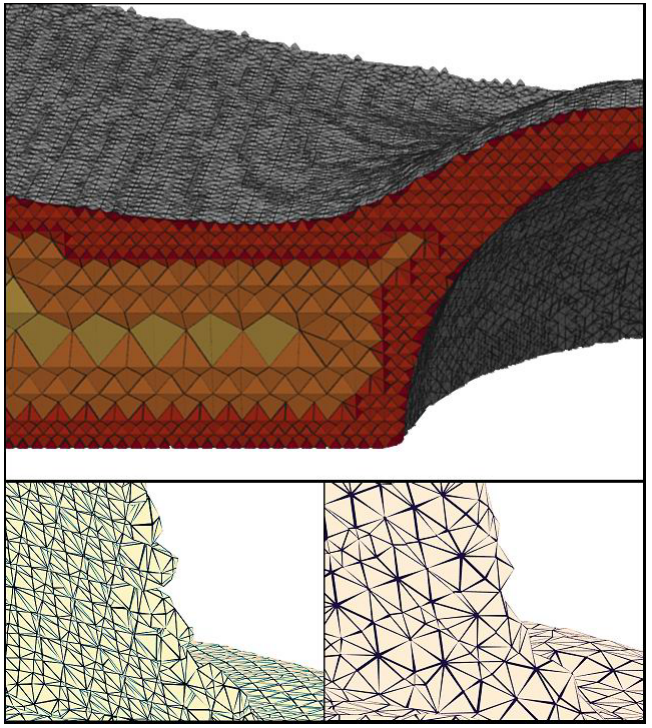


Intergraph Corporation

# Why Volumetric Representations?

- Some algorithms require solids
  - Physically-based simulation

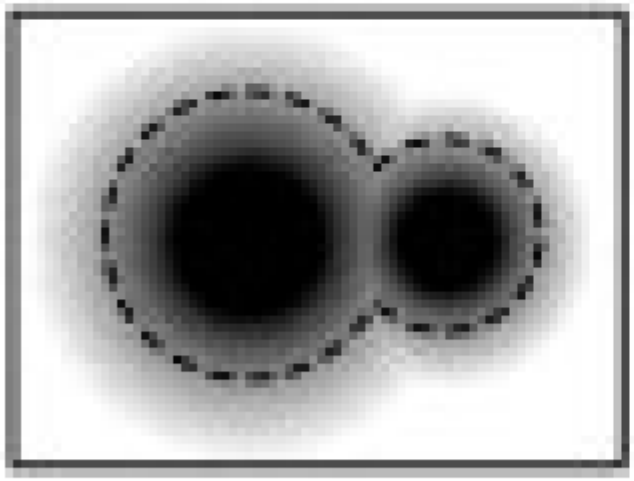




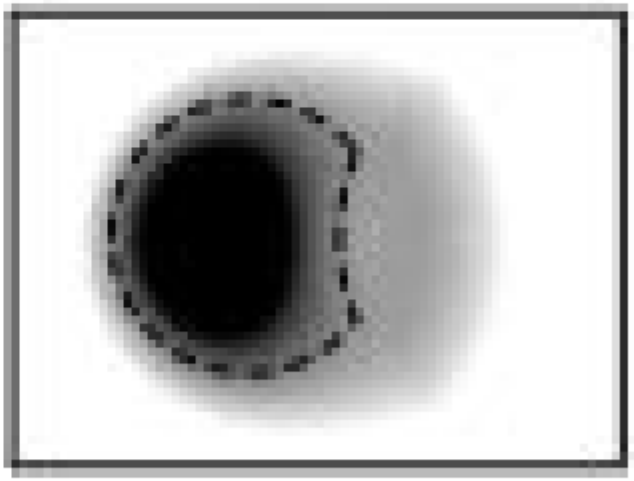
Source: Chentanez

# Why Volumetric Representations?

- Some operations are easier with solids
  - Example: union, difference, intersection



Union



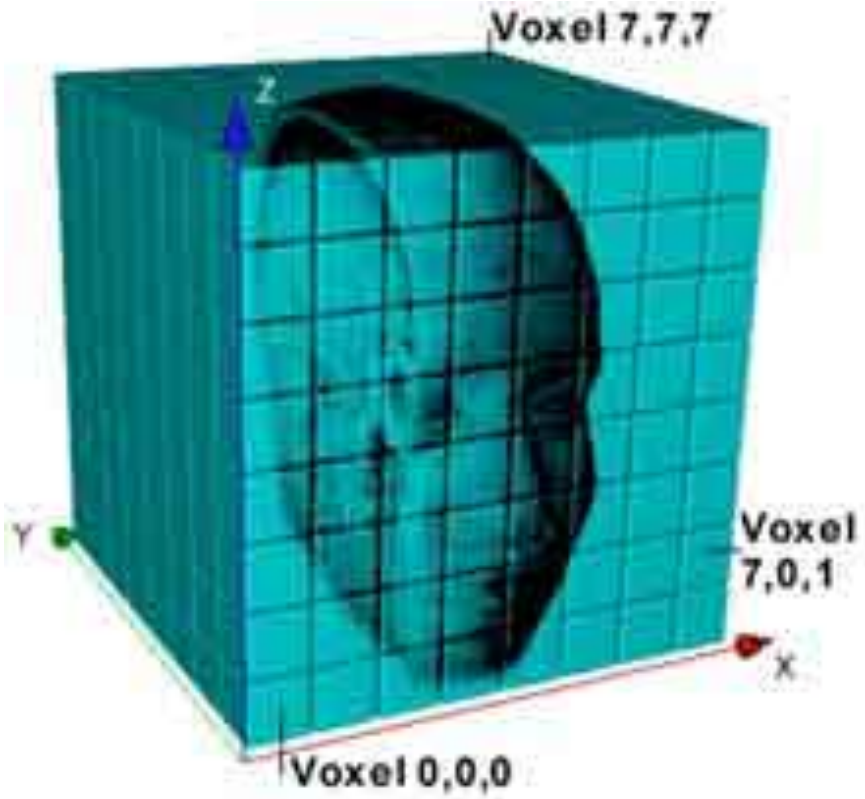
Difference

# The Plan For Today

- Discrete Volume Representations
  - Voxels
    - Voxelization (from surfaces to voxels)
    - Marching Cubes (from voxels to surfaces)
  - Octrees
  - Tetrahedra
- Implicit Representations
  - Distance Fields

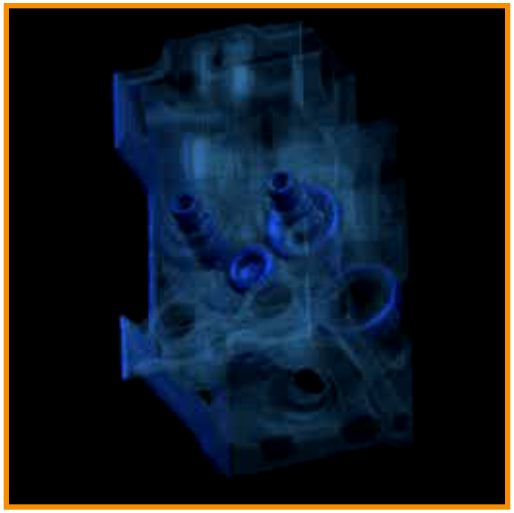
# Voxels (Volume Elements)

- Partition space into a uniform grid
  - Grid cells are called voxels (like pixels)



# Voxels

- Store properties of solid object with each voxel
  - Occupancy
  - Color
  - Density
  - Temperature
  - etc.

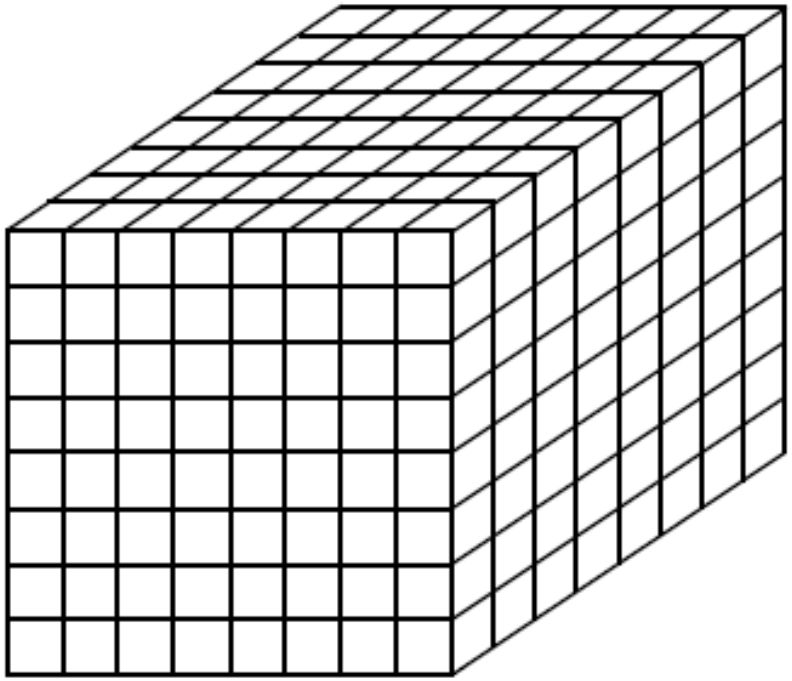


Engine Block
*Stanford University*

Visible Human
*(National Library of Medicine)*

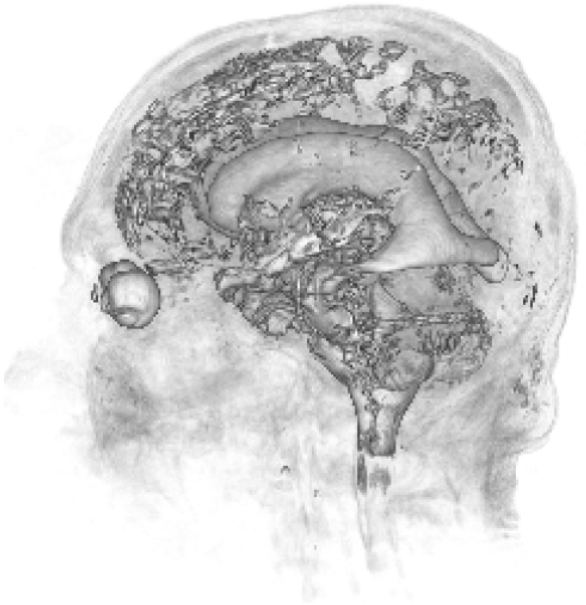
# Voxel Storage

- $O(n^3)$ storage for n x n x n grid
  - 1 billion voxels for 1000 x 1000 x 1000



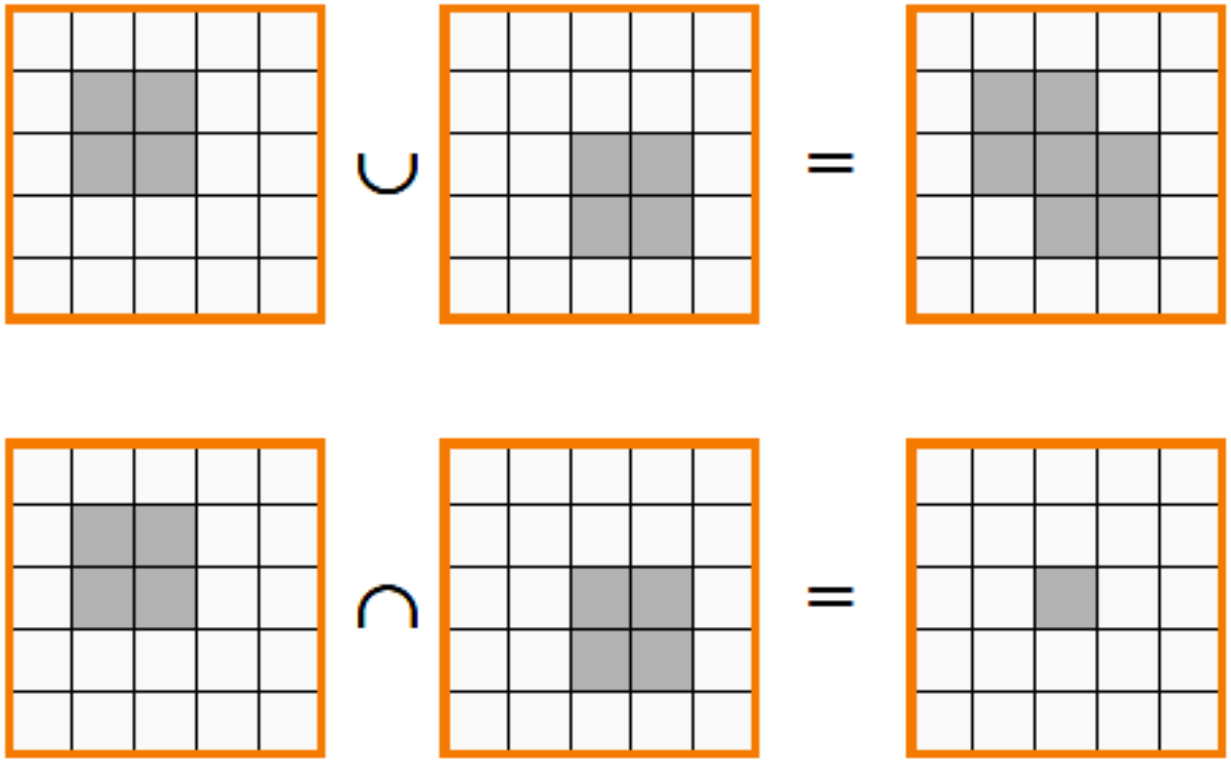
# Voxel Processing

- Signal processing (just like images)
  - Reconstruction
  - Resampling
- Typical operations
  - Blur
  - Edge detection
  - Warp
  - etc.
- Often fully analogous to image processing



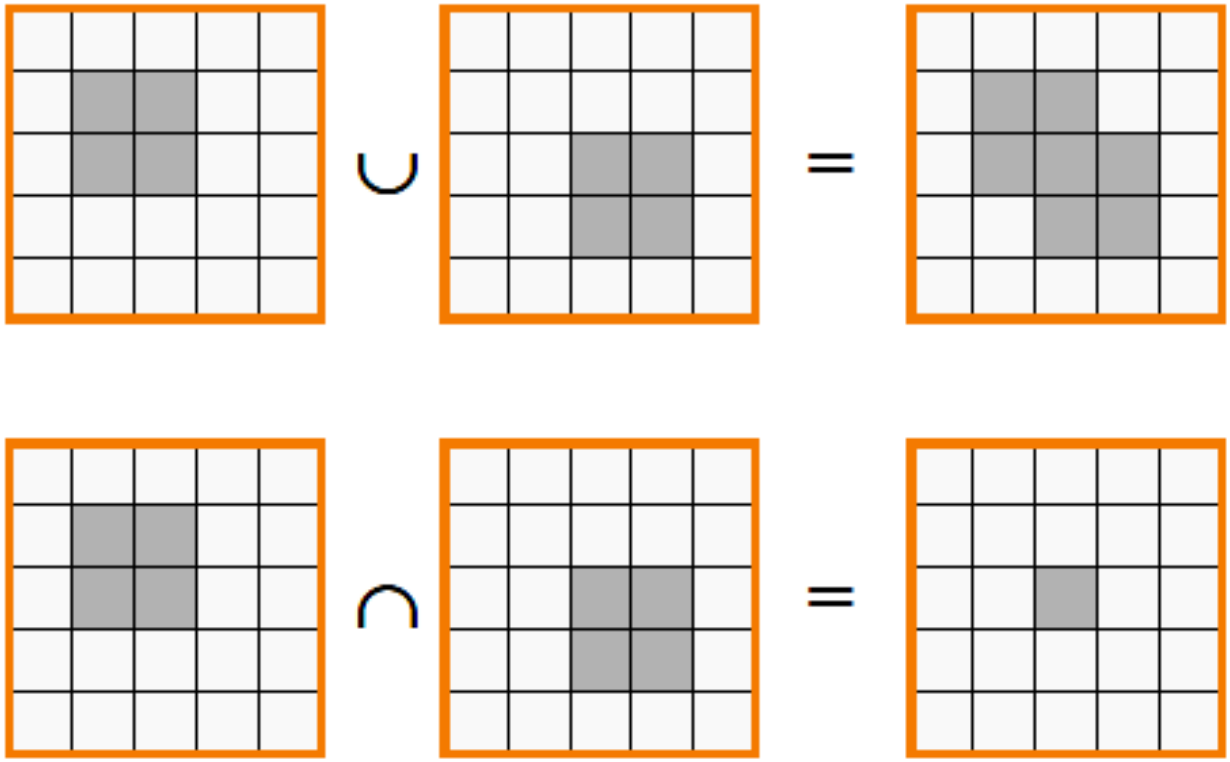
# Voxel Boolean Operations

- How?



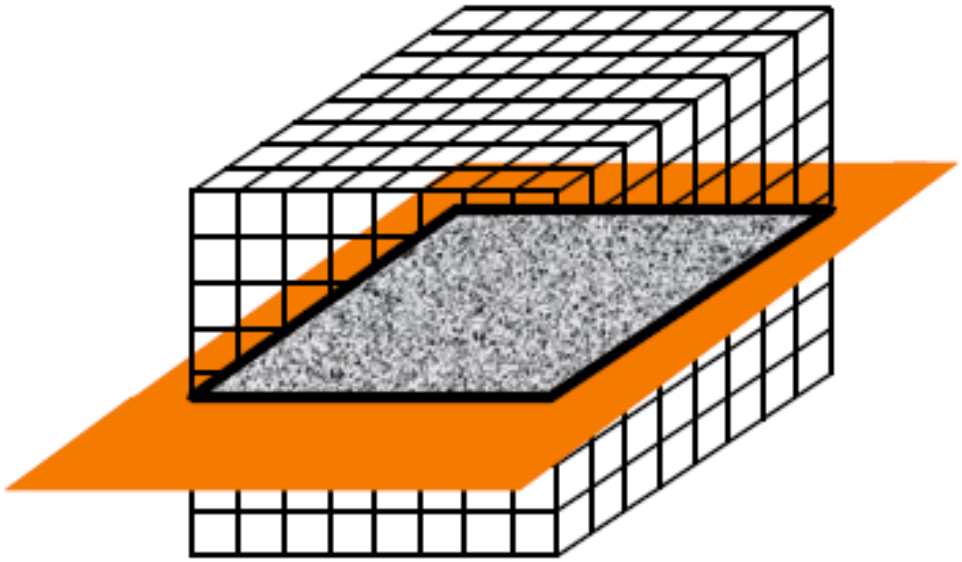
# Voxel Boolean Operations

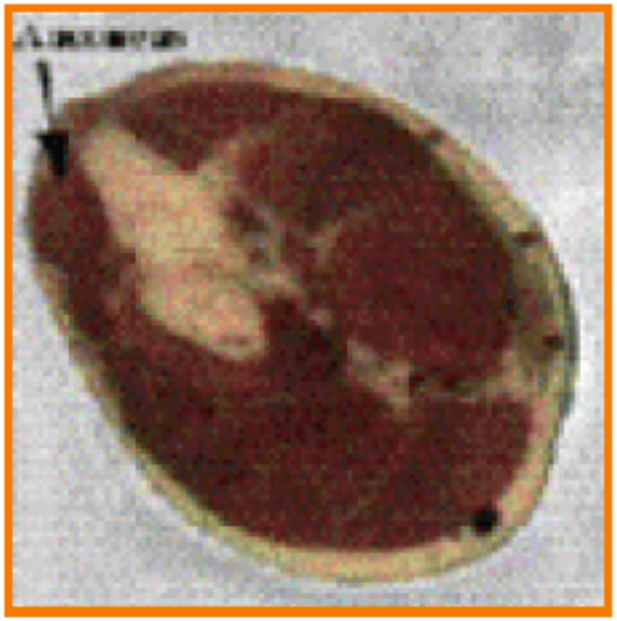
- Compare objects voxel by voxel
  - Trivial



# Voxel Display

- Slicing
  - Draw 2D image resulting from intersecting voxels with a plane

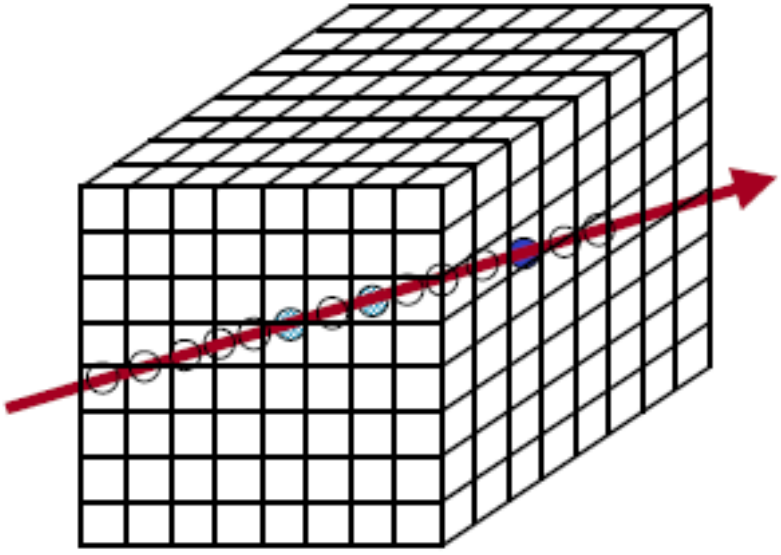


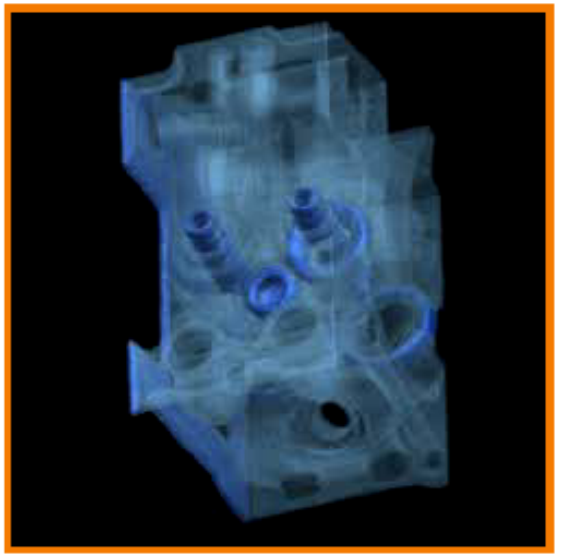


Visible Human
(National Library of Medicine)

