

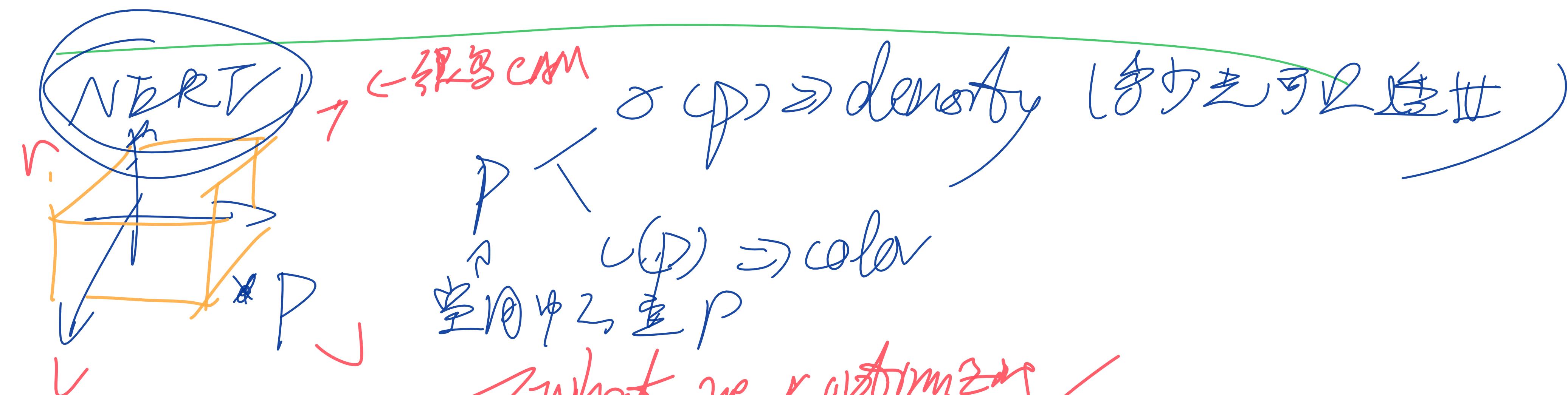
# EULER CHARACTERISTICS TOPOLOGICAL

$$\textcircled{1} V - E + F = 2(1 - g) (= 0)$$

GENUS  
意义在下面的PP7有

$$\textcircled{2} 3F = 2E \quad (\because \text{边被 } 2^q \text{ 取决})$$

$(F = 2V)$

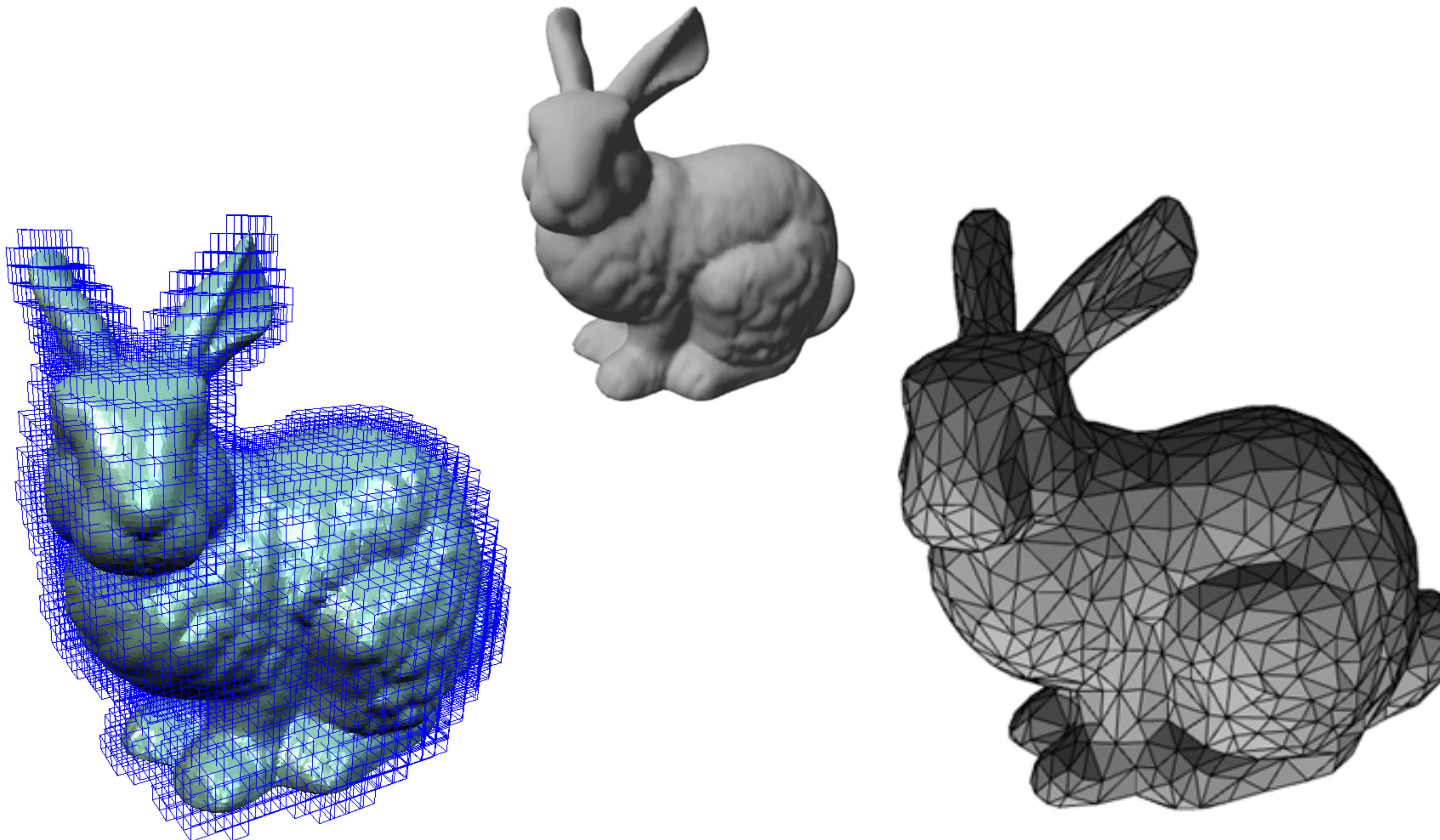


$$I_F(\theta, \phi, \text{CAMERA}_I) = \text{Image}_I$$

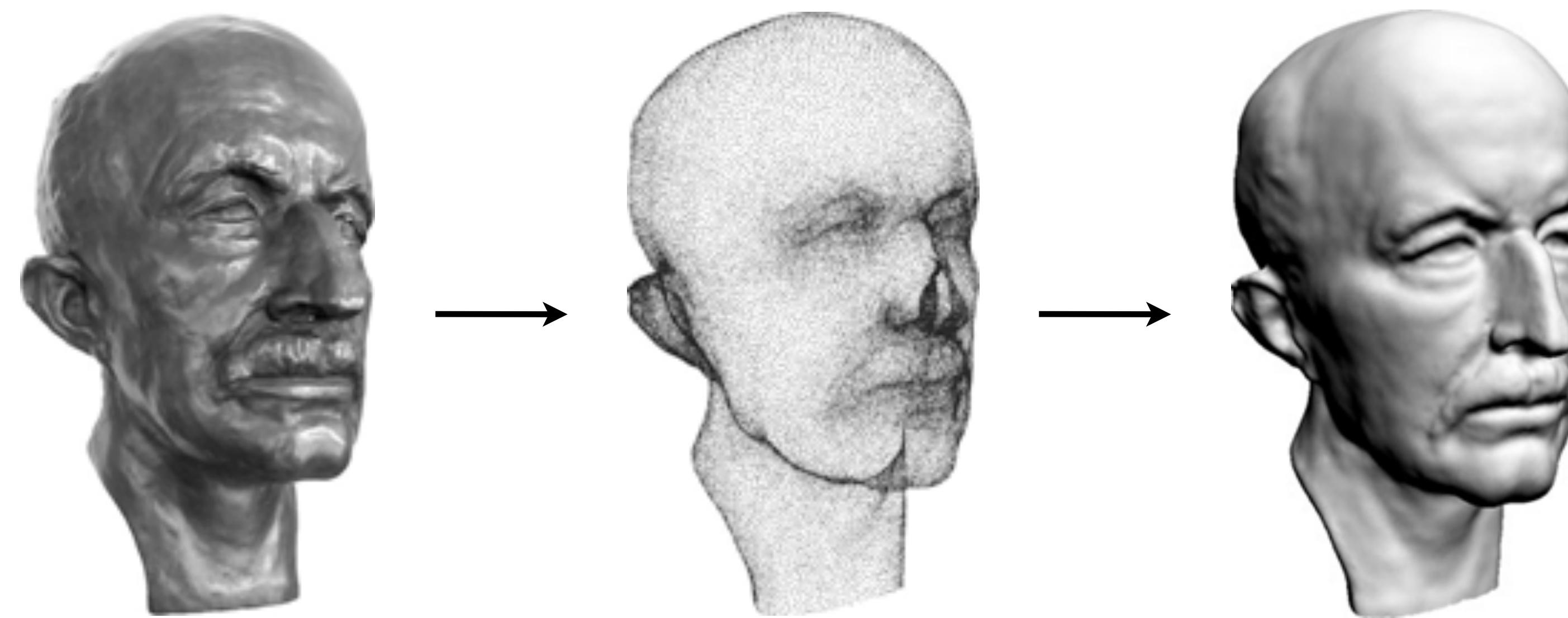
↑ image formation model

↑  $\text{Image}_G$

# Implicit and Explicit Representations



# Last Week

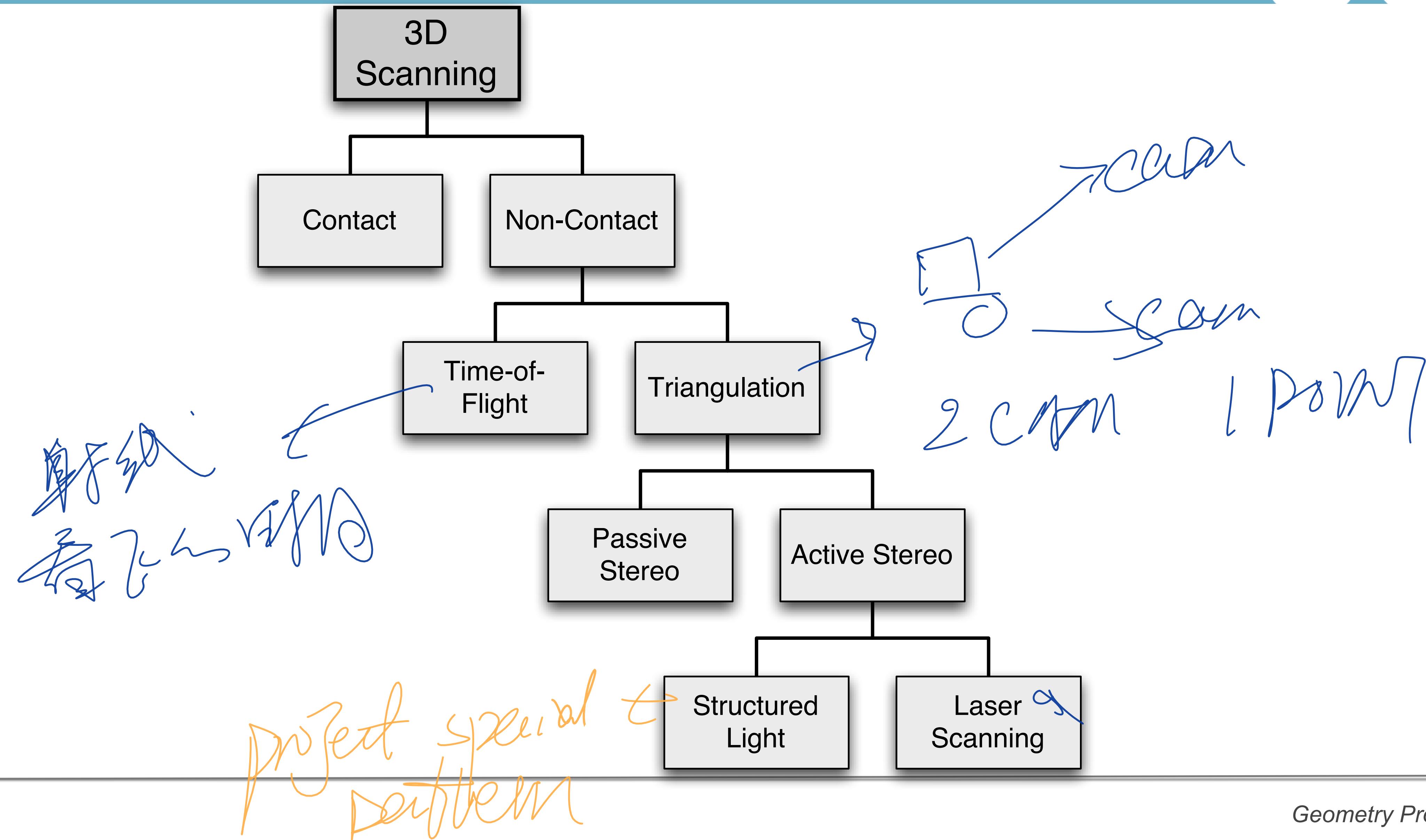


physical  
model

acquired  
point cloud

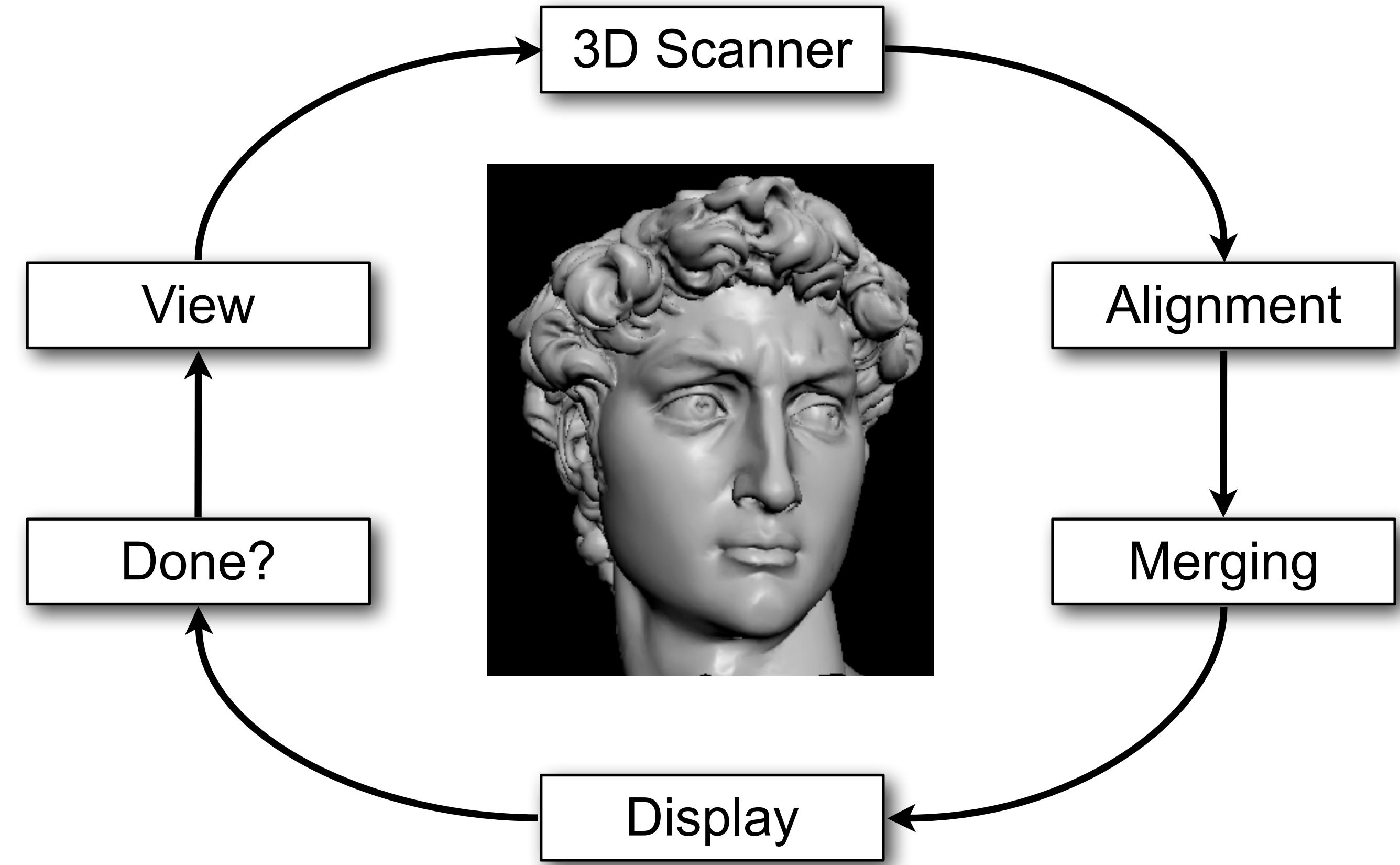
reconstructed  
model

# 3D Scanning Techniques



# Last Time

- 3D Model Acquisition Pipeline



# Recap



## Basic ICP Algorithm (rigidly align Q to P)

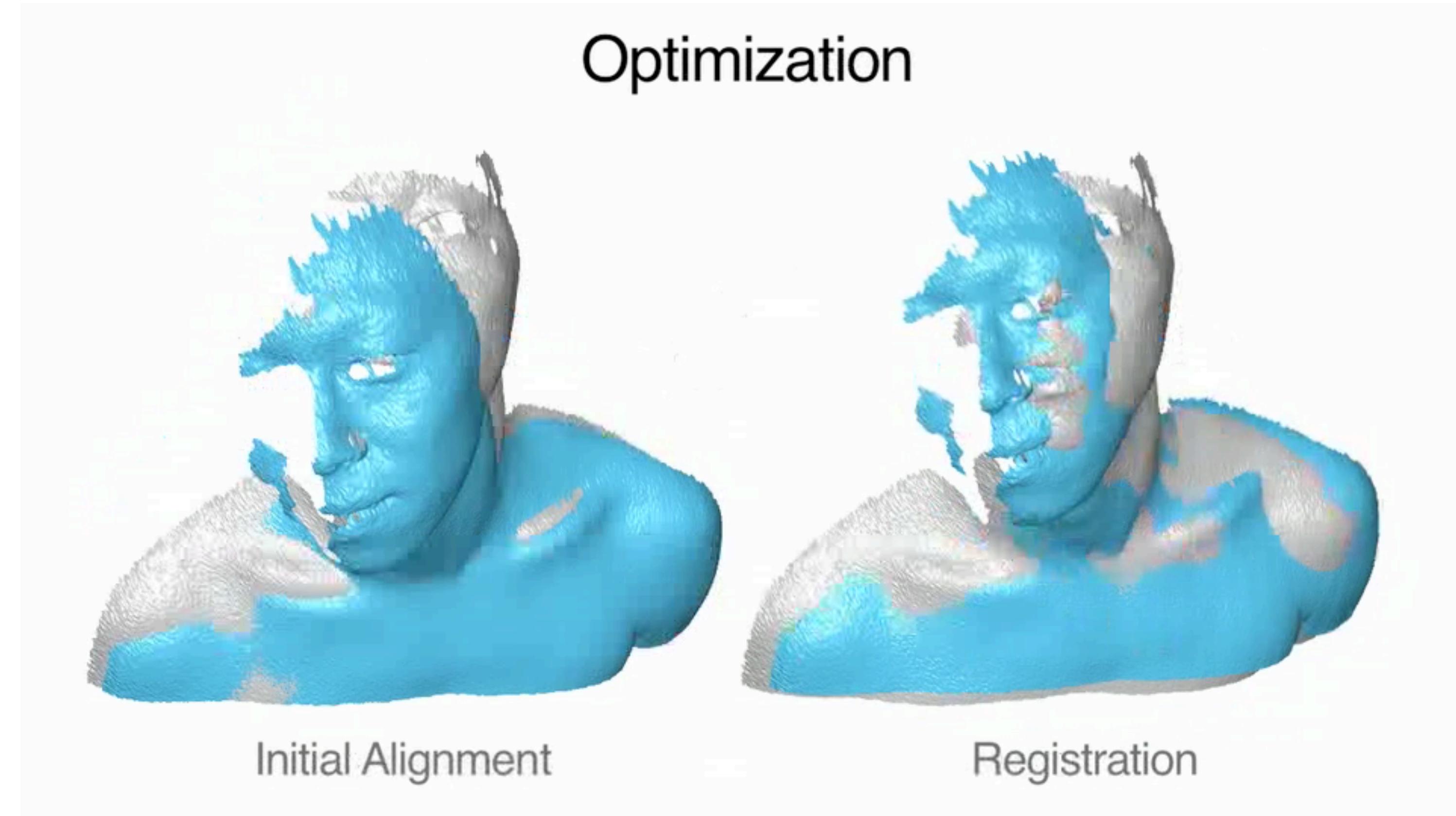
Iterate until convergence:

1. **Select** a subset of points  $\mathbf{p}_i$
2. **Match** each  $\mathbf{p}_i$  to closest point  $\mathbf{q}_i$  on other scan
3. **Reject** “bad” pairs  $(\mathbf{p}_i, \mathbf{q}_i)$
4. **Compute** rotation  $\mathbf{R}$  and translation  $\mathbf{t}$  to minimize

$$\min_{\mathbf{R}, \mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

5. **Iterate** after scan alignment:  $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$

# Non-rigid Registration



# Triangle Meshes

- **Topology:** vertices, edges, triangles

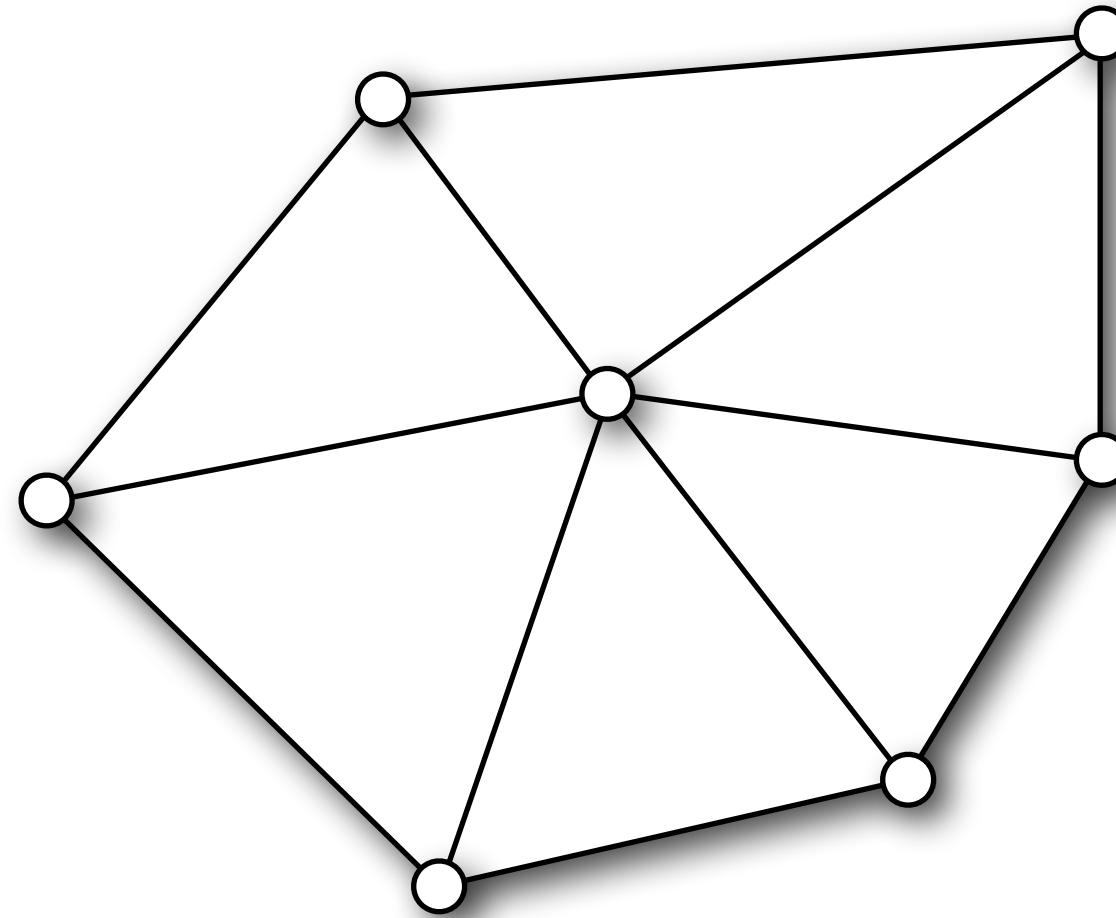
$$\mathcal{V} = \{v_1, \dots, v_n\}$$

$$\mathcal{E} = \{e_1, \dots, e_k\} , \quad e_i \in \mathcal{V} \times \mathcal{V}$$

$$\mathcal{F} = \{f_1, \dots, f_m\} , \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$$

- **Geometry:** vertex positions

$$\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\} , \quad \mathbf{p}_i \in \mathbb{R}^3$$



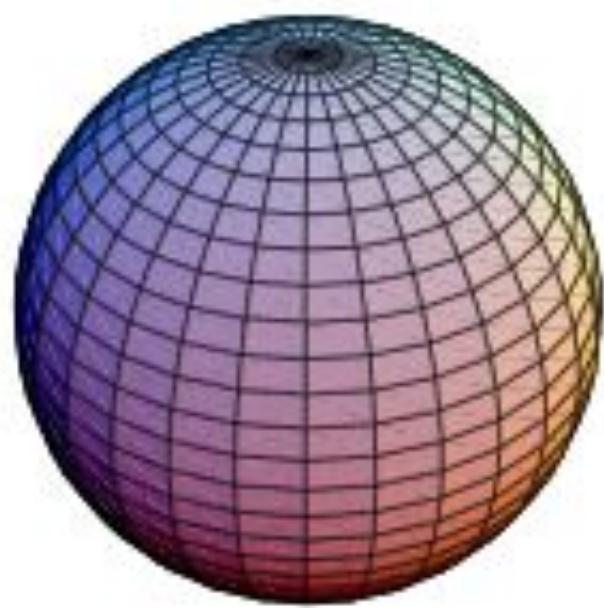
# Global Connectivity: Genus



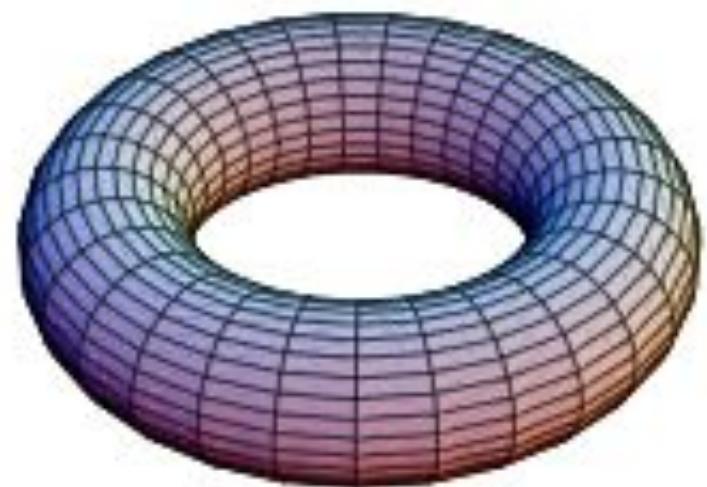
## Genus:

Maximal number of closed simple cutting curves that do not disconnect the graph into multiple components.

