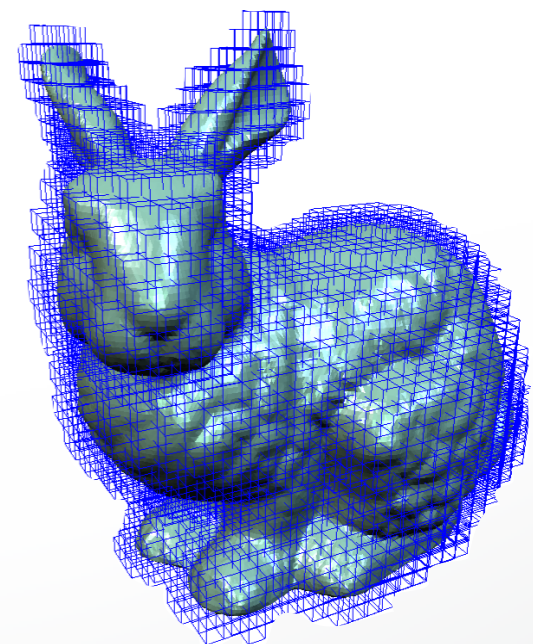


## **2.1 Explicit & Implicit Surfaces**

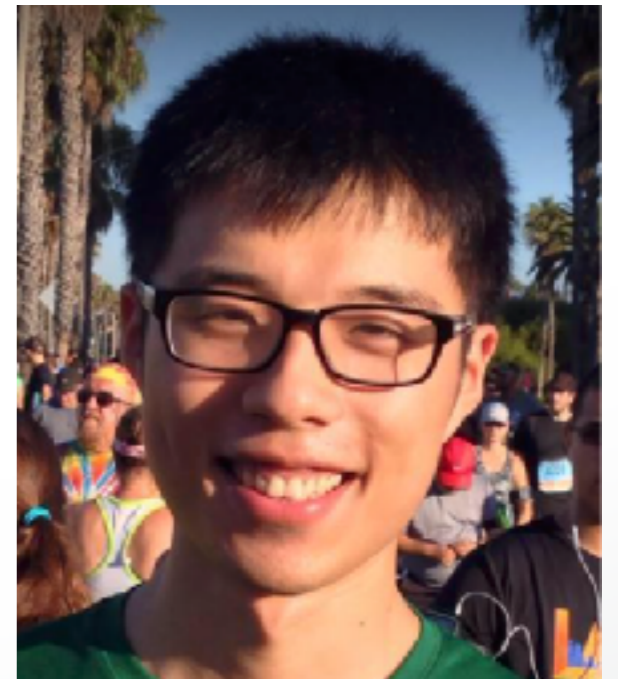


Hao Li

<http://cs621.hao-li.com>

# Administrative

- **Exercise 1 discussion: Later**
- **Hao Li (Instructor)**
  - Office Hour: Tue 12:30 PM - 1:30 PM, SAL 244
- **Tianye Li**
  - Office Hour: Mon 3:00 PM - 4:00 PM, PHE 108
  - Tianye Li, [tianyeli@usc.edu](mailto:tianyeli@usc.edu)



# Last Time

## Polygonal meshes are

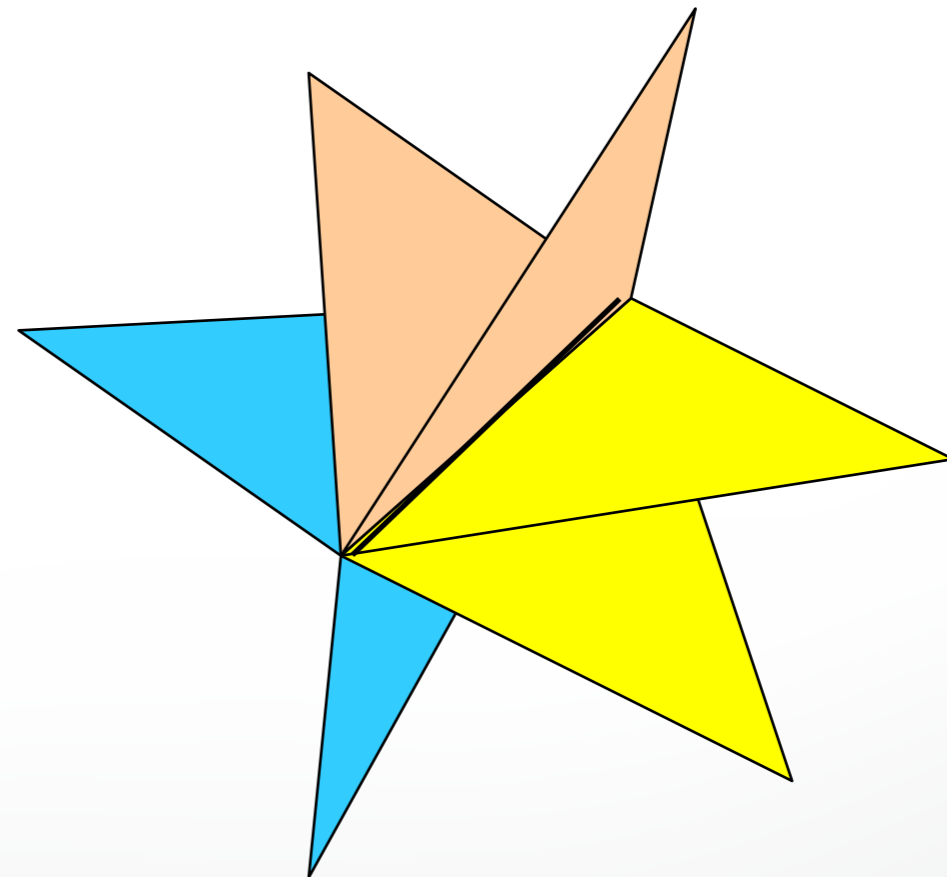
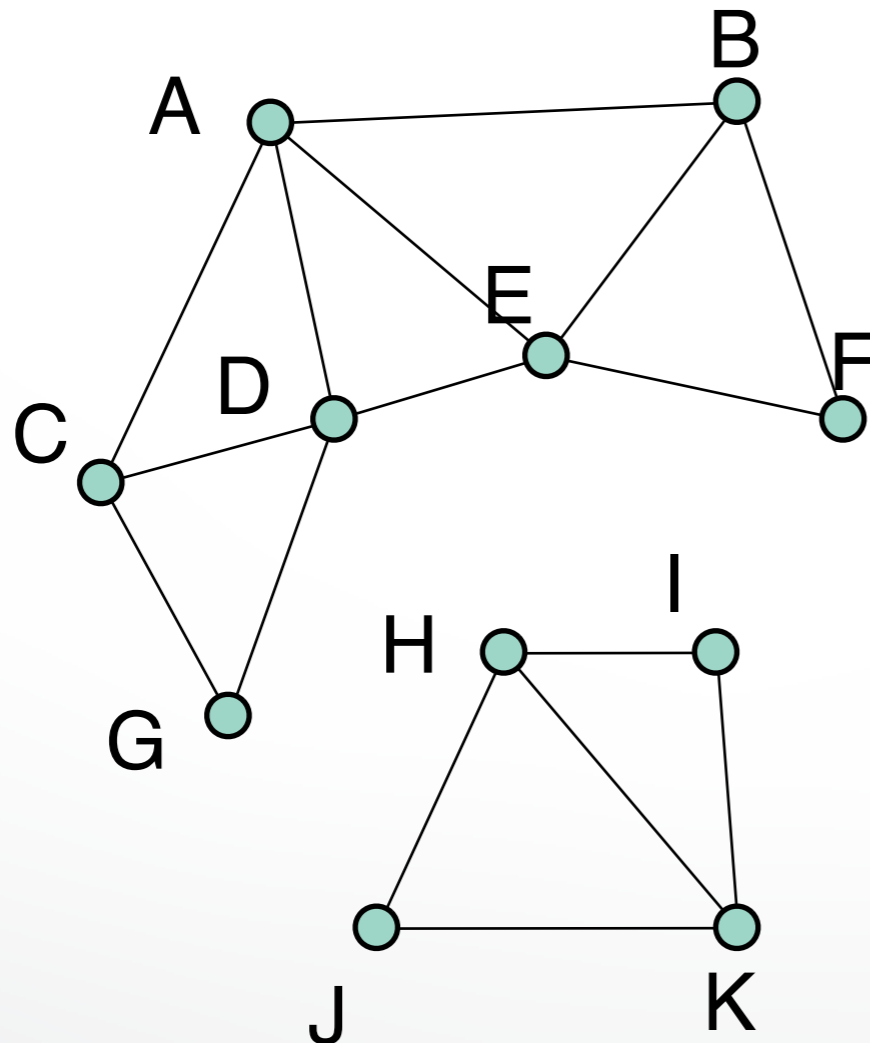
- Effective representations
- Flexible
- Efficient, simple, enables unified processing



# Last Time

## Connection between Meshes and Graphs

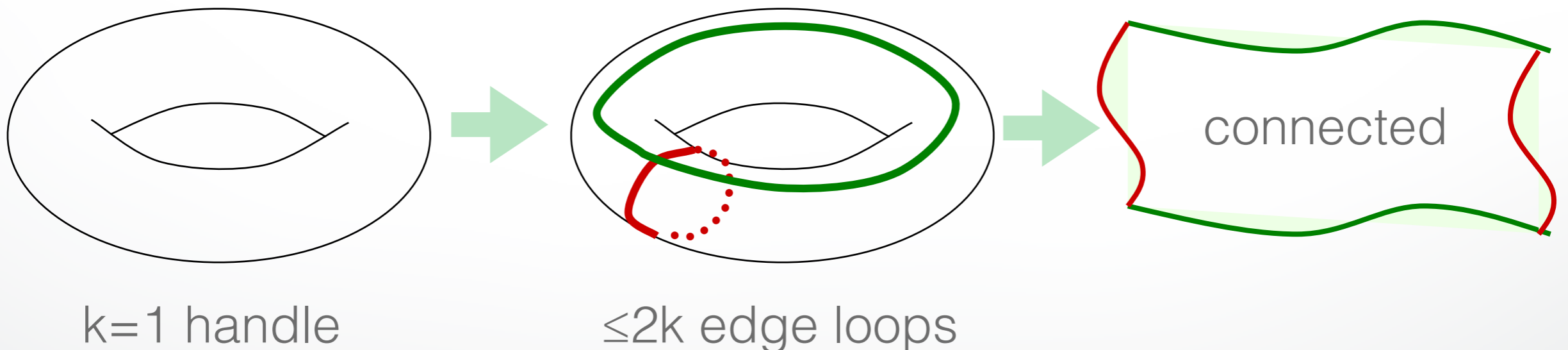
- Formalism (valence, connections, subgraph, embedding...)
- Definitions (boundary, regular edge, singular edge, closed mesh)
- triangulation  $\rightarrow$  triangle mesh



# Last Time

## Topology

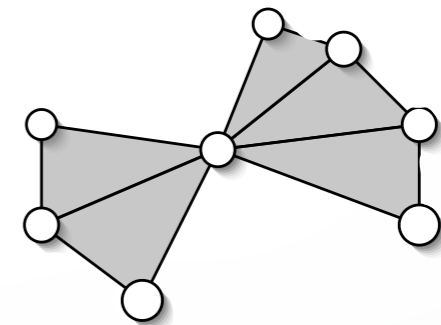
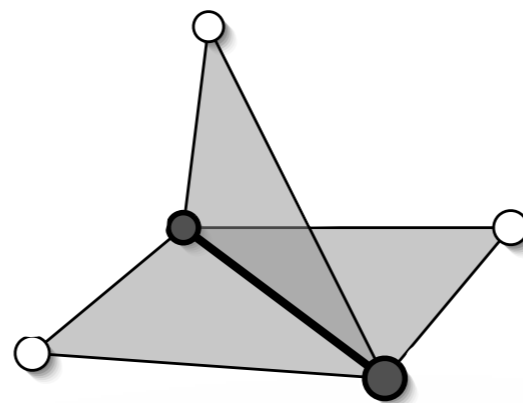
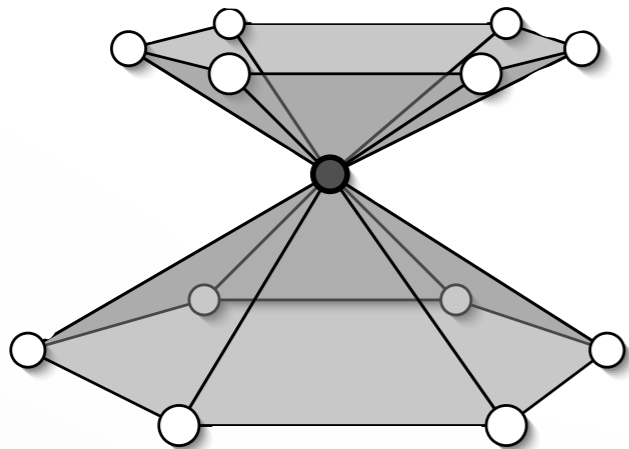
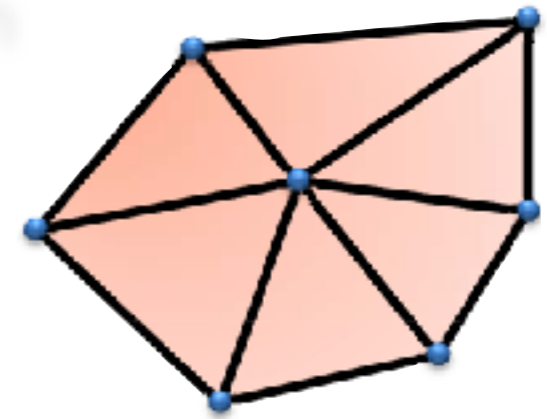
- Genus, Euler characteristic
- Euler Poincaré formula  $V - E + F = 2(1 - g)$
- Average valence of triangle mesh: 6
- Triangles:  $F = 2V$ ,  $E = 3V$
- Quads:  $F = V$ ,  $E = 2V$



# Last Time

## 2-Manifold Surface

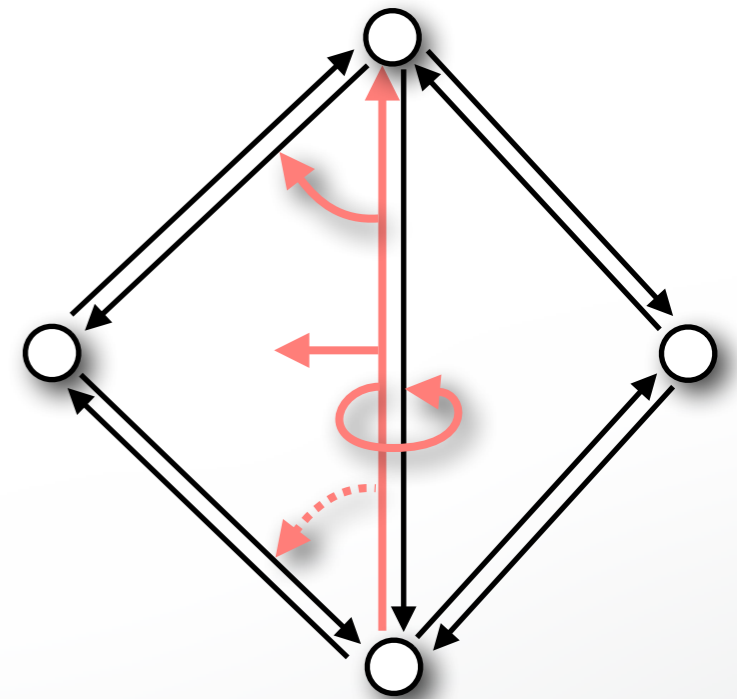
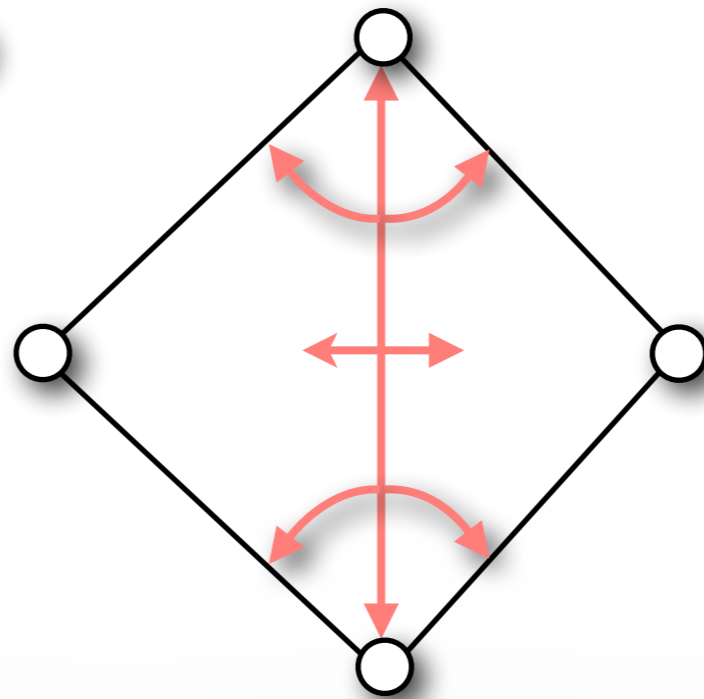
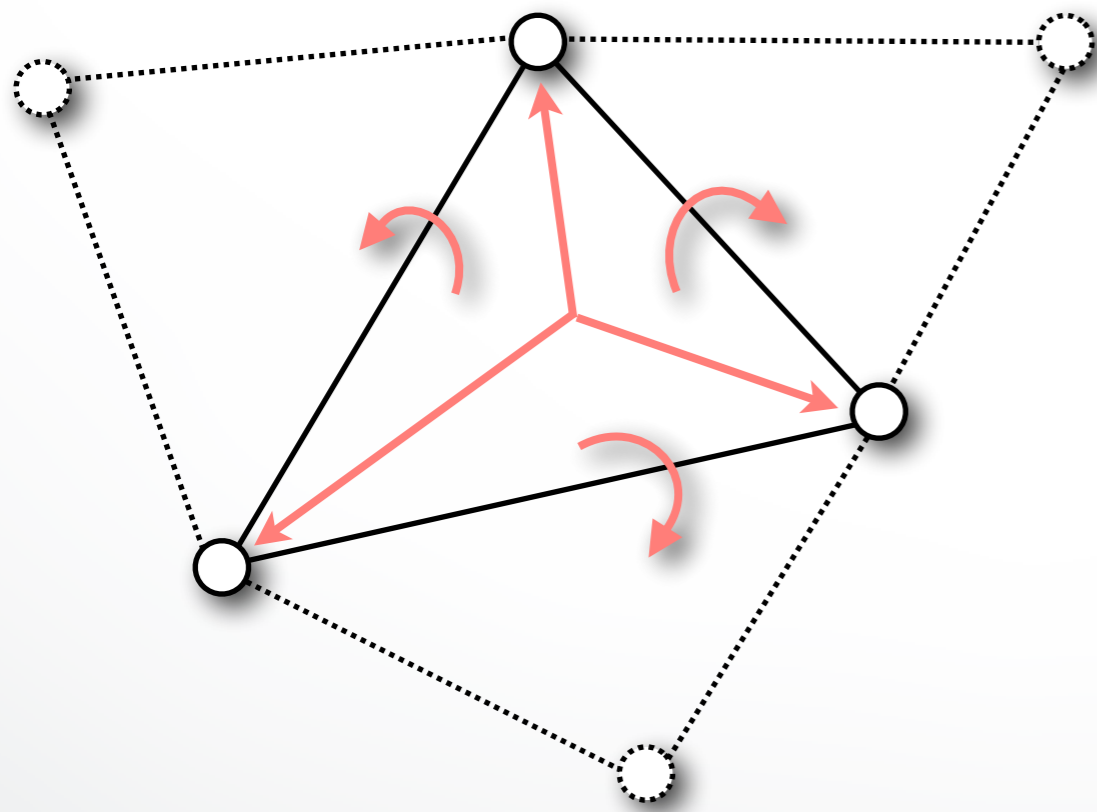
- Local Neighborhood is disk-shaped  $\mathbf{f}(D_\epsilon[u, v]) = D_\delta[\mathbf{f}(u, v)]$
- Guarantees meaningful neighbor enumeration
- Non-manifold



# Last Time

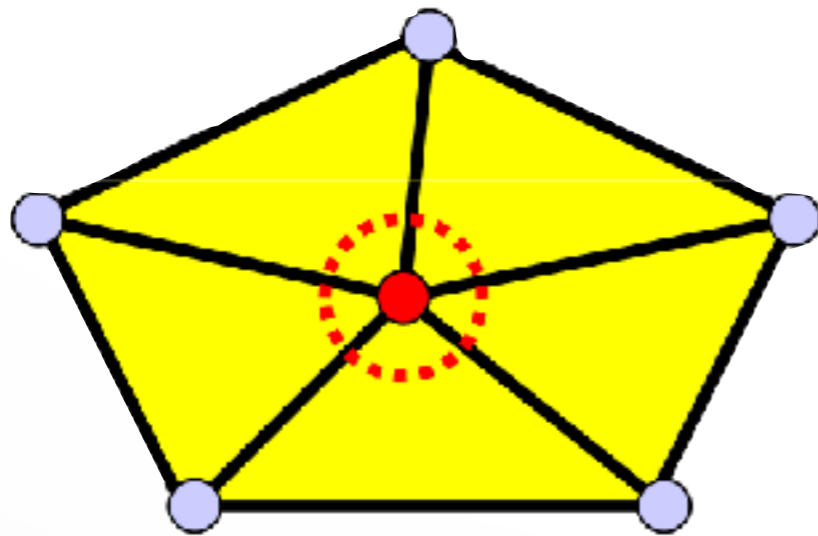
## Data Structures

- Face-Based
- Edge-Based, edges always have two faces
- Halfedge-Based

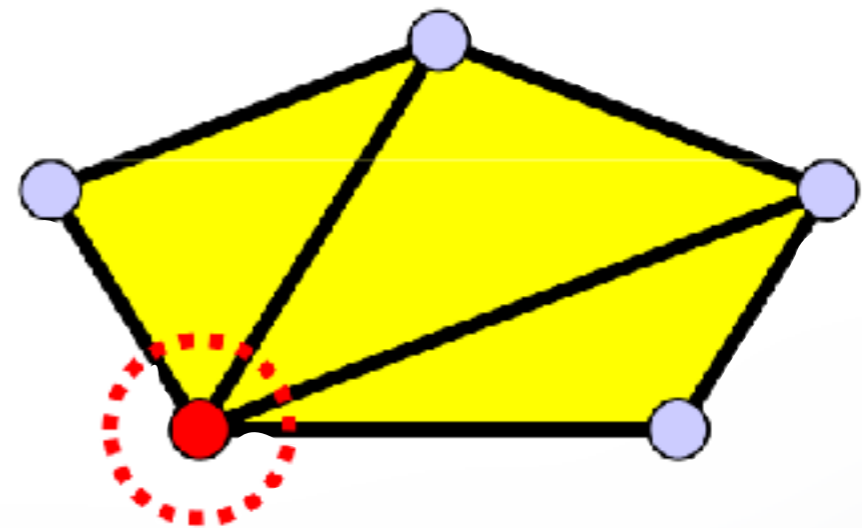


# When is a Triangle Mesh a Manifold?

- Every Edge incident to 1 or 2 Triangles
- Faces incident to a vertex form closed or open fan



closed fan



open fan



# Outline

- **Surface Representations**
- Explicit Surfaces
- Implicit Surfaces
- Conversion

