UC San Diego

Shape Representations

Instructor: Hao Su

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

POINTS → IMPLICIT

Implicit Surface Reconstruction

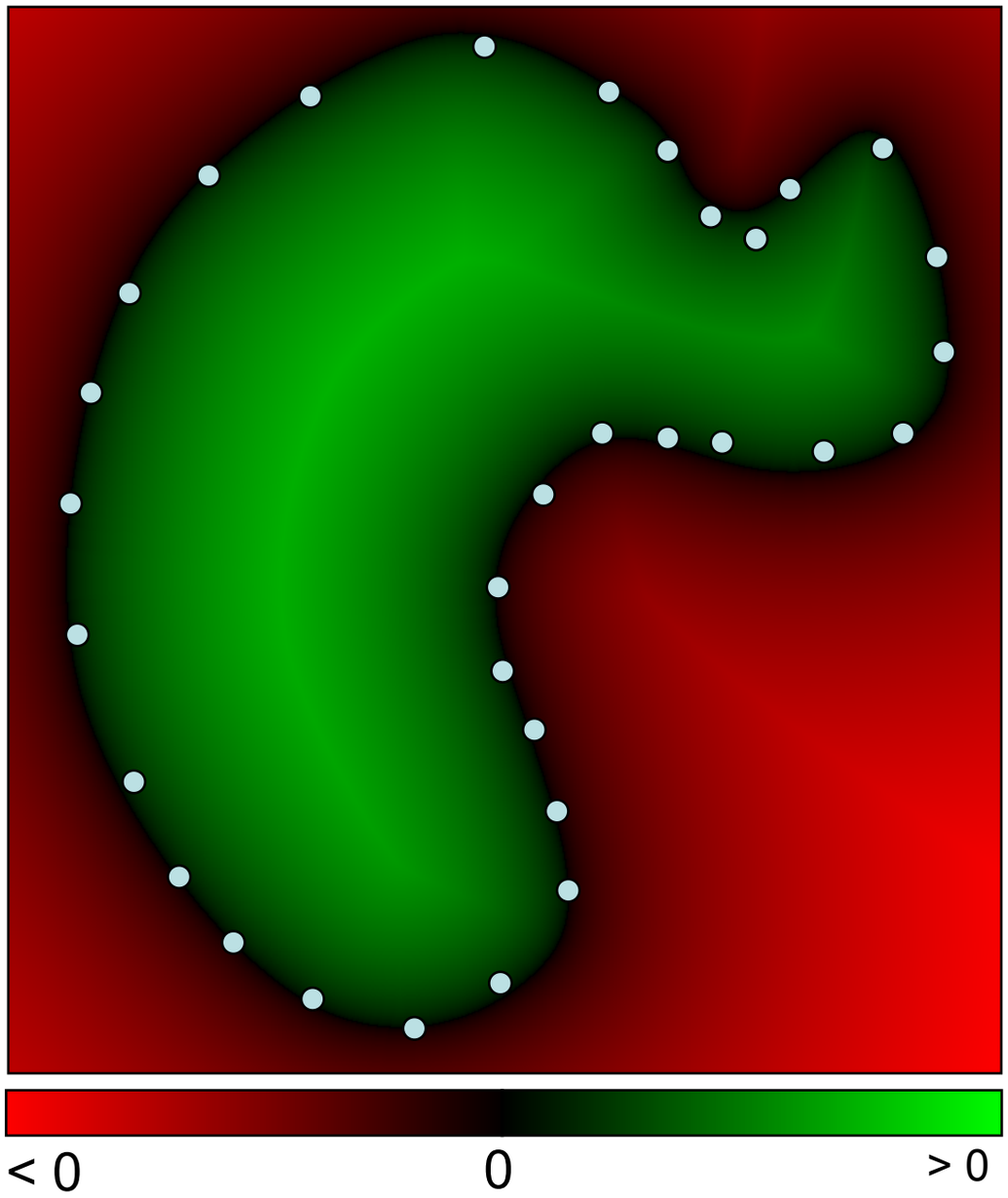
UC San Diego

Implicit Function Approach

- Define a function

$$f : R^3 \rightarrow R$$

with value < 0 outside the shape and > 0 inside

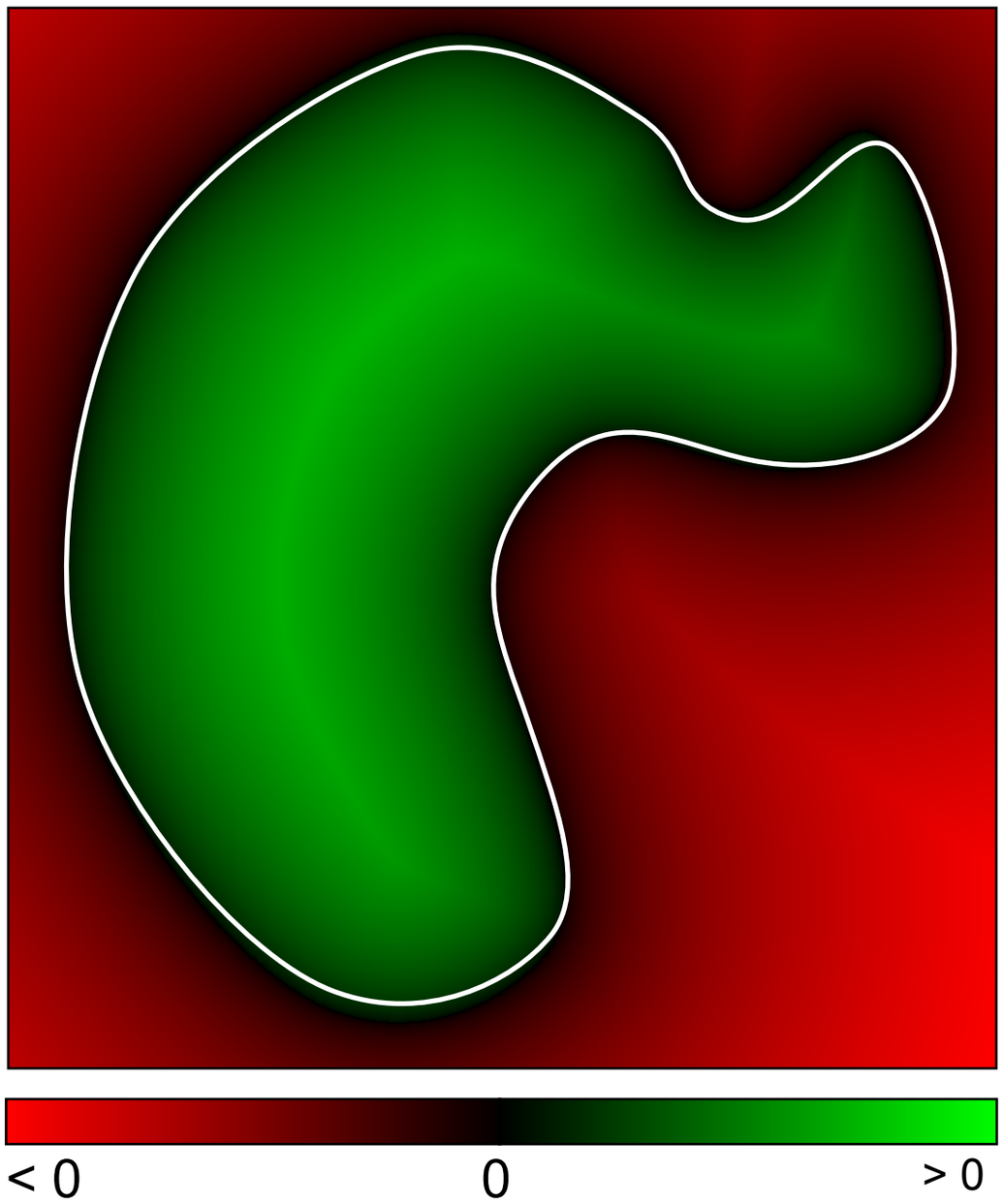


Implicit Function Approach

- Define a function

$$f : R^3 \rightarrow R$$

with value < 0 outside
the shape and > 0 inside

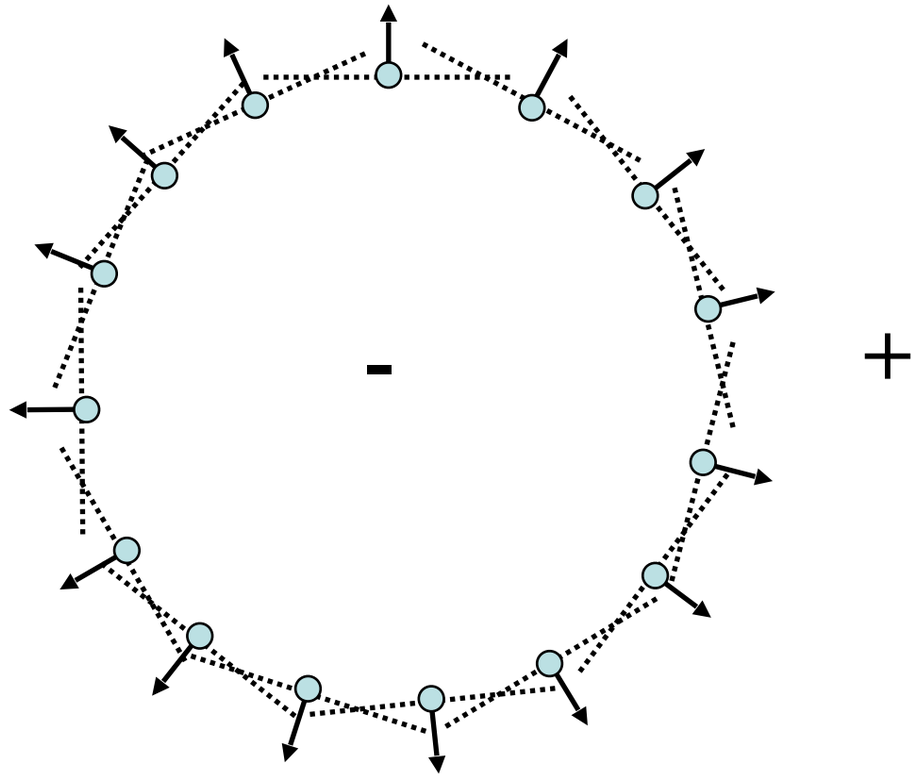


- Extract the zero-set

$$\{x : f(x) = 0\}$$

SDF from Points and Normals

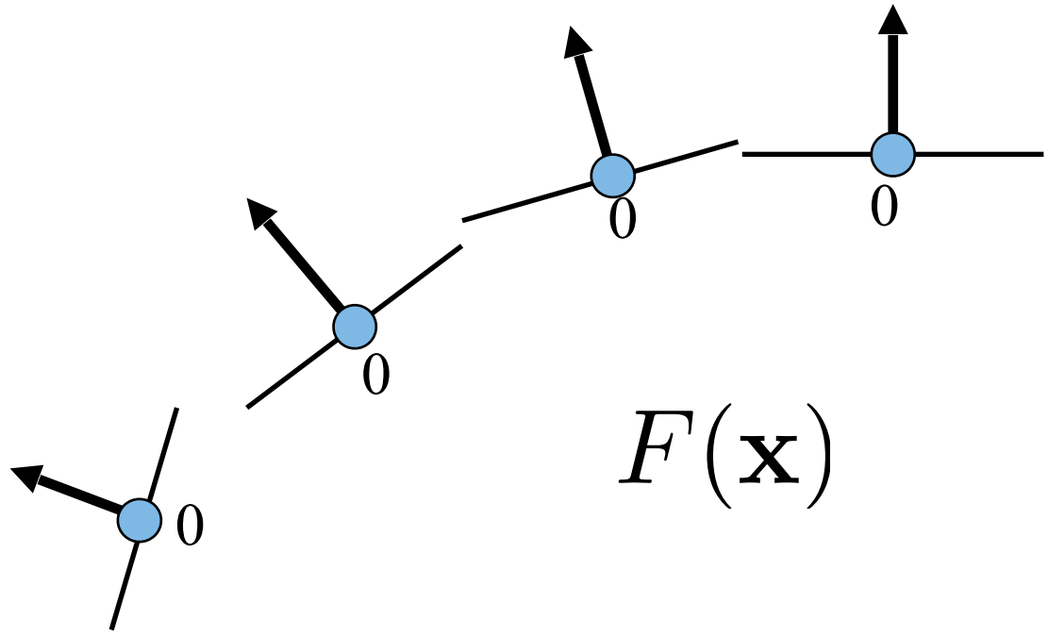
- Input: Points + Normals
- Normals help to distinguish between inside and outside
- Computed via locally fitting planes at the points



“Surface reconstruction from unorganized points”, Hoppe et al., ACM SIGGRAPH 1992
http://research.microsoft.com/en-us/um/people/hoppe/proj/recon/

Smooth SDF

- Find smooth implicit F.
- Scattered data interpolation:
  - $F(\mathbf{p}_i) = 0$
  - F is smooth
  - Avoid trivial $F \equiv 0$

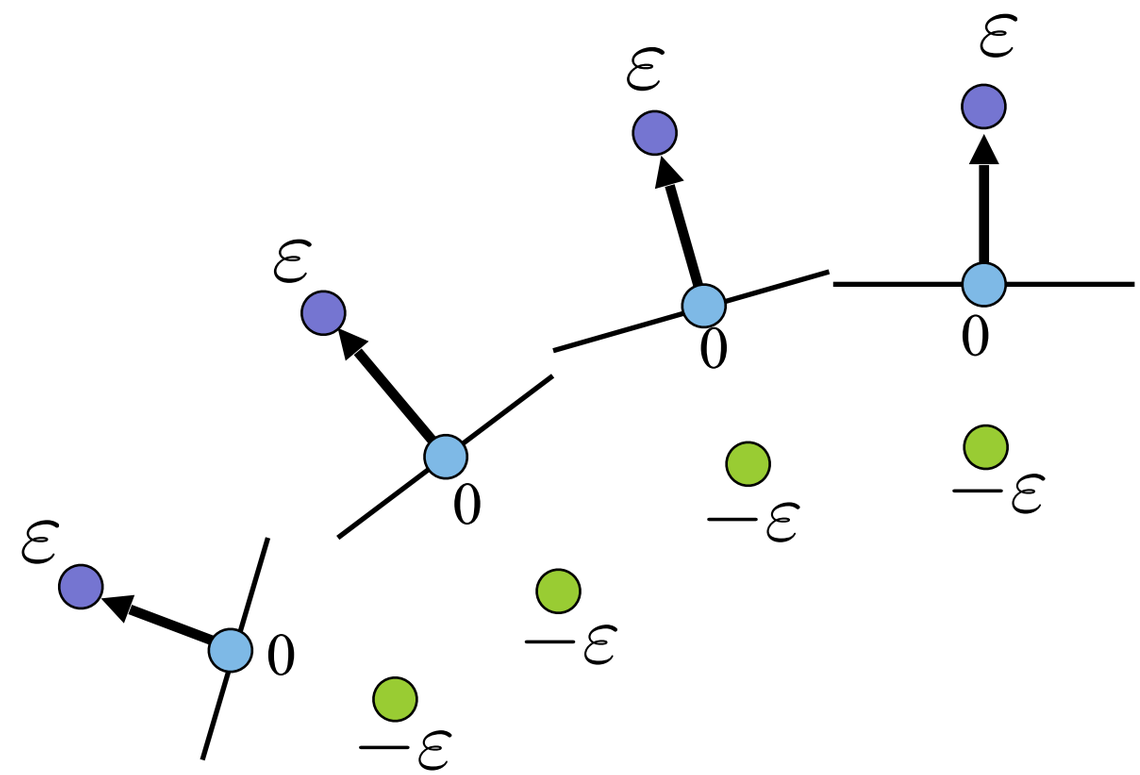


“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

Smooth SDF

- Scattered data interpolation:
  - $F(\mathbf{p}_i) = 0$
  - F is smooth
  - Avoid trivial $F \equiv 0$

- Add off-surface constraints



$$F(\mathbf{p}_i + \varepsilon \mathbf{n}_i) = \varepsilon$$

$$F(\mathbf{p}_i - \varepsilon \mathbf{n}_i) = -\varepsilon$$

“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

Radial Basis Function Interpolation

- RBF: Weighted sum of shifted, smooth kernels

$$F(\mathbf{x}) = \sum_{i=0}^{N-1} w_i \, \varphi(\|\mathbf{x} - \mathbf{c}_i\|) \quad N = 3n$$

Scalar weights
Unknowns

Smooth kernels
(basis functions)
centered at constrained
points.
For example:

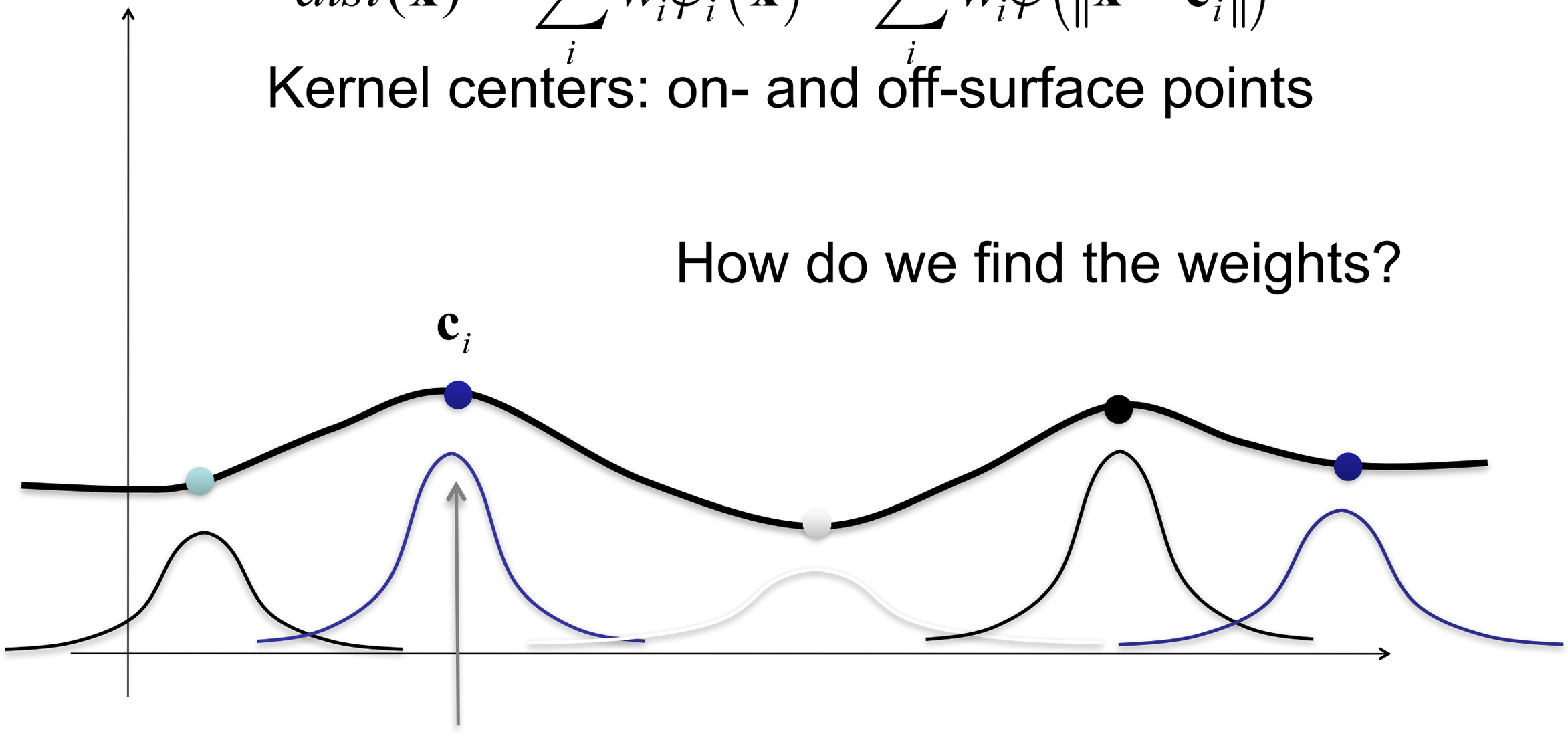
$$\varphi(r) = r^3$$

Radial Basis Function Interpolation

$$dist(\mathbf{x}) = \sum_i w_i \varphi_i(\mathbf{x}) = \sum_i w_i \varphi(\|\mathbf{x} - \mathbf{c}_i\|)$$

Kernel centers: on- and off-surface points

How do we find the weights?

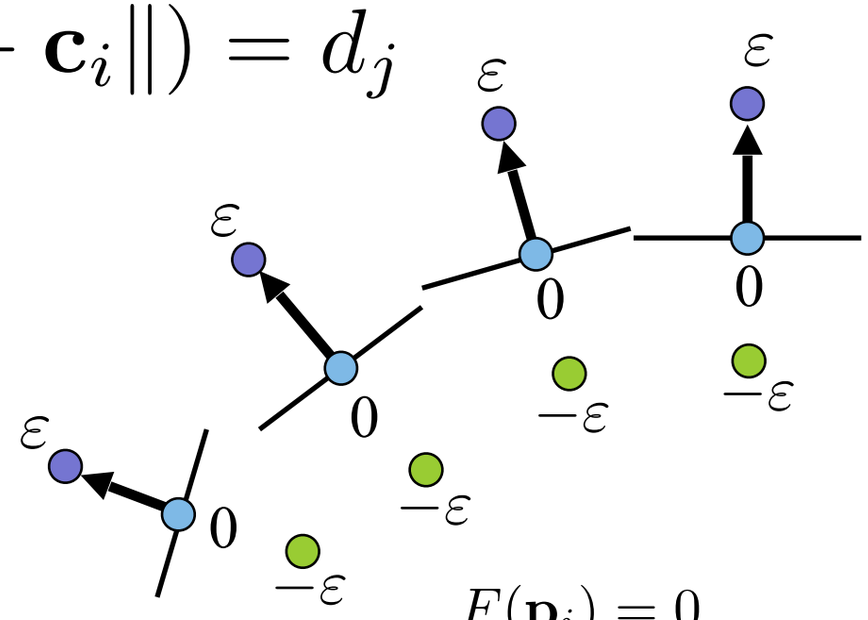


Radial Basis Function Interpolation

- Interpolate the constraints:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon\mathbf{n}_i,\ \mathbf{p}_i - \varepsilon\mathbf{n}_i\}$$

$$\forall j = 0, \ldots, N-1, \quad \sum_{i=0}^{N-1} w_i\, \varphi(\|\mathbf{c}_j - \mathbf{c}_i\|) = d_j$$



$$F(\mathbf{p}_i) = 0$$
$$F(\mathbf{p}_i + \varepsilon\mathbf{n}_i) = \varepsilon$$
$$F(\mathbf{p}_i - \varepsilon\mathbf{n}_i) = -\varepsilon$$

Radial Basis Function Interpolation

- Interpolate the constraints:

$$\{\mathbf{c}_{3i}, \mathbf{c}_{3i+1}, \mathbf{c}_{3i+2}\} = \{\mathbf{p}_i,\ \mathbf{p}_i + \varepsilon\mathbf{n}_i,\ \mathbf{p}_i - \varepsilon\mathbf{n}_i\}$$

  - Symmetric linear system to get the weights:

$$\begin{pmatrix} \varphi(\|\mathbf{c}_0 - \mathbf{c}_0\|) & \cdots & \varphi(\|\mathbf{c}_0 - \mathbf{c}_{N-1}\| \\ \vdots & \ddots & \vdots \\ \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_0\|) & \cdots & \varphi(\|\mathbf{c}_{N-1} - \mathbf{c}_{N-1}\|) \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_{N-1} \end{pmatrix} = \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix}$$

$3n$ equations

$3n$ variables

RBF Kernels

- Globally supported $\varphi(r) = r^3$
- Leads to dense symmetric linear system
- C2 smoothness
- Works well for highly irregular sampling

RBF Kernels

- Polyharmonic spline

$$\varphi(r) = r^k \log(r),\ \ k = 2, 4, 6 \ldots$$

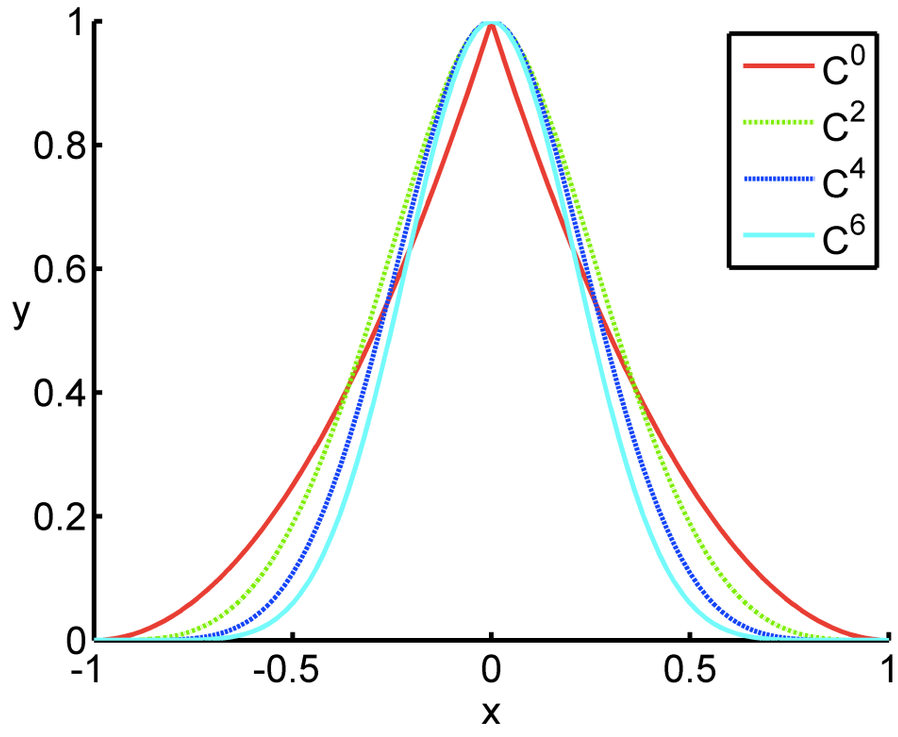
$$\varphi(r) = r^k,\ \ k = 1, 3, 5 \ldots$$

- Multiquadratic

$$\varphi(r) = \sqrt{r^2 + \beta^2}$$

- Gaussian

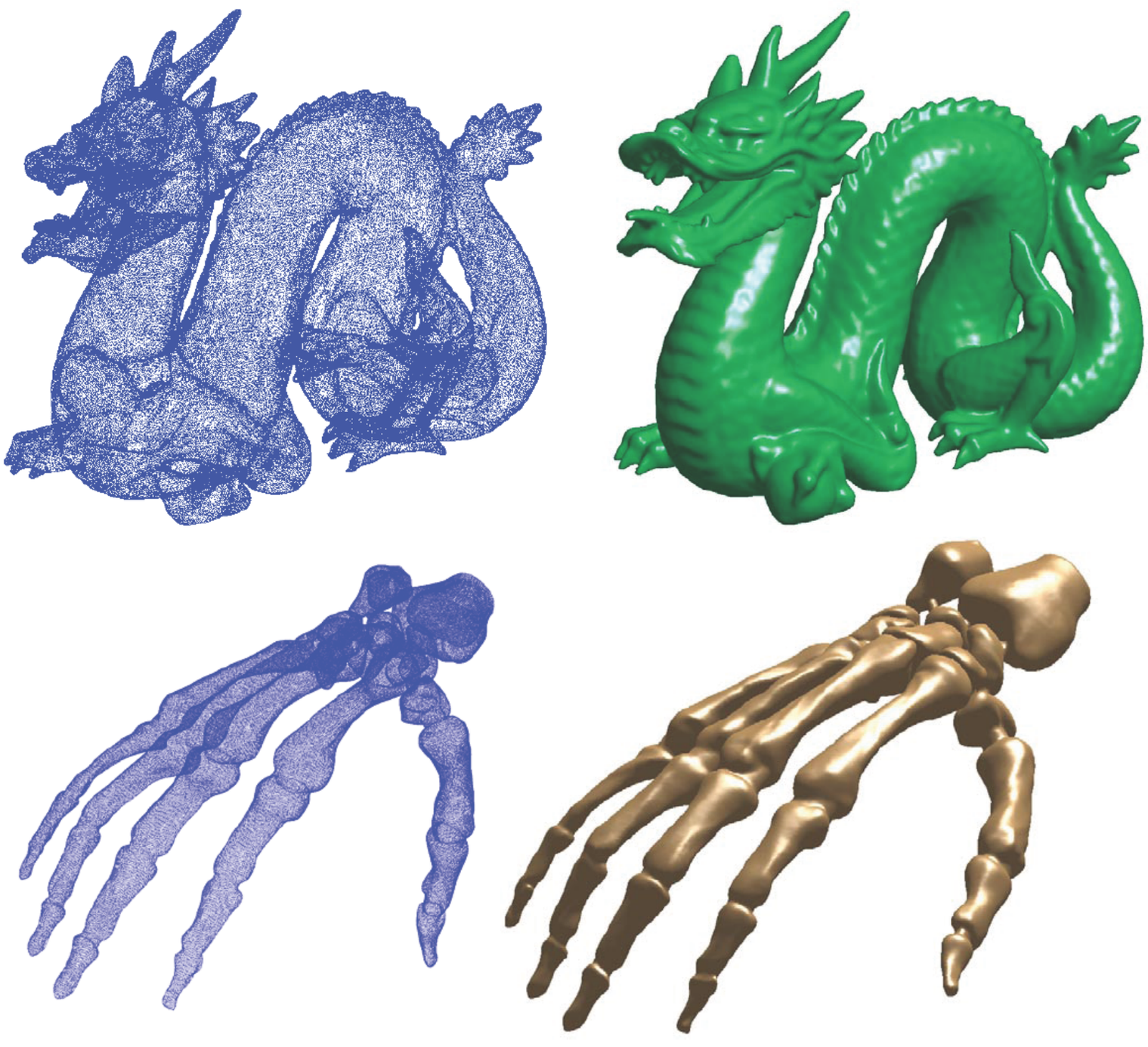
$$\varphi(r) = e^{-\beta r^2}$$



- B-Spline (compact support)

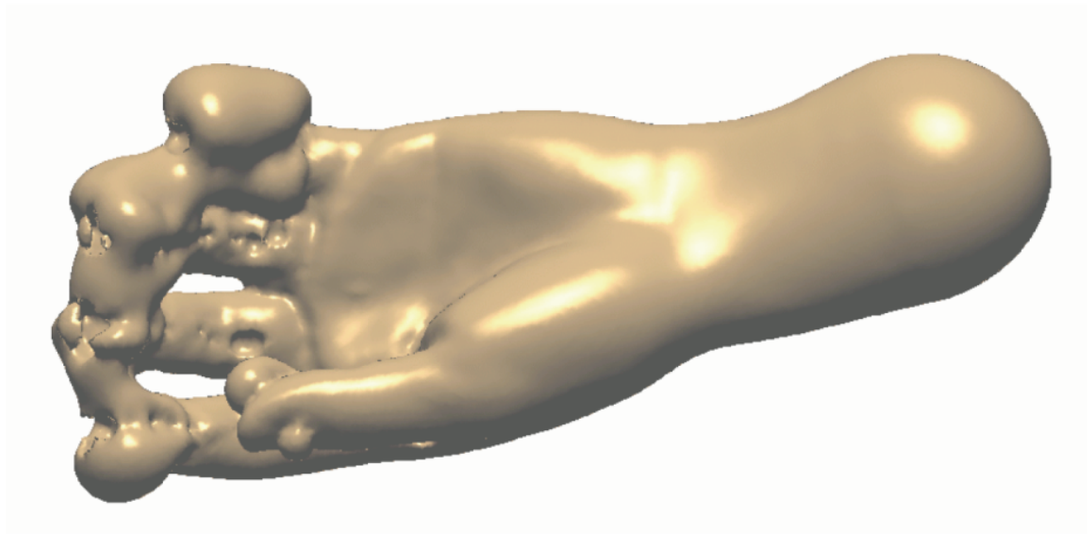
$$\varphi(r) = \text{piecewise-polynomial}(r)$$

RBF Reconstruction Examples

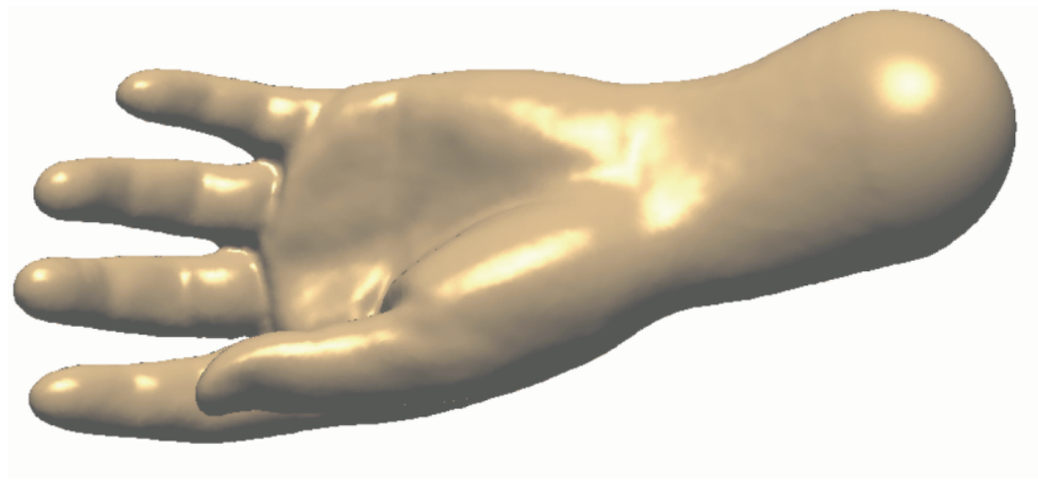


“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

Off-Surface Points



Insufficient number/
badly placed off-surface points



Properly chosen off-surface points

“Reconstruction and representation of 3D objects with radial basis functions”, Carr et al., ACM SIGGRAPH 2001