# Voxel Display

- Ray casting
  - Integrate density along rays trough voxels

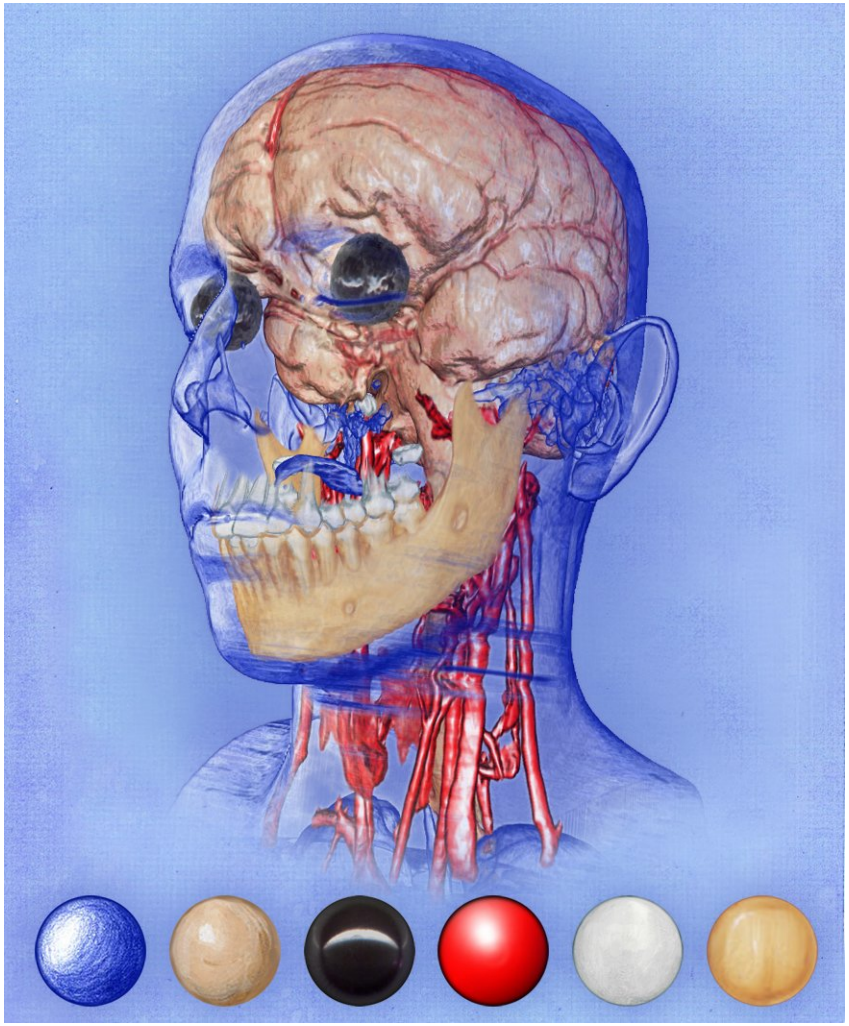




Engine Block
Stanford University

# Voxel Display

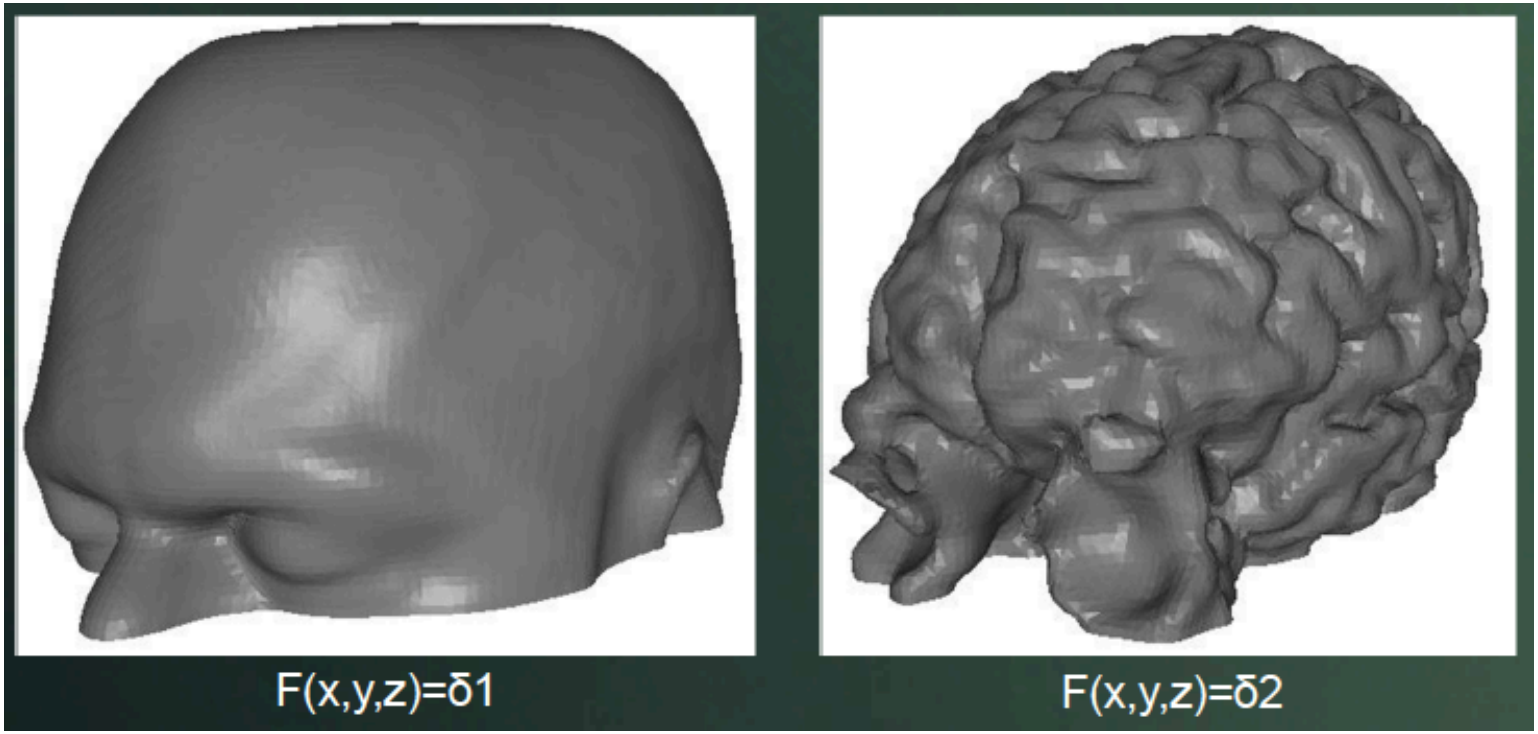
- Extended ray casting
  - Complex transfer functions
  - Map voxel densities to materials
  - Evaluate “normals” at material transitions



Bruckner et al. 2007

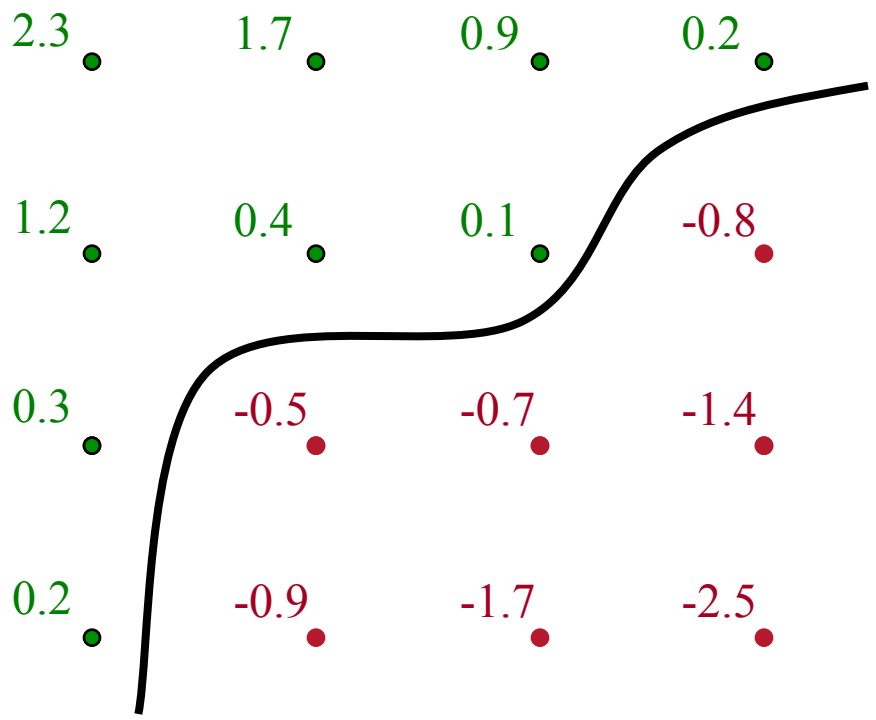
# Voxel Display

- Isosurface extraction
  - Treat the voxel grid as a regular sampling of some function F(x,y,z), and extract the isosurface satisfying F(x,y,z)=δ.



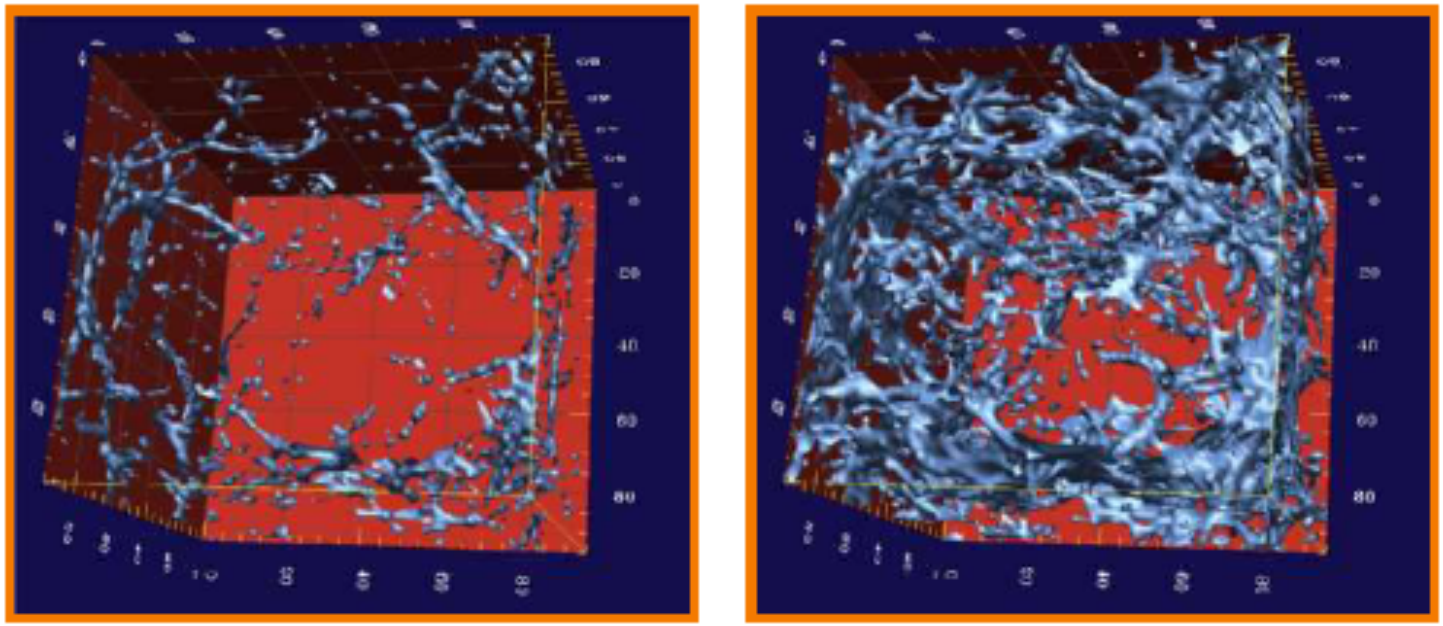
# Voxel Display

- Isosurface rendering
  - Interpolate samples stored on regular grid
  - Isosurface at f(x,y,z)=0 defines surface



# Voxel Display

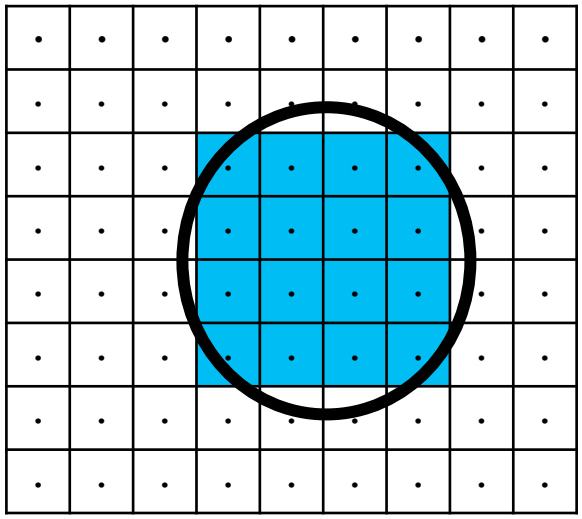
- Isosurface rendering
  - Render surfaces bounding volumetric regions of constant value (e.g., density)



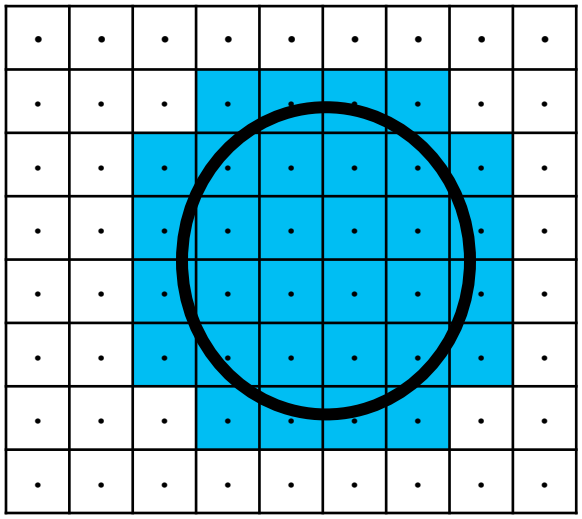
Isosurface Visualization
Princeton University

# Voxelization: From Surfaces to Voxels

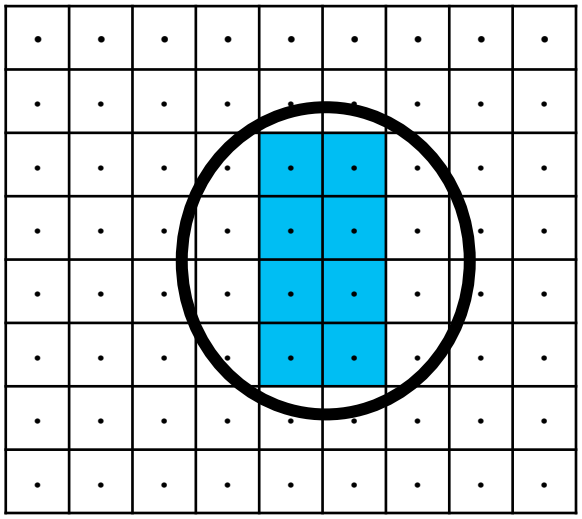
- Binary classification
  - 1: inside the volume, 0: outside



Center Inside



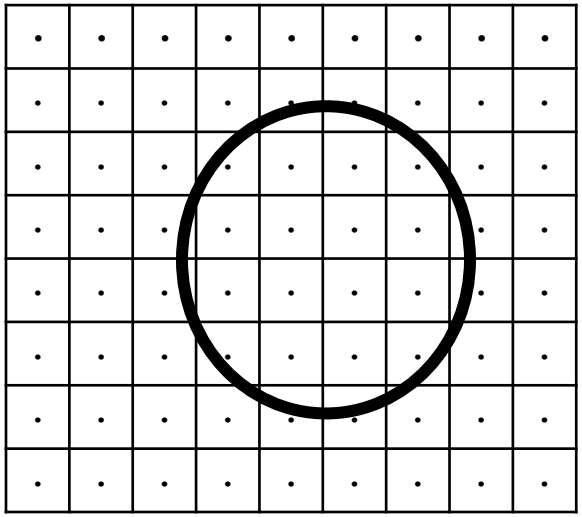
Partially Inside



Completely Inside

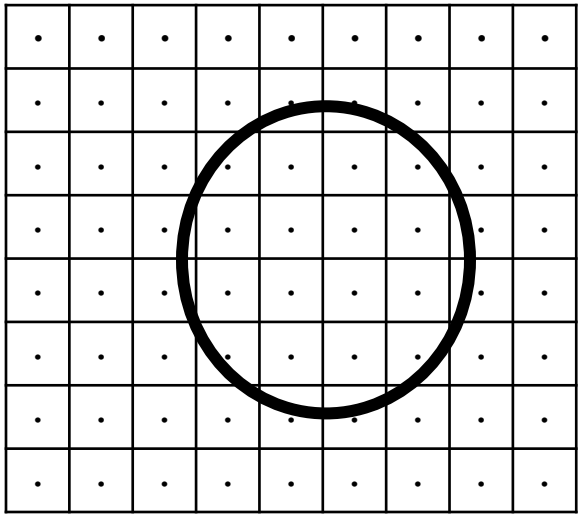
# Voxelization: From Surfaces to Voxels

- How?



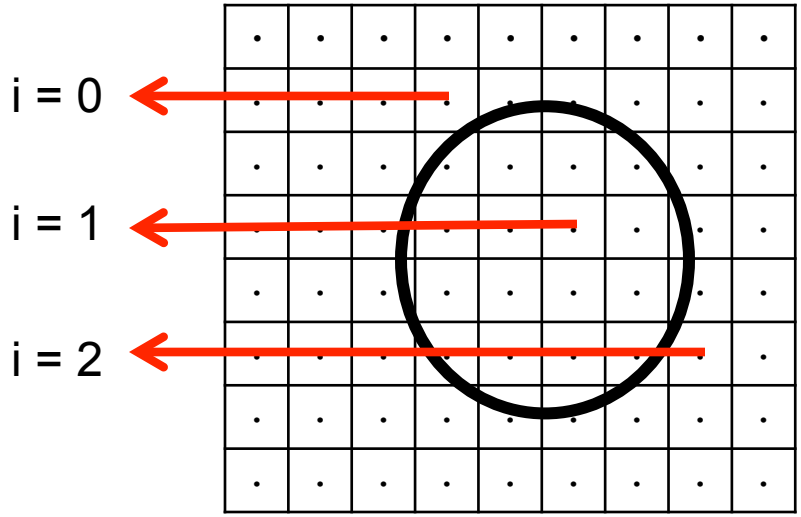
# Voxelization: From Surfaces to Voxels

- Ray casting



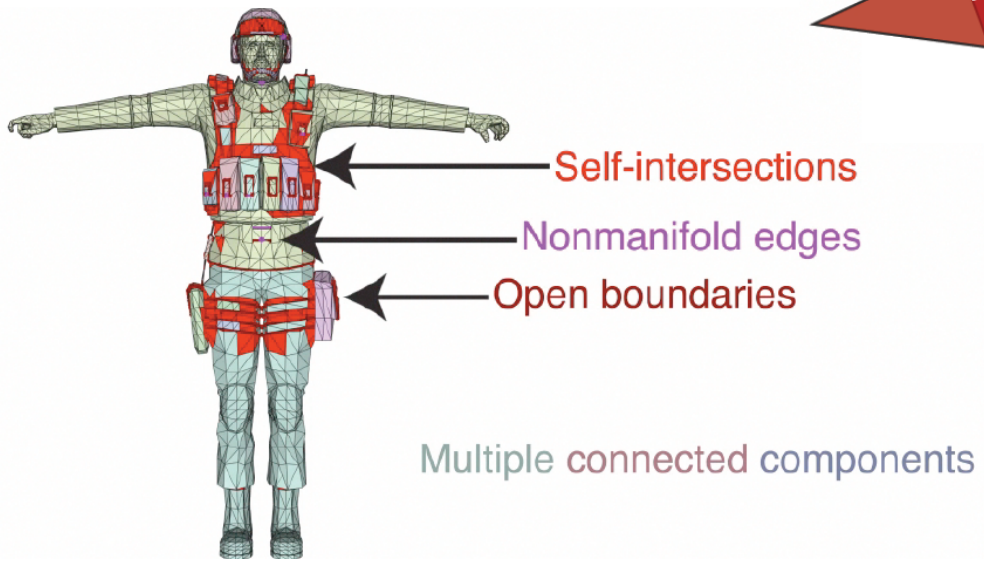
# Voxelization: From Surfaces to Voxels

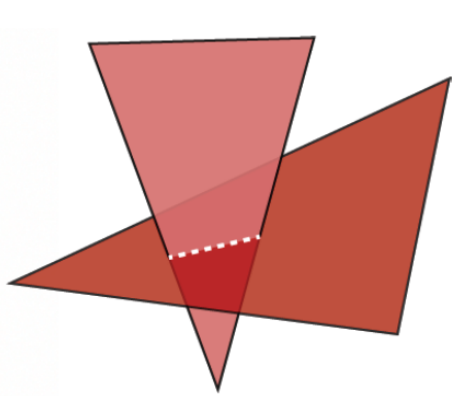
- Ray casting
  - Trace a ray from each voxel center
  - Count intersections
    - Odd: inside
    - Even: outside