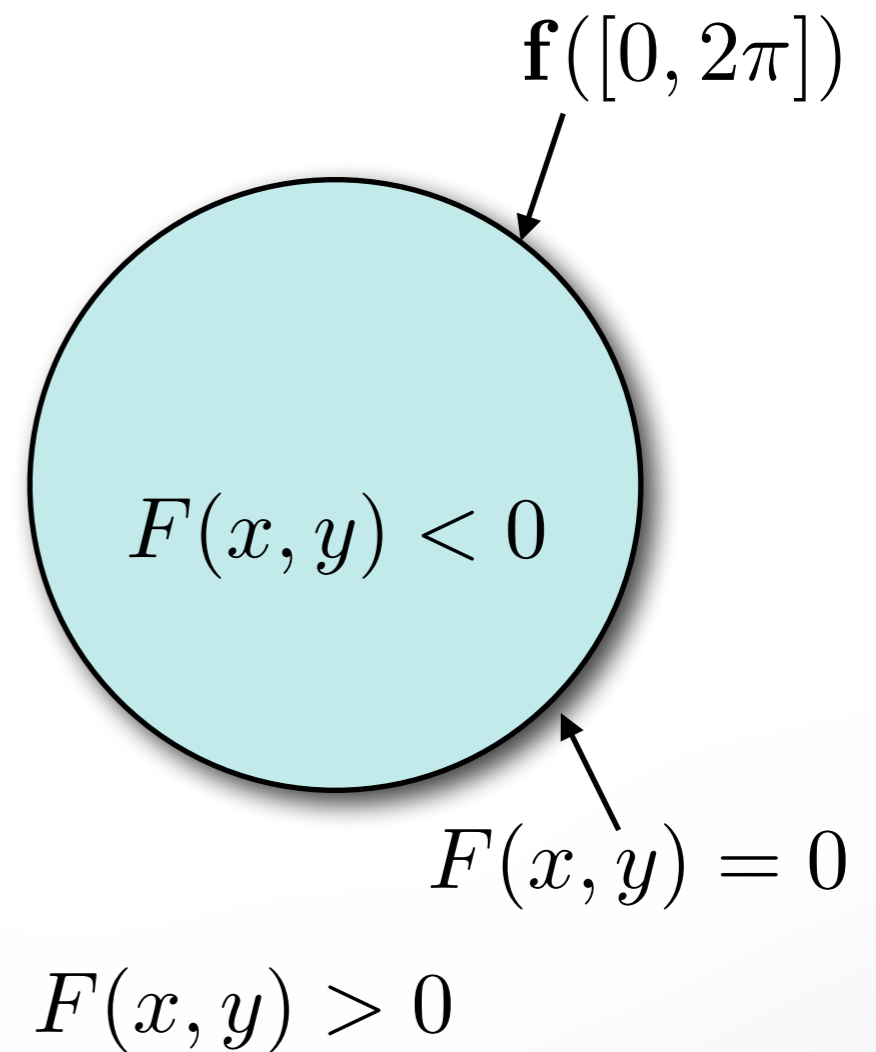
# Explicit vs. Implicit

**Explicit:**  $\mathbf{f}(x) = (r \cos(x), r \sin(x))^T$

- Range of parameterization function

**Implicit:**  $F(x, y) = \sqrt{x^2 + y^2} - r$

- Kernel of implicit function



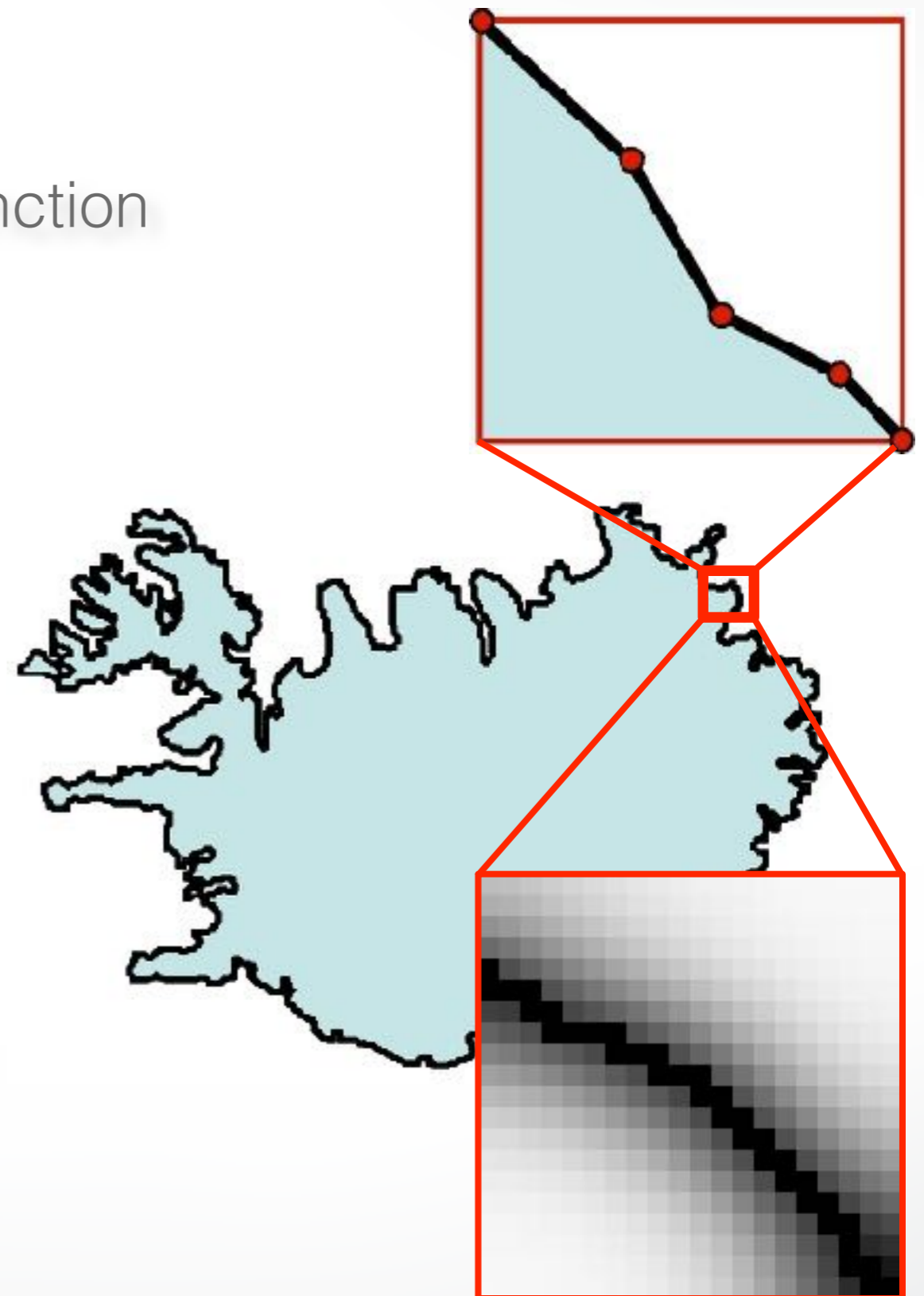
# Explicit vs. Implicit

**Explicit:**  $\mathbf{f}(x) = ?$

- Range of parameterization function
- Piecewise approximation

**Implicit:**  $F(x, y) = ?$

- Kernel of implicit function
- Piecewise approximation



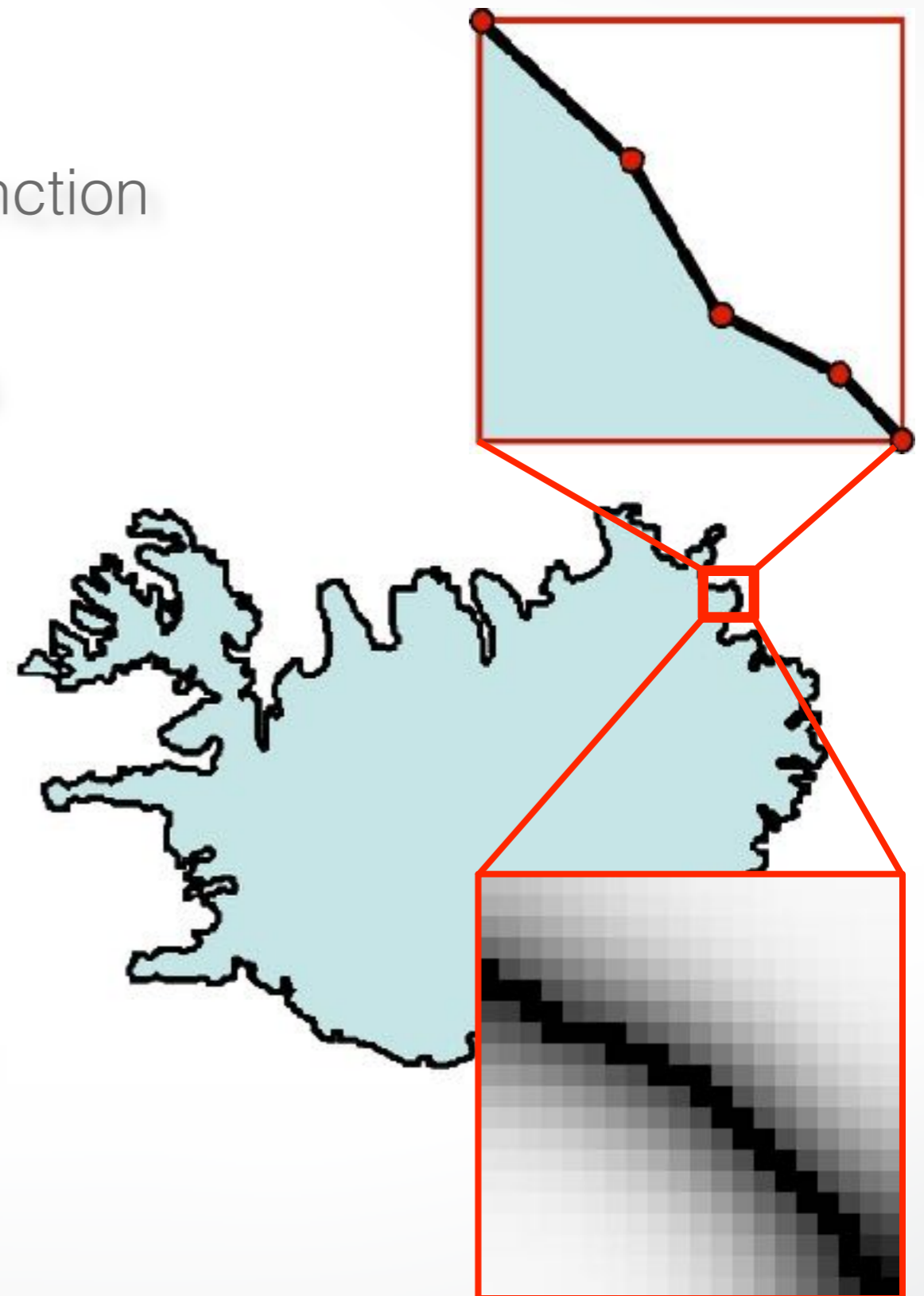
# Explicit vs. Implicit

## Explicit:

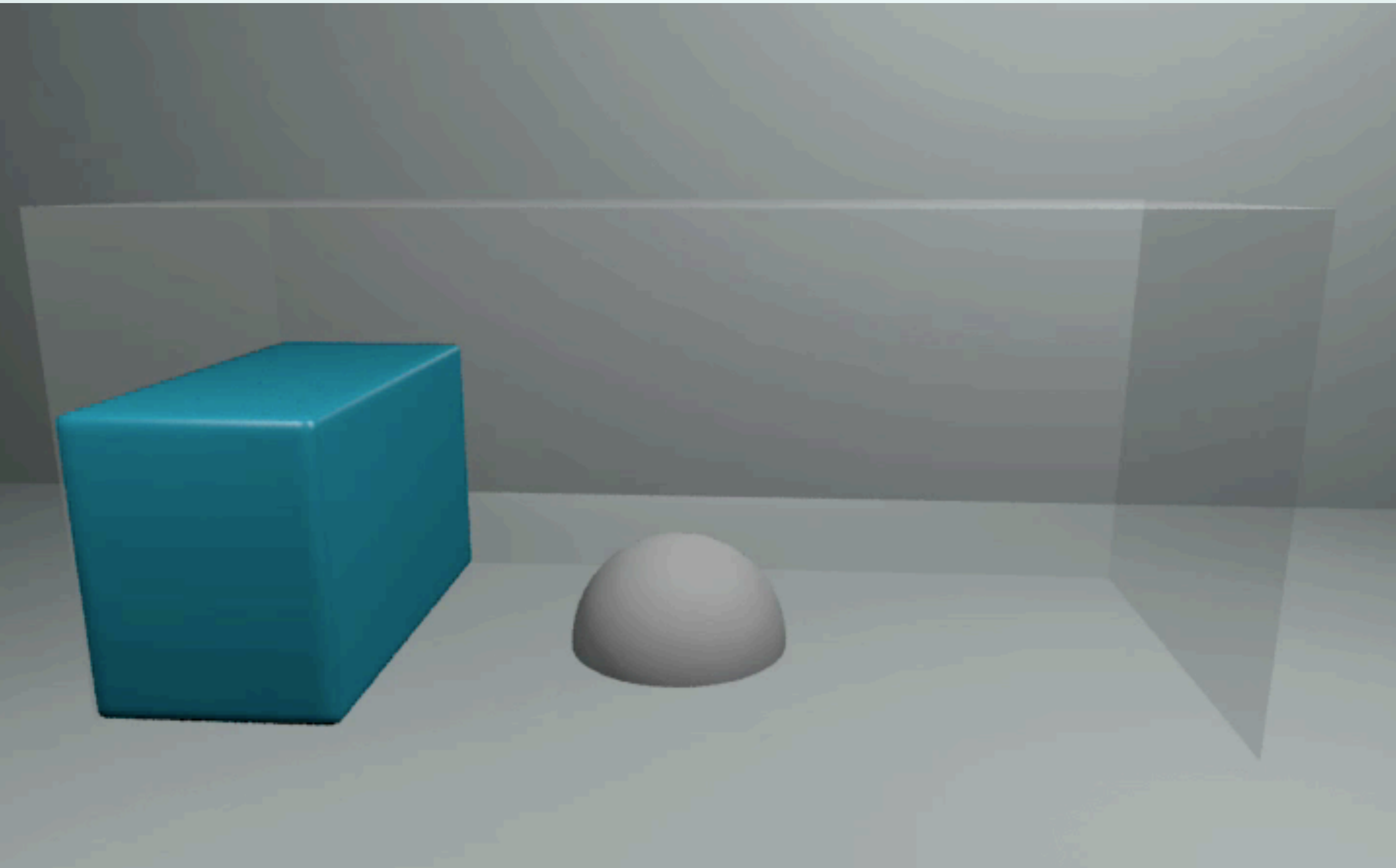
- Range of parameterization function
- Piecewise approximation
- Splines, triangle mesh, points
- Easy enumeration
- Easy geometry modification

## Implicit:

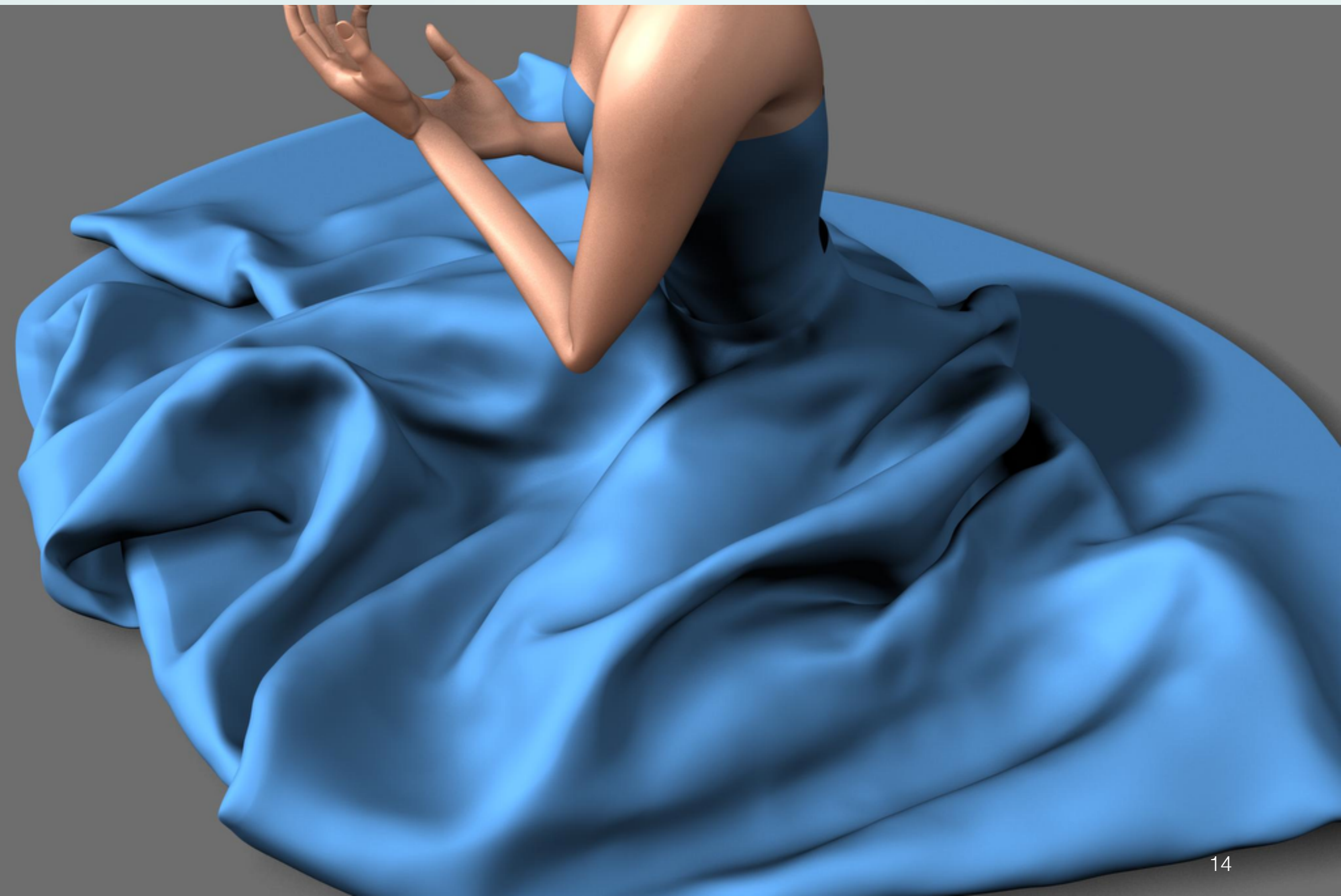
- Kernel of implicit function
- Piecewise approximation
- Scalar-valued 3D grid
- Easy in/out test
- Easy topology modification



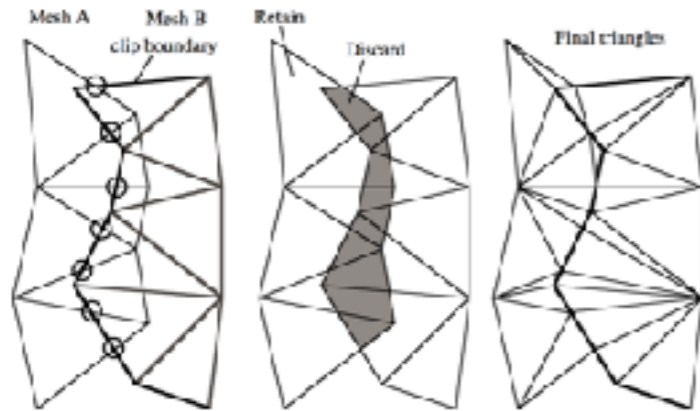
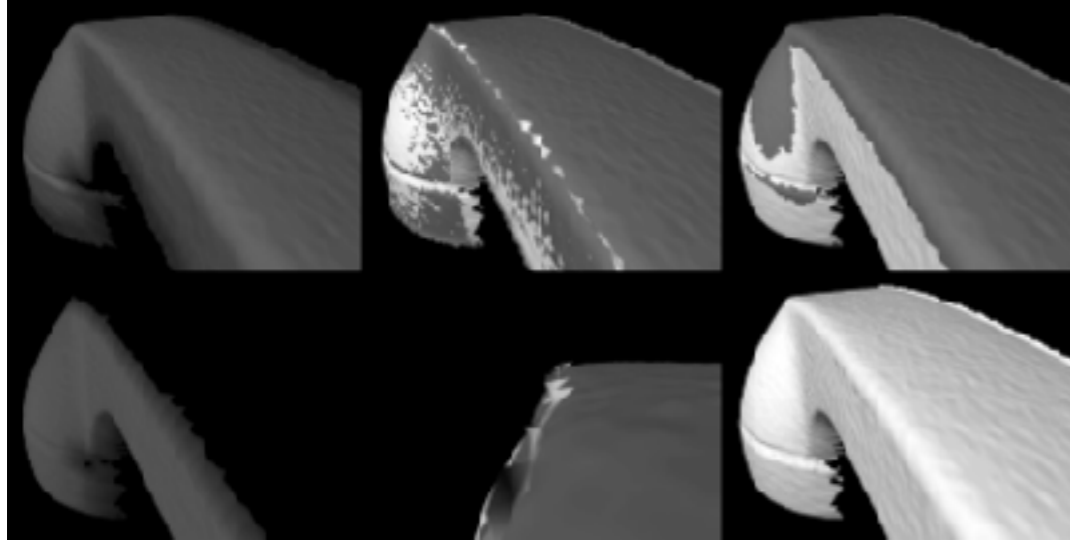
# Examples: Fluid Simulation



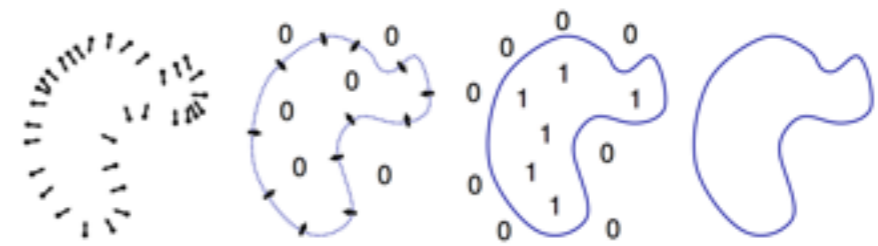
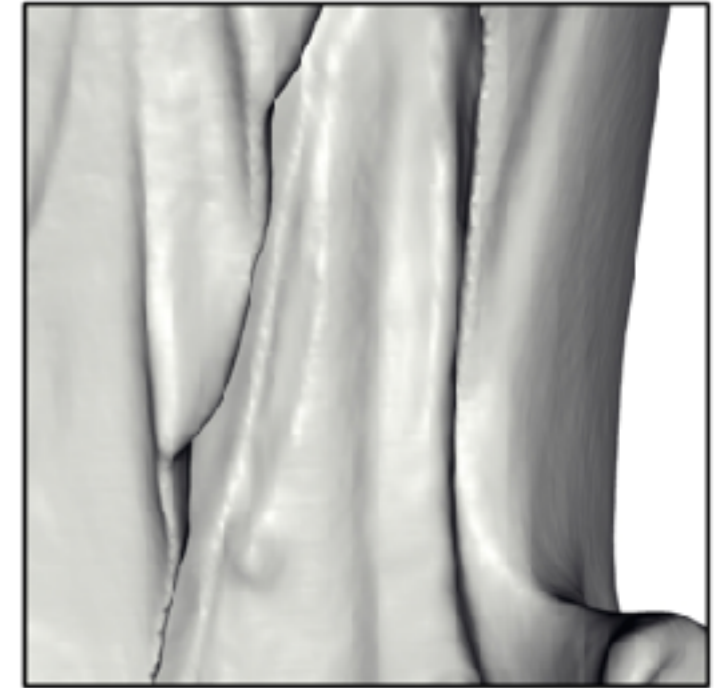
# Examples: Collisions



# Examples: 3D Reconstruction



Zippering



Poisson Reconstruction

# Examples: Kinect Fusion



1. Capture
2. Align
3. Fuse



<http://msdn.microsoft.com/en-us/library/dn188670.aspx>



# Outline

- Surface Representations
- **Explicit Surfaces**
- Implicit Surfaces
- Conversion