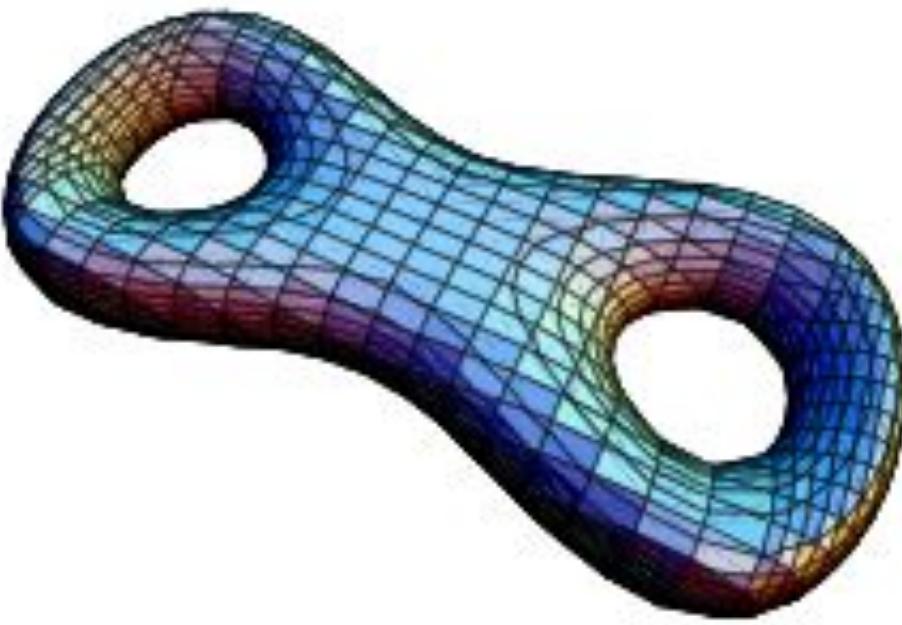
*Informally, the number of holes or handles.*



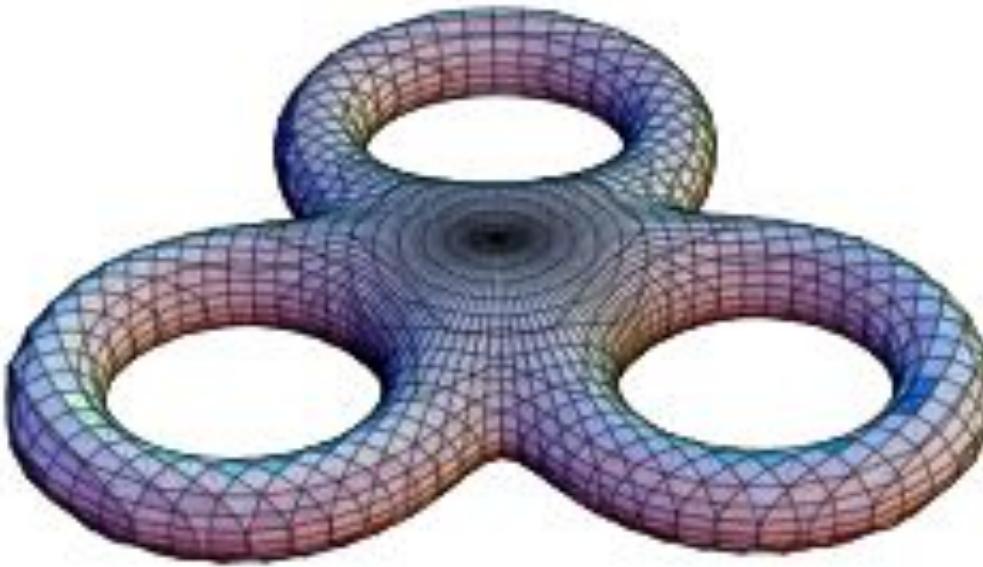
Genus 0



Genus 1



Genus 2



Genus 3

# Euler-Poincare Formula



- For a closed polygonal mesh of genus  $g$ , the relation of the number  $V$  of vertices,  $E$  of edges, and  $F$  of faces is given by *Euler's formula*  $\chi$

$$V - E + F = 2(1 - g)$$

- The term  $2(1 - g)$  is called the *Euler characteristic*

# Consequences of Euler Characteristics

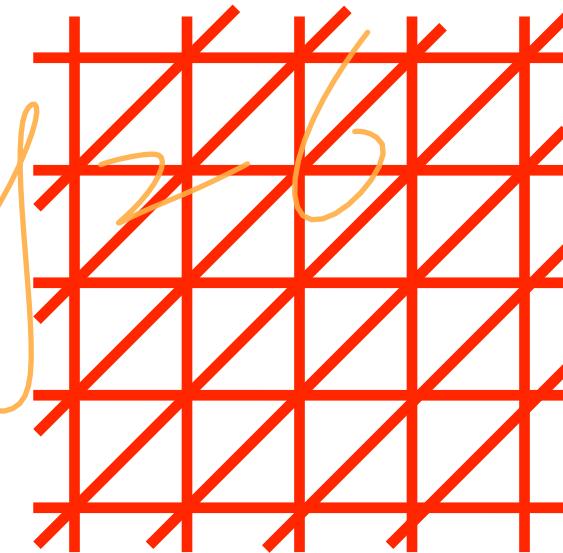
- Triangle meshes

- $F \approx 2V$

- $E \approx 3V$

- Average valence = 6

$\Rightarrow V - E \Rightarrow$   
手稿  
九章算术

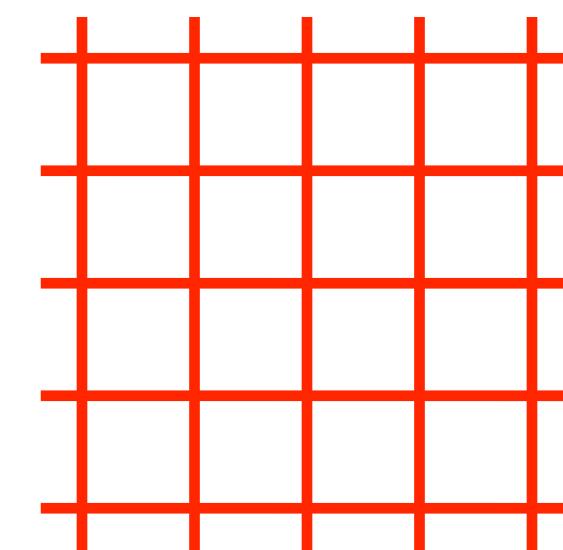


- Quad meshes

- $F \approx V$

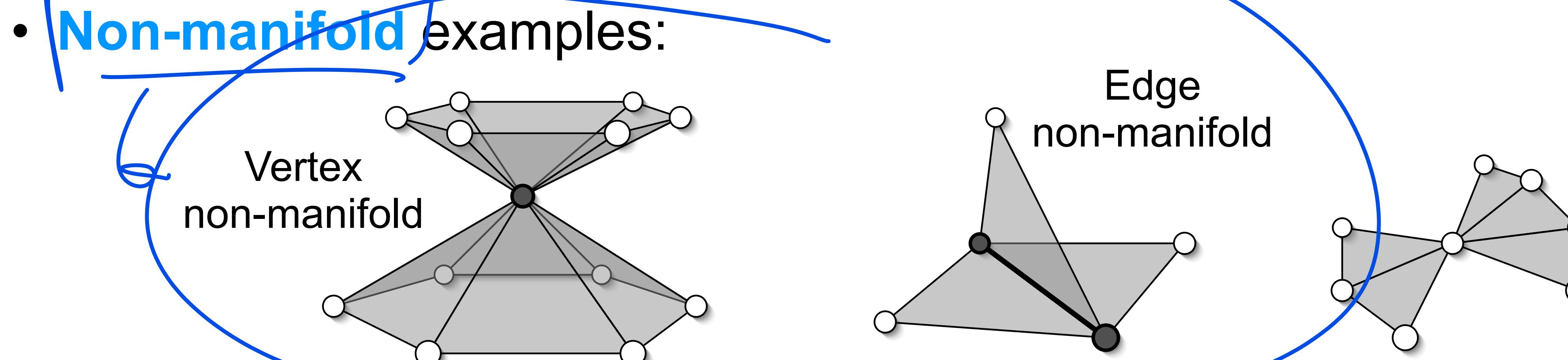
- $E \approx 2V$

- Average valence = 4



# 2-Manifold Surfaces

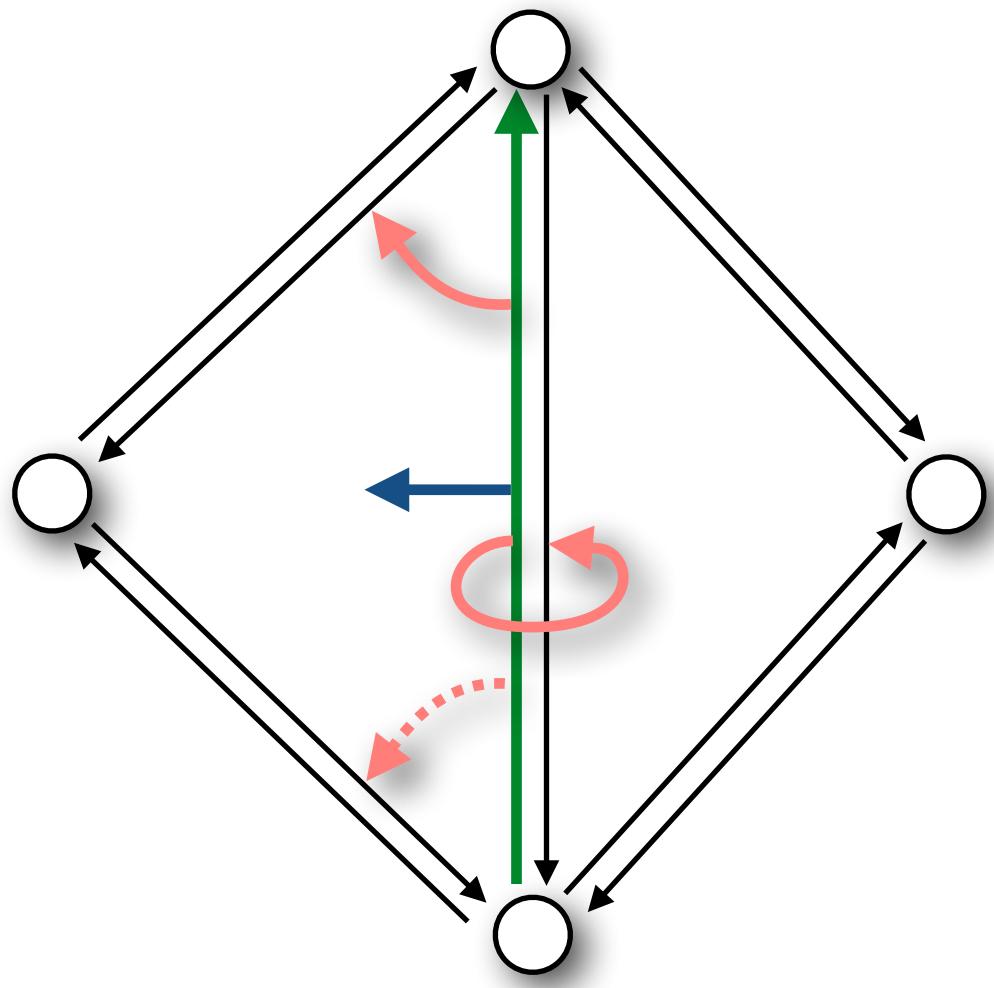
- Local neighborhoods are disk-shaped  
$$f(D_\varepsilon[u, v]) = D_\delta[f(u, v)]$$
- Guarantees meaningful neighbor enumeration
  - required by most algorithms



# Halfedge-Based Connectivity



- Vertex
  - position
  - 1 halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- Face
  - 1 halfedge



96 to 144 B/v  
no case distinctions  
during traversal

# Halfedge-Based Libraries

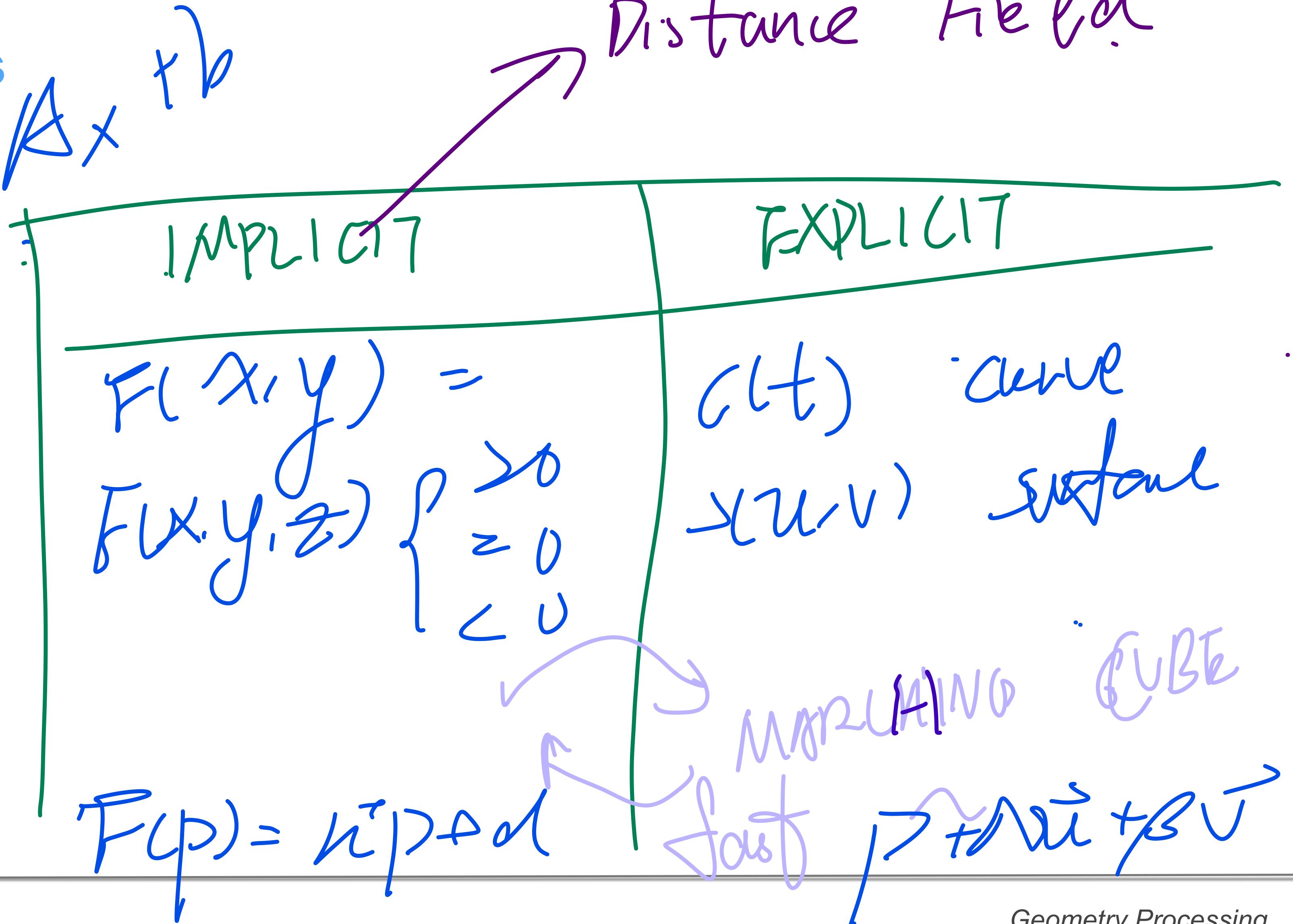


- CGAL
  - [www.cgal.org](http://www.cgal.org)
  - Computational geometry
  - Free for non-commercial use
- OpenMesh
  - [www.openmesh.org](http://www.openmesh.org)
  - Mesh processing
  - Free, LGPL licence

( )

# Overview

- Surface representations
  - Explicit vs. Implicit
- Explicit representations
  - Triangle meshes
- Implicit representations
  - Signed distance fields
- Conversions
  - Explicit  $\leftrightarrow$  Implicit



# Explicit vs. Implicit

Explicit:

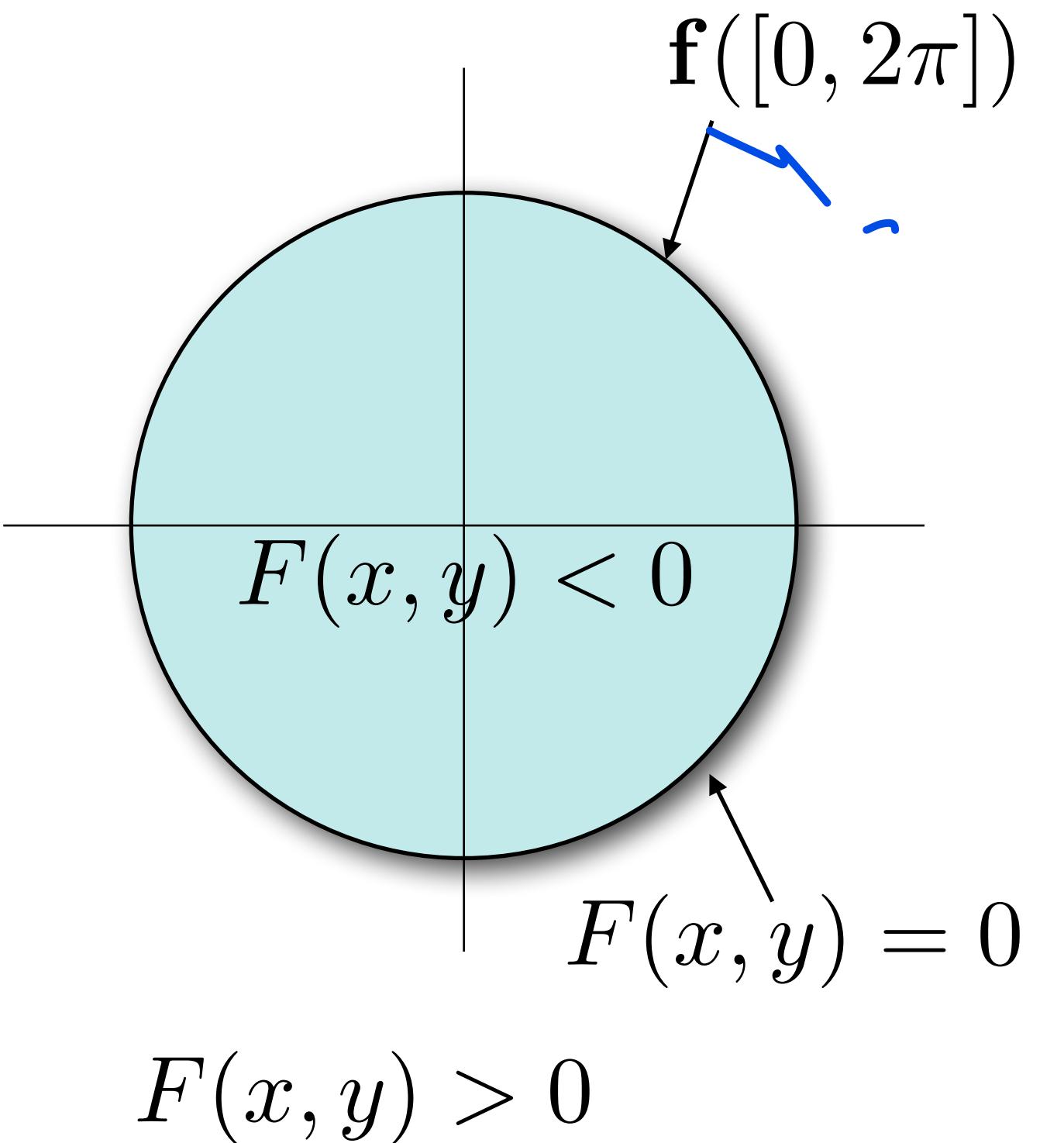
- Range of parametrization function

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

Implicit:

- Kernel of implicit function

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



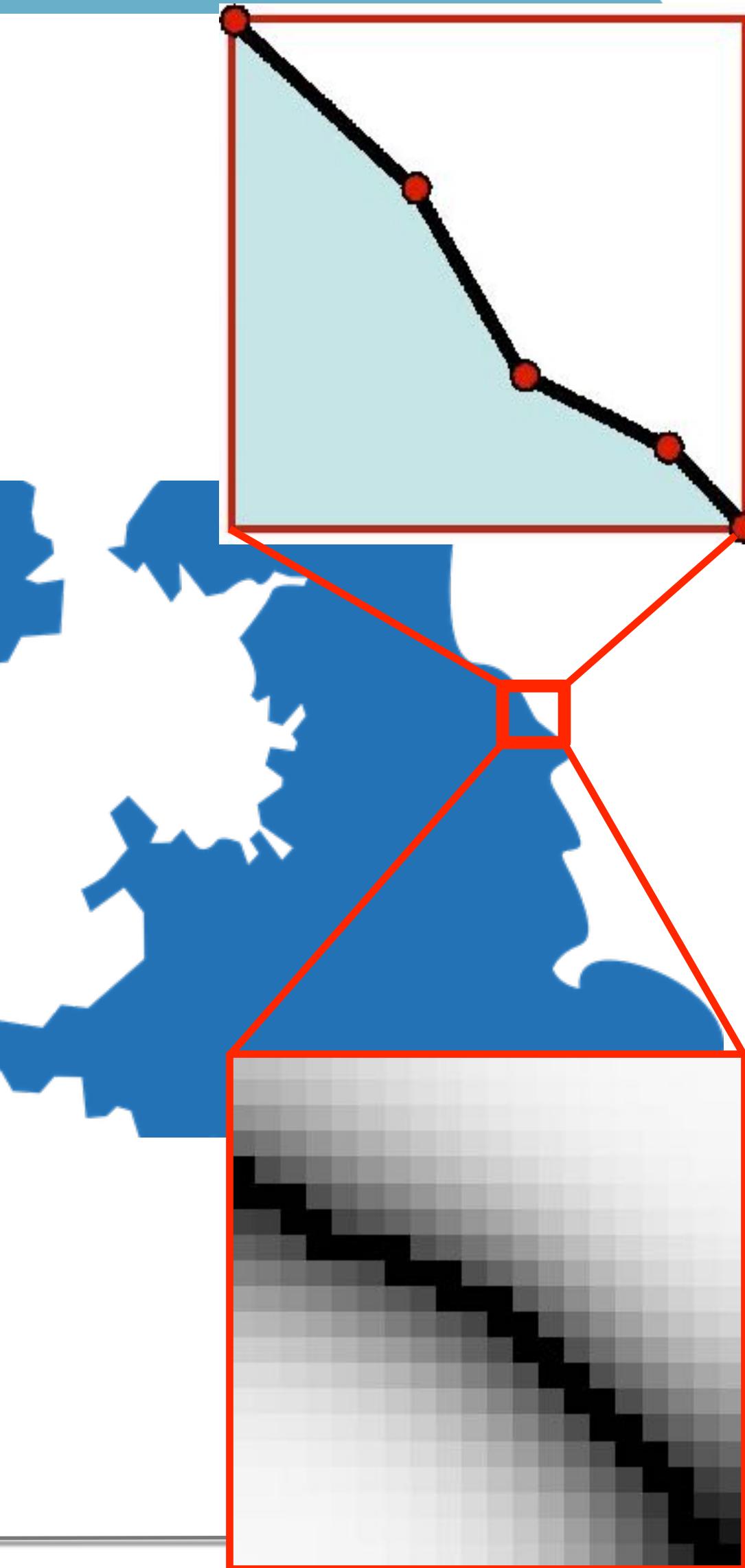
# Explicit vs. Implicit

- Explicit:
  - Range of parametrization function
  - Piecewise approximation!

$$f(x) =$$

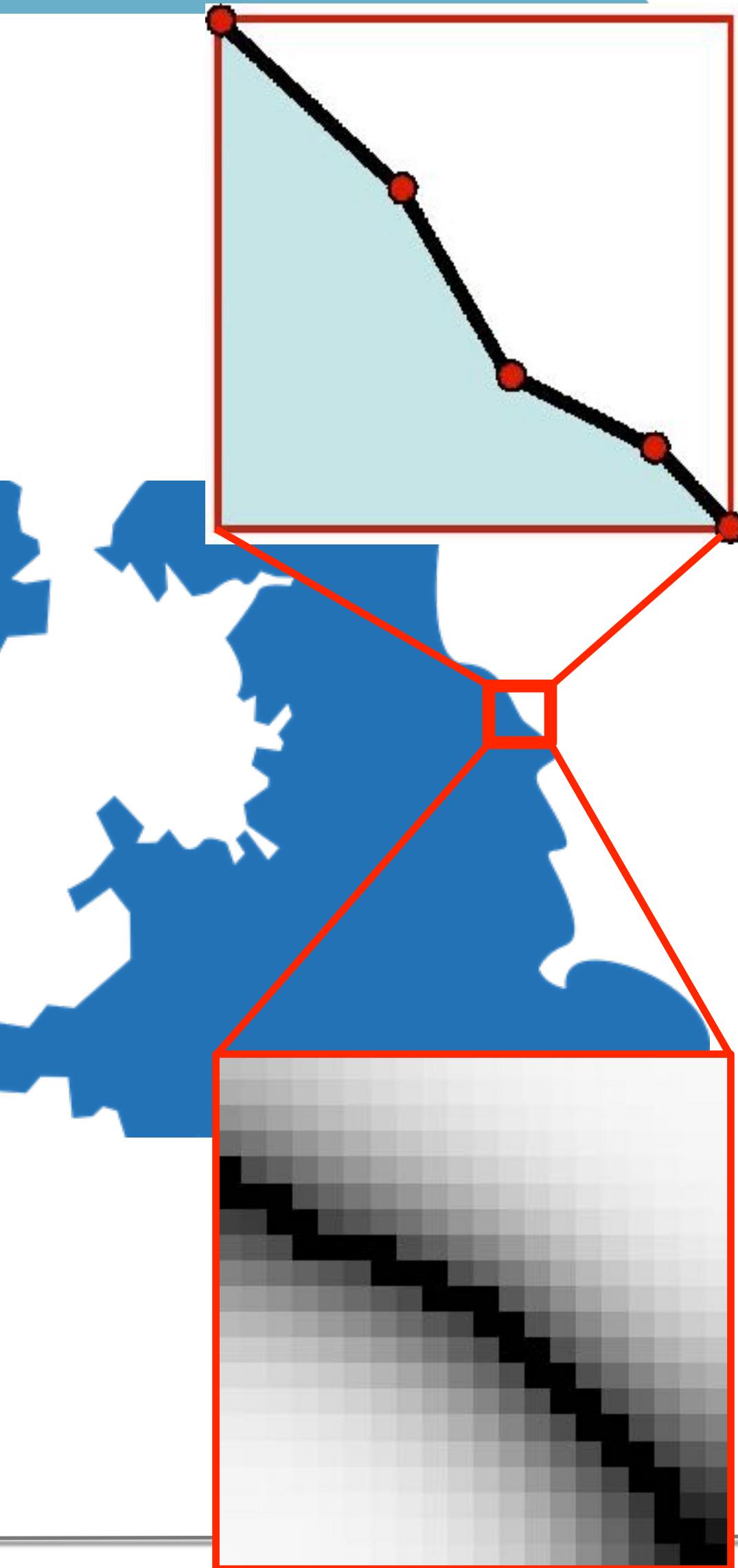
- Implicit:
  - Kernel of implicit function
  - Piecewise approximation!

$$F(x, y) =$$



# Explicit vs. Implicit

- **Explicit:**
  - Range of parametrization function
  - Piecewise approximation!
  - E.g., splines, triangle mesh
  - Easy enumeration
  - Easy geometry modification
- **Implicit:**
  - Kernel of implicit function
  - Piecewise approximation!
  - E.g., scalar-valued 3D grid
  - Easy in/out test
  - Easy topology modification (CSG)



# Overview



- Surface representations
  - Explicit vs. Implicit
- **Explicit representations**
  - Triangle meshes
- Implicit representations
  - Signed distance fields
- Conversions
  - Explicit  $\leftrightarrow$  Implicit

# Polynomial Approximation

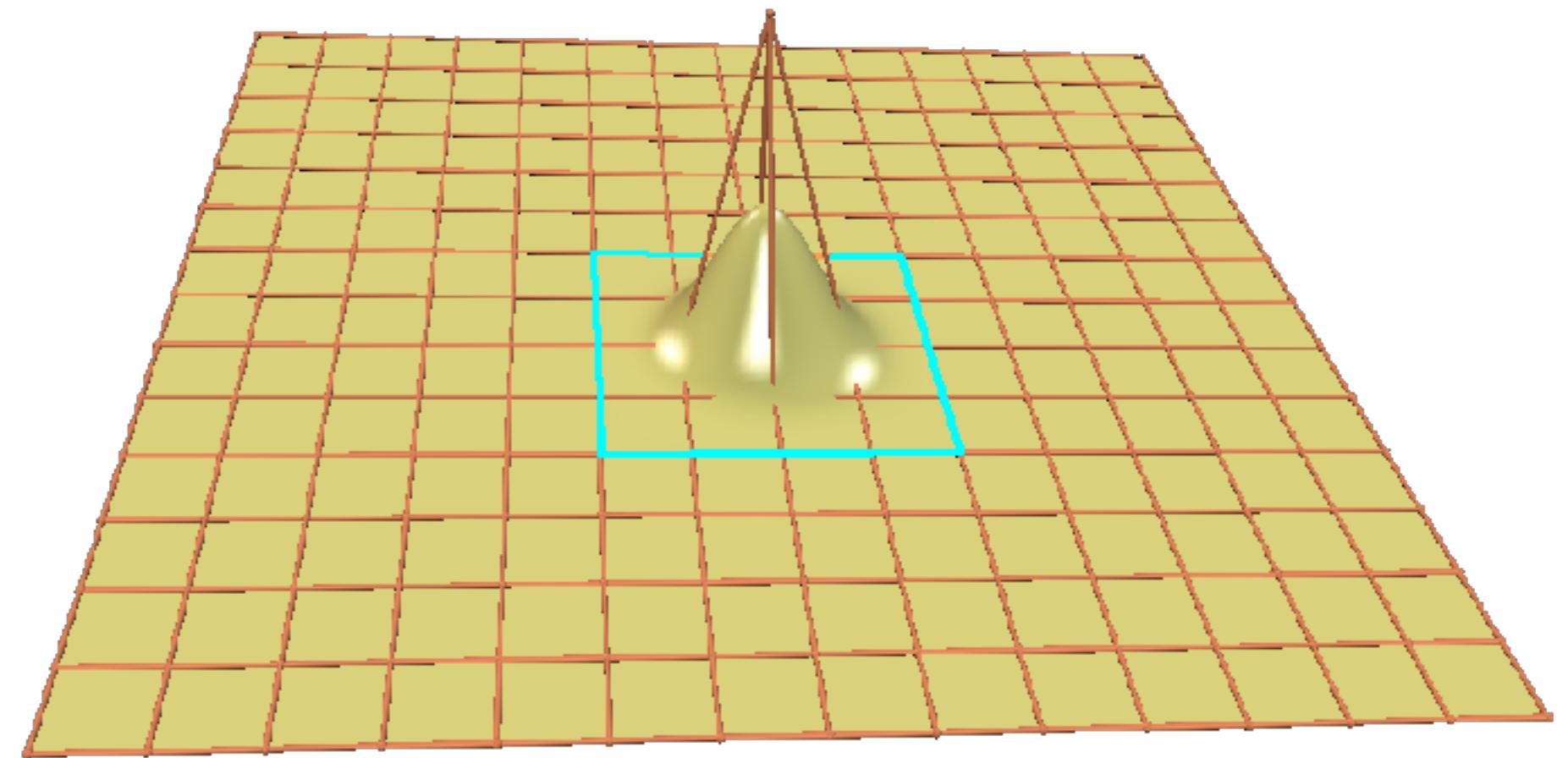


- Polynomials of degree  $p$ , elements of size  $h$ 
  - Approximation error is  $O(h^{p+1})$
- Improve approximation quality by
  - increasing  $p$  ... higher order polynomials
  - decreasing  $h$  ... smaller / more segments
- Issues
  - smoothness of the target data
    - \_ smoothness conditions between segments  $\max_t f^{p+1}(t)$