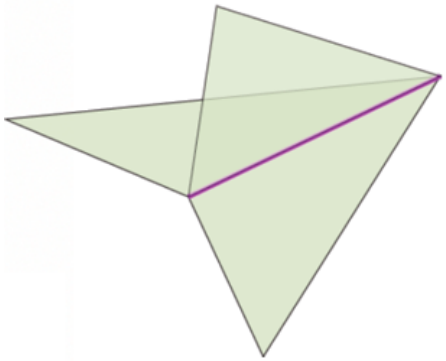


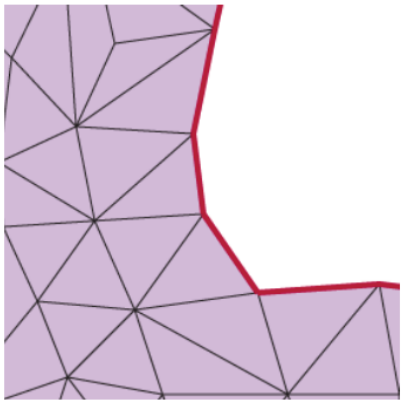
# Voxelization

- Imperfect input meshes
  - Voxelization fails



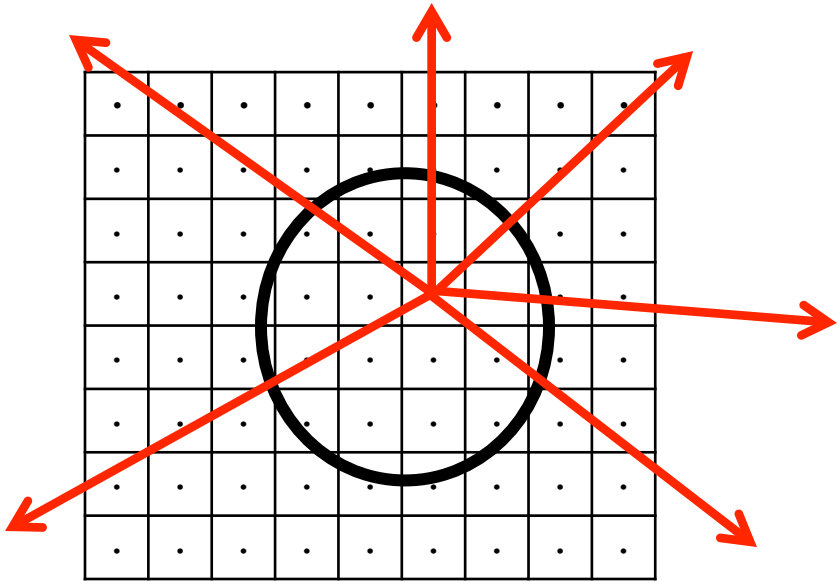




Source: Jacobson, 2013

# Robust Voxelization

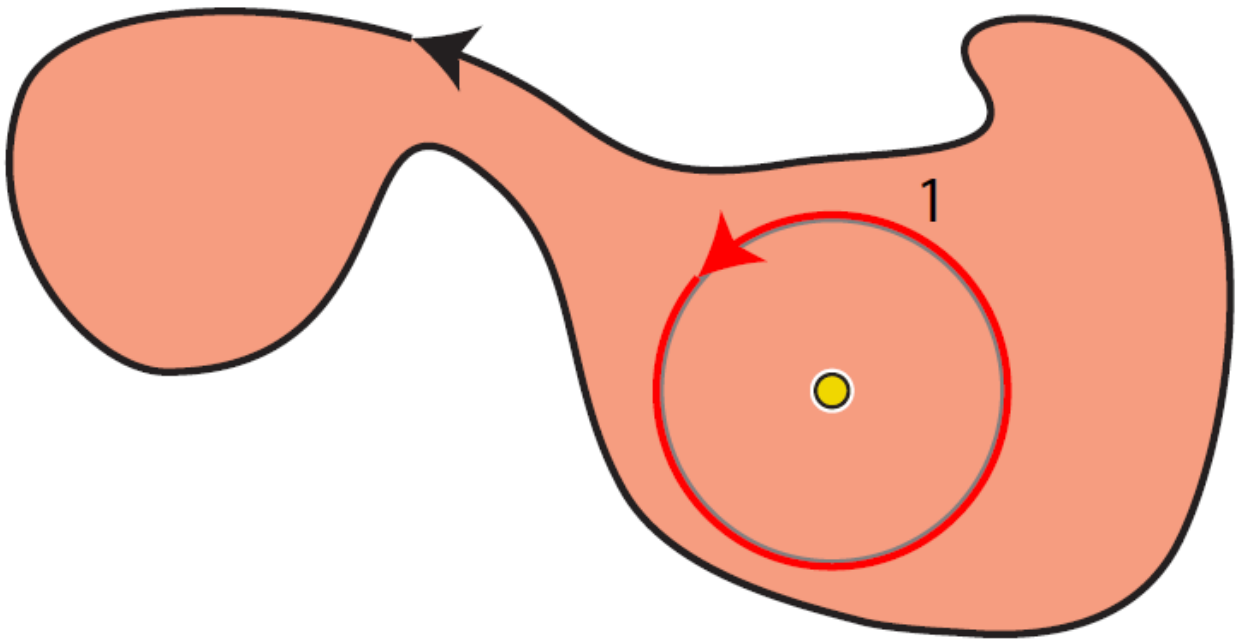
- Ray casting
  - Trace many rays in different directions
  - Combine results



# Robust Voxelization using Winding Numbers

- If shape is watertight, winding number is perfect measure of inside/outside
- # of full revolutions the observer took (signed #)

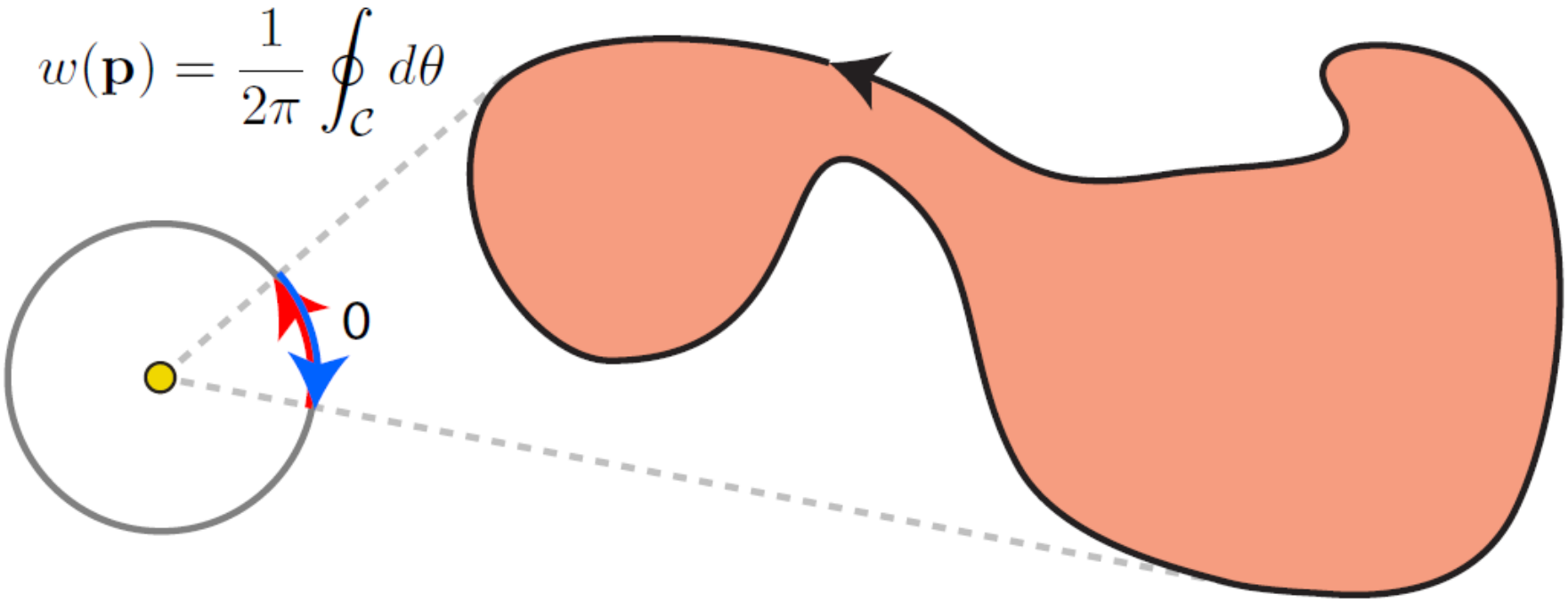
$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_{\mathcal{C}} d\theta$$




Source: Jacobson, 2013. SIGGRAPH 2013. Robust Inside-Outside Segmentation using Generalized Winding Numbers

# Robust Voxelization using Winding Numbers

- If shape is watertight, winding number is perfect measure of inside/outside

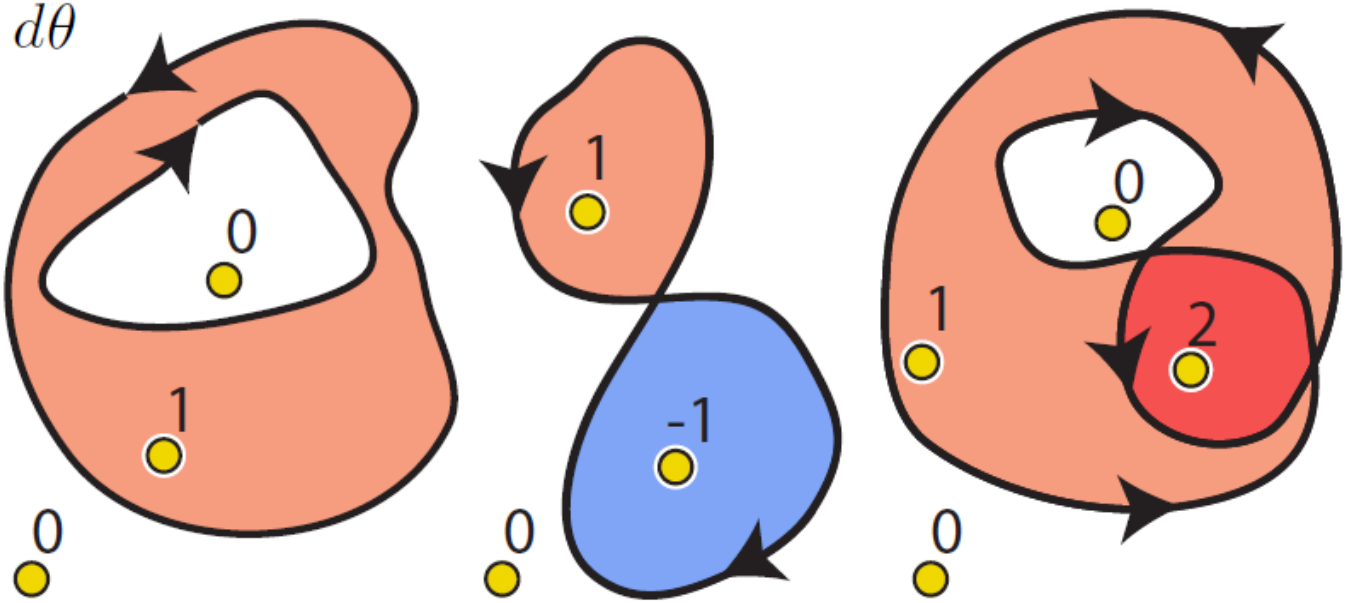



Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- Winding number uses orientation to treat inside/ outside as a signed integer

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



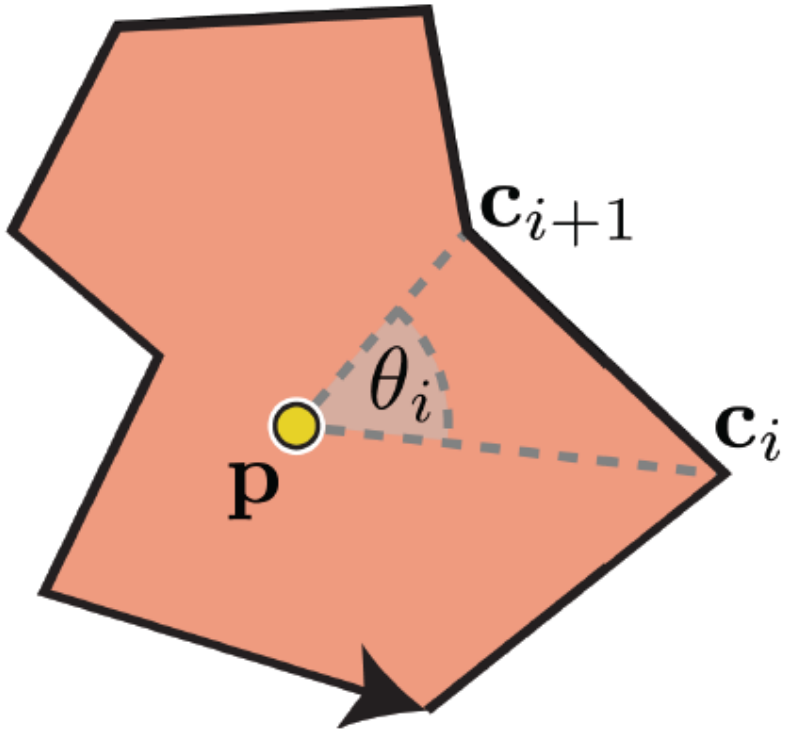

Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- Naïve discretization is simple and exact

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$

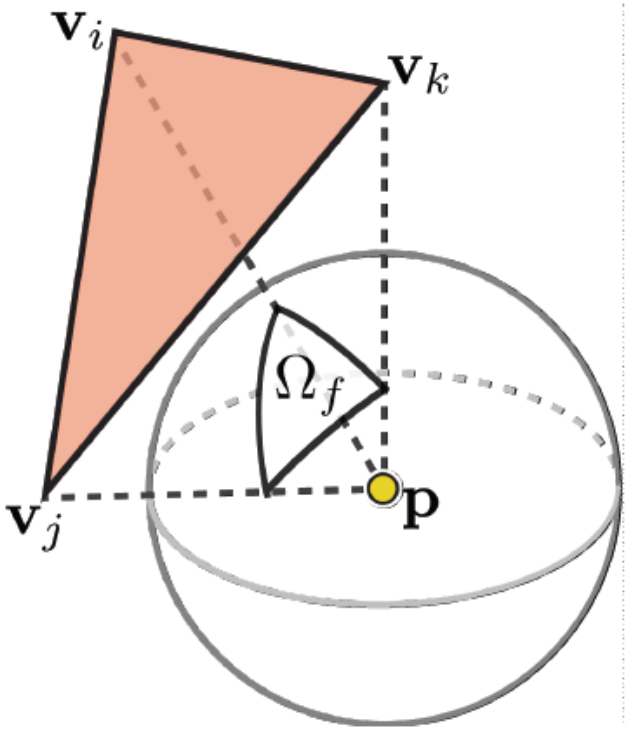
$$w(\mathbf{p}) = \frac{1}{2\pi} \sum_{i=1}^{n} \theta_i$$




Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- Generalizes elegantly to 3D via solid angle




$$w(\mathbf{p}) = \frac{1}{4\pi} \iint_{\mathcal{S}} \sin(\phi) d\theta d\phi$$

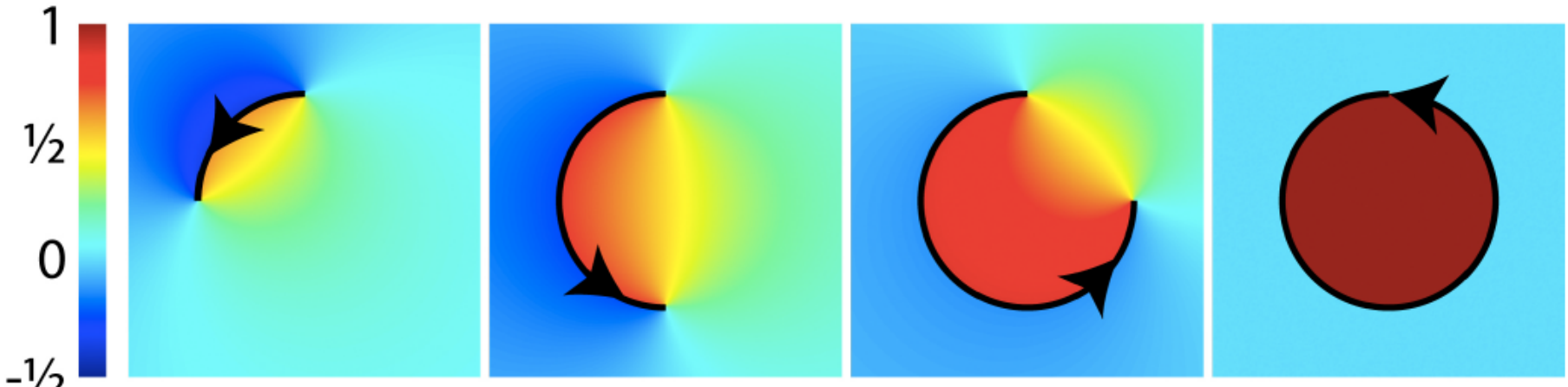
$$w(\mathbf{p}) = \frac{1}{4\pi} \sum_{f=1}^{m} \Omega_f$$

Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- What happens if the shape is open?

