UC San Diego

Shape Representations

Instructor: Hao Su

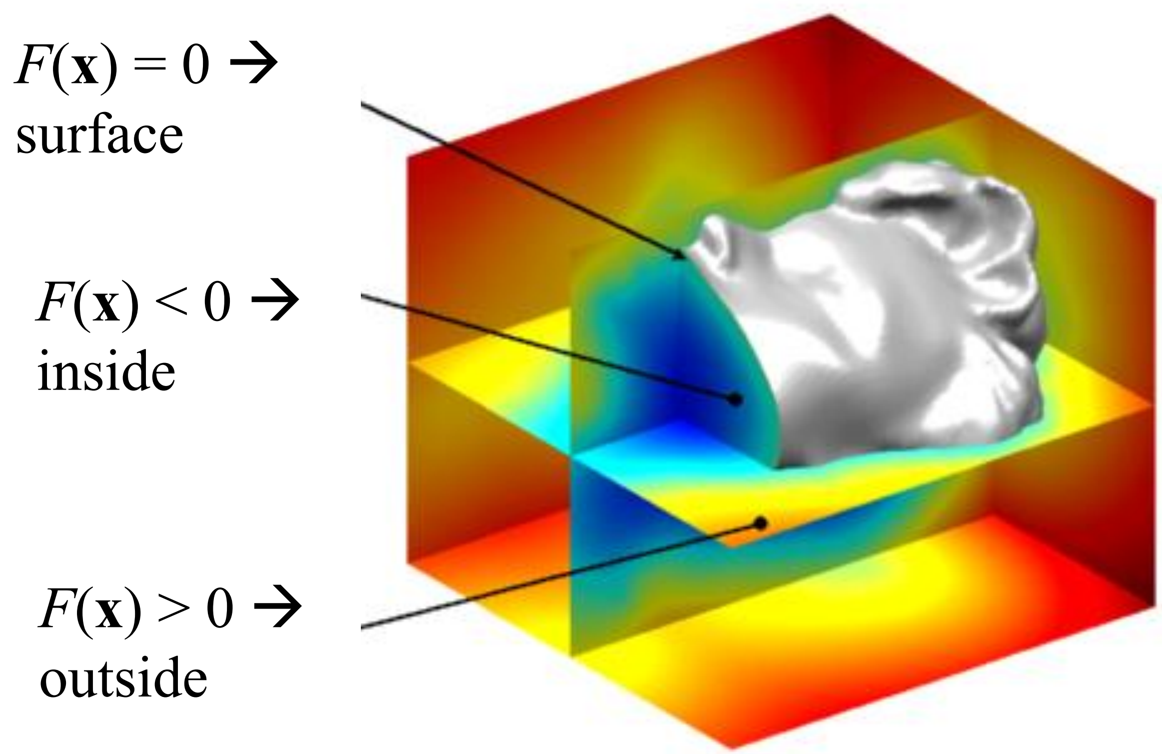
Slides credits: Olga Diamanti, Olga Sorkine-Hornung, Daniele Panozzo, ETH Zurich, Maks Ovsjanikov, Mario Botsch

IMPLICIT → MESH

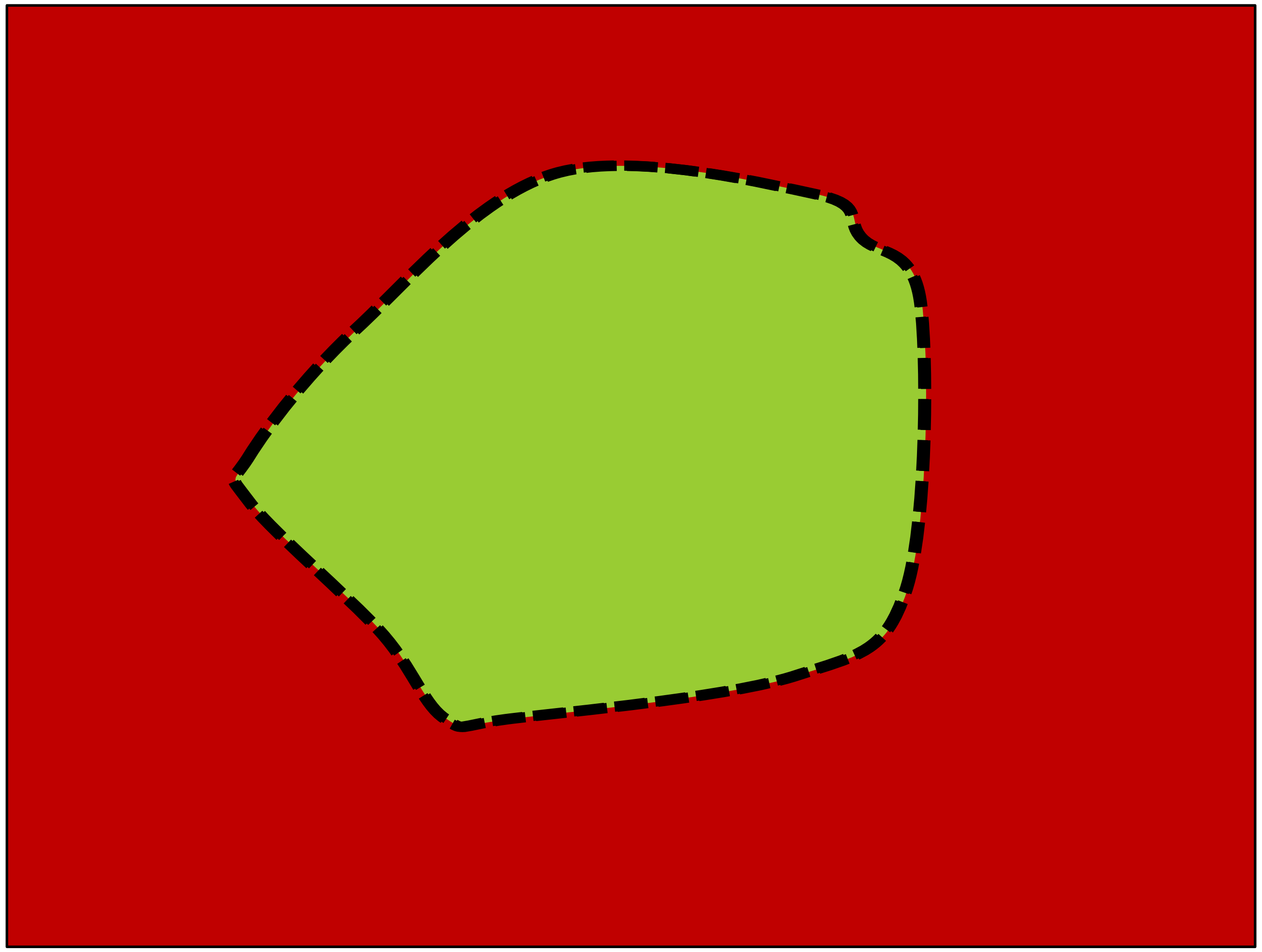
Marching Cubes

Extracting the Surface

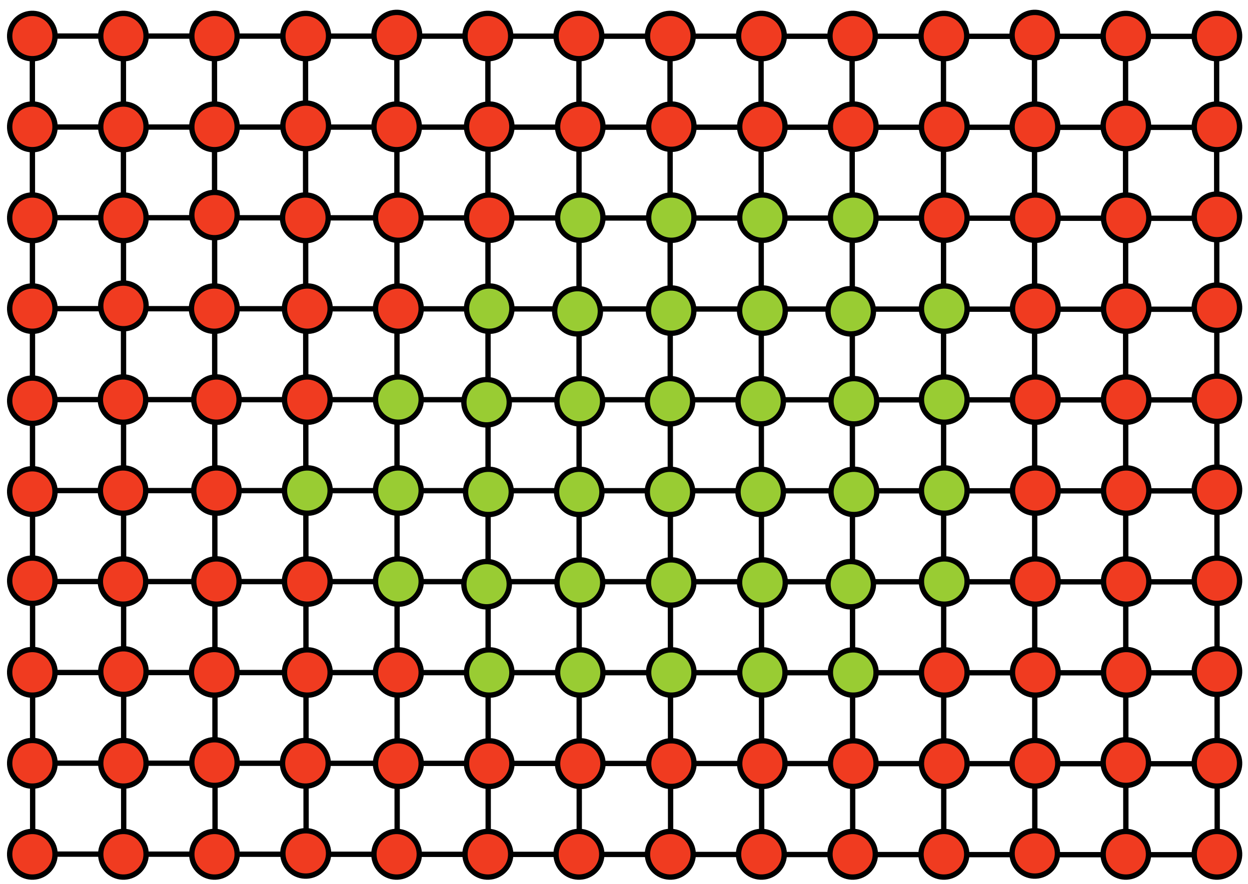
- Wish to compute a manifold mesh of the level set



