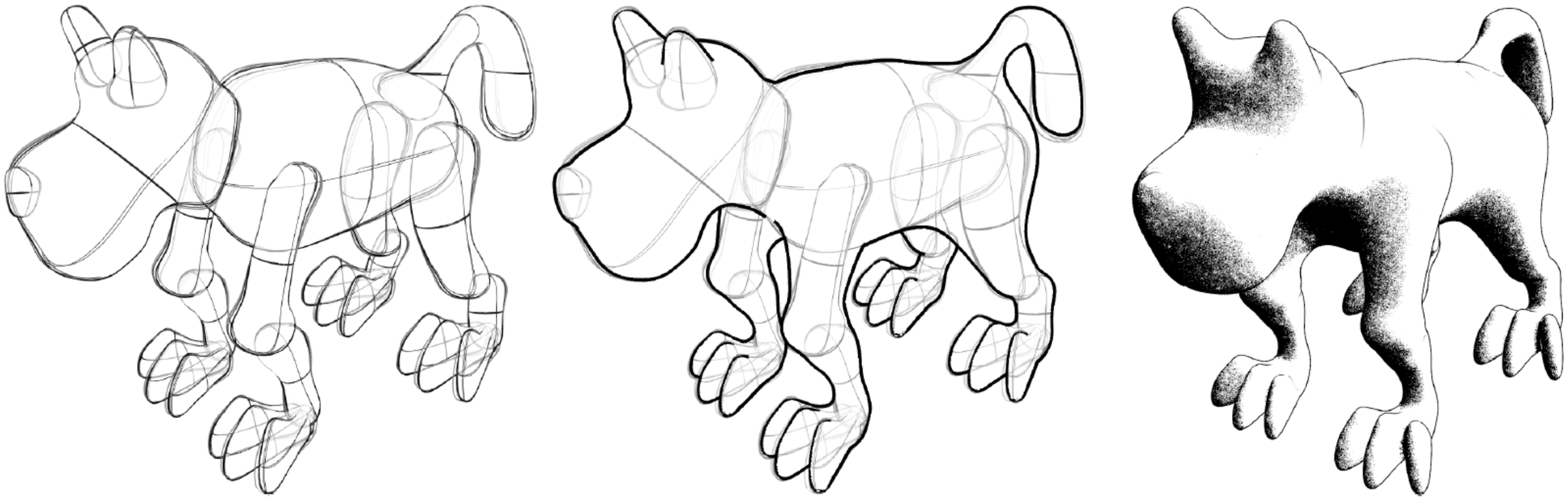


Implicit Modelling and Reconstruction

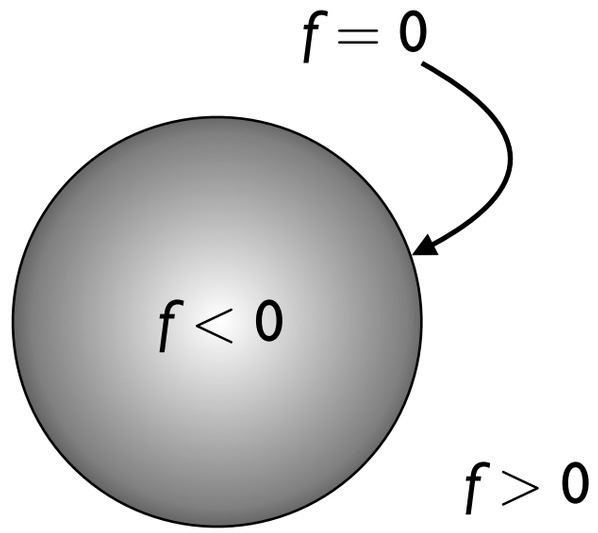
Introduction to Computer Graphics
CSE 533/333