# Spline Surfaces

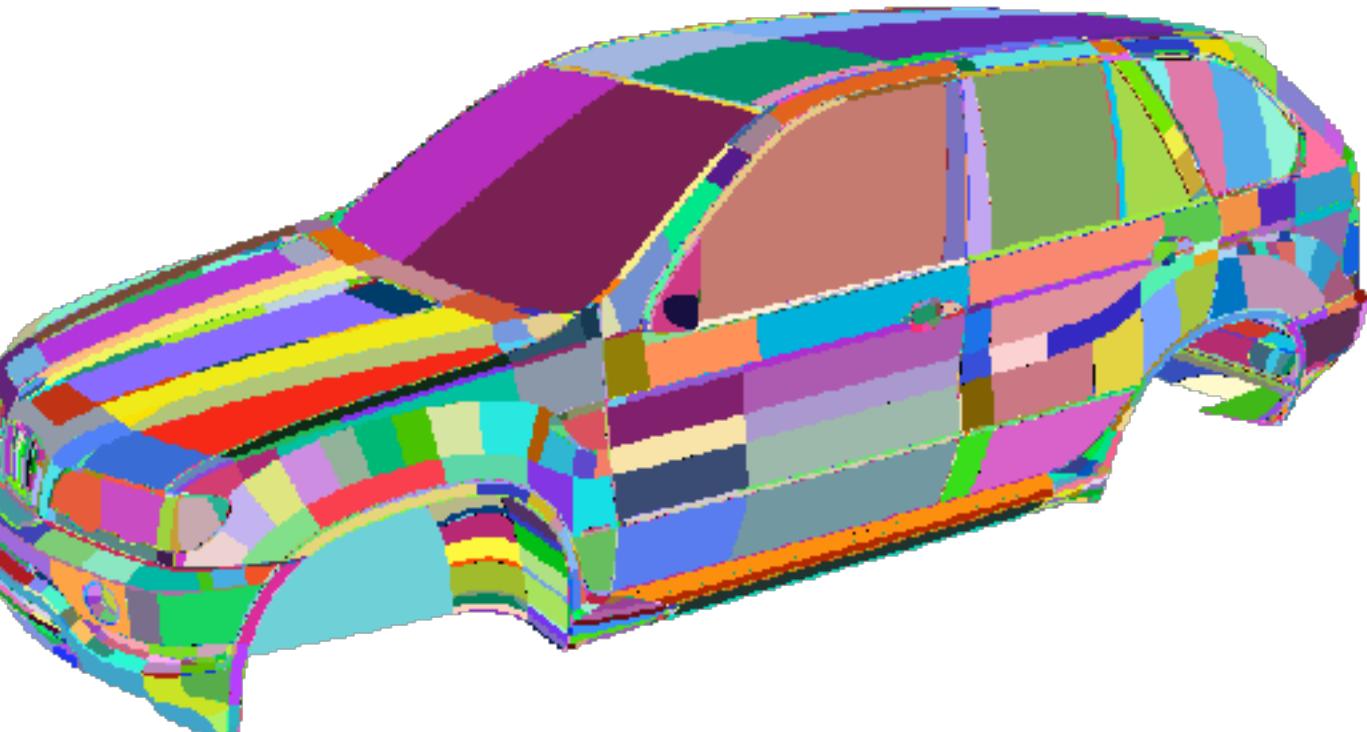
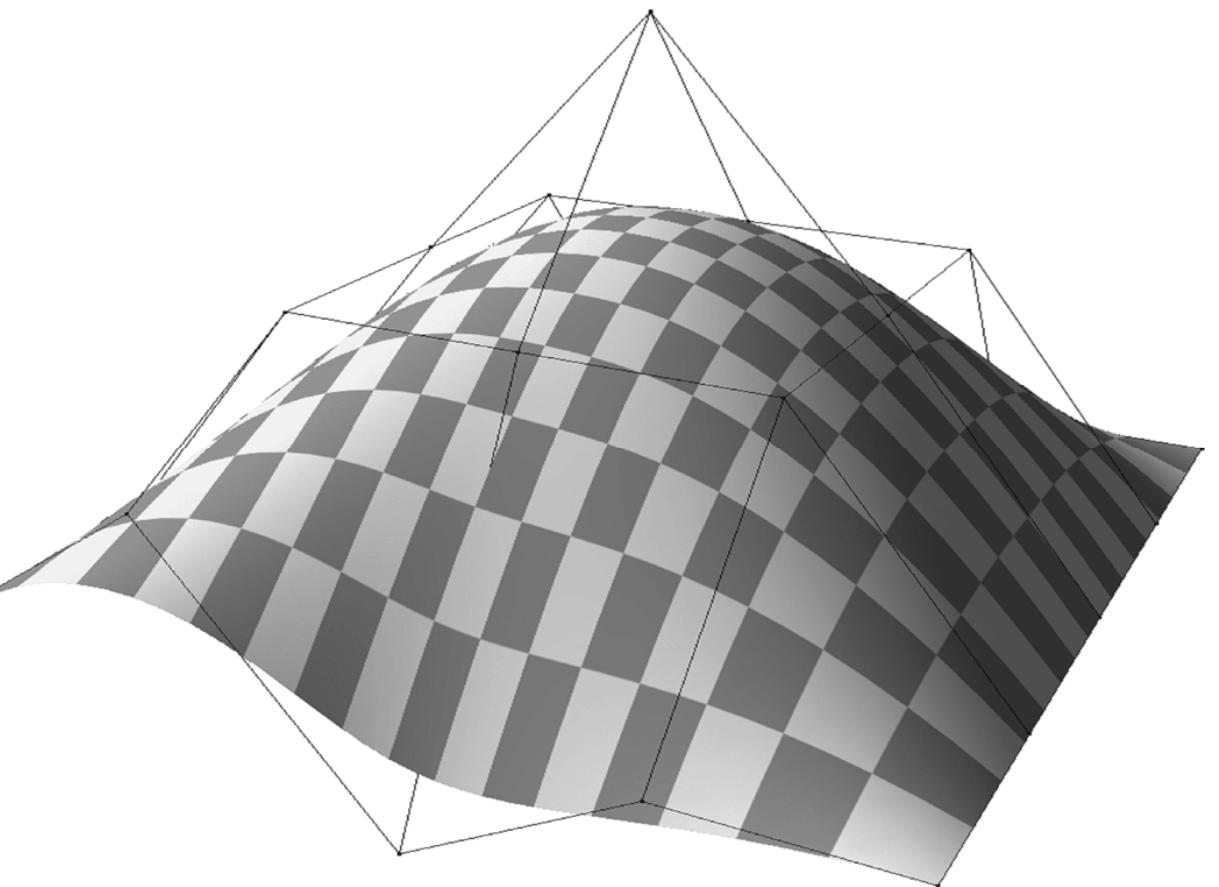
Piecewise polynomial approximation

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



# Spline Surfaces

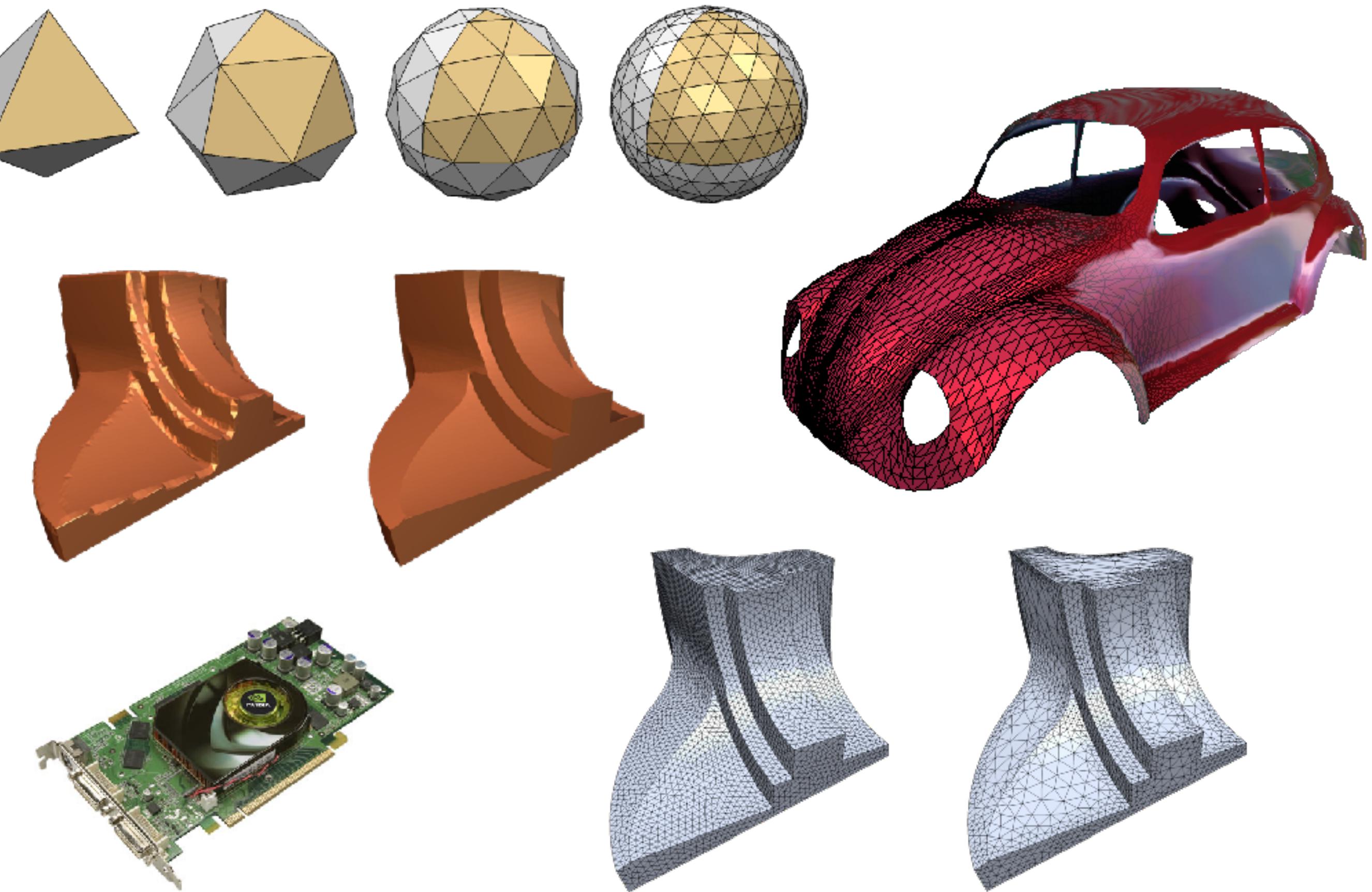
- Piecewise polynomial approximation
- Topological constraints
  - Rectangular patches
  - Regular control mesh
- Geometric constraints
  - Large number of patches
  - Continuity between patches
  - Trimming



# Polygon Meshes

- Polygonal meshes for surface approximation

- approximation  $O(h^2)$
- arbitrary topology
- piecewise smooth surfaces
- adaptive refinement
- efficient rendering



# Triangle Meshes

- **Topology:** vertices, edges, triangles

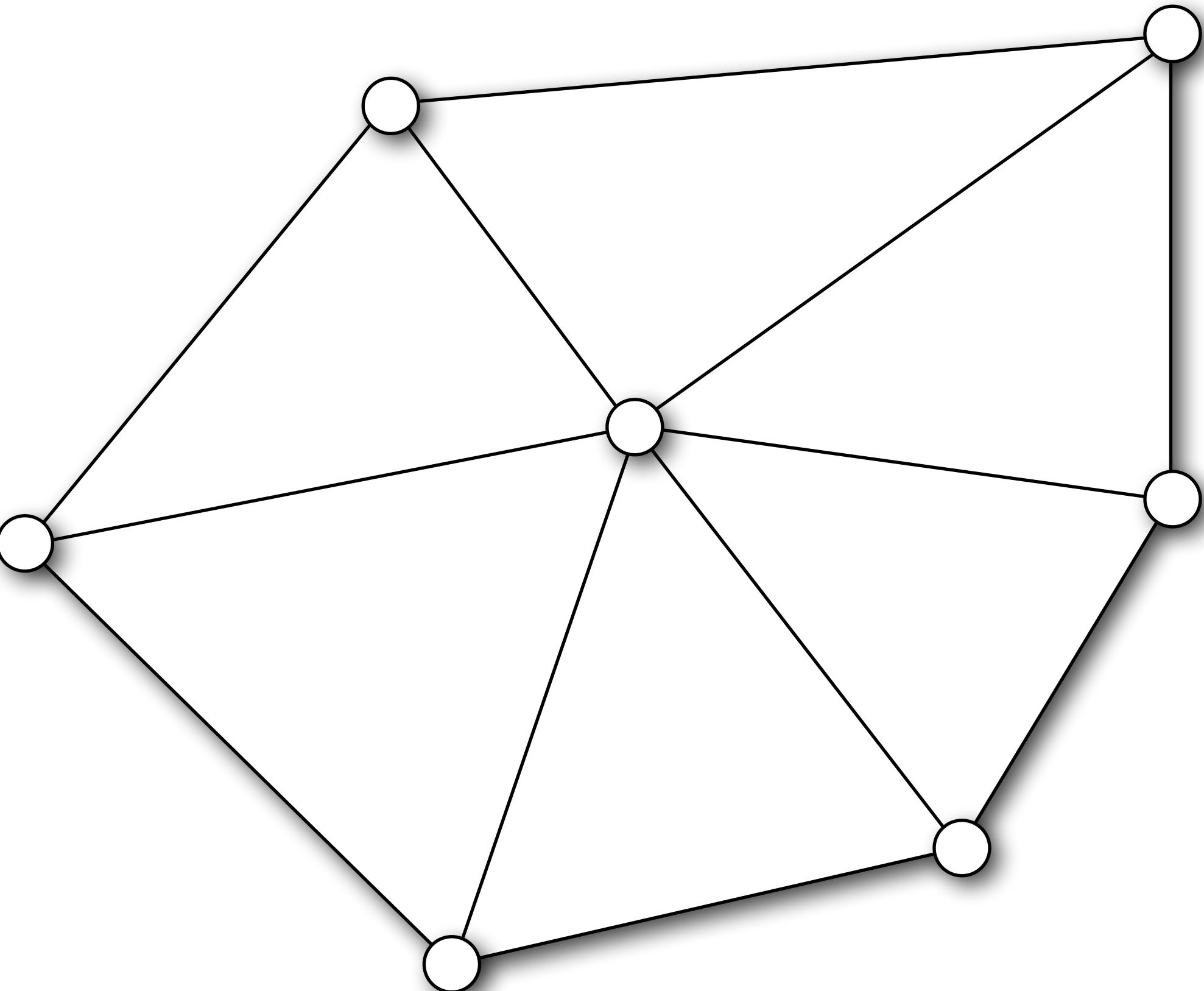
$$\mathcal{V} = \{v_1, \dots, v_n\}$$

$$\mathcal{E} = \{e_1, \dots, e_k\}, \quad e_i \in \mathcal{V} \times \mathcal{V}$$

$$\mathcal{F} = \{f_1, \dots, f_m\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$$

- **Geometry:** vertex positions

$$\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}, \quad \mathbf{p}_i \in \mathbb{R}^3$$



# Overview



- Surface representations
  - Explicit vs. Implicit
- Explicit representations
  - Triangle meshes
- **Implicit representations**
  - Signed distance fields
- Conversions
  - Explicit  $\leftrightarrow$  Implicit

# Implicit vs Explicit Functions



Implicit

Explicit



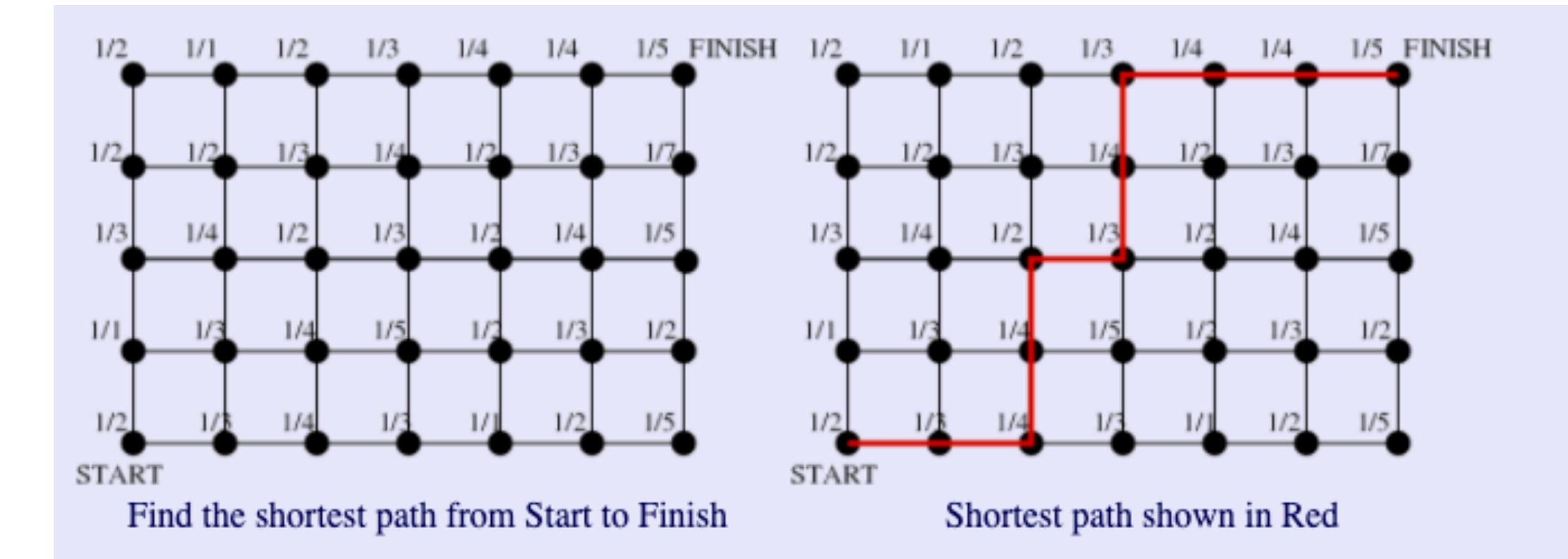
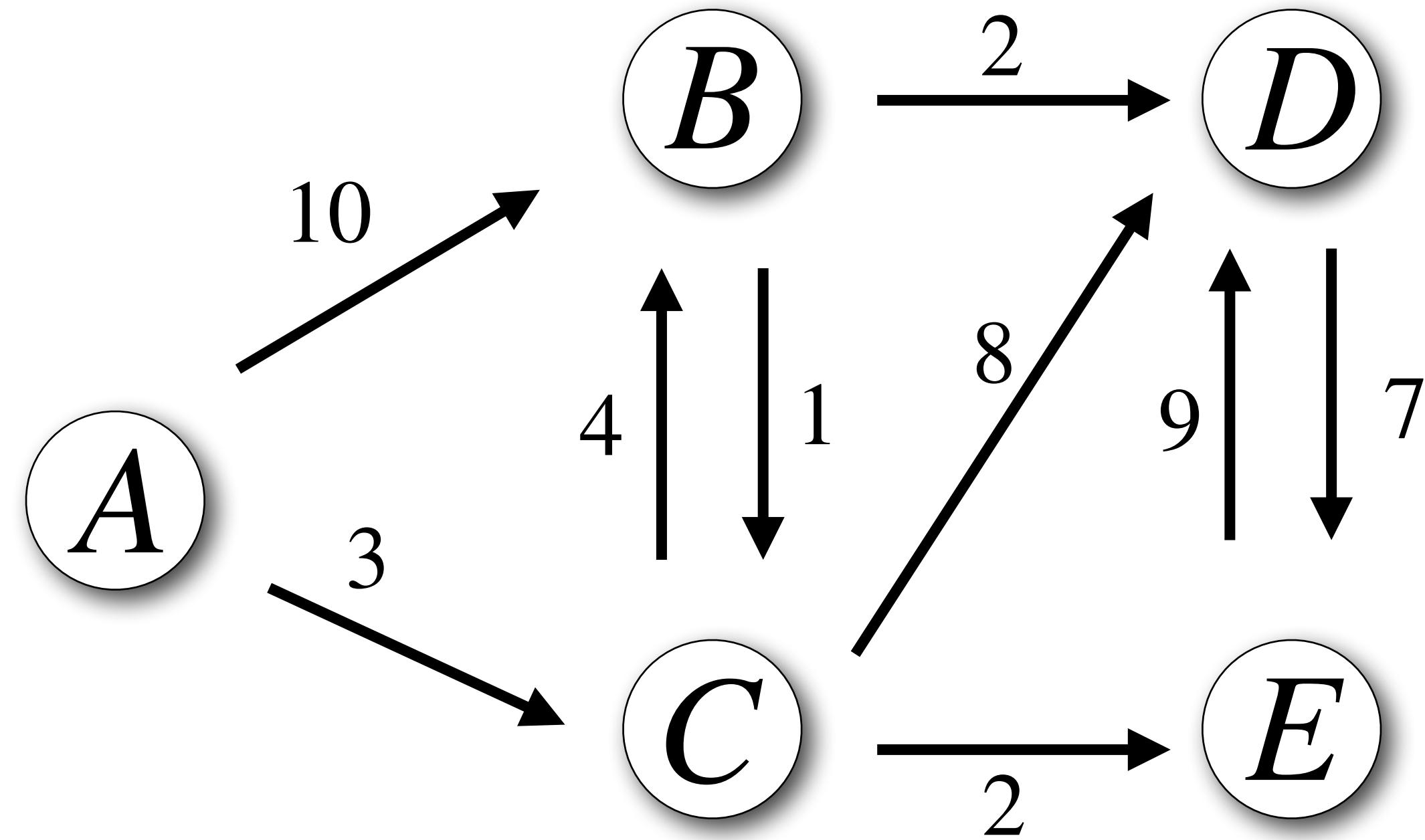
# Implicit → Explicit



# Explicit → Implicit



# Dijkstra's Algorithm (Recap)



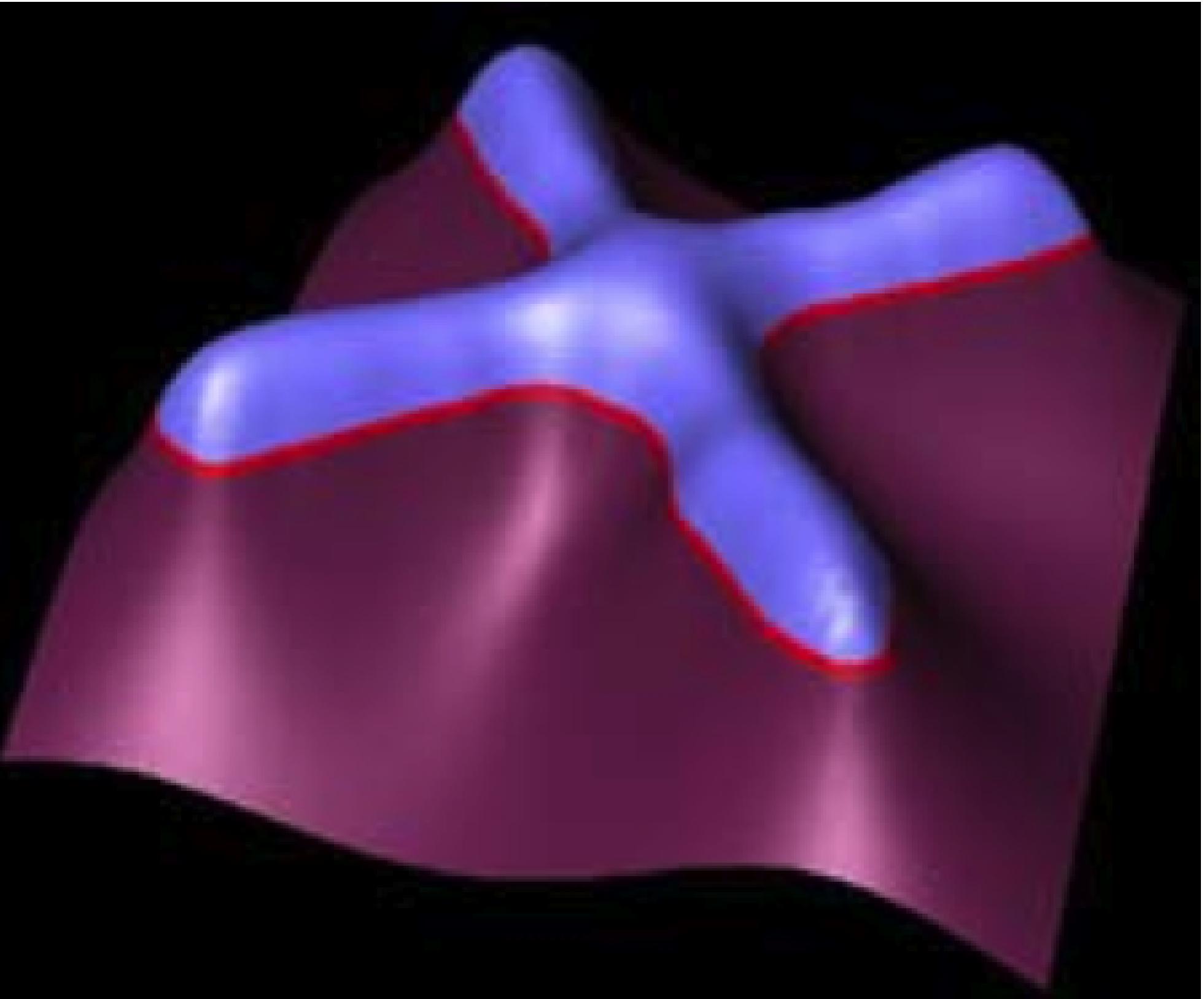
# Explicit → Implicit



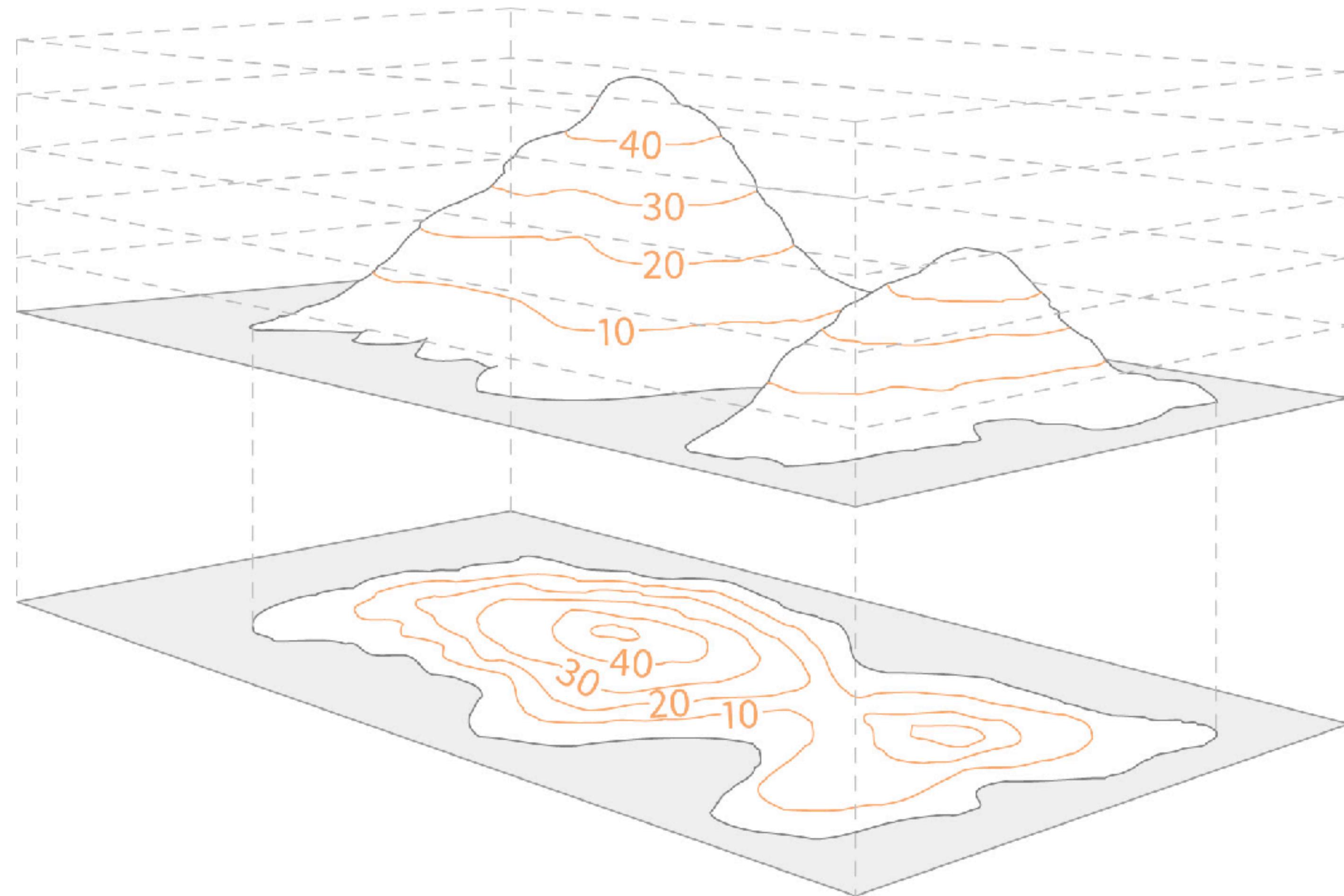
# Implicit Representations



Level set of 2D function defines 1D curve



# Contour Map

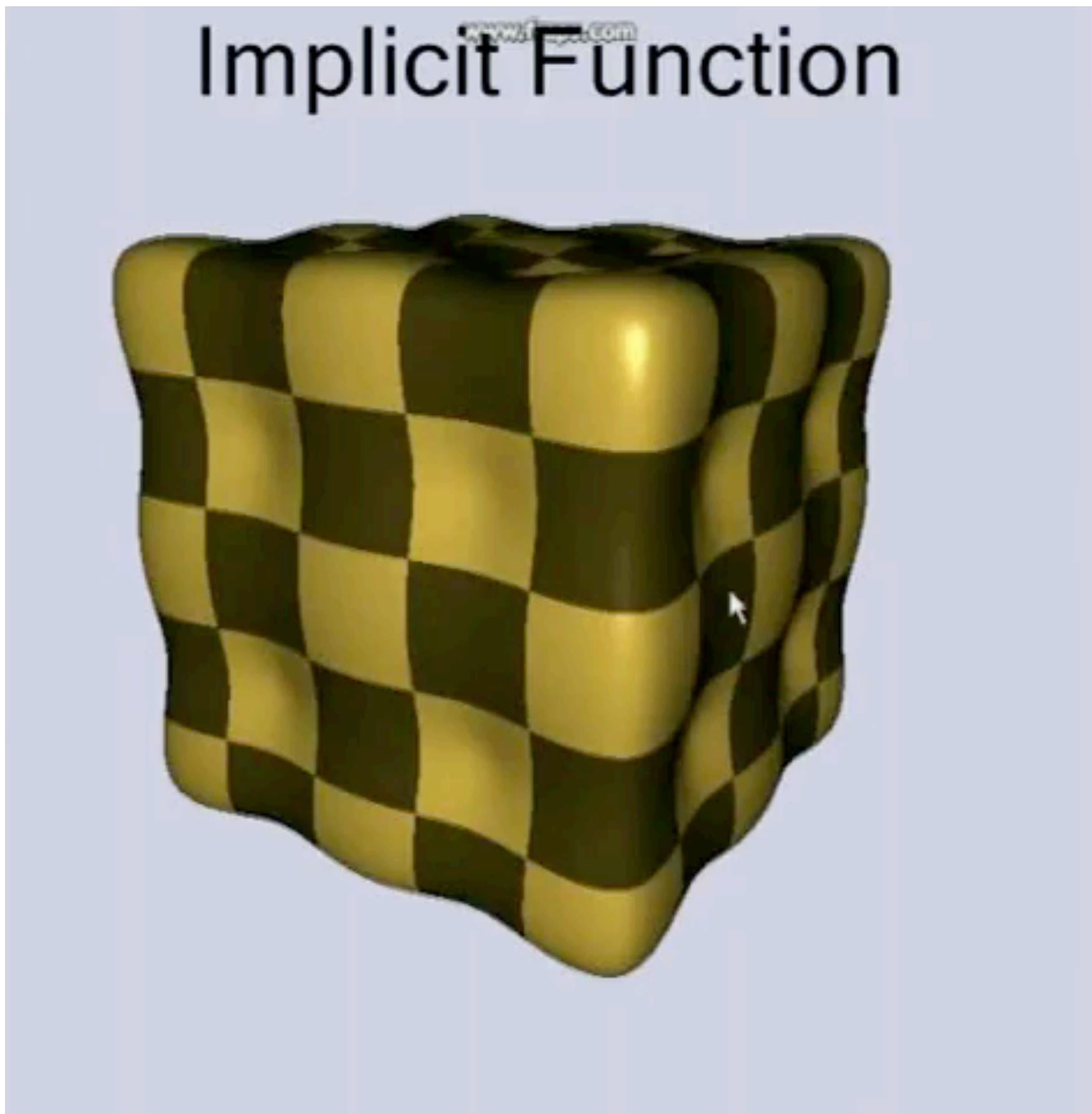


<https://getoutside.ordnancesurvey.co.uk/guides/understanding-map-contour-lines-for-beginners/>