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$

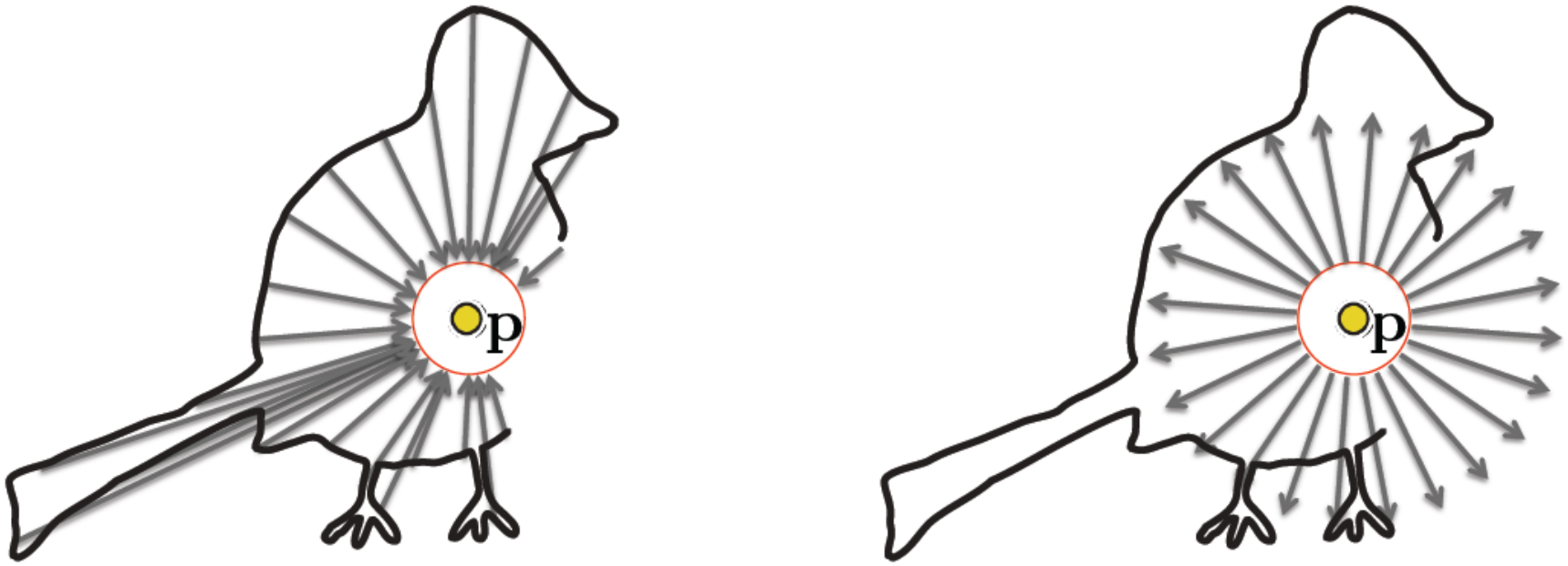



Gracefully tends toward perfect indicator as shape tends towards watertight

Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

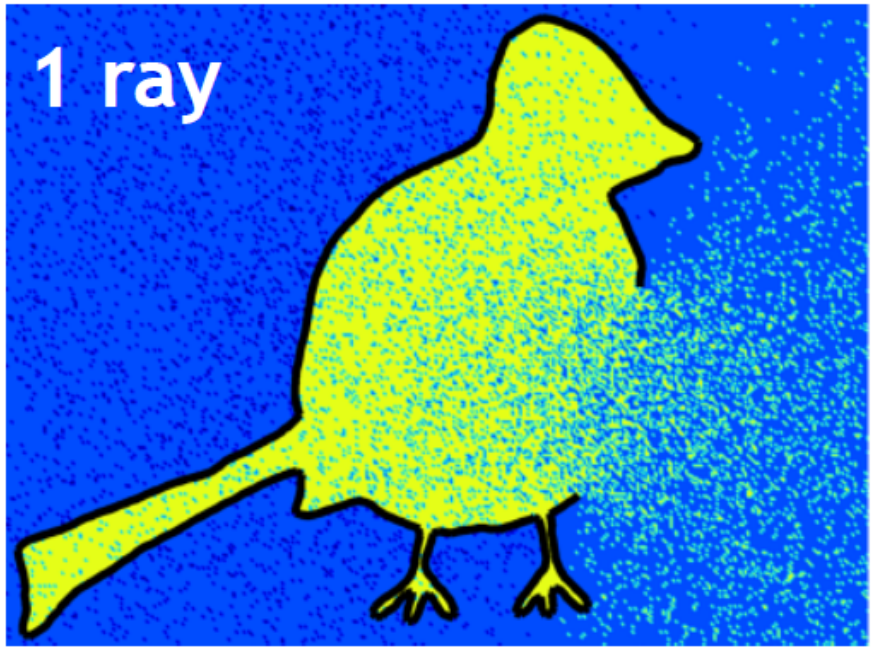
- Direct vs. sampling computation




Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

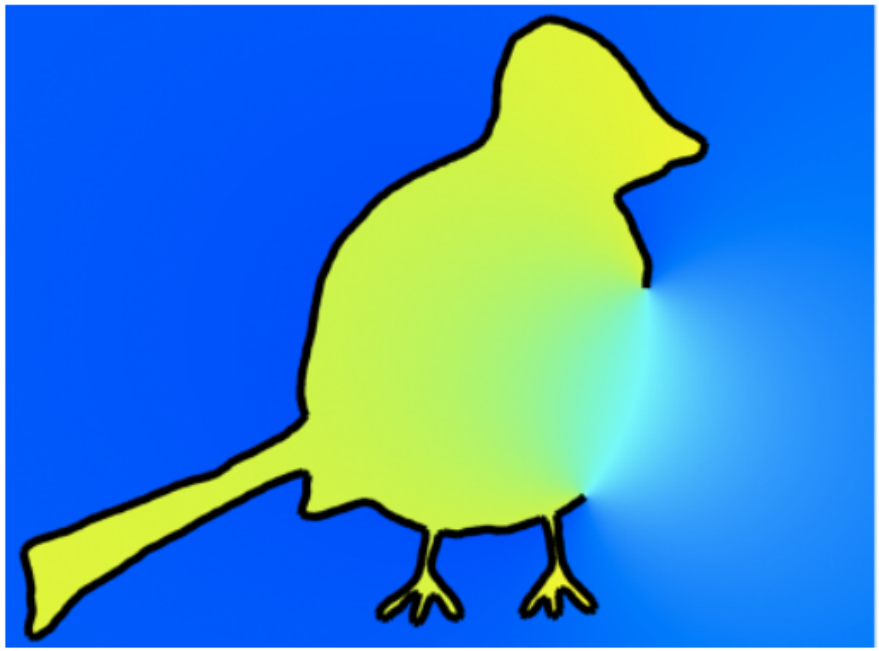
- Direct vs. sampling computation

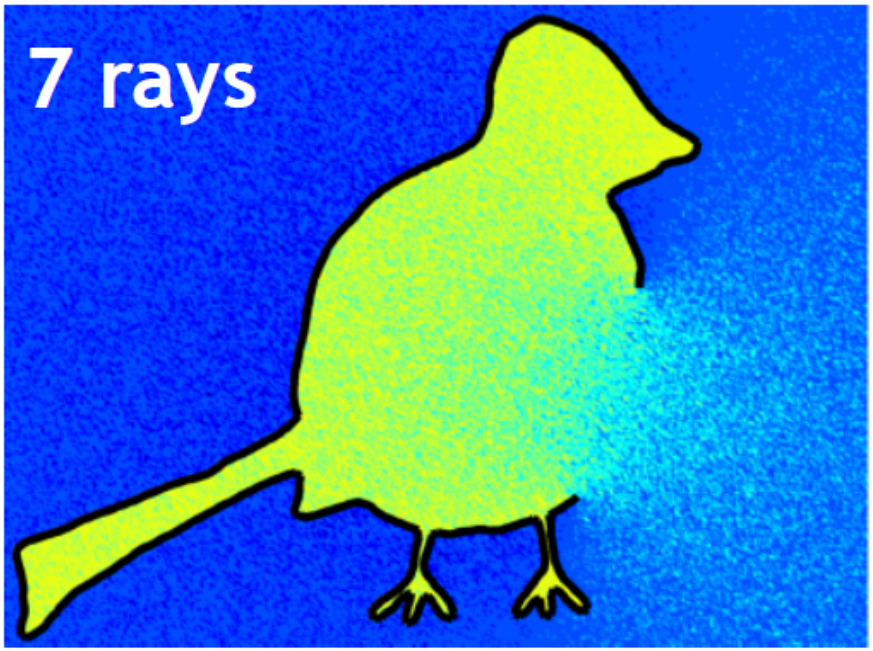



Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

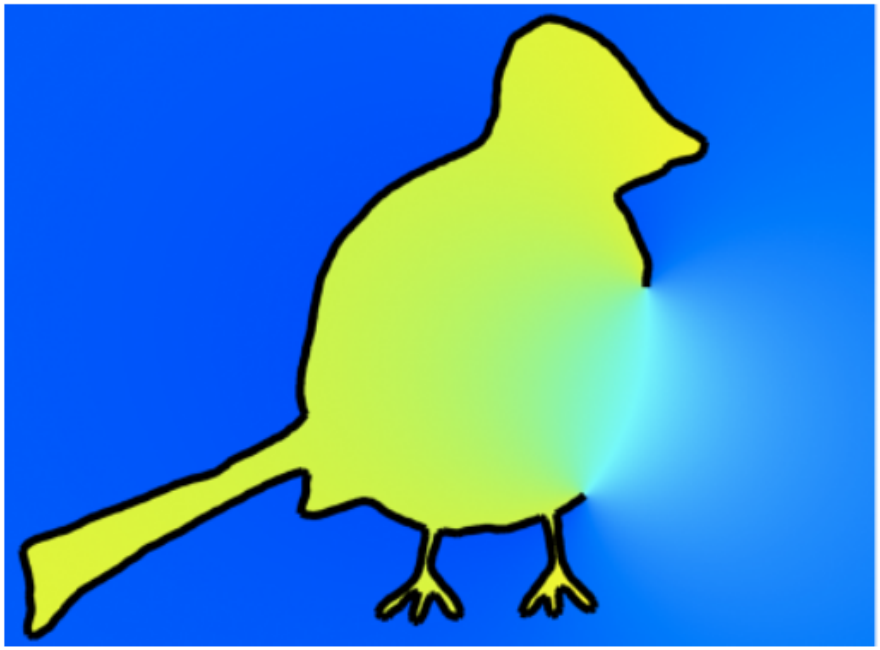


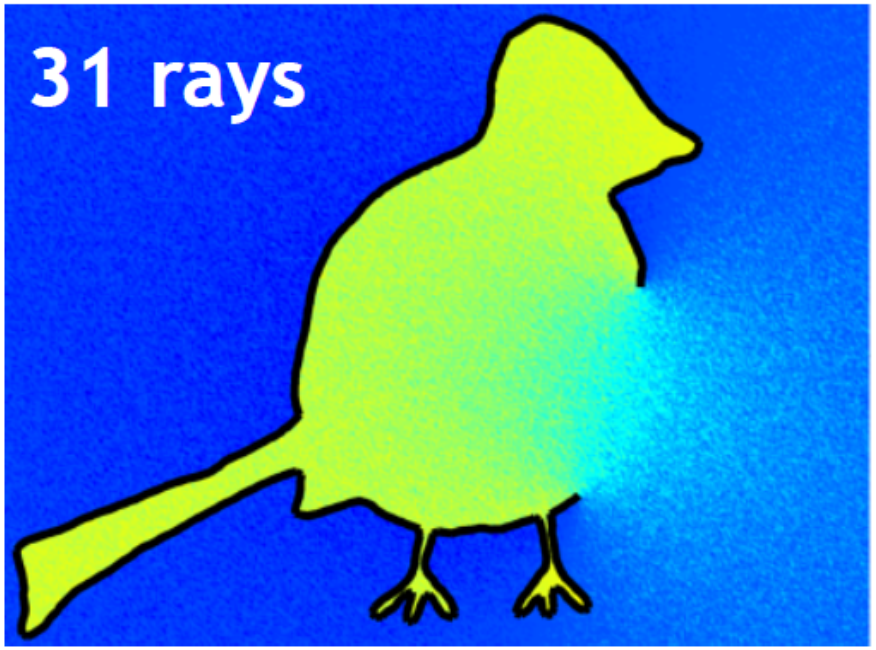



Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

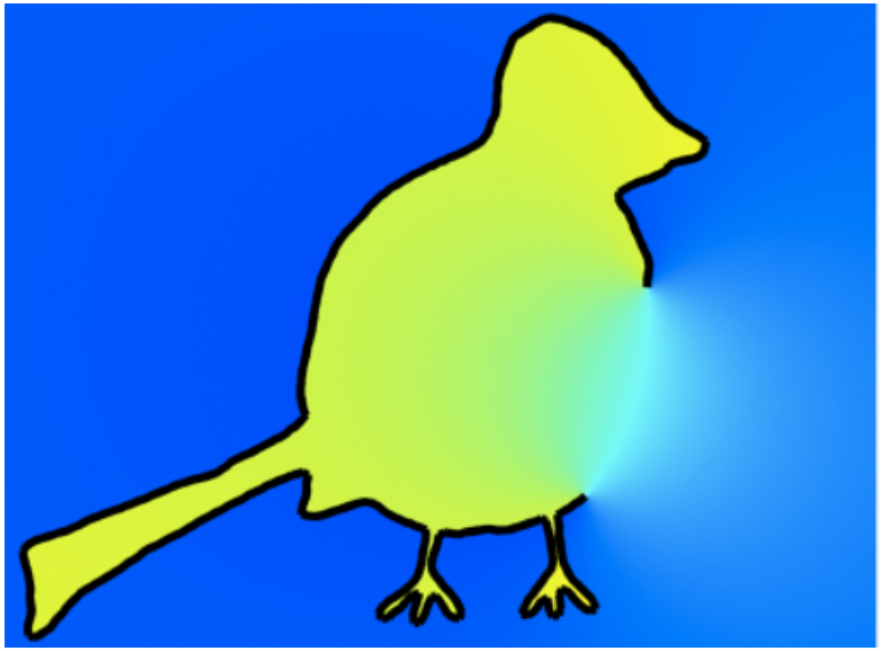


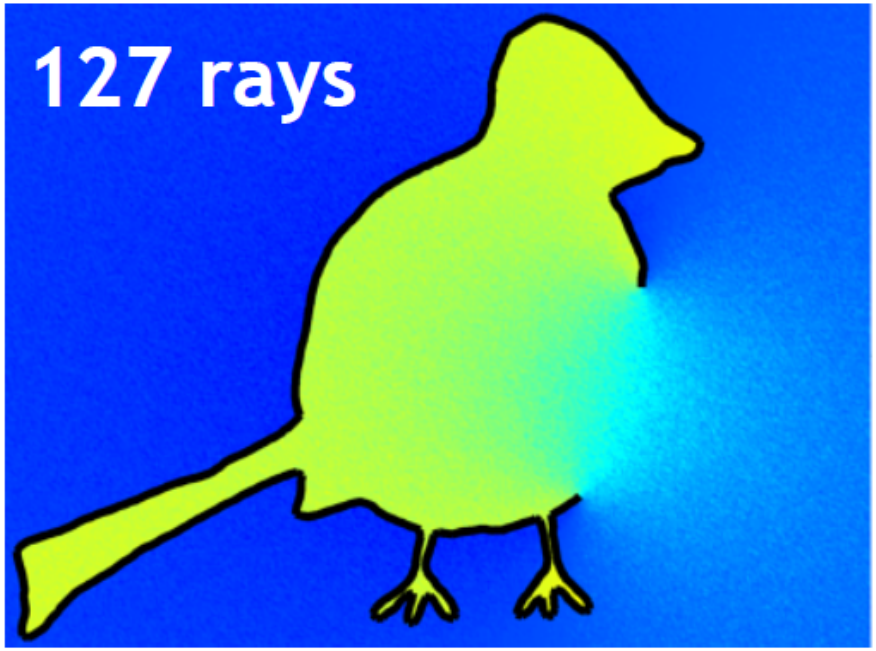



Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

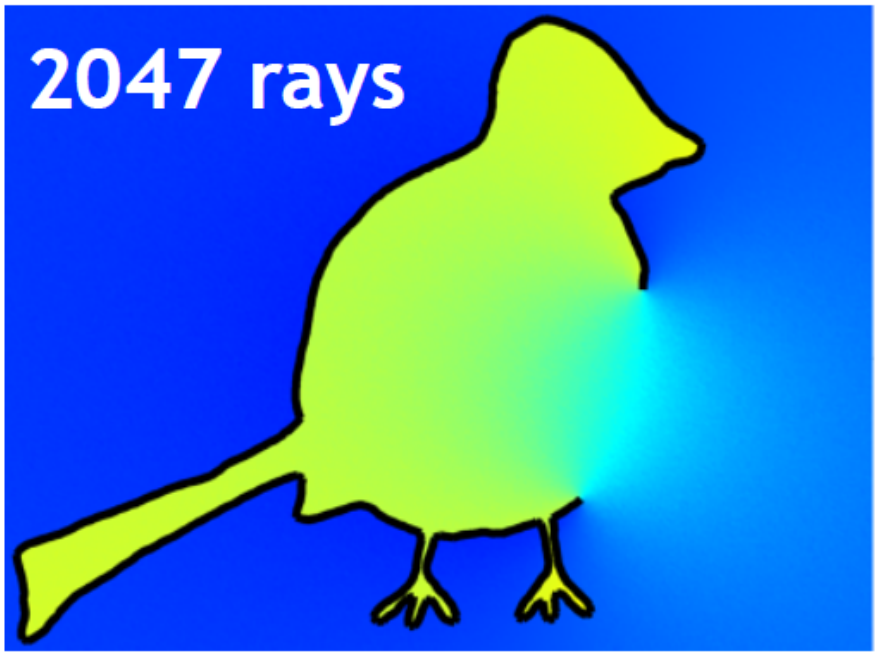





Source: Jacobson, 2013

# Robust Voxelization using Winding Numbers

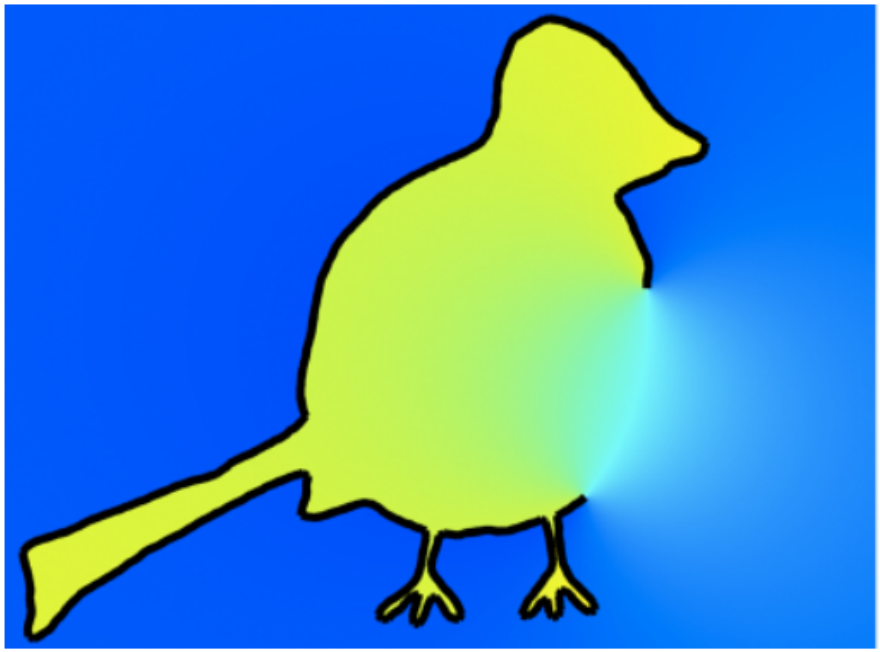
- Direct vs. sampling computation

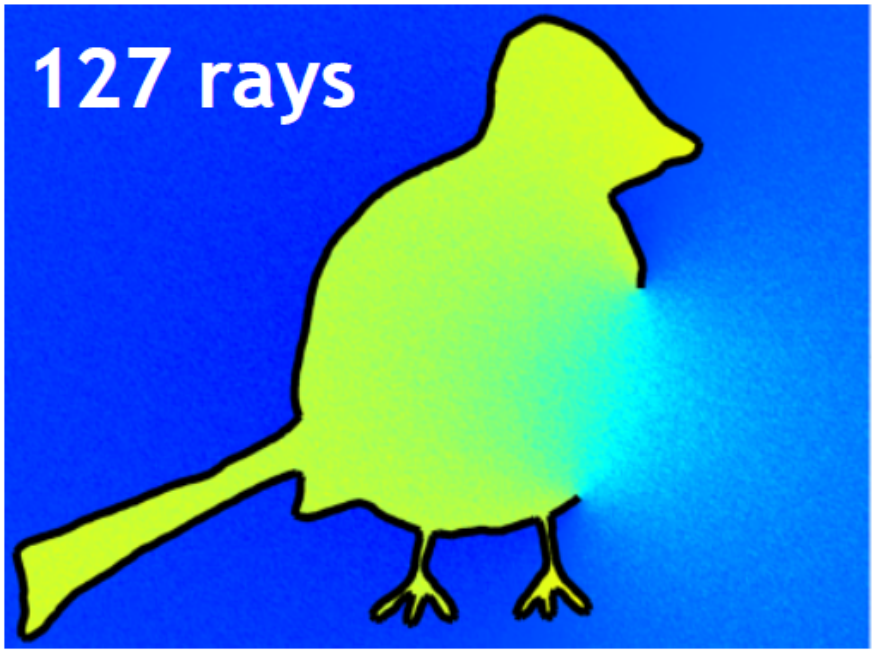



Source: Jacobson, 2013

# Robust Voxelization

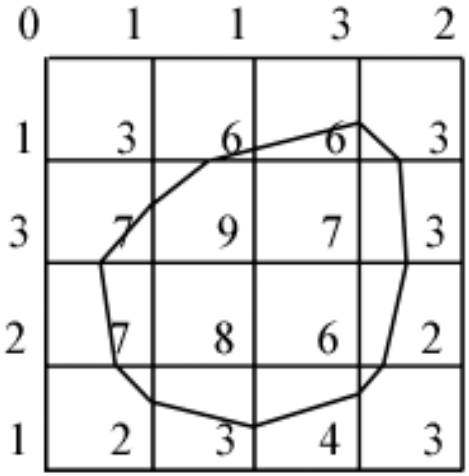
- Winding numbers
  - For each voxel
    - Computer its winding number by traversing over all triangles
- Ray casting
  - For each voxel
    - Trace many rays in different directions
    - Combine results





# From Voxels Grids to Surfaces

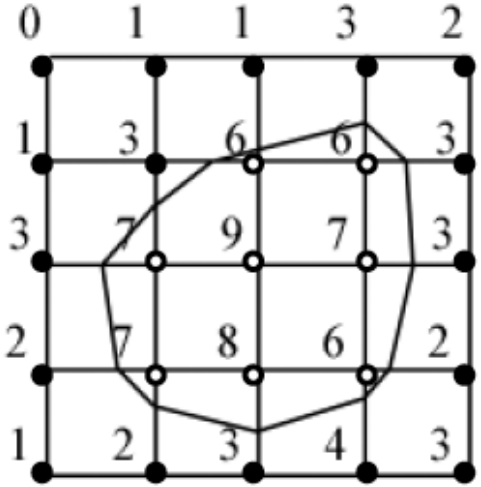
- Marching squares (2D)
  - A simple example: extract isosurface equal to 5




http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

# From Voxels Grids to Surfaces

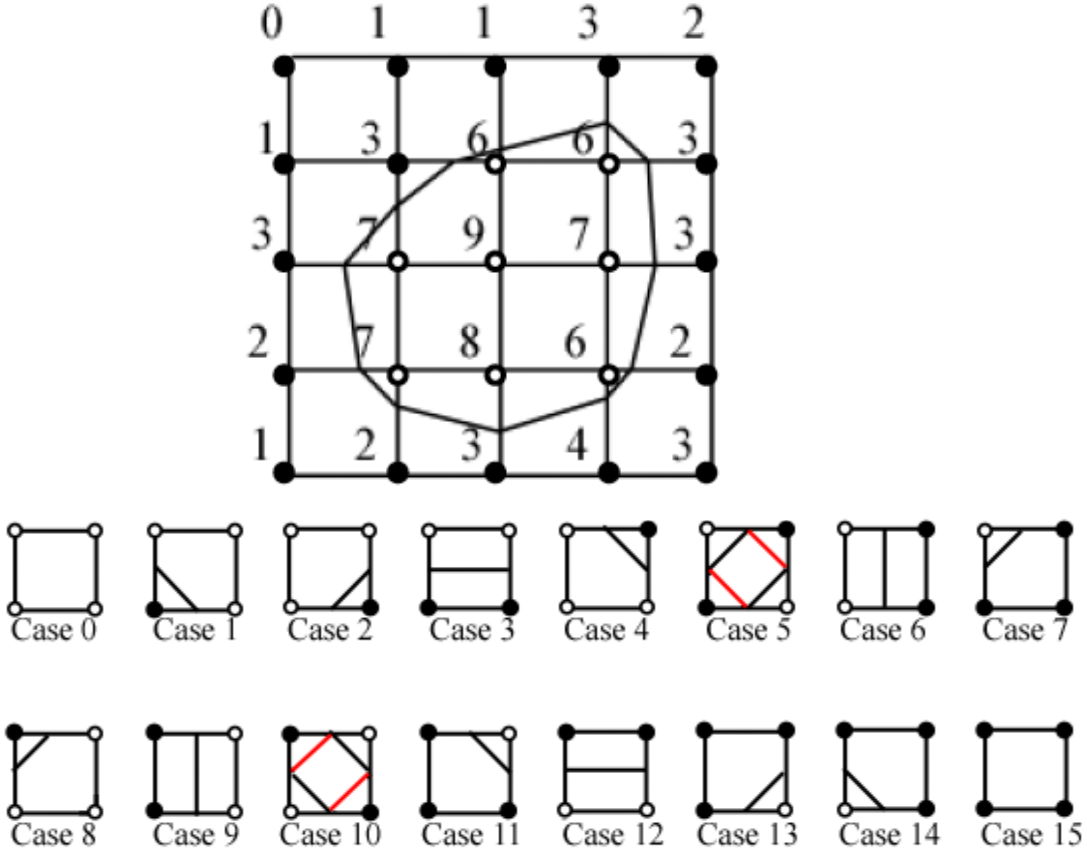
- Marching squares (2D)
  - A simple example: extract isosurface equal to 5




http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

# From Voxels Grids to Surfaces

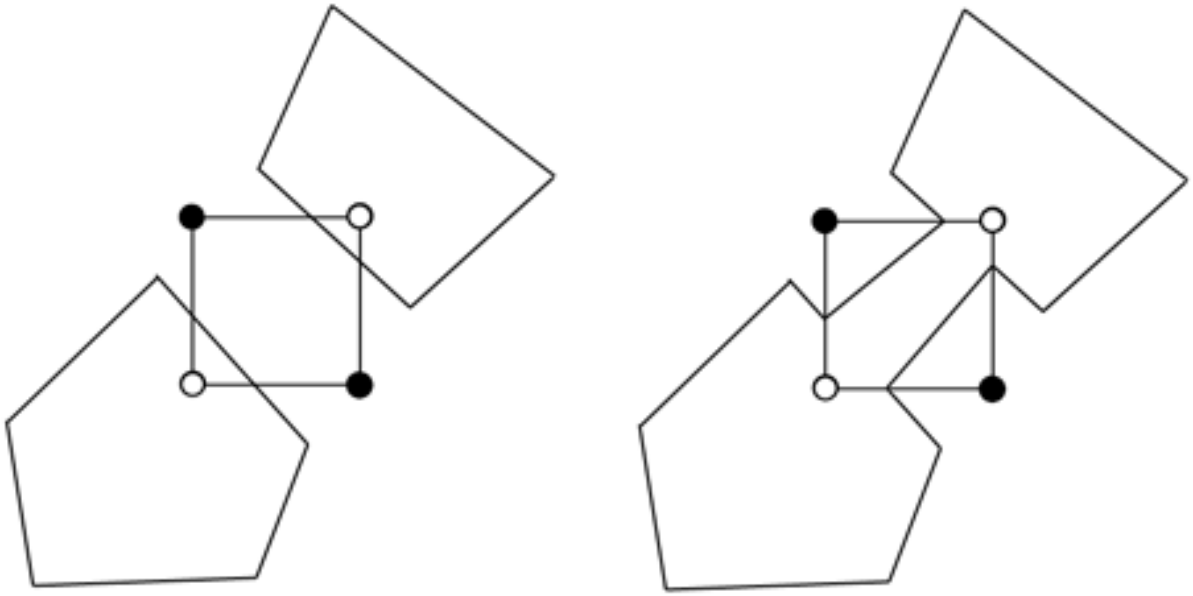
- Marching squares (2D)
  - A simple example: extract isosurface equal to 5