# Polynomial Approximation

**Polynomials are computable functions**

$$f(t) = \sum_{i=0}^p c_i t^i = \sum_{i=0}^p \tilde{c}_i \phi_i(t)$$

**Taylor expansion up to degree  $p$**

$$g(h) = \sum_{i=0}^p \frac{1}{i!} g^{(i)}(0) h^i + O(h^{p+1})$$

**Error for approximation  $g$  by polynomial  $f$**

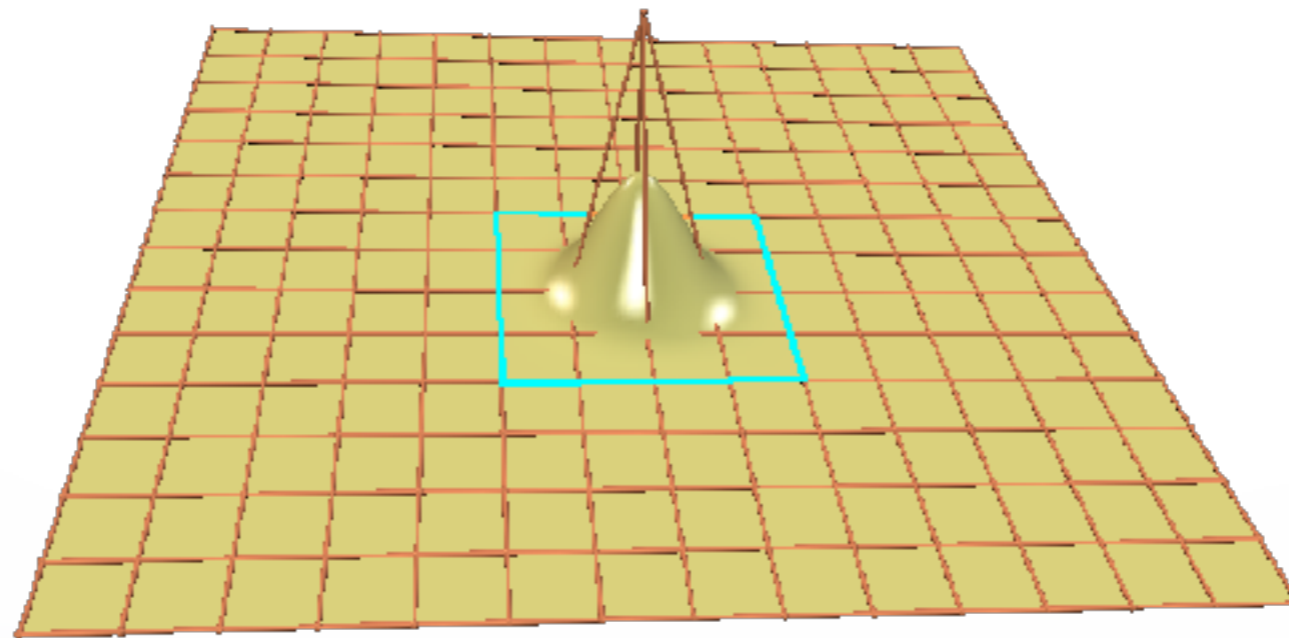
$$f(t_i) = g(t_i), \quad 0 \leq t_0 < \dots < t_p \leq h$$

$$|f(t) - g(t)| \leq \frac{1}{(p+1)!} \max f^{(p+1)} \prod_{i=0}^p (t - t_i) = O(h^{(p+1)})$$

# Spline Surfaces

## Piecewise polynomial approximation

$$\mathbf{f}(u, v) = \sum_{i=0}^n \sum_{j=0}^m \mathbf{c}_{ij} N_i^n(u) N_j^m(v)$$

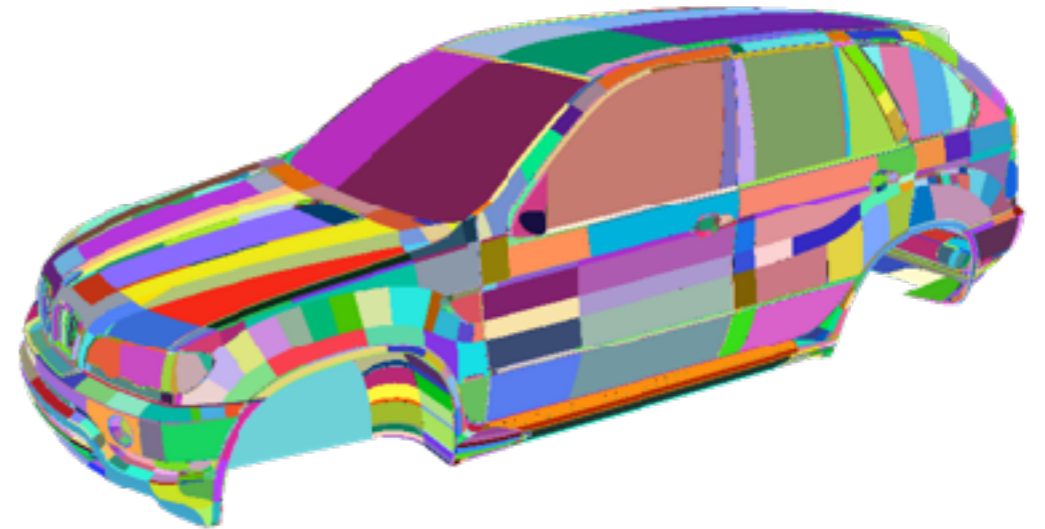


# Spline Surfaces

## Piecewise polynomial approximation

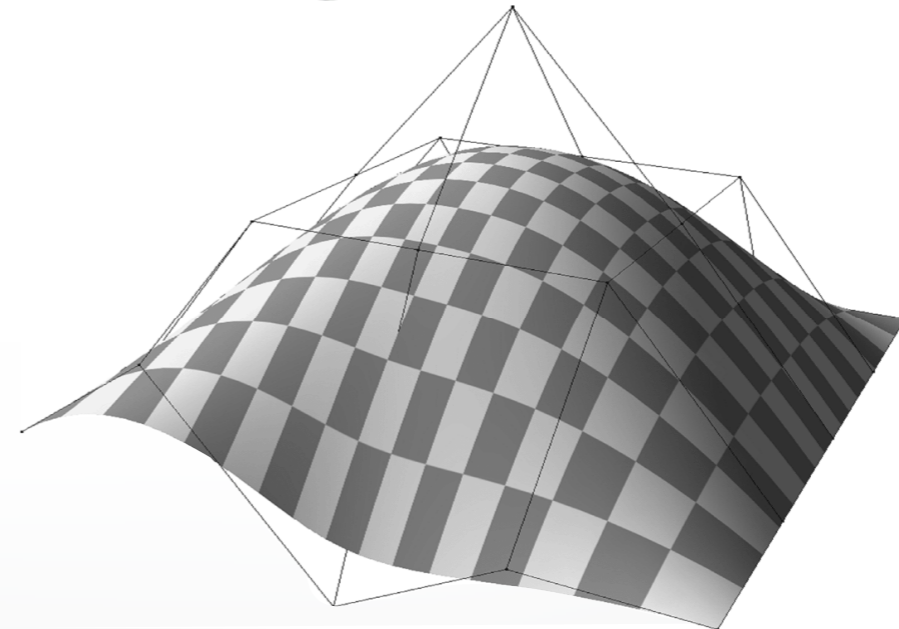
### Geometric constraints

- Large number of patches
- Continuity between patches
- Trimming



### Topological constraints

- Rectangular patches
- Regular control mesh



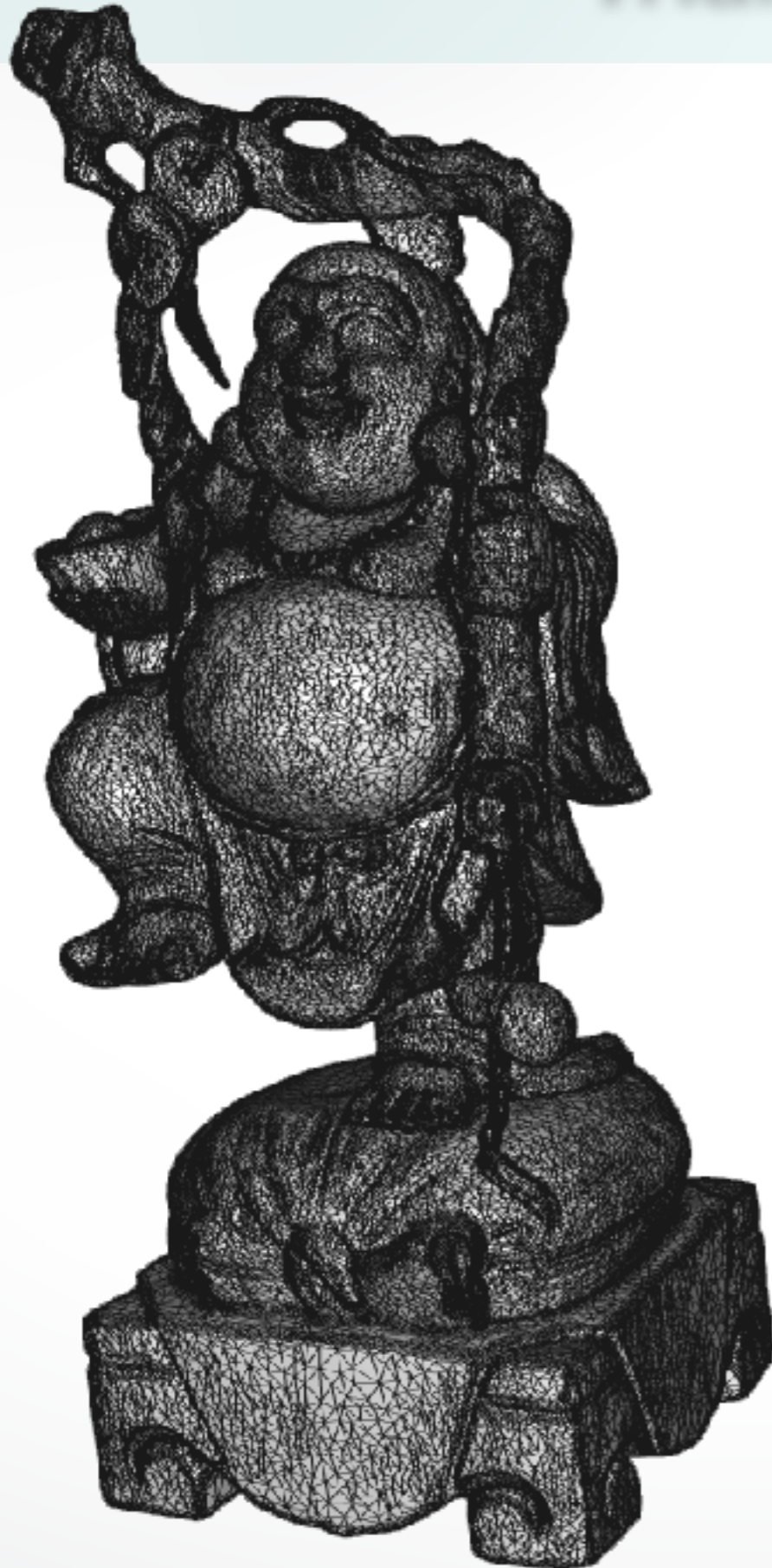
# Polygon Meshes

## Polygonal meshes are a good compromise

- Piecewise linear approximation  $\rightarrow$  error is  $O(h^2)$
- Error inversely proportional to #faces
- Arbitrary topology surfaces
- Piecewise smooth surfaces
- Adaptive sampling
- Efficient GPU-based rendering/processing



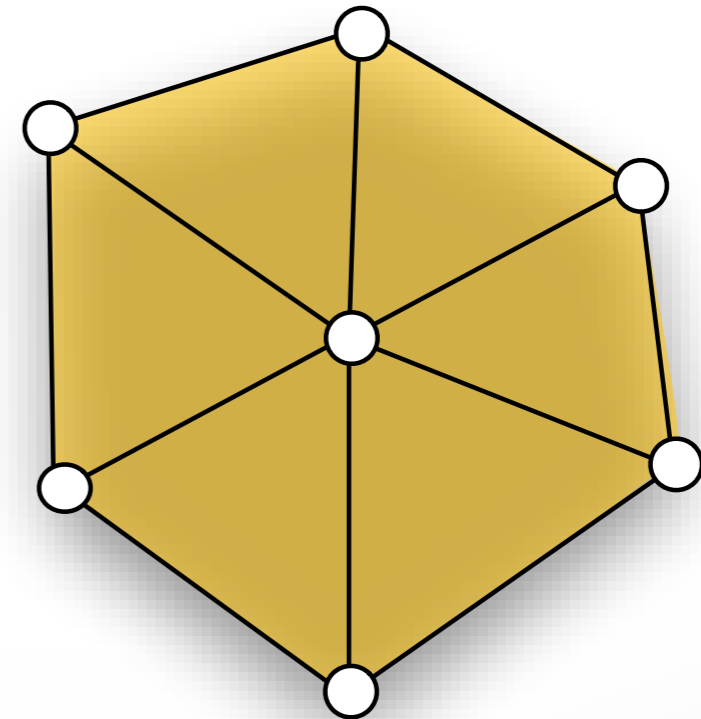
# Triangle Meshes



$$\mathcal{M} = (\{\mathbf{v}_i\}, \{e_j\}, \{f_k\})$$

**geometry**  $\mathbf{v}_i \in \mathbb{R}^3$

**topology**  $e_i, f_i \subset \mathbb{R}^3$

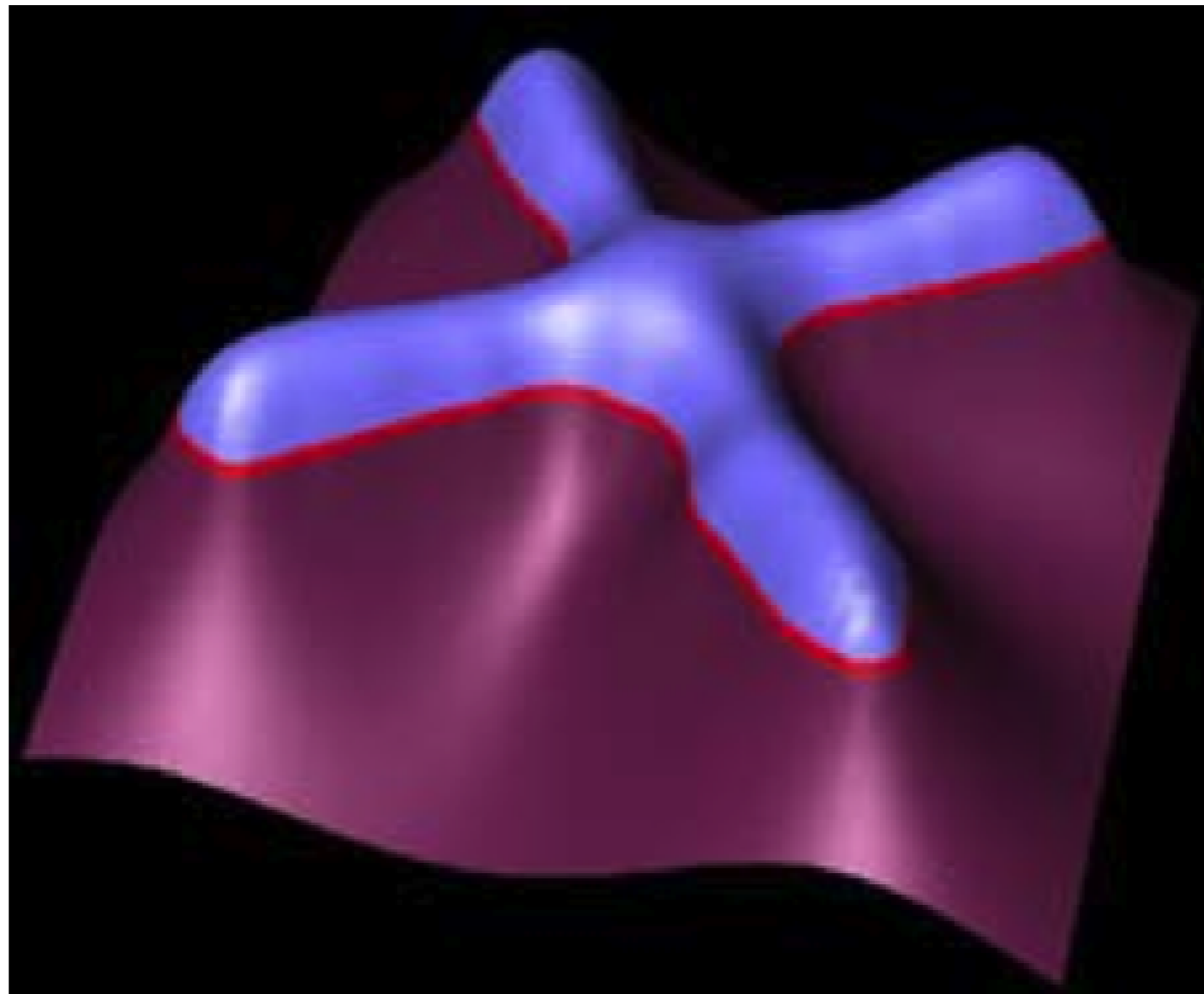


# Outline

- Surface Representations
- Explicit Surfaces
- **Implicit Surfaces**
- Conversion

# Implicit Representations

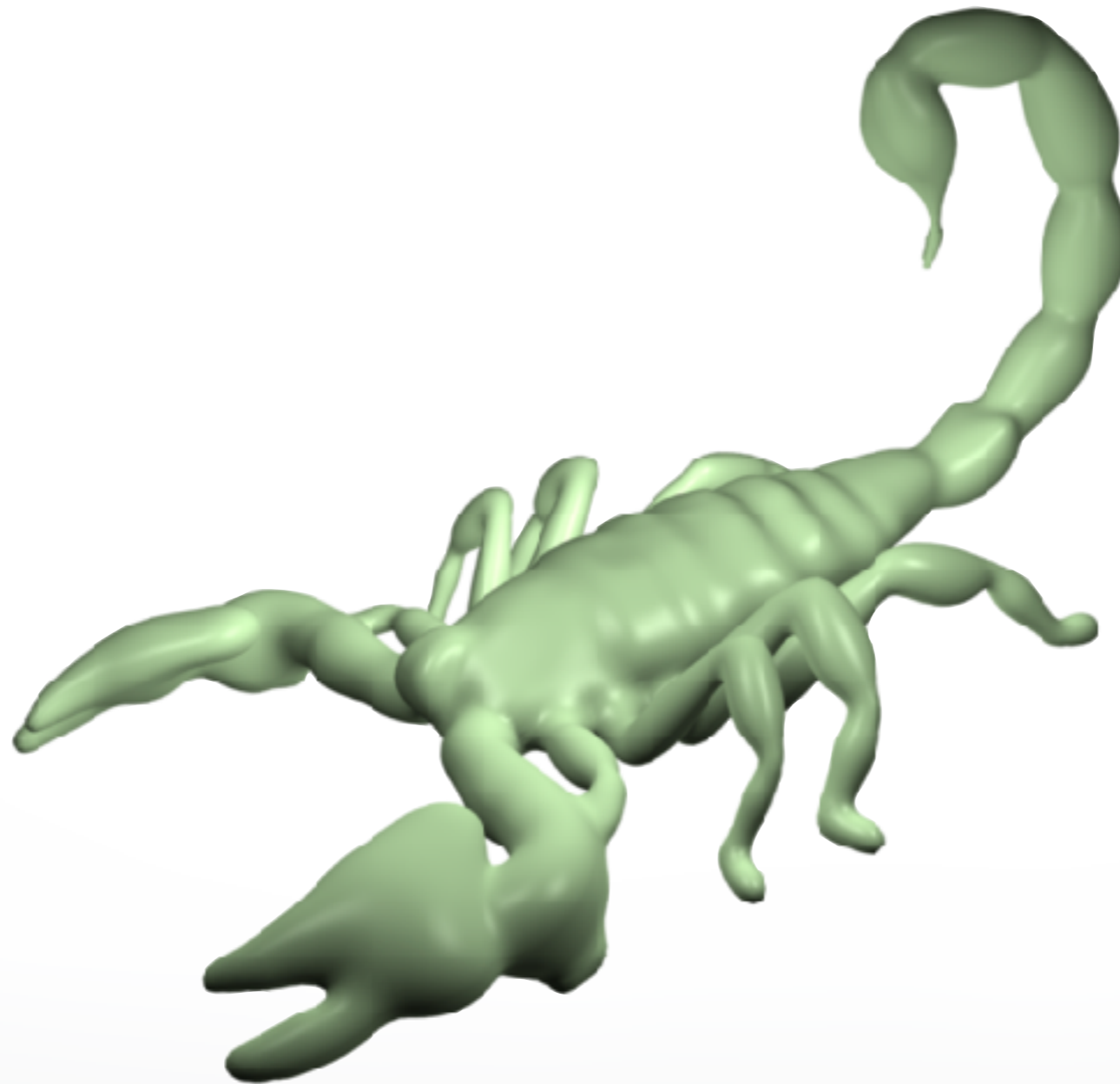
Level set of 2D function defines 1D curve





# Implicit Representations

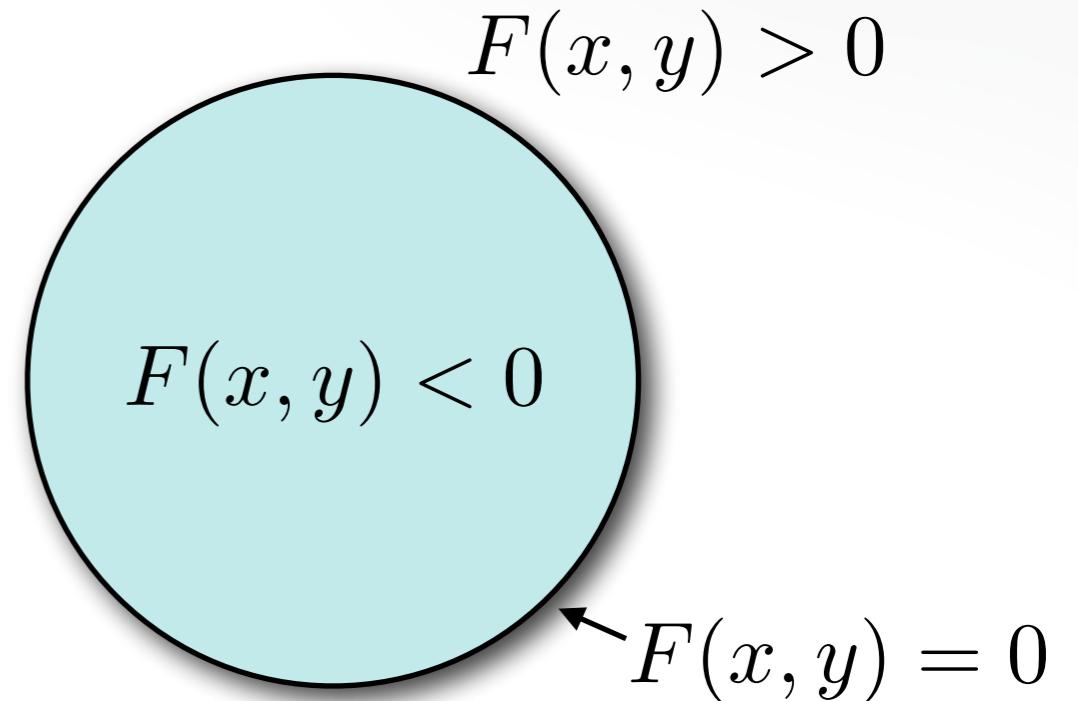
Level set of 3D function defines 2D surface