# Implicit Representations



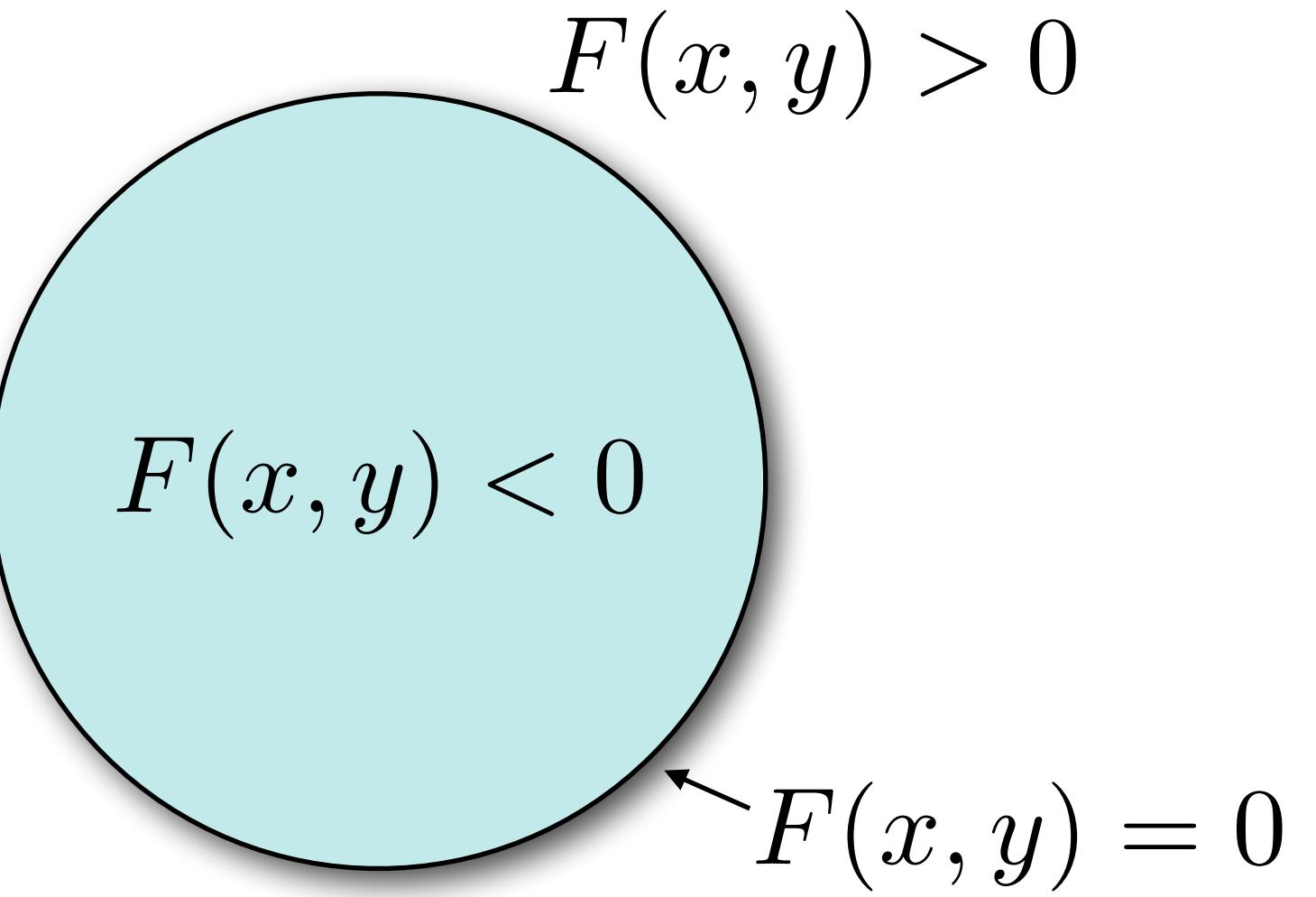
Level set of 3D function defines 2D surface



# 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

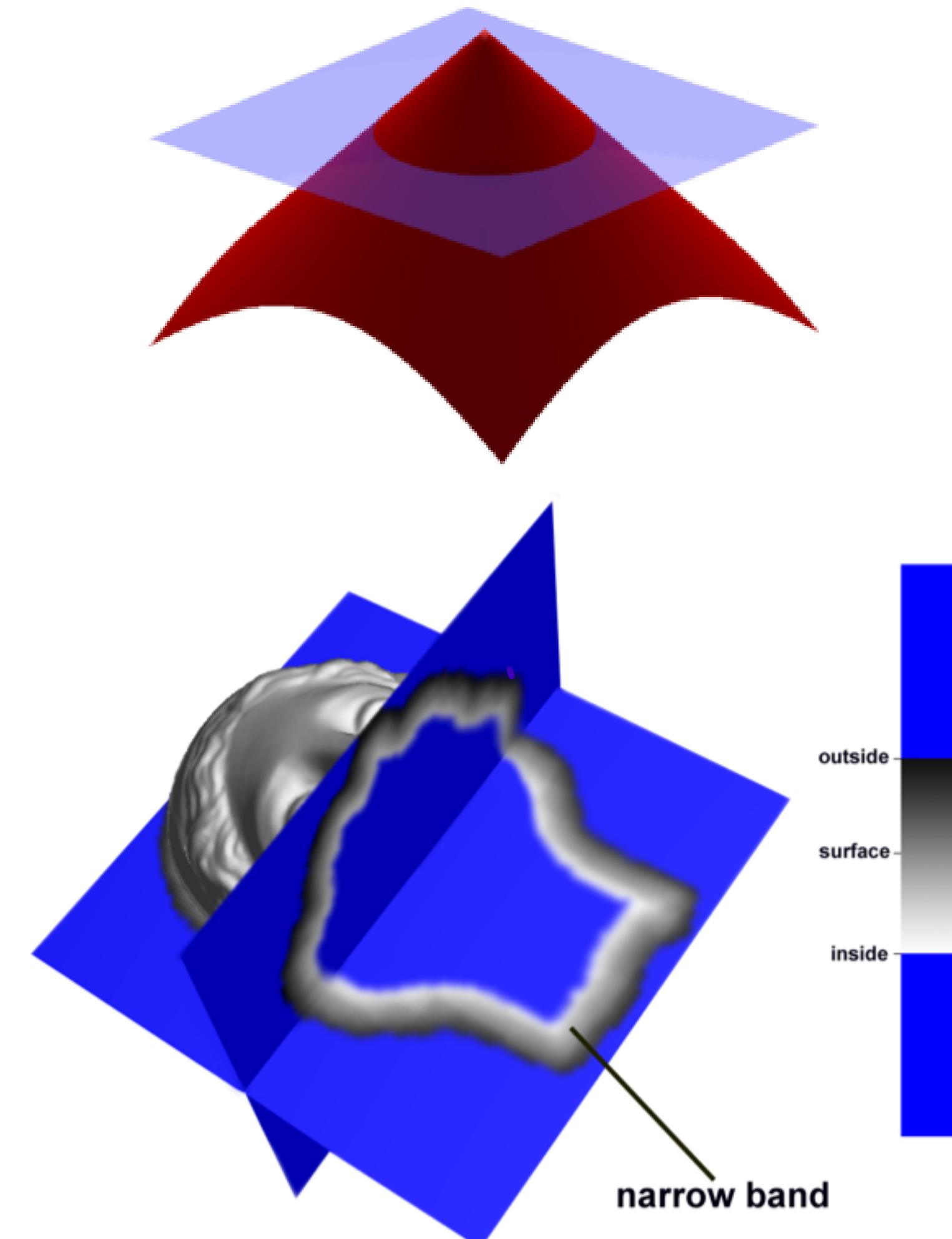


# Signed Distance Function

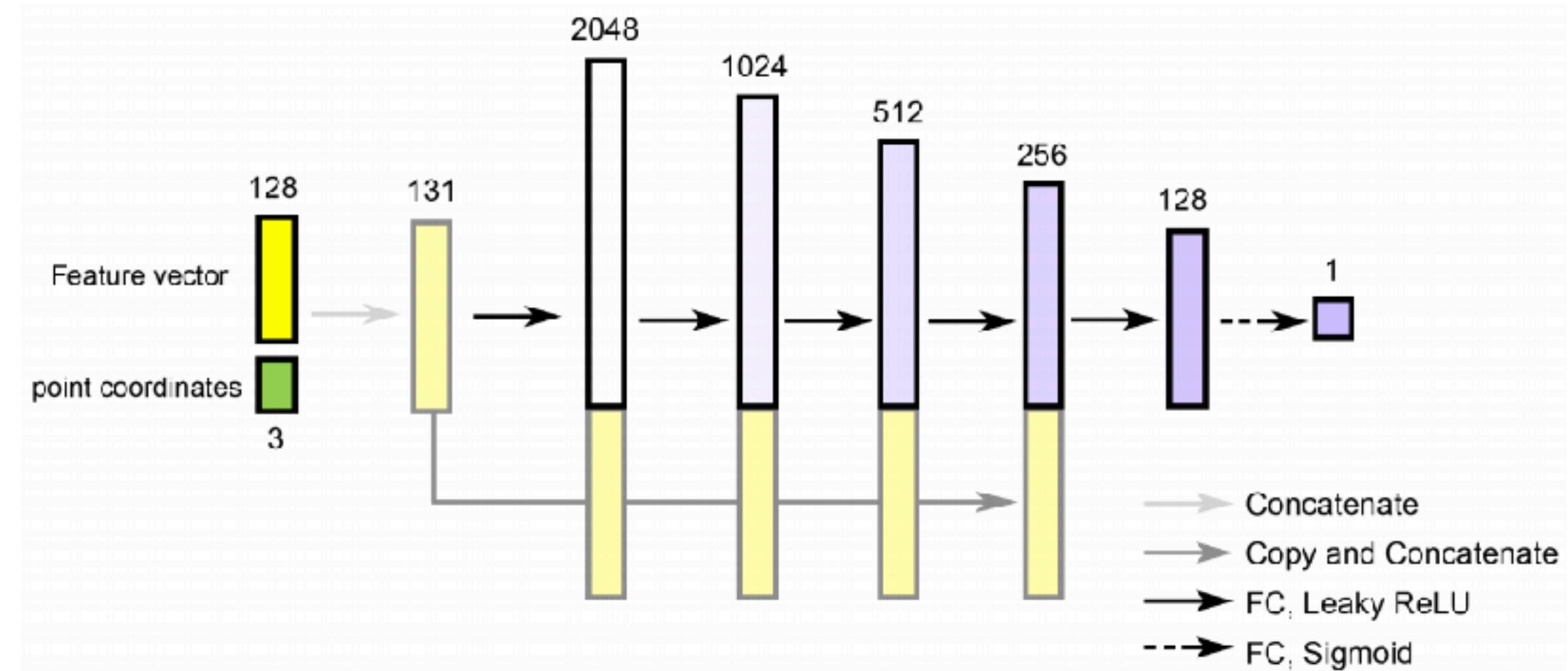
- SDF of 2D circle?
- General shapes

(TSDF)

↓ truncated  
(近づいたときに)

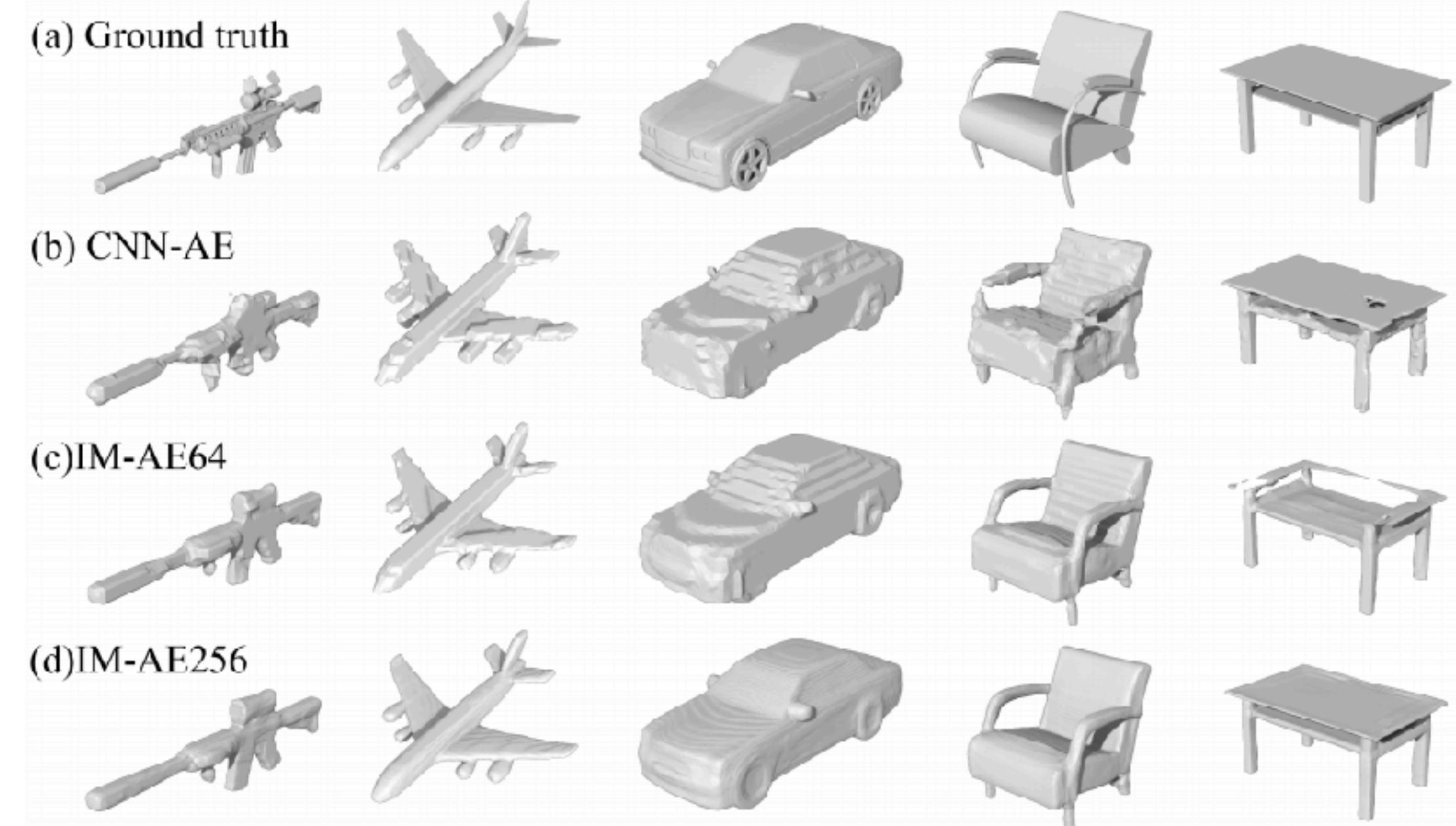


# Learning an SDF



$$f_{\theta}(\mathbf{p}, \mathbf{z})$$

$$\mathbb{R}^{3+f} \rightarrow \mathbb{R}$$

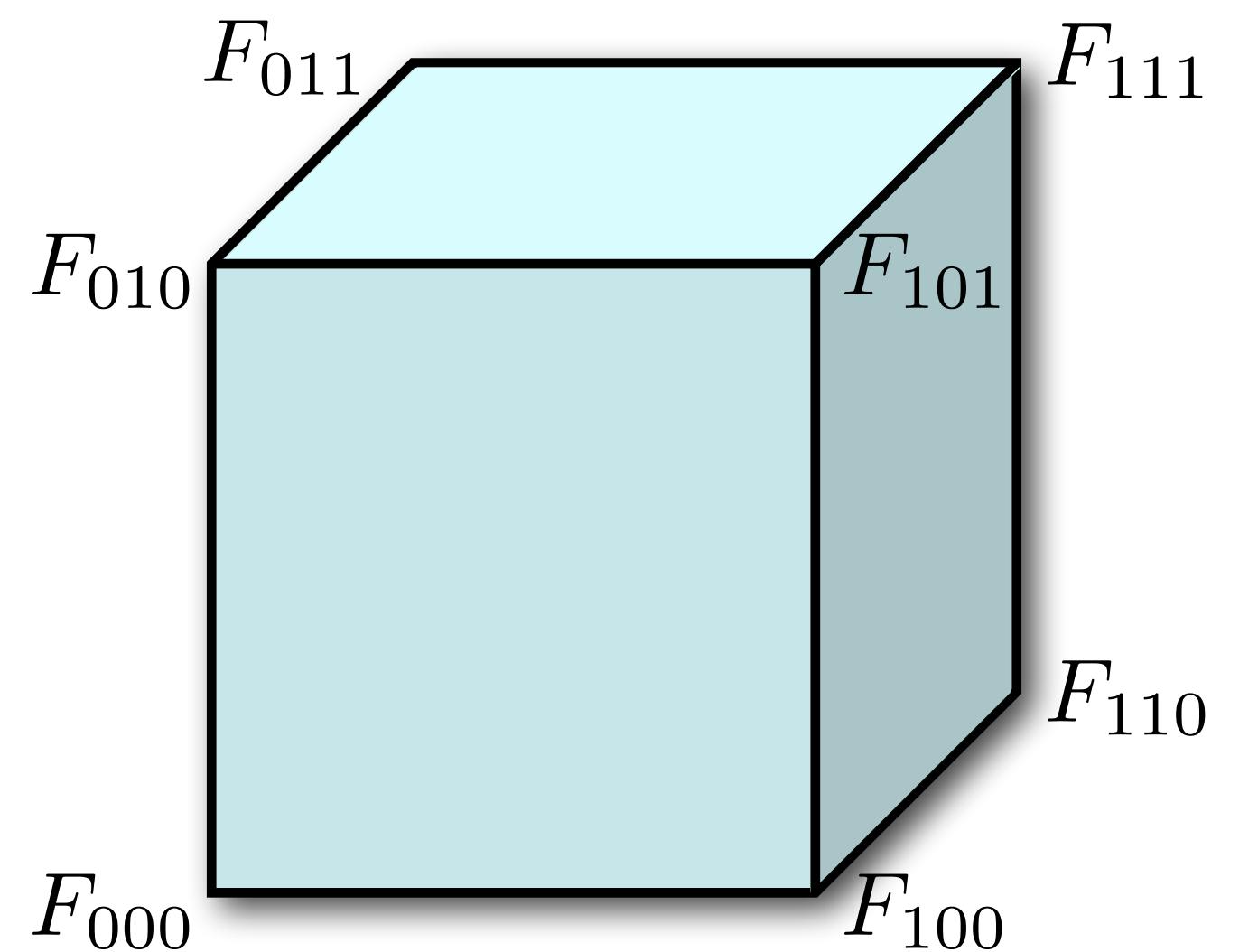


[Chen and Zhang, 2018]

# SDF Discretization

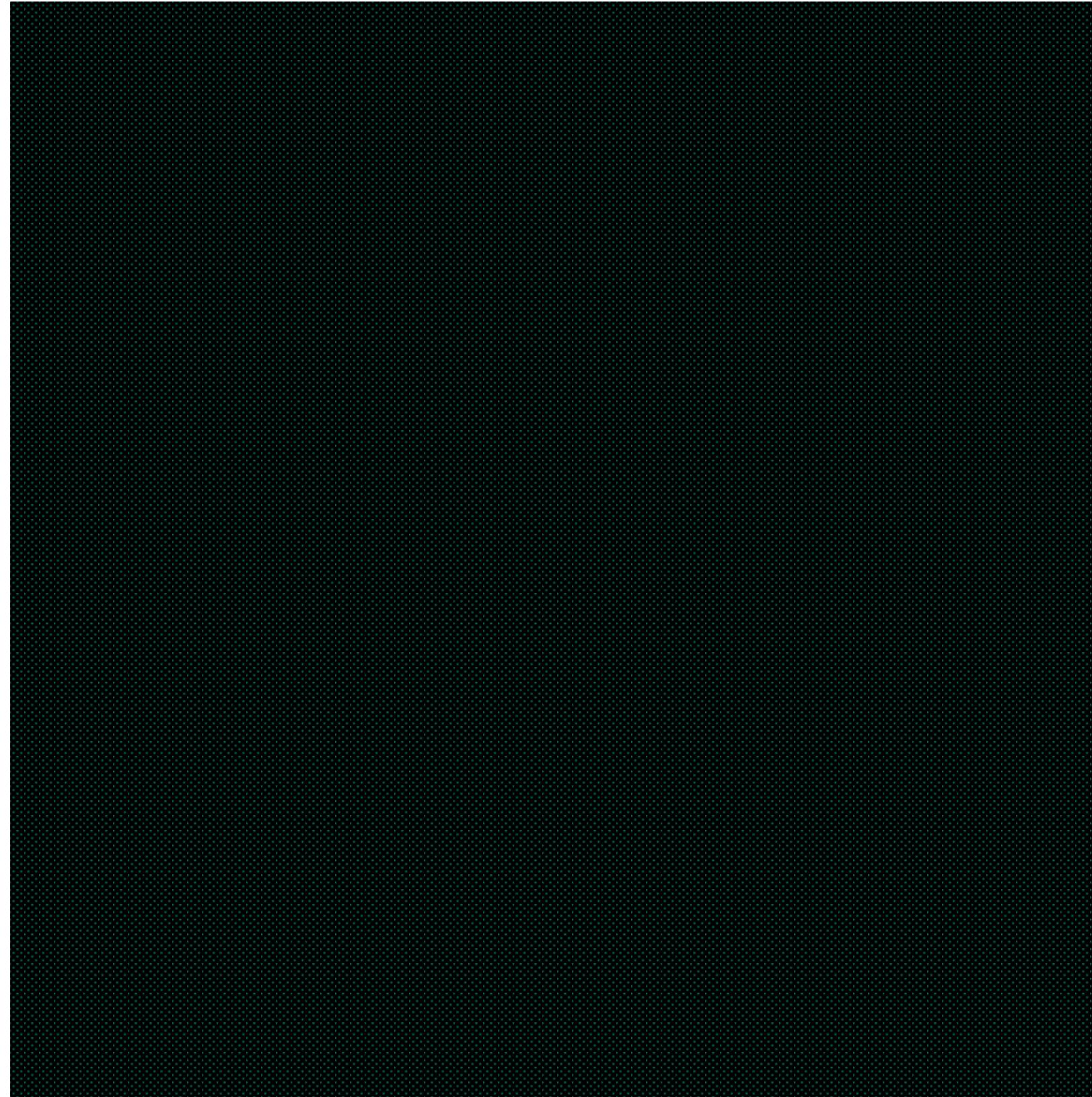


- Regular Cartesian 3D grid
    - Compute signed distance at nodes
    - Tri-linear interpolation within cells

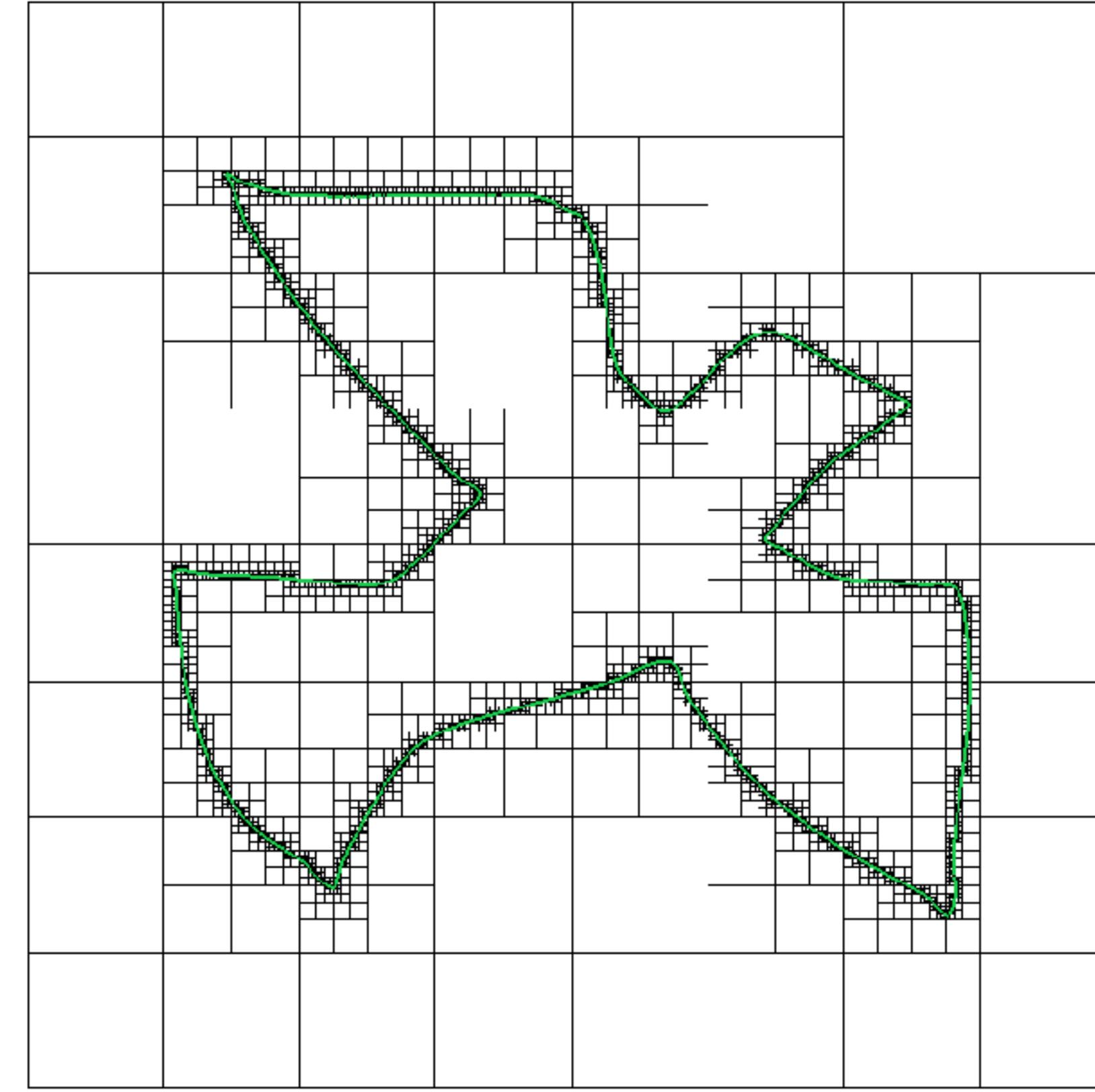


$$\begin{aligned}
F_{000} & \quad (1-u) \quad (1-v) \quad (1-w) \quad + \\
F_{100} & \quad \quad \quad u \quad (1-v) \quad (1-w) \quad + \\
F_{010} & \quad (1-u) \quad \quad \quad v \quad (1-w) \quad + \\
F_{001} & \quad (1-u) \quad (1-v) \quad \quad \quad w \quad + \\
& \vdots \\
F_{111} & \quad \quad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