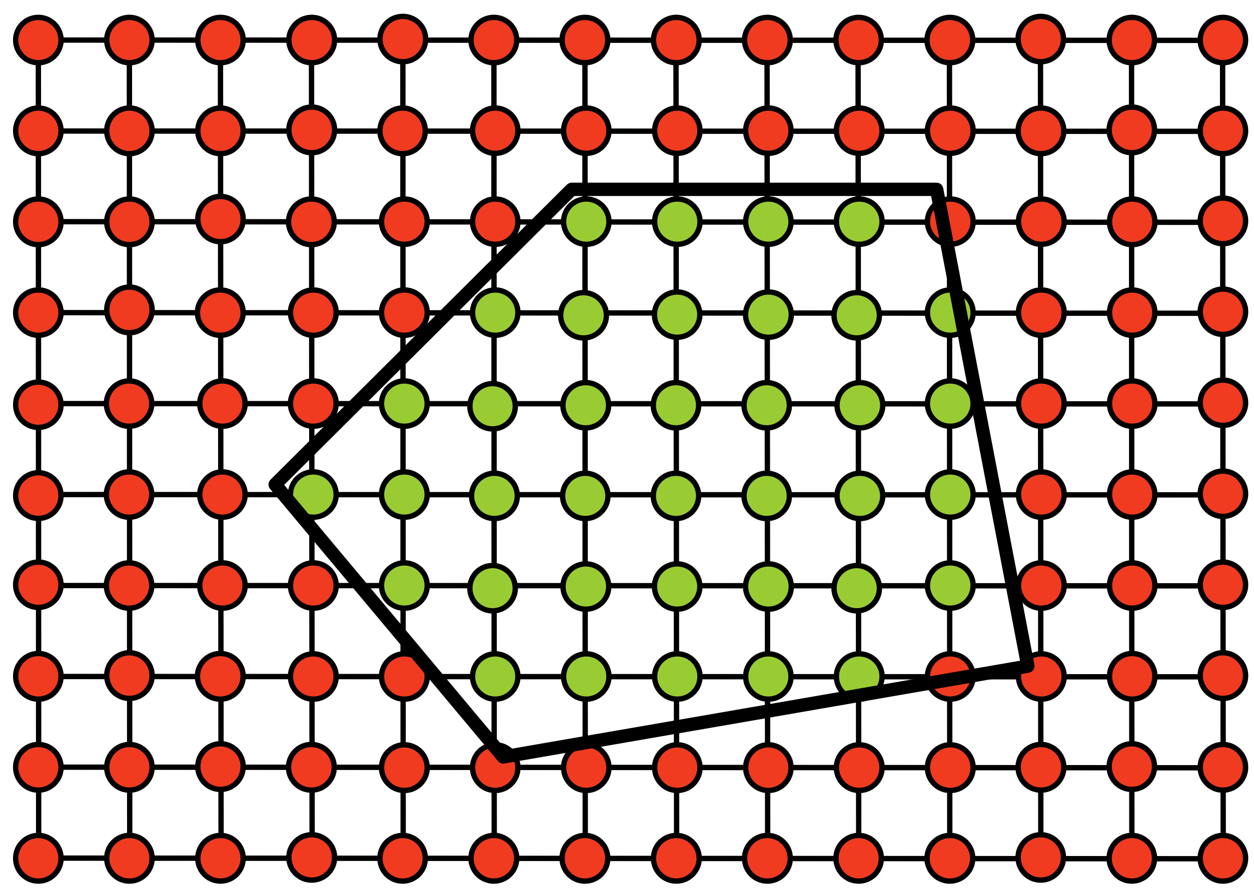
Sample the SDF



Sample the SDF



Sample the SDF

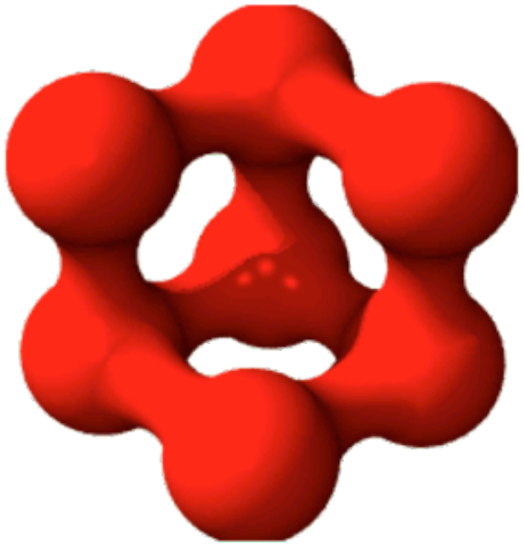


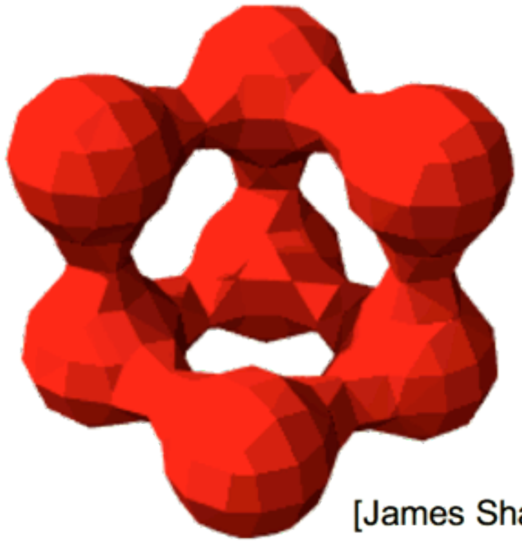
Marching Cubes

Converting from implicit to explicit representations.

Goal: Given an implicit representation: $\{\mathbf{x}, \text{s.t.}\, f(\mathbf{x}) = 0\}$

Create a triangle mesh that approximates the surface.





[James Sharman]

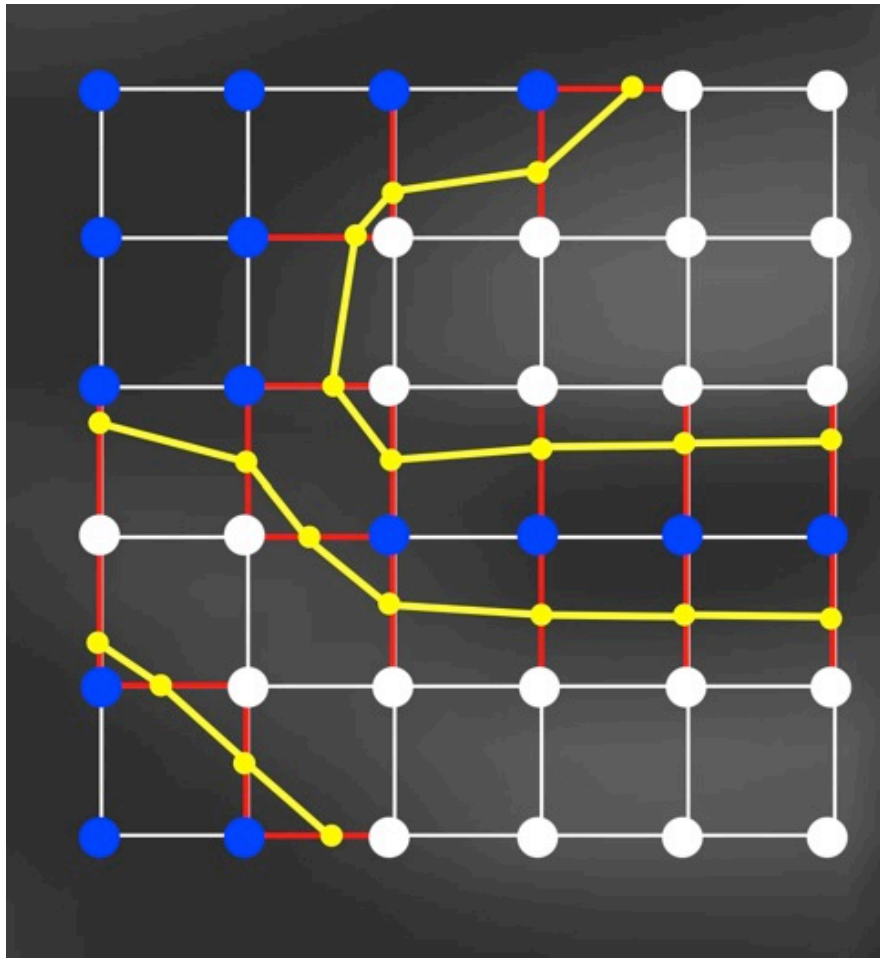
Lorensen and Cline, SIGGRAPH ‘87

Marching Squares (2D)

Given a function: $f(x)$

- $f(\mathbf{x}) < 0$ inside
- $f(\mathbf{x}) > 0$ outside

1. Discretize space.
2. Evaluate $f(x)$ on a grid.

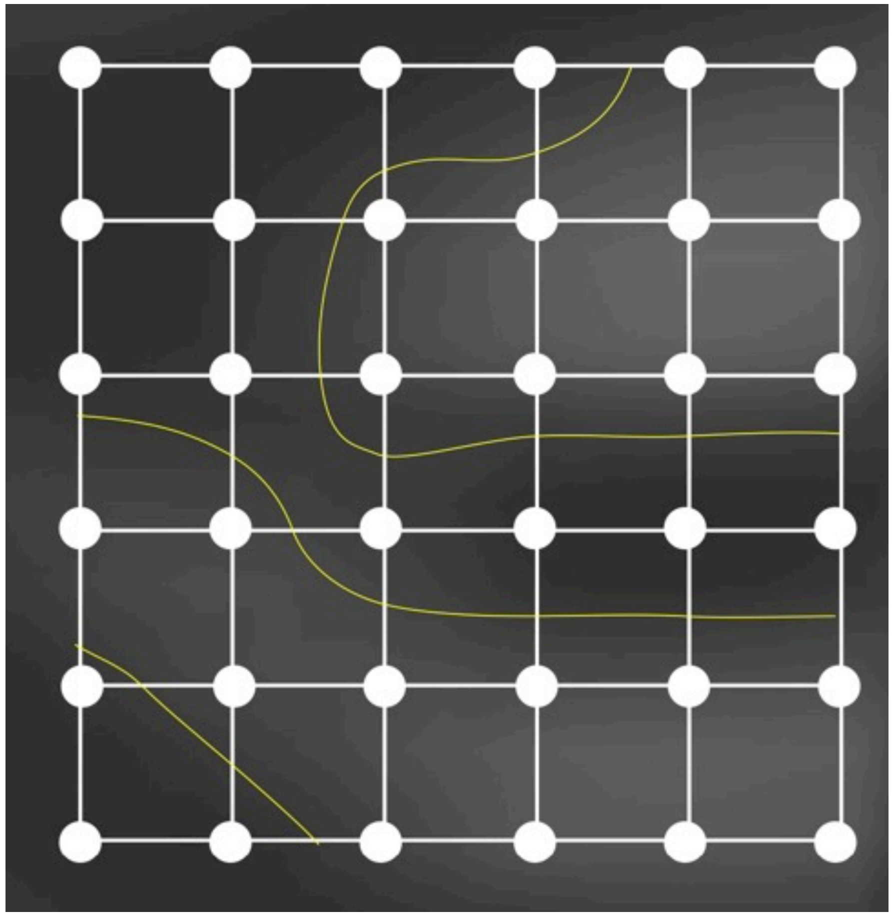


Marching Squares (2D)

Given a function: $f(x)$

- $f(\mathbf{x}) < 0$ inside
- $f(\mathbf{x}) > 0$ outside

1. Discretize space.
2. Evaluate $f(x)$ on a grid.
3. Classify grid points (+/-)
4. Classify grid edges
5. Compute intersections
6. Connect intersections



Marching Squares (2D)

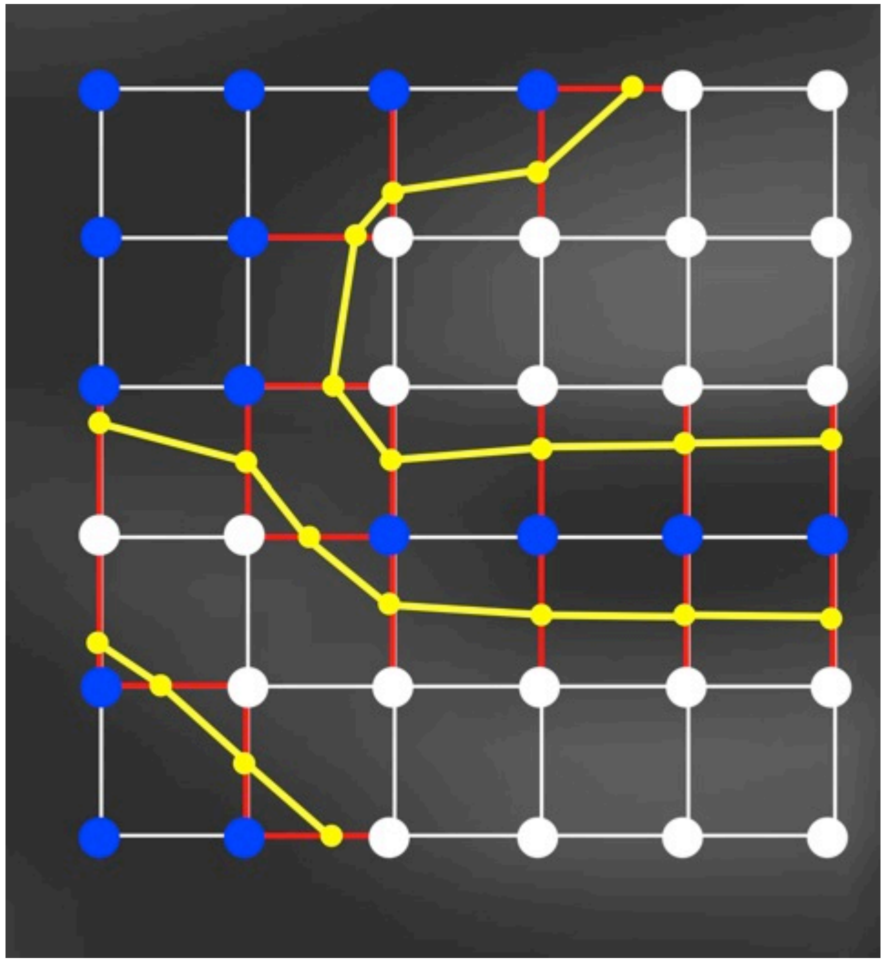
Computing the intersections:

- Edges with a sign switch contain intersections.

$$f(x_1) < 0, f(x_2) > 0 \Rightarrow$$
$$f(x_1 + t(x_2 - x_1)) = 0$$
$$\text{for some } 0 \leq t \leq 1$$

- Simplest way to compute t: assume f is linear between x1 and x2:

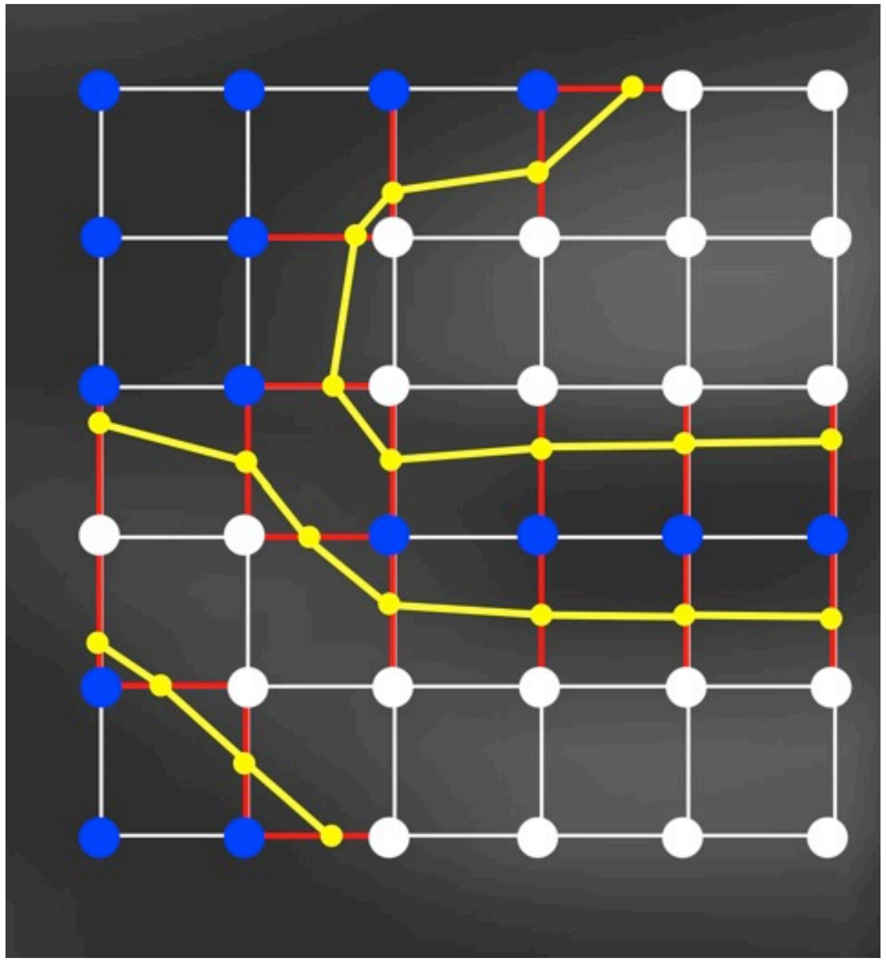
$$t = \frac{f(x_1)}{f(x_2) - f(x_1)}$$



Marching Squares (2D)