http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

# From Voxels Grids to Surfaces

- Marching squares (2D)
  - Ambiguous cases



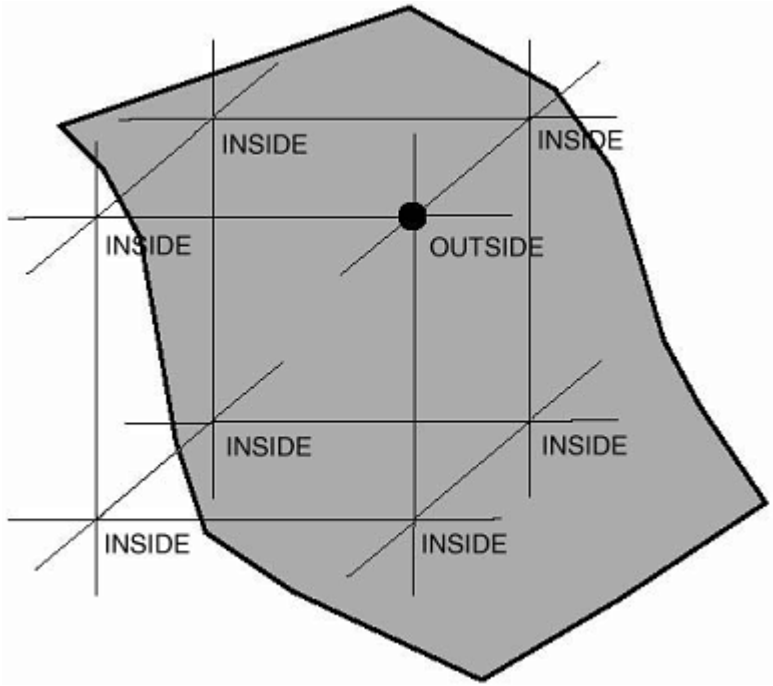
Break contour

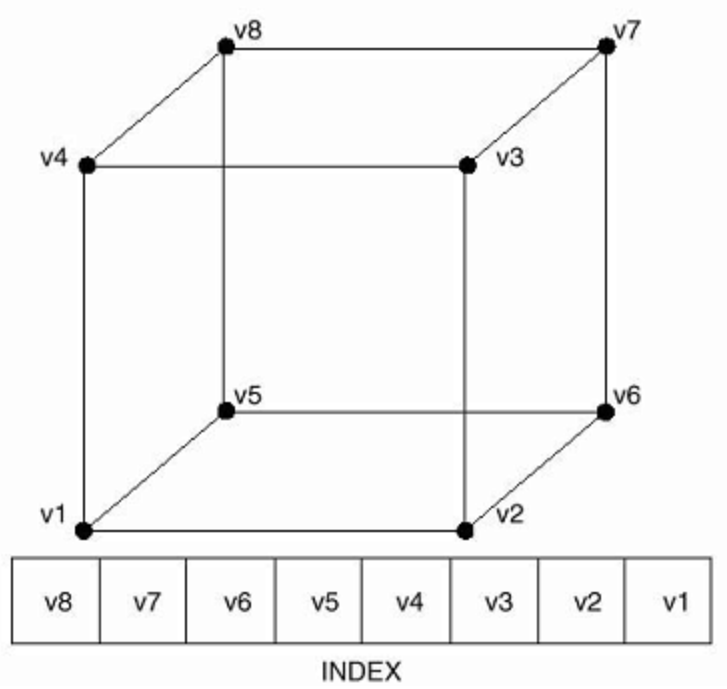
Join contour

http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

# From Voxels Grids to Surfaces

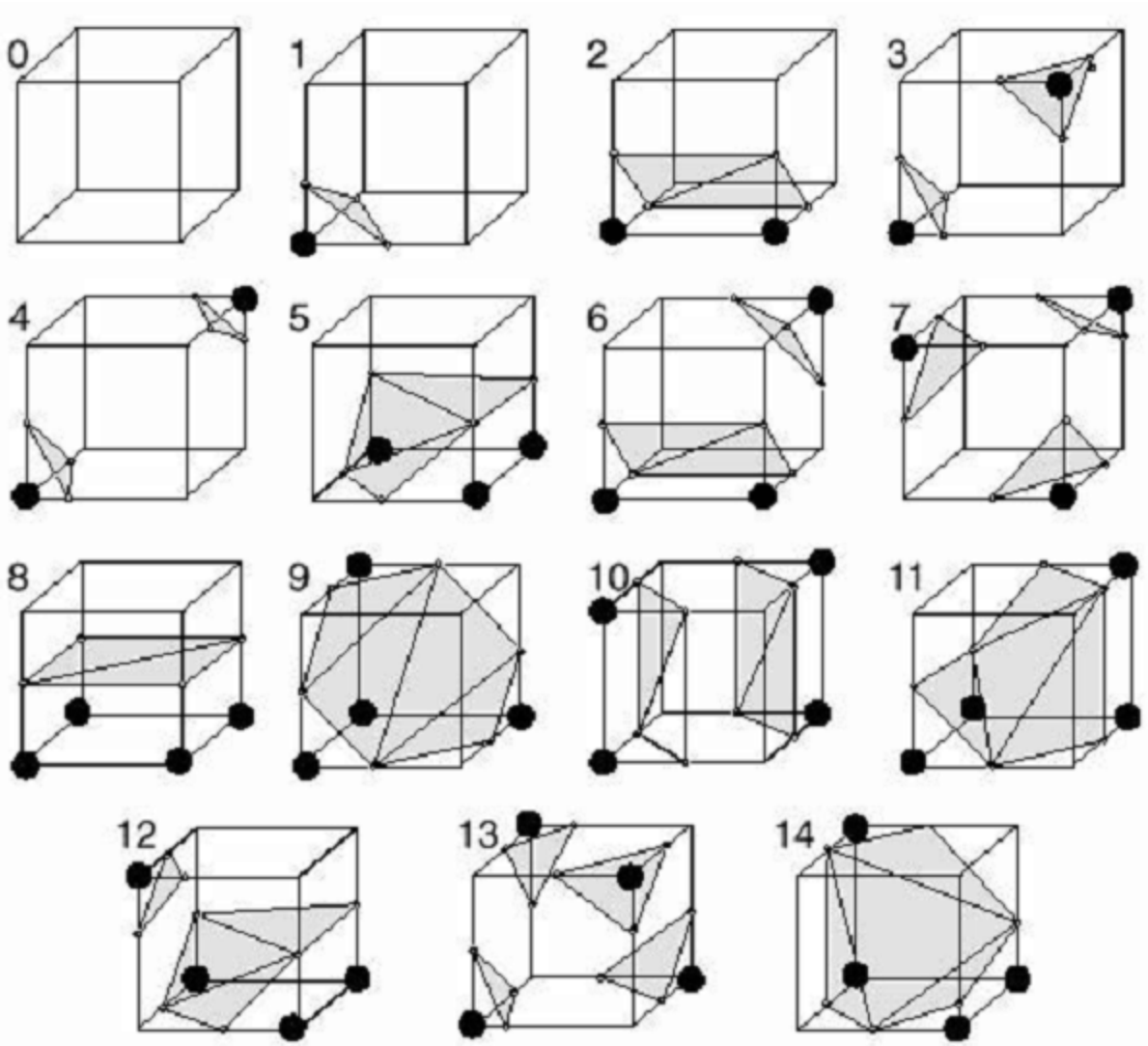
- Marching cubes (3D)
  - 256 ($2^8$) different situations
  - Generalized to 15 cases by rotations and symmetry



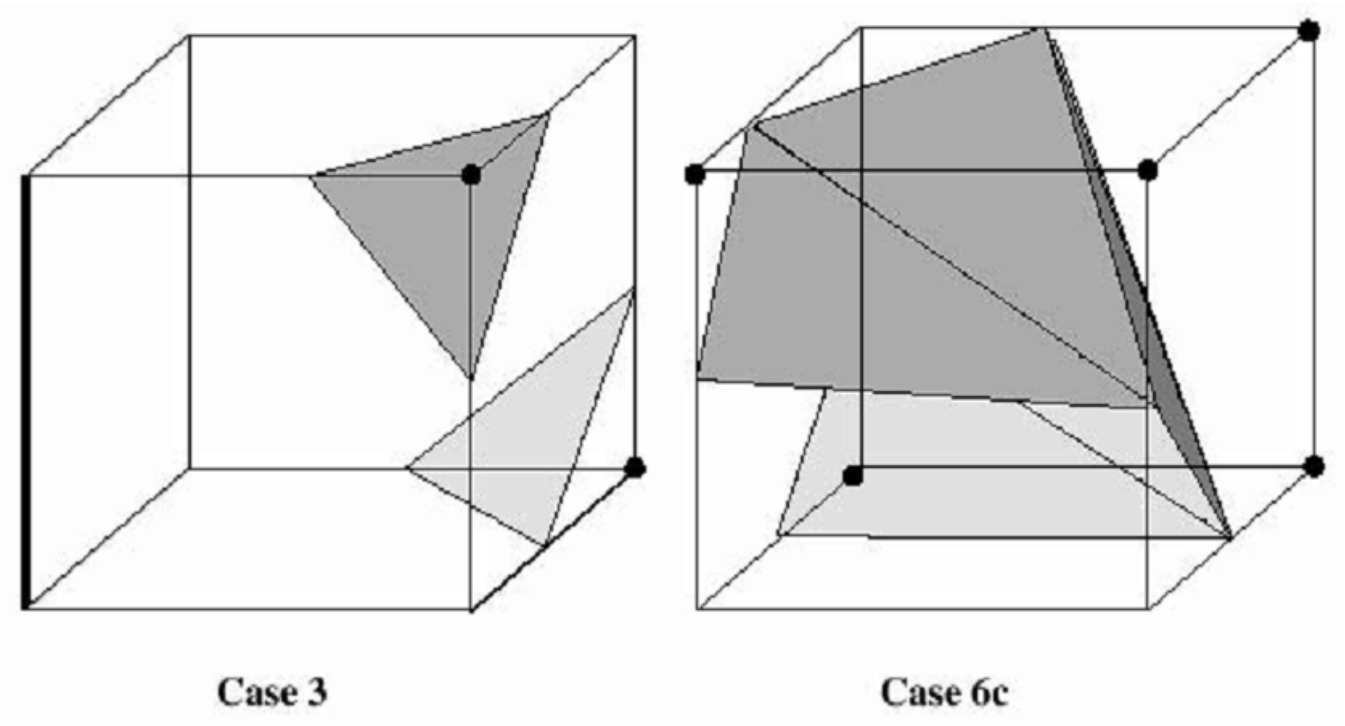

http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

# Marching Cubes

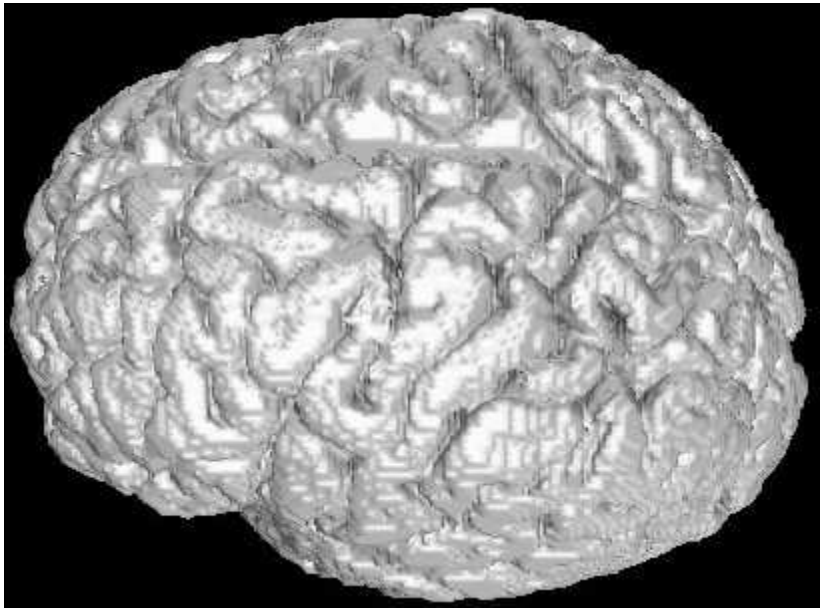


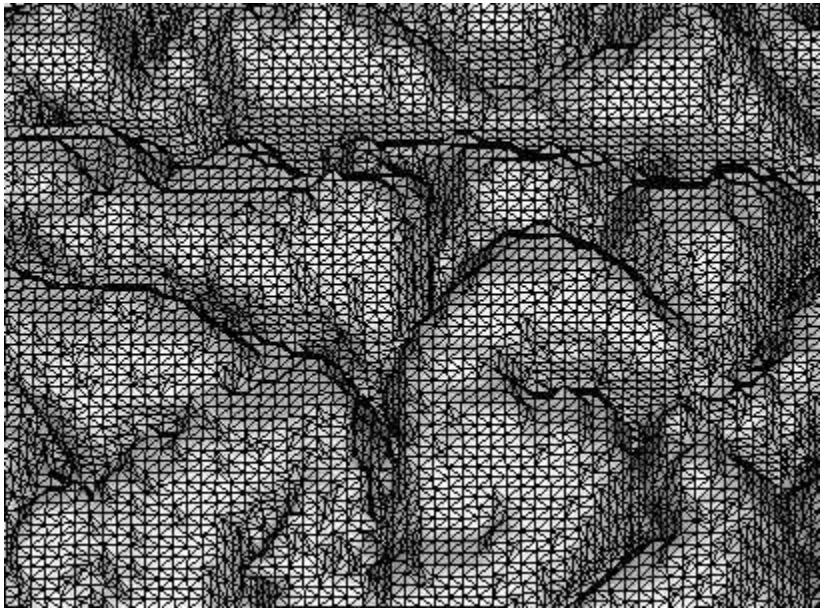
# Marching Cubes

- Sometimes have to match neighbors



# Marching Cubes Results





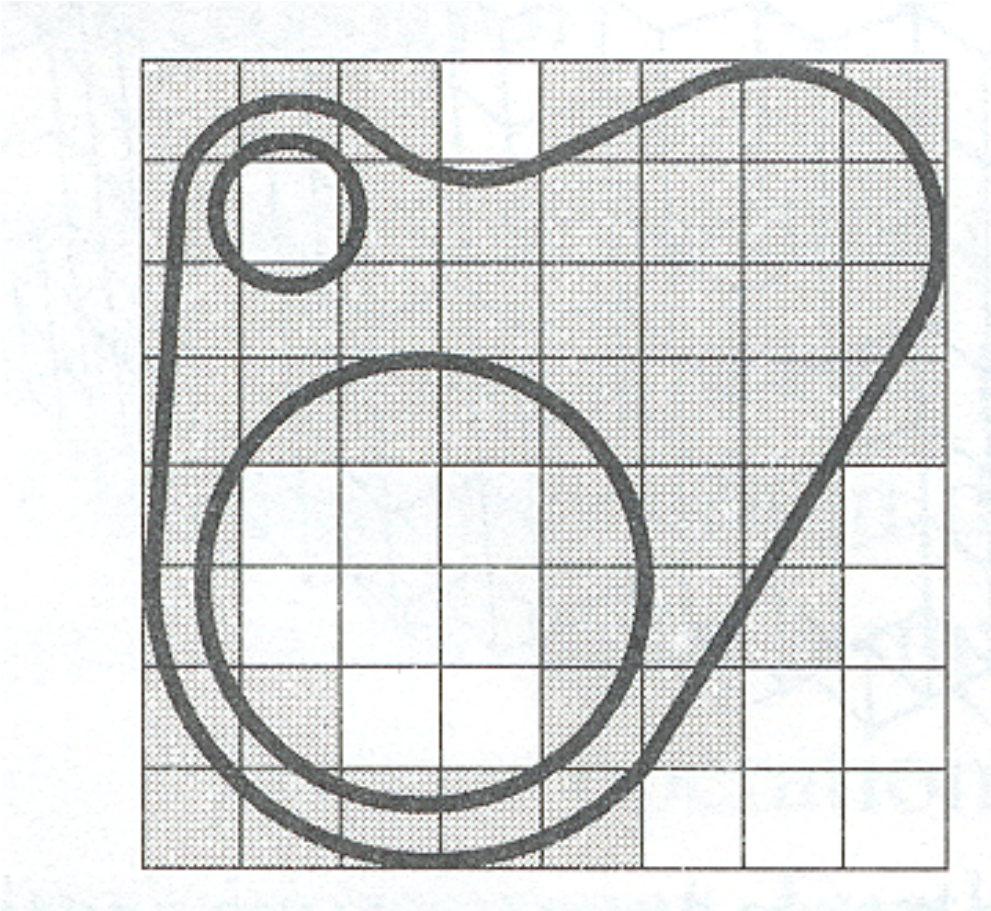
http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

# Voxels

- Advantages
  - Simple, intuitive, unambigiuous
  - Same complexity for all objects
  - Natural acquisition for some applications
  - Trivial boolean operations
- Disadvantages
  - Approximate
  - Large storage requirements
  - Expensive display

# Voxels

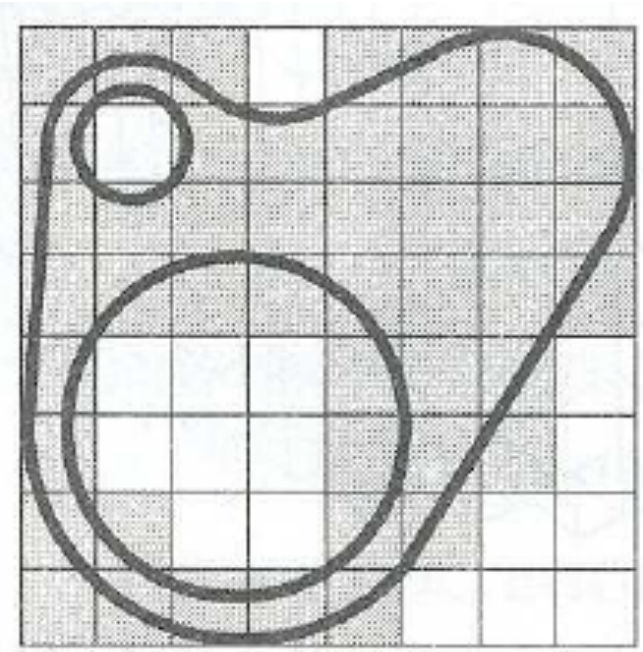
- What resolution should be used?



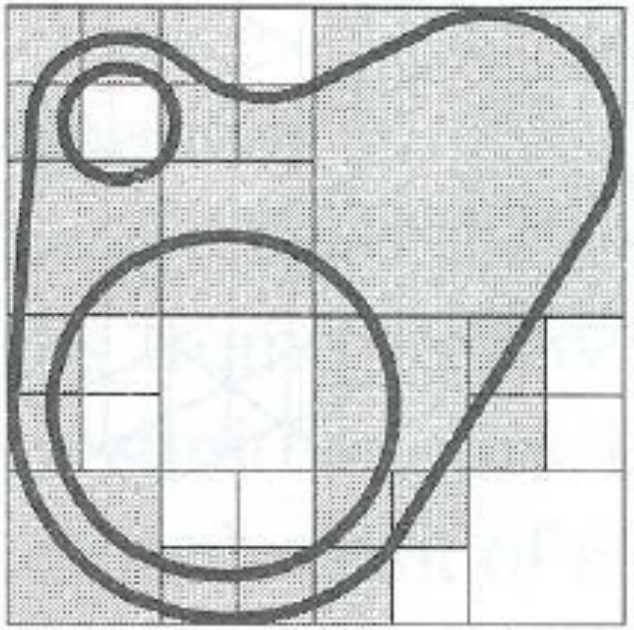
FvDFH Figure 12.21

# Octrees

- Refine resolution of voxels hierarchically
  - More concise and efficient for non-uniform objects



Uniform Voxels

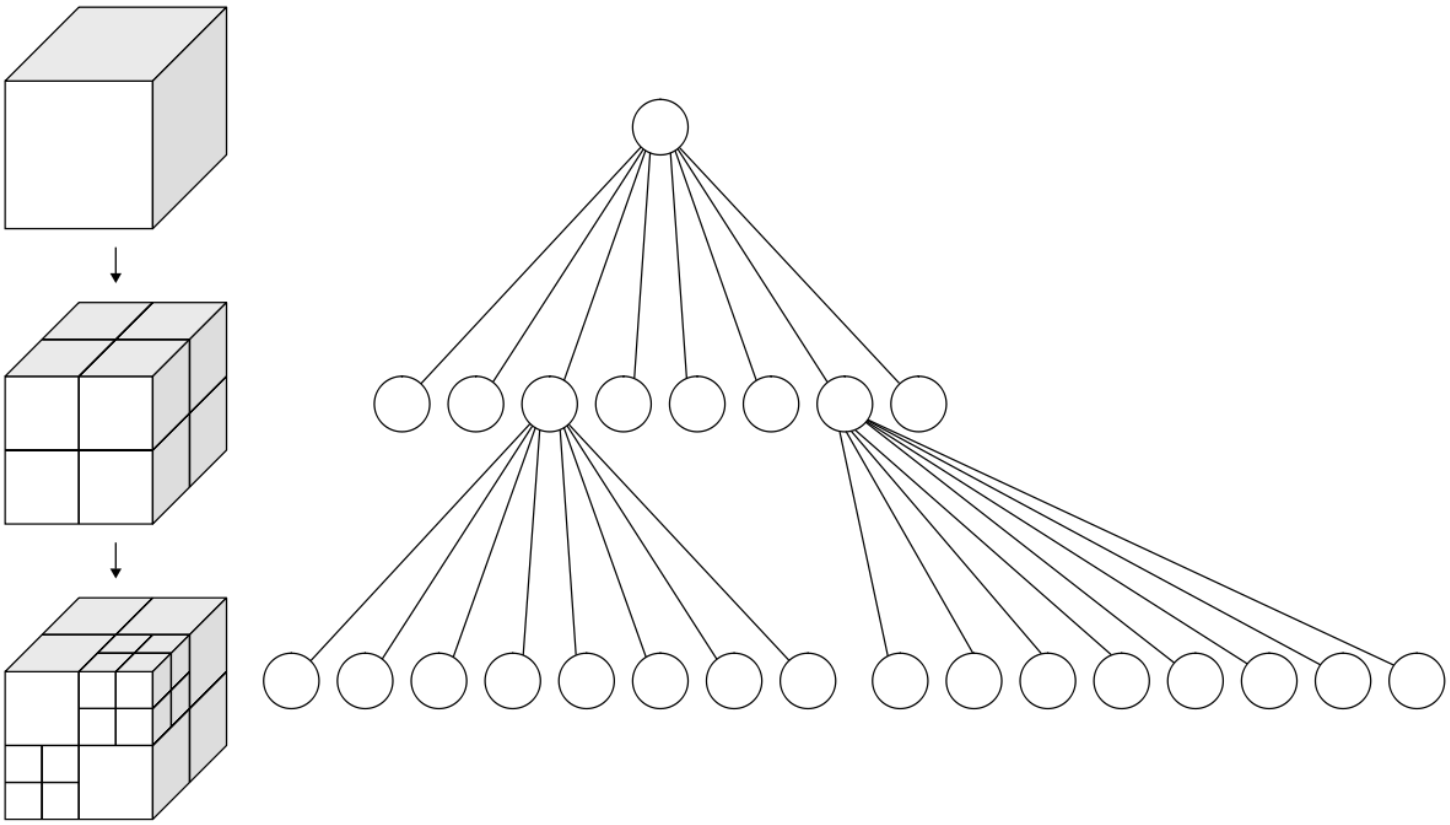


Quadtree

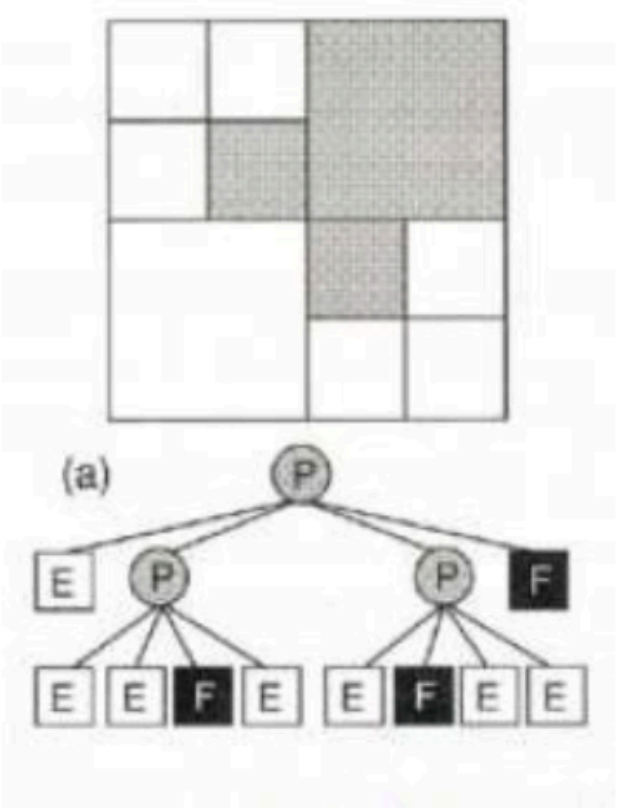
Source: FvDFH Figure 12.20

# Octrees

- Encoded using a standard tree data structure
  - Empty nodes do not need to be encoded explicitly