# 3-Color Octree

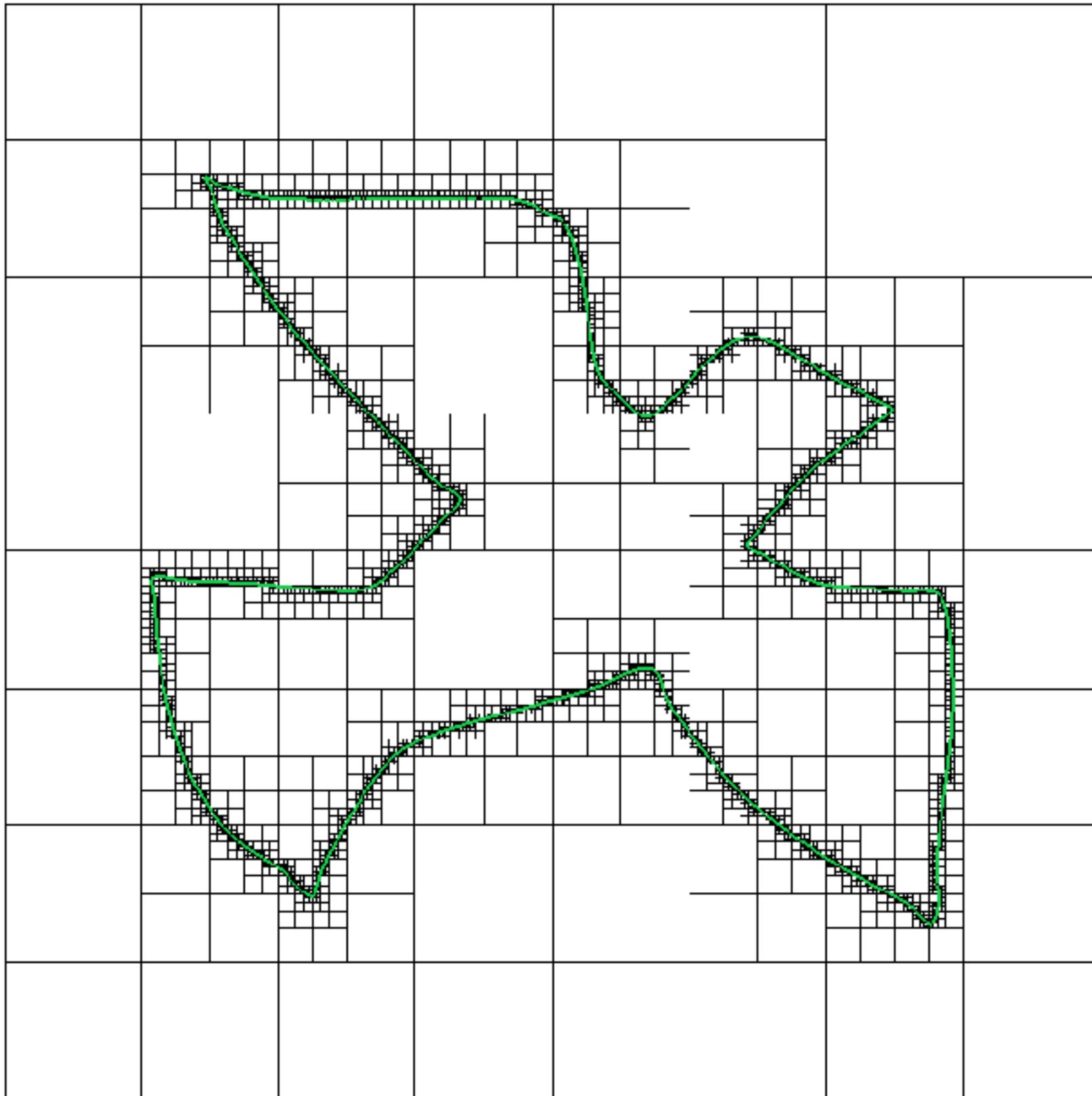


104856 cells

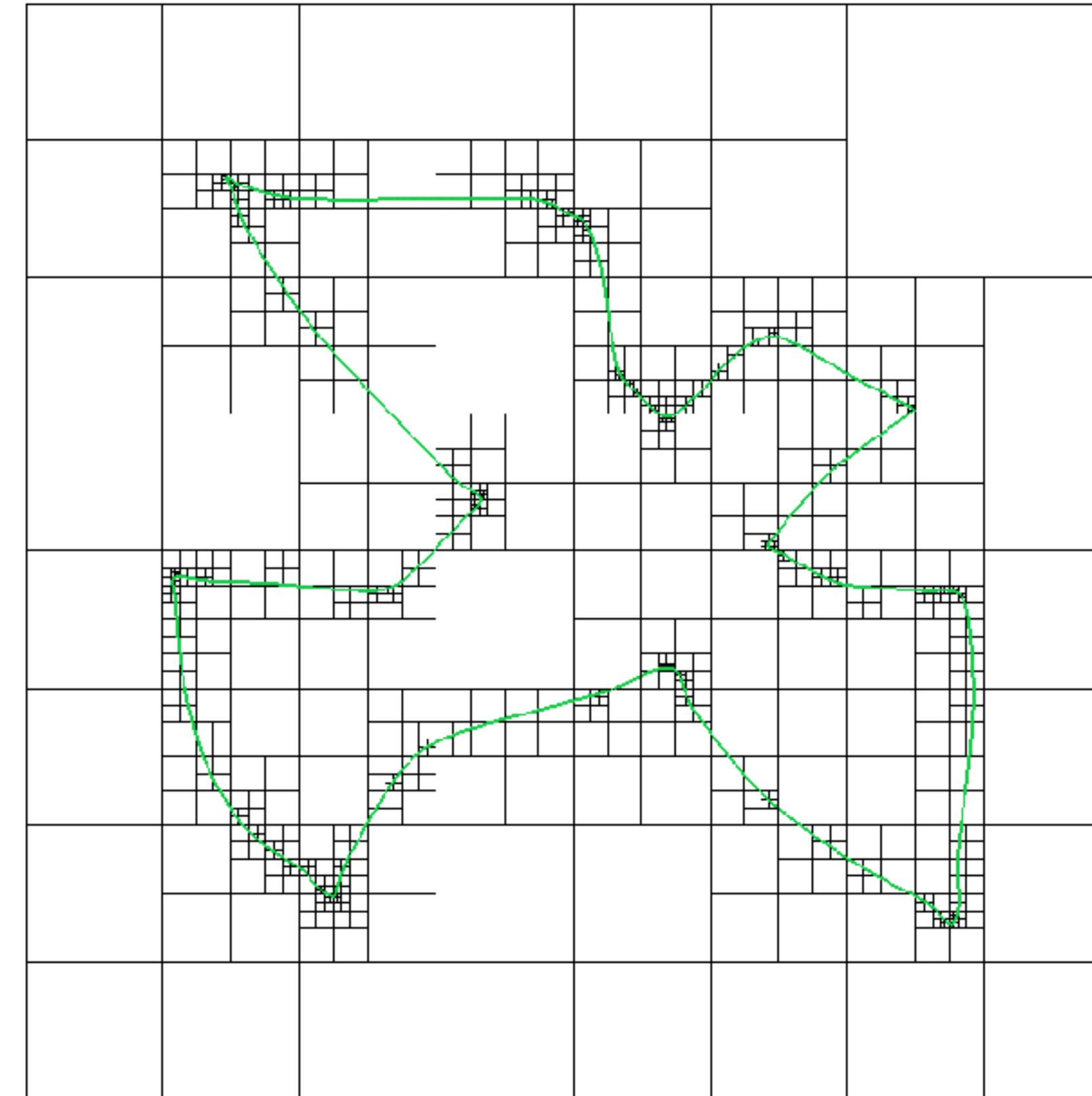


12040 cells

# Adaptively Sampled Distance Fields

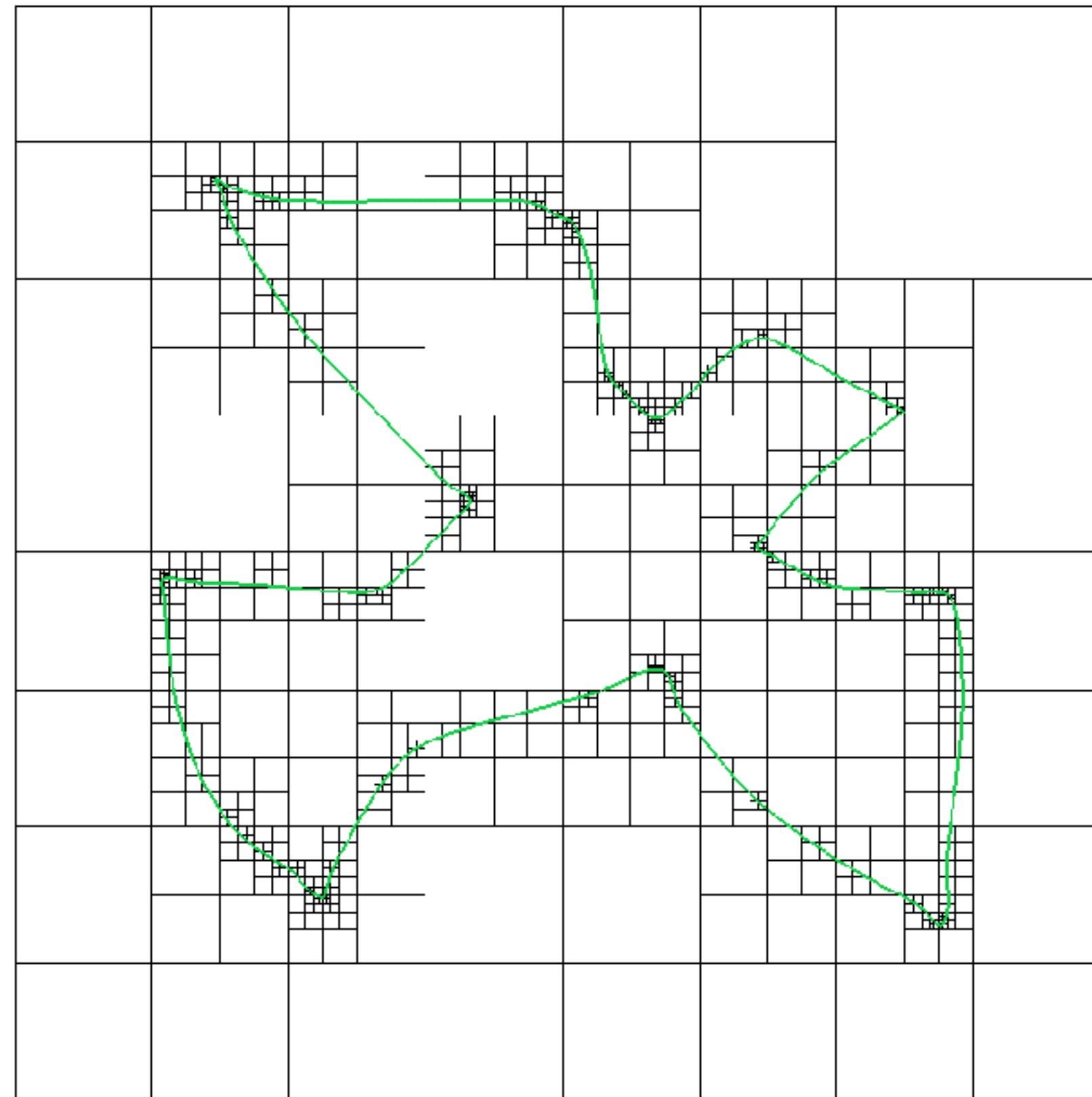


12040 cells

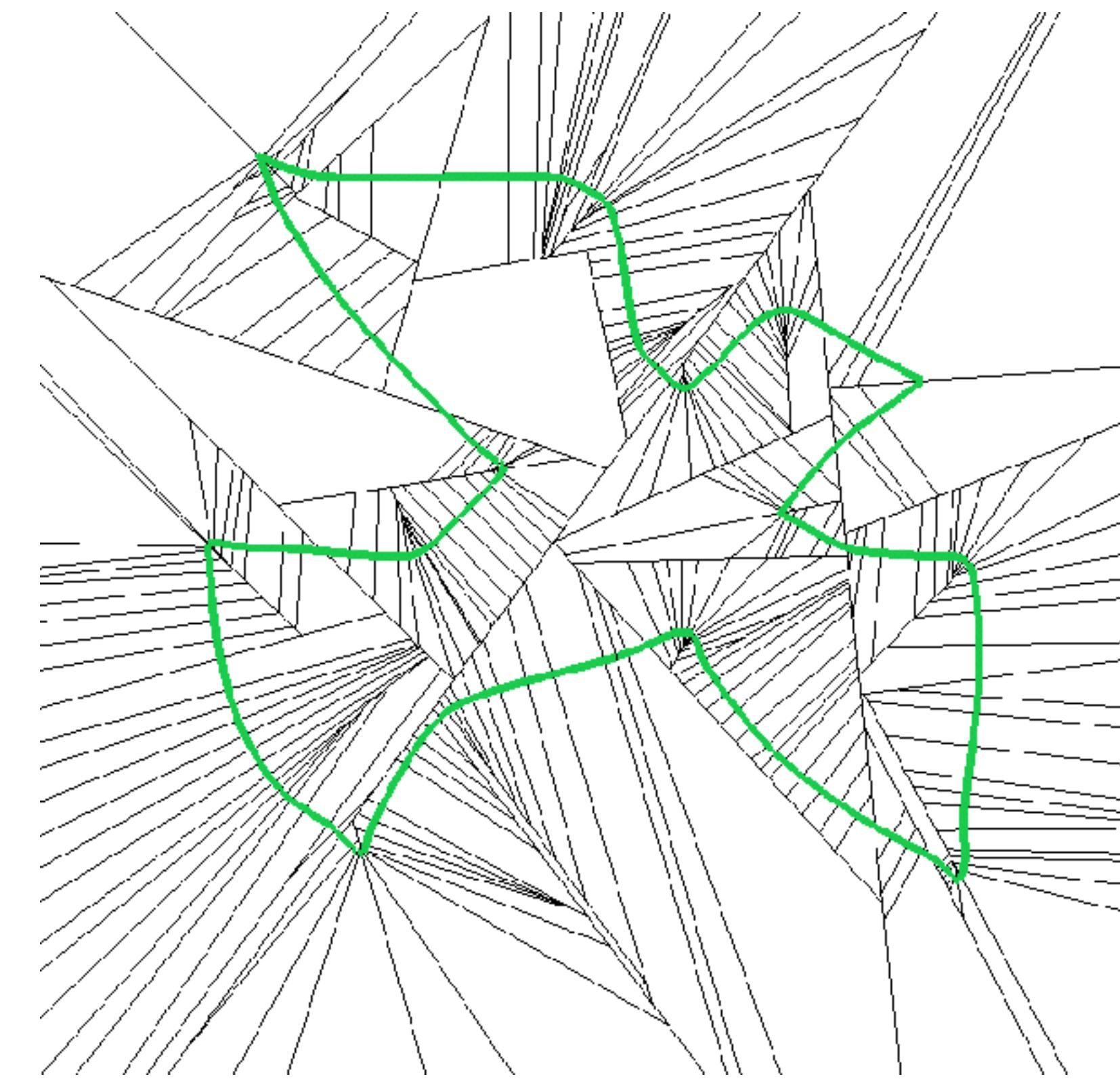


895 cells

# Binary Space Partitions



895 cells



254 cells

# Regularity vs. Complexity



- Implicit surface discretizations
  - Uniform, regular voxel grids
  - Adaptive, 3-color octrees
    - surface-adaptive refinement
    - feature-adaptive refinement
  - Irregular hierarchies
    - binary space partition (BSP)

# 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



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

# Implicit Representations



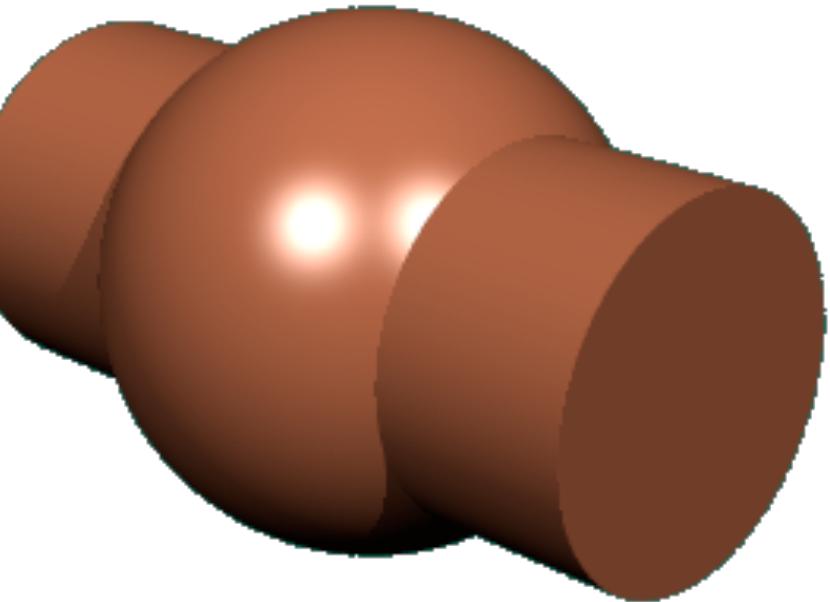
- Natural representation for volumetric data,  
e.g., CT scans, density fields, etc.
- Advantageous when modeling shapes with complex and/or changing topology (e.g., fluids)
- Very suitable representation for *Constructive Solid Geometry* (CSG)

# Constructive Solid Geometry



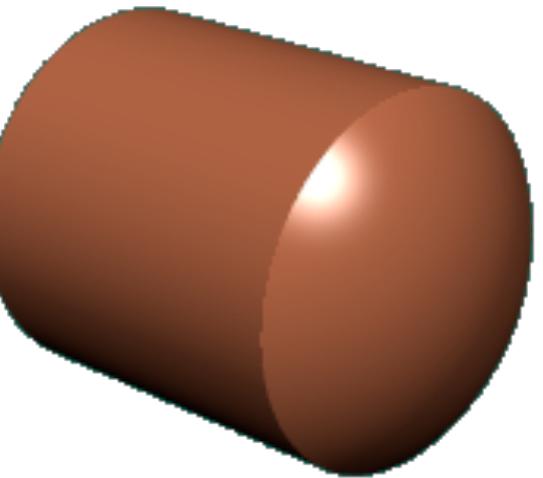
- Union

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



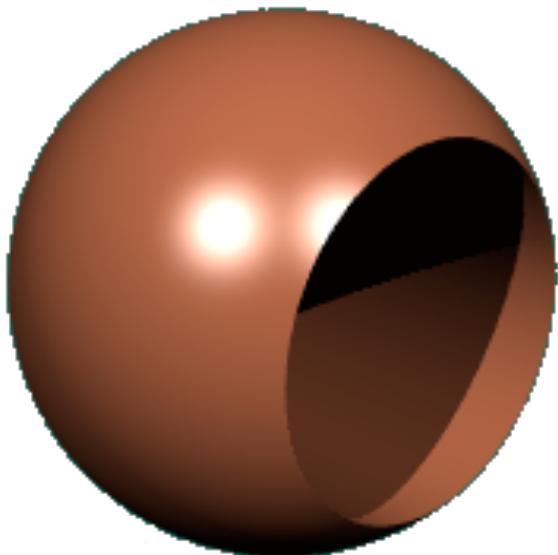
- Intersection

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



- Difference

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

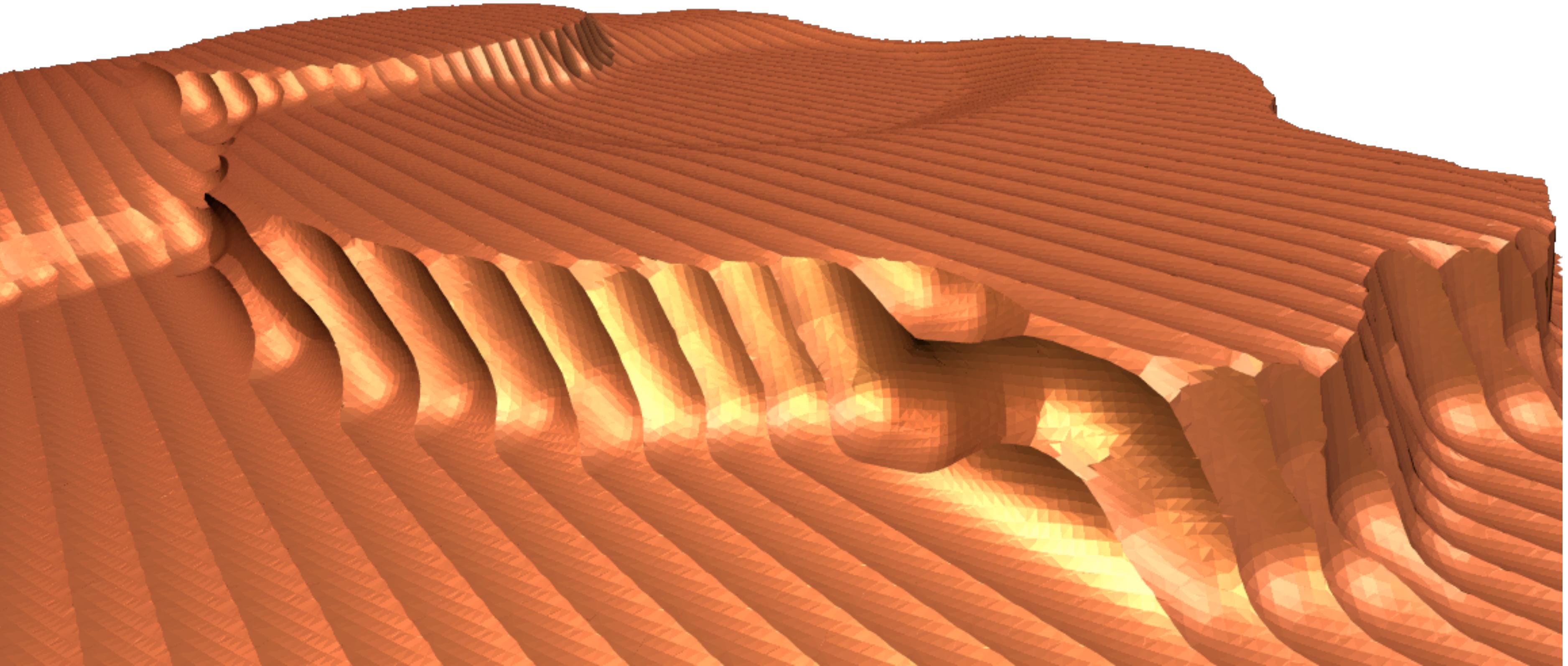


# CSG Example: Milling

- Union

- Intersection

- Difference



# Overview



- Surface representations
  - Explicit vs. Implicit
- Explicit representations
  - Triangle meshes
- Implicit representations
  - Signed distance fields
- **Conversions**
  - Explicit  $\leftrightarrow$  Implicit

# Conversions



- **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
  - See <http://www.geometrictools.com>
- Inside or outside?
  - Based on interpolated surface normals

# Signed Distance Heuristic



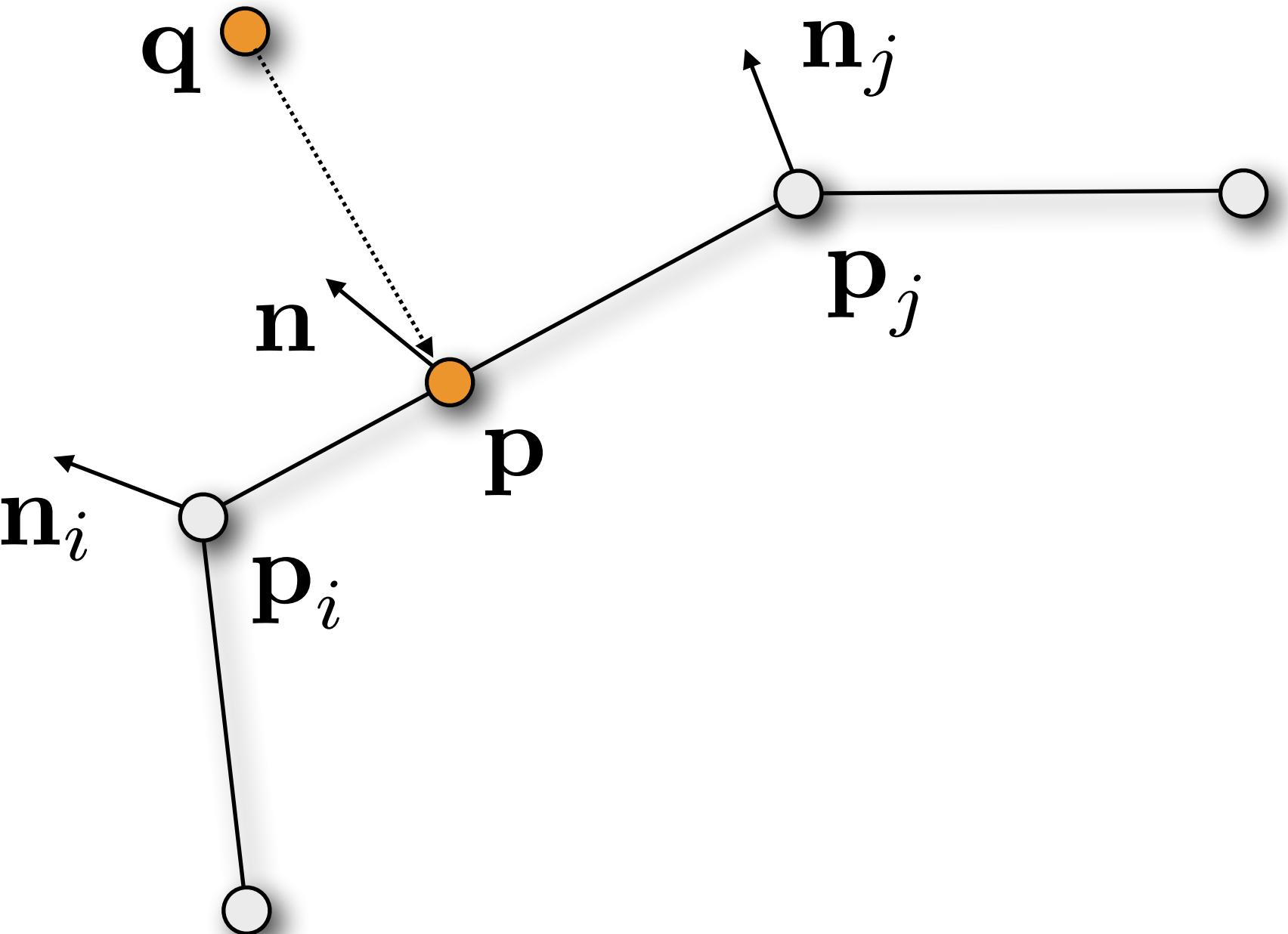
1. Closest point

$$\mathbf{p} = \alpha \mathbf{p}_i + (1 - \alpha) \mathbf{p}_j$$

2. Interpolated normal  $\mathbf{n} = \alpha \mathbf{n}_i + (1 - \alpha) \mathbf{n}_j$

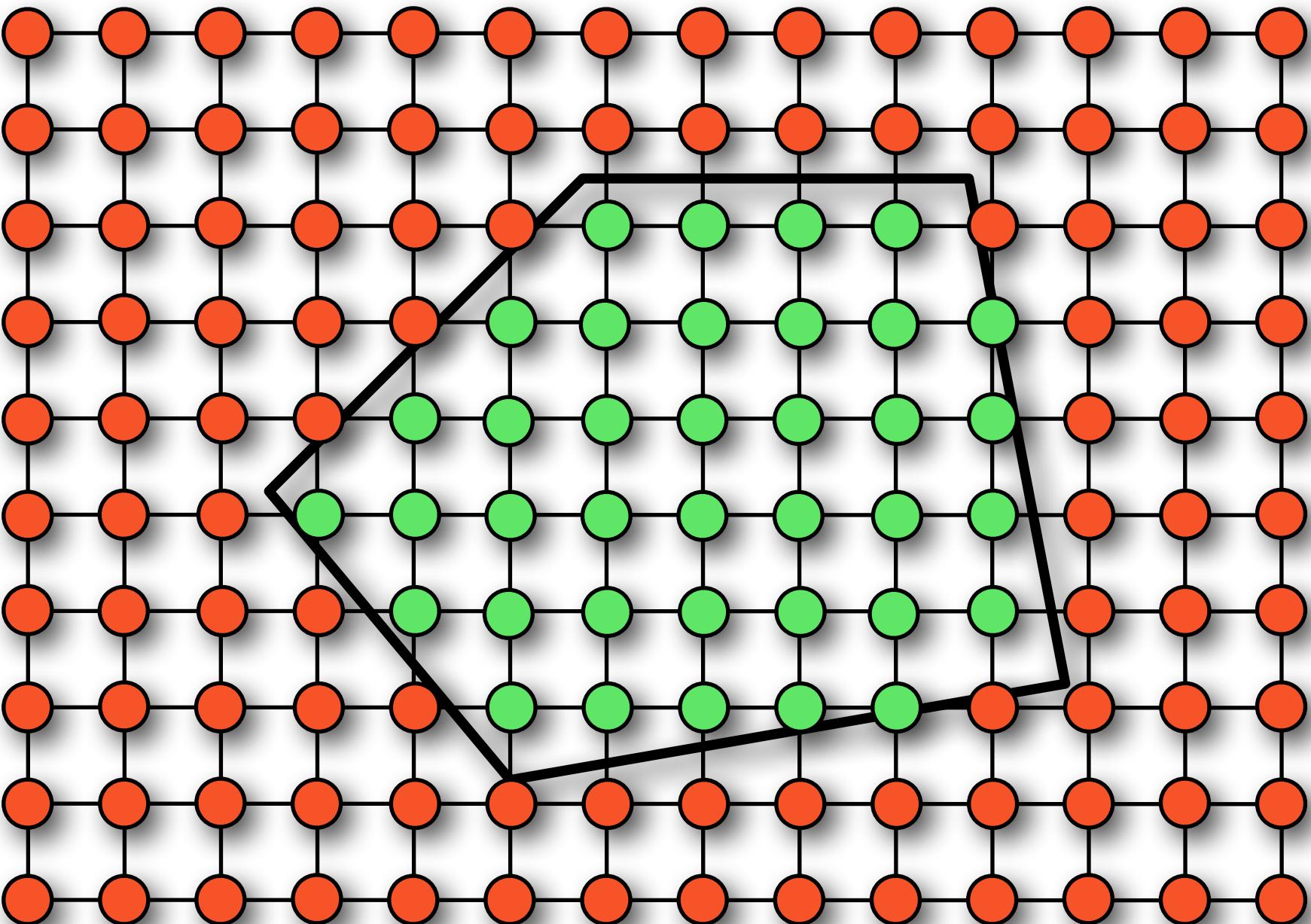
3. Inside if

$$(\mathbf{q} - \mathbf{p})^T \mathbf{n} < 0$$



# Fast Marching Techniques

1. Initialize with exact distance in mesh's vicinity
2. Fast-march outwards
3. 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