Ojaswa Sharma

Implicit Modelling

- Different methods for defining implicit surfaces:
  - Skeletal implicit modelling
  - Offset surfaces
  - Level sets
  - Variational surfaces
  - Algebraic surfaces
- Consider equation for circle:

$$f(x, y) = x^2 + y^2 - r^2 = 0$$



Implicit Functions

- An implicit function is defined as a function *f* applied to a point $\mathbf{p} \in \mathbb{R}^3$ yielding a scalar value $\in \mathbb{R}$
- An implicit function $f_i(x, y, z)$ may be split into
  - A distance function $d_i(x, y, z)$
  - A *fall-off filter function* $g_i(r)$, r is distance from skeleton

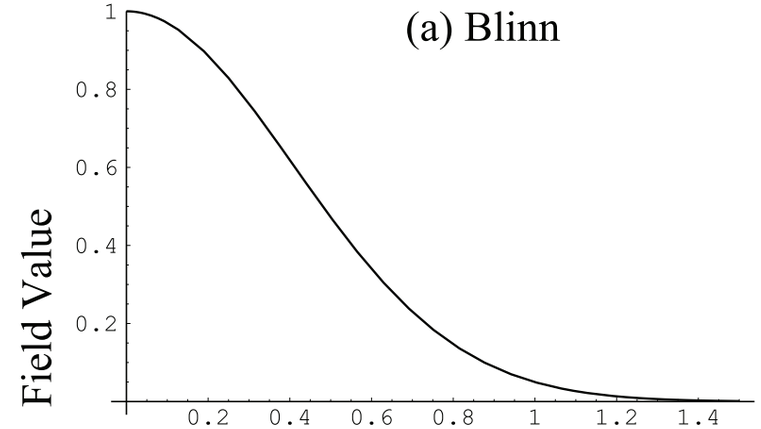
$$f_i(x, y, z) = g_i \circ d_i(x, y, z)$$

Fall-off Filter Function

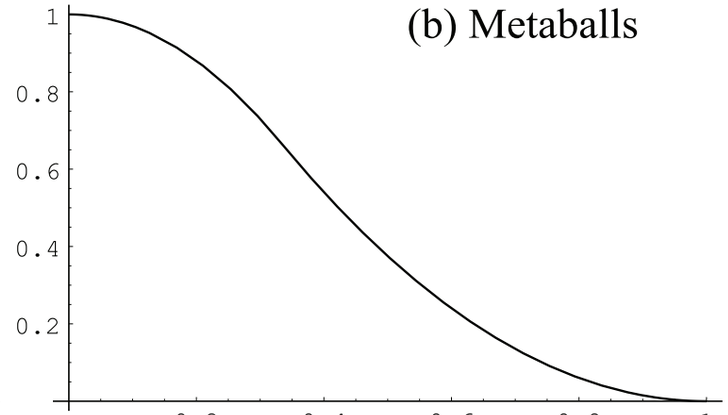
- Field values are maximum at the skeleton
- These fall off to zero at a certain distance from the skeleton (finite support)
- In case of simple blending, global field is sum of individual fields