# Implicit Representations

## General implicit function:

- Interior:  $F(x, y, z) < 0$
- Exterior:  $F(x, y, z) > 0$
- Surface:  $F(x, y, z) = 0$



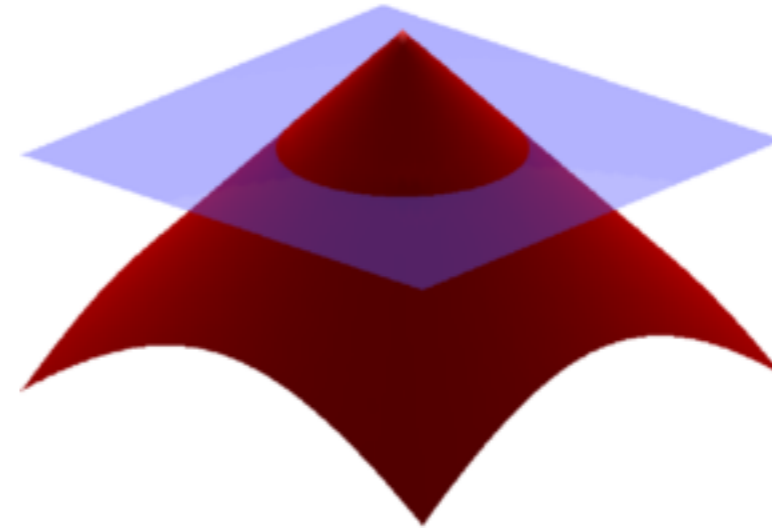
**Gradient  $\nabla F$  is orthogonal to level set**

## Special case

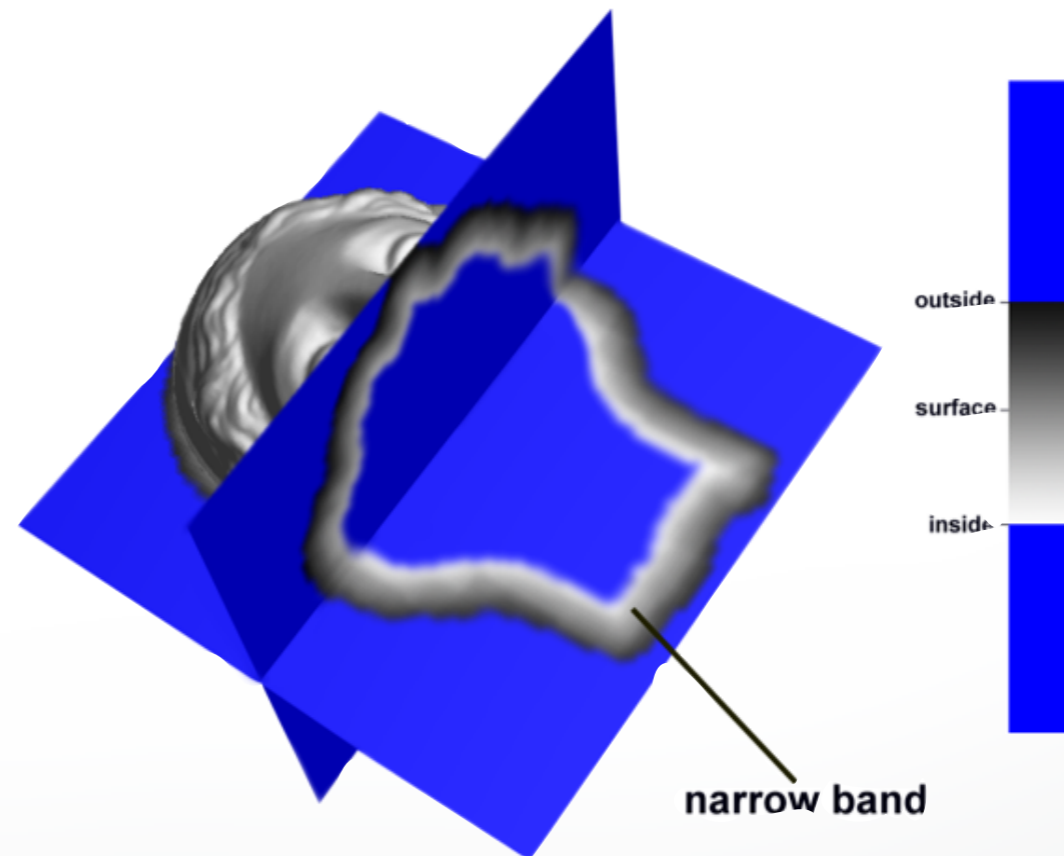
- Signed distance function (SDF)
- Gradient  $\nabla F$  is unit surface normal

# Signed Distance Function

SDF of a circle?



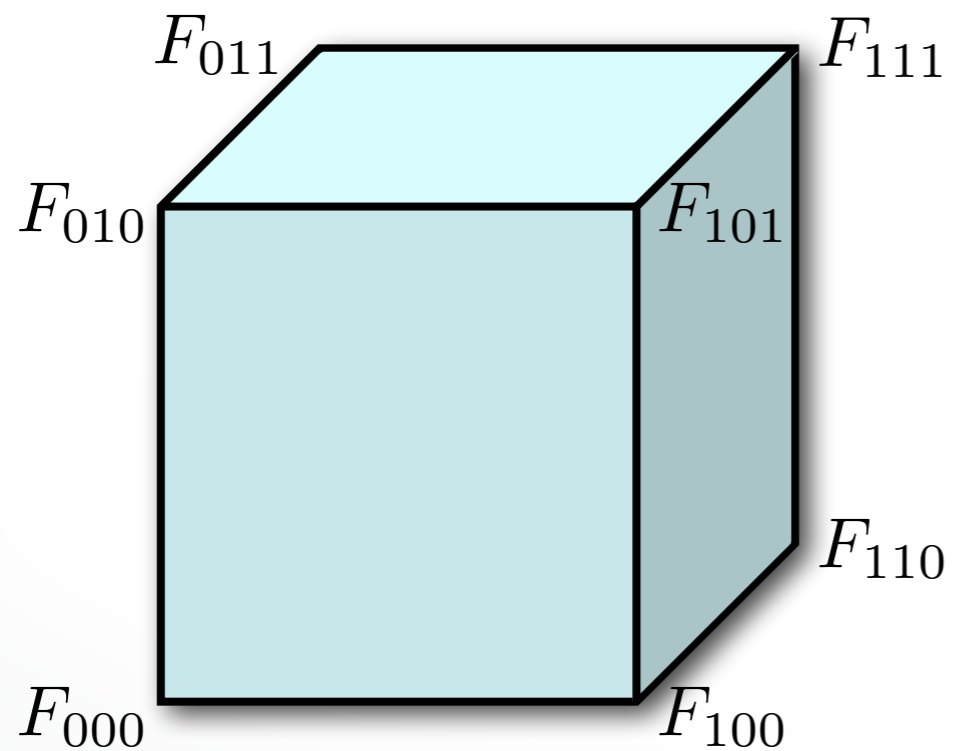
General shapes



# SDF Discretization

## Regular cartesian 3D grid

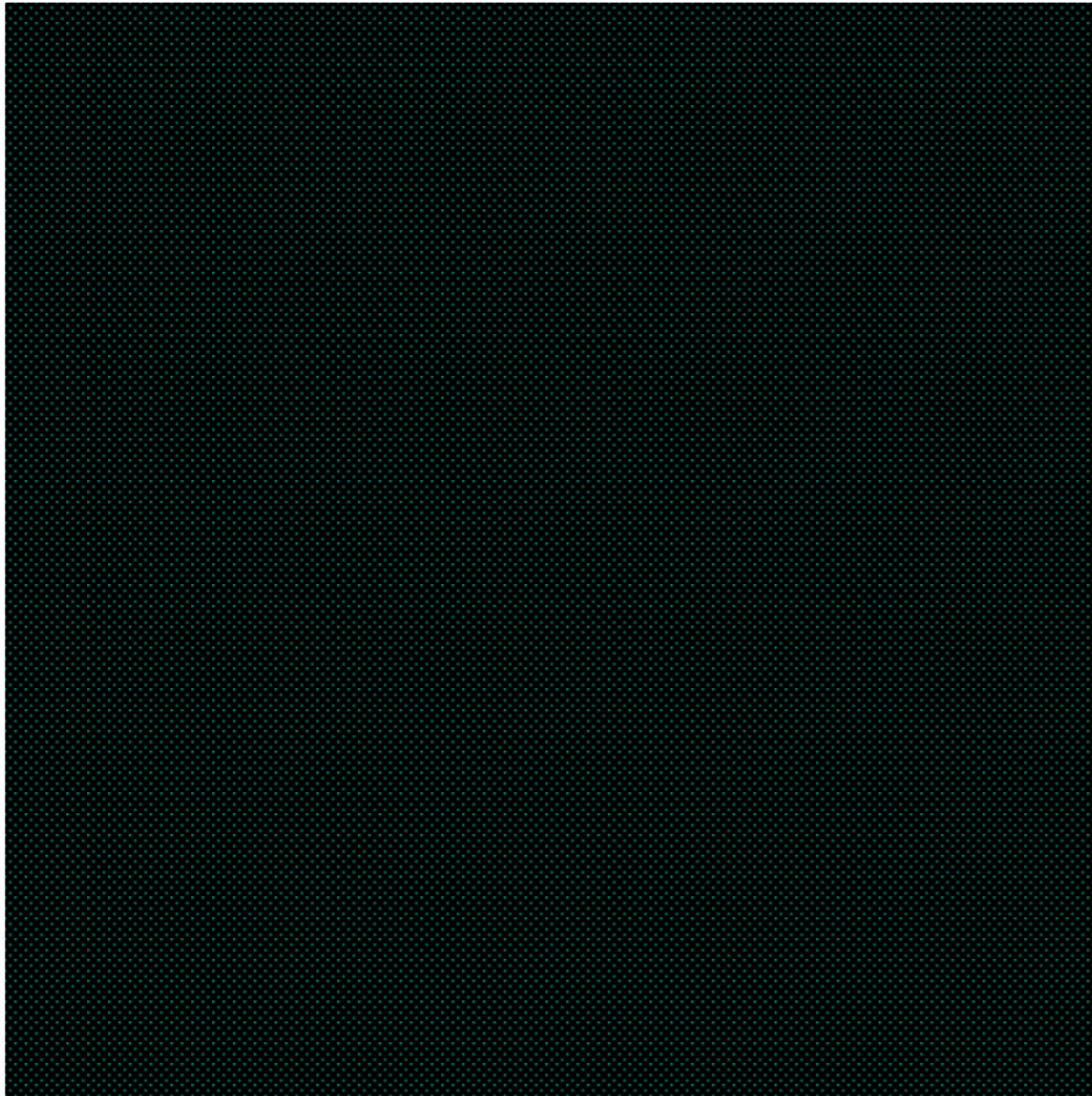
- Compute signed distance at nodes
- Tri-linear interpolation within cells



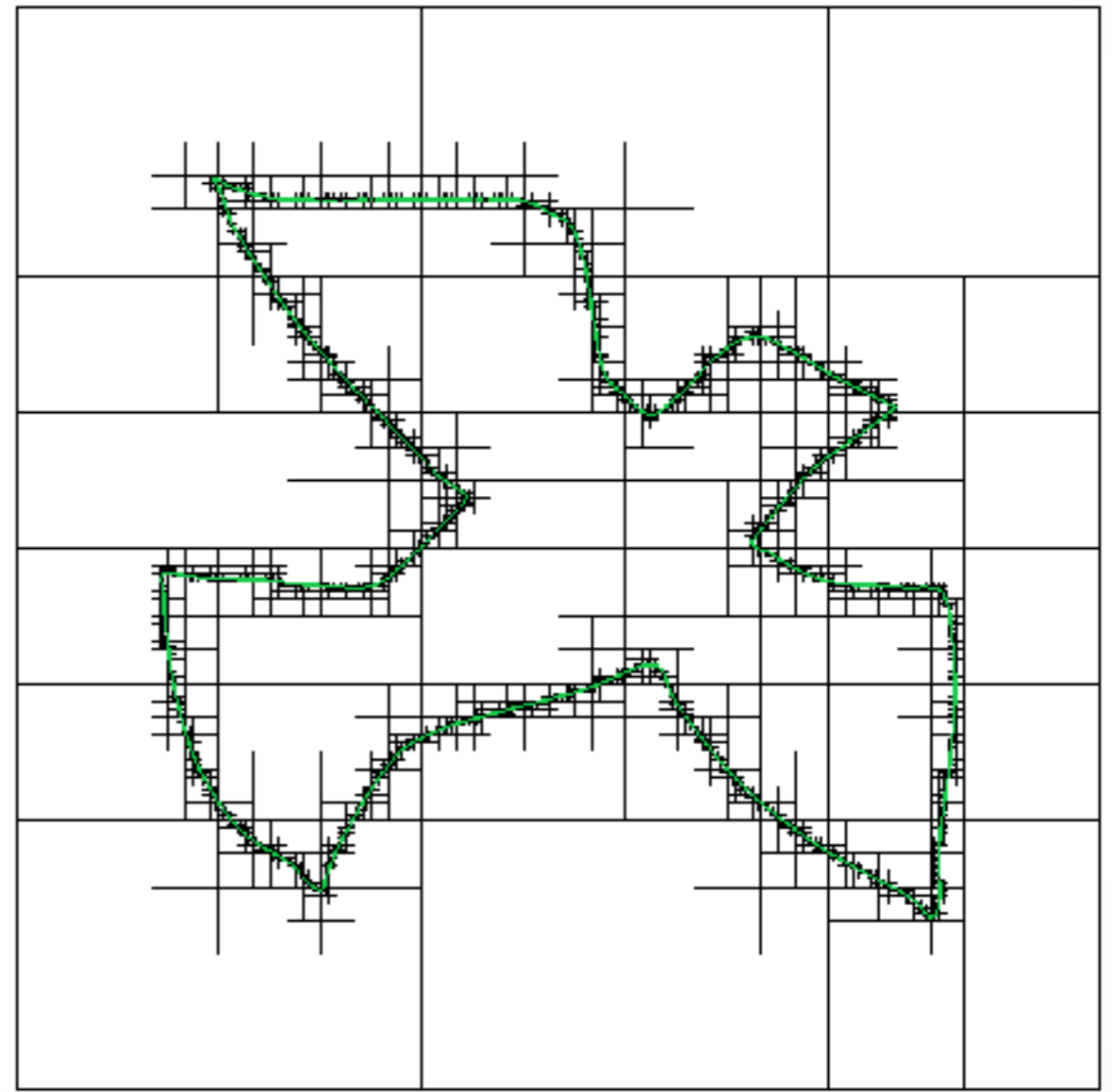
$$\begin{array}{l} F_{000} \\ F_{100} \\ F_{010} \\ F_{001} \\ \vdots \\ F_{111} \end{array} \begin{array}{llll} (1-u) & (1-v) & (1-w) & + \\ & u & (1-v) & (1-w) & + \\ (1-u) & & v & (1-w) & + \\ (1-u) & (1-v) & & w & + \\ & & & & \\ & & & & \\ & u & & v & & w \end{array}$$