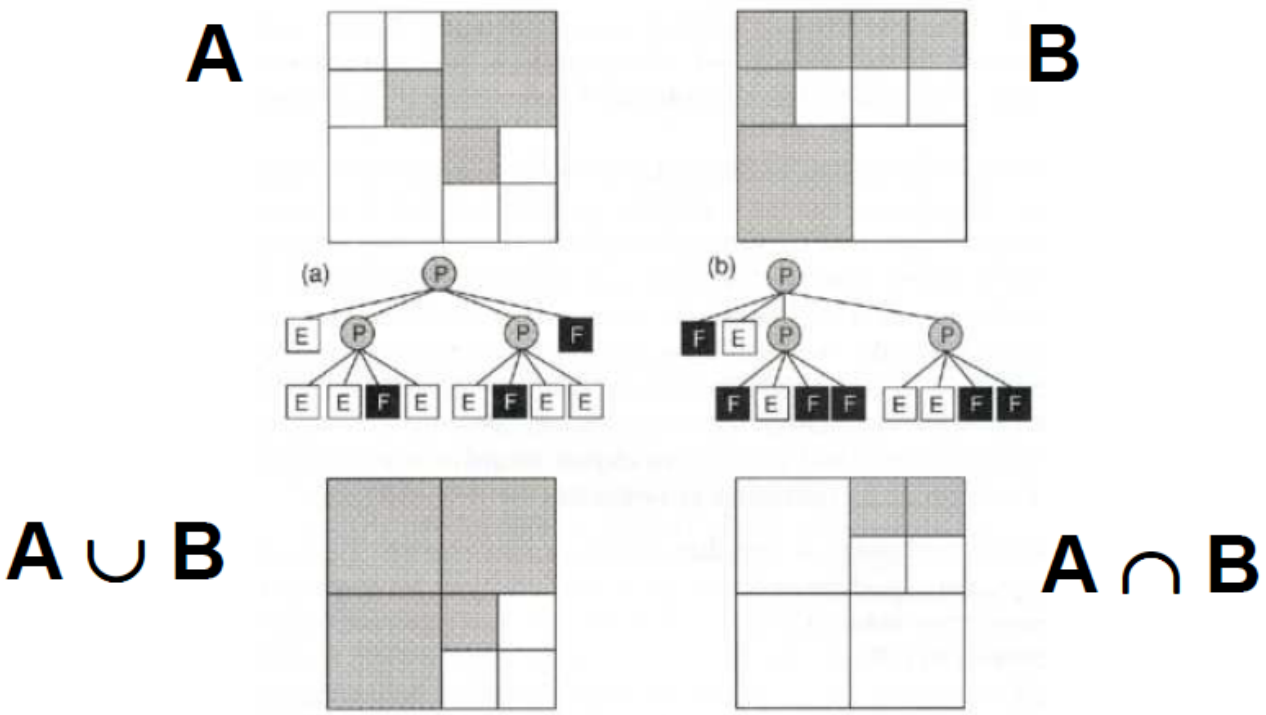
# Octrees




P-Partial
F-Full
E-Empty

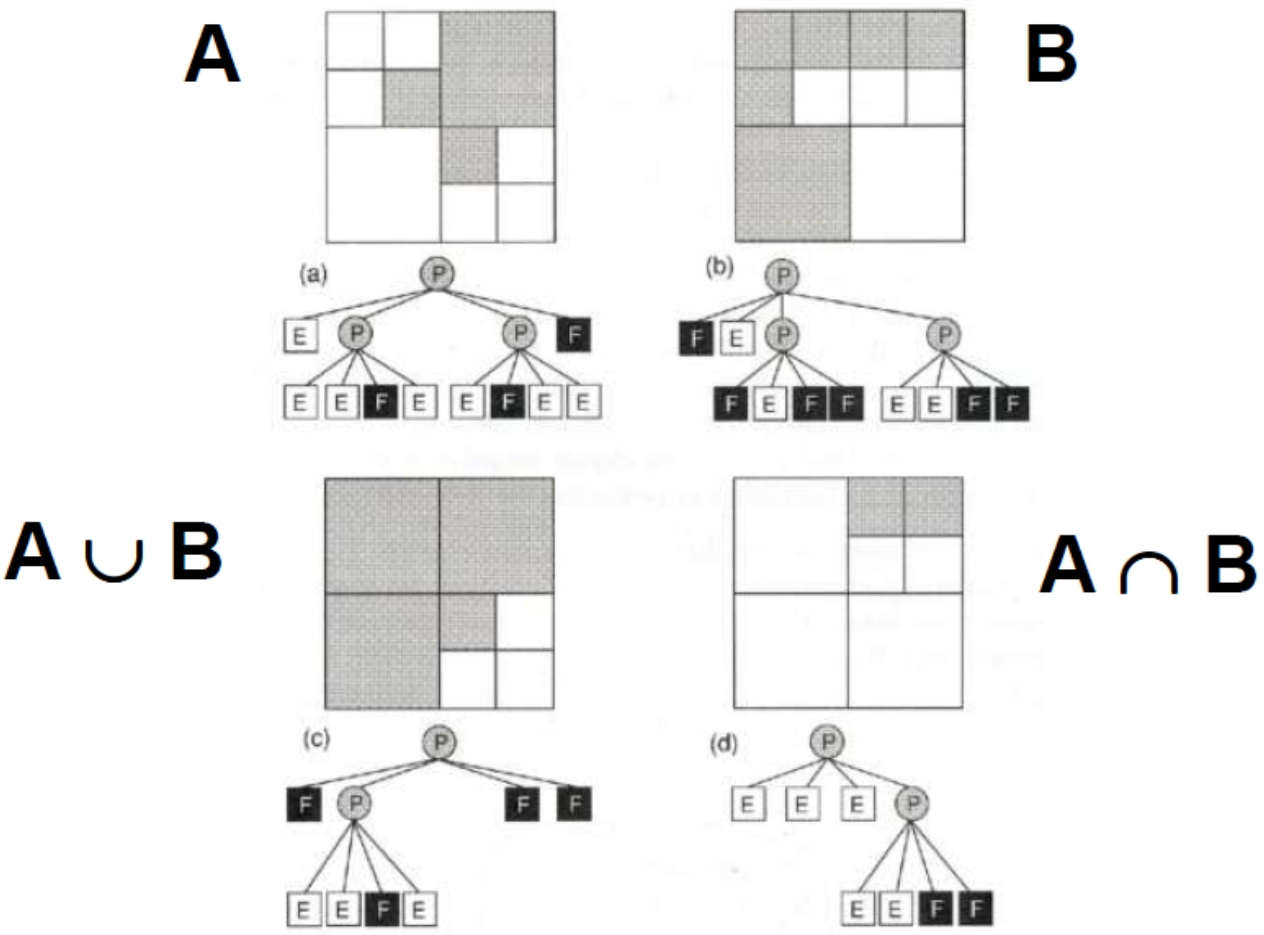
Source: FvDFH Figure 12.24

# Octree Boolean Operations




Source: FvDFH Figure 12.24

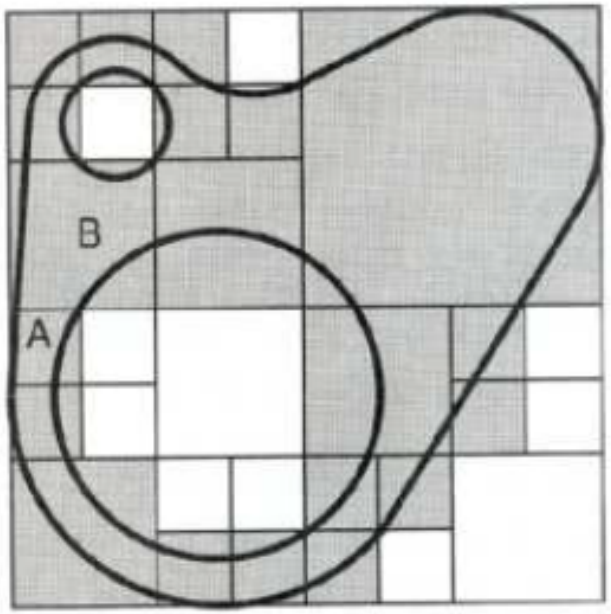
# Octree Boolean Operations

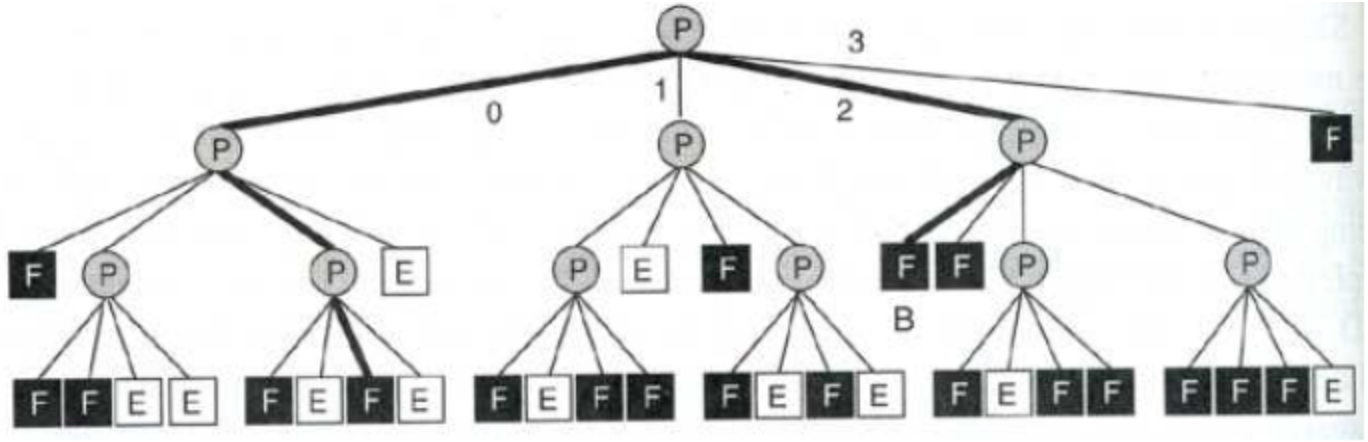



Source: FvDFH Figure 12.24

# Octree Display

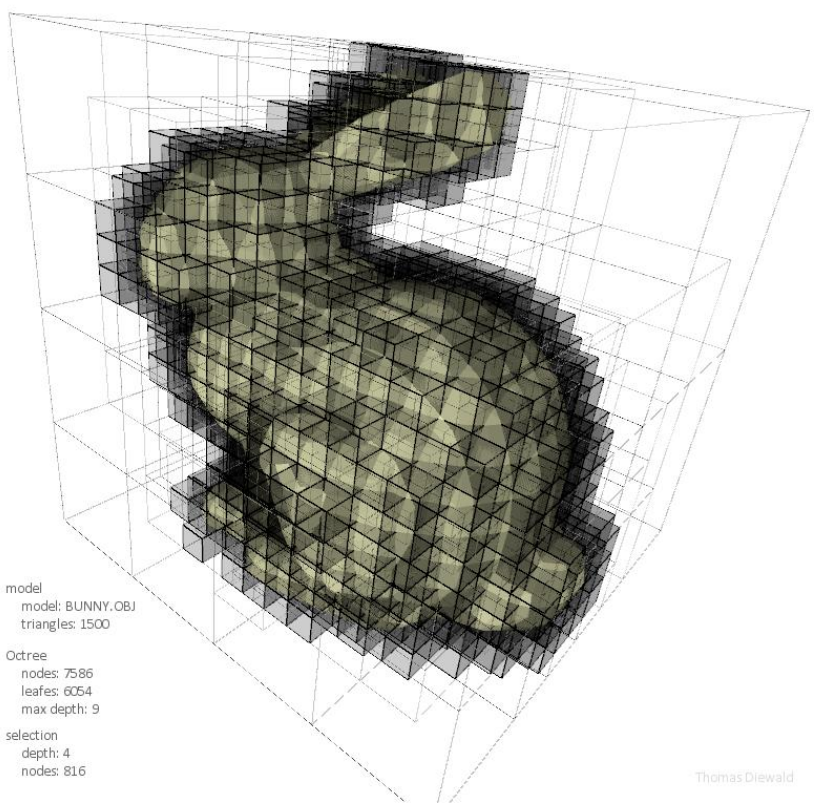
- Extend voxel methods
  - Slicing
  - Ray casting
- Finding neighbor cell requires traversal of hierarchy

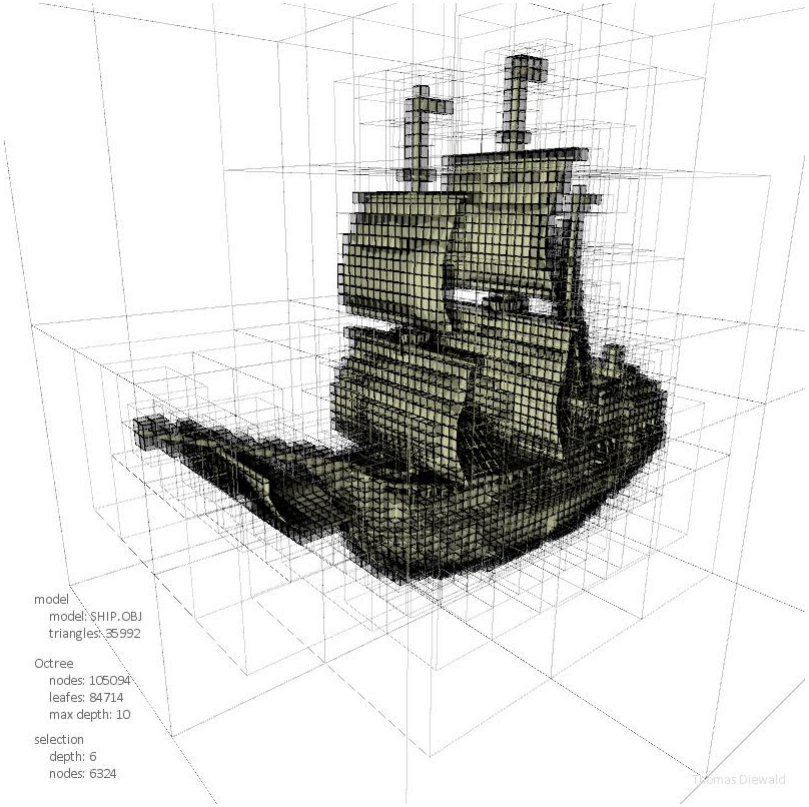



Source: FvDFH Figure 12.25

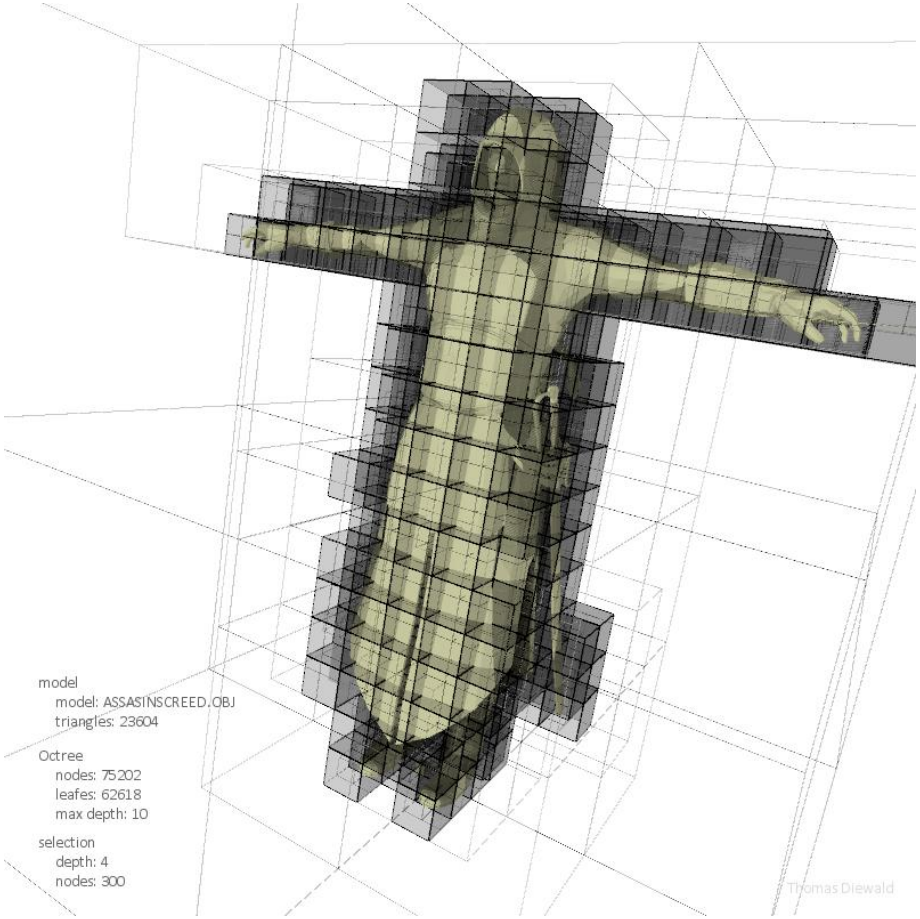
# Octree Examples

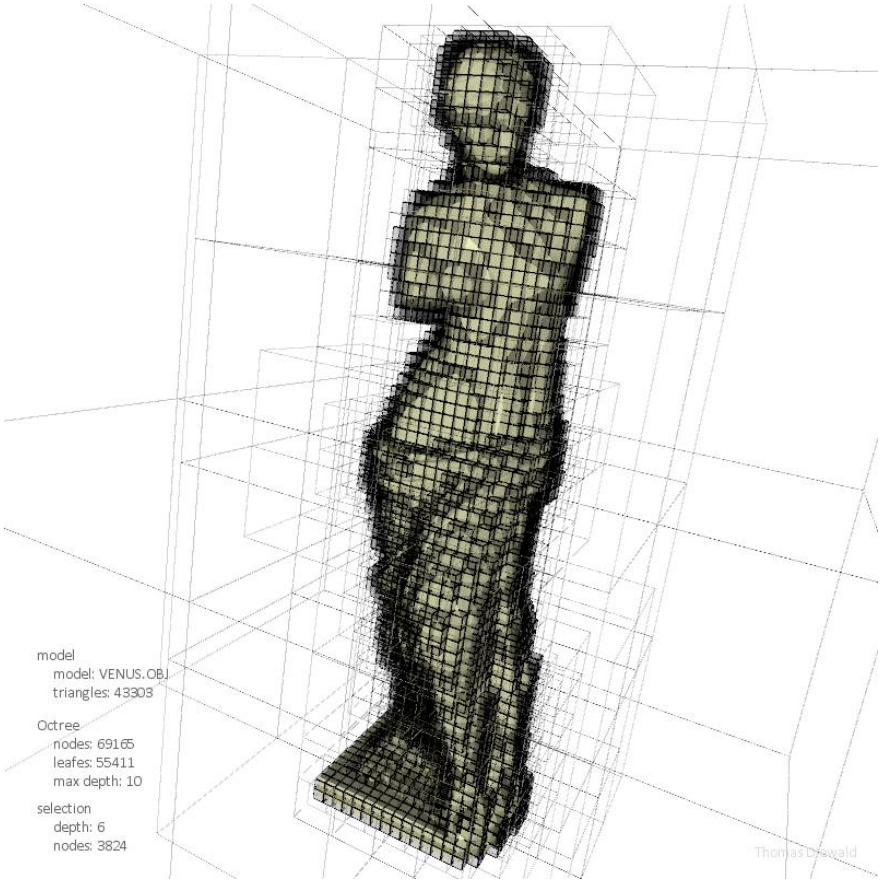



http://thomasdiewald.com

# Octree Examples



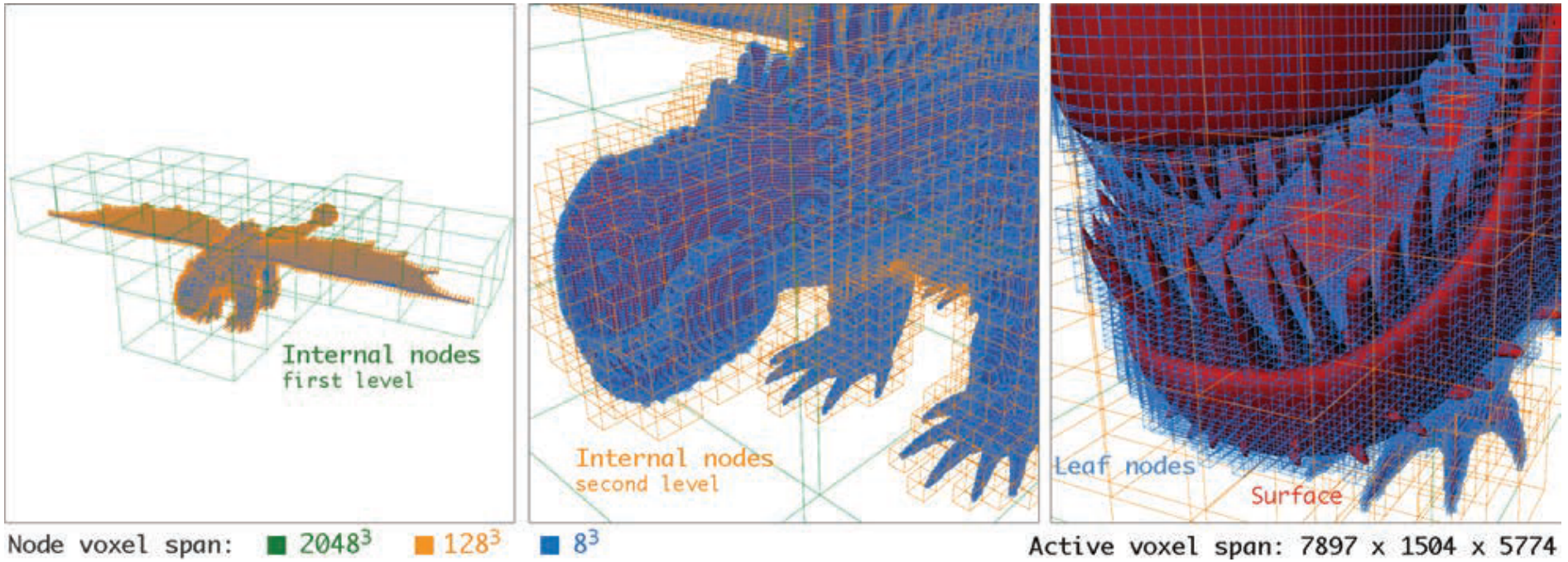




http://thomasdiewald.com

# Another Voxel Representation: OpenVDB

http://www.openvdb.org

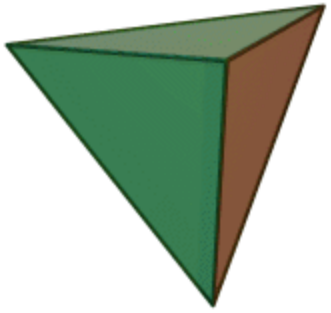
- Developed by DreamWorks Animation
  - Hierarchical, sparse representation for time-varying voxel data
  - Compact storage and fast data access