Connecting the intersections:

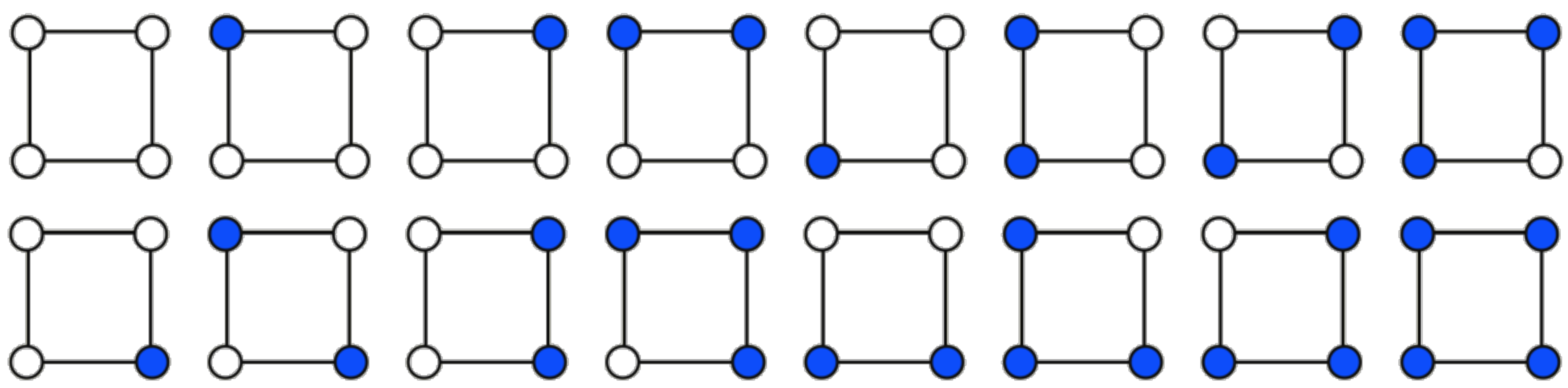
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.



Marching Squares (2D)

Connecting the intersections:

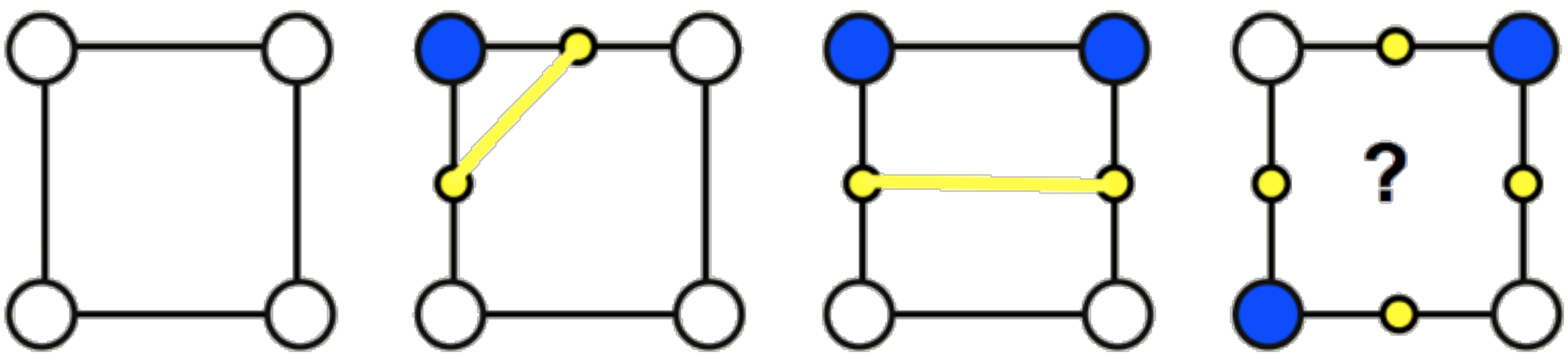
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.
- Group those leading to the same intersections



Marching Squares (2D)

Connecting the intersections:

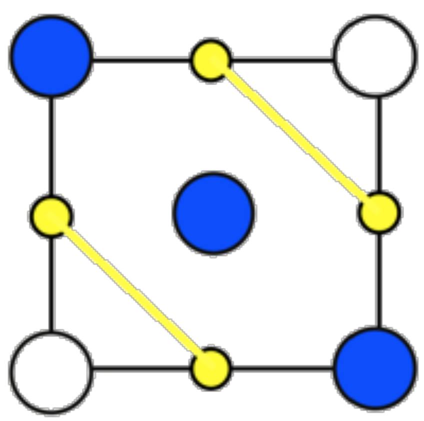
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.
- Group those leading to the same intersections.
- Group equivalent after rotation.
- Connect intersections

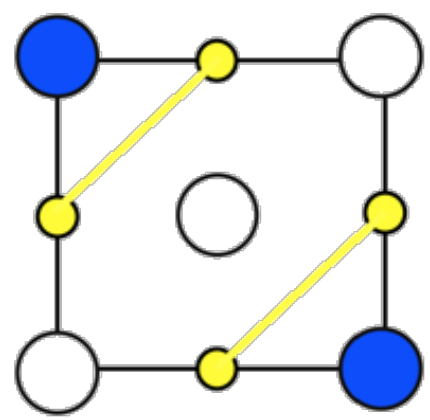


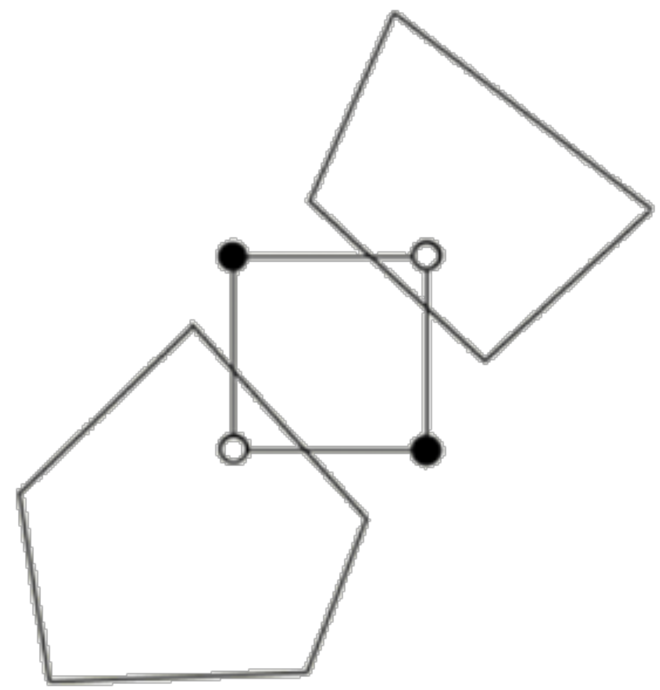
Marching Squares (2D)

Connecting the intersections:

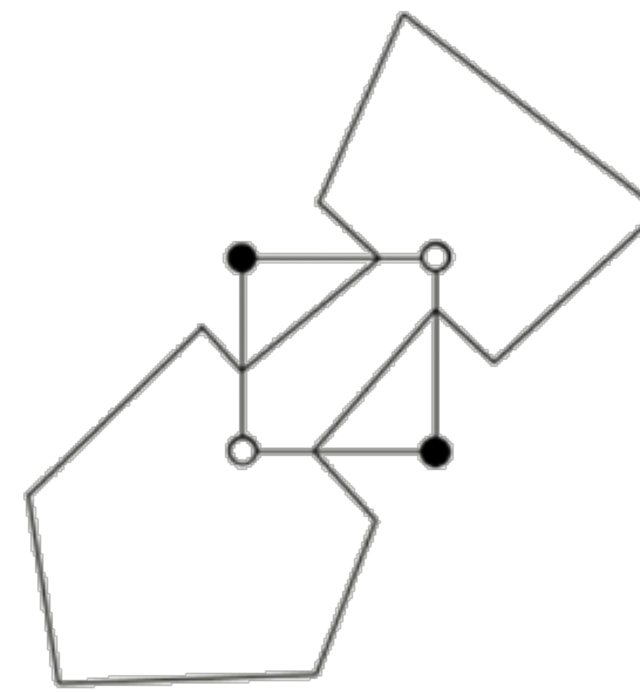
Ambiguous cases:







Break contour



Join contour

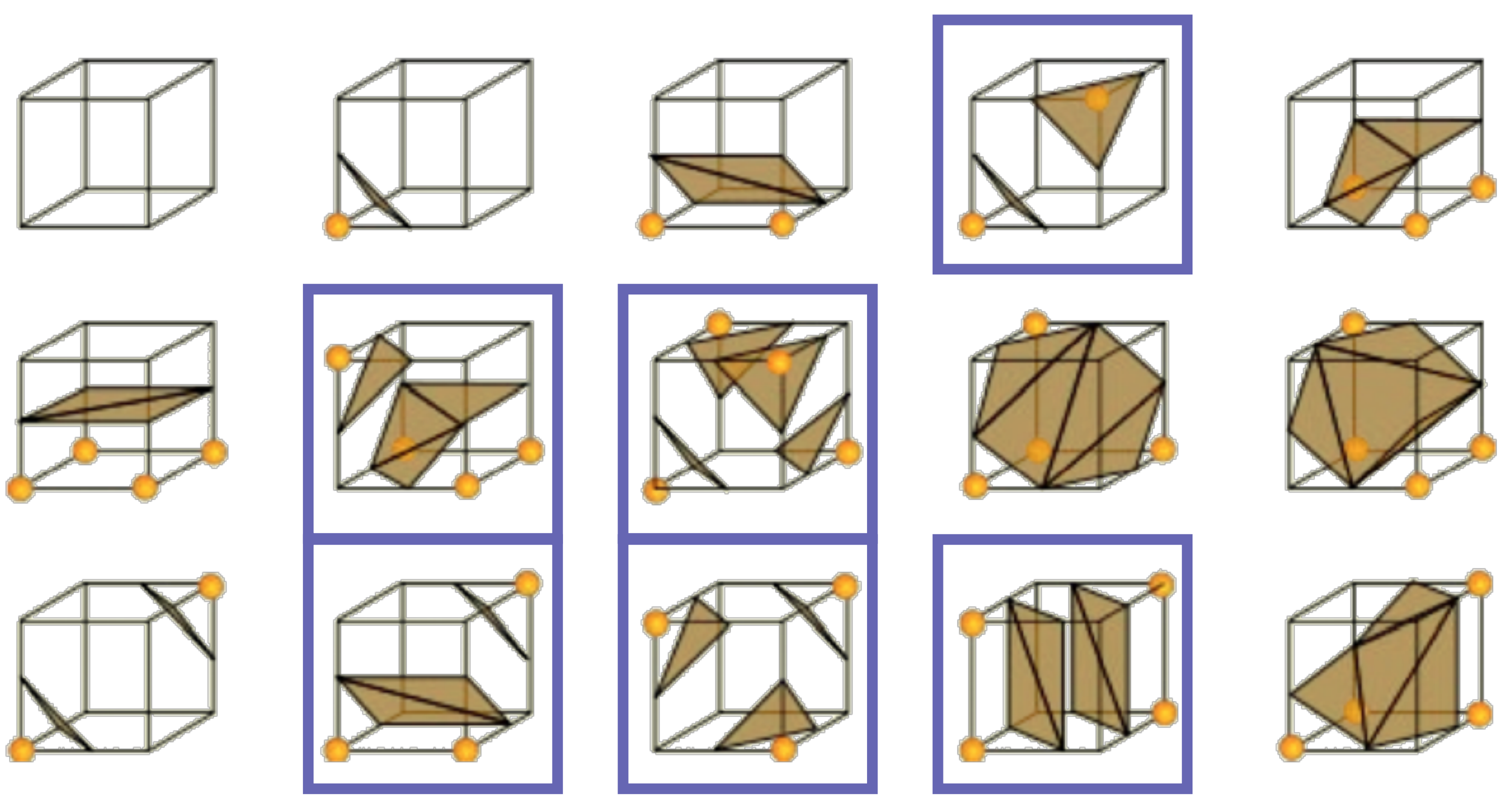
Two options:

1) Can resolve ambiguity by subsampling inside the cell.
2) If subsampling is impossible, pick one of the two possibilities.