# 3-Color Octree

3 Colors: **interior, exterior, boundary**

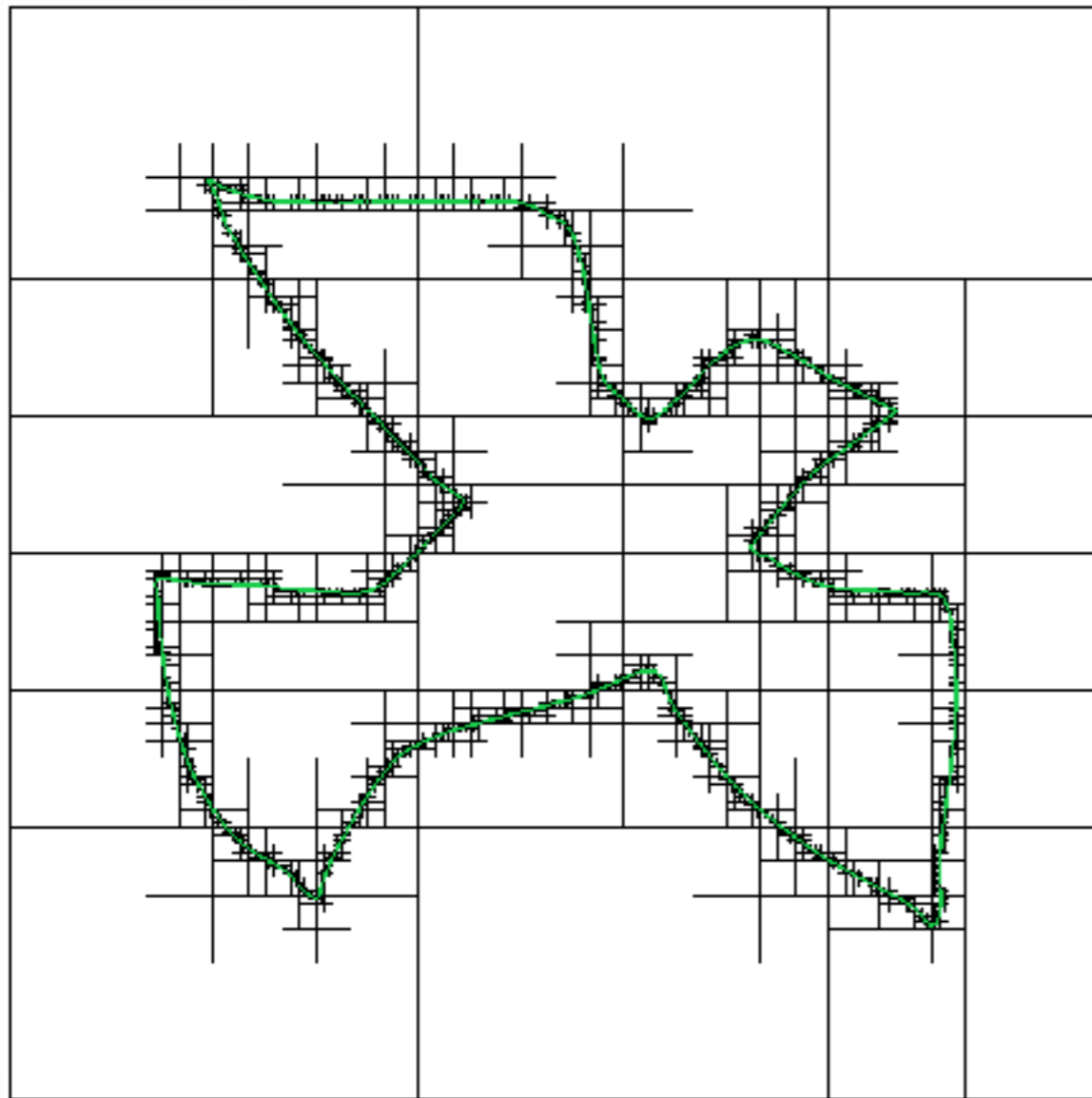


**1048576 cells**

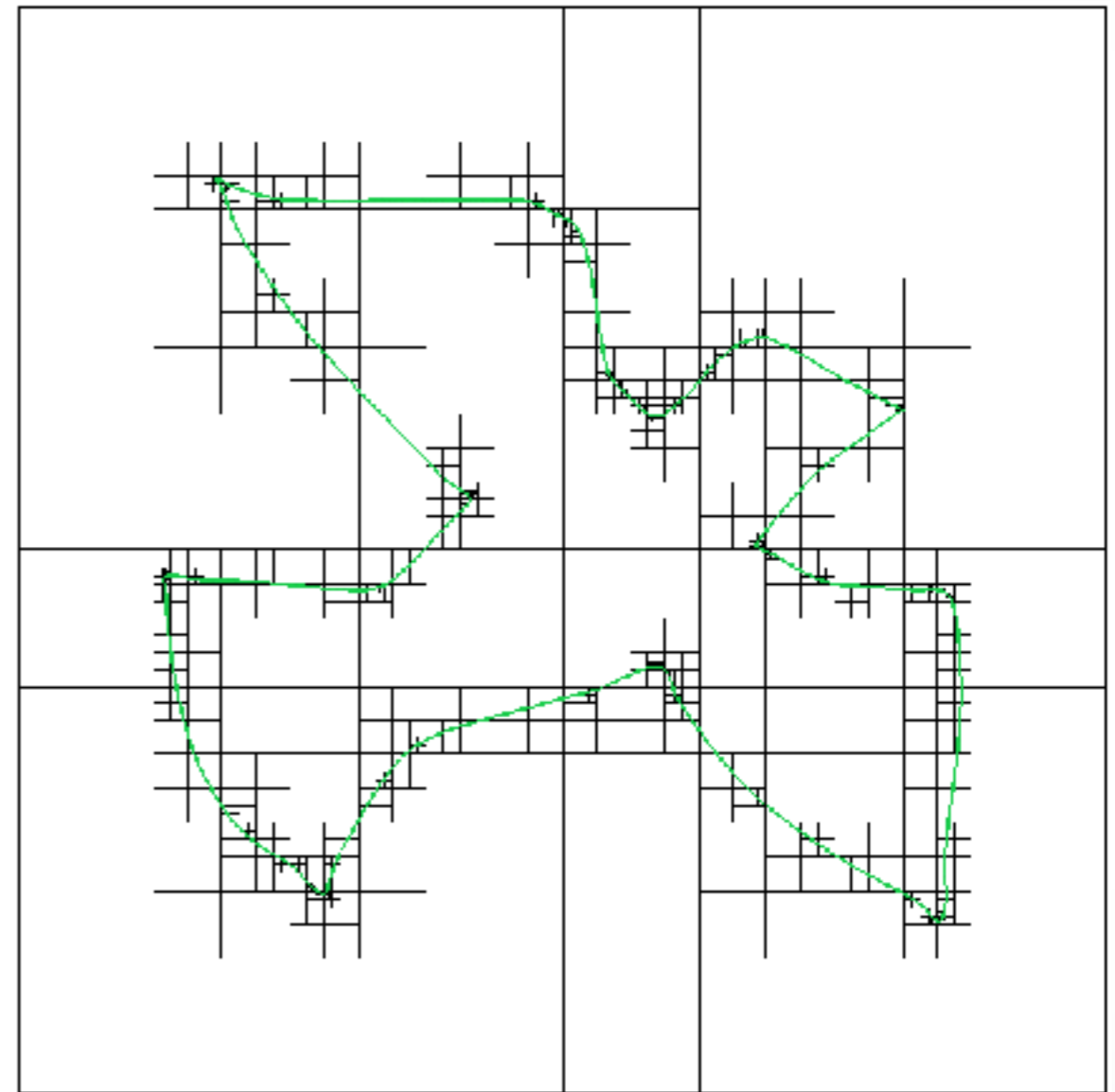


**12040 cells**

# Adaptively Sampled Distance Fields

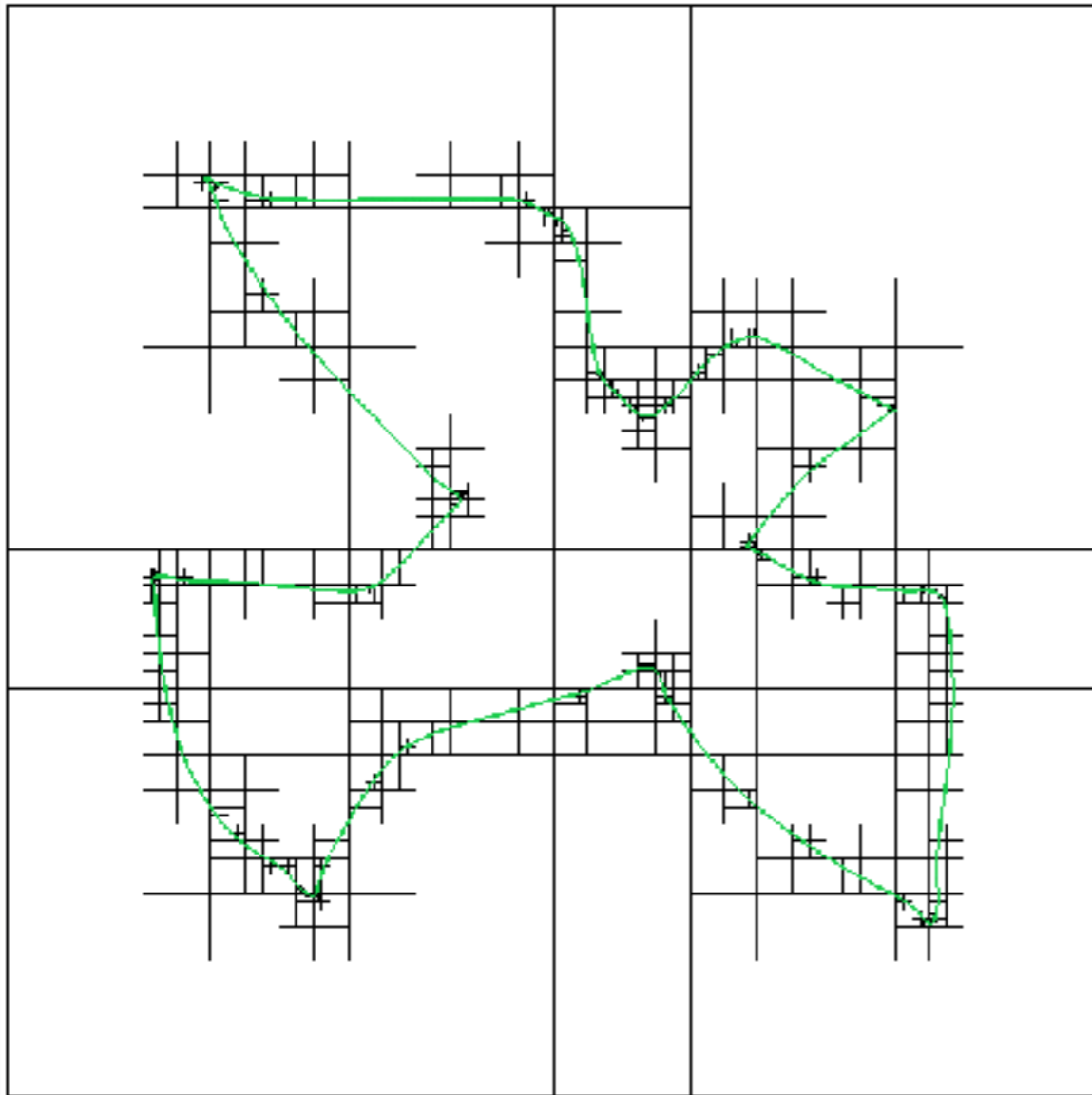


12040 cells

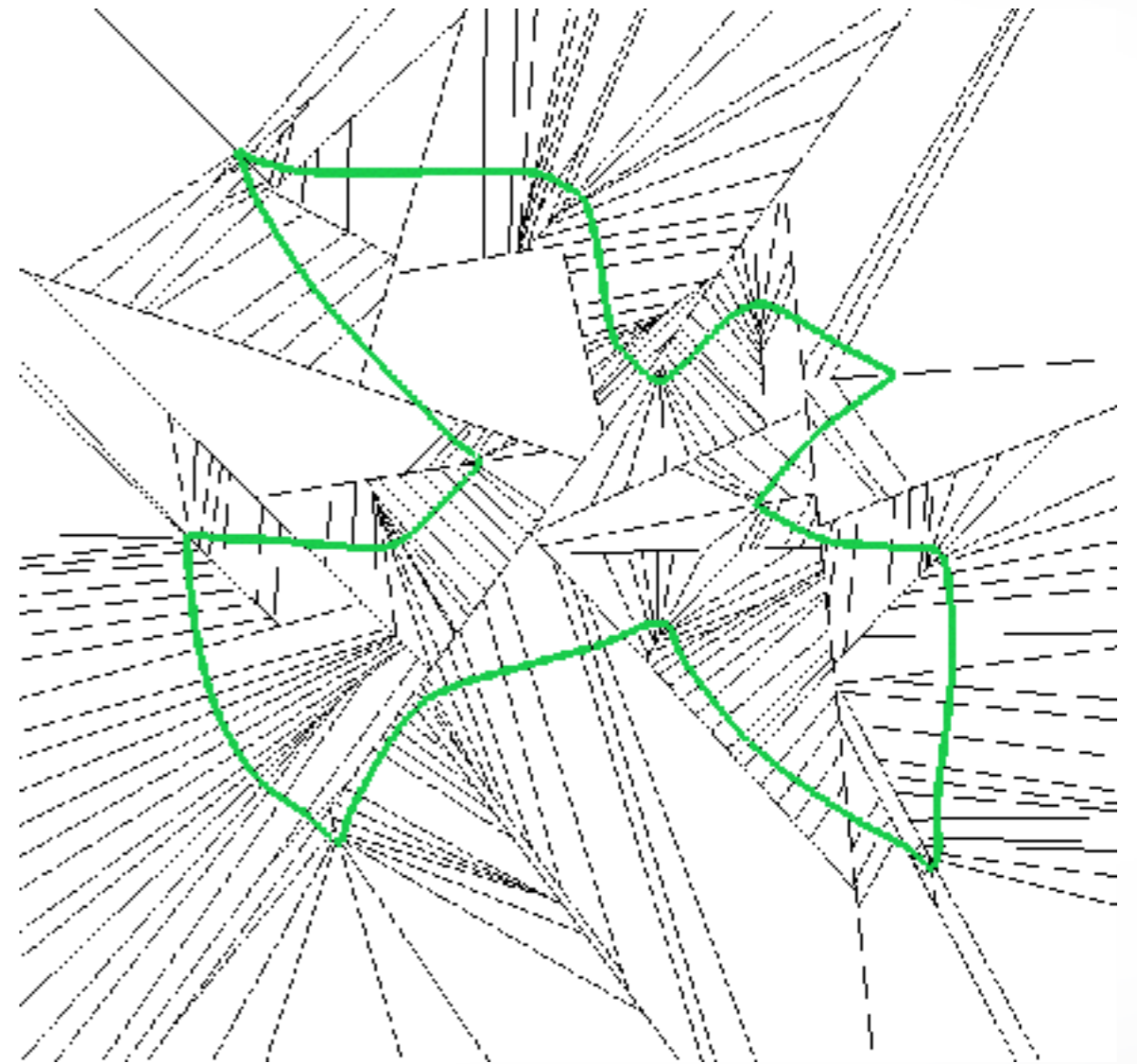


895 cells

# Binary Space Partitions



**895 cells**



**254 cells**

# Regularity vs. Complexity

## Implicit surface discretizations

- Uniform, regular voxel grids  $O(h^{-3})$
- Adaptive, 3-color octrees
  - Surface-adaptive refinement  $O(h^{-2})$
  - Feature-adaptive refinement  $O(h^{-1})$
- Irregular hierarchies
  - Binary space partition (BSP)  $O(h^{-1})$



# Literature

- Frisken et al., “Adaptively Sampled Distance Fields: A general representation of shape for computer graphics”, SIGGRAPH 2000
- Wu & Kobbel, “Piecewise Linear Approximation of Signed Distance Fields”, VMV 2003

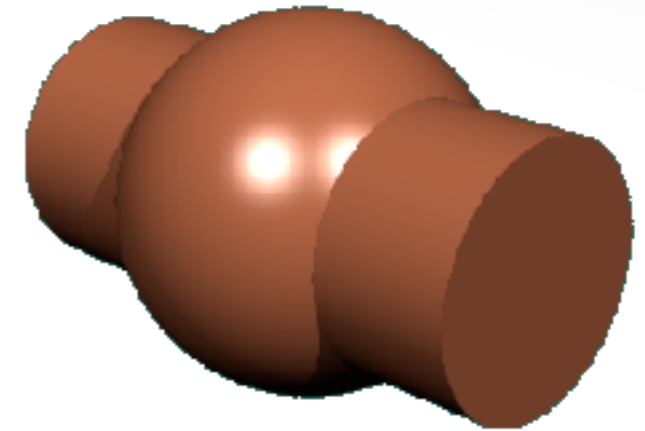
# Implicit Representations

- Natural representation for **volumetric data**: CT scans, density fields, etc.
- Advantageous when modeling shapes with **complex and/or changing topology** (e.g., fluids)
- Very suitable representation for **Constructive Solid Geometry (CSG)**

# CSG Example

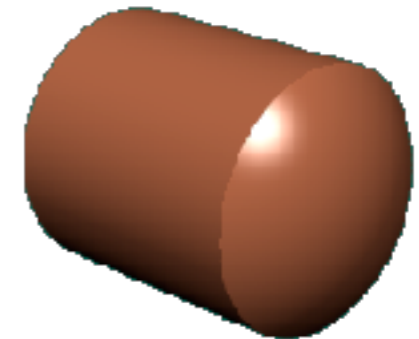
## Union

$$F_{C \cup S}(\cdot) = \min \{F_C(\cdot), F_S(\cdot)\}$$



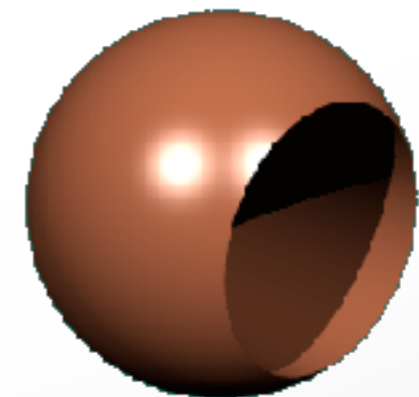
## Intersection

$$F_{C \cap S}(\cdot) = \max \{F_C(\cdot), F_S(\cdot)\}$$

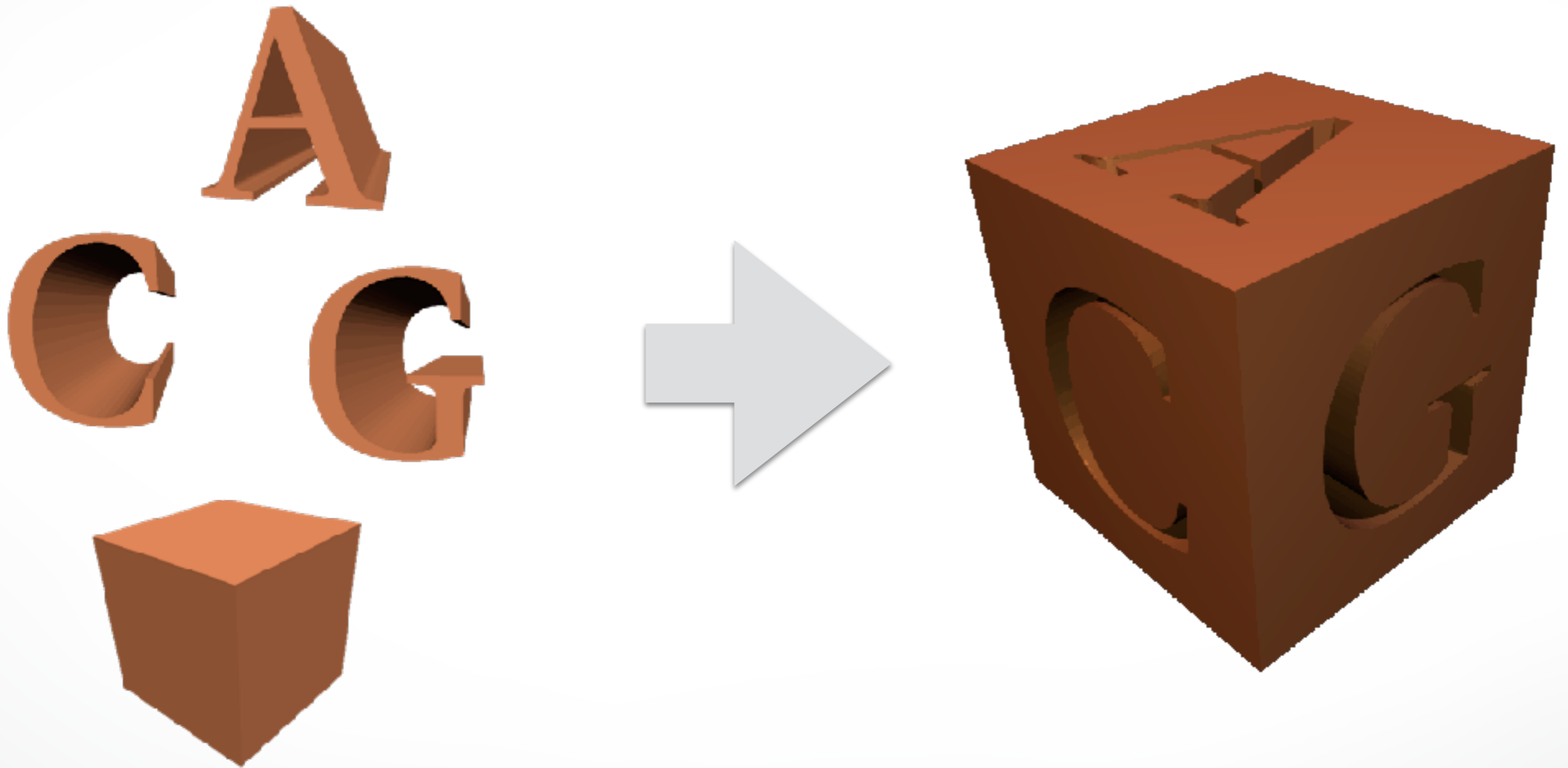


## Difference

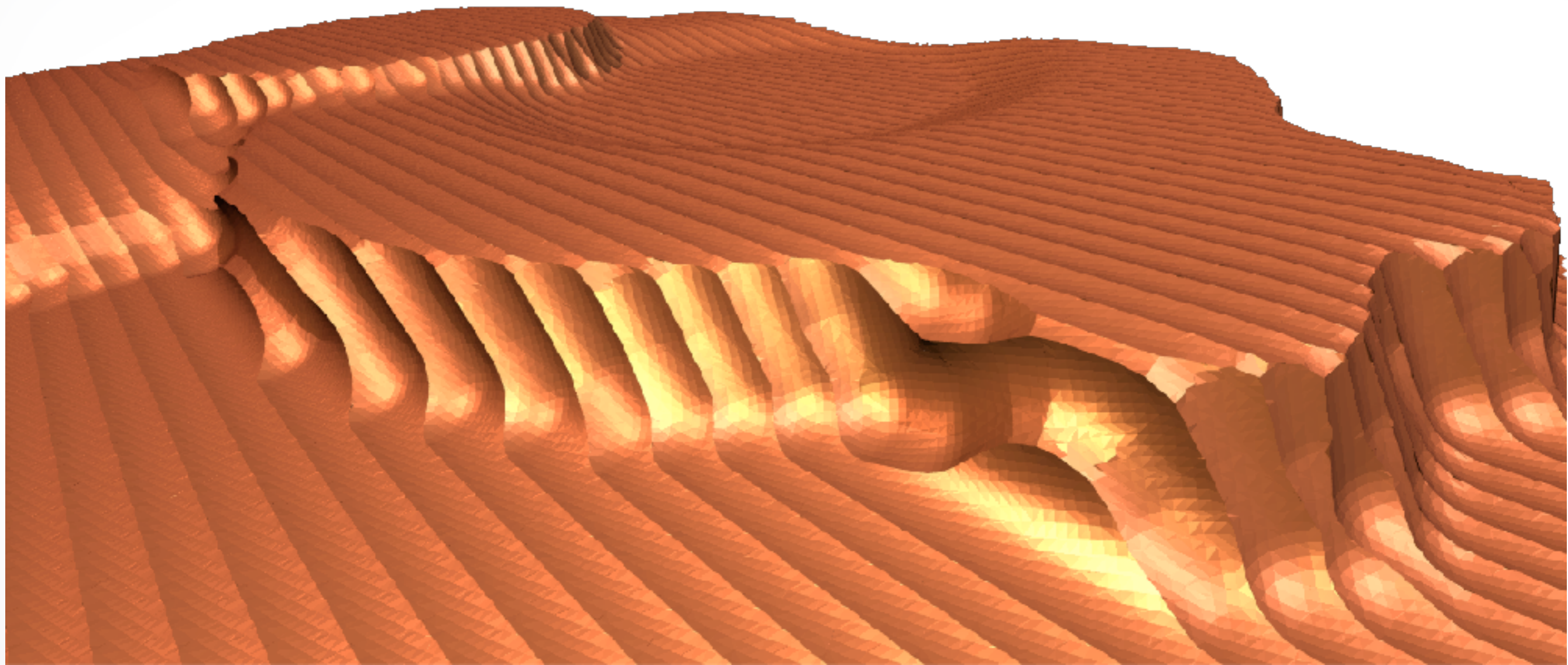
$$F_{S \setminus C}(\cdot) = \max \{-F_C(\cdot), F_S(\cdot)\}$$



# CSG Example



# CSG Example: Milling



# Outline

- Surface Representations
- Explicit Surfaces
- Implicit Surfaces
- **Conversion**

# Conversion

## Explicit to Implicit

- Compute signed distance at grid points
- Compute distance point-mesh
- Fast marching

## Implicit to Explicit

- Extract zero-level iso-surface  $F(x, y, z) = 0$
- Other iso-surfaces  $F(x, y, z) = C$
- Medical imaging, simulations, measurements, ...

# Signed Distance Computation

## Find closest mesh triangle

- Use spatial hierarchies (octree, BSP tree)

## Distance point-triangle

- Distance to plane, edge, or vertex
- <http://www.geometrictools.com>

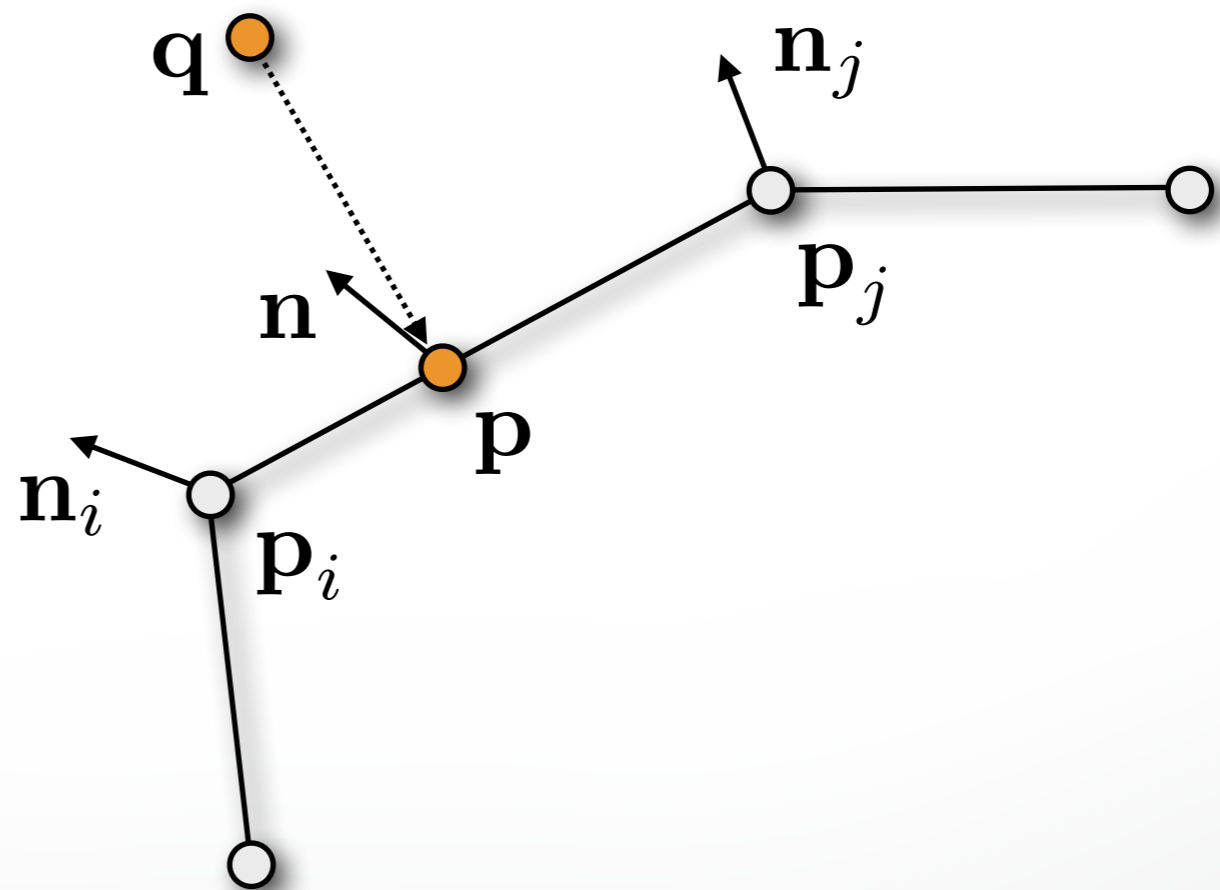
## Inside or outside?

- Based on interpolated surface normals



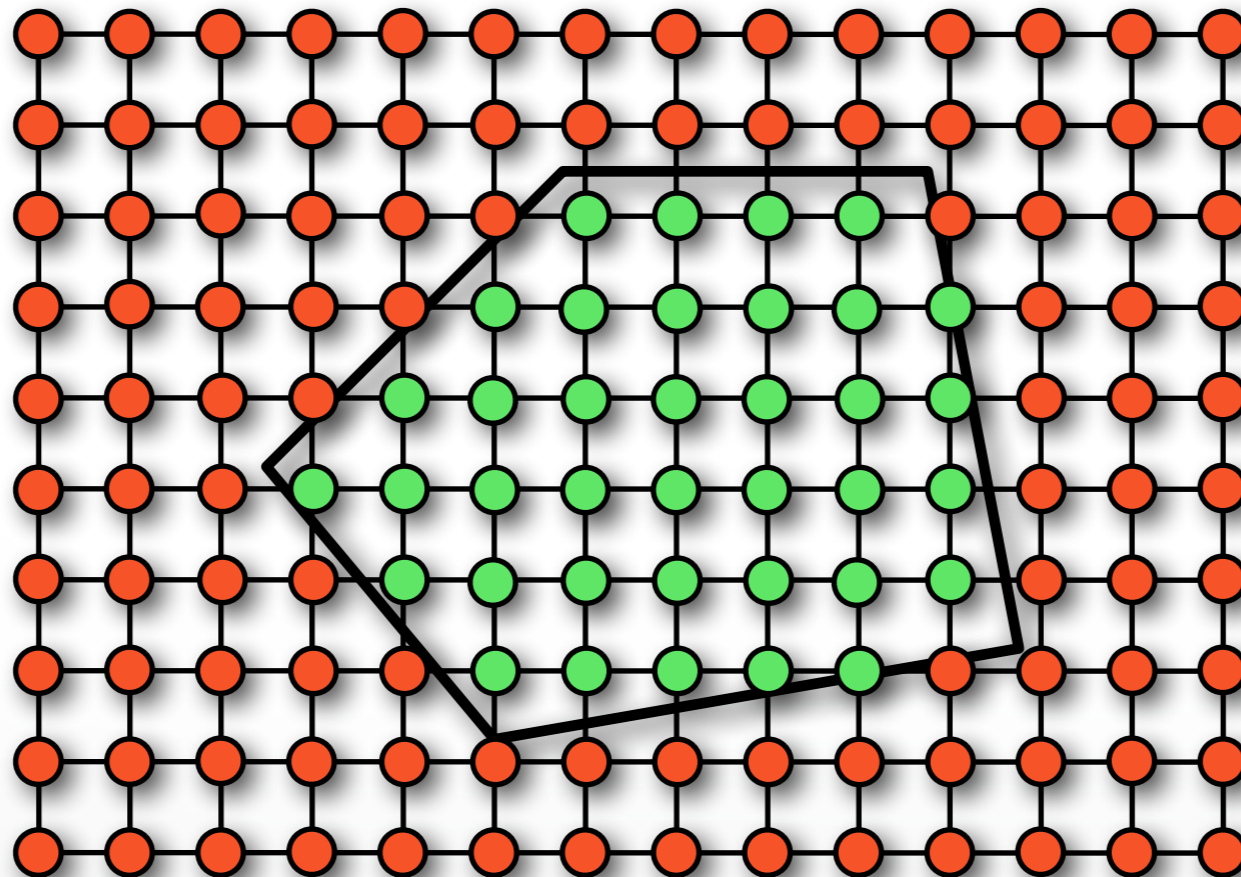
# Signed Distance Computation

- Closest point  $\mathbf{p} = \alpha\mathbf{p}_i + (1 - \alpha)\mathbf{p}_j$
- Interpolated normal  $\mathbf{n} = \alpha\mathbf{n}_i + (1 - \alpha)\mathbf{n}_j$
- Inside if  $(\mathbf{q} - \mathbf{p})^\top \mathbf{n} < 0$



# Fast Marching Techniques

- Initialize with exact distance in mesh's vicinity
- Fast-march outwards
- Fast-march inwards



# Literature

- Schneider, Eberly, “Geometric Tools for Computer Graphics”, Morgan Kaufmann, 2002
- Sethian, “Level Set and Fast Marching Methods”, Cambridge University Press, 1999

# Conversion

## Explicit to Implicit

- Compute signed distance at grid points
- Compute distance point-mesh
- Fast marching

## Implicit to Explicit

- Extract zero-level iso-surface  $F(x, y, z) = 0$
- Other iso-surfaces  $F(x, y, z) = C$
- Medical imaging, simulations, measurements, ...