$$f(x, y, z) = \sum_{i=1}^{n} f_i(x, y, z)$$

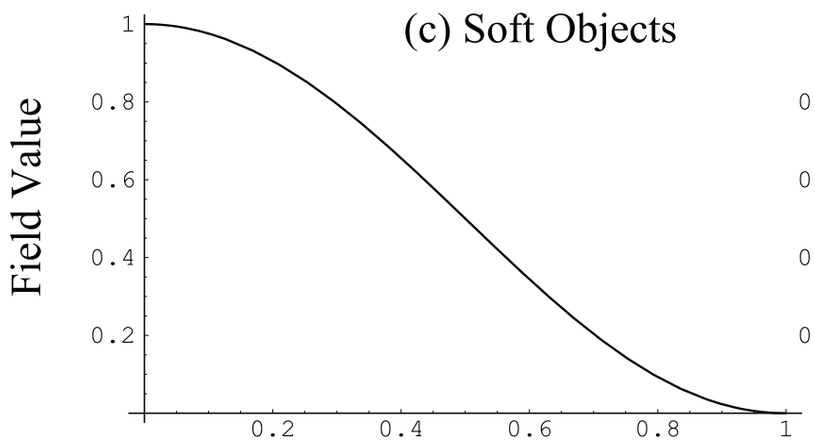
Fall-off Filter Function



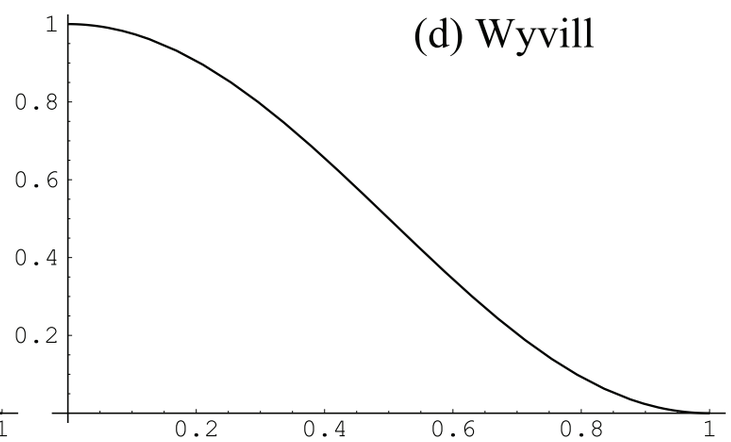
$$g(d) = e^{-rd^2}$$



$$g(d) = \begin{cases} 1 - 3\left(\frac{d}{r}\right)^2 & 0 \le d \le \frac{r}{3} \\ \frac{3}{2}(1 - \frac{d}{r})^2 & \frac{r}{3} \le d \le r \\ 0 & d > r \end{cases}$$