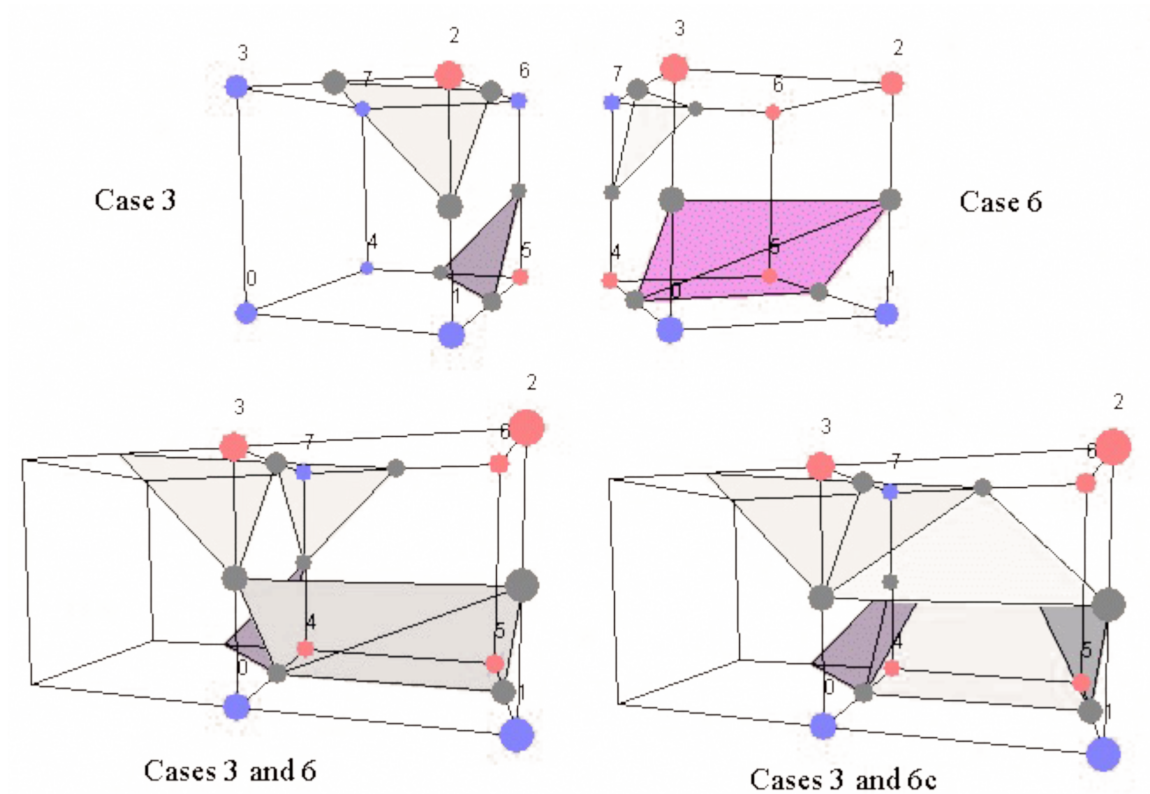
Marching Cubes (3D)

Same machinery: cells → **cubes** (voxels), lines → triangles

- 256 different cases - 15 after symmetries, 6 ambiguous cases
- More subsampling rules → 33 unique cases



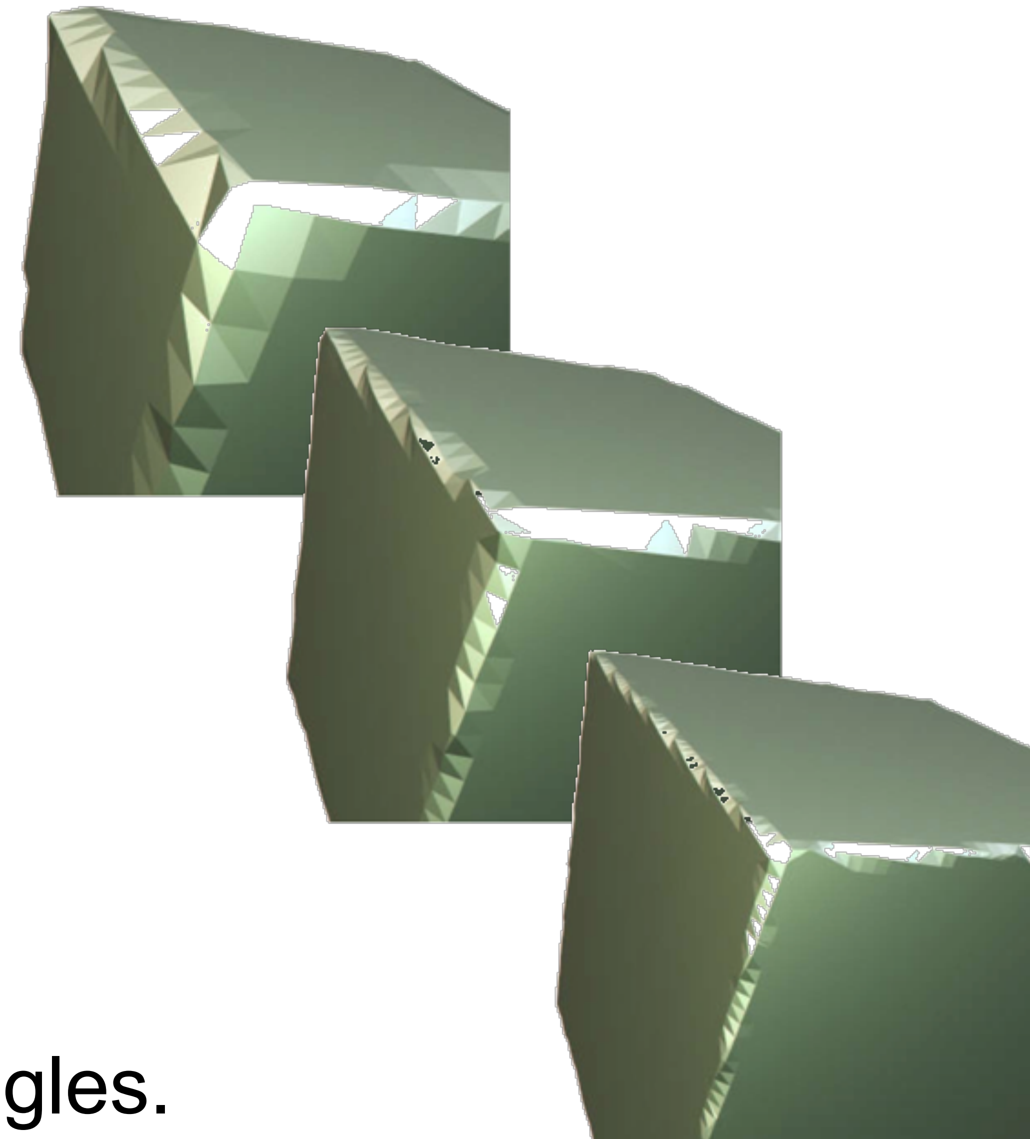
the 15 cases



explore ambiguity to avoid holes!

Chernyaev, Marching Cubes 33,’95

Marching Cubes (3D)



Main Strengths:

- Very multi-purpose.
- Extremely fast and parallelizable.
- Relatively simple to implement.
- Virtually parameter-free

Main Weaknesses:

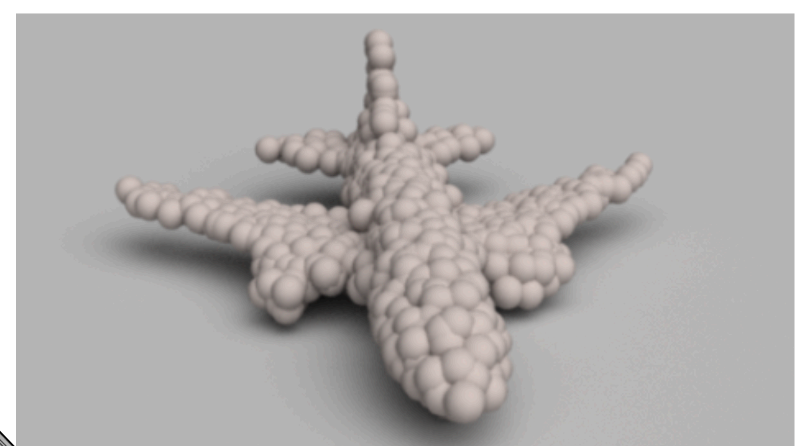
- Can create badly shaped (skinny) triangles.
- Many special cases (implemented as big lookup tables).
- No sharp features.

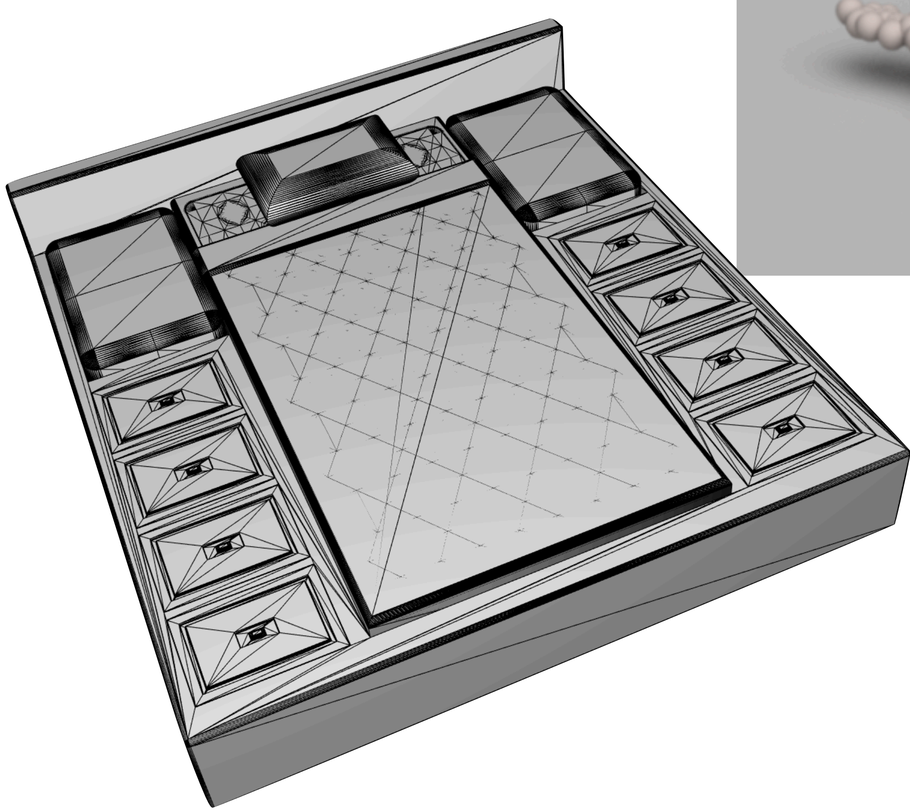
MESH-> POINT CLOUD

Sampling

From Surface to Point Cloud - Why?

- Points are simple but expressive!
  - Few points can suffice
- Flexible, unstructured, few constraints
- Also: ML applications!