# 2D: Marching Square

## 1. Classify grid nodes as inside/outside

- Is  $F(\mathbf{x}_{i,j}) > 0$  or  $< 0$  ?

## 2. Classify cell: $2^4$ configurations

- In/out for each corner

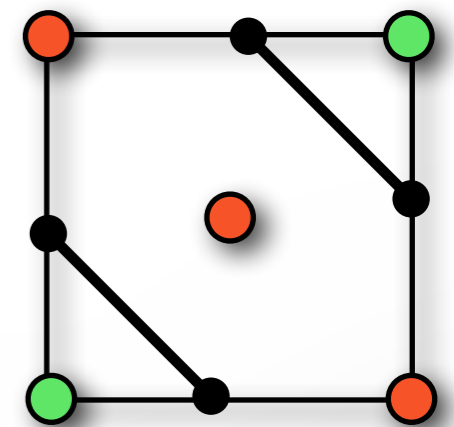
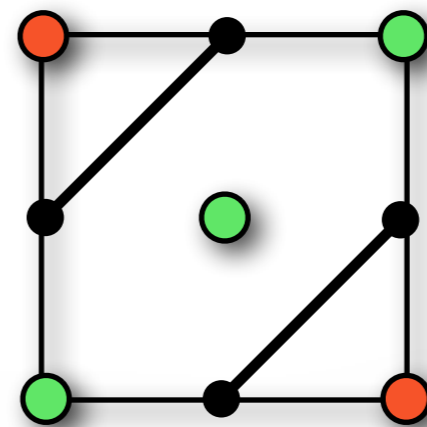
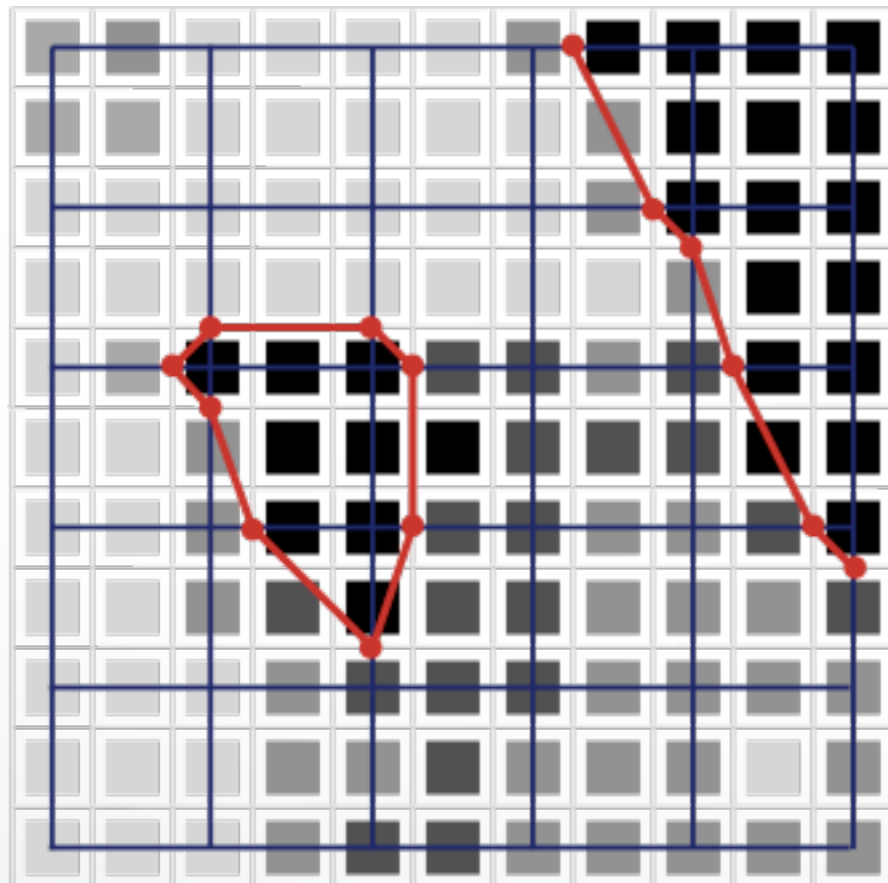
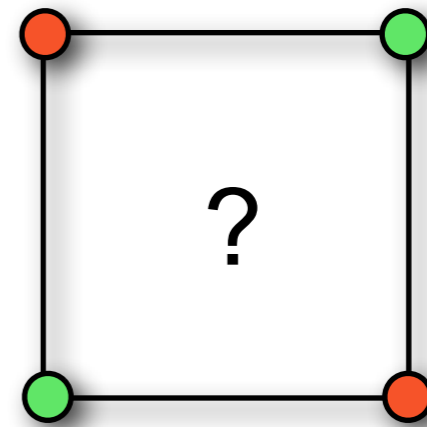
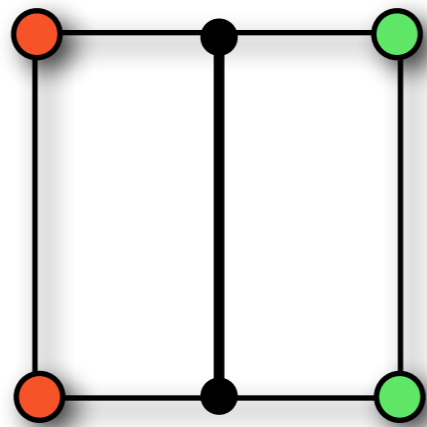
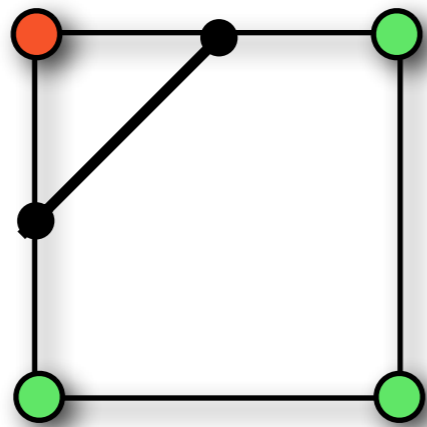
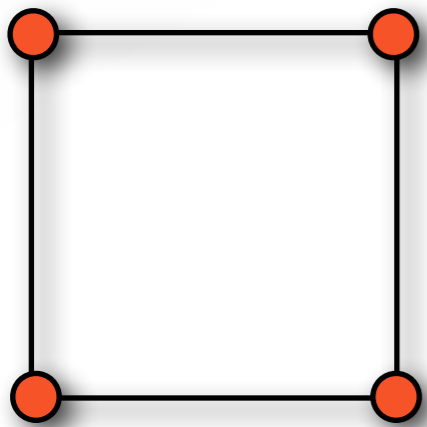
## 3. Compute intersection points

- Linear interpolation along edges

## 4. Connect them by edges

- Look-up table for edge configuration

# 2D: Marching Square



# 3D: Marching Cubes

## 1. Classify grid nodes as inside/outside

- Is  $F(\mathbf{x}_{i,j,k}) > 0$  or  $< 0$

## 2. Classify cell: $2^8$ configurations

- In/out for each corner

## 3. Compute intersection points

- Linear interpolation along edges

## 4. Connect them by edges

- Look-up table for path configuration
- Disambiguation by modified table [Montani '94]

# 3D: Marching Cubes

**Classify grid nodes  $\mathbf{x}_{i,j,k}$  based on  $F_{i,j,k} = F(\mathbf{x}_{i,j,k})$**

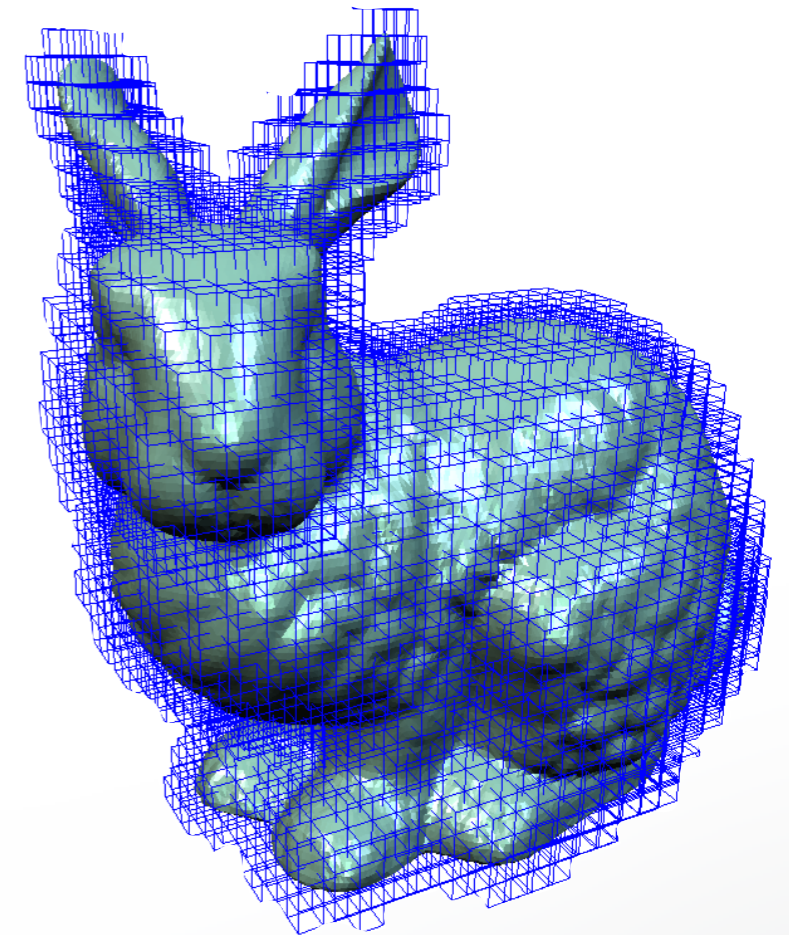
- Inside or outside

**Classify all cubes based on  $F_{i,j,k}$**

- Inside, outside, or intersecting

**Refined only intersected cells**

- 3-color adaptive octree
- $O(h^{-2})$  complexity

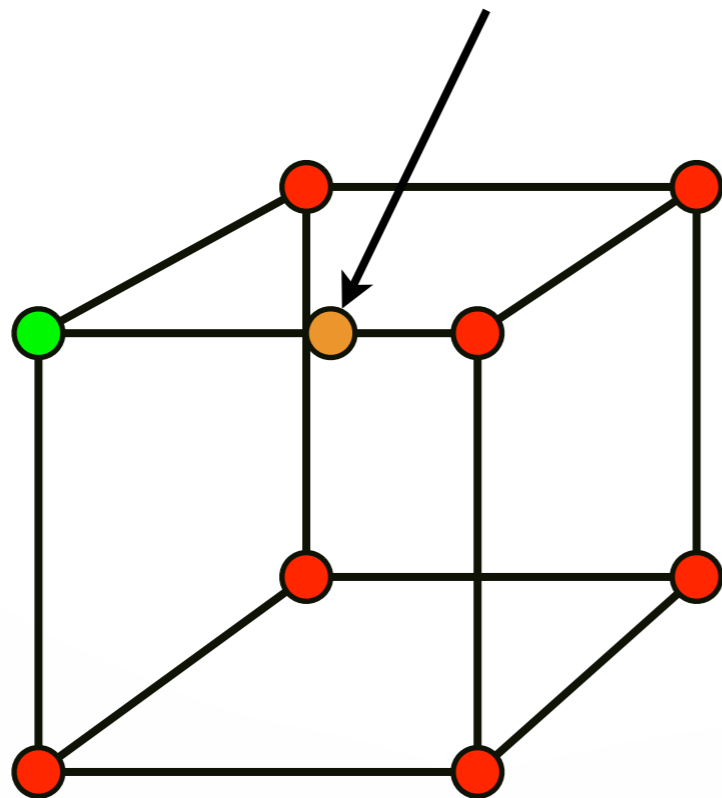




# Intersection Points

## Linear interpolation along edges

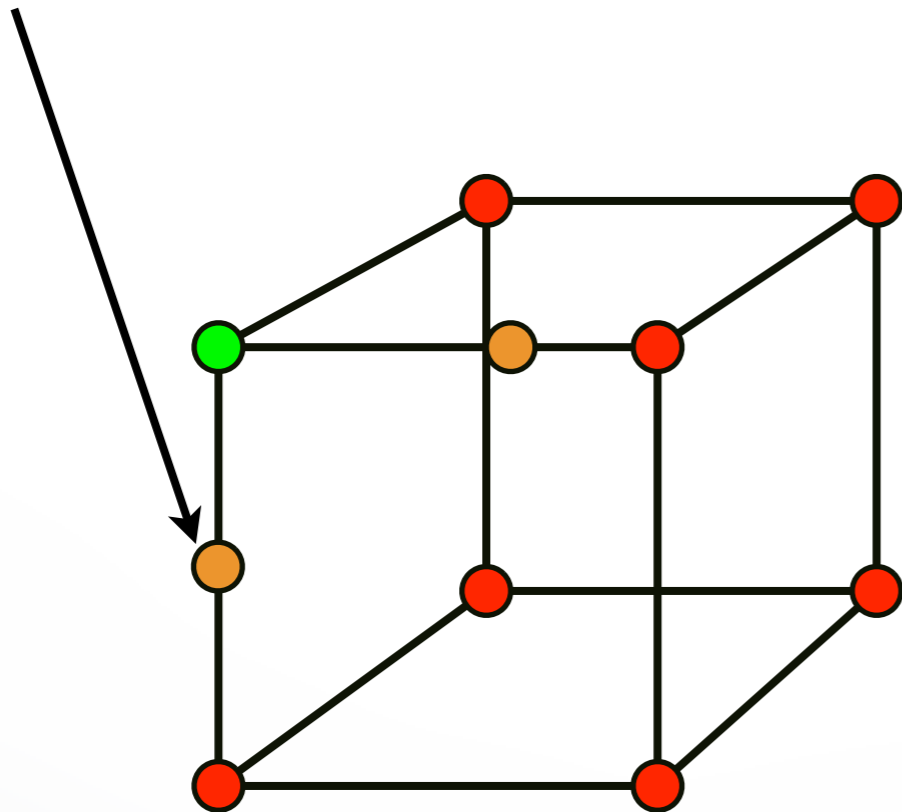
$$\frac{\mathbf{x}_{i,j,k} \cdot |F_{i+1,j,k}| + \mathbf{x}_{i+1,j,k} \cdot |F_{i,j,k}|}{|F_{i,j,k}| + |F_{i+1,j,k}|}$$



# Intersection Points

## Linear interpolation along edges

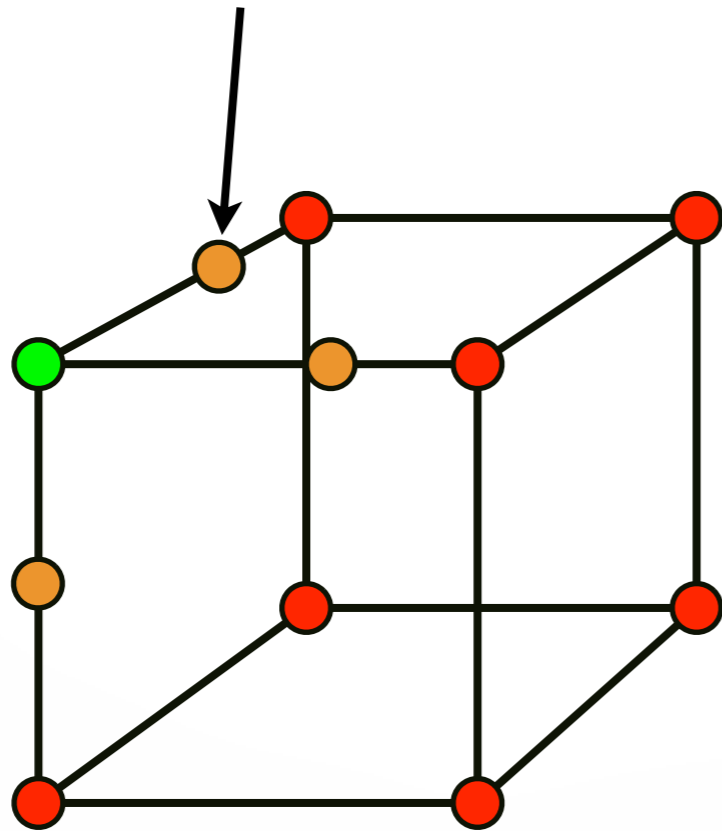
$$\frac{\mathbf{x}_{i,j,k} \cdot |F_{i,j+1,k}| + \mathbf{x}_{i,j+1,k} \cdot |F_{i,j,k}|}{|F_{i,j,k}| + |F_{i,j+1,k}|}$$



# Intersection Points

## Linear interpolation along edges

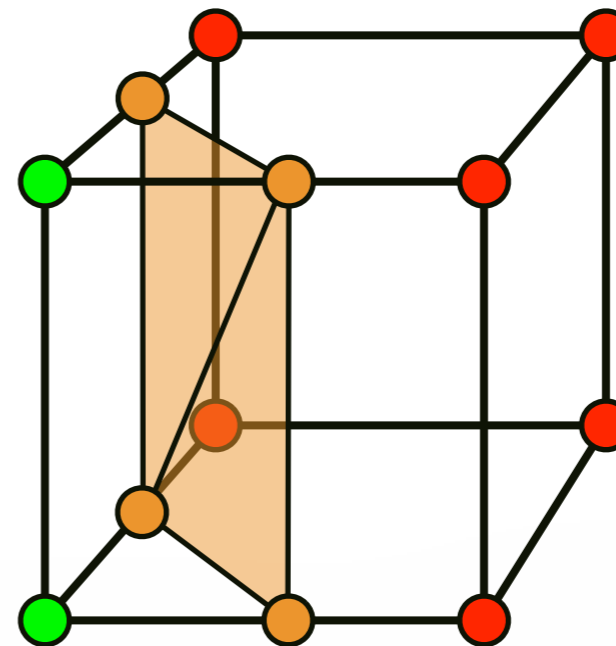
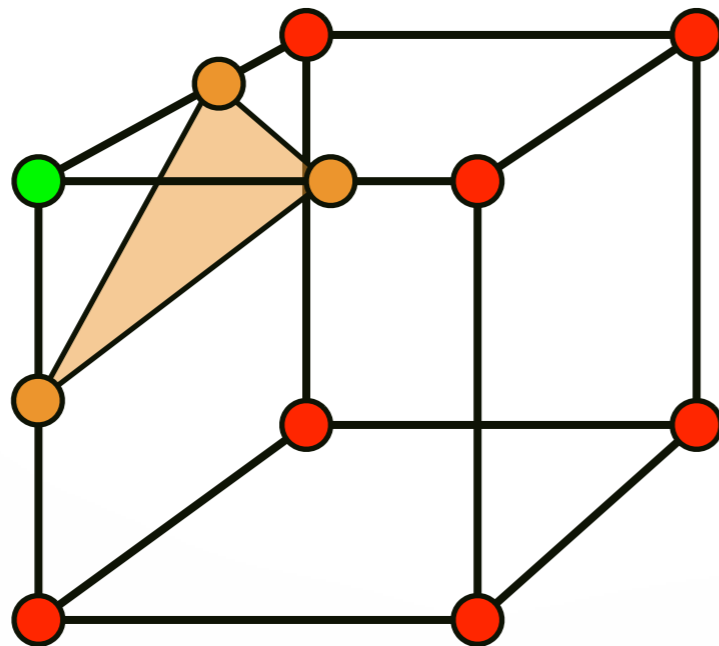
$$\frac{\mathbf{x}_{i,j,k} \cdot |F_{i,j,k+1}| + \mathbf{x}_{i,j,k+1} \cdot |F_{i,j,k}|}{|F_{i,j,k}| + |F_{i,j,k+1}|}$$