IMPLICIT → MESH

Marching Cubes

UC San Diego

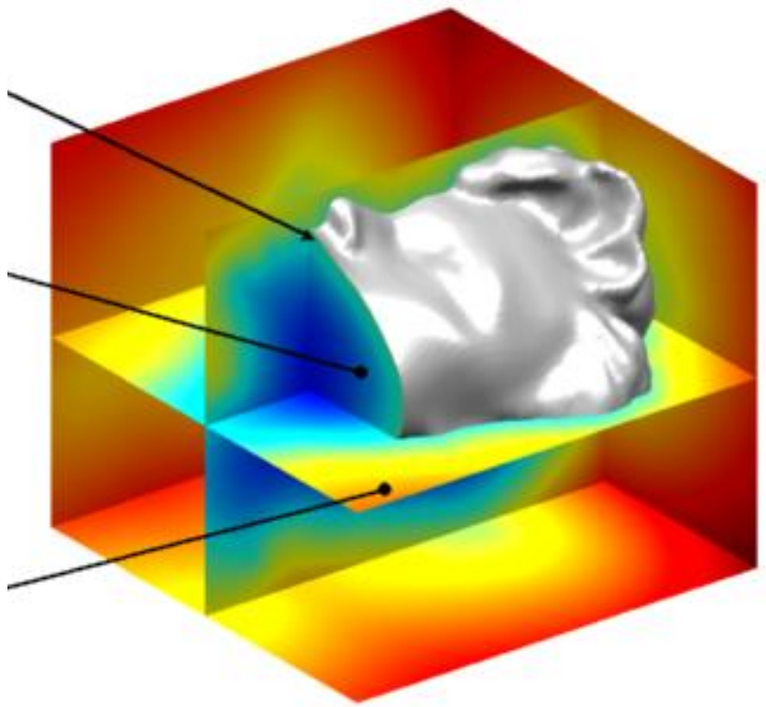
Extracting the Surface

- Wish to compute a manifold mesh of the level set

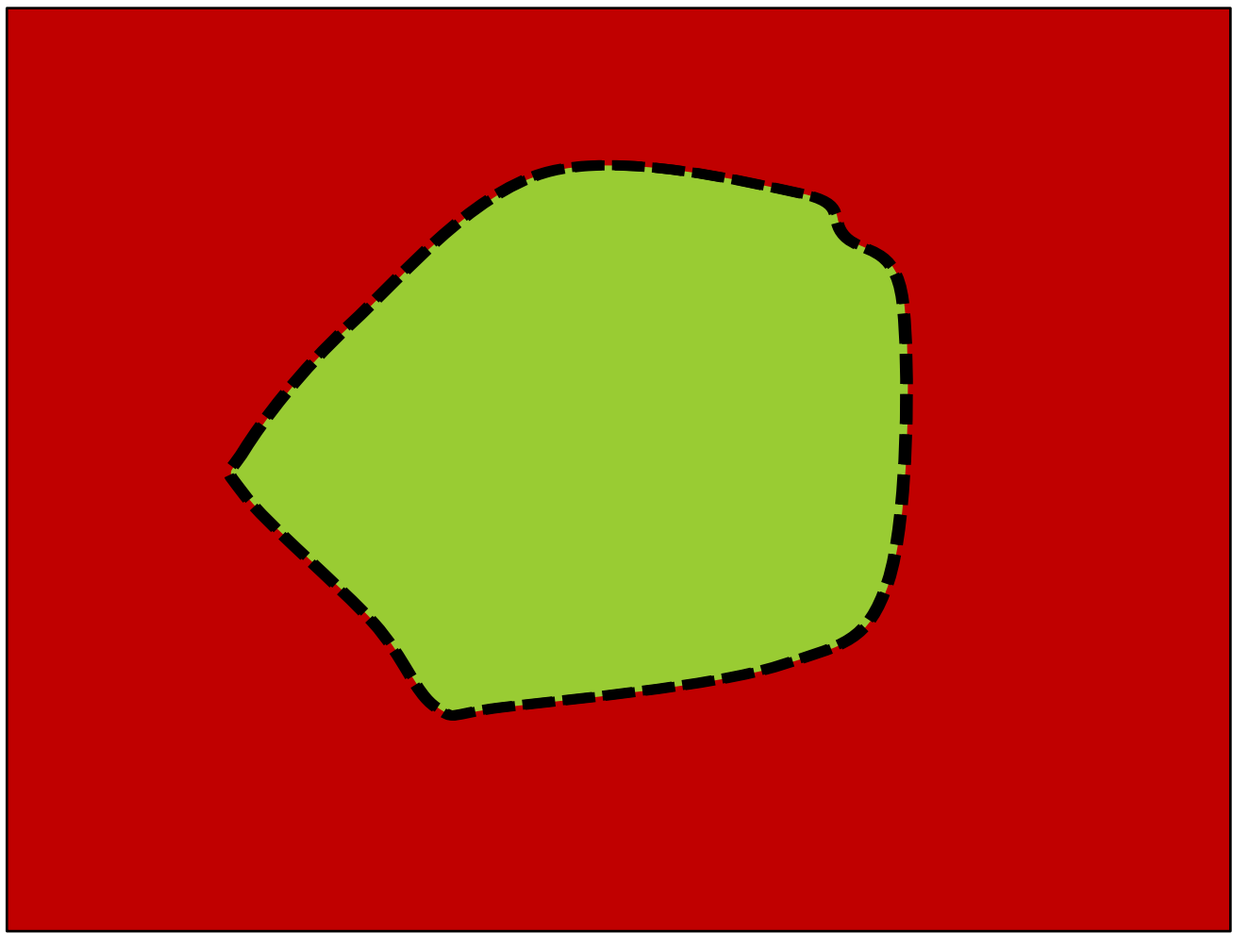
$F(\mathbf{x}) = 0 \rightarrow$ surface

$F(\mathbf{x}) < 0 \rightarrow$ inside

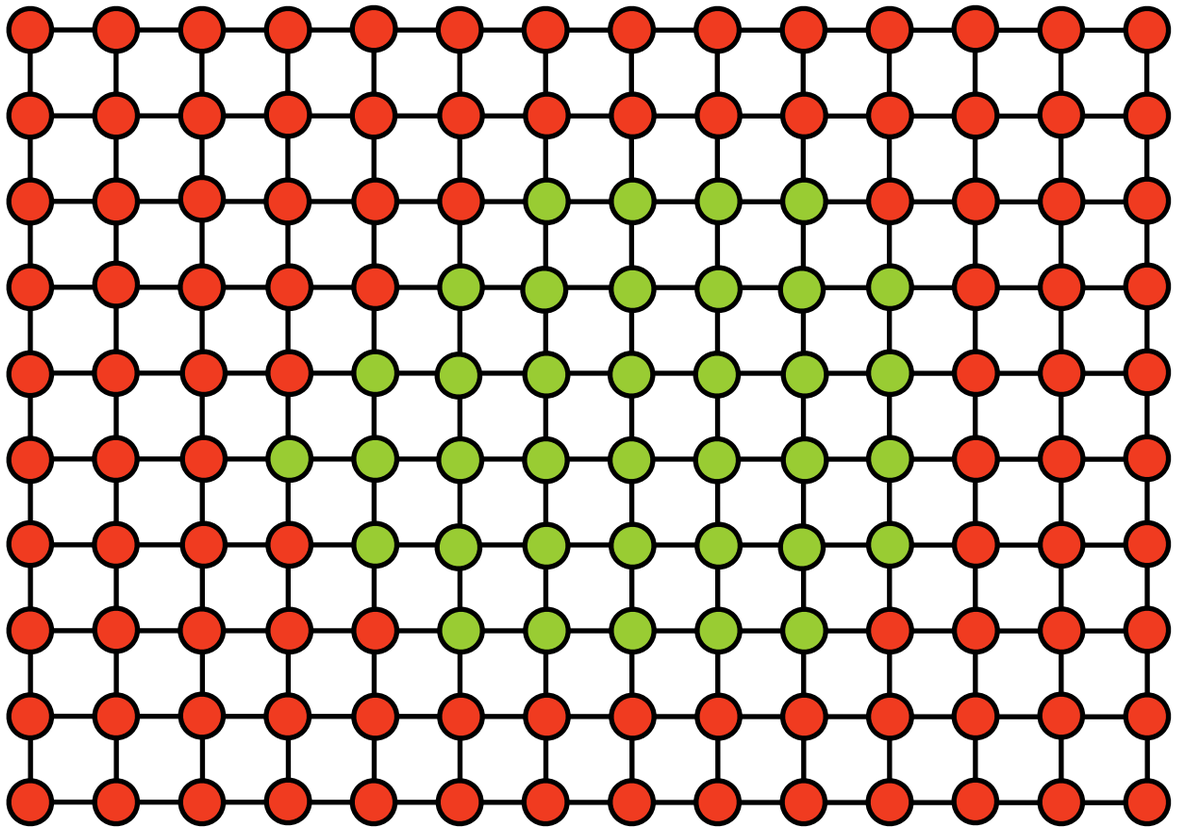
$F(\mathbf{x}) > 0 \rightarrow$ outside



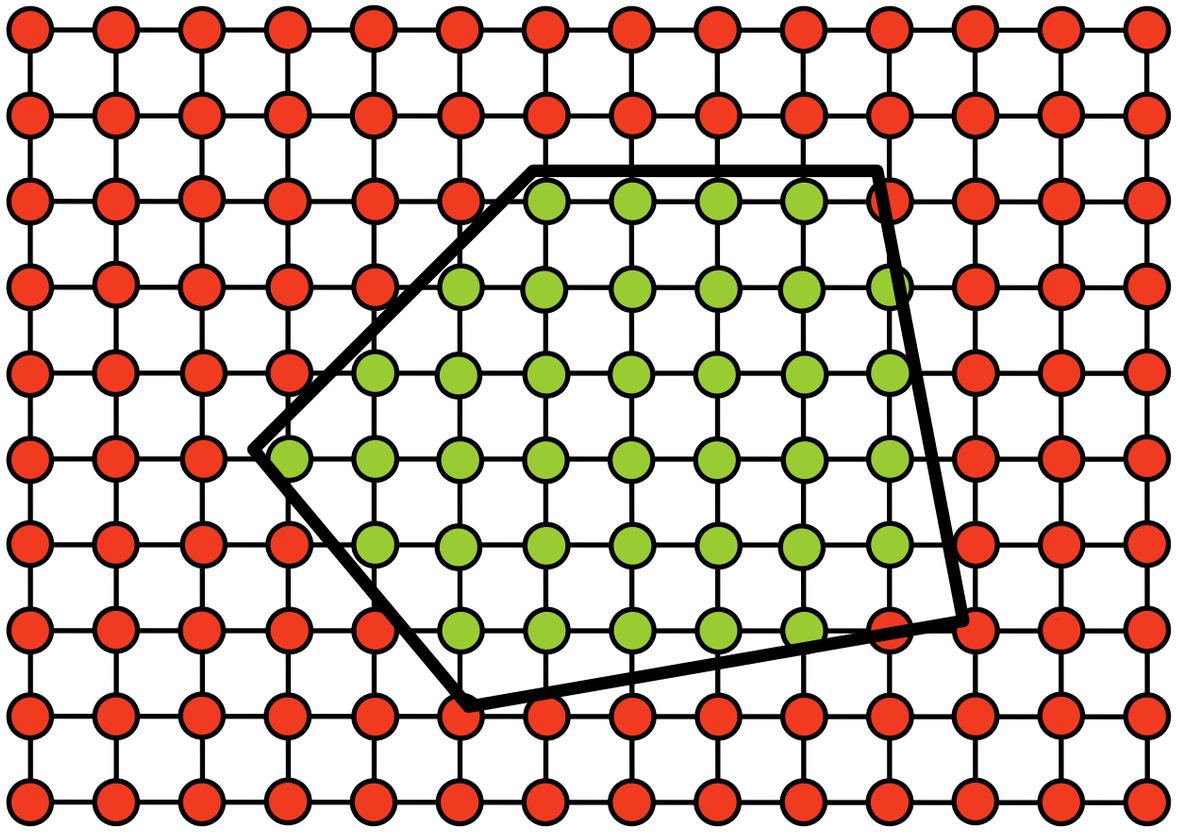
Sample the SDF



Sample the SDF



Sample the SDF

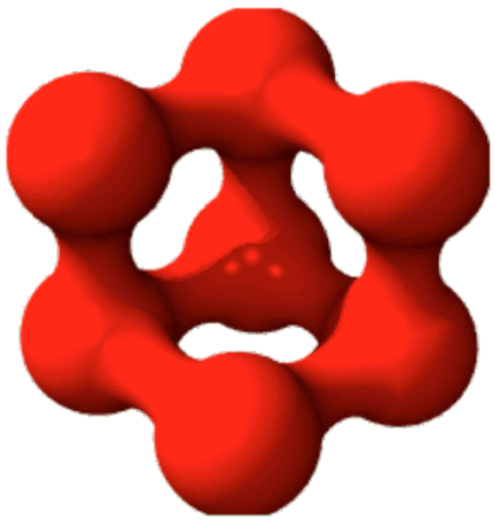


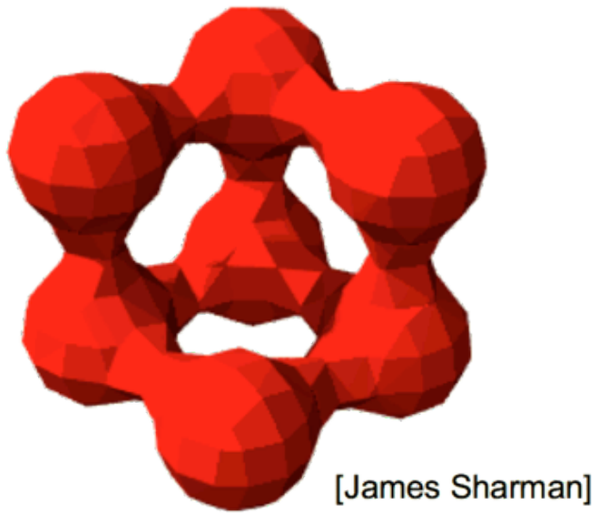
Marching Cubes

Converting from implicit to explicit representations.

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

Create a triangle mesh that approximates the surface.





[James Sharman]

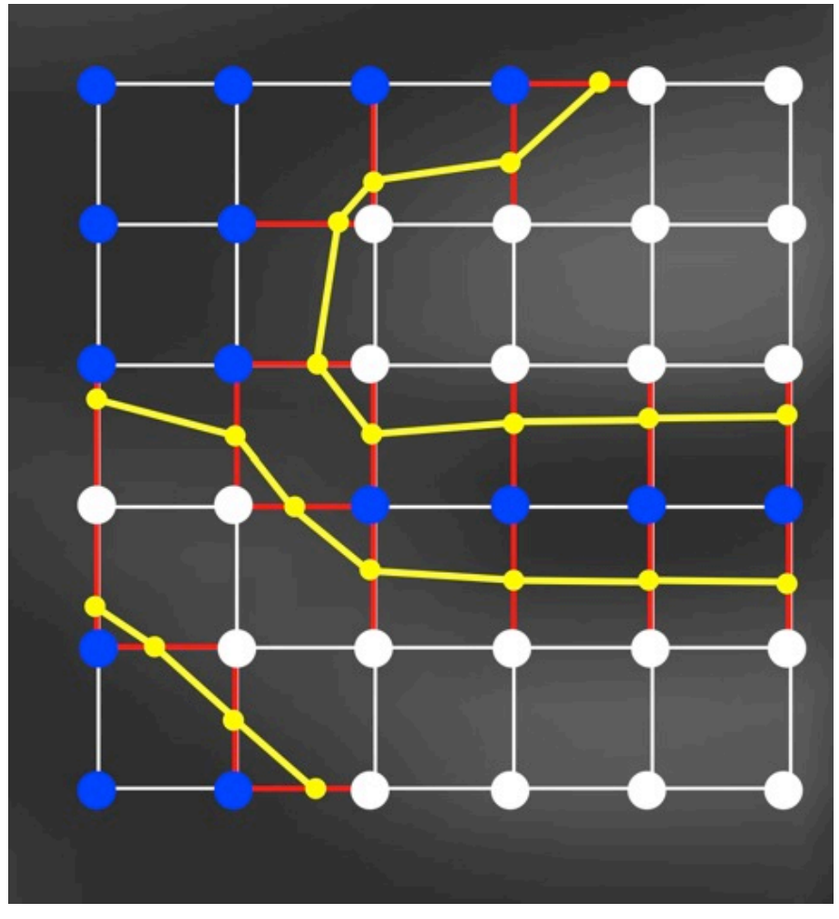
Lorensen and Cline, SIGGRAPH ‘87

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.