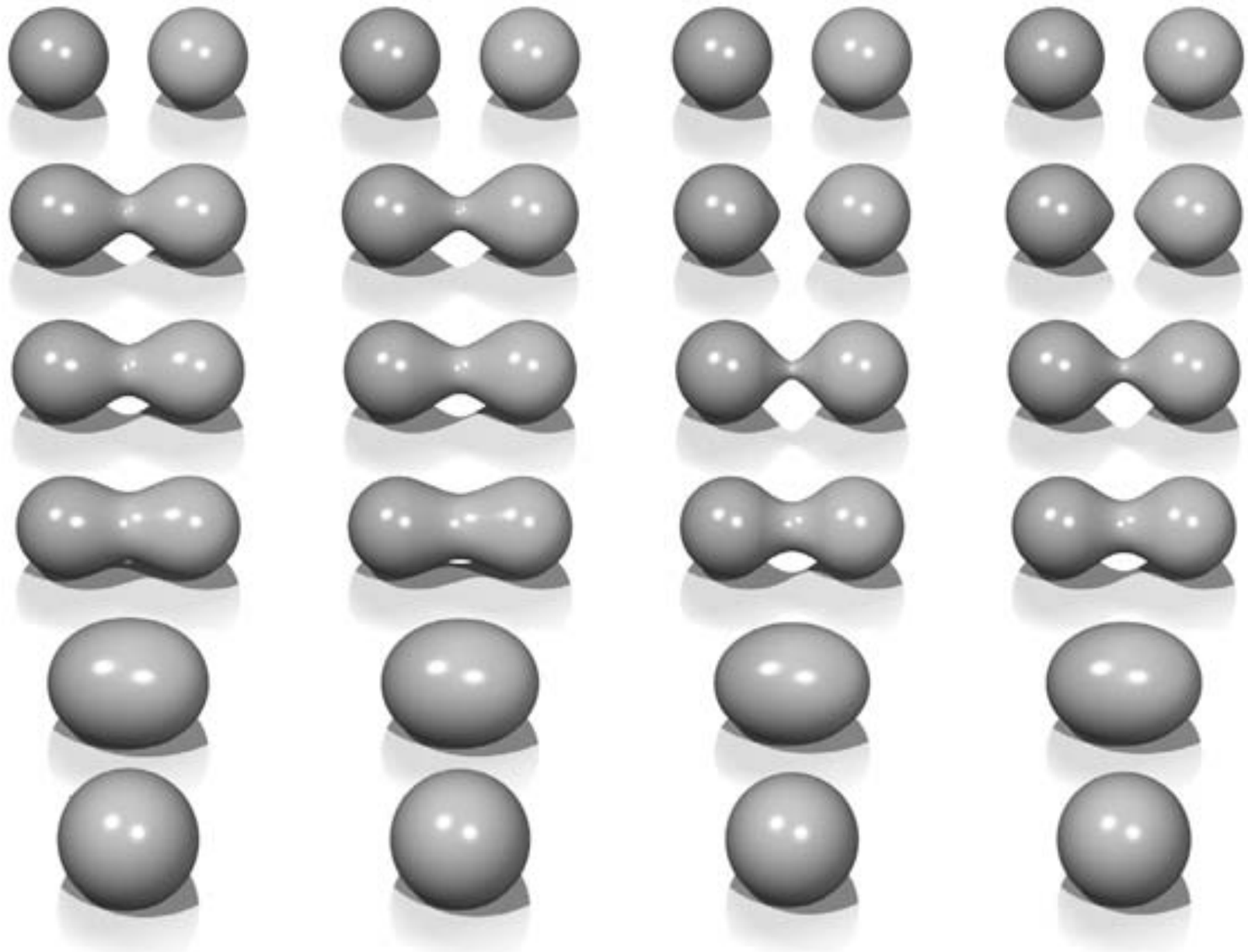


$$g(d) = \left(1 - \frac{4d^6}{9r^6} + \frac{17d^4}{9r^4} - \frac{22d^2}{9r^2}\right)$$



$$g(d) = \left(1 - \frac{d^2}{r^2}\right)^3$$

Fall-off Filter Function



Blobby *Metaball* *Soft Objects* *Wyvill*

Continuity and Gradient

- C^0 continuity ensures no jump in the function
- For a 3D scalar field, gradient is defined as

$$\nabla f(\mathbf{p}) = \left\{ \frac{\partial f(\mathbf{p})}{\partial x}, \frac{\partial f(\mathbf{p})}{\partial y}, \frac{\partial f(\mathbf{p})}{\partial z} \right\}$$

- If $\nabla f(\mathbf{p})$ is C^0 for each component then f is C^1
  i.e. to say that the surface normal varies smoothly
  - If no unique normal can be defined on an edge of a surface (e.g. a cube) then we say that the surface is C^0

Distance Fields

- Distance field is defined with respect to some geometric object *T*

$$\mathbf{F}(T, \mathbf{p}) = \min_{\mathbf{q} \in T} |\mathbf{q} - \mathbf{p}|$$

- **F**(*T*, **p**) is the shortest distance from **p** to T
- T is any geometric entity embedded in 3D - a point, curve, or solid

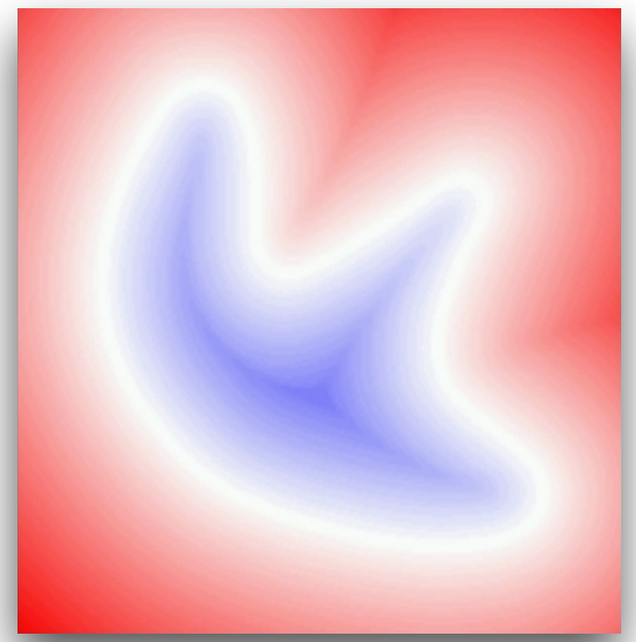
Distance Fields

- Distance field is defined with respect to some geometric object T

$$\mathbf{F}(T, \mathbf{p}) = \min_{\mathbf{q} \in T} |\mathbf{q} - \mathbf{p}|$$