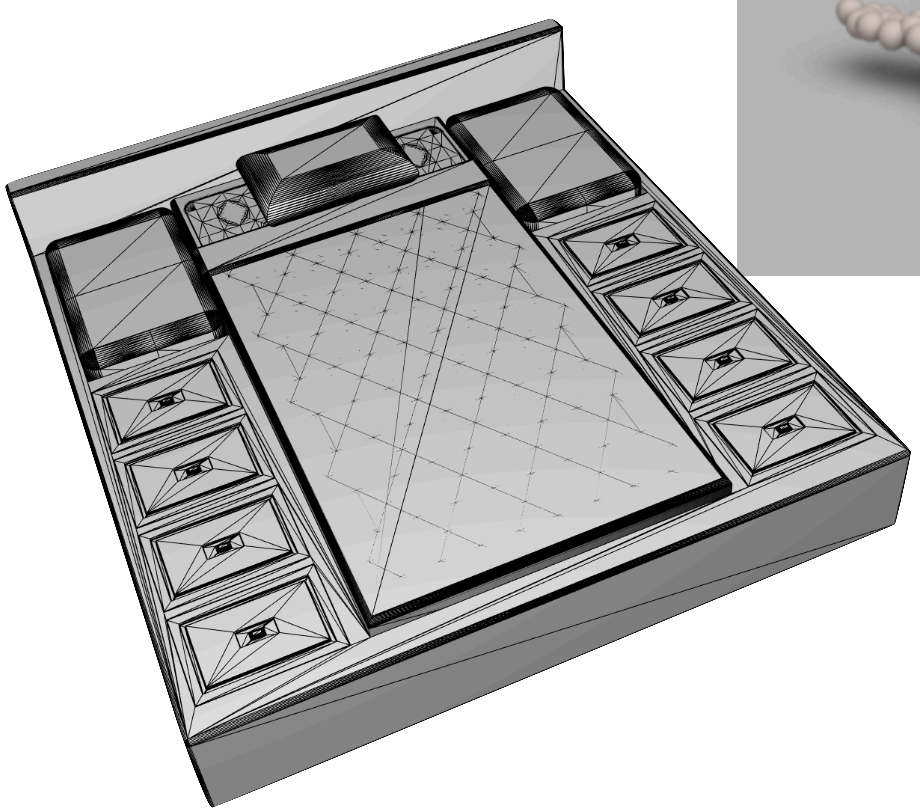


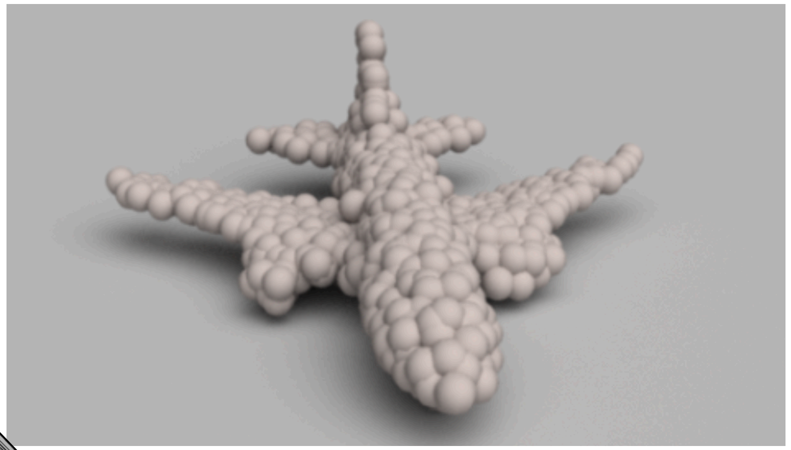


CAD meshes:
many components
bad triangles
connectivity problems

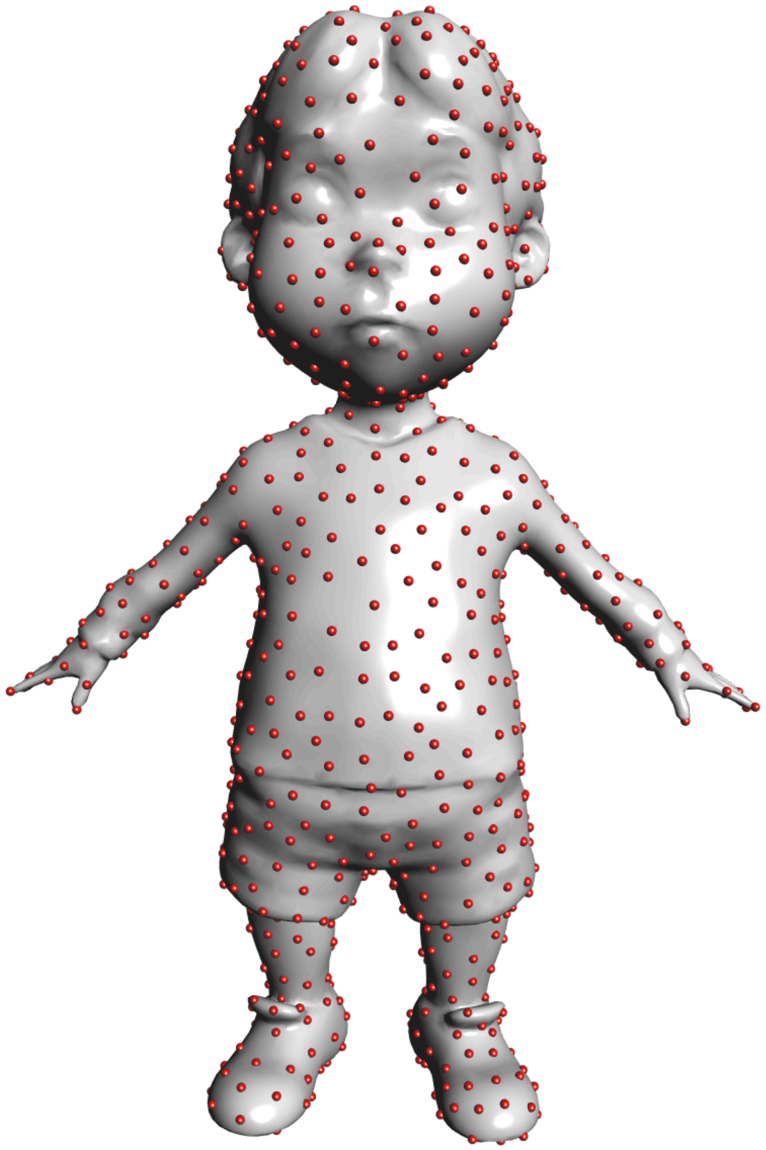
From Surface to Point Cloud - Why?

- Points are simple but expressive!
  - Few points can suffice
- Flexible, unstructured, few constraints
- Also: ML applications!





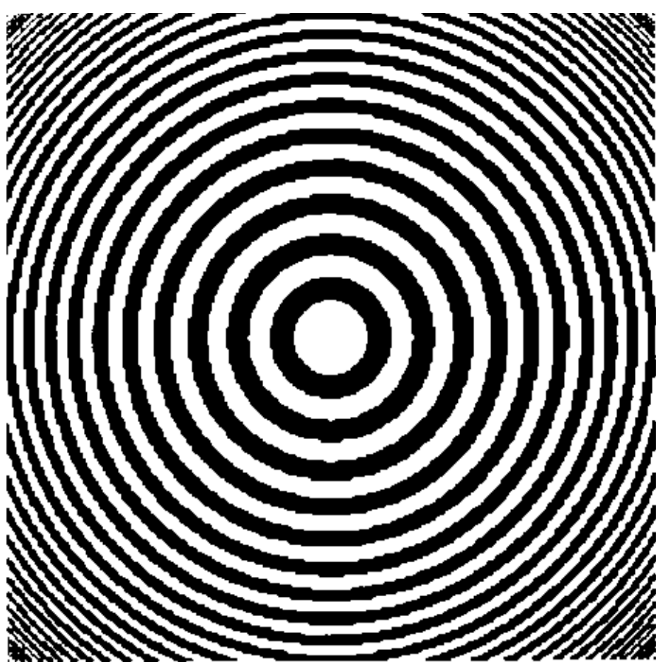
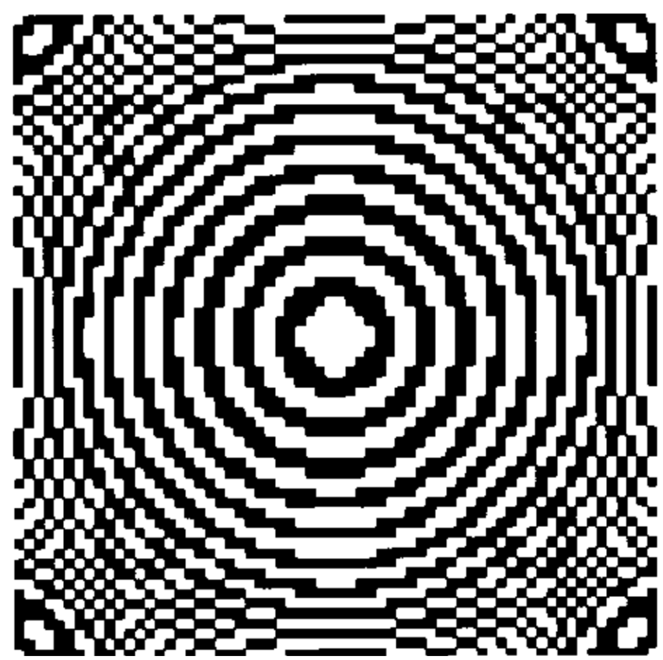
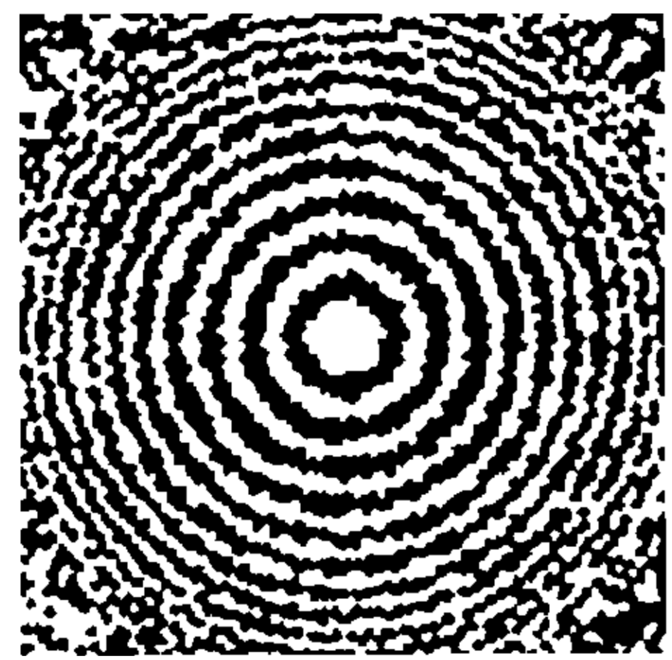
CAD meshes:
many components
bad triangles
connectivity problems



the problem:
sampling the mesh

Farthest Point Sampling

- Introduced for progressive transmission/acquisition of images
- Quality of approximation improves with increasing number of samples
  - as opposed eg. to raster scan
- Key Idea: repeatedly place next sample in the middle of the least-known area of the domain.

Gonzalez 1985, “Clustering to minimize the maximum intercluster distance”
Hochbaum and Shmoys 1985, “A best possible heuristic for the k-center problem”

Pipeline

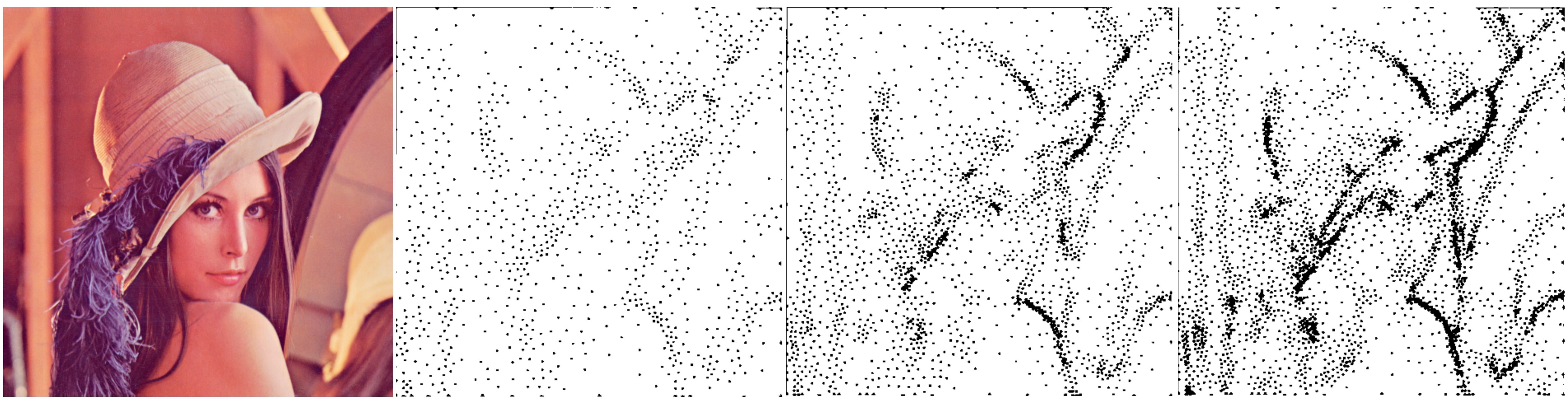
1. Create an initial sample point set *S*
   - Image corners + additional random point.
2. Find the point which is the farthest from all point in *S*

$$d(p, S) = \max_{q \in A}(d(q, S))$$
$$= \max_{q \in A}\left(\min_{0 \leq i < N}(d(q, s_i))\right)$$

3. Insert the point to *S* and update the distances
4. While more points are needed, iterate

Farthest Point Sampling

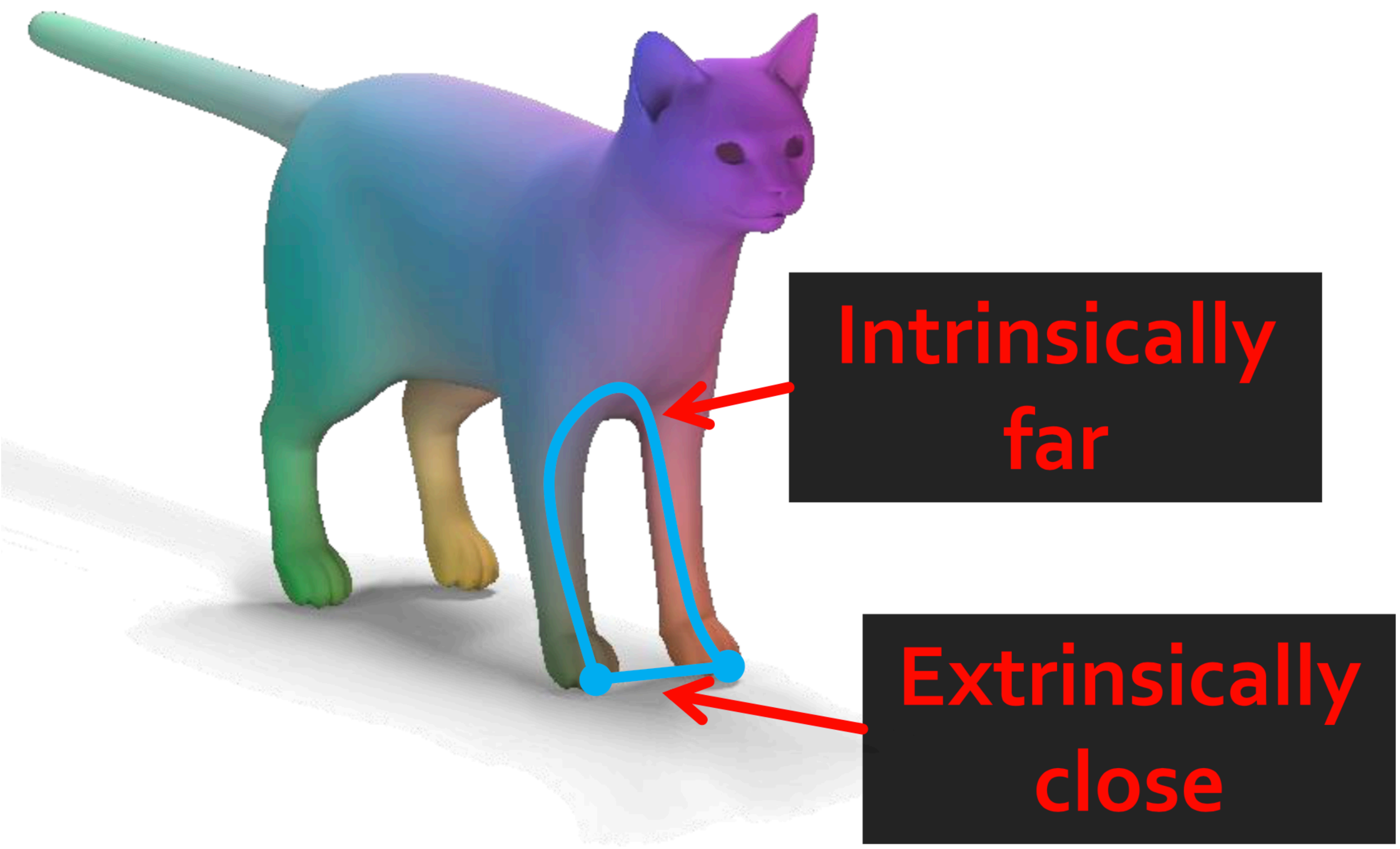
- Depends on a notion of distance on the sampling domain
- Can be made adaptive, via a weighted distance