# Intersection Points

Linear interpolation along edges

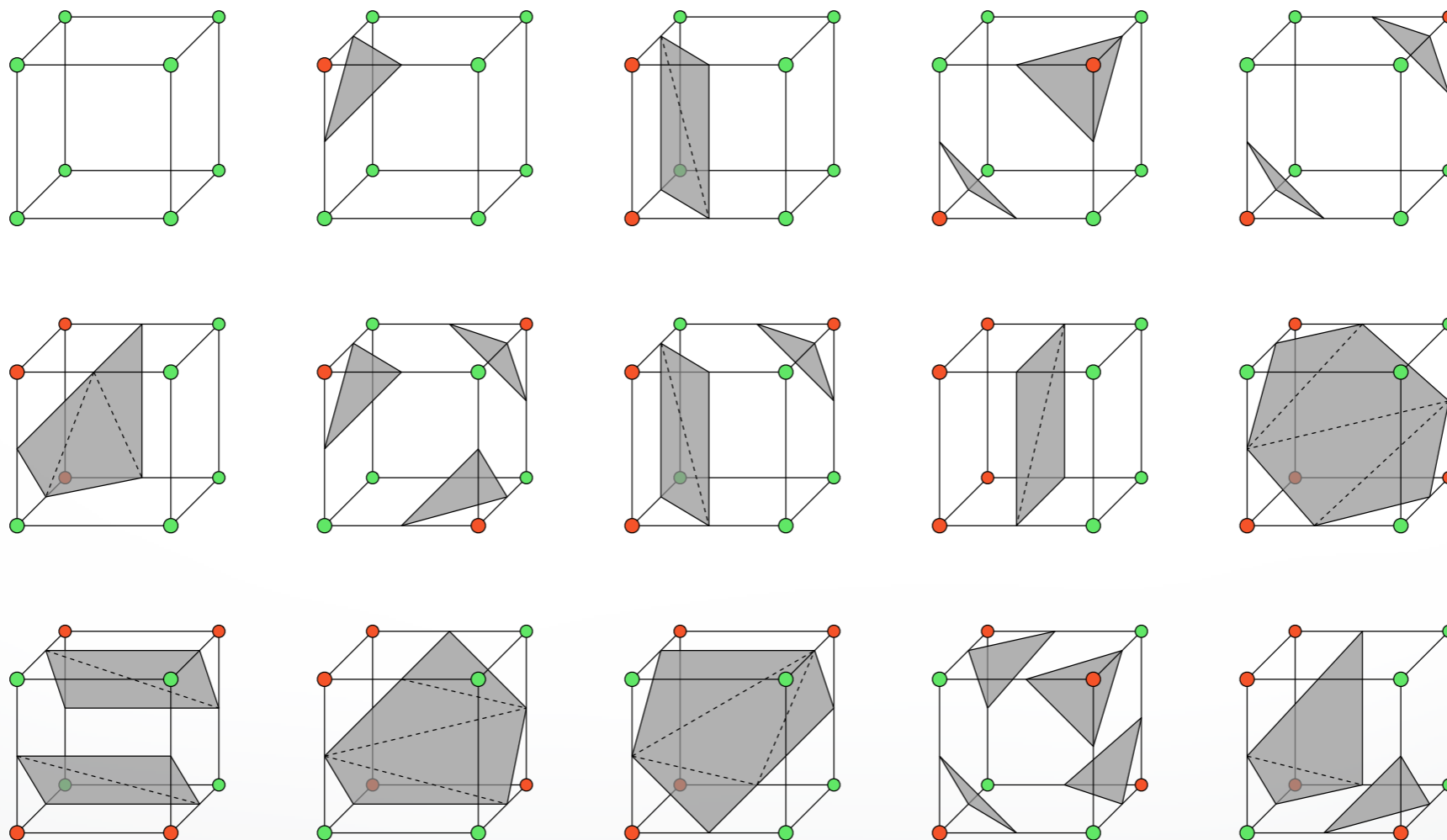
Lookup table for patch configuration



# Marching Cubes

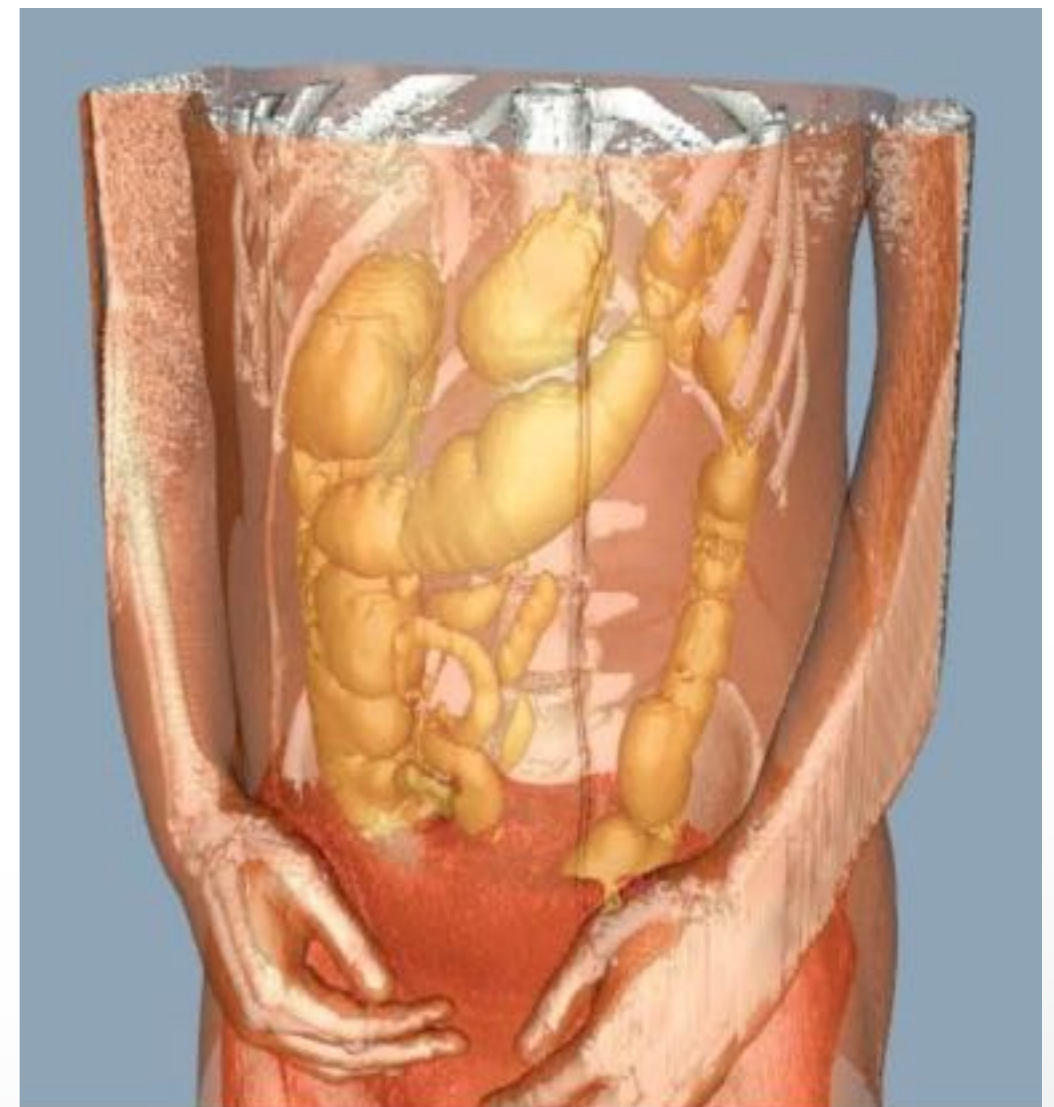
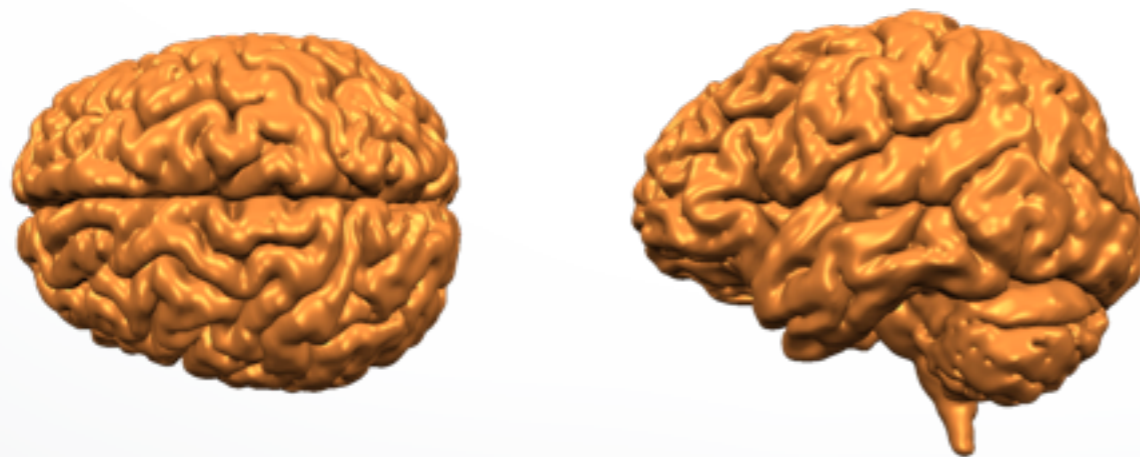
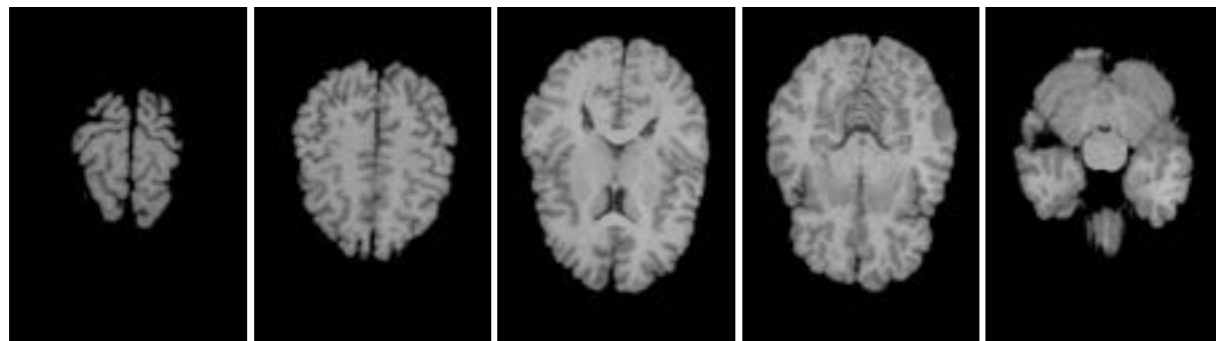
## Look-up table with $2^8$ entries

- 15 representative cases shown
- Others follow by symmetry



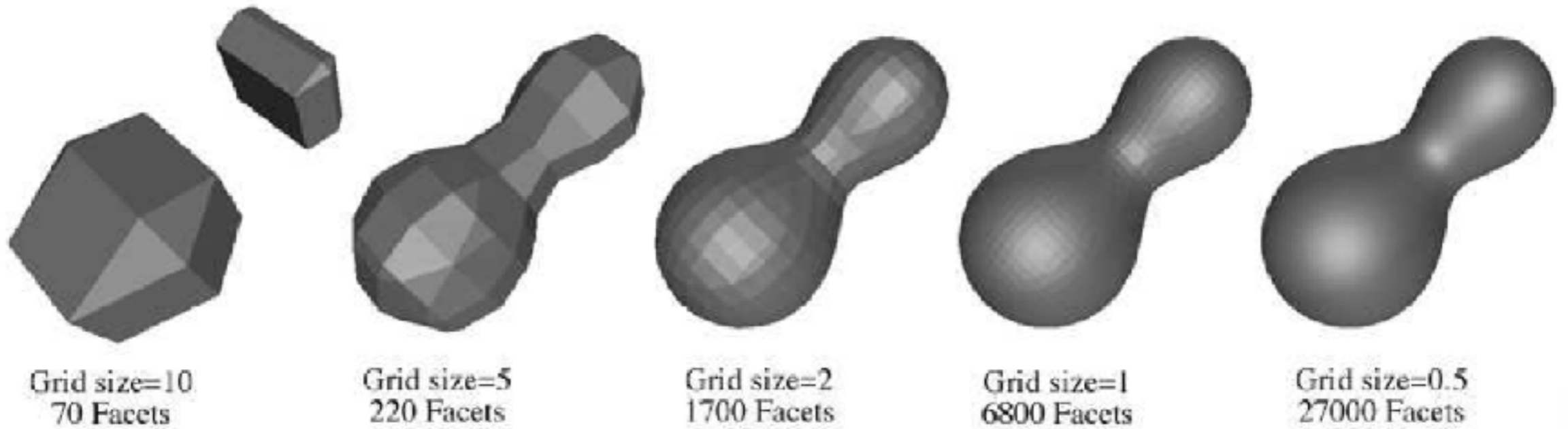
# Marching Cubes

Algorithm for isosurface extraction from medical scans (CT, MRI)



# Marching Cubes

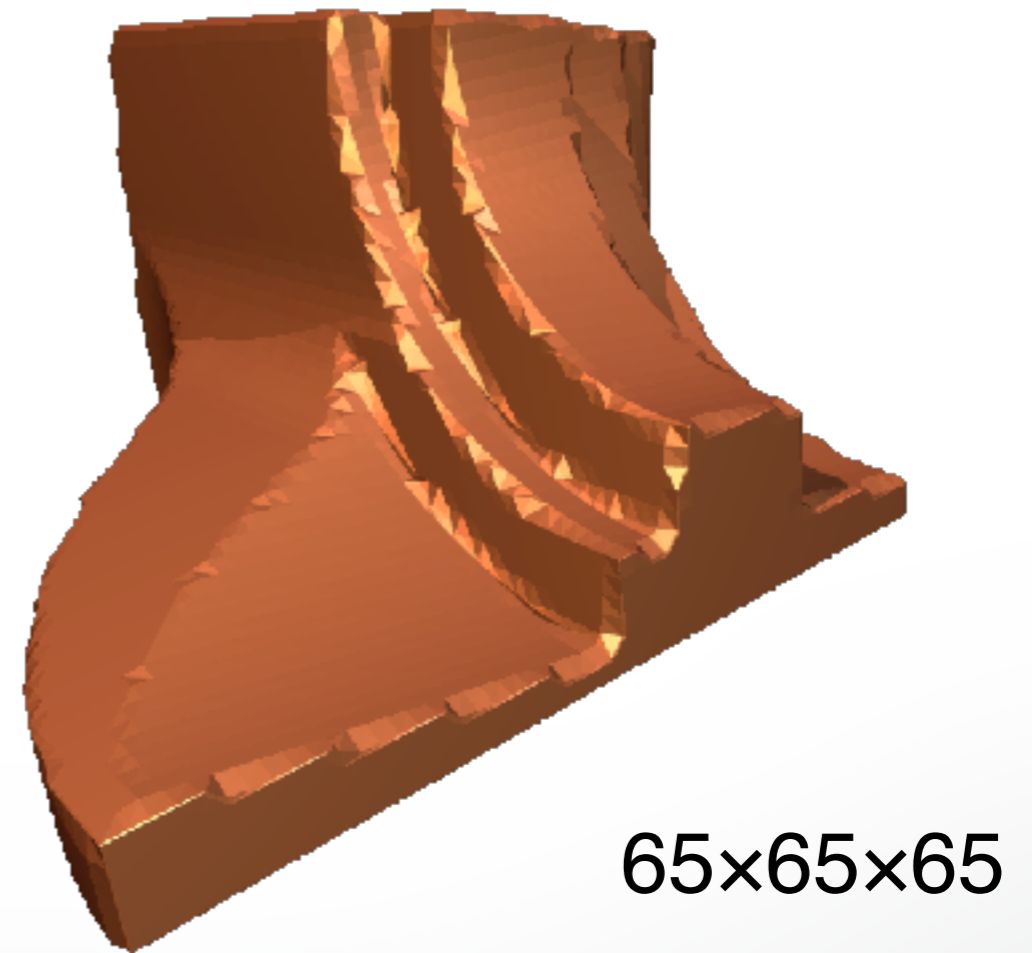
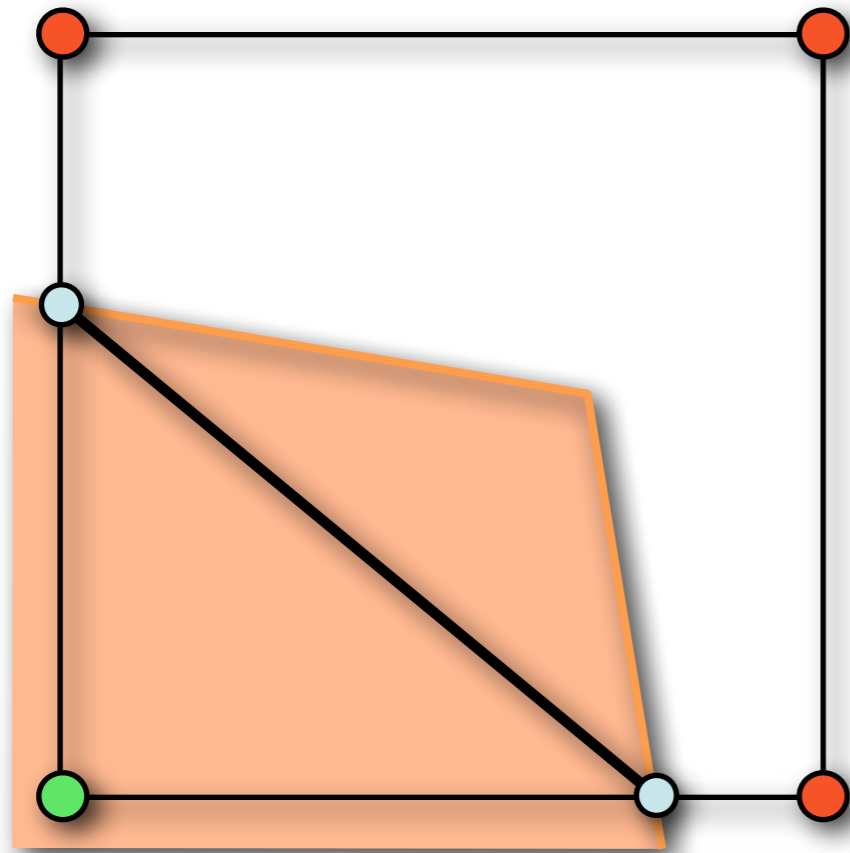
## Effect of grid size



# Marching Cubes

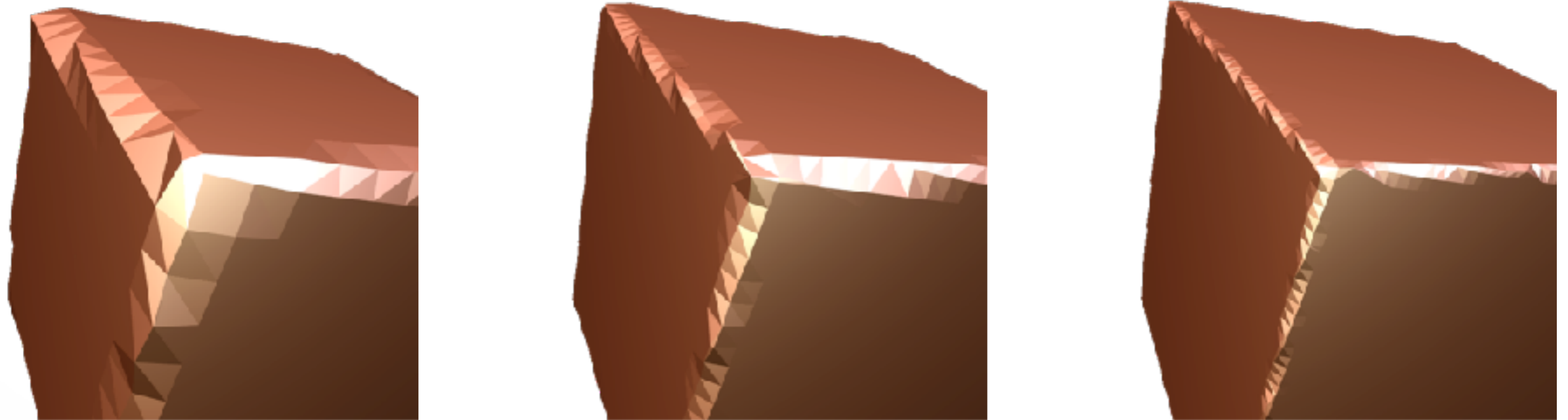
Sample points restricted to edges of regular grid

Alias artifacts at sharp features





# Increasing Resolution

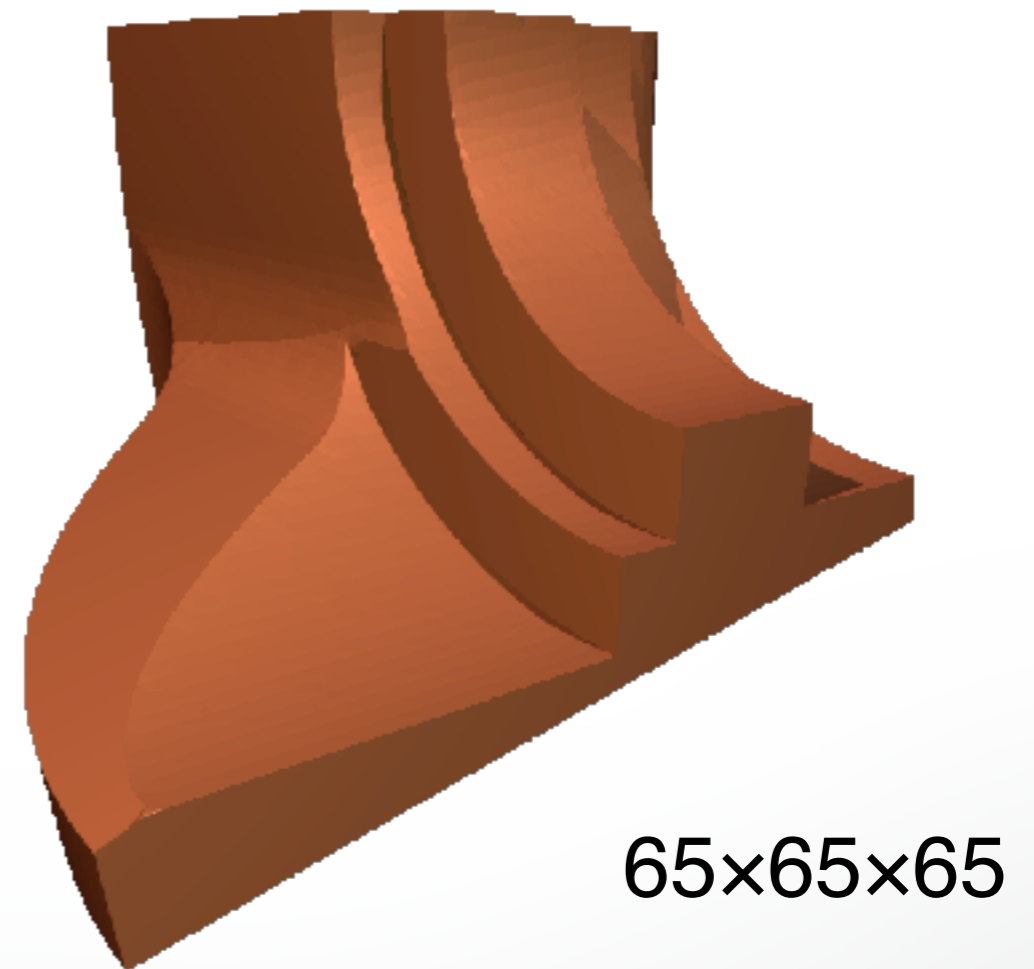
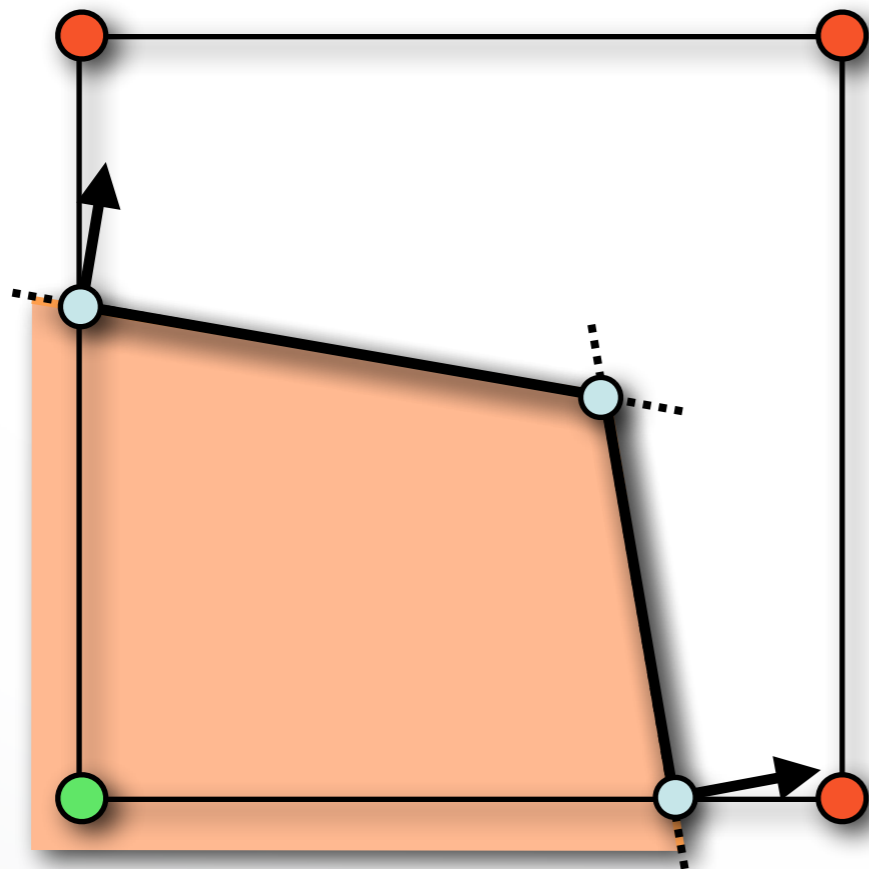