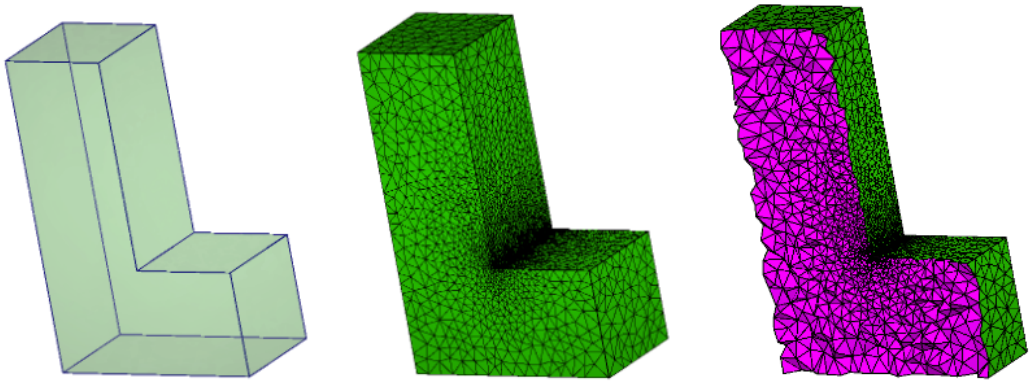

Source: Ken Museth

# Tetrahedra as Volume Representations

- Tetrahedron (Tet)
  - 4 vertices, 4 faces
- Tetrahedral mesh (Tet Mesh)
  - Similar to a standard mesh
    - A list of vertices
    - A list of tetrahedra



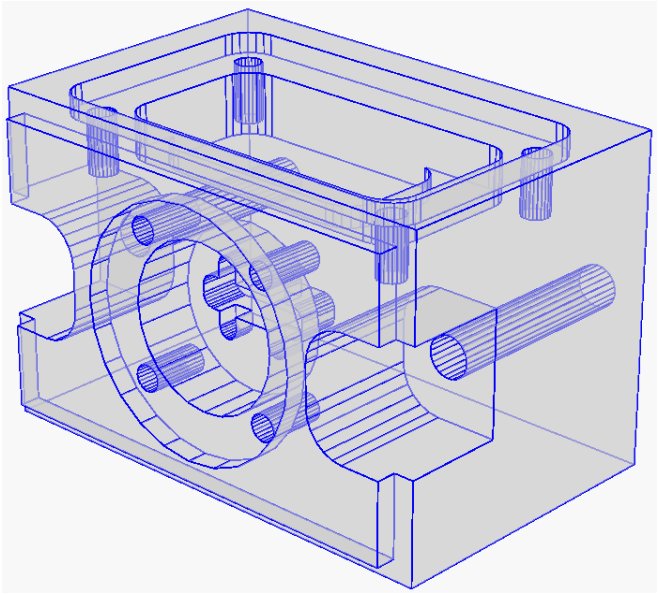
Source: Wikipedia



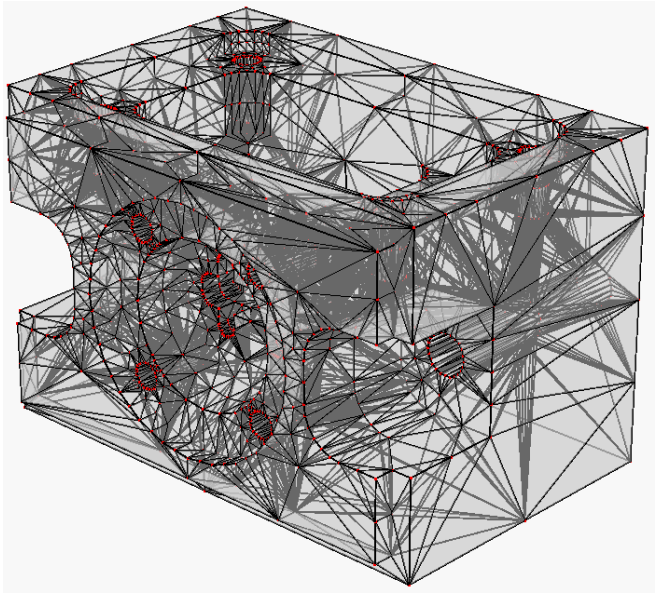
Source: Tetgen.org

# Tetrahedralization

- Conversion from a surface representation to a tetrahedral mesh
  - Tetgen by Hang Si



Input Mesh

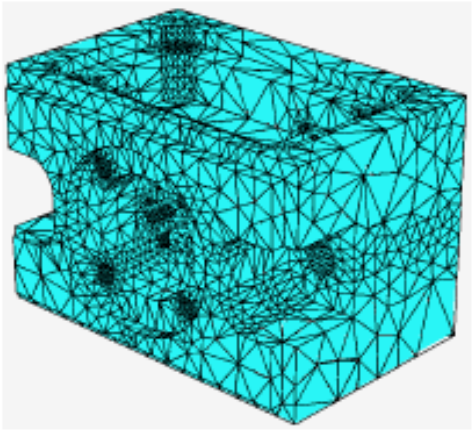


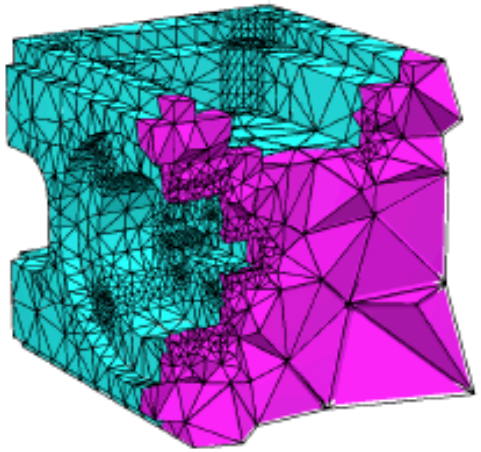
Constrained Delaunay Tetrahedralization

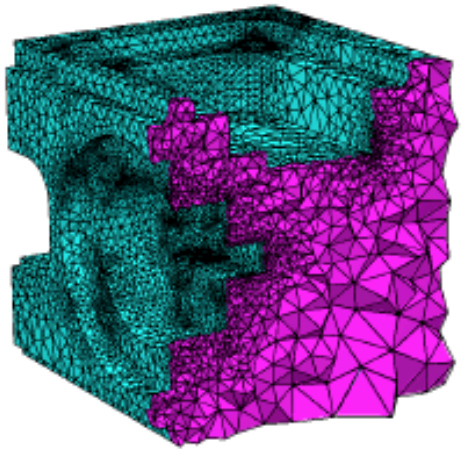
Source: Tetgen.org

# Tetrahedralization

- Preserves mesh boundary
  - Boundary trignales and tet boundary coincide
- Knobs to refine subdivision
  - New vertices can be inserted
    - Guarantee tet quality (volume, angles)



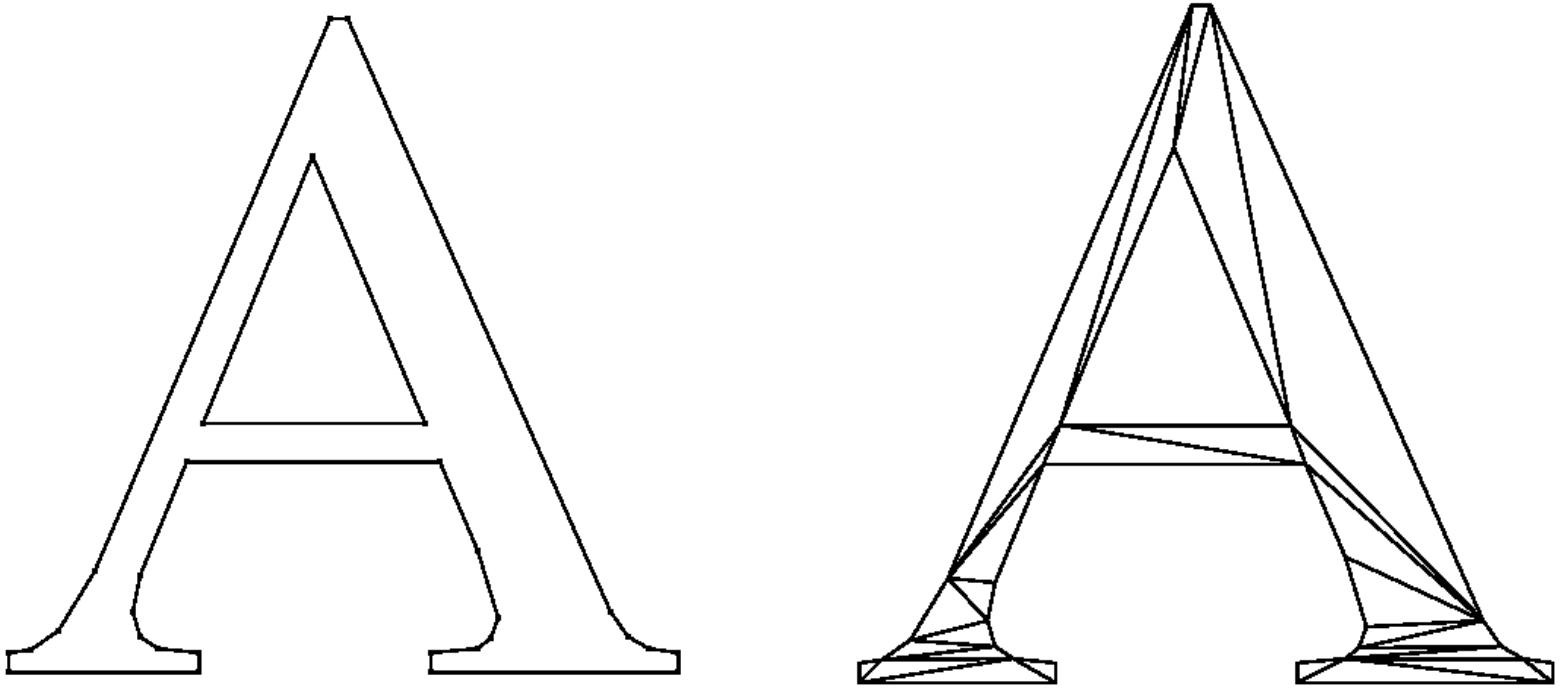




Source: Tetgen.org

# Tetrahedralization

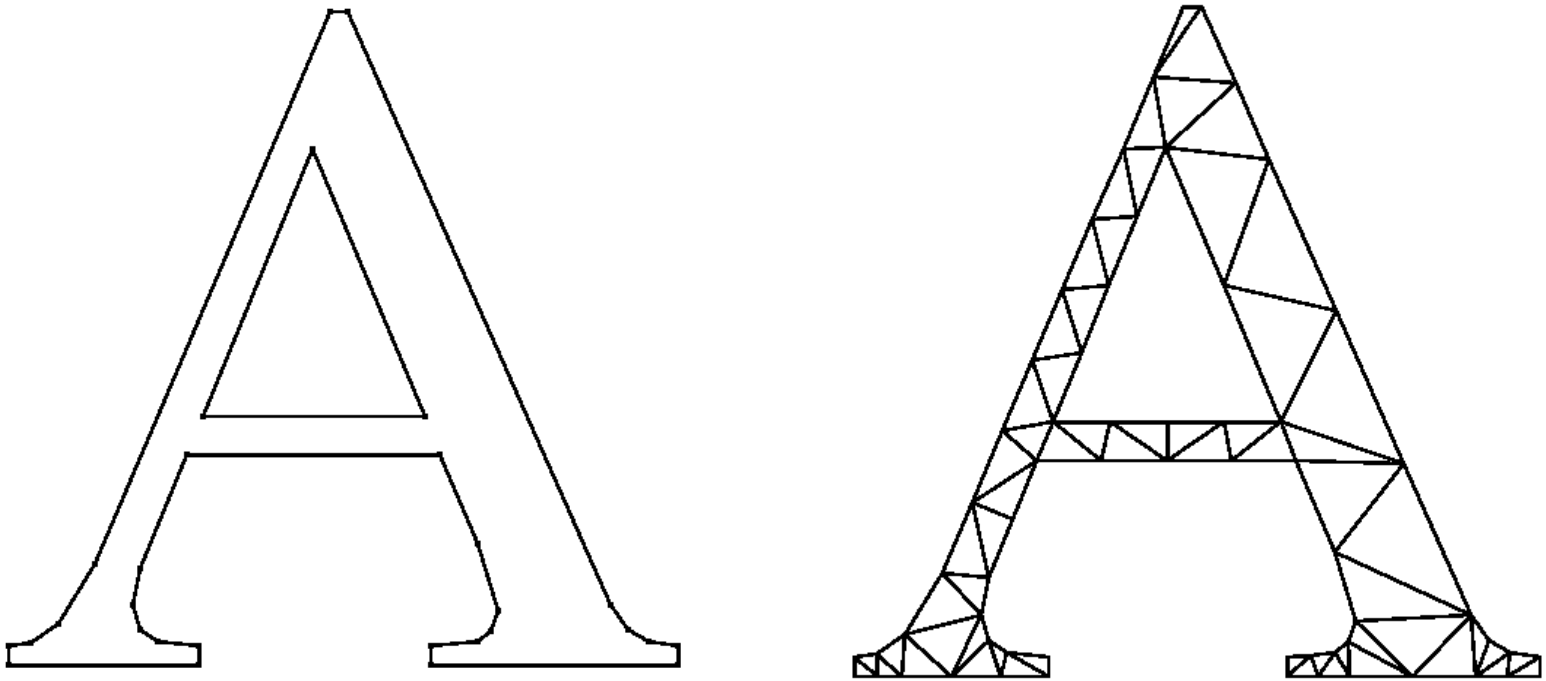
- Boundary edges are forced to be in the tetrahedralization



http://www.cs.cmu.edu/~quake/triangle.defs.html

# Tetrahedralization

- Faces can be subdivided into several faces by inserting additional vertices



http://www.cs.cmu.edu/~quake/triangle.defs.html

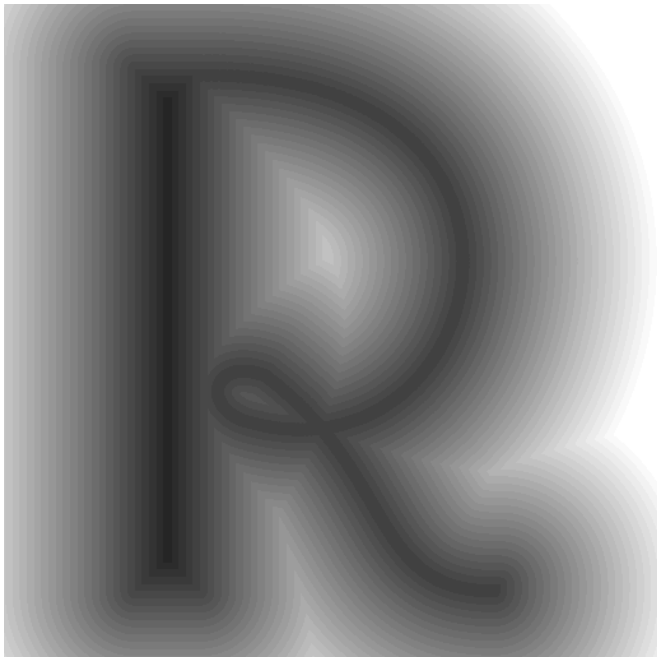
# The Plan For Today

- Discrete Volume Representations
  - Voxels
    - Voxelization (from surfaces to voxels)
    - Marching Cubes (from voxels to surfaces)
  - Octrees
  - Tetrahedra
- **Implicit Representations**
  - **Distance Fields**

# 2D Euclidean Distance Field Example

- An object's distance field represents, for any point in space, the distance from that point to the object
  - The distance can be signed to distinguish between the inside and outside of the object

R shape

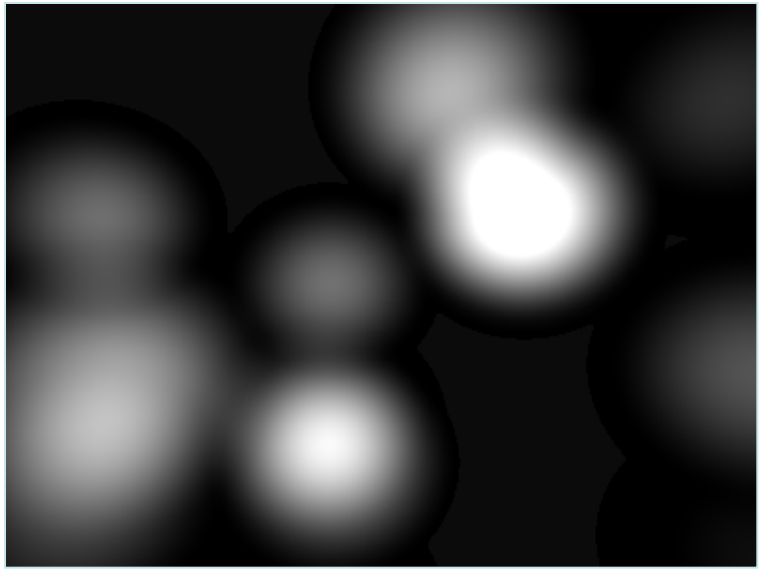


Distance field of R

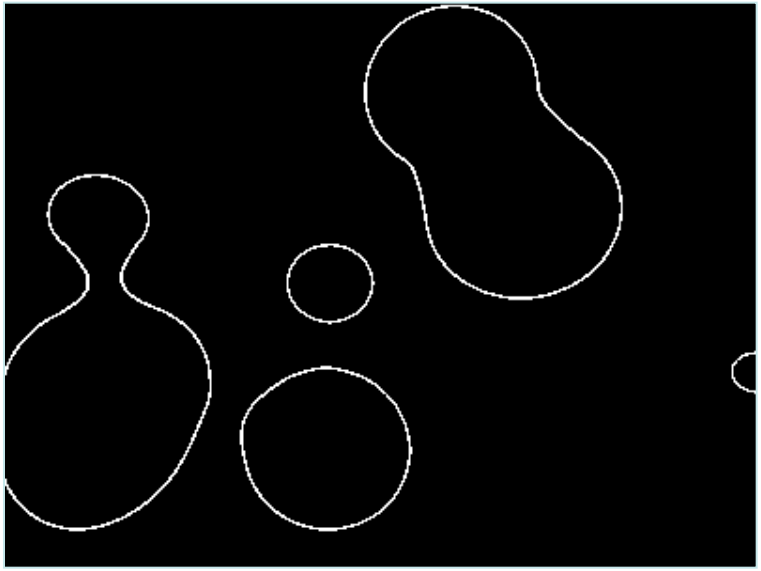
Perry and Frisken

# Distance Fields

- Typically, for a shape represented by a distance field, the shape's boundary, $\Omega$, is the zero-valued iso-surface of the distance function



*A 2D cross section of F(x)*

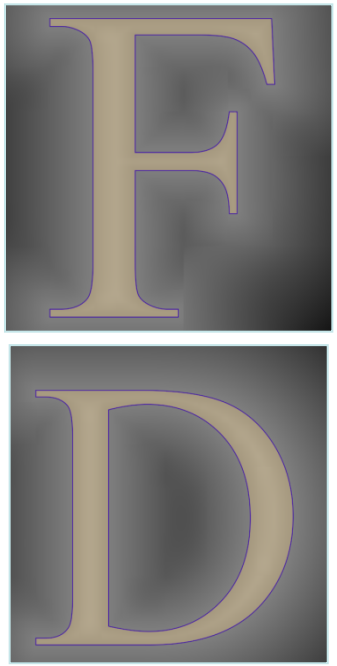


*An iso-contour of F(x) where F(x) = 0*

Perry and Frisken

# Boolean Operations on Distance Fields

- Fast and simple



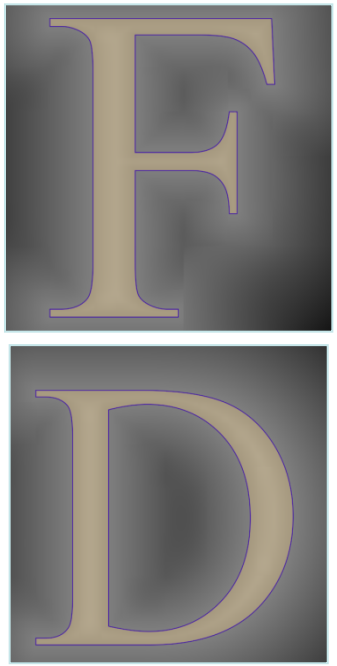
*Union of two shapes*

*Union of two distance fields*

# Boolean Operations on Distance Fields

- Fast and simple

*Intersection: dist(A ∩ B) = min (dist(A), dist(B))*
*Union: dist(A ∪ B) = max (dist(A), dist(B))*
*Difference: dist(A - B) = min (dist(A), –dist(B))*

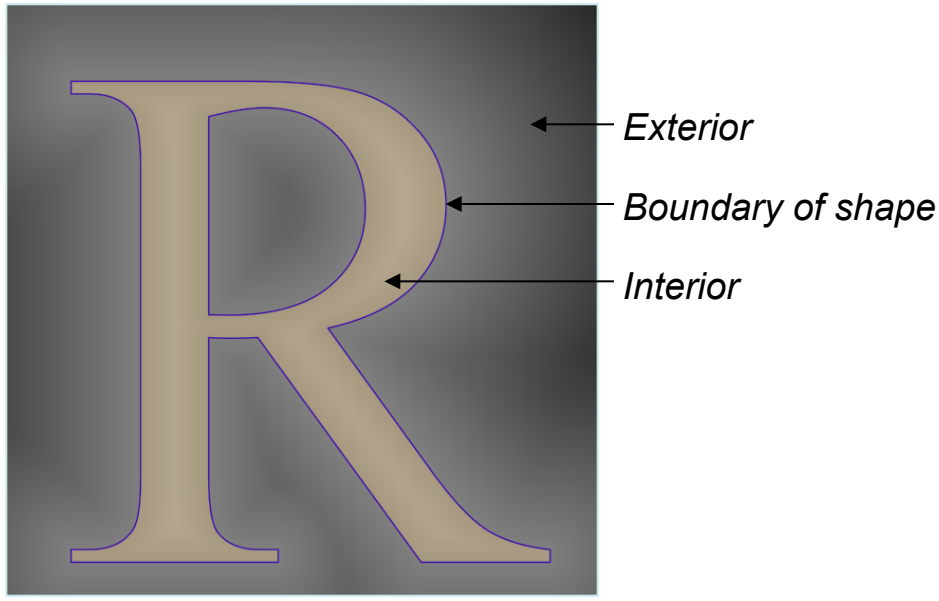


*Union of two shapes*

*Union of two distance fields*

# Advantages

- Distance fields represent more than just the object outline
  - Represent the object interior, exterior, and its boundary