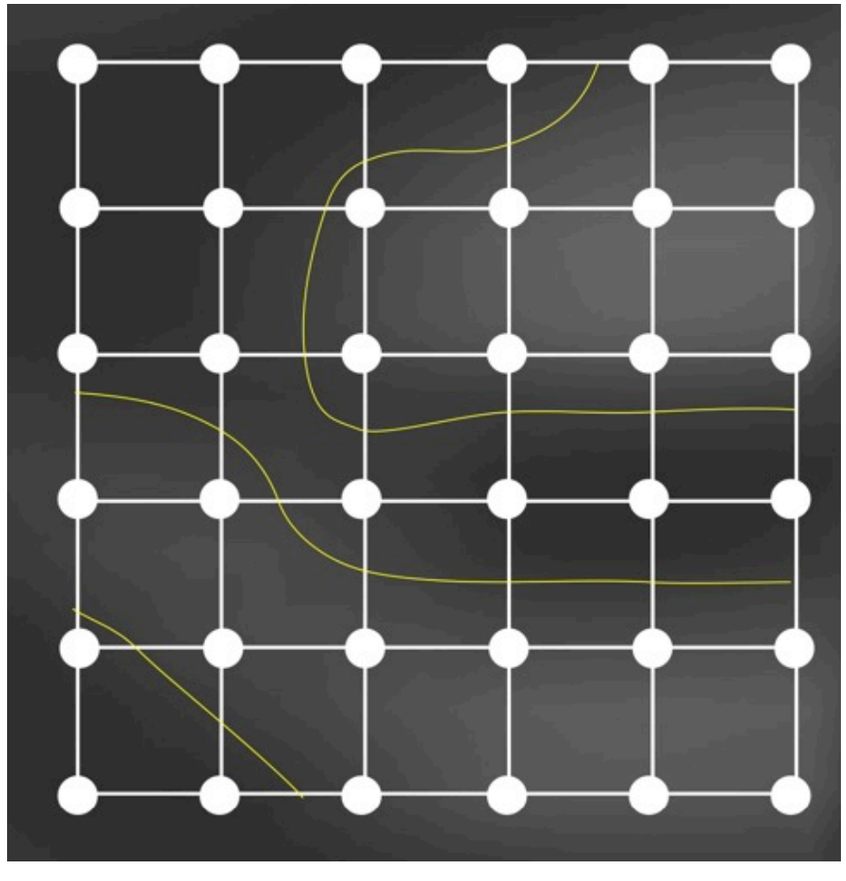


Marching Squares (2D)

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)

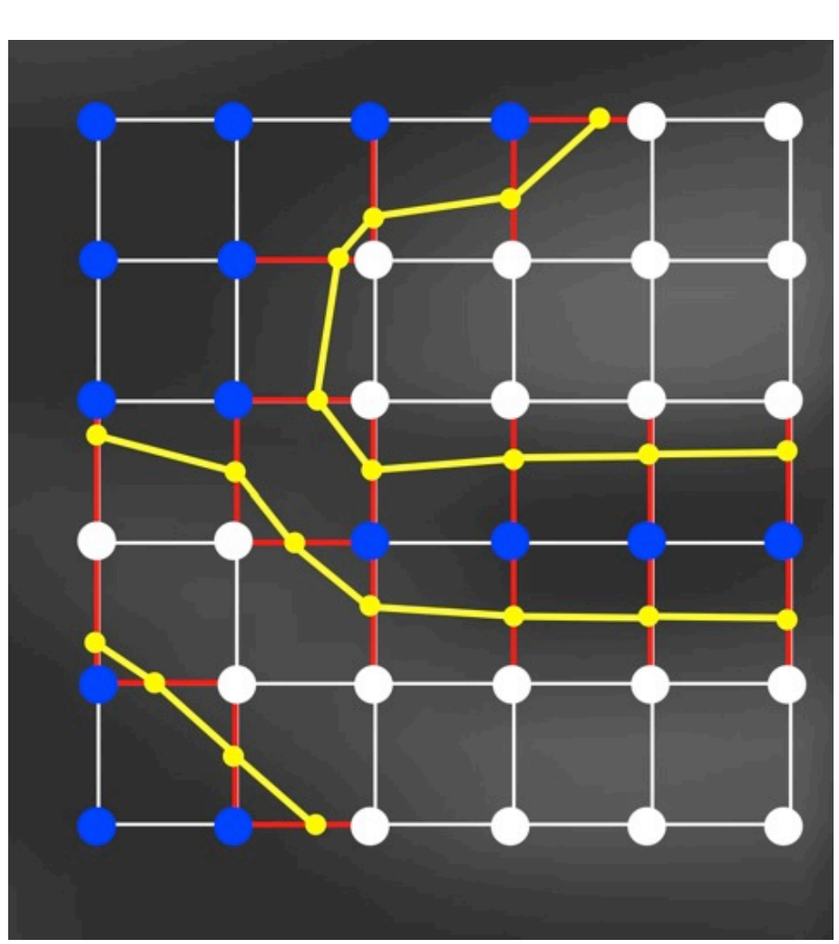
Computing the intersections:

- 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:

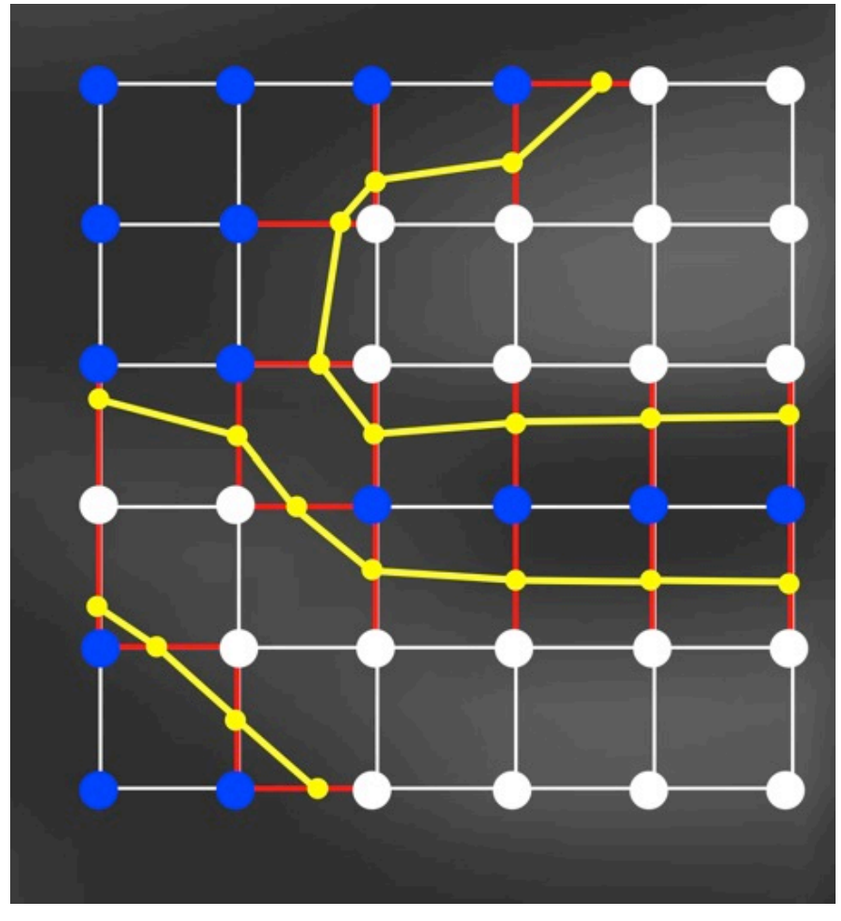
$$t = \frac{f(x_1)}{f(x_2) - f(x_1)}$$



Marching Squares (2D)

Connecting the intersections:

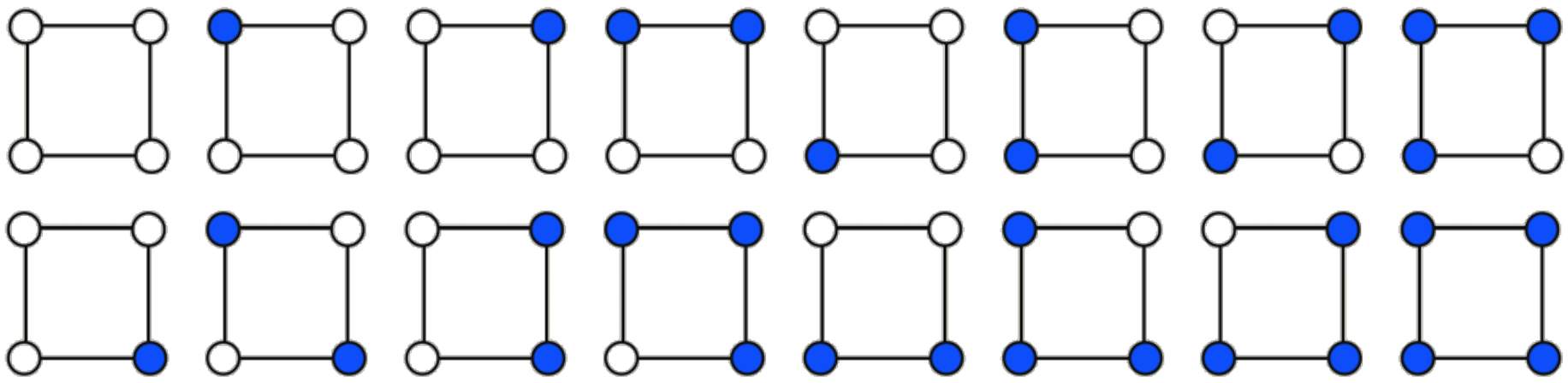
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.



Marching Squares (2D)

Connecting the intersections:

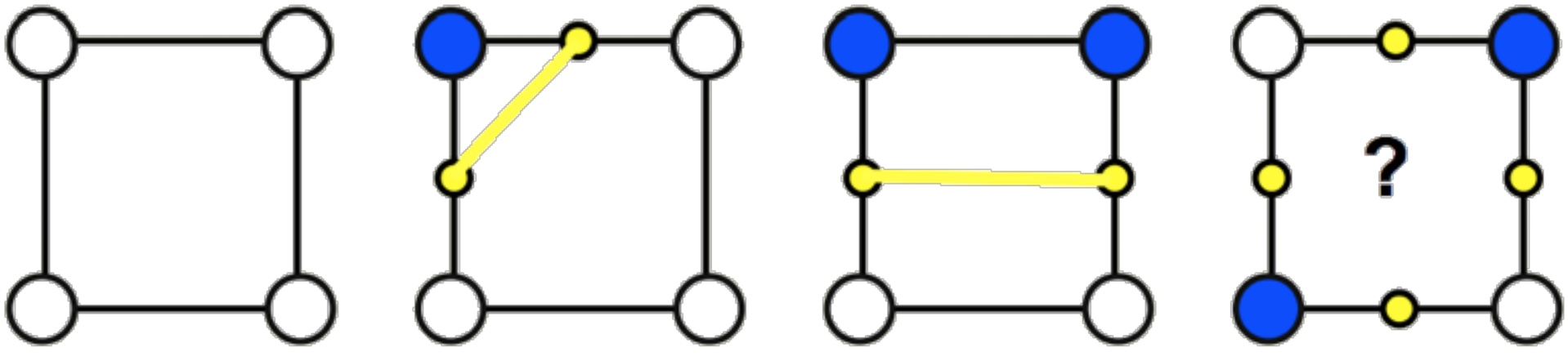
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.
- Group those leading to the same intersections



Marching Squares (2D)

Connecting the intersections:

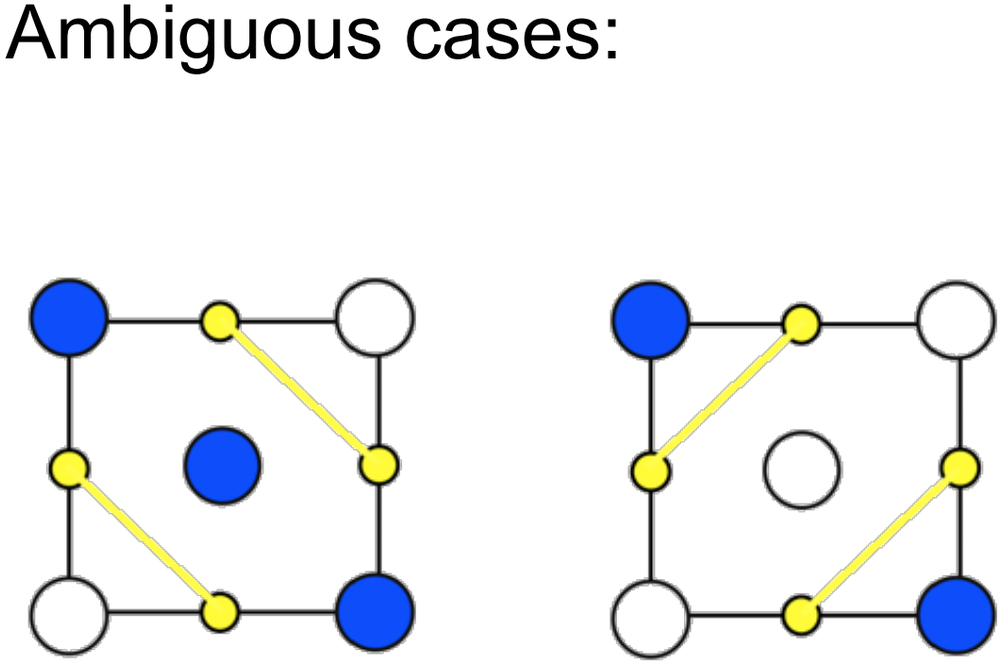
- Grand principle: treat each cell separately!
- Enumerate all possible inside/outside combinations.
- Group those leading to the same intersections.
- Group equivalent after rotation.
- Connect intersections

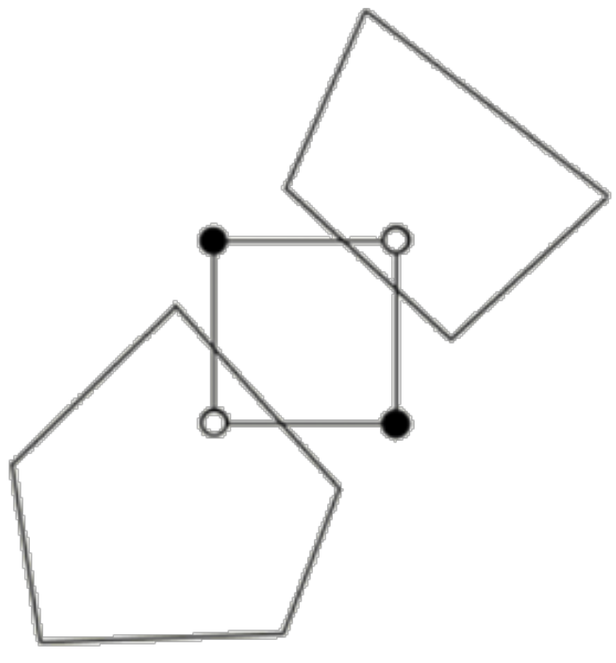


Marching Squares (2D)

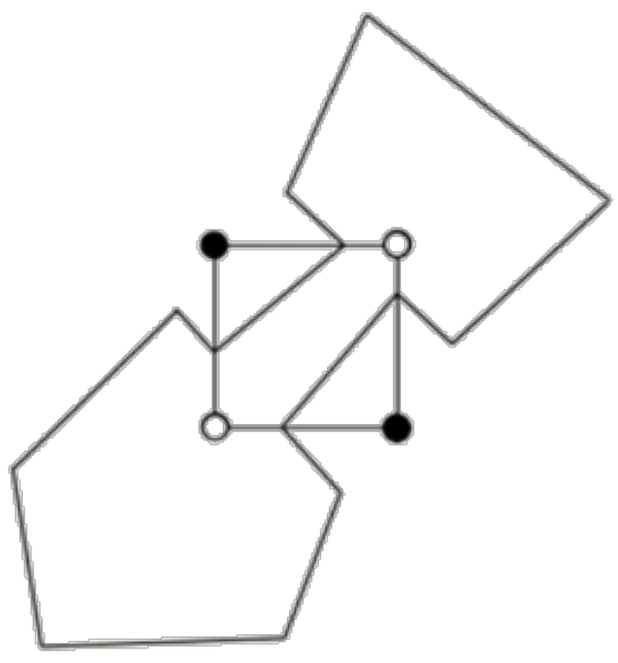
Connecting the intersections:

Ambiguous cases:





Break contour



Join contour

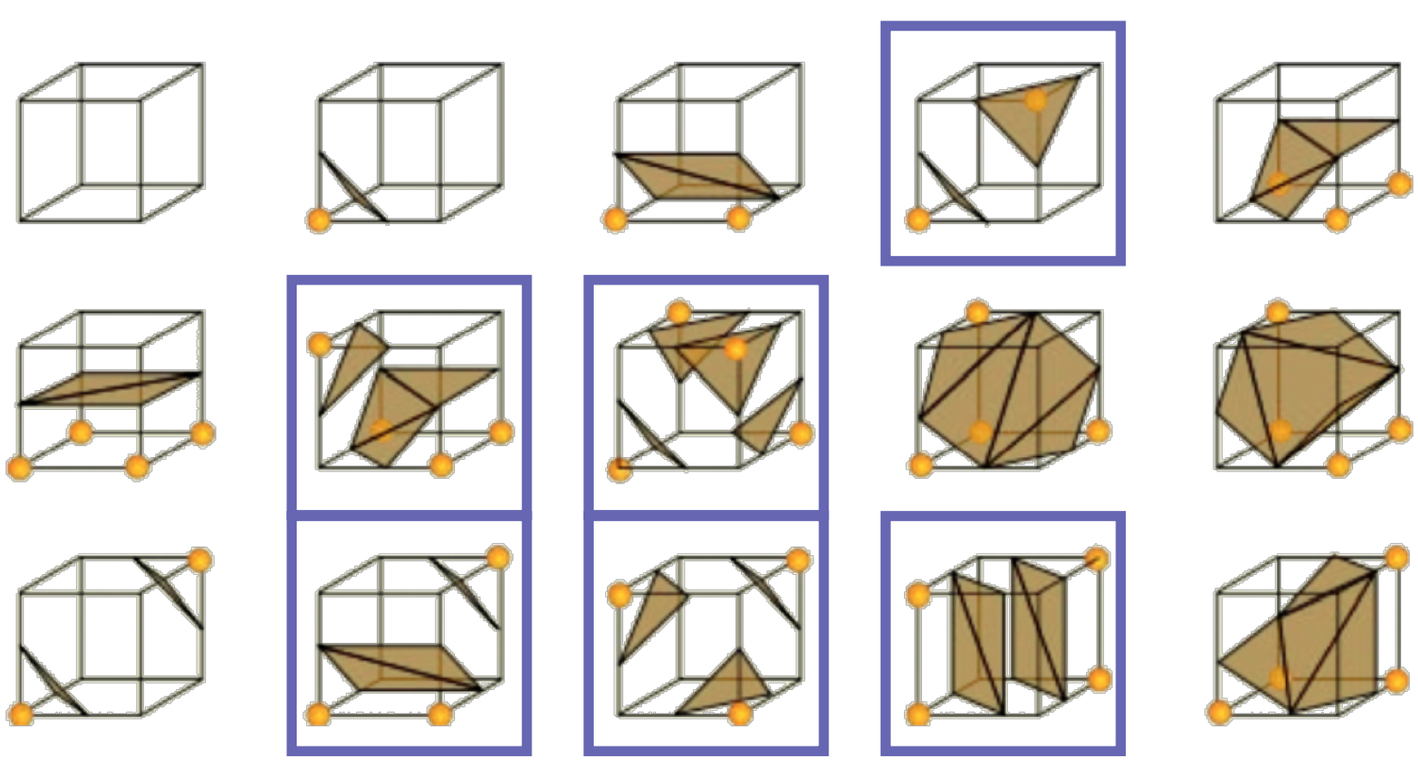
Two options:

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

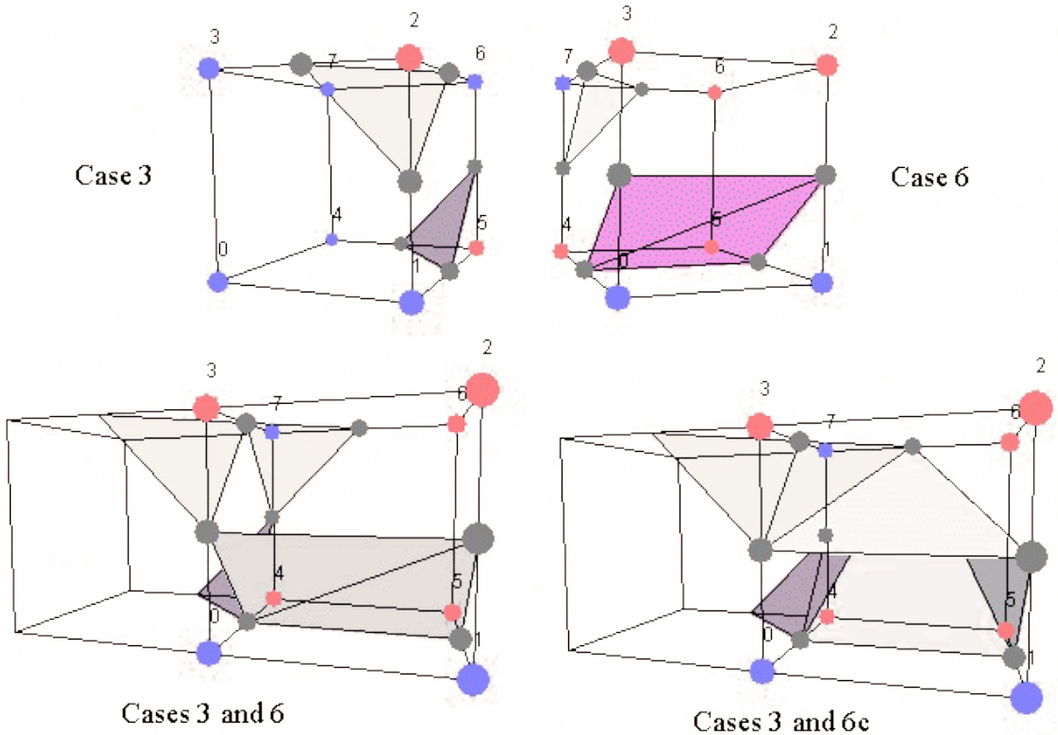
Marching Cubes (3D)

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



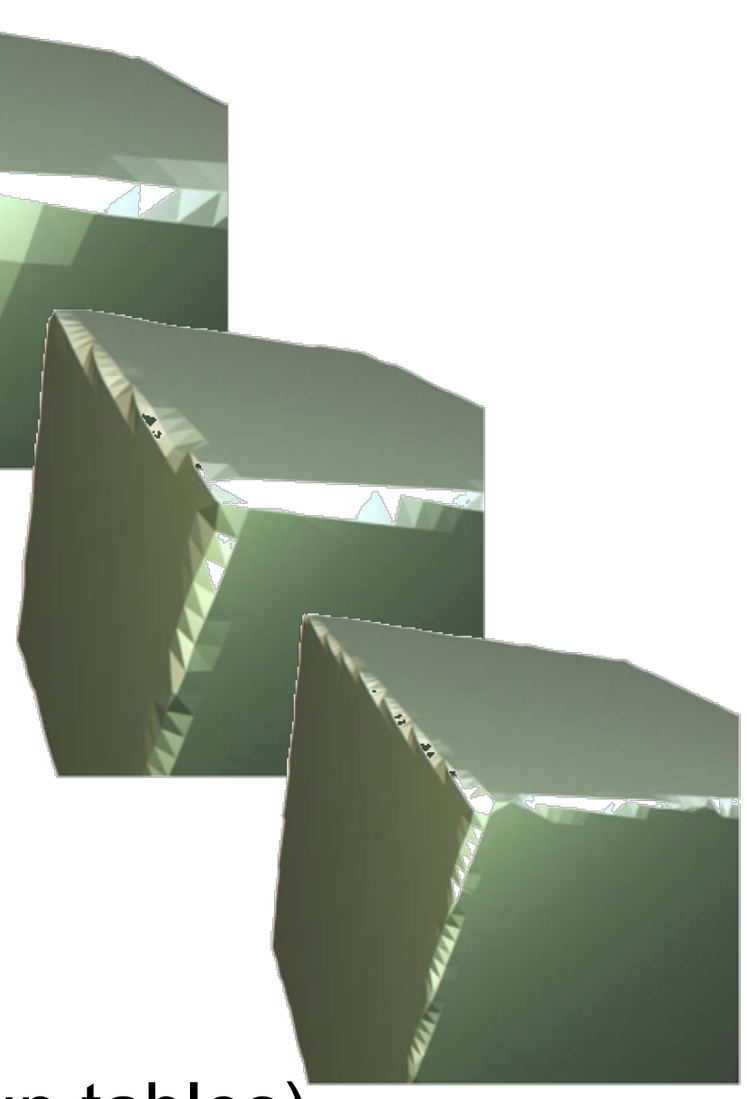
explore ambiguity to avoid holes!

Chernyaev, Marching Cubes 33,'95

Marching Cubes (3D)

Main Strengths:

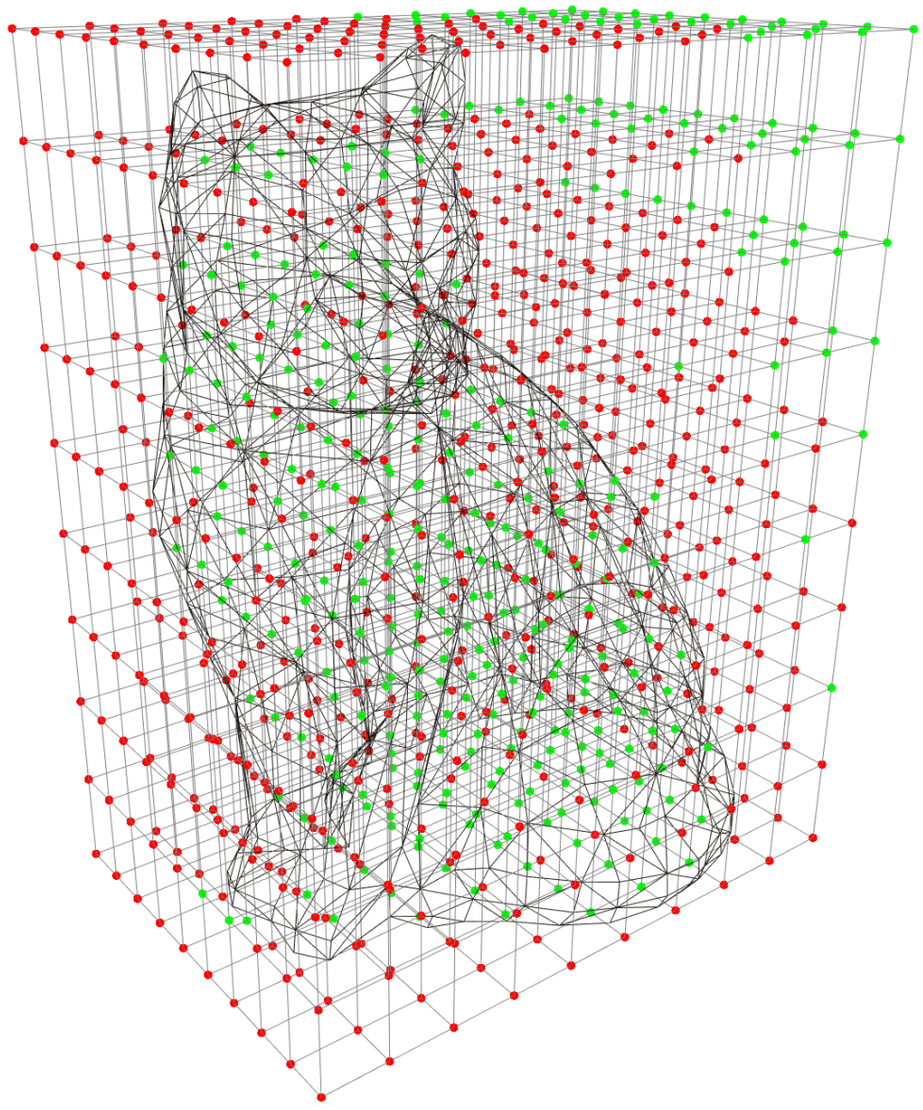
- Very multi-purpose.
- Extremely fast and parallelizable.
- Relatively simple to implement.
- Virtually parameter-free

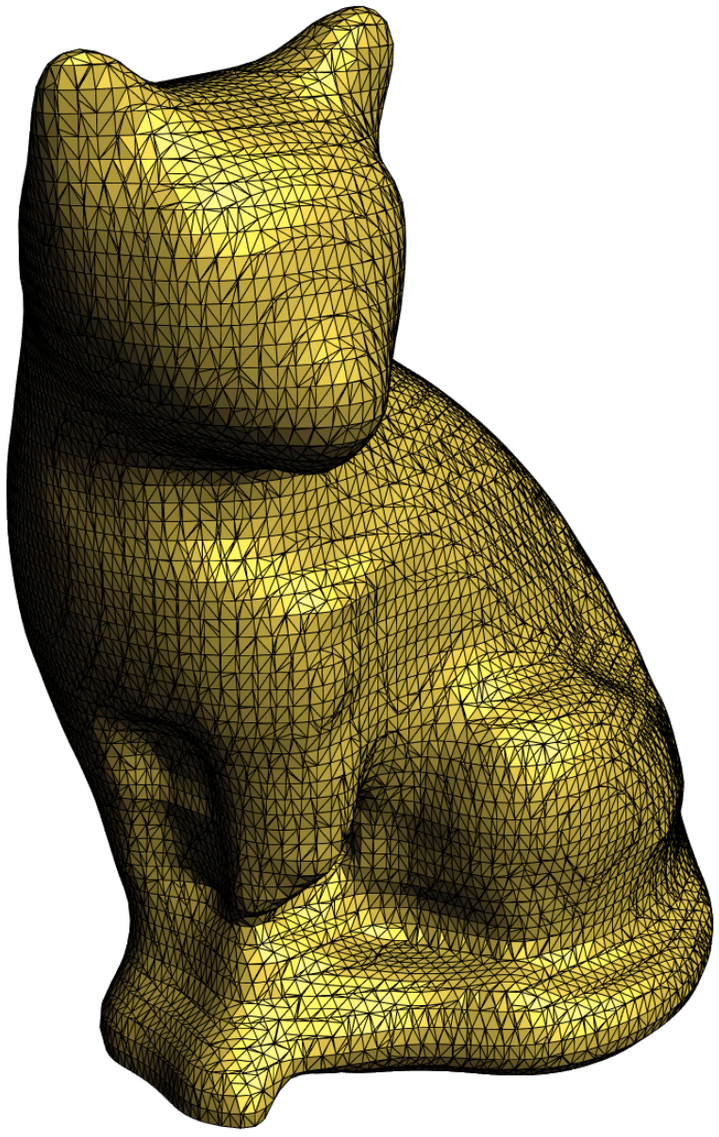


Main Weaknesses:

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

Recap: Points→Implicit→Mesh





Next Time: Mesh → Point Cloud!