# Conversions



- 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 Squares



1. Classify grid nodes as inside/outside

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

2. Classify cell:  $2^4$  configurations

- In/out for each corner

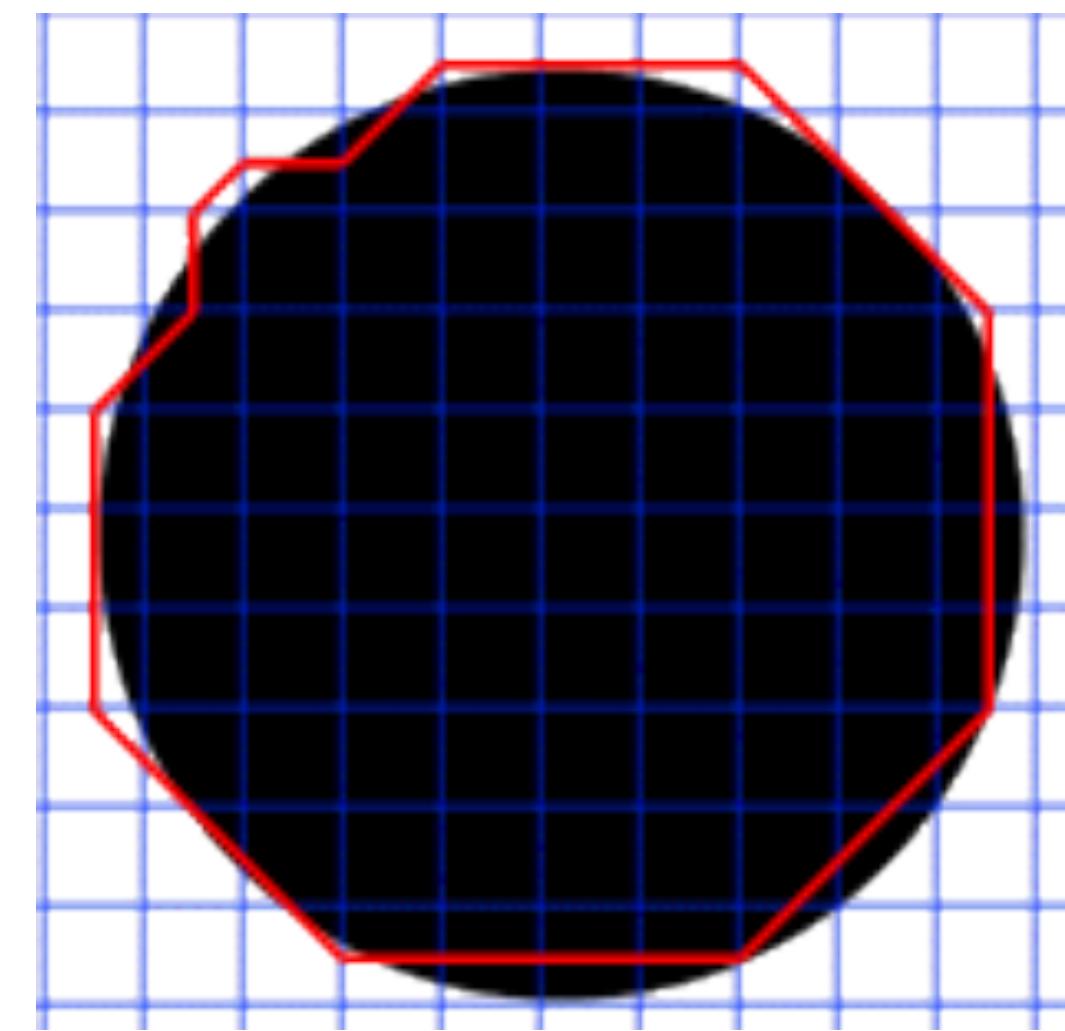
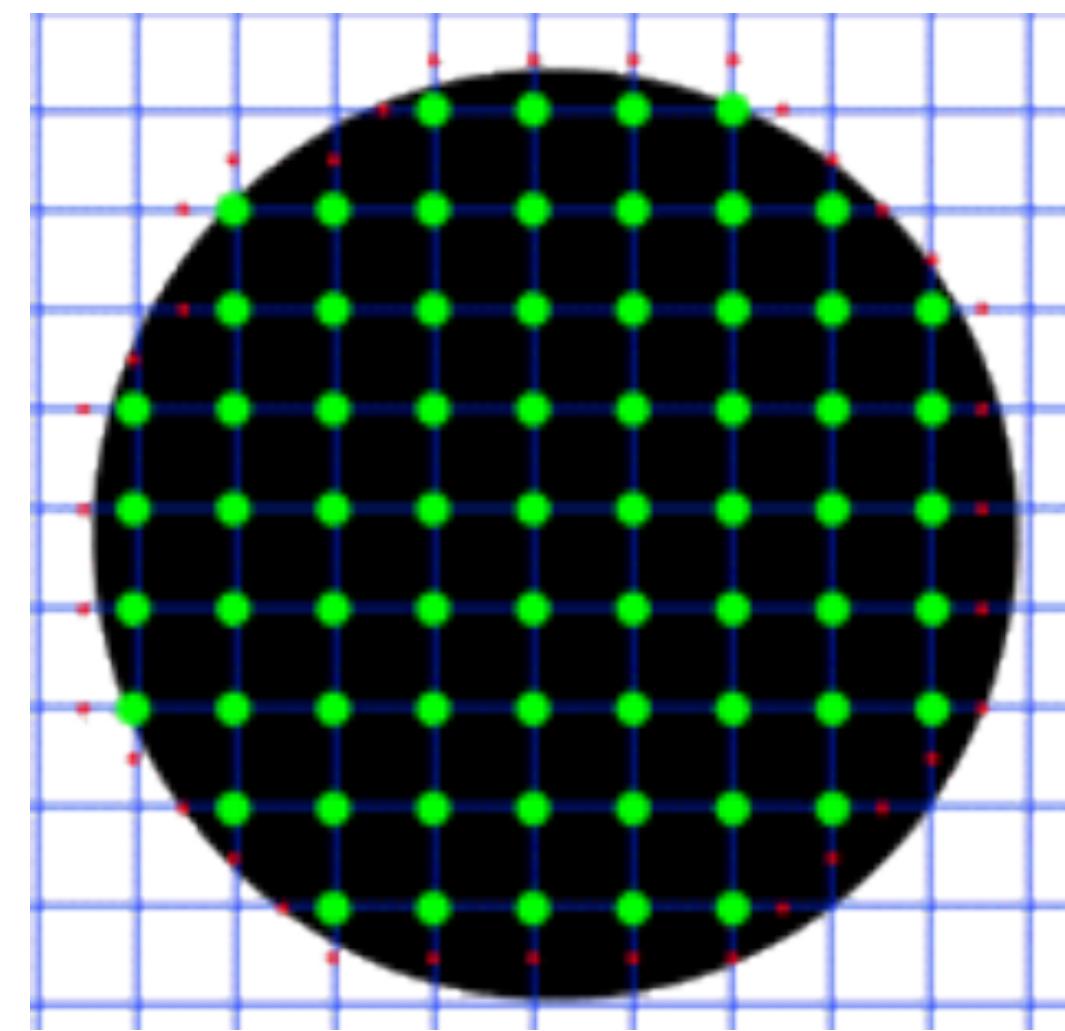
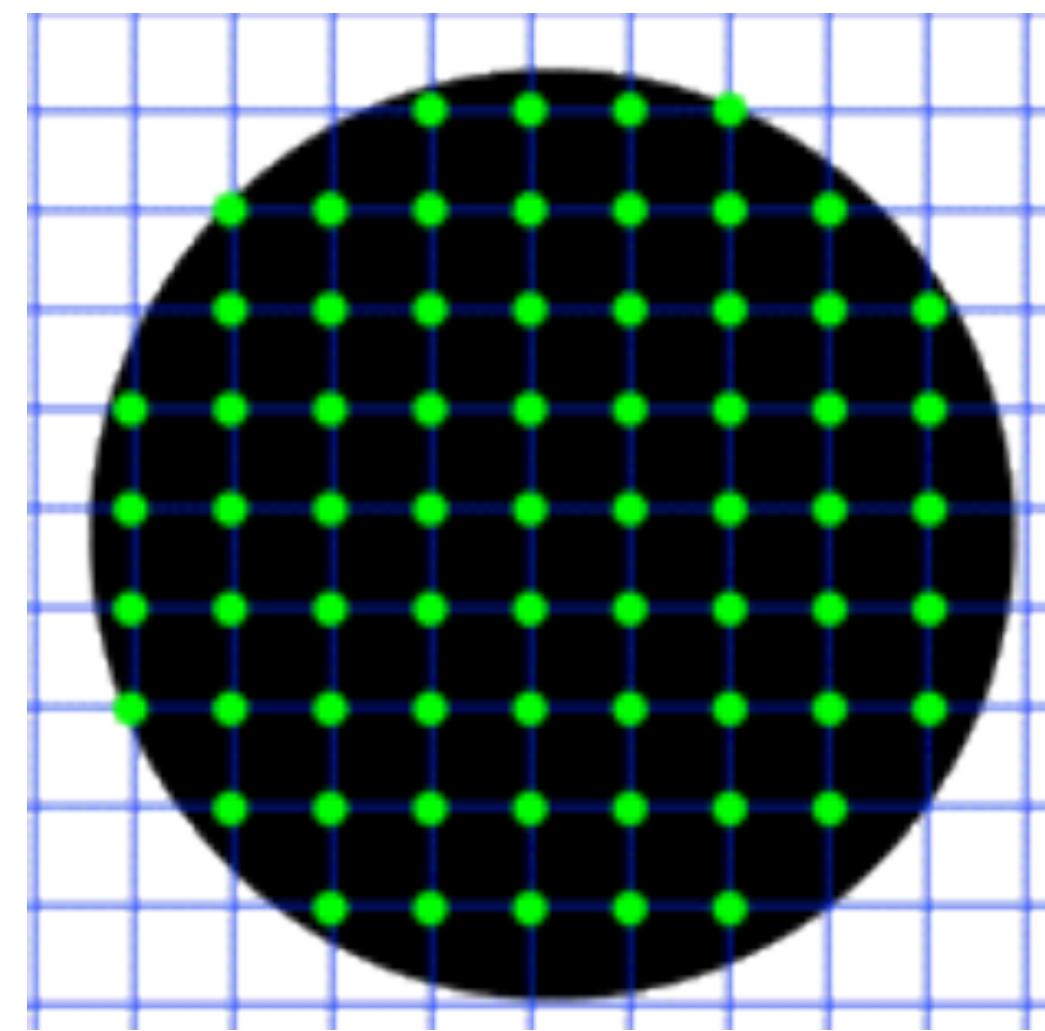
3. Compute intersection points

- Linear interpolation along edges

4. Connect them by edges

- Look-up table for edge configuration

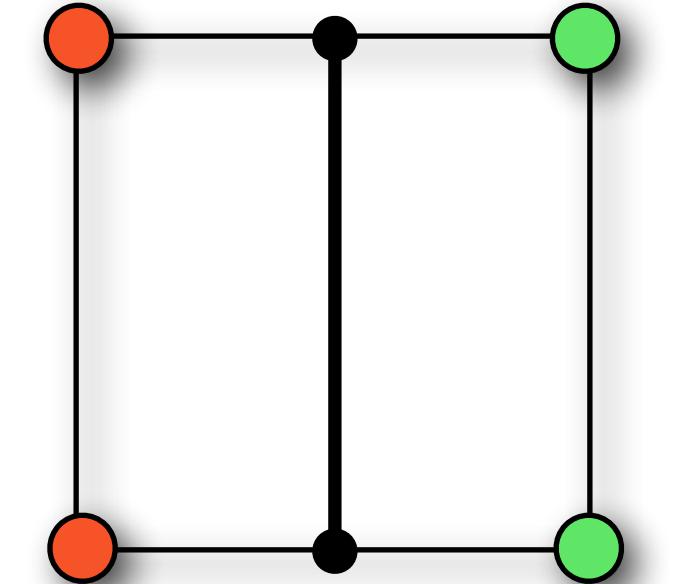
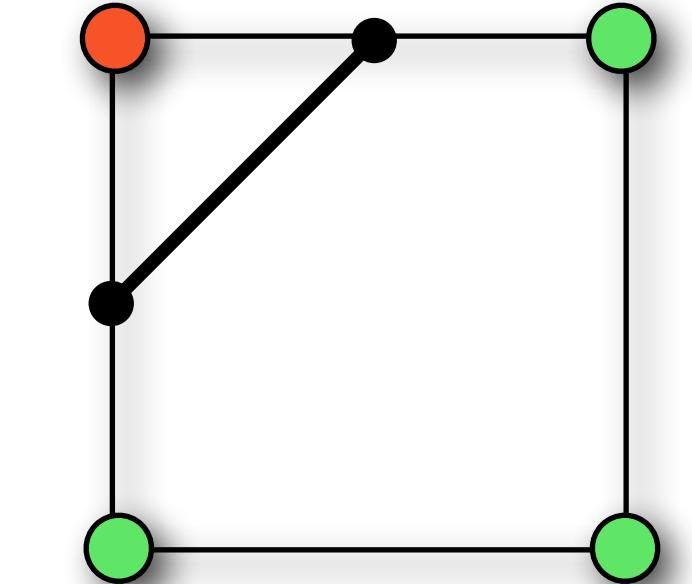
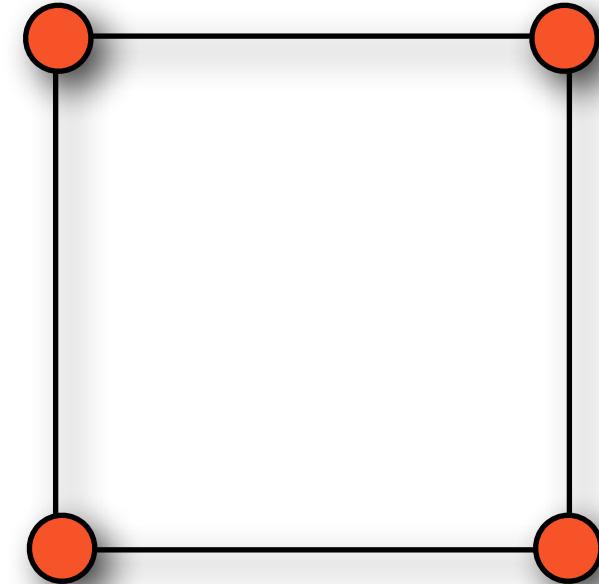
# Marching Cube in 2D



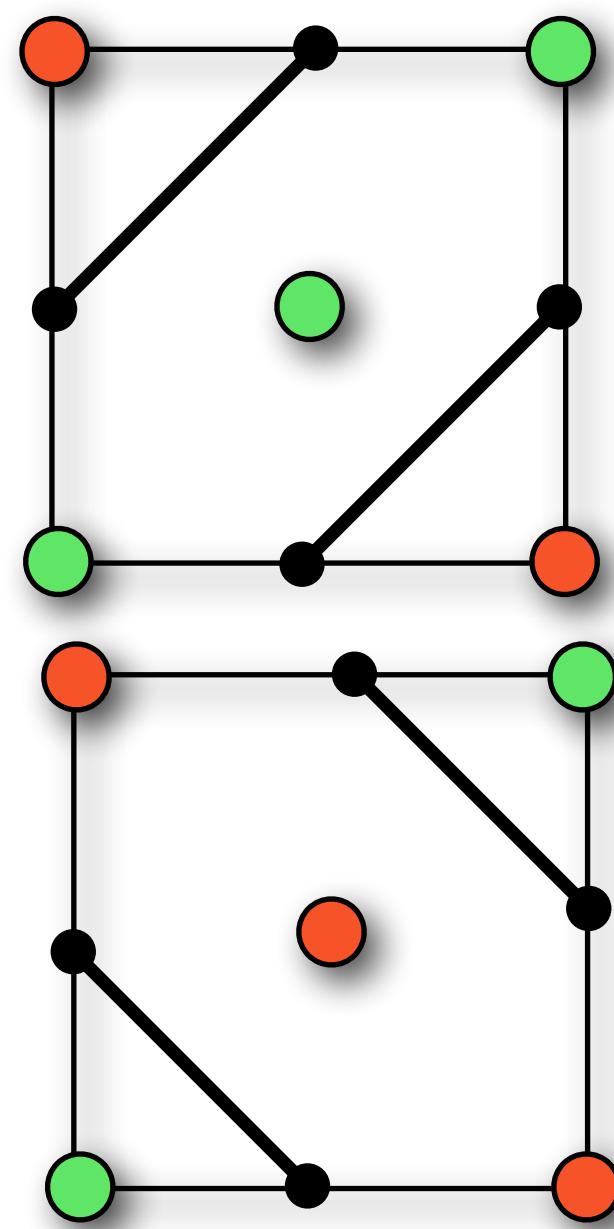
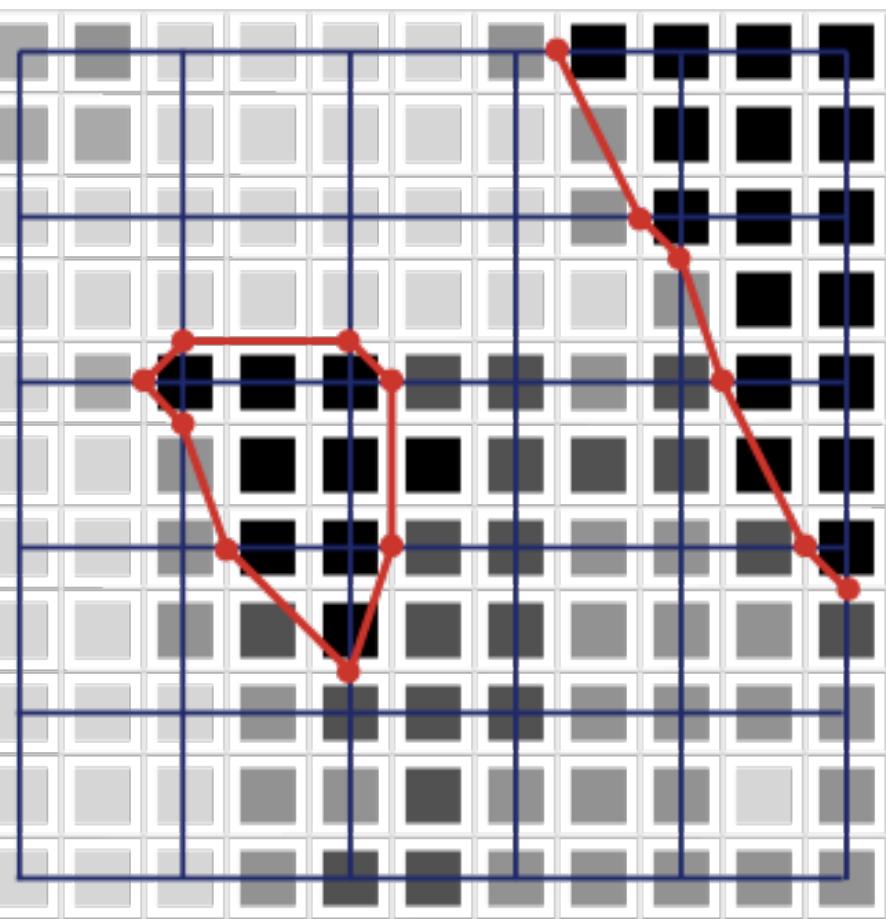
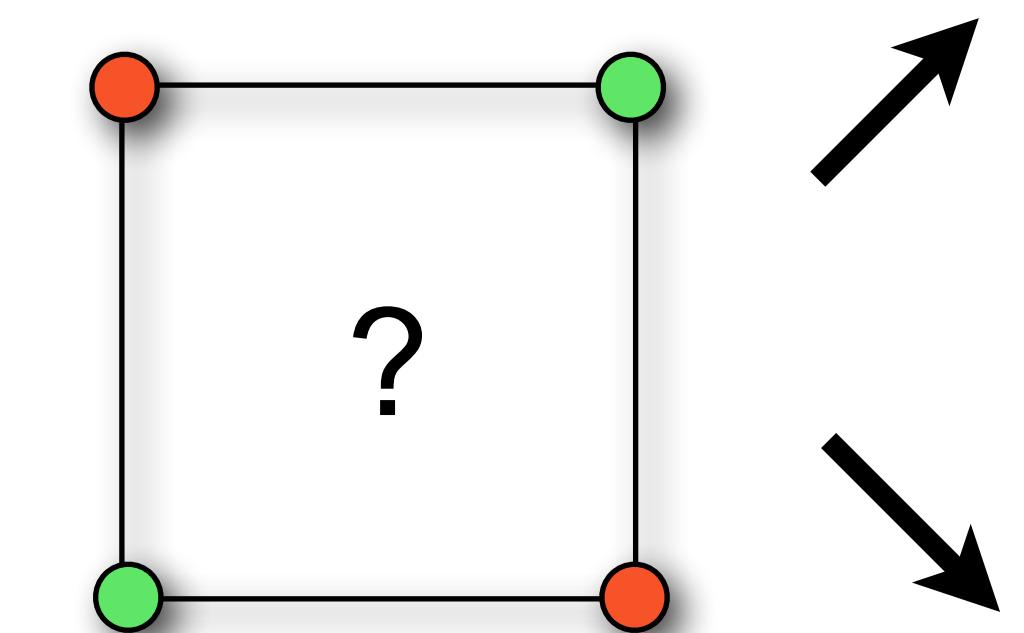
[<http://www.exaflop.org/docs/marchcubes/>]

# 2D: Marching Squares

- Classify grid nodes as inside/outside
- Classify cell:  $2^4 = 16$  configurations
- Linear interpolation along edges
- Look-up table for edge configuration



source: wikipedia



# 3D: Marching Cubes

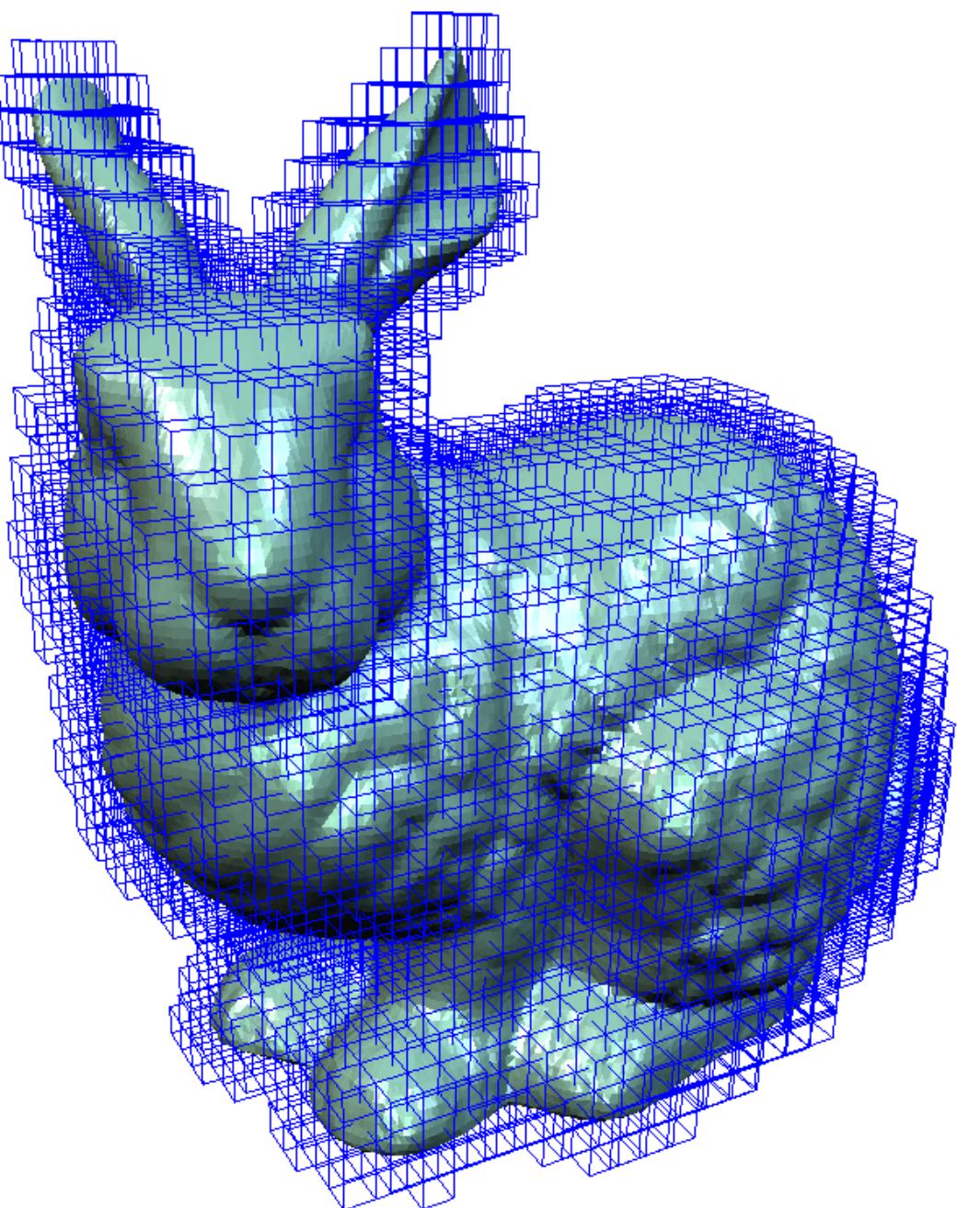


1. Classify grid nodes as inside/outside
  - Is  $F(\mathbf{x}_{ijk}) > 0$  or  $< 0$ ?
2. Classify cell:  $2^3$  configurations
  - In/out for each corner
3. Compute intersection points
  - Linear interpolation along edges
4. Connect them by triangles
  - Look-up table for path configuration
  - Disambiguation by modified table [Montani 94]

# Marching Cubes



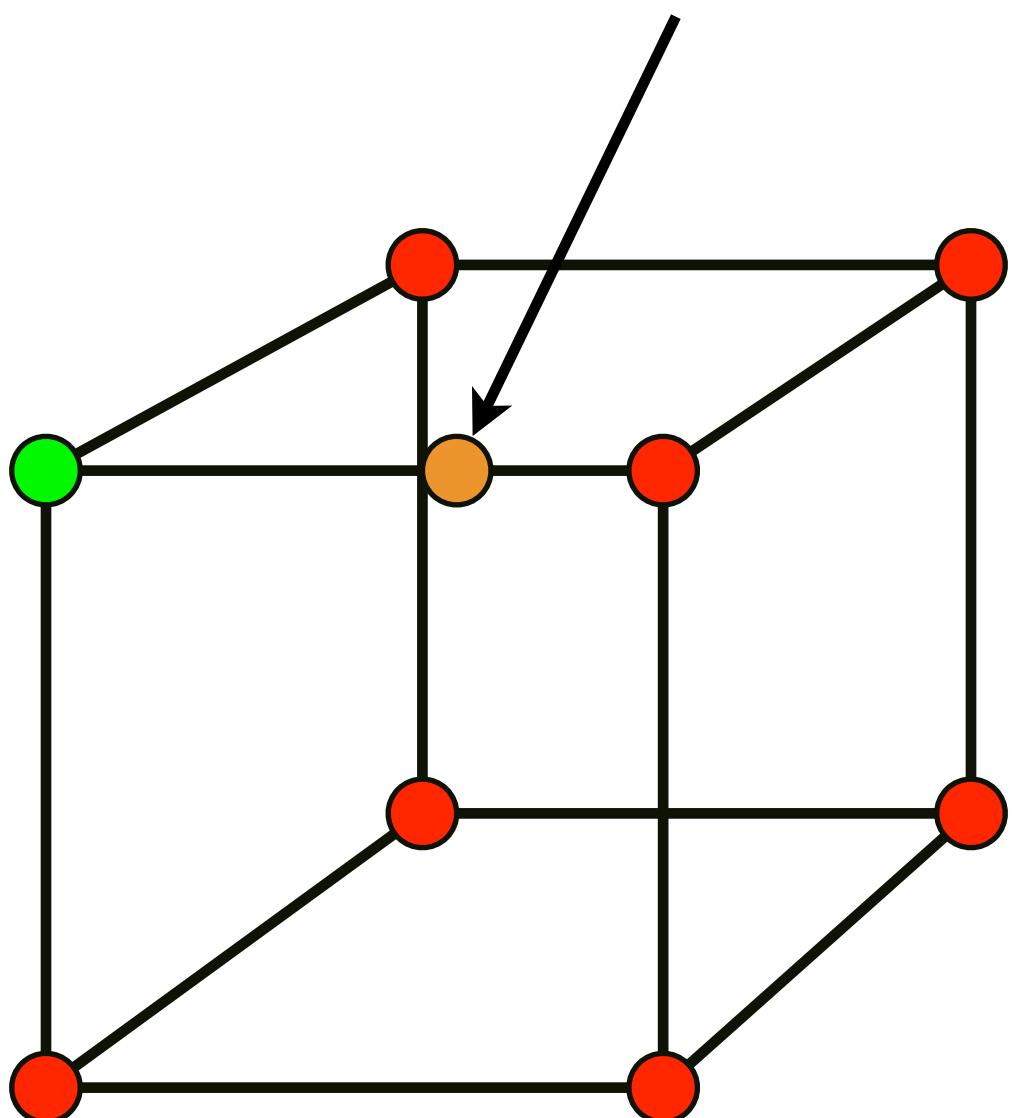
- Classify grid nodes  $\mathbf{x}_{i,j,k}$  based on  $F_{i,j,k} = F(\mathbf{x}_{i,j,k})$ 
  - Inside or outside
- Classify all cubes based on  $F_{i,j,k}$ 
  - Inside, outside, or intersecting
- Refined only intersected cells
  - 3-color adaptive octree
  - $O(h^{-2})$  complexity