- Solution to the *Eikonal equation*

$$|\nabla \mathbf{F}| = 1$$



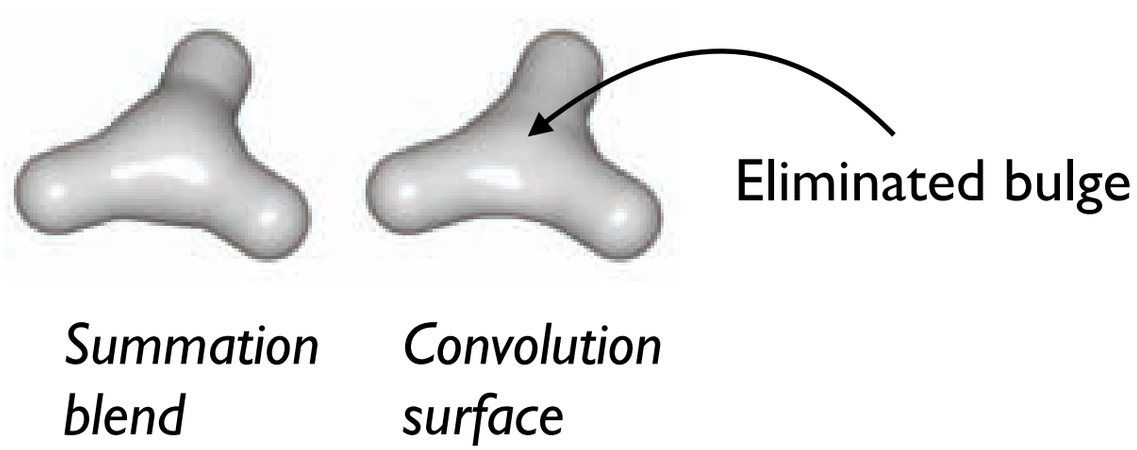
Source: UC Berkeley

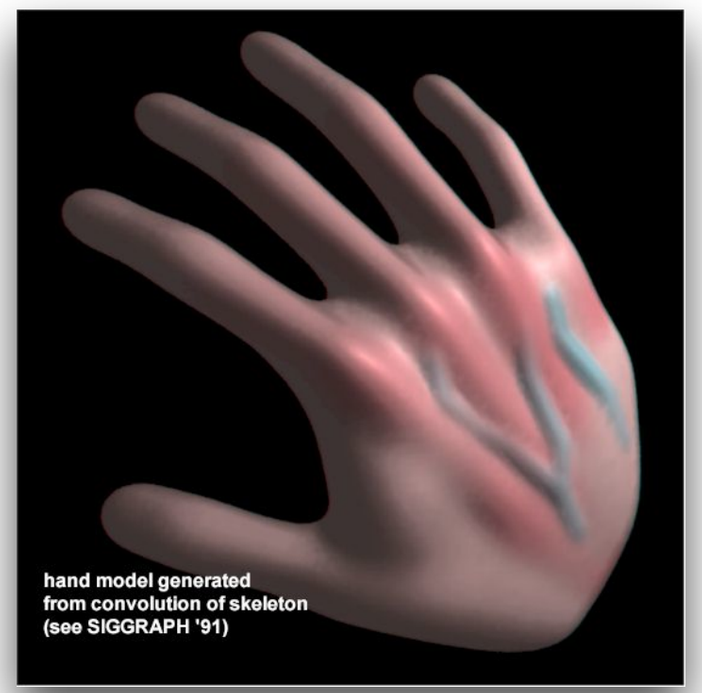
Convolution Surfaces

- Surfaces produced by convolving a geometric skeleton S with a kernel function *h*

$$f(\mathbf{p}) = \int_S g(\mathbf{r})h(\mathbf{p} - \mathbf{r})\mathrm{d}\mathbf{r}$$