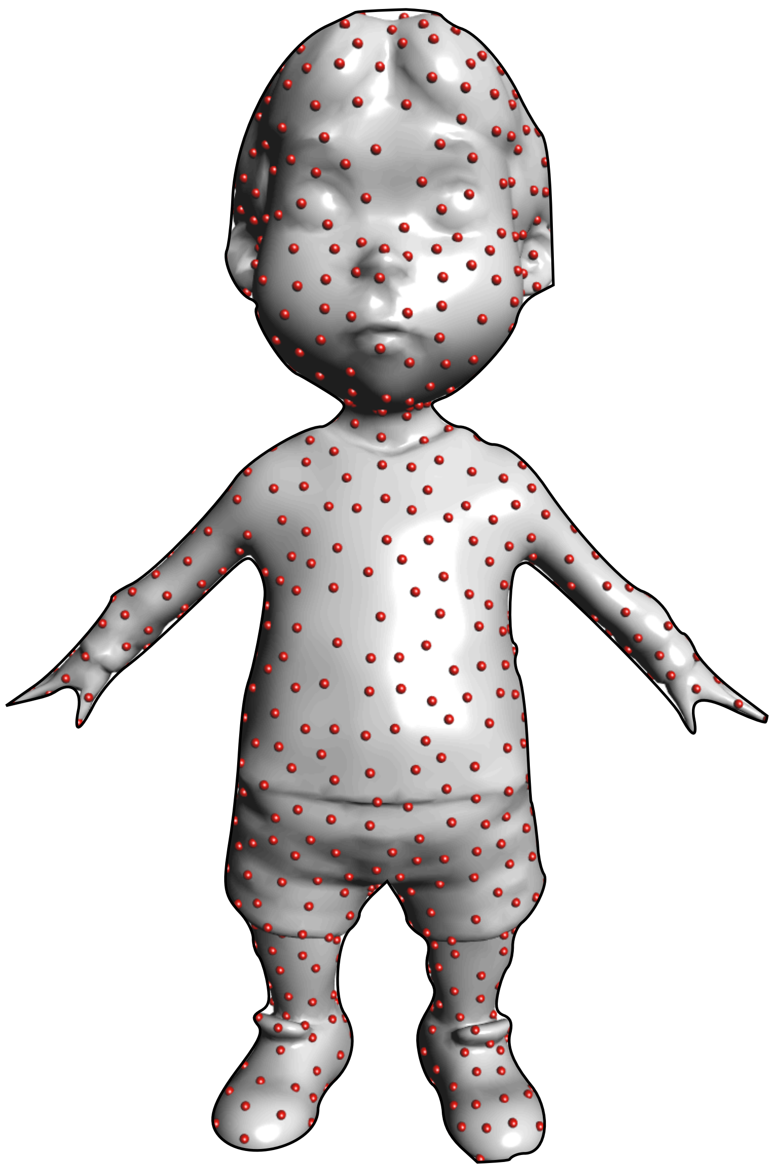


Eldar et al. 1997, “The Farthest Point Strategy for Progressive Image Sampling”

FPS on surfaces

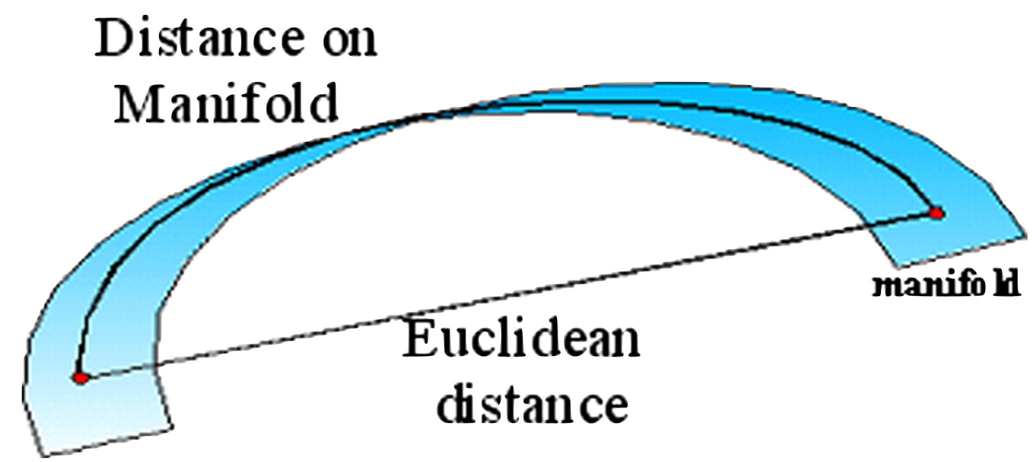
- What's an appropriate distance?

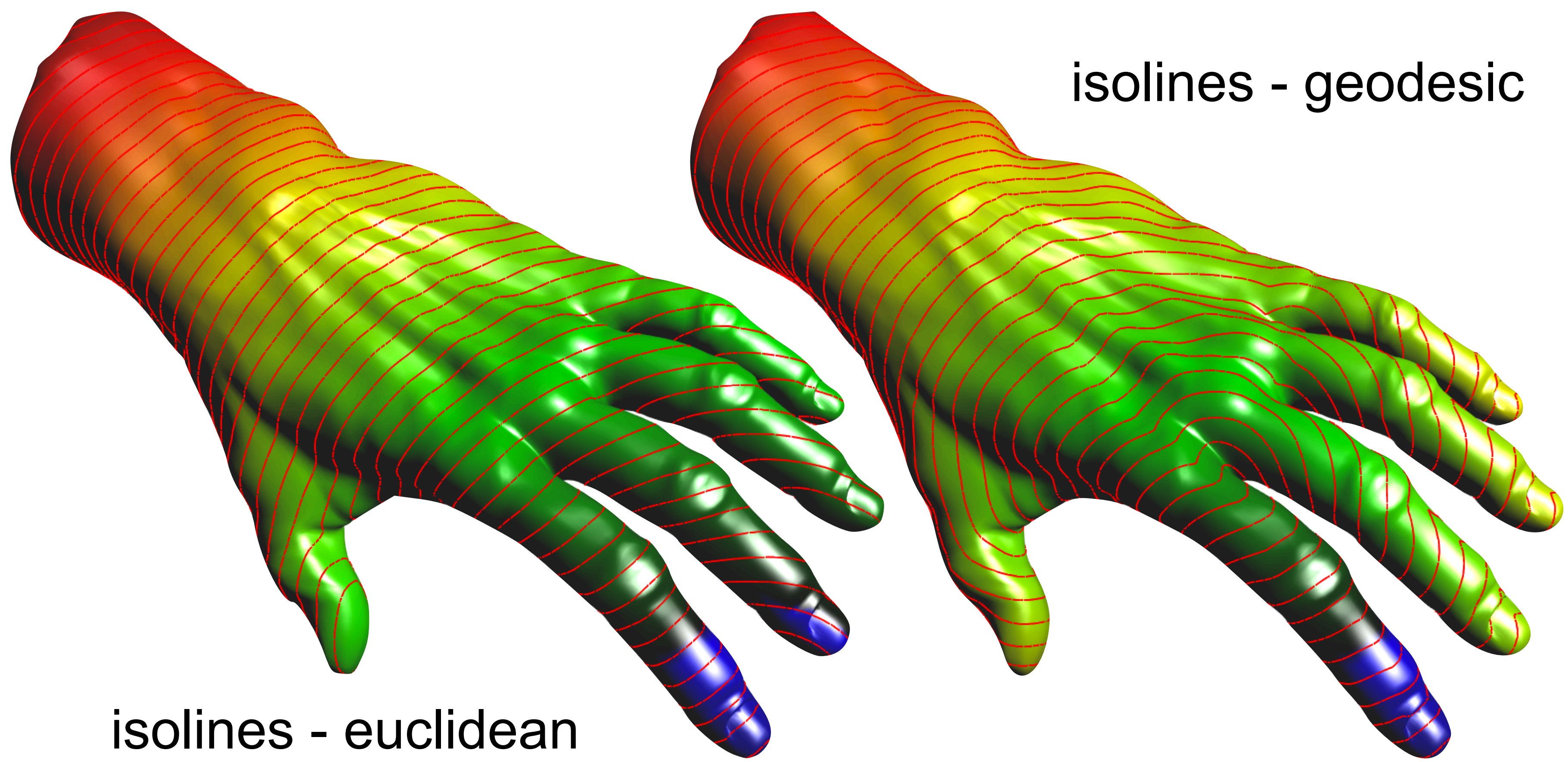




On-Surface Distances

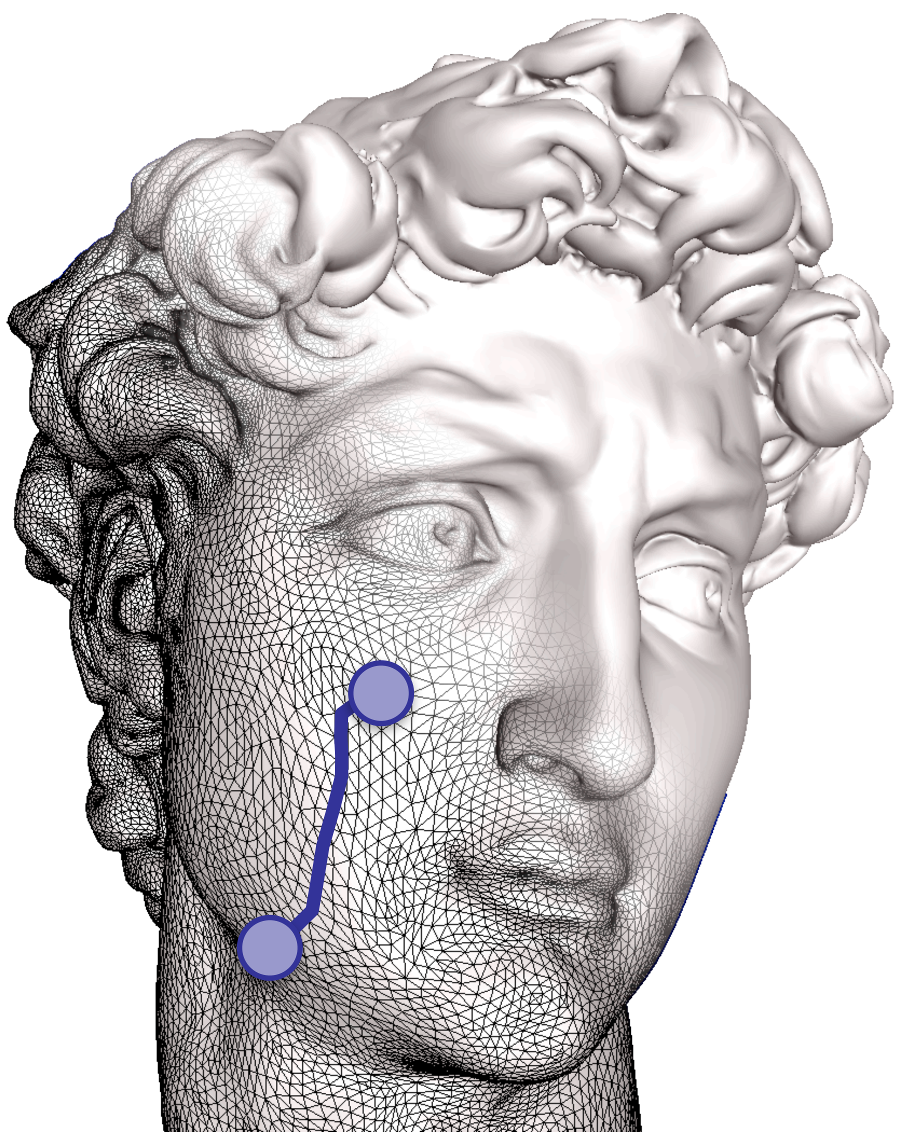
- Geodesics: Straightest and **locally shortest** curves





Discrete Geodesics

- Recall: a mesh is a graph!
- Approximate geodesics as paths along edges



v_0= initial vertex
d_i = current distance to vertex i
S = verticies with known optimal distance