# Intersection Points

- Linear interpolation along edges

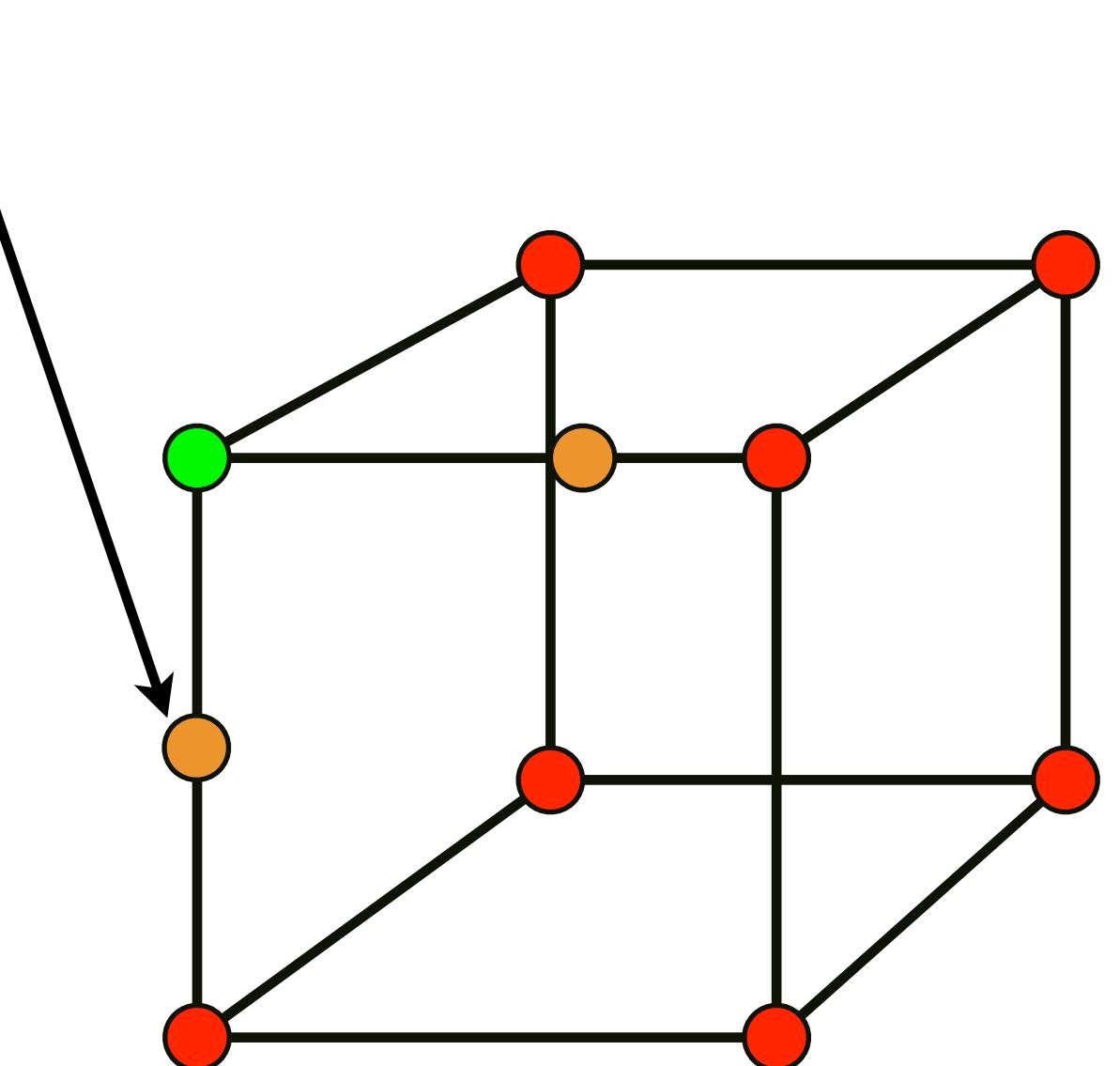
$$\frac{\mathbf{x}_{i,j,k} \cdot |F_{i+1,j,k}| + \mathbf{x}_{i+1,j,k} \cdot |F_{i,j,k}|}{|F_{i,j,k}| + |F_{i+1,j,k}|}$$



# Intersection Points

- Linear interpolation along edges

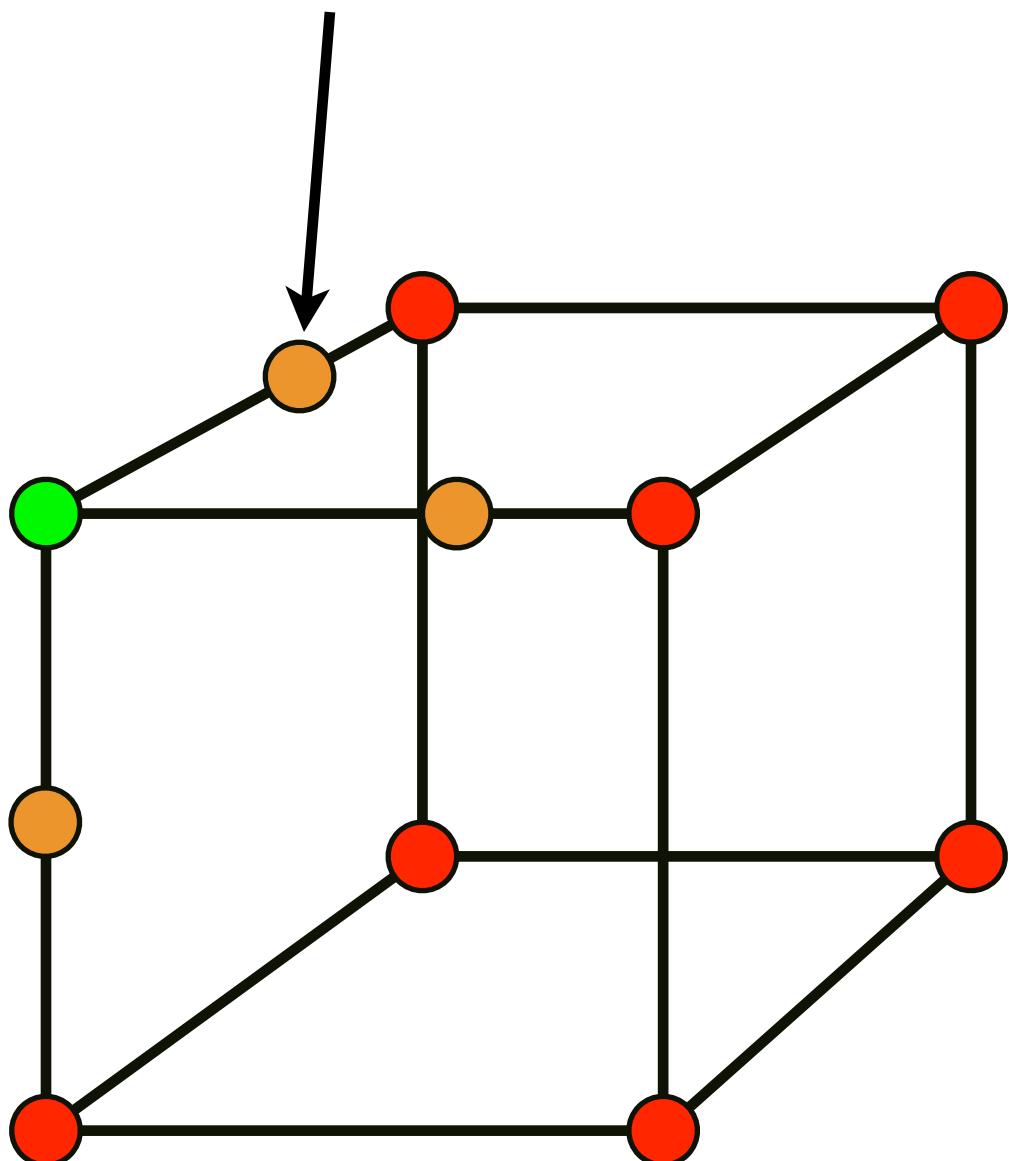
$$\frac{\mathbf{x}_{i,j,k} \cdot |F_{i,j+1,k}| + \mathbf{x}_{i,j+1,k} \cdot |F_{i,j,k}|}{|F_{i,j,k}| + |F_{i,j+1,k}|}$$



# Intersection Points

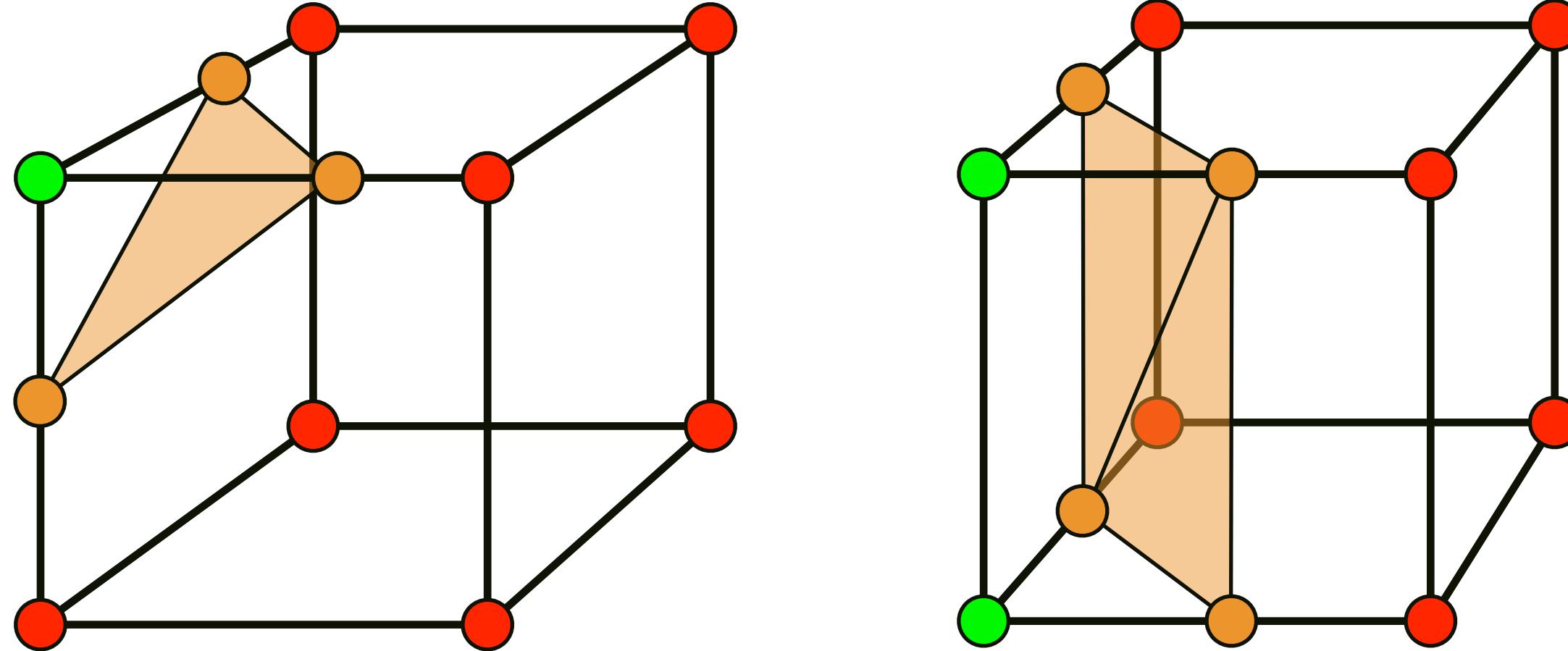
- Linear interpolation along edges

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



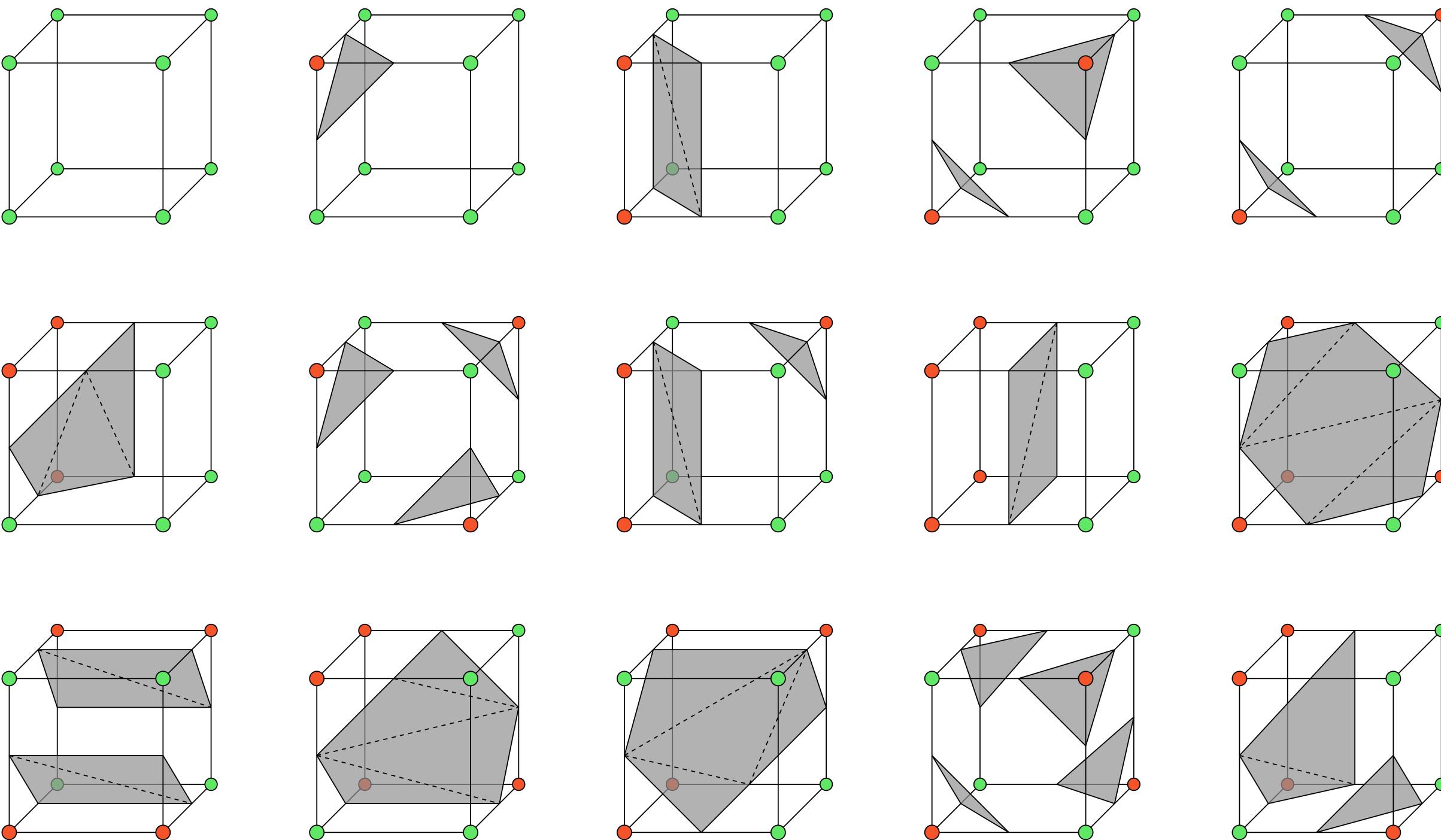
# Intersection Points

- Linear interpolation along edges
- Lookup table for patch configuration



# Marching Cubes

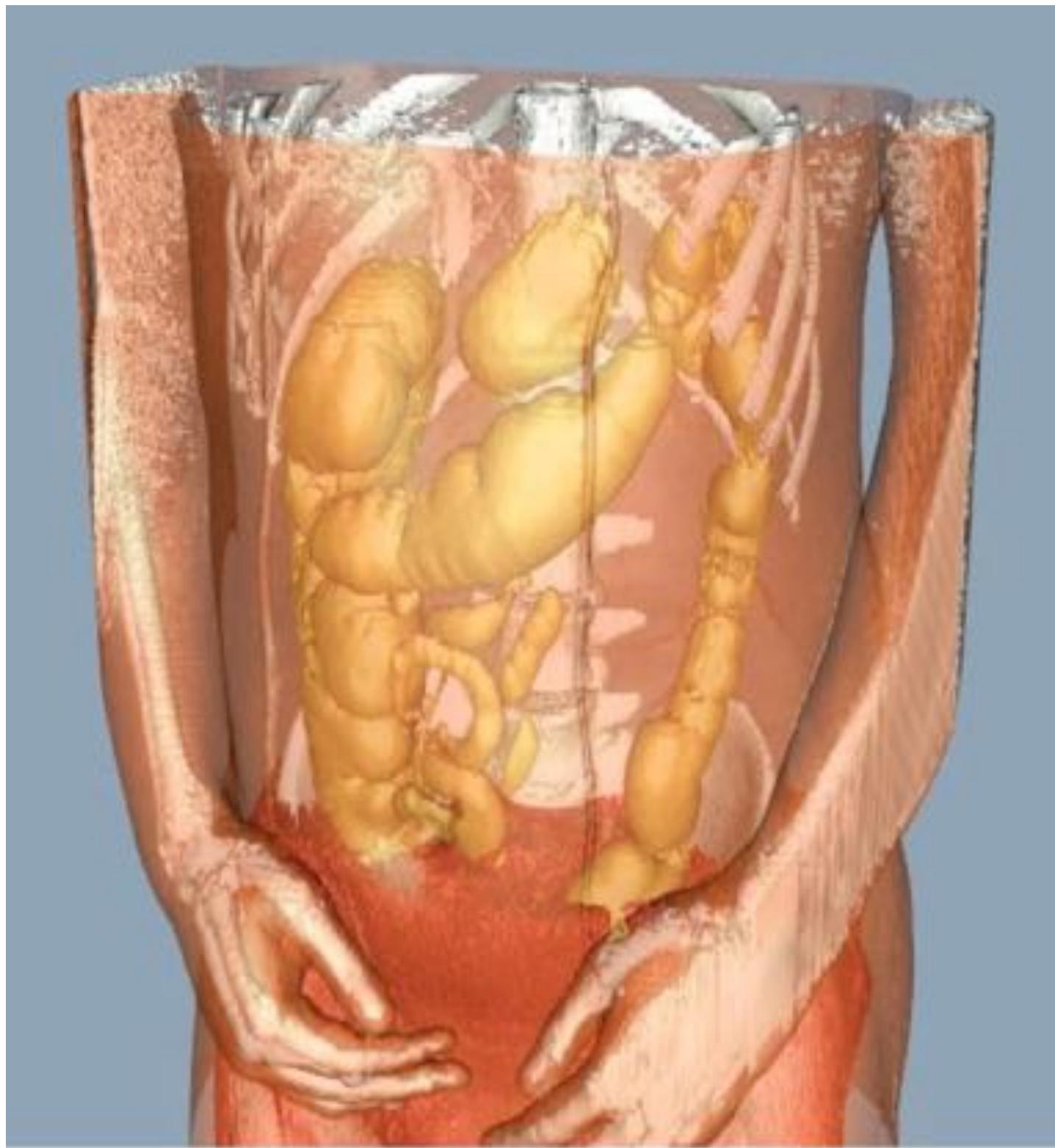
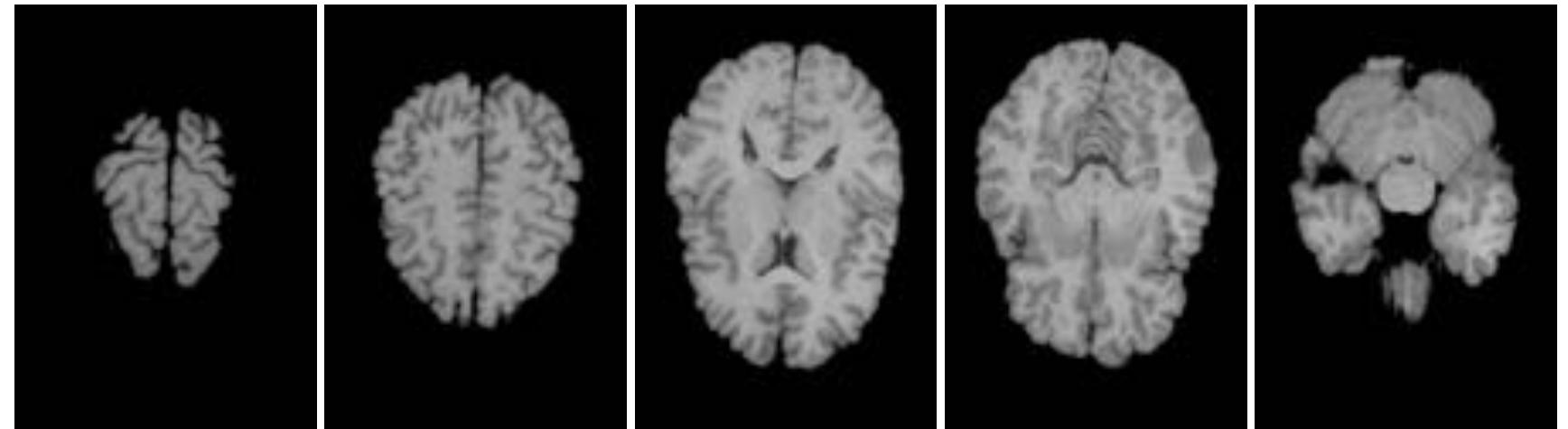
- Look-up table with  $2^8$  entries
  - 15 representative cases shown
  - Others follow by symmetry



# Marching Cubes

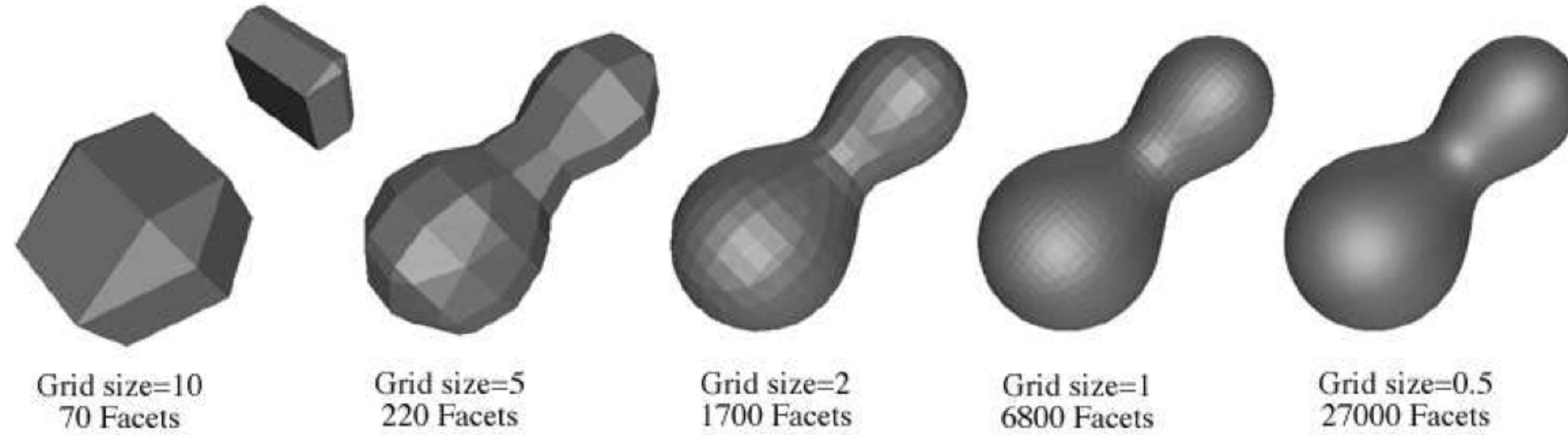


*THE* algorithm for isosurface extraction from medical scans (CT, MRI)



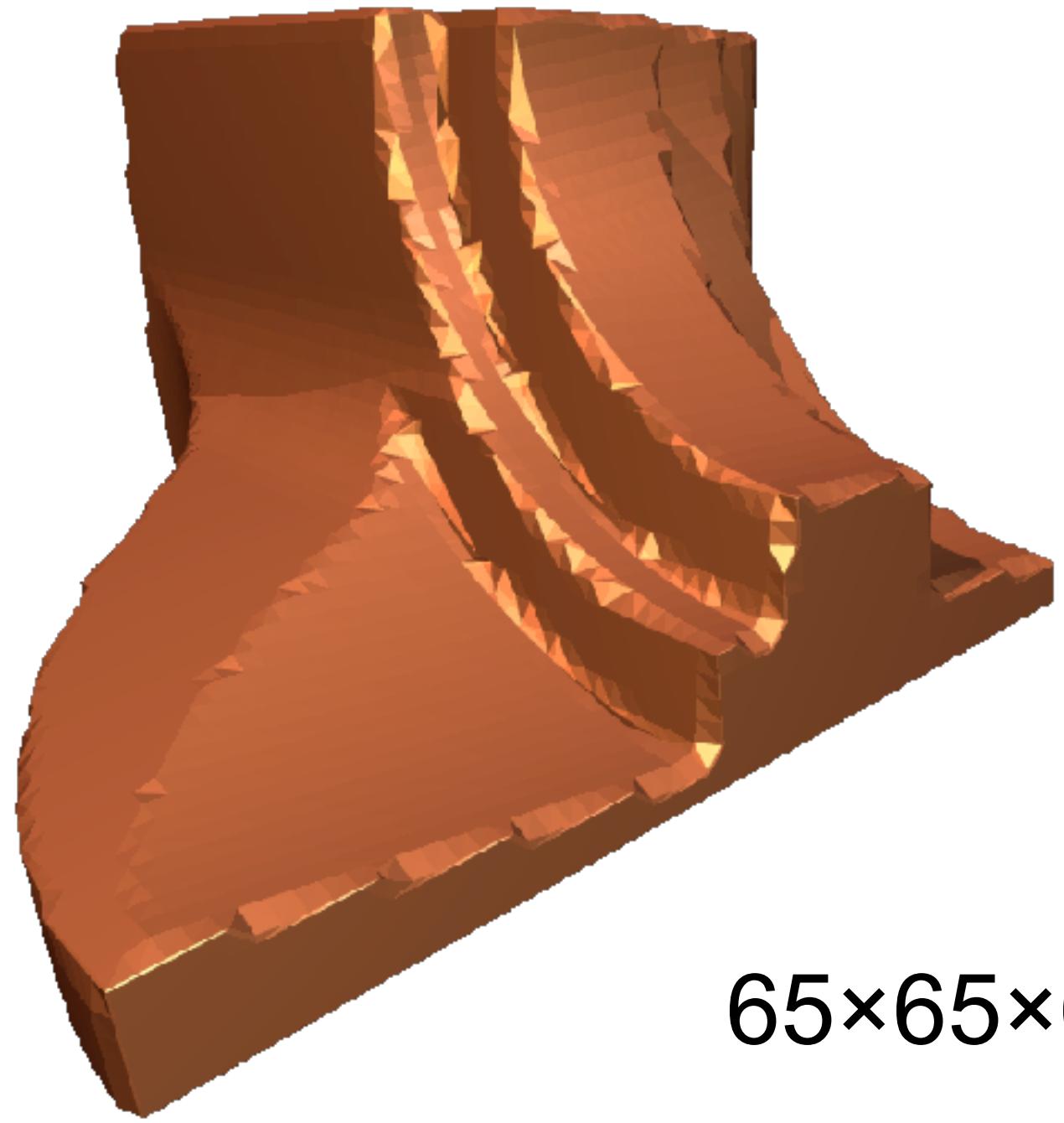
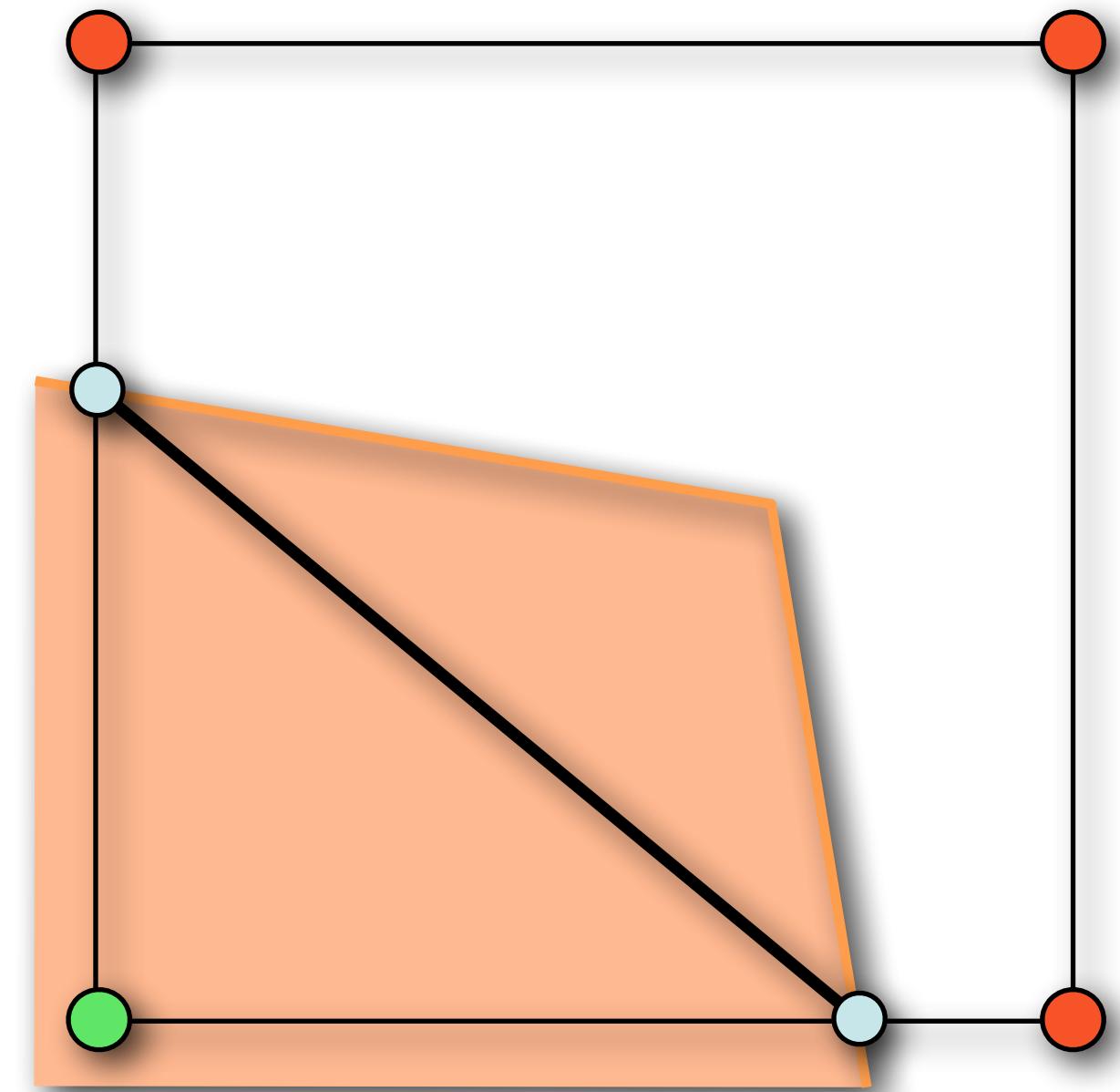
# Marching Cubes