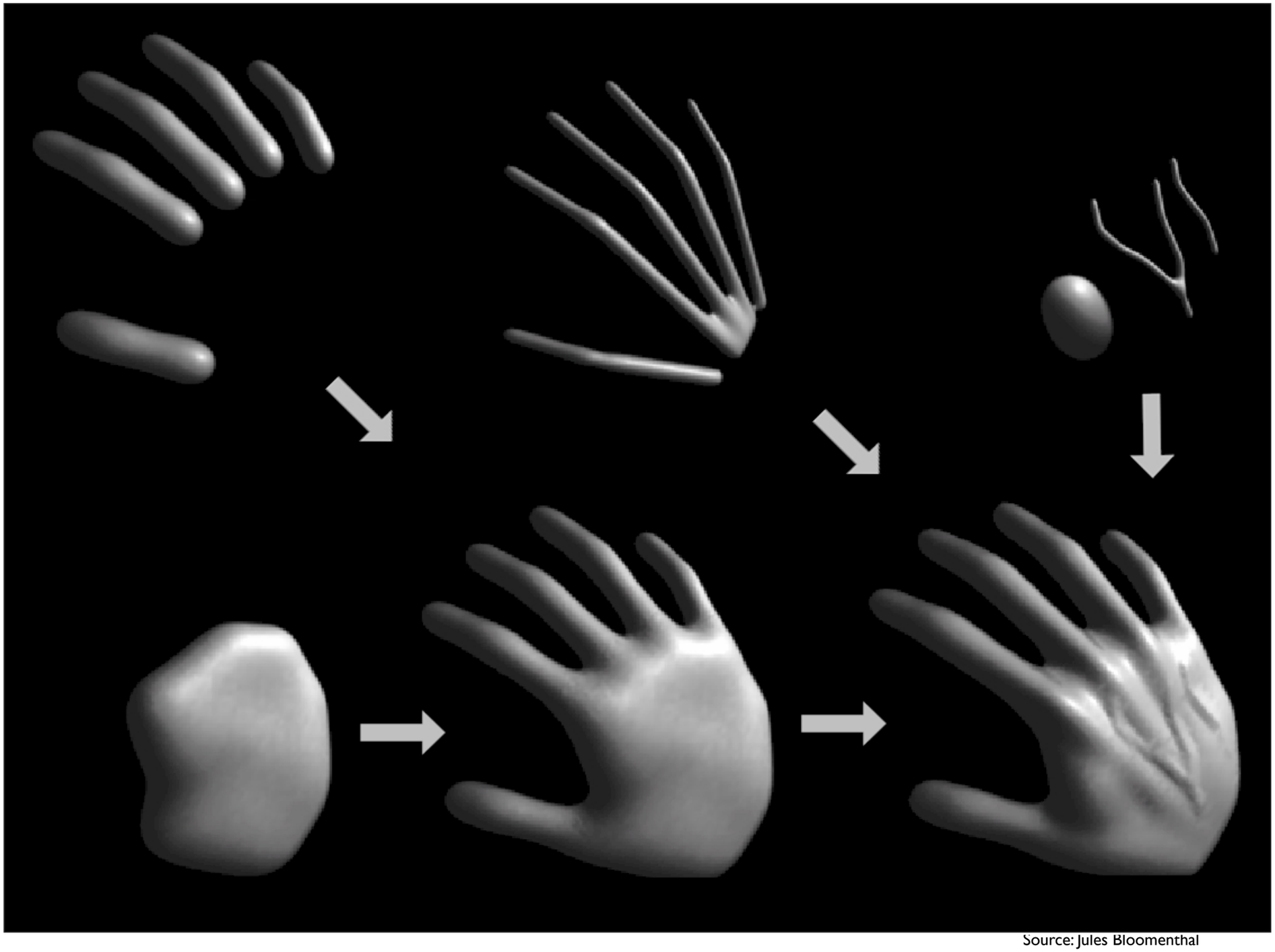
where *h* has finite support





Source: Jules Bloomenthal

Convolution Surfaces



Source: Jules Bloomenthal

Defining Skeletal Primitives

- In order to define an implicit function field, distance to the skeletal primitives need to be computed
- Simple for point primitives, but tricky for complex geometries