**Does not remove alias problems!**

# Extended Marching Cubes

Locally extrapolate distance gradient

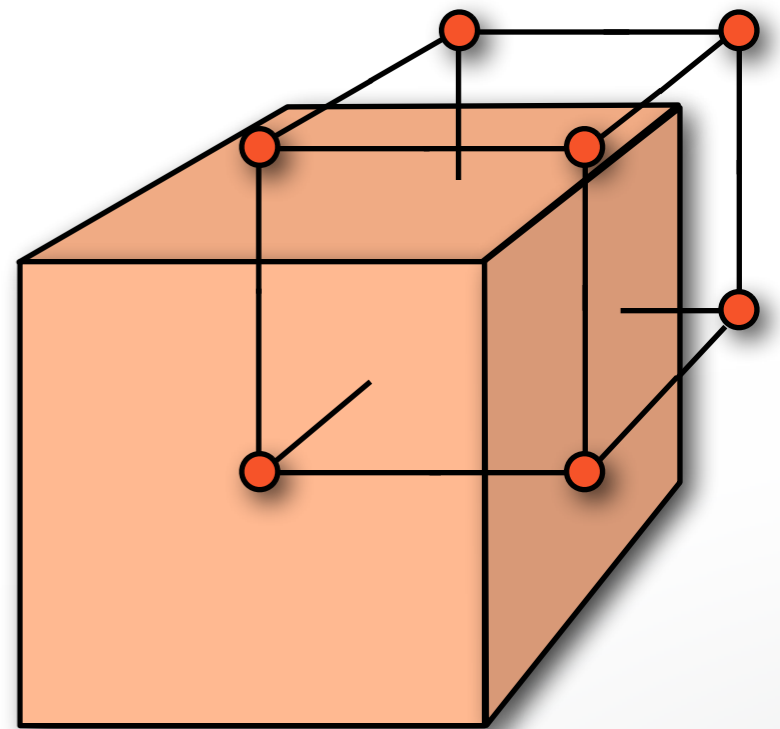
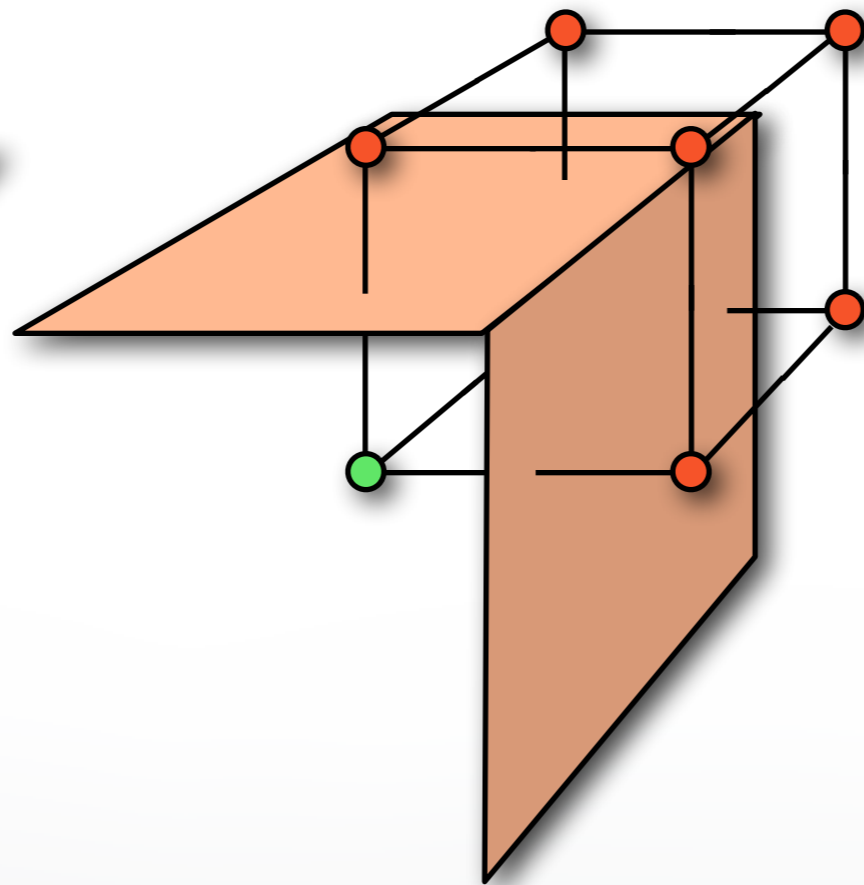
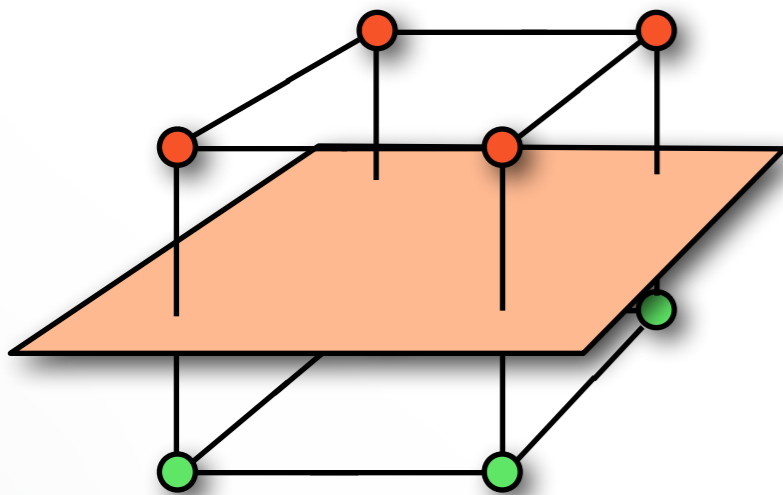
Place samples on estimated features



# Extended Marching Cubes

## Feature detection

- Based on angle between normals  $\mathbf{n}_i$
- Classify into edges / corners



# Extended Marching Cubes

## Feature sampling

- Intersect tangent planes  $(\mathbf{s}_i, \mathbf{n}_i)$

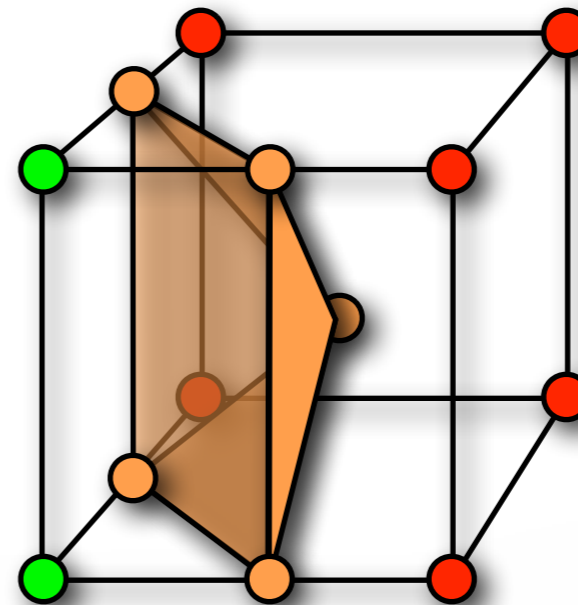
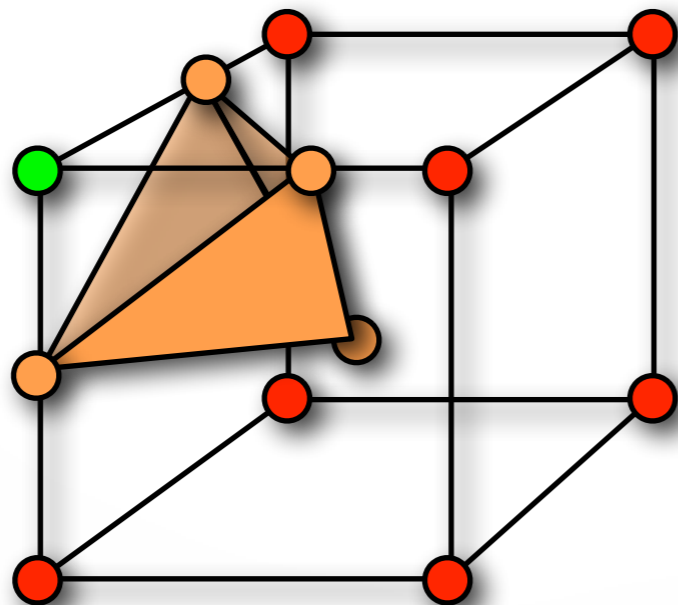
$$\begin{pmatrix} \vdots \\ \mathbf{n}_i \\ \vdots \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \vdots \\ \mathbf{n}_i^T \mathbf{s}_i \\ \vdots \end{pmatrix}$$

- Over- or under-determined system
- Solve by SVD pseudo-inverse

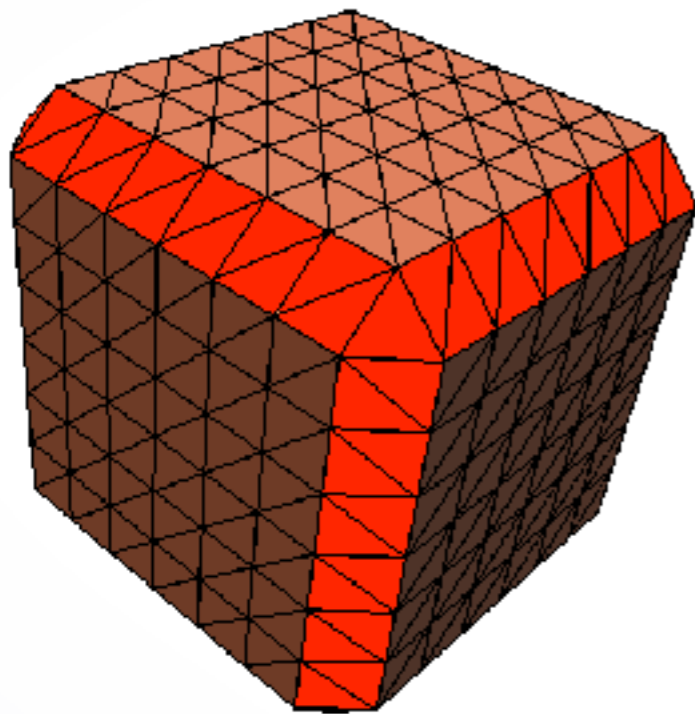
# Extended Marching Cubes

## Feature sampling

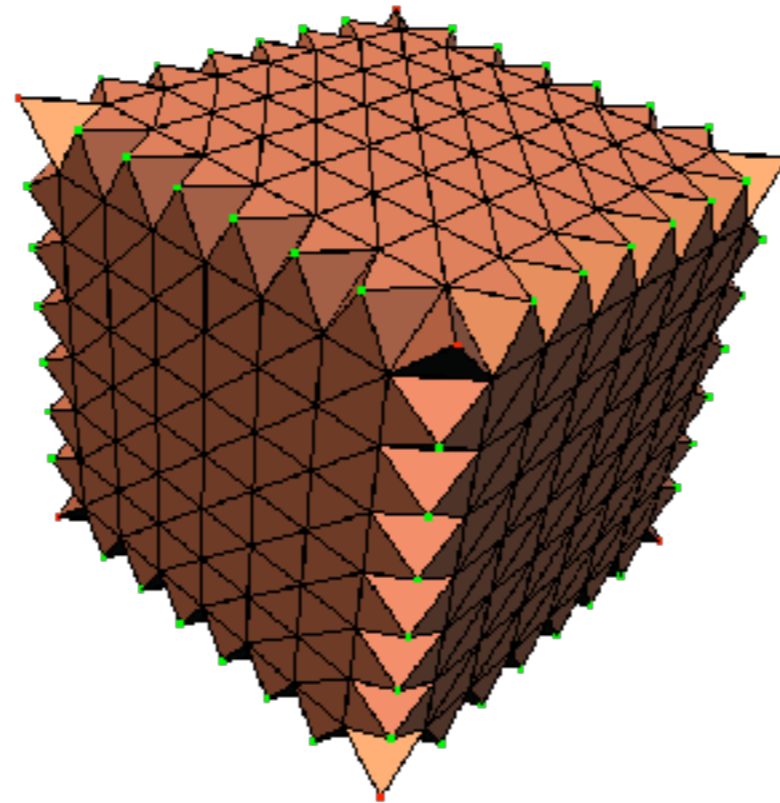
- Intersect tangent planes  $(\mathbf{s}_i, \mathbf{n}_i)$
- Triangle fans centered at feature point



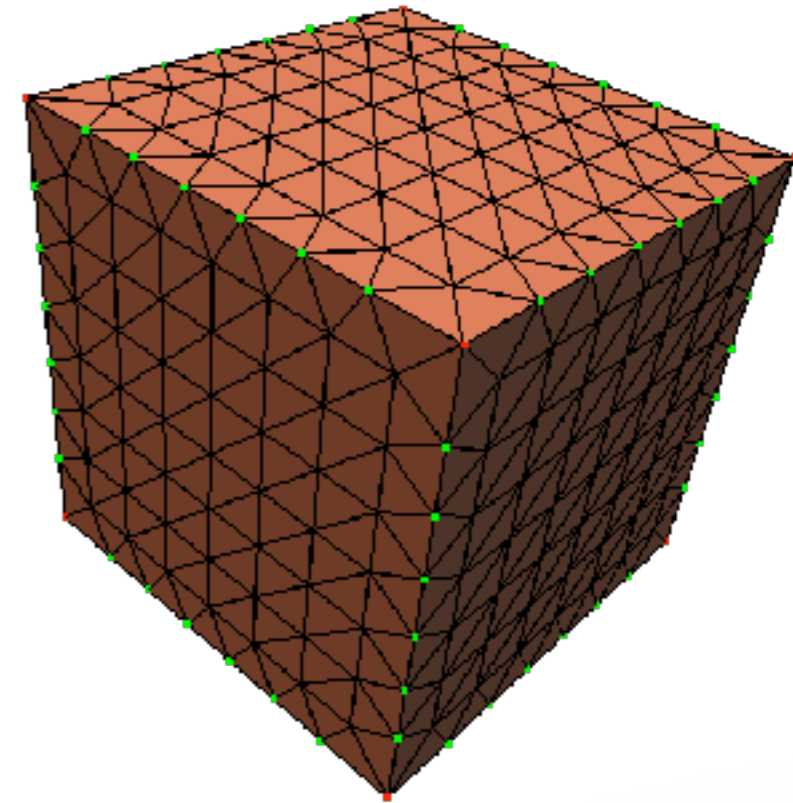
# Extended Marching Cubes



**Feature  
Detection**

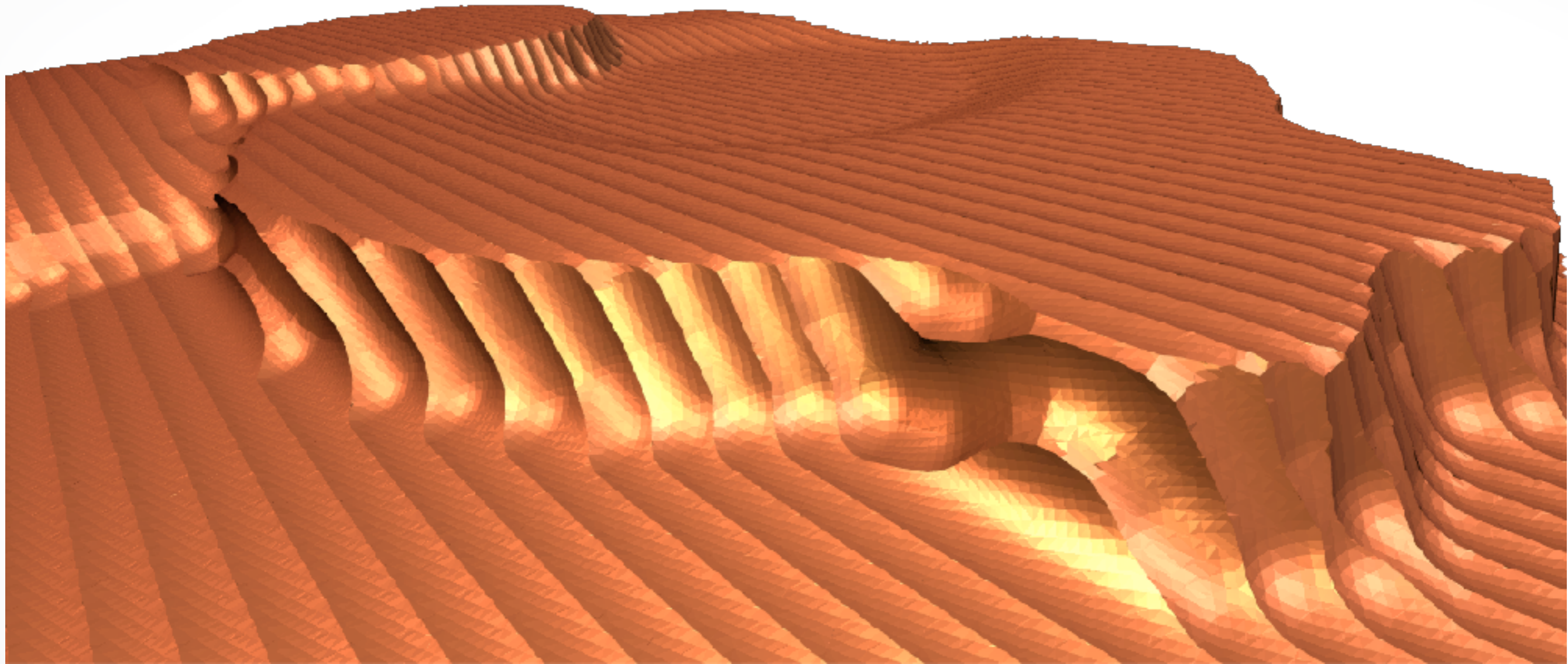


**Feature  
Sampling**



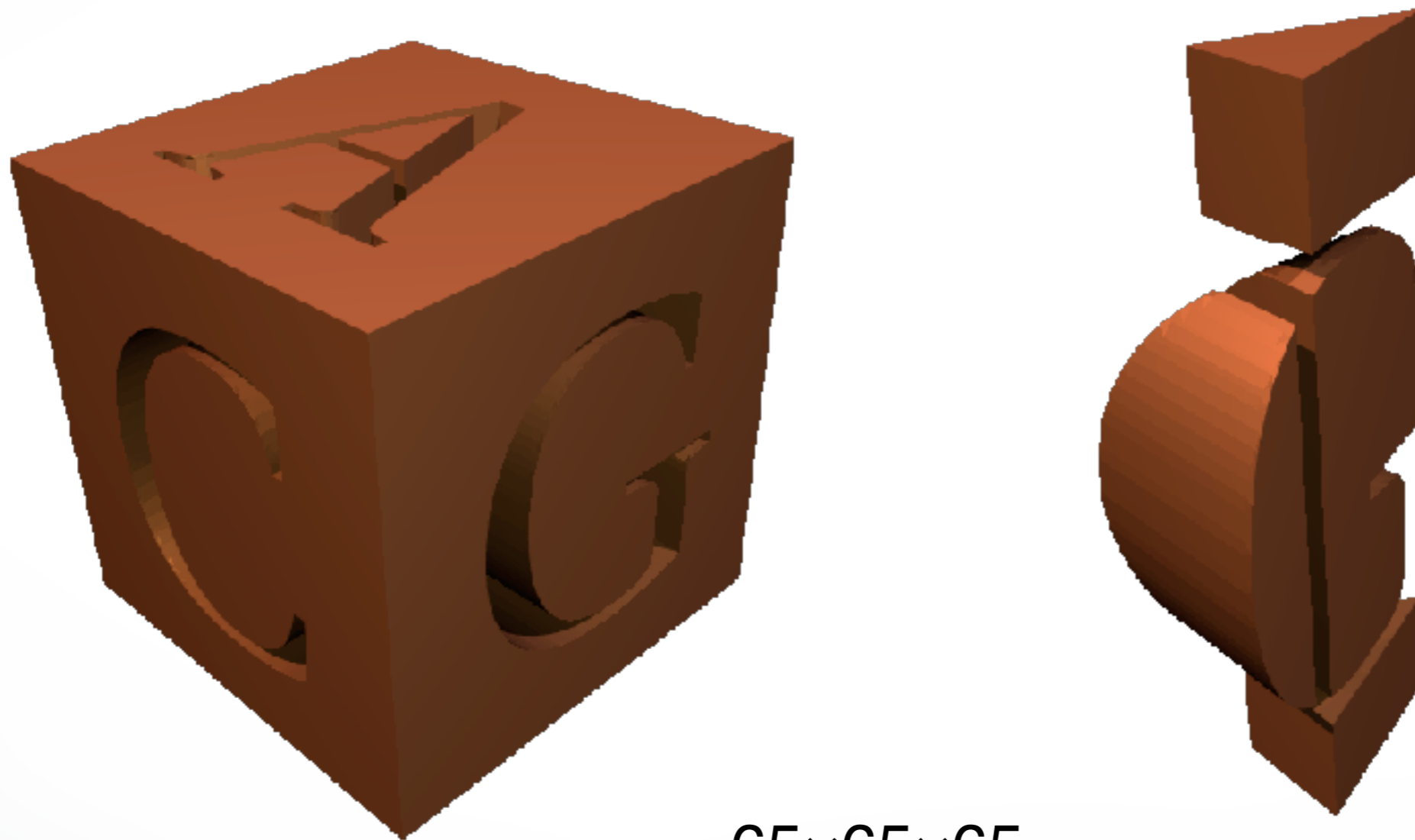
**Edge  
Flipping**

# Milling Simulation



$257 \times 257 \times 257$

# CSG Modeling



65×65×65



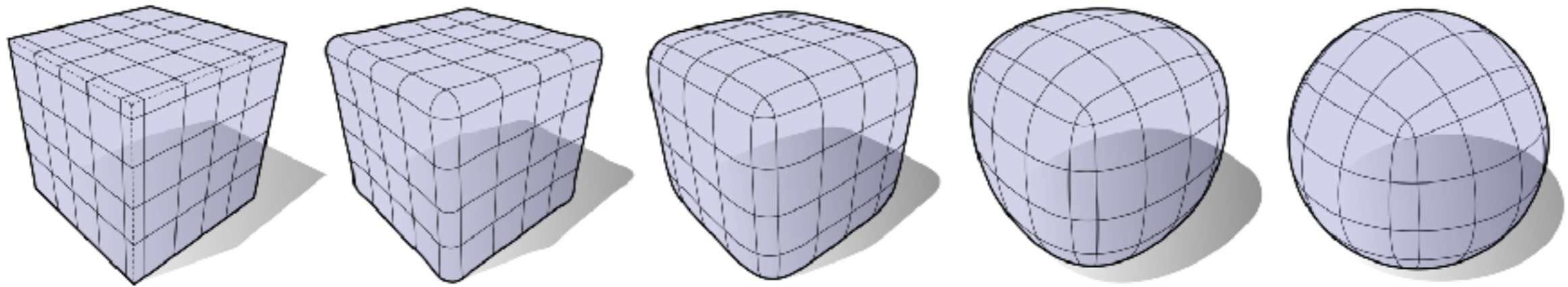
# Marching Cubes

- + Result is watertight, closed 2-manifold surface!
- + Easy to parallelize
- Uniform (over-) sampling (→ mesh decimation)
- Degenerate triangles (→ remeshing)
- MC does not preserve features
- + EMC preserves features, but...
  - about 10% more triangles
  - 20-40% computational overhead

# Literature

- Lorensen & Cline, “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, SIGGRAPH 1987
- Montani et al., “A modified look-up table for implicit disambiguation of Marching Cubes”, Visual Computer 1994
- Kobbelt et al., “Feature Sensitive Surface Extraction from Volume Data”, SIGGRAPH 2001

# Next Time



**Discrete Differential Geometry**

<http://cs621.hao-li.com>

# Thanks!