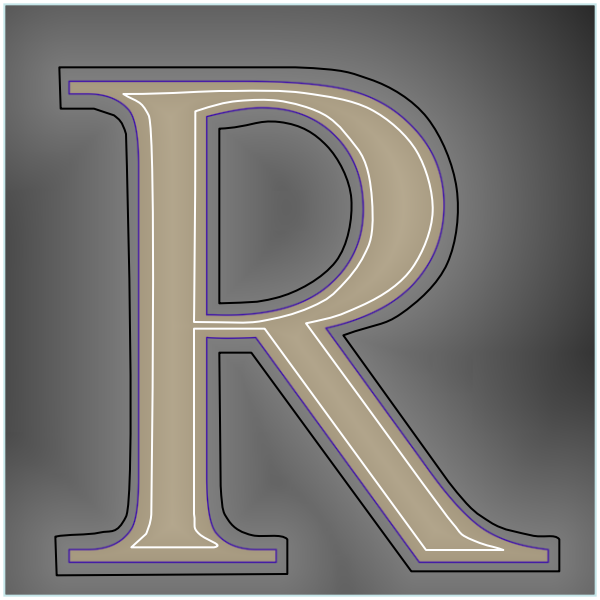



*Shape's distance field*

Perry and Frisken

# Advantages

- Distance fields represent more than just the object outline
  - Represent an infinite number of offset surfaces

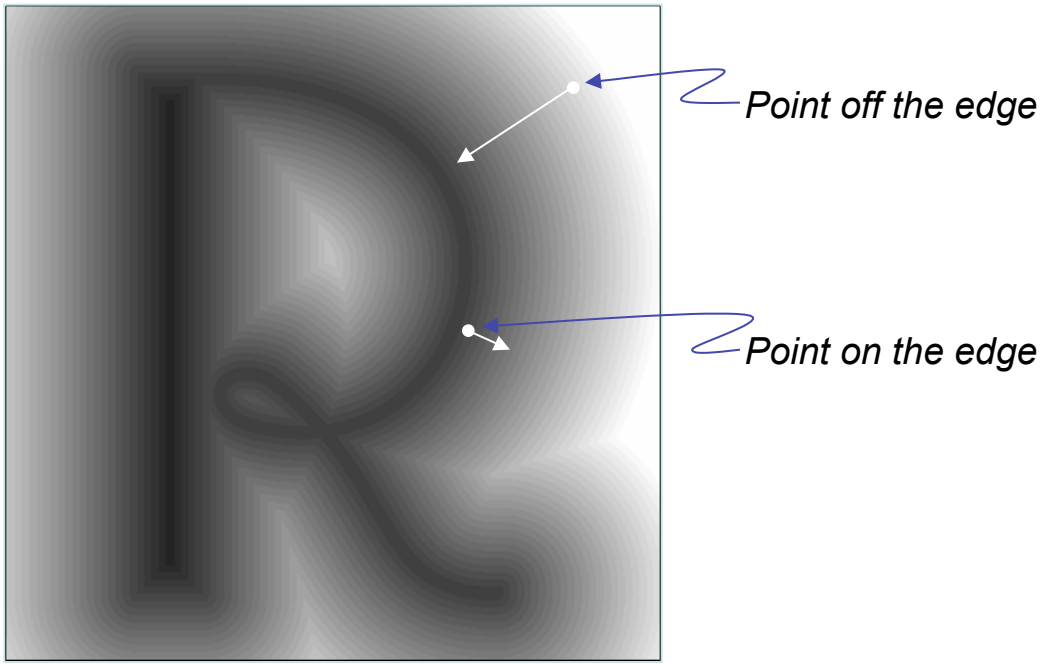


*Boundary offsets*

Perry and Frisken

# Advantages

- Gradient of the distance field yields
  - Surface normal for points on the edge
  - Direction to the closest edge point for points off the edge

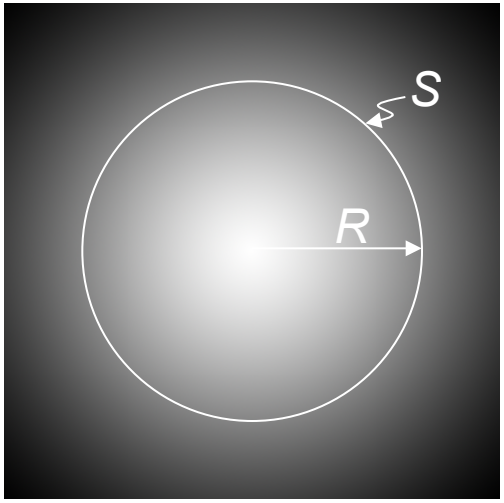


# General Properties

- Distance fields are defined everywhere in space
  - Not just on the surface like boundary representations
- It is trivial to determine whether a point is inside, outside or on the boundary of a shape
  - Evaluate F(x) and compare it to the value of the iso-surface
- Gradients of the distance field provide geometric information
  - On the surface: the gradient is normal to the surface
  - Off the surface: the gradient points in the direction of the closest surface point

# Representing Distance Fields

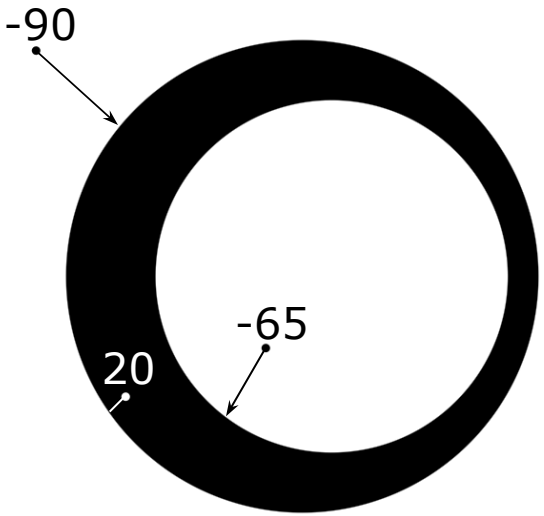
- Implicit representation
  - e.g., sphere, dist(x,y,z) = R - $\sqrt{(x - c_x^*)^2 + (y - c_y^*)^2 + (z - c_z^*)^2}$



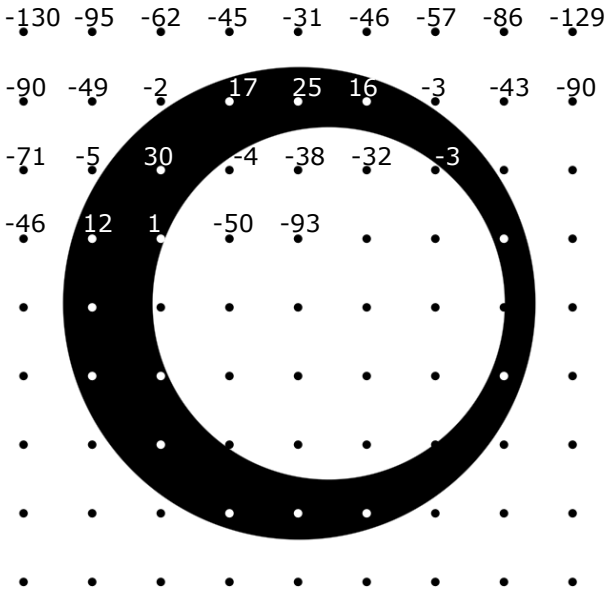

  - Complex models can be represented via CSG

# Representing Distance Fields

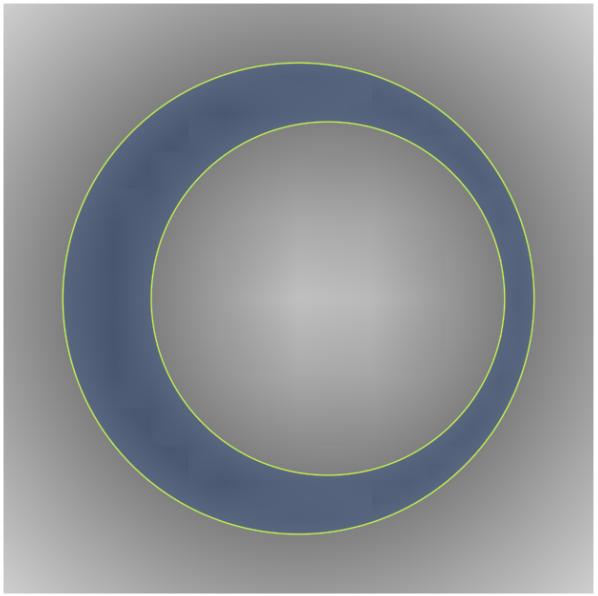
- Sampled volumes
  - Distances are computed and stored in a regular 3D grid
  - Distances at non-grid locations are interpolated




2D shape with sampled distances to the surface




Regularly sampled distance values

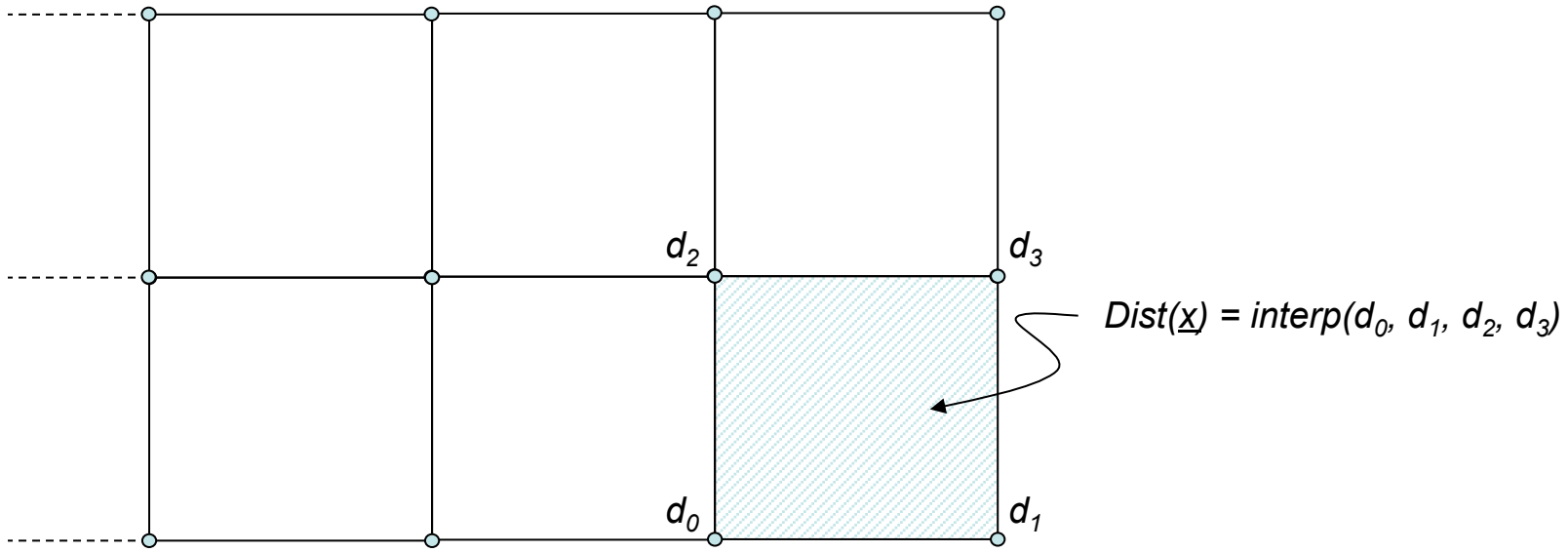


2D distance field

Perry and Frisken

# Locally Implicit Representation

- Reconstruction from samples
  - Distances at positions between sample points are reconstructed from sampled distances




*Sampling on a rectilinear grid with a bi-linear reconstruction function*

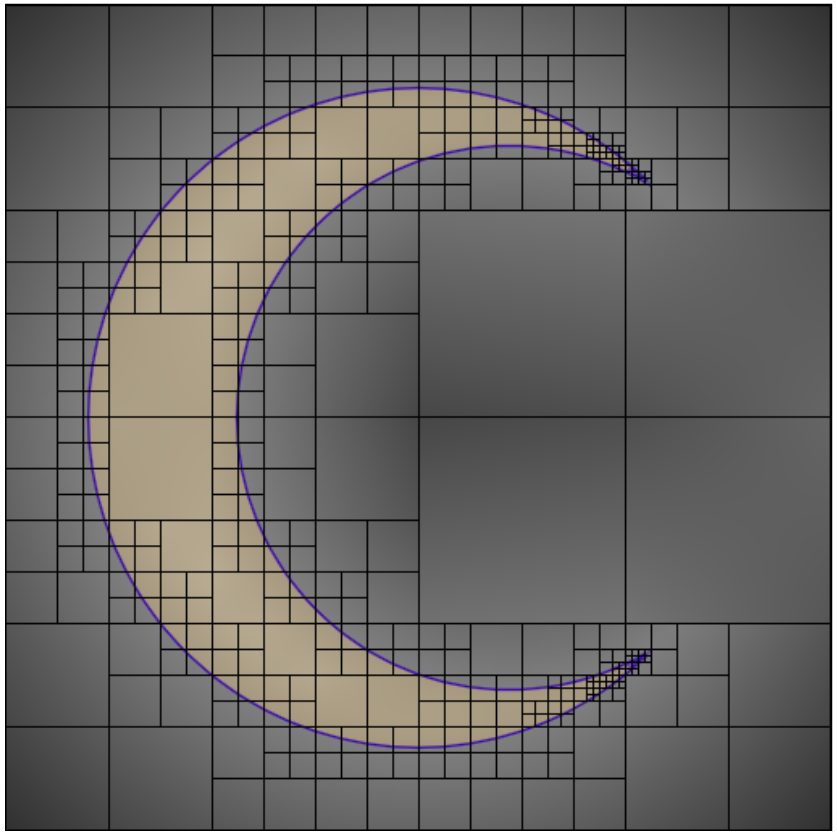
Perry and Frisken

# Regularly Sampled Distance Fields

- Distance fields must be sampled at high enough rates to avoid aliasing (jagged edges)
- Very dense sampling is required when fine detail is present
- Regularly sampled distance fields require excessive memory when *any* fine detail is present

# Adaptively Sampled Distance Fields

Sample at low rates where the distance field is smooth. Sample at higher rates only where necessary (e.g., near corners).



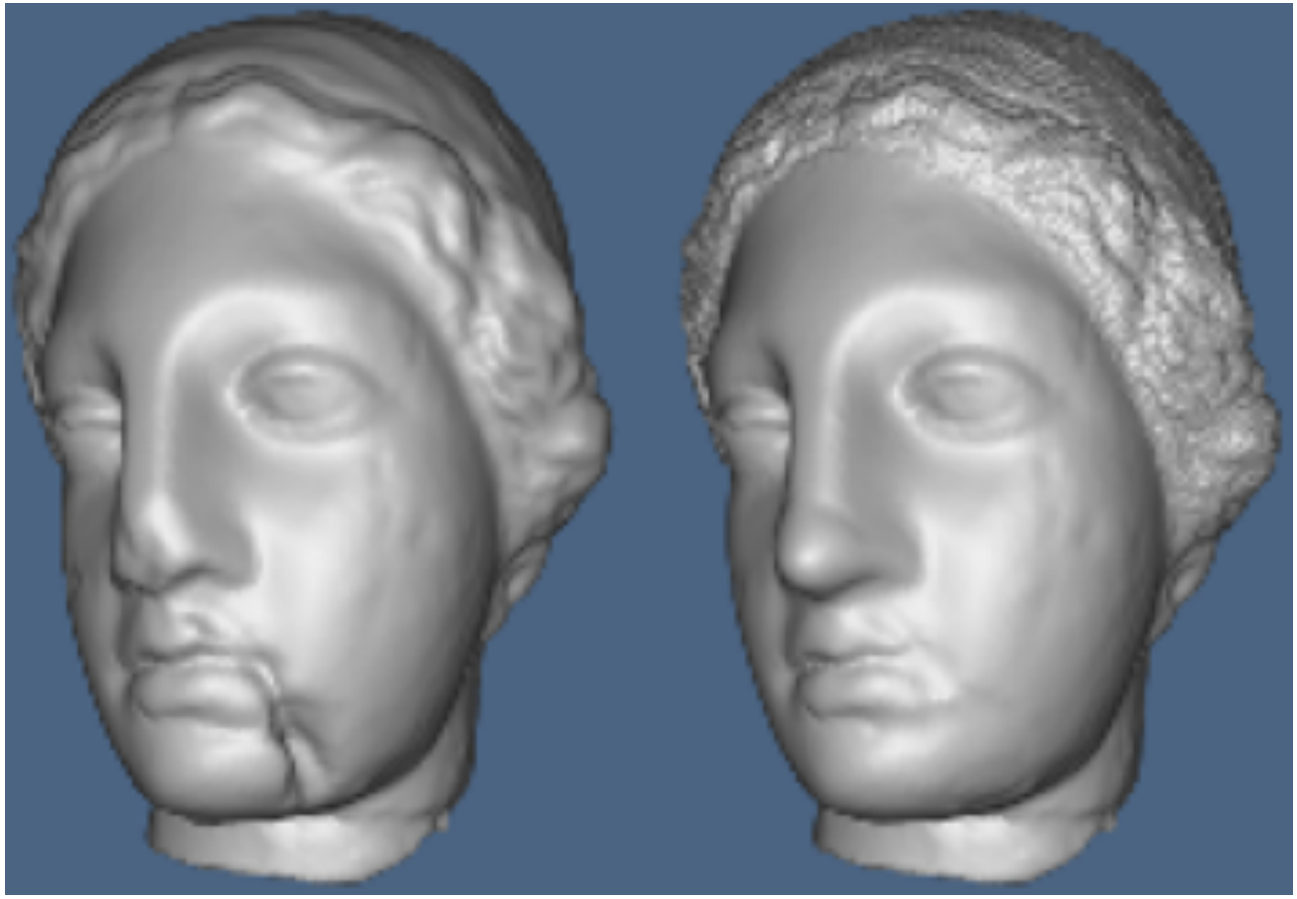
A 2D crescent ADF and its quadtree data structure

Perry and Frisken

# Adaptively Sampled Distance Fields

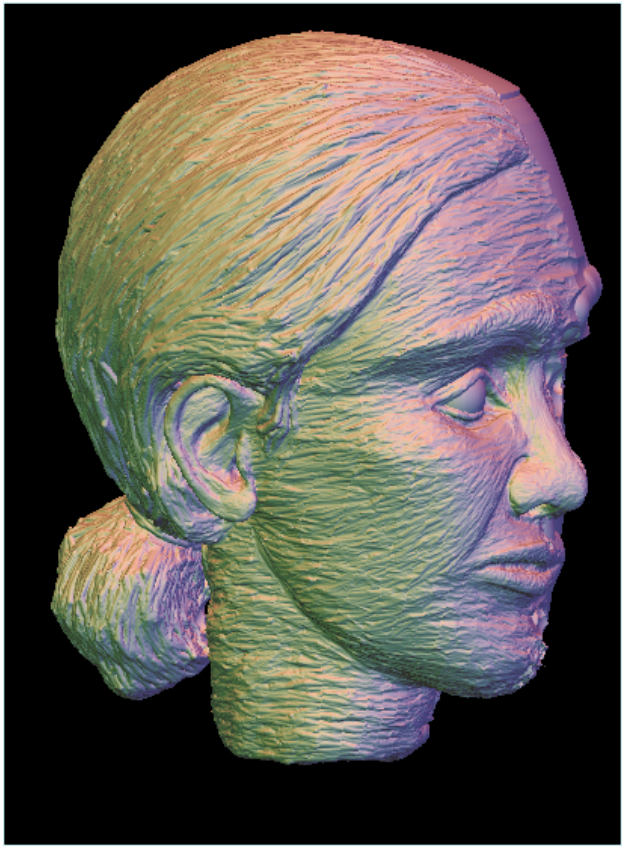
- Detail-directed sampling
  - high sampling rates only where needed
- Spatial data structure
  - fast localization for efficient processing
- ADFs consist of
  - adaptively sampled distance values …
  - organized in a spatial data structure …
  - with a method for reconstructing the distance field from the sampled distance values

# Applications

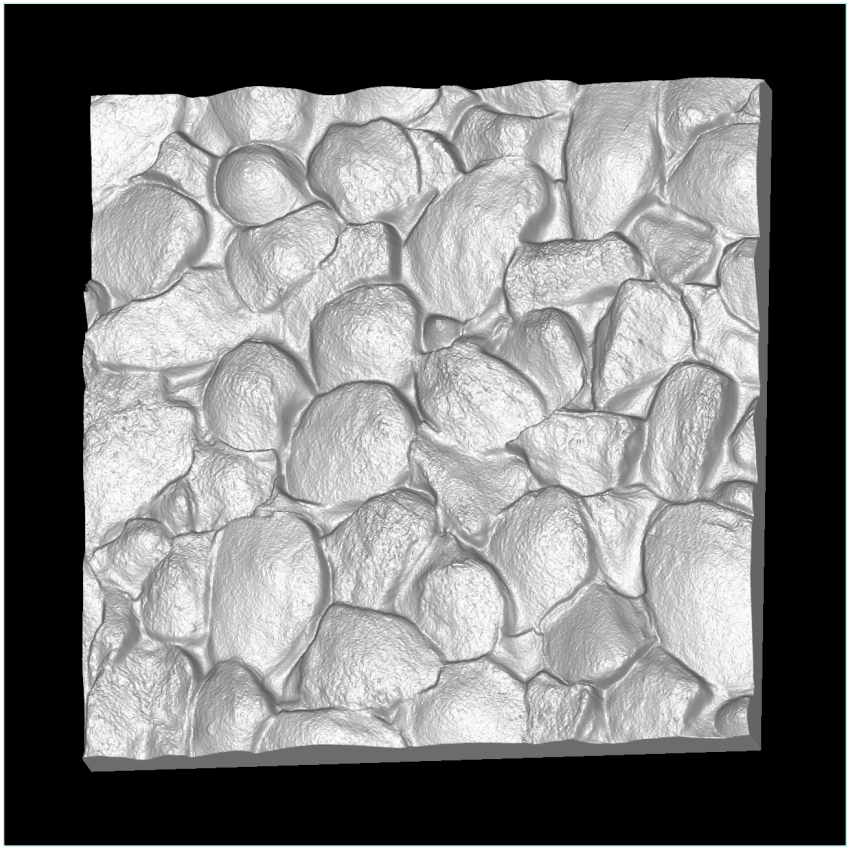


Museth et al. “Level Set Surface Editing Operators”, SIGGRAPH 2002

# Applications



*Freeform model carved from a sphere*



*Detailed model of stones from range data*

Perry and Frisken, *"Kizamu: A System for Sculpting Digital Characters", SIGGRAPH 2001*

# That's All For Today

- Readings:
  - Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics
    - http://www.merl.com/publications/docs/TR2000-15.pdf
  - Marching cubes: A high resolution 3D surface construction algorithm
    - http://dl.acm.org/citation.cfm?id=37422