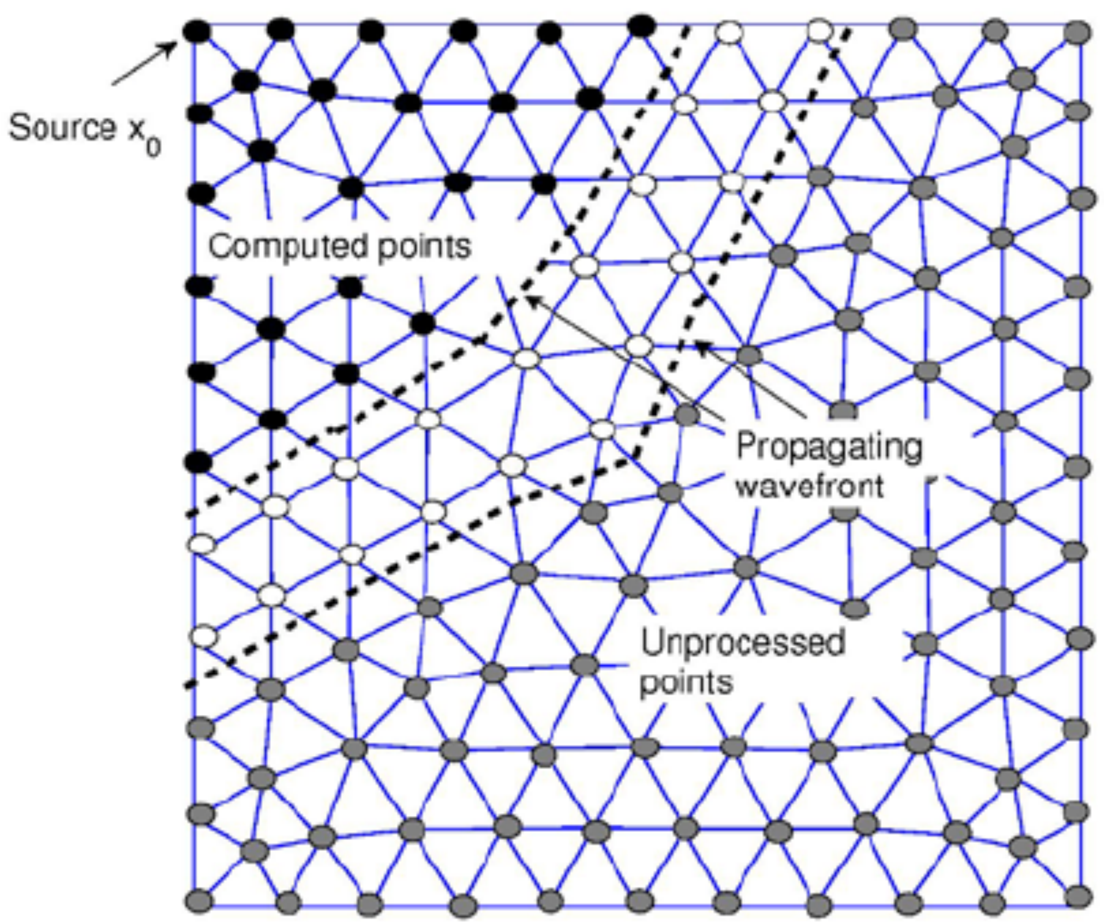
\# initialize
$d_0 = 0$
d_i = [inf for d in d_i]
$S = \{\}$

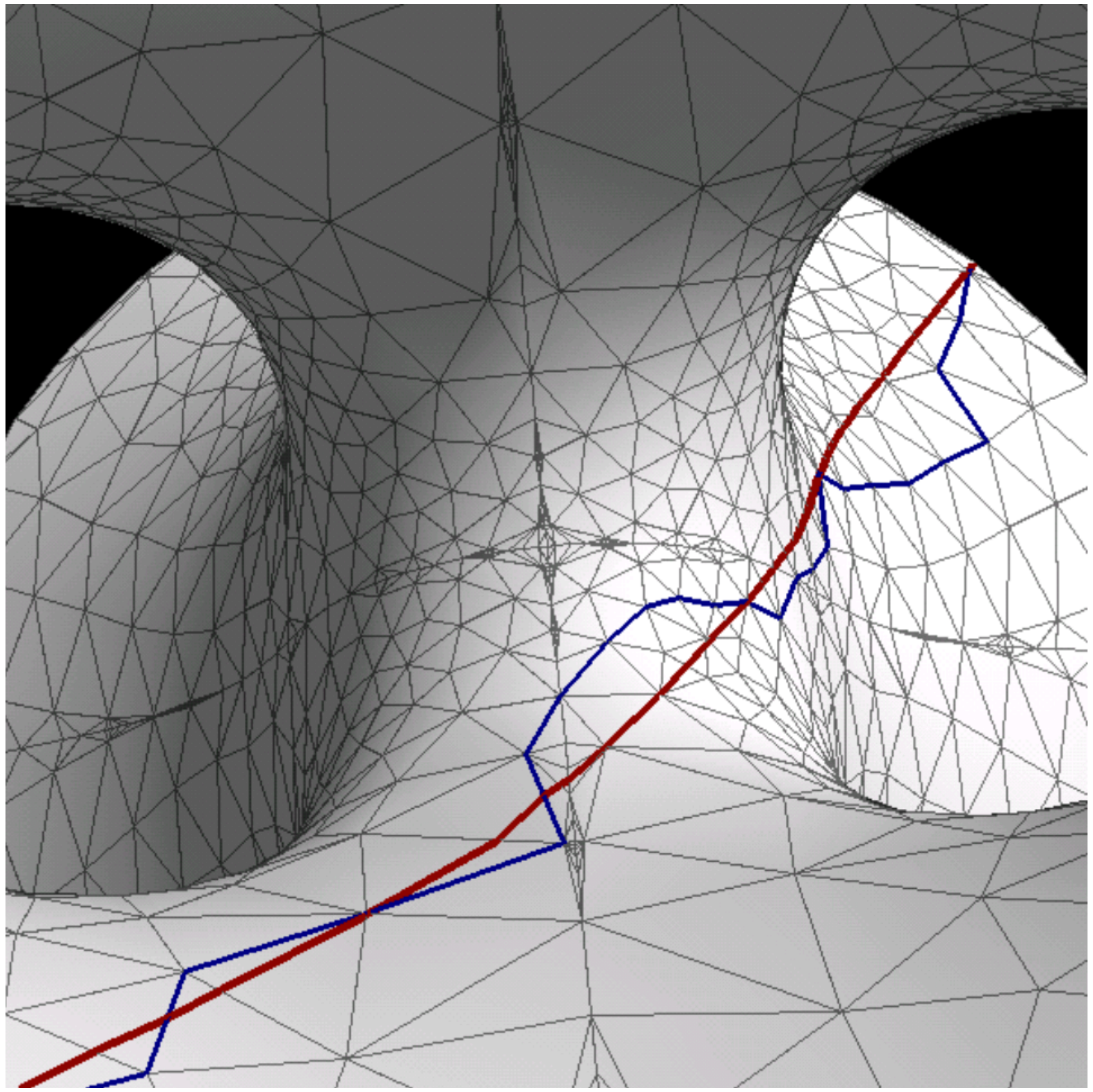
for each iteration k:
    \# update
    k = argmin(d_k), for v_k not in S
    S.append(v_k)
    for neighbors index v_l of v_k:
        d_l = min([d_l, d_k + d_{kl}])

Dijkstra's algorithm!

Fast Marching Geodesics

- A better approximation: allow fronts to cross triangles!





Kimmel and Sethian 1997, “Computing Geodesic Paths on Manifolds”

Software

- Libigl http://libigl.github.io/libigl/tutorial/tutorial.html
  - MATLAB-style (flat) C++ library, based on indexed face set structure
- OpenMesh www.openmesh.org
  - Mesh processing, based on half-edge data structure
- CGAL www.cgal.org
  - Computational geometry
- MeshLab http://www.meshlab.net/
  - Viewing and processing meshes

Software

- Alec Jacobson's GP toolbox
  - https://github.com/alecjacobson/gptoolbox
  - MATLAB, various mesh and matrix routines
- Gabriel Peyre's Fast Marching Toolbox
  - https://www.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching
  - On-surface distances (more next time!)
- OpenFlipper https://www.openflipper.org/
  - Various GP algorithms + Viewer

Topology of Surfaces

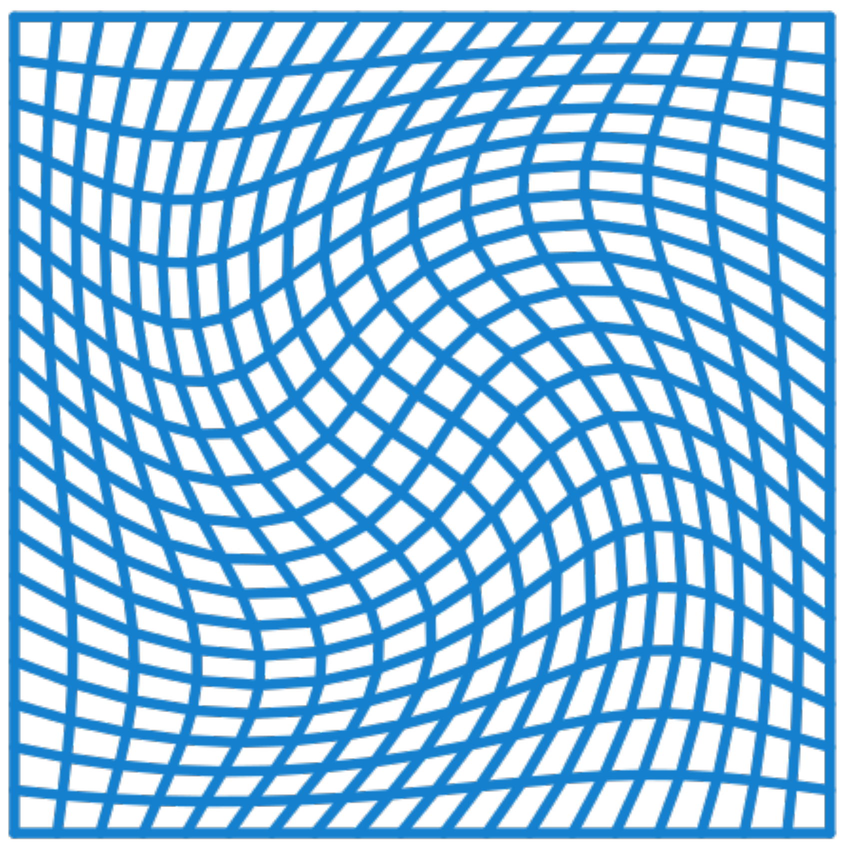
(slides mostly by Glenn Eguchi)

Topology of Surfaces

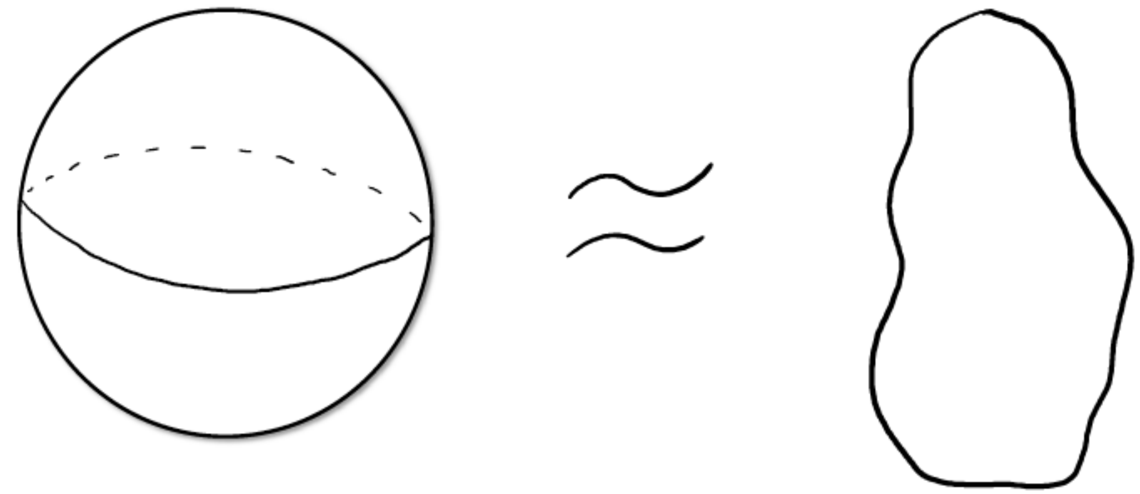
- We will say two surfaces M and N are *topologically equivalent* or are of the same topological type if

M and N are diffeomorphic

- We normally write $M \approx N$ to denote this



A square under a diffeomorphism from the square onto itself.