## Effect of grid size

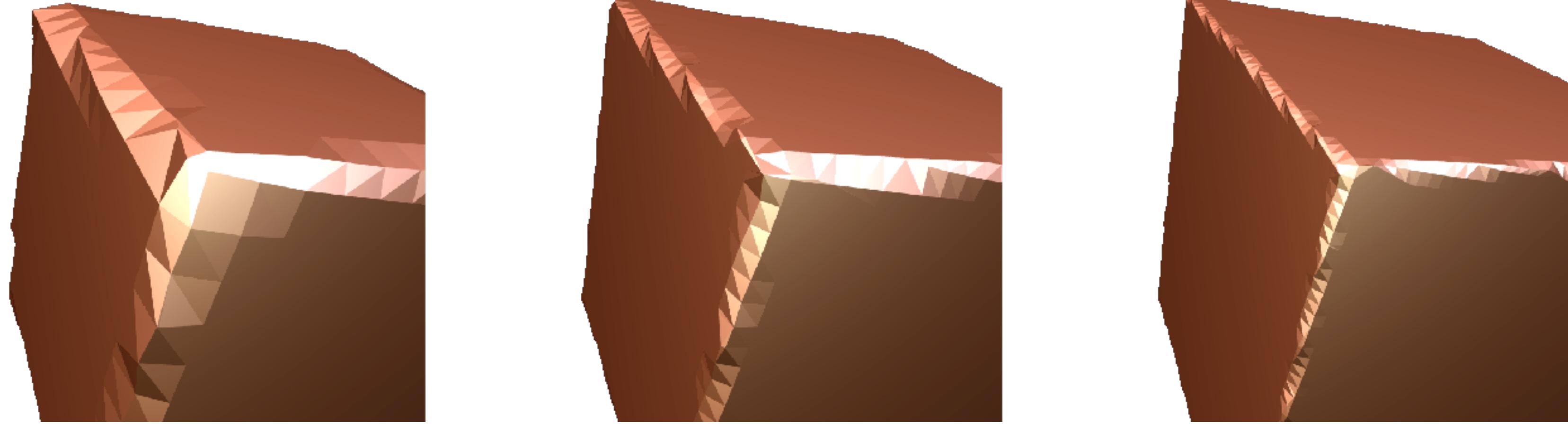


# Marching Cubes

- Sample points restricted to edges of regular grid
- Alias artifacts at sharp features



# Increasing Resolution?

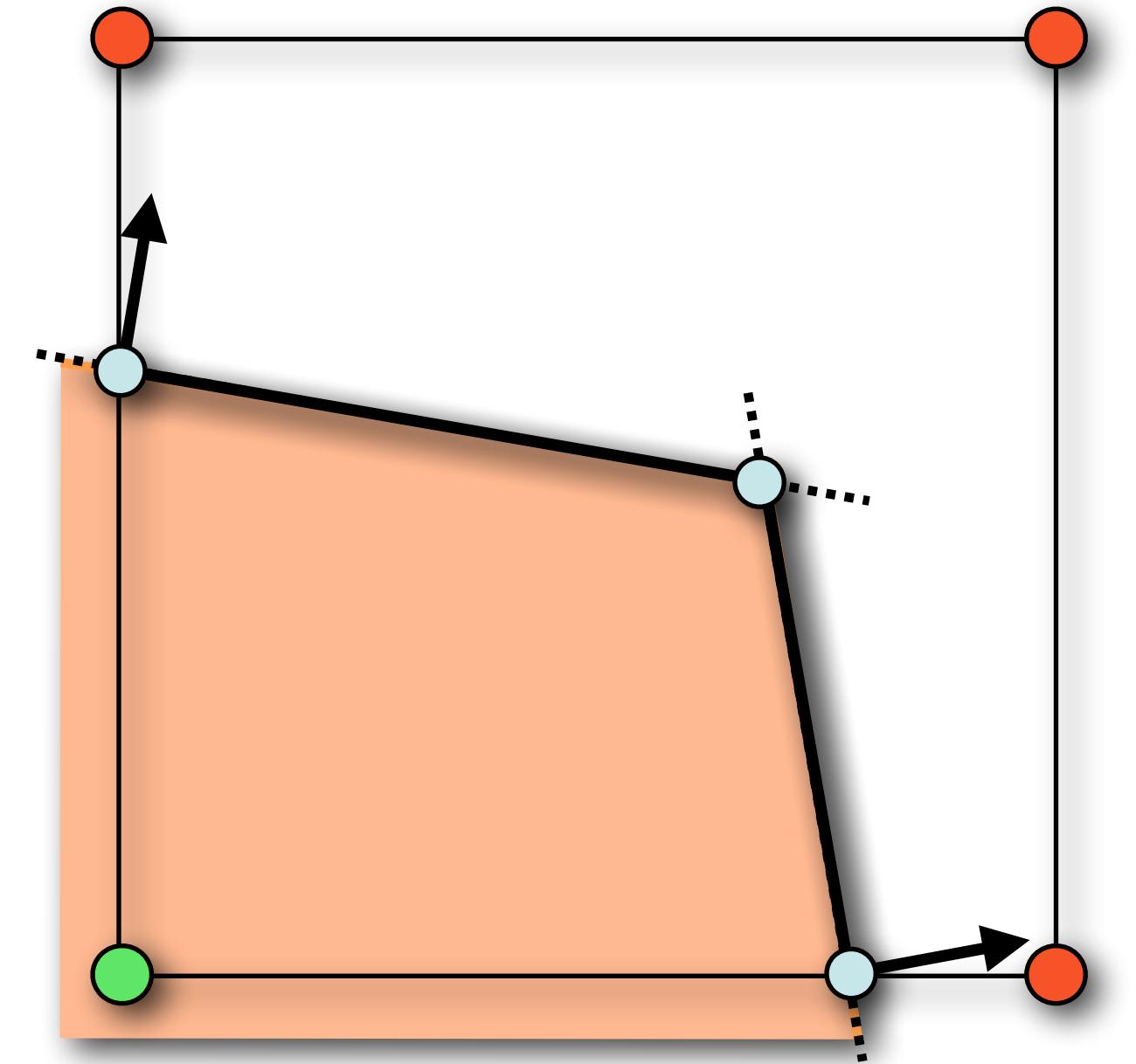


Does not remove alias problems!

# Extended Marching Cubes

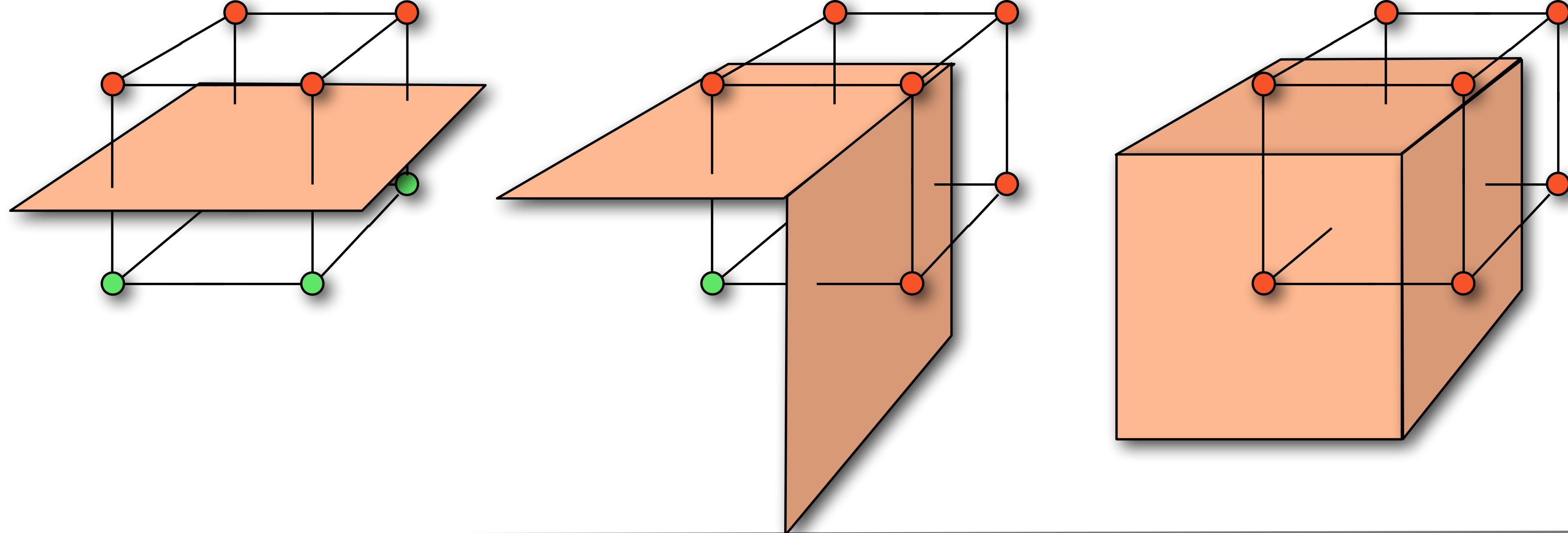


- Locally extrapolate distance gradient
- Place samples on estimated feature



# Extended Marching Cubes

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



# Extended Marching Cubes



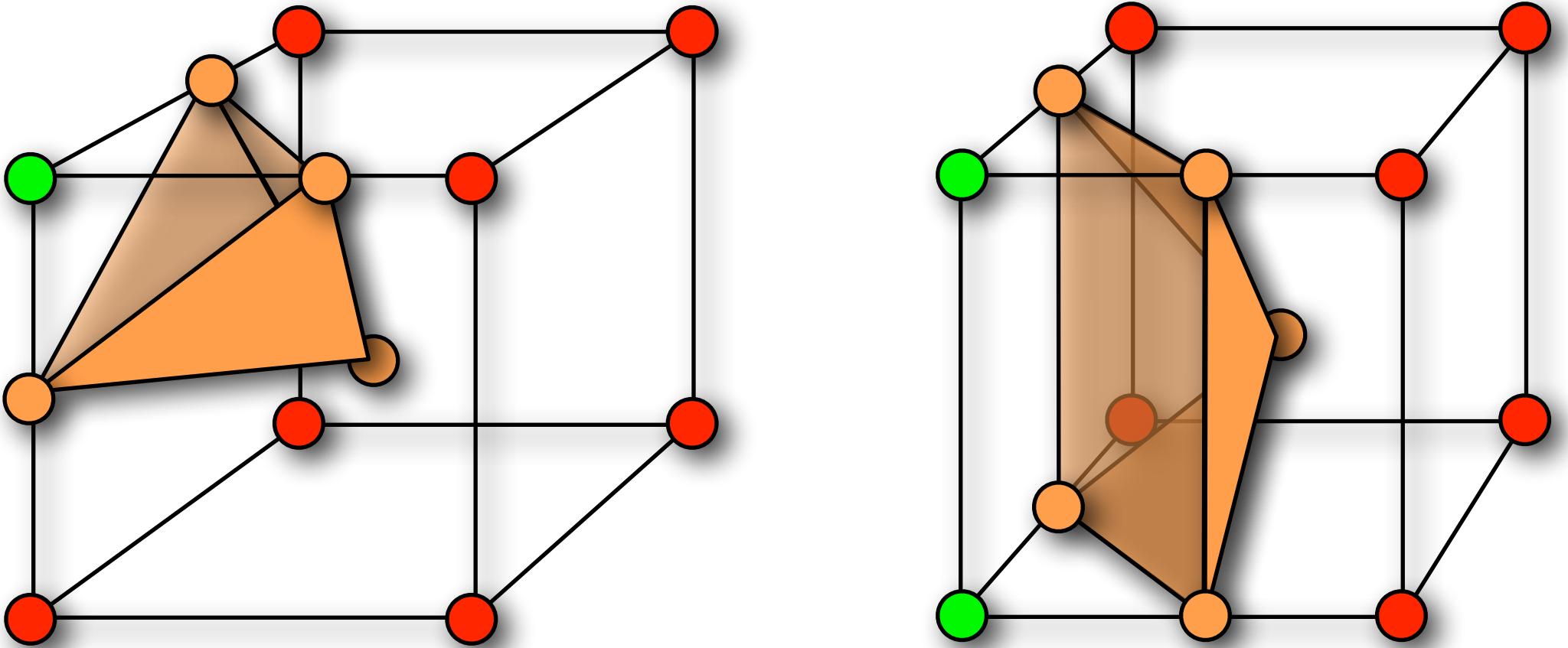
- Feature sampling
  - Intersect tangent planes ( $\mathbf{s}_i$ ,  $\mathbf{n}_i$ )

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

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

# Extended Marching Cubes

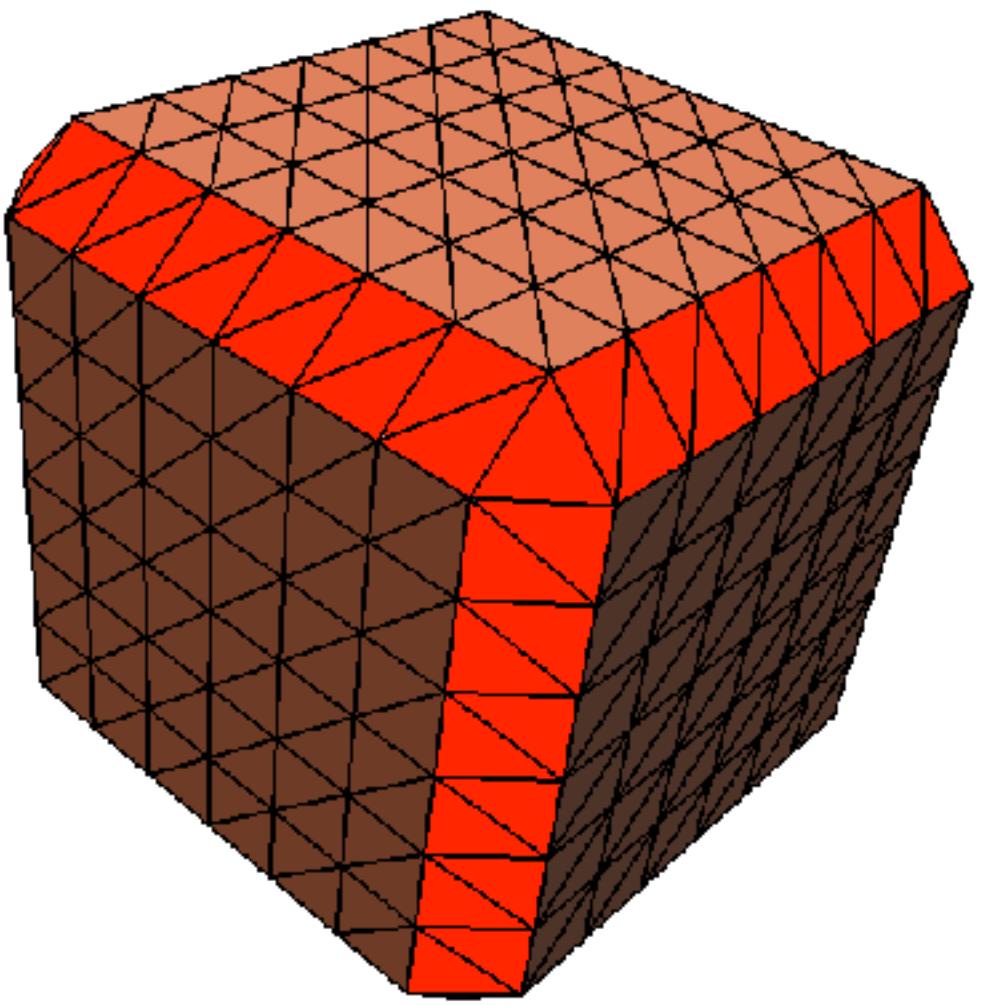
- Feature sampling
  - Intersect tangent planes ( $\mathbf{s}_i, \mathbf{n}_i$ )
  - Triangle fans centered at feature point



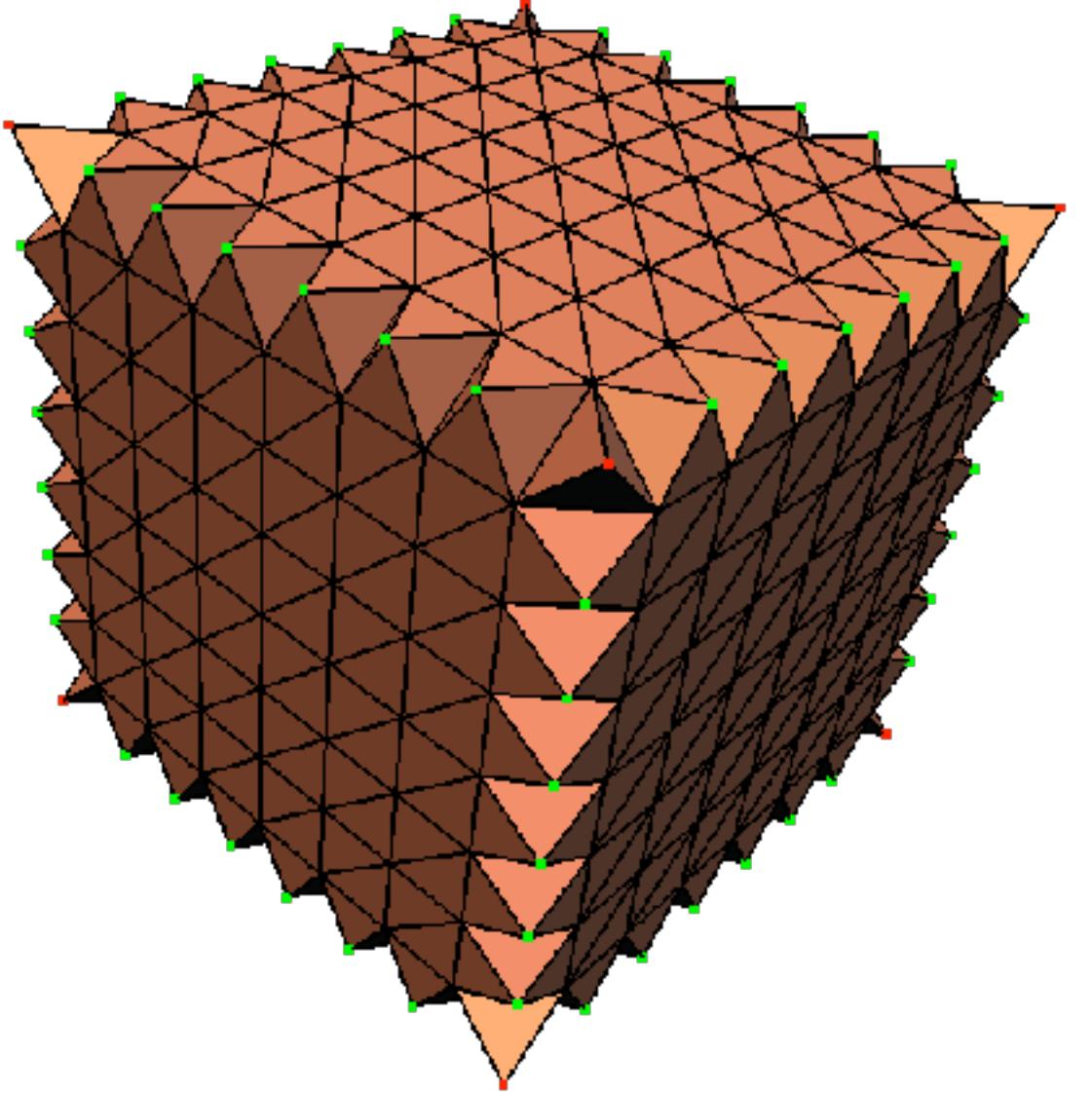
# Extended Marching Cubes



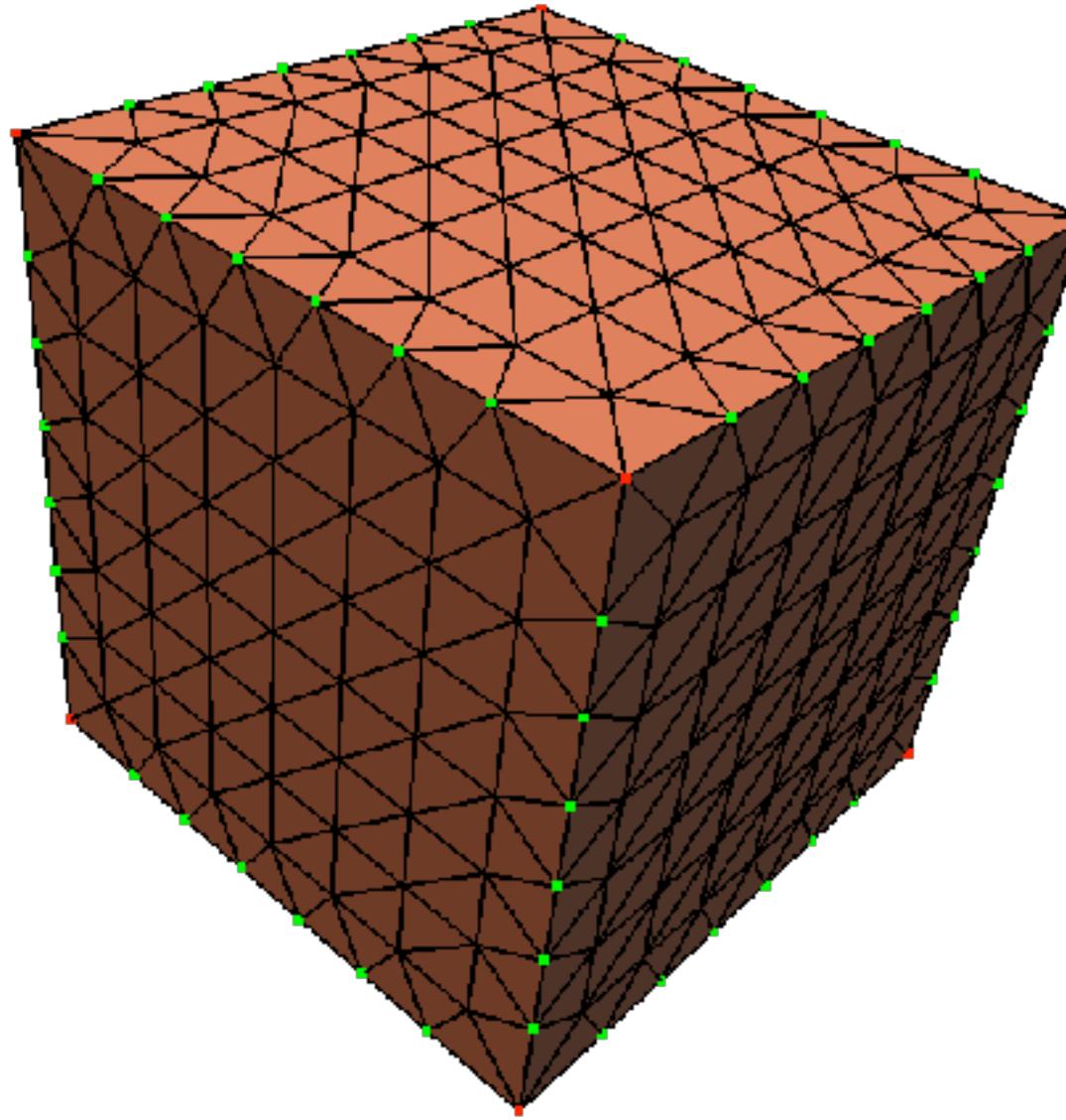
- Feature reconstruction



Feature  
Detection

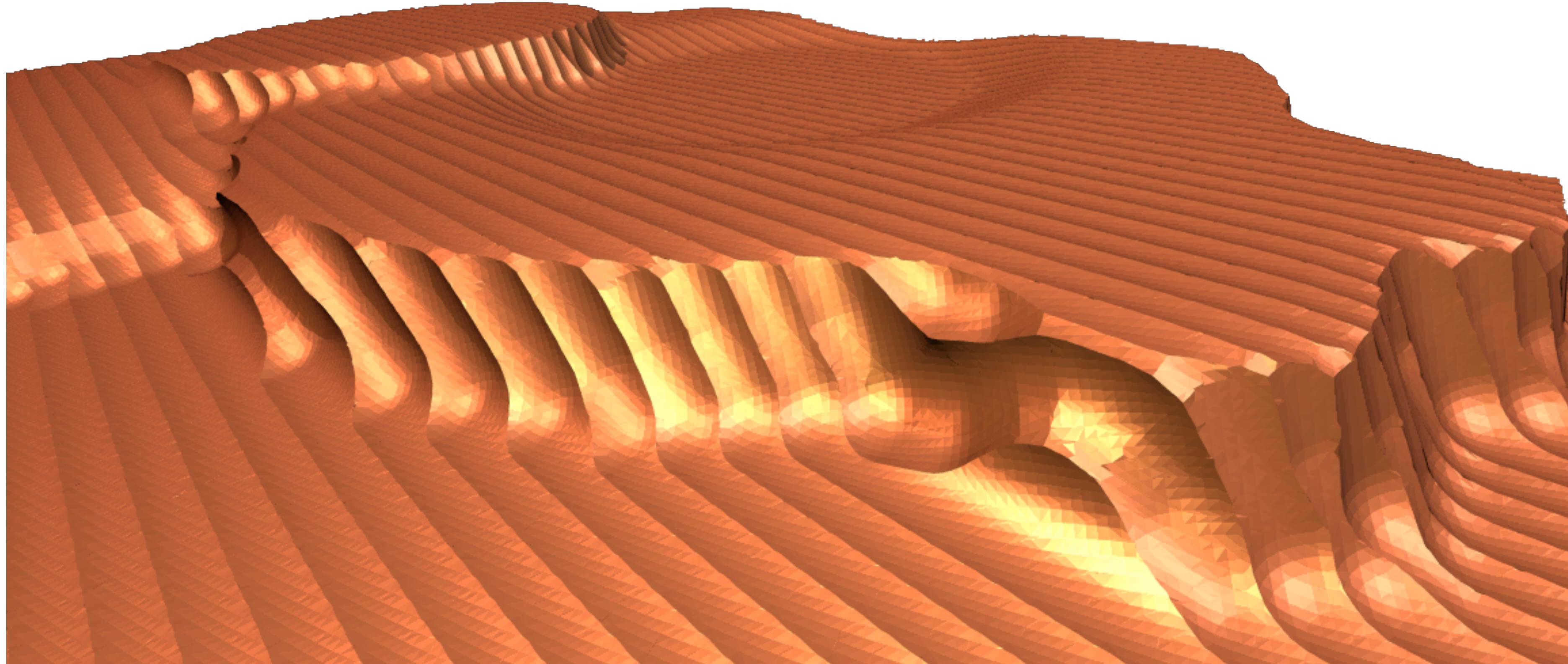


Feature  
Sampling



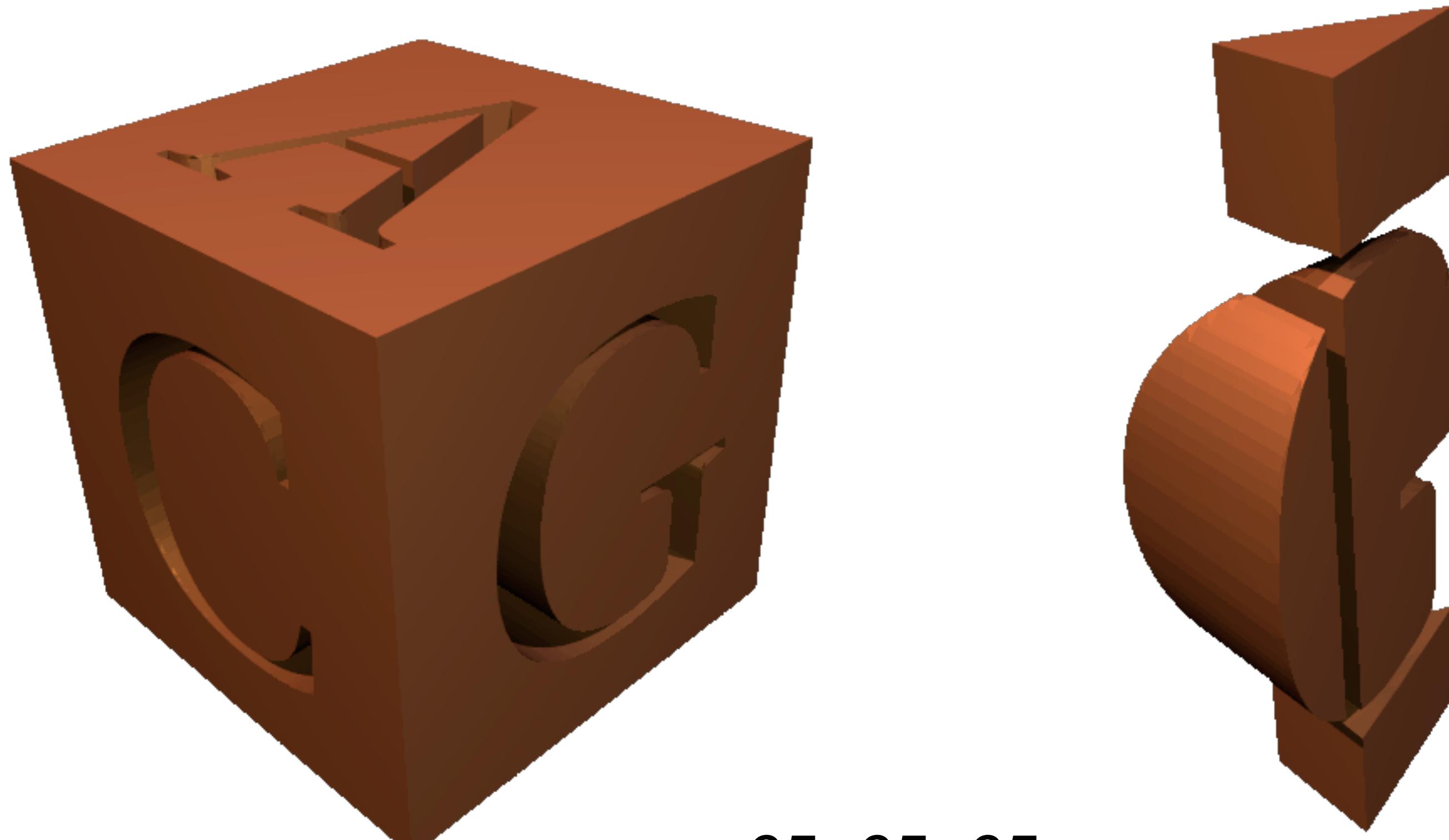
Edge  
Flipping

# 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 ( $\rightarrow$  mesh decimation)
- Degenerate triangles ( $\rightarrow$  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