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

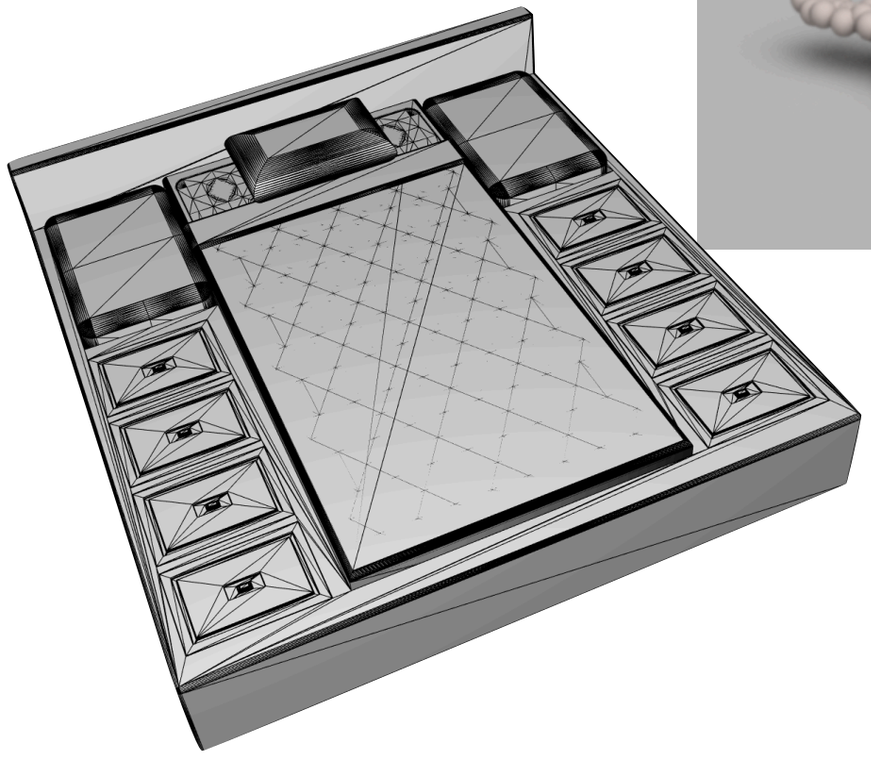
MESH-> POINT CLOUD

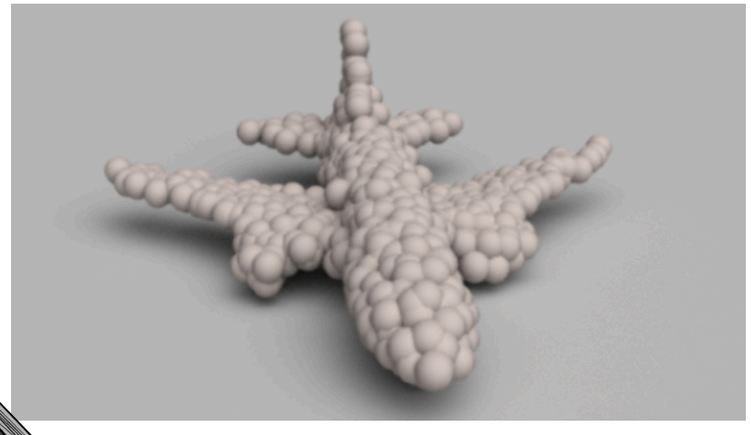
Sampling

UC San Diego

From Surface to Point Cloud - Why?

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

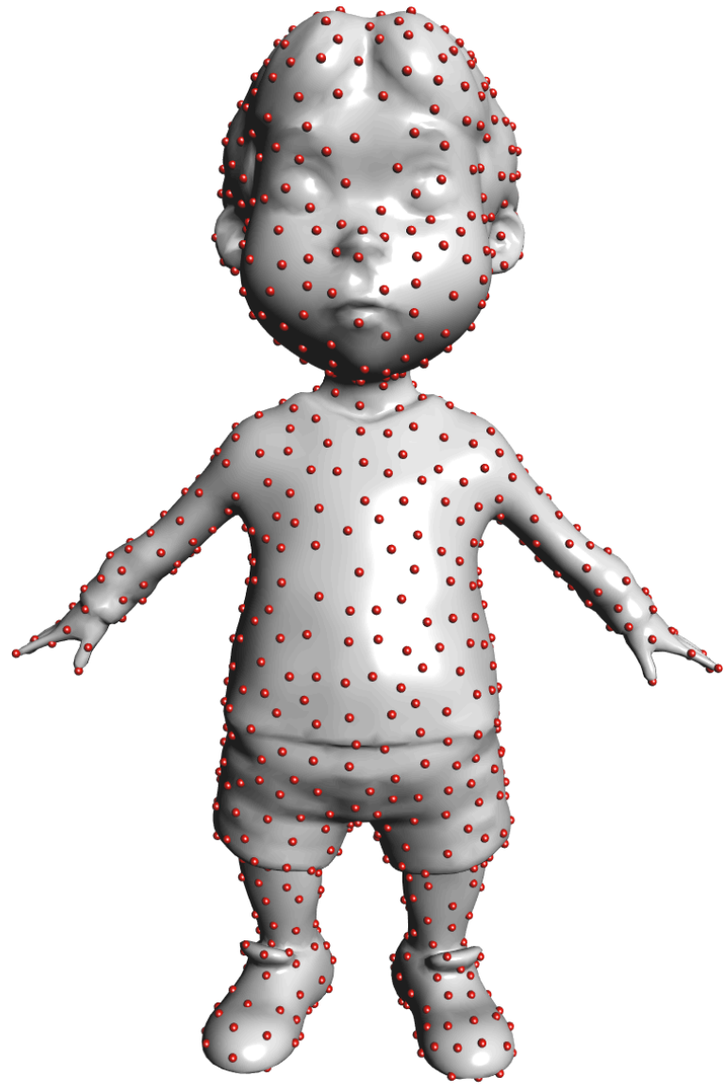




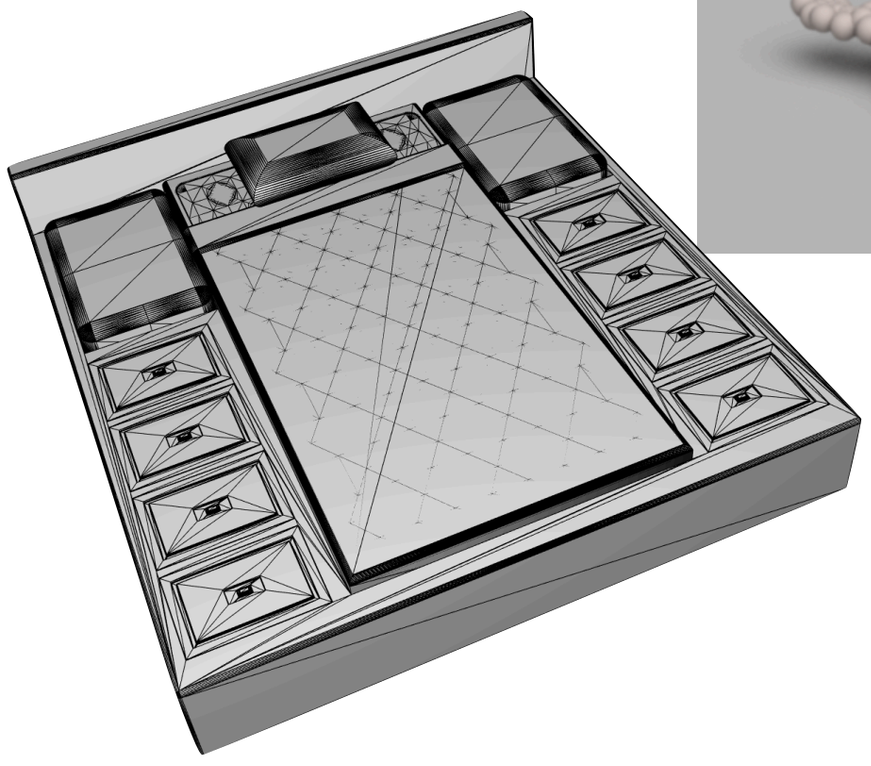
CAD meshes:
many components
bad triangles
connectivity problems

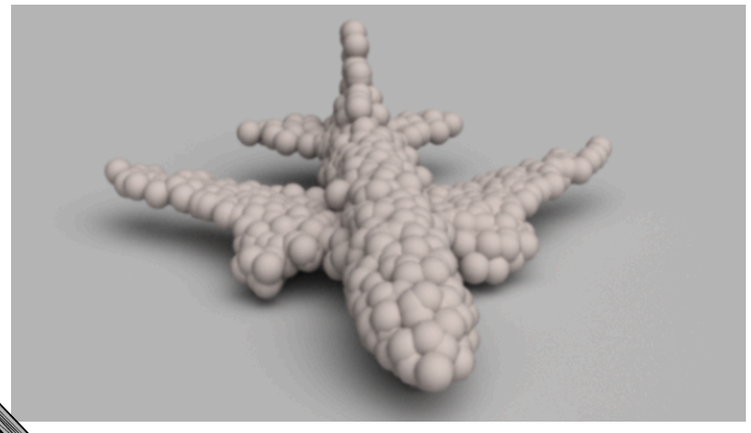
From Surface to Point Cloud - Why?

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



the problem:
sampling the mesh

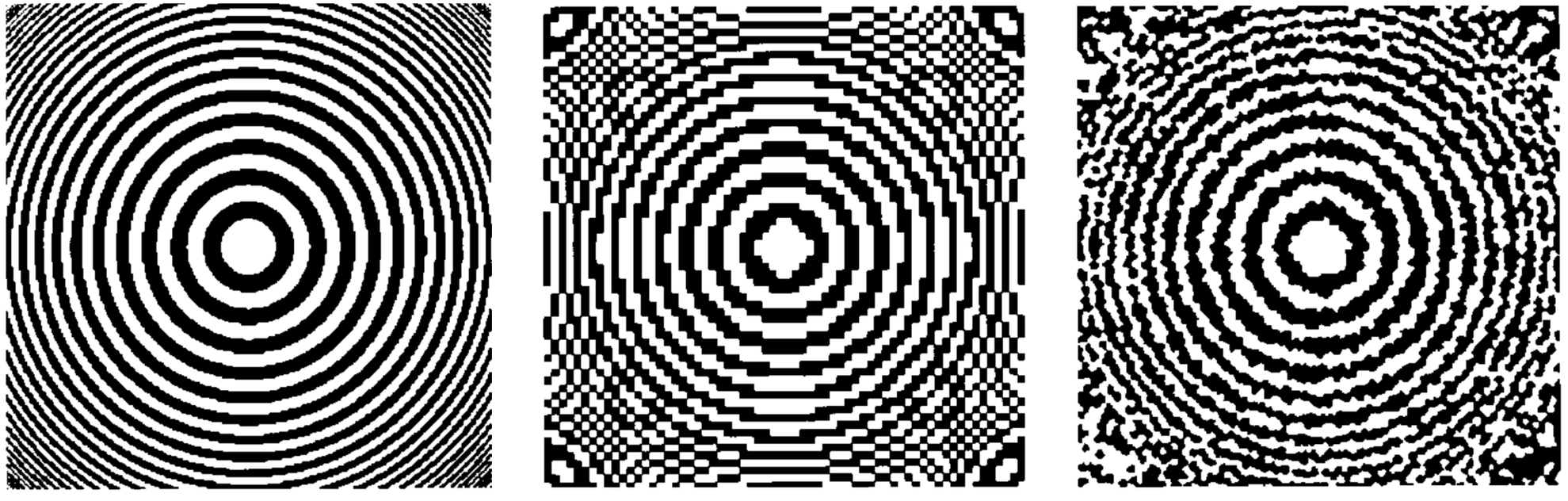




CAD meshes:
many components
bad triangles
connectivity problems

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*

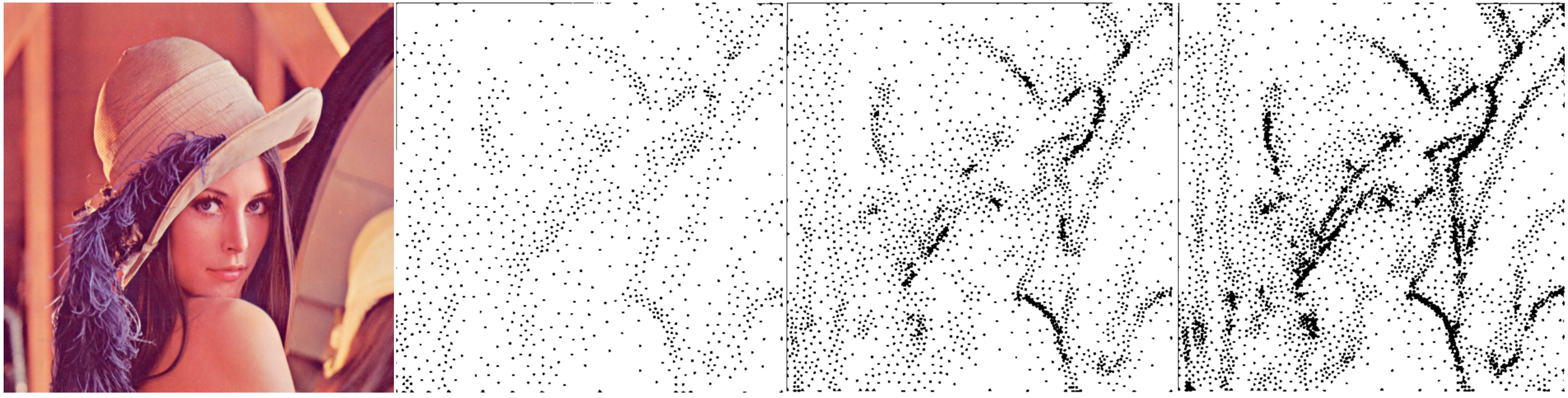
$$d(p, S) = \max_{q \in A}(d(q, S))$$

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

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

Farthest Point Sampling

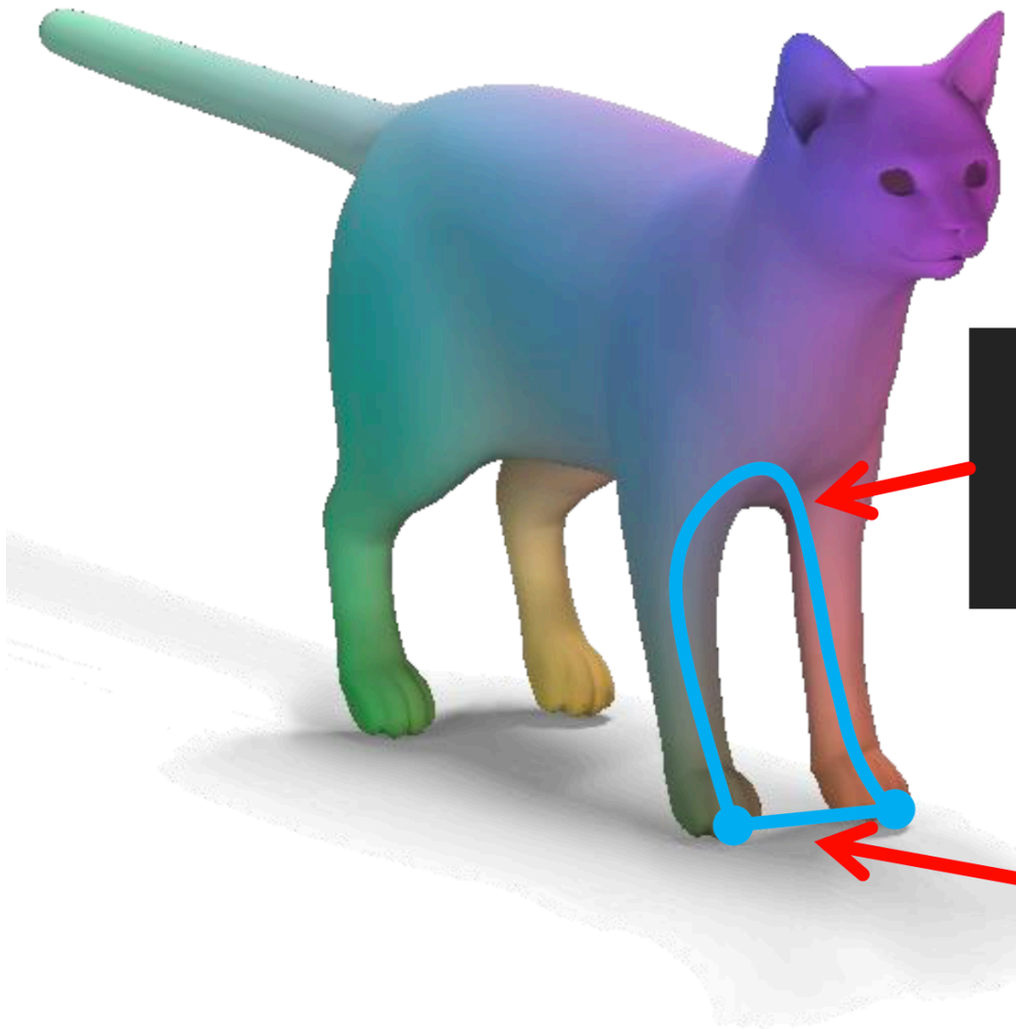
- Depends on a notion of distance on the sampling domain
- Can be made adaptive, via a weighted distance