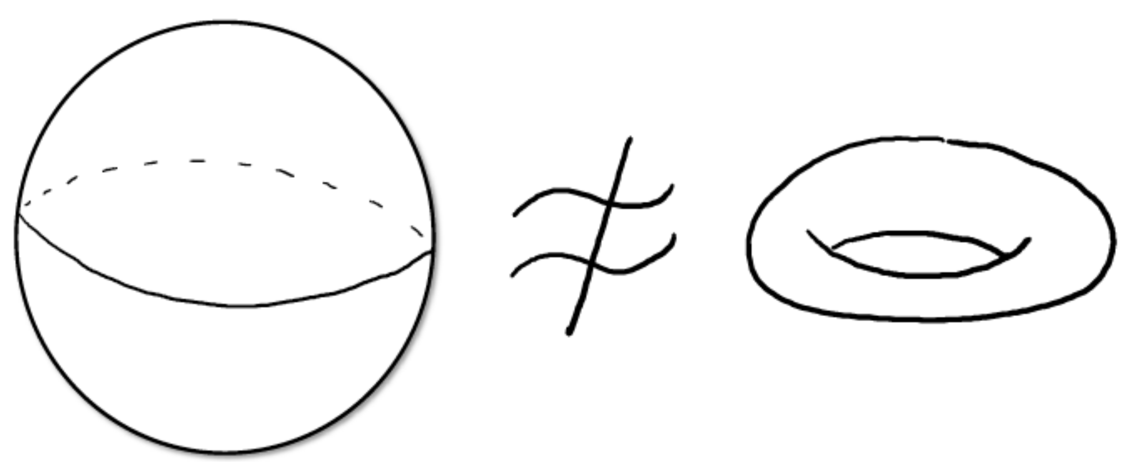


Sphere

Surface of Potato

torus = surfare of doughnut

coffee cup



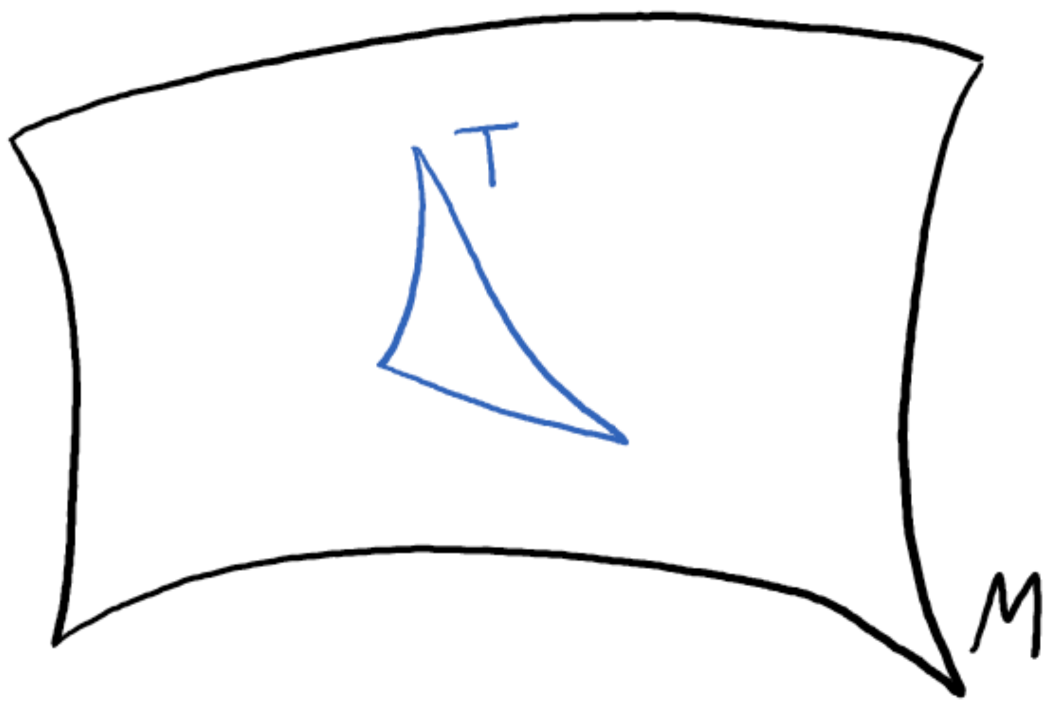
Topology by a Single Number

- It's an astonishing fact that to determine whether two surfaces are topologically equivalent (diffeomorphic) comes down to computing exactly one number of that surface.

- That number is called the **Euler characteristic**.

Triangulation

- A *triangle* T in M is a simple region in M bounded by 3 smooth curve segments.



- Here `simple' means that T is topologically a disk

Triangulation

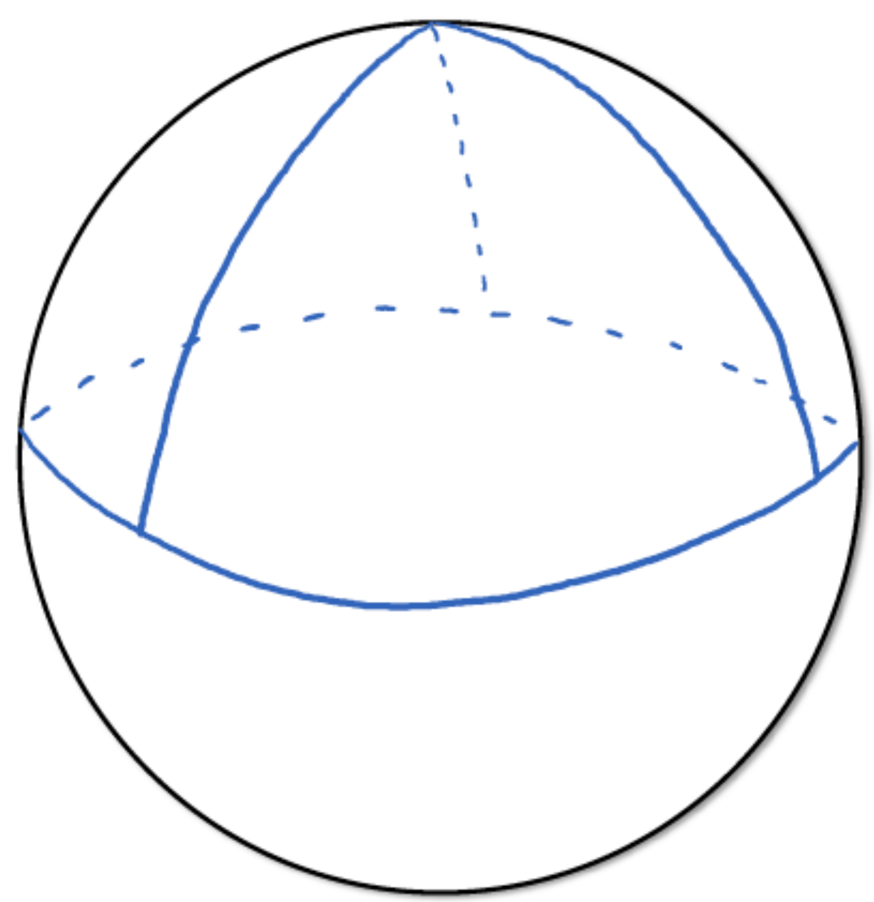
- A *triangulation* of M is a decomposition of M into a finite number of triangles $T_1, T_2, \ldots, T_n$ such that

(1) $\bigcup_{i=1}^{n} T_i = M$

(2) If $T_i \cap T_j \neq \emptyset$, then $T_i \cap T_j$ is either a common edge or a vertex.

- It's a fact (we will not prove) that every compact surface can be triangulated.

Example. The figure below shows a triangulation of the sphere. Note that the edges of the triangles are great circles and hence geodesics. The triangulation has the same topology type as a tetrahedron. The number of faces is $F = 4$, the number of edges is $E = 6$, and the number of vertices is $V = 4$.



Tetrahedron

Euler Characteristic

Definition. Let M be a compact surface and consider any triangulation of M. Then the *Euler characteristic* of M is

$$\chi(M) = V - E + F$$

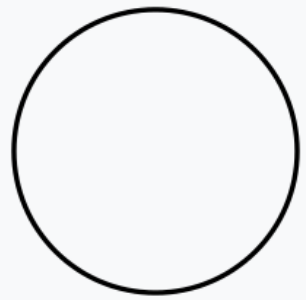
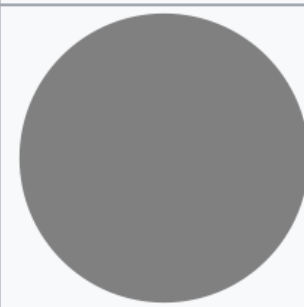
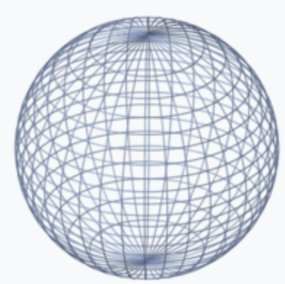
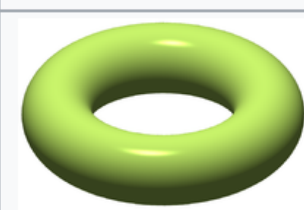
where

$$\begin{aligned} V &= \text{number of vertices} \\ E &= \text{number of edges} \\ F &= \text{number of faces} \end{aligned}$$

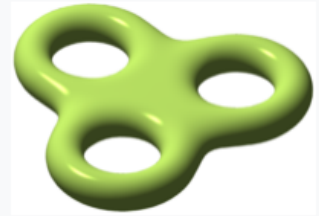
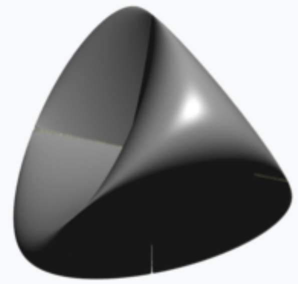
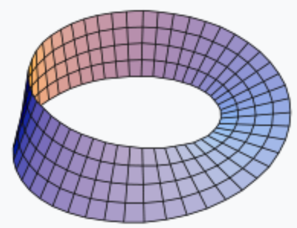
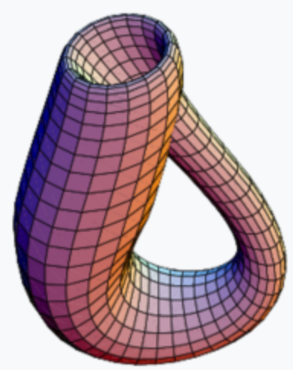
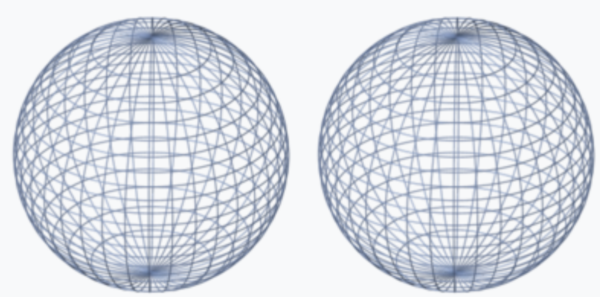
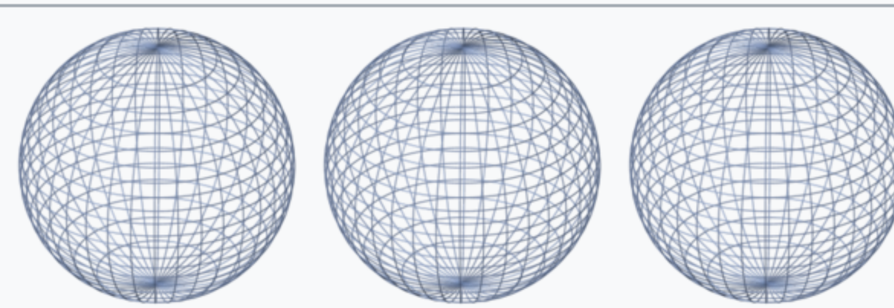
The following fact (we will not prove) justifies the definition of $\chi(M)$.

Theorem 6.9. *The Euler characteristic $\chi(M)$ does not depend on the particular triangulation of M.*

Examples

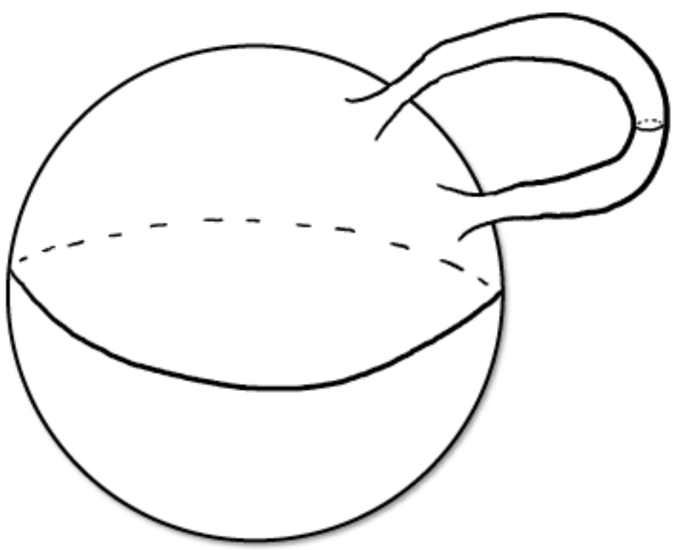
Name	Image	Euler characteristic
Interval		1
Circle		0
Disk		1
Sphere		2
Torus (Product of two circles)		0
Double torus		−2

Examples

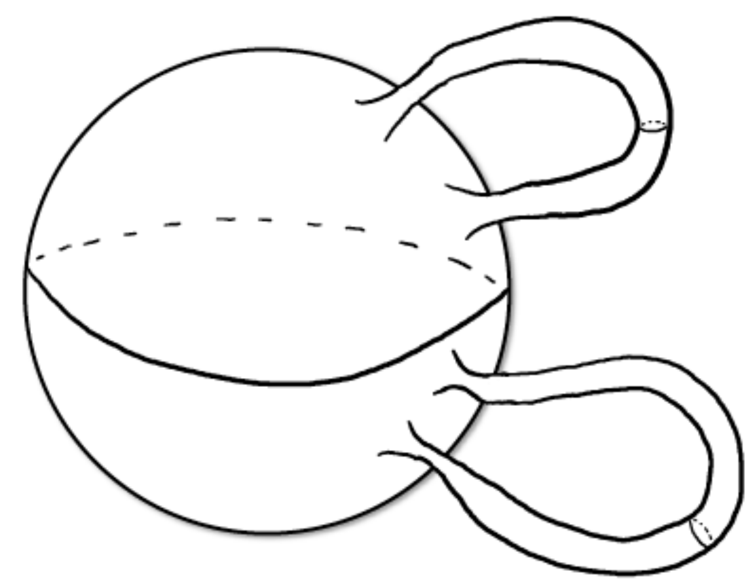
Triple torus		**−4**
Real projective plane		**1**
Möbius strip		**0**
Klein bottle		**0**
Two spheres (not connected) (Disjoint union of two spheres)		2 + 2 = **4**
Three spheres (not connected) (Disjoint union of three spheres)		2 + 2 + 2 = **6**

Genus

- There is an easy way to construct surfaces with different topology. The idea is to `glue' handles onto a sphere.



Sphere with one handle attached = genus one surface



Sphere with two handles attached = genus two surface

Definition. When we construct a surface M in this way with g handles, then we say M is a *surface of genus* g.

Genus and Euler characteristics

Proposition 6.11. *If M is a surface of genus g, then $\chi(M) = 2(1-g)$.*

The proof follows from a formula involving the *connected sum* of two surfaces: $\chi(M_1 \# M_2) = \chi(M_1) + \chi(M_2) - 2$.

A more general relationship in high-dimensional space:

$$\chi(M \# N) = \chi(M) + \chi(N) - \chi(S^n)$$