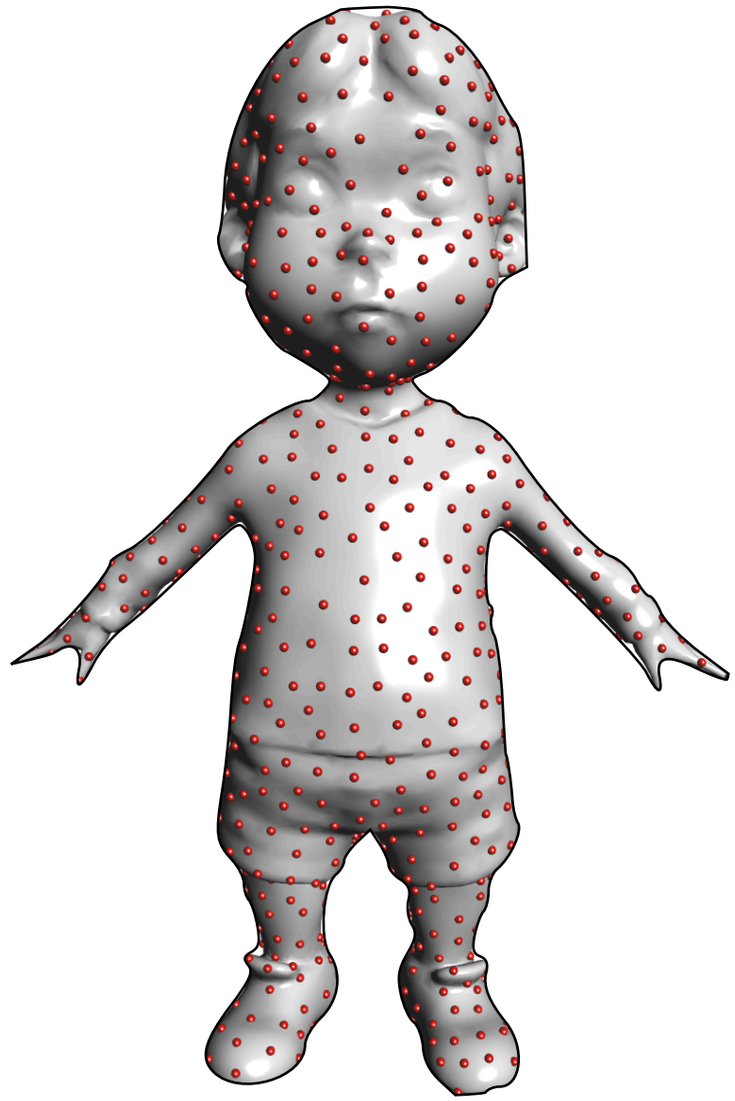


Eldar et al. 1997, “The Farthest Point Strategy for Progressive Image Sampling”

FPS on surfaces

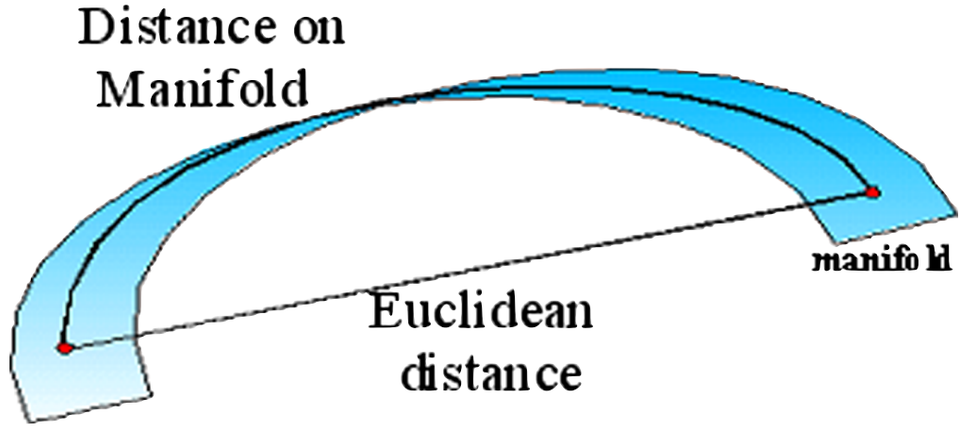
- What's an appropriate distance?

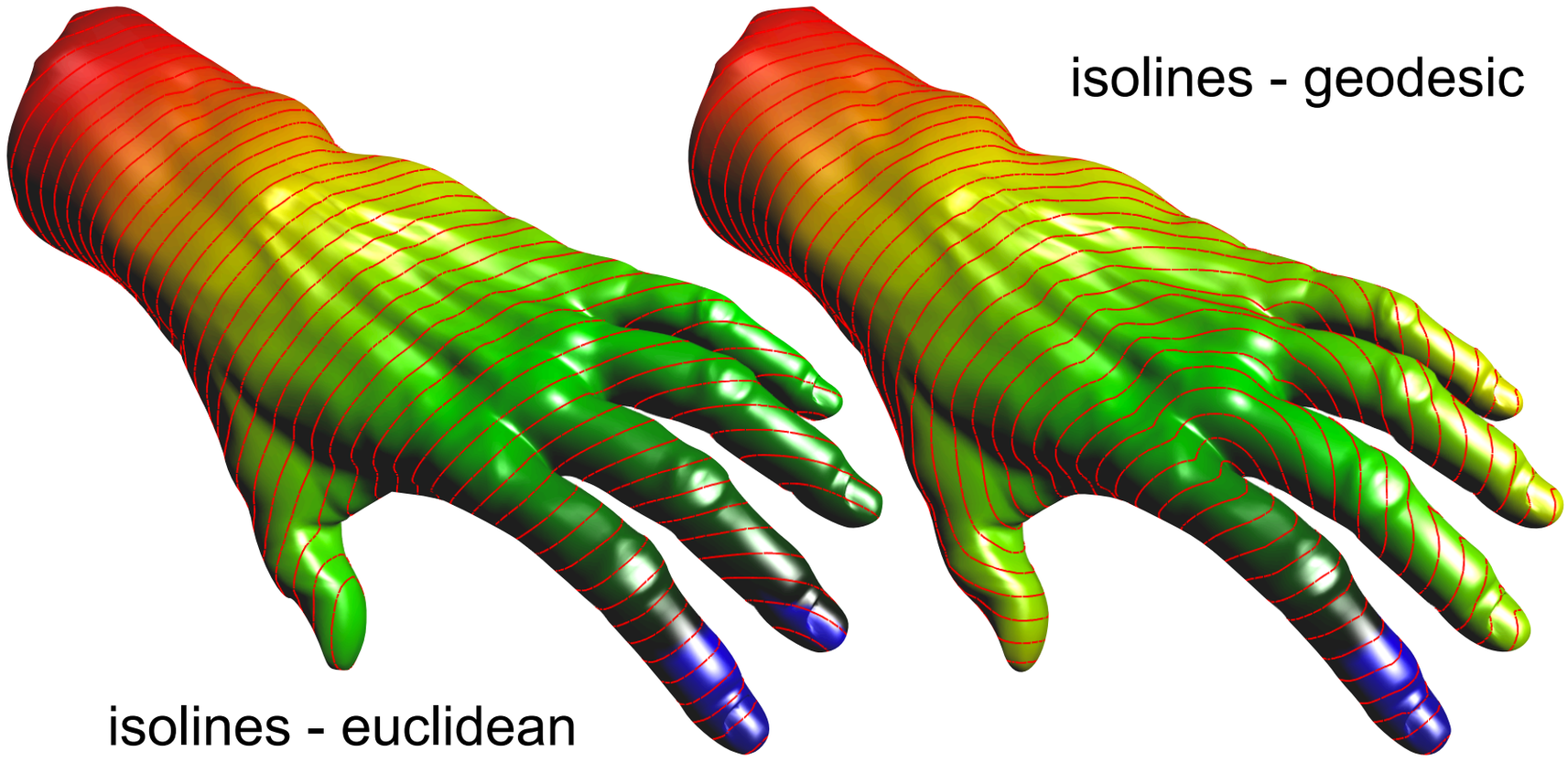




On-Surface Distances

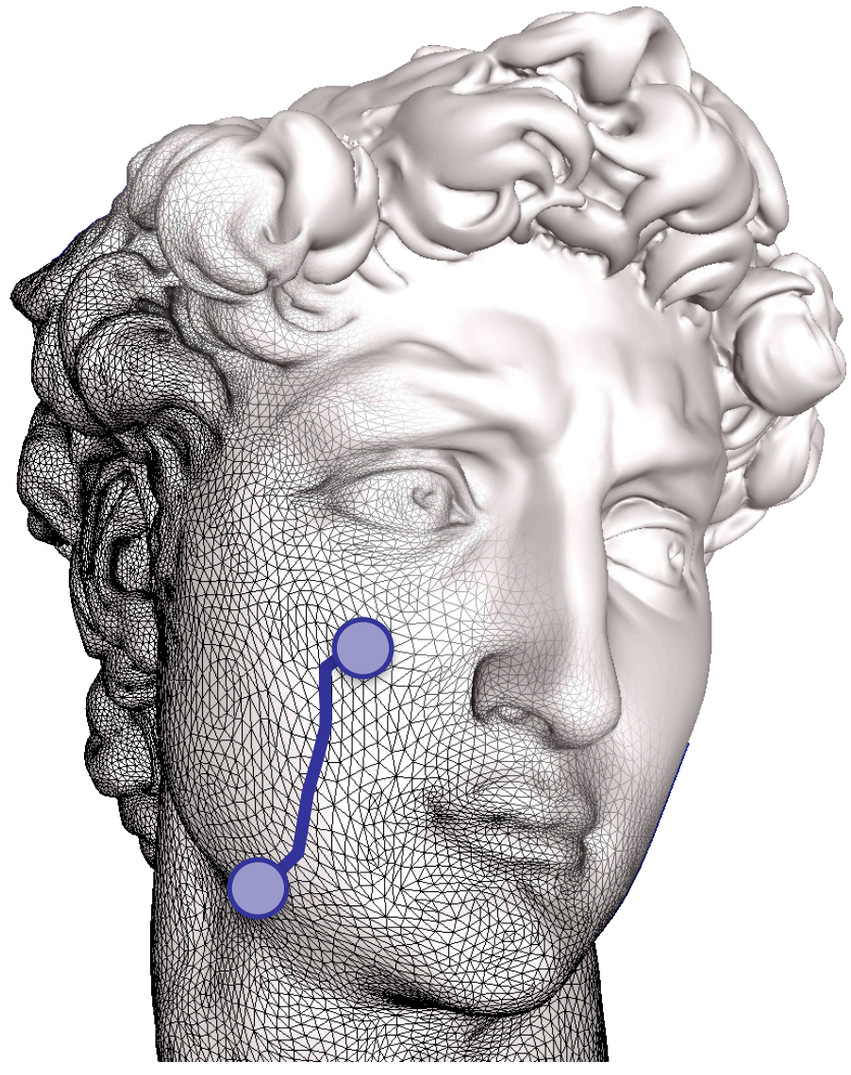
- Geodesics: Straightest and **locally shortest** curves





Discrete Geodesics

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

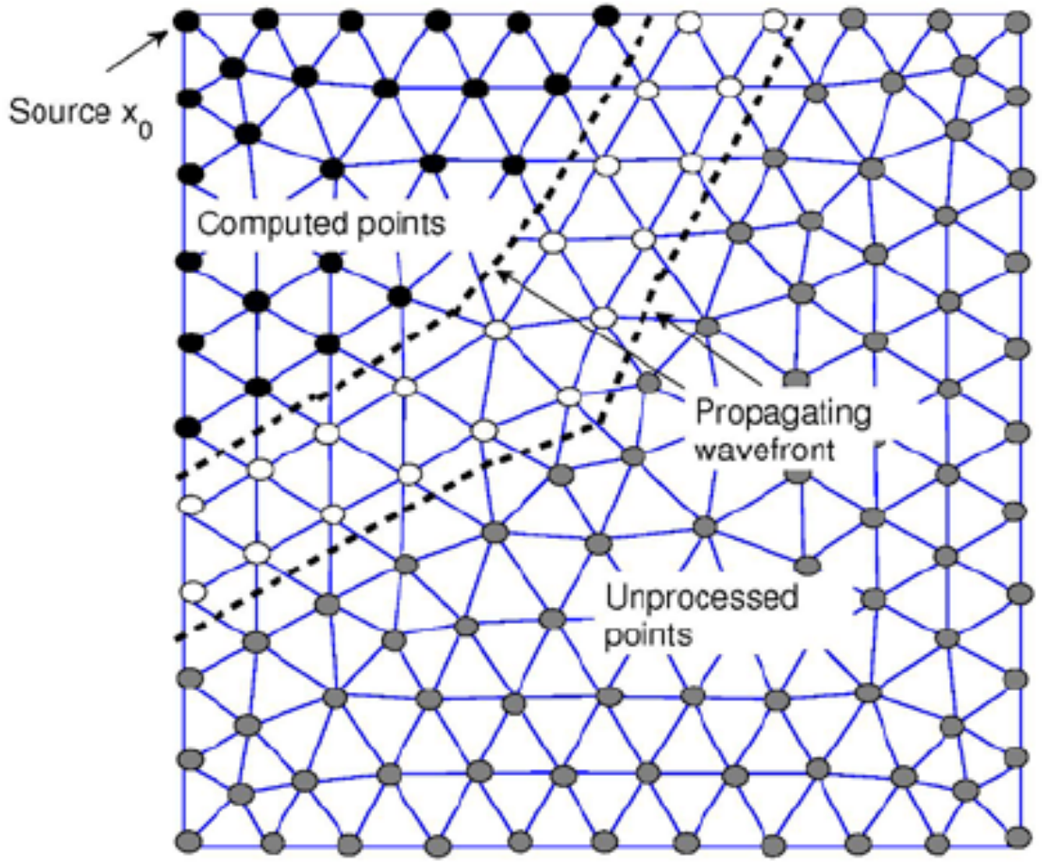


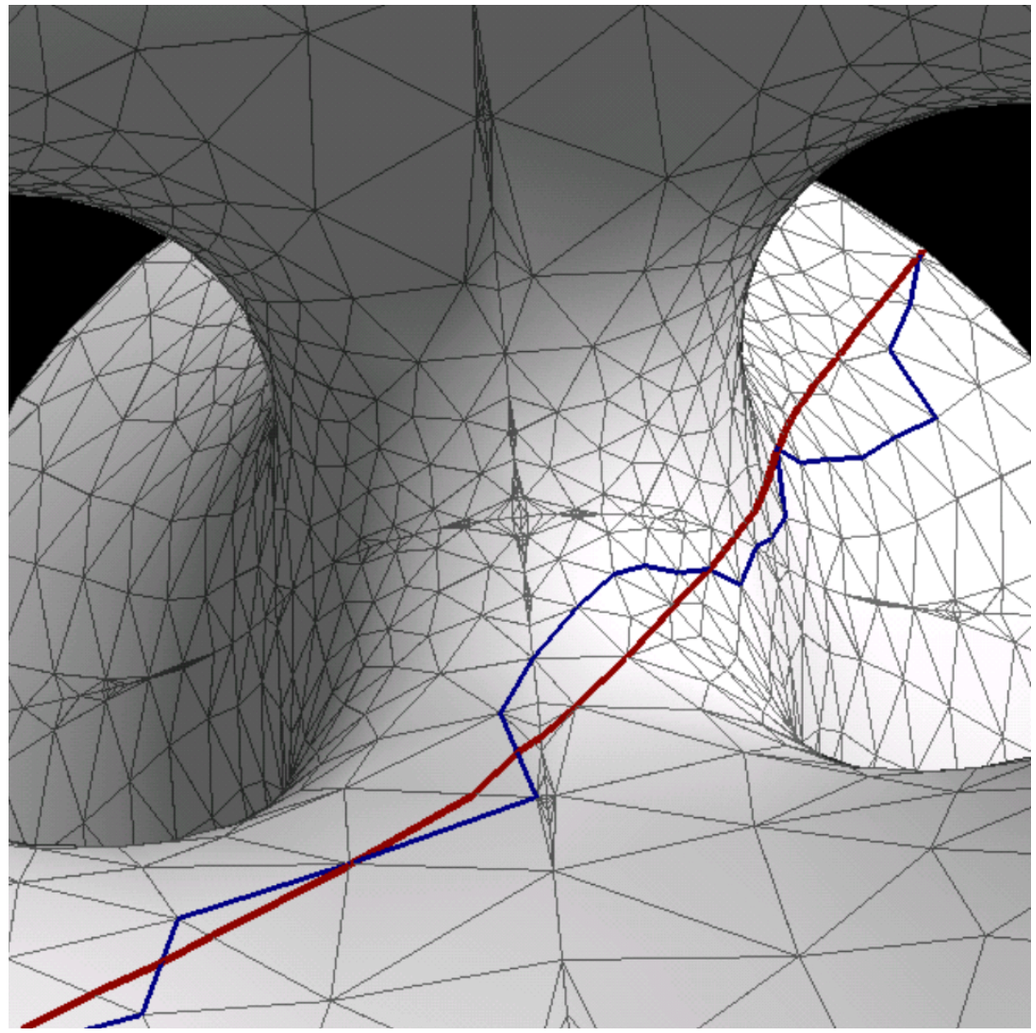
```
v_0= initial vertex
d_i = current distance to vertex i
S = verticies with known optimal distance

# 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

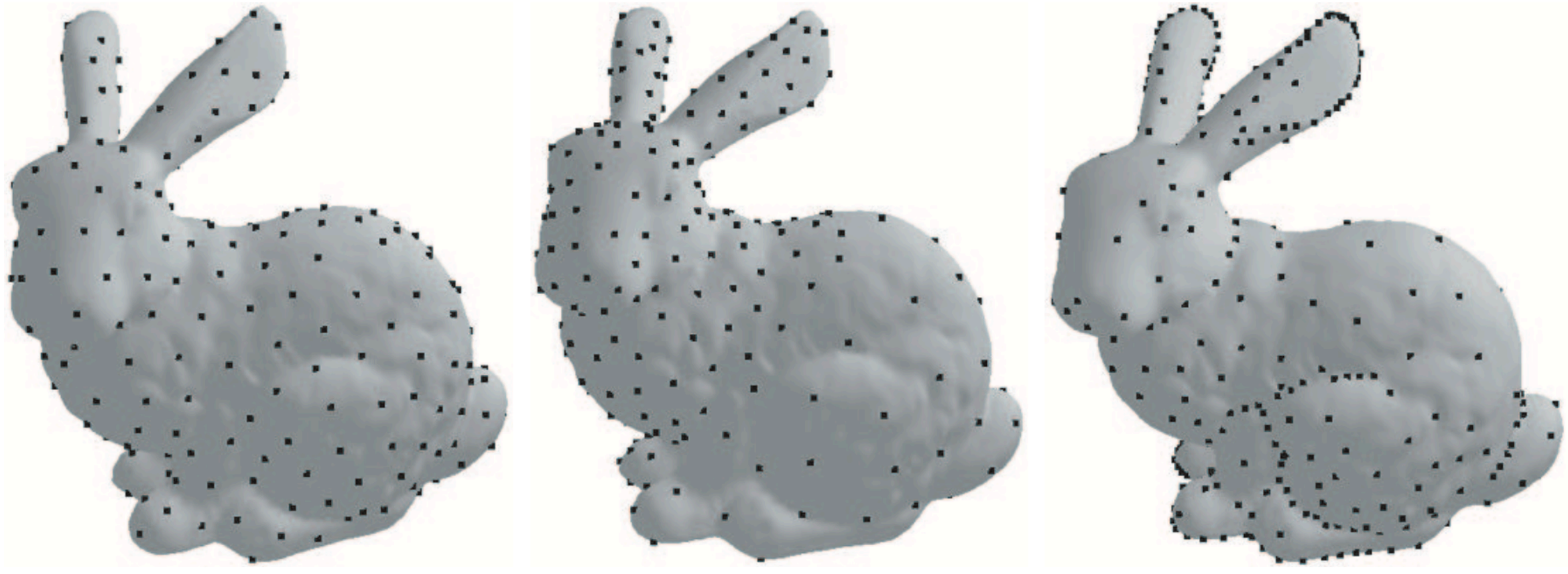
- A better approximation: allow fronts to cross triangles!





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

FPS on a Mesh



Peyré and Cohen 2003, Geodesic Remeshing Using Front Propagation

Topology of Surfaces

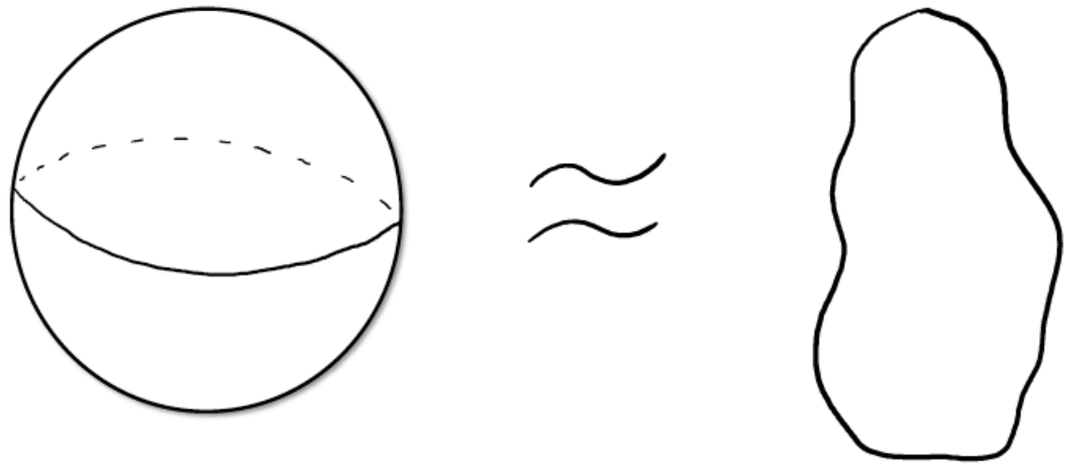
(slides mostly by Glenn Eguchi)

Topology of Surfaces

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

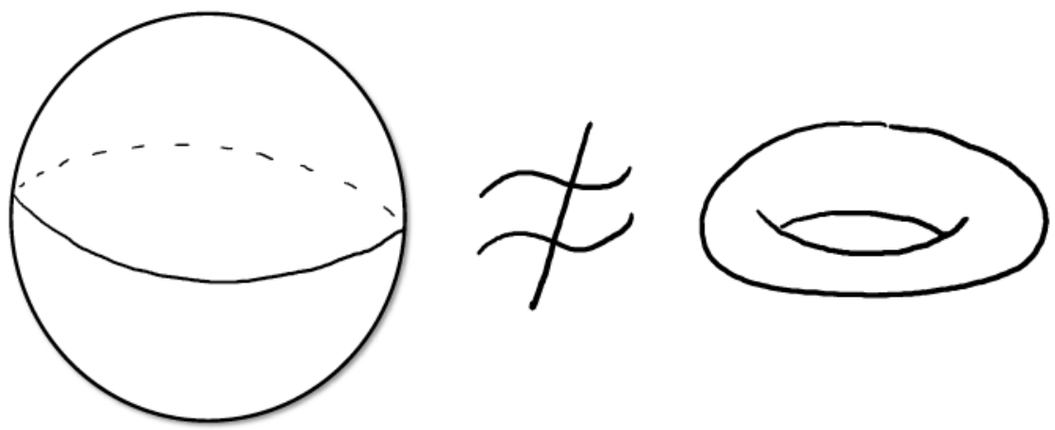
M and N are diffeomorphic

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



Sphere ≈ Surface of Potato

torus = surface of doughnut ≈ coffee cup



Sphere ≠ torus

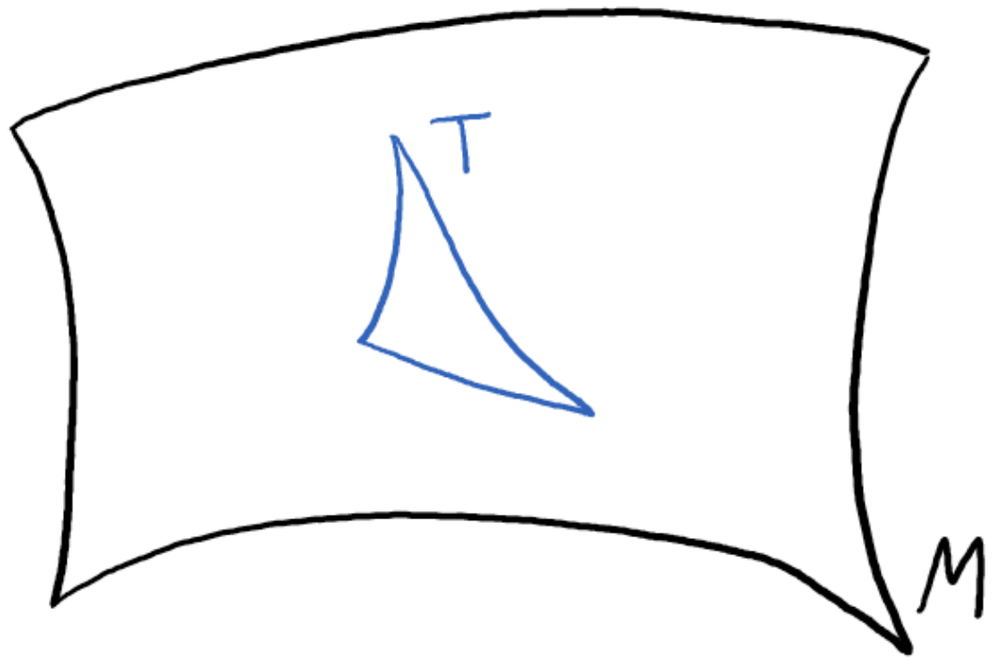
Topology by a Single Number

- 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

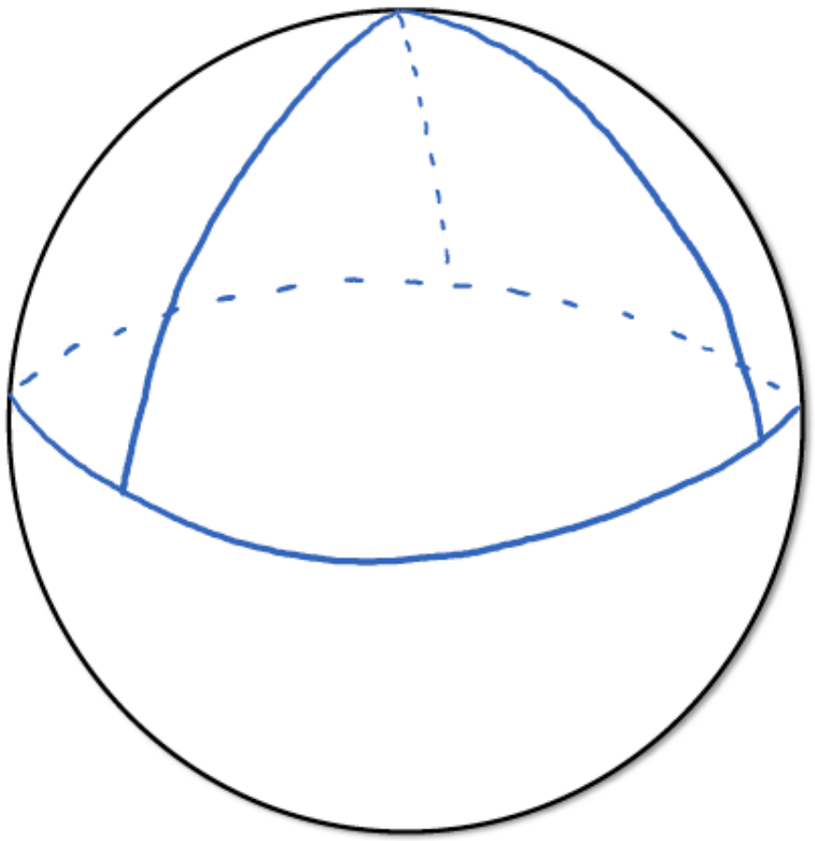
- A *triangulation* of M is a decomposition of M into a finite number of triangles $T_1, T_2, \ldots, T_n$ such that

(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

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

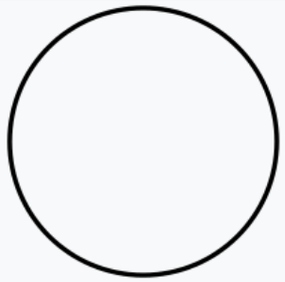
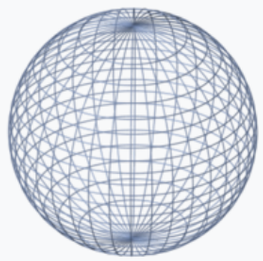
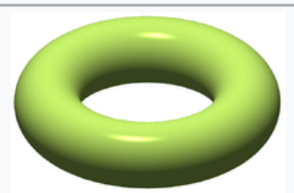
where

$$V = \text{number of vertices}$$
$$E = \text{number of edges}$$
$$F = \text{number of faces}$$

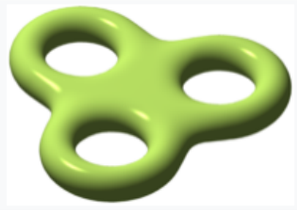
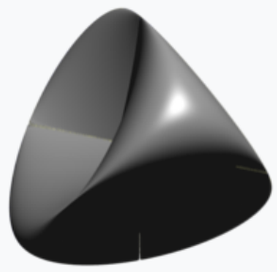
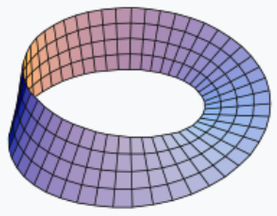
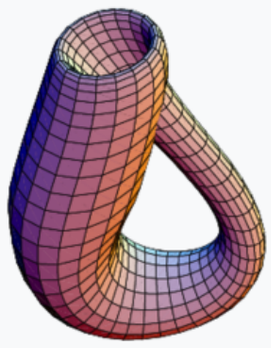
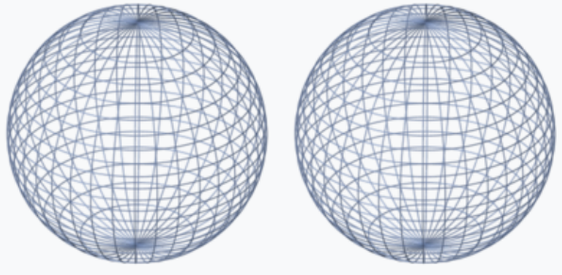
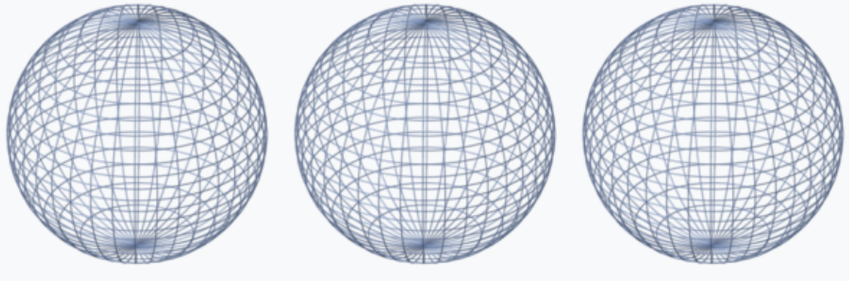
The following fact (we will not prove) justifies the definition of $\chi(M)$.

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

Examples

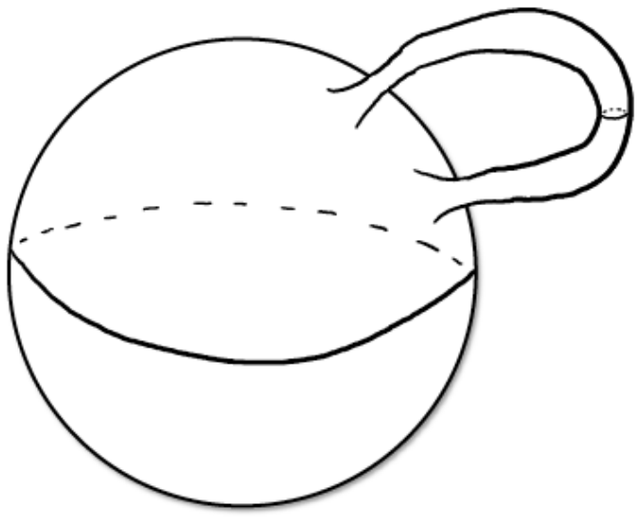
Name	Image	Euler characteristic
Interval		1
Circle		0
Disk		1
Sphere		2
Torus (Product of two circles)		0
Double torus		–2

Examples

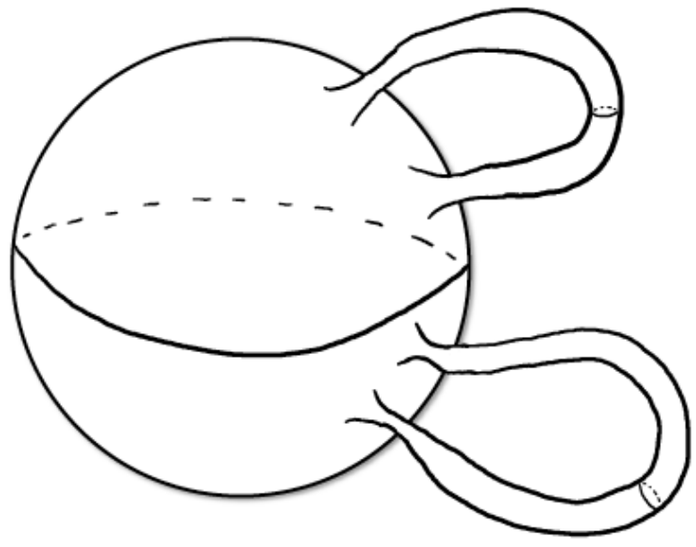
Triple torus		**–4**
Real projective plane		**1**
Möbius strip		**0**
Klein bottle		**0**
Two spheres (not connected) (Disjoint union of two spheres)		2 + 2 = **4**
Three spheres (not connected) (Disjoint union of three spheres)		2 + 2 + 2 = **6**

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.