Line Segment Skeleton

- A line segment primitive (*AB*) can be defined as a cylinder around a line with hemispherical end caps

P_0 lies on the surface where $f(P_0)$ = iso

$f(P_1) = 0$ since P_1 lies outside the influence of line primitive

Line Segment Skeleton

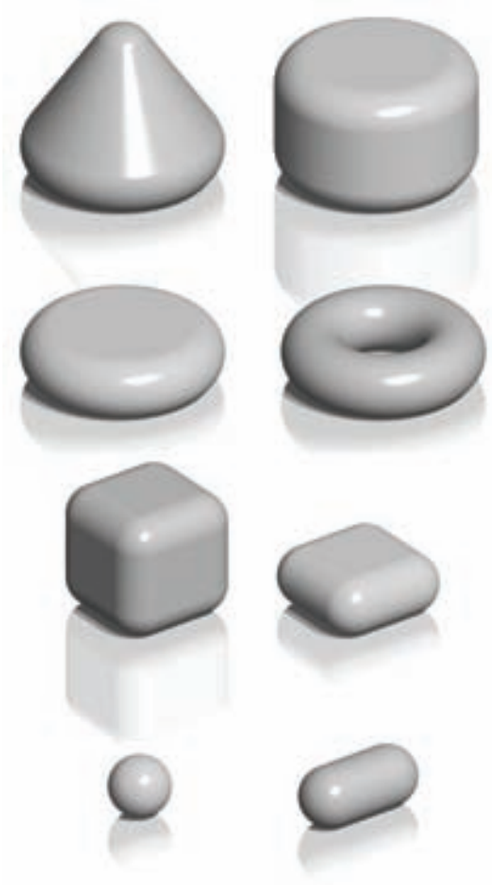
- The distance from some P_0 is found by projecting projecting onto AB and calculating perpendicular distance $|CP_0|$

Use AC to compute distance $|CP_0|$

$$\vec{AC} = \vec{AB}\frac{\vec{AP_0} \cdot \vec{AB}}{||AB||^2}$$

Defining Skeletal Primitives

- Distance computation for other skeletal primitives can be similarly defined



Rendering

Rendering an implicit function requires searching for a surface at an iso value

Two main techniques are:

1. Ray tracing
   i. ray marching in small steps
   ii. Algebraic evaluation
2. Polygonization (converting an iso-surface into a mesh)

Polygonization

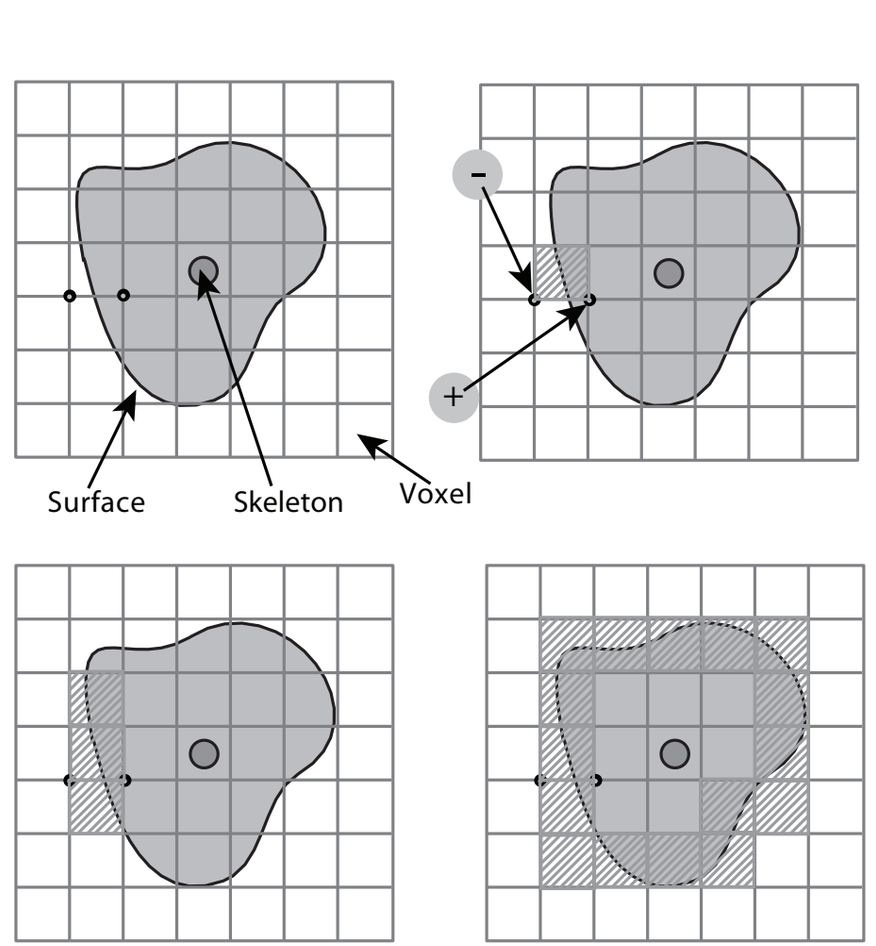
Algorithm based on *numerical continuation*:

- divide space into cubic voxels
- search for surface starting from a skeletal element (a starting point)
- add voxel to queue, mark it visited
- search neighbours
- when done, replace voxel with polygons

Polygonization, Phase - I

Polygonization

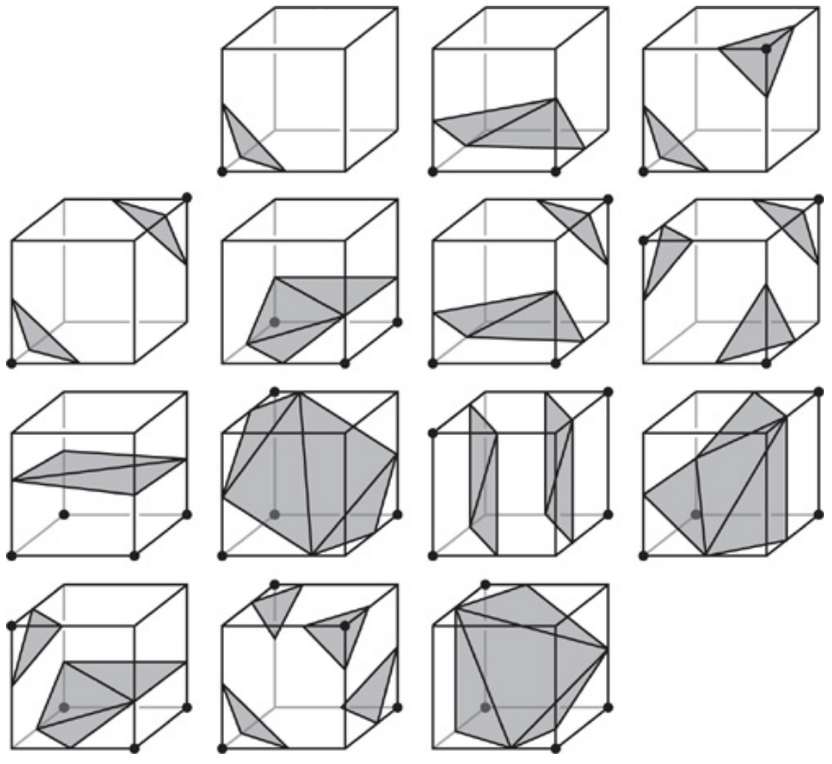
- Starting from the skeletal elements, search for a voxel containing at least one *intersecting edge*
- From this seed voxel, traverse neighbours and find other voxels containing intersecting edges
- Add such voxels to a queue for processing



Polygonization, Phase - II

Polygonization

- Each voxel is replaced by a set of triangles that best match the shape of the surface in that voxel
- A total of 256 (2^8) possible cases, but only 14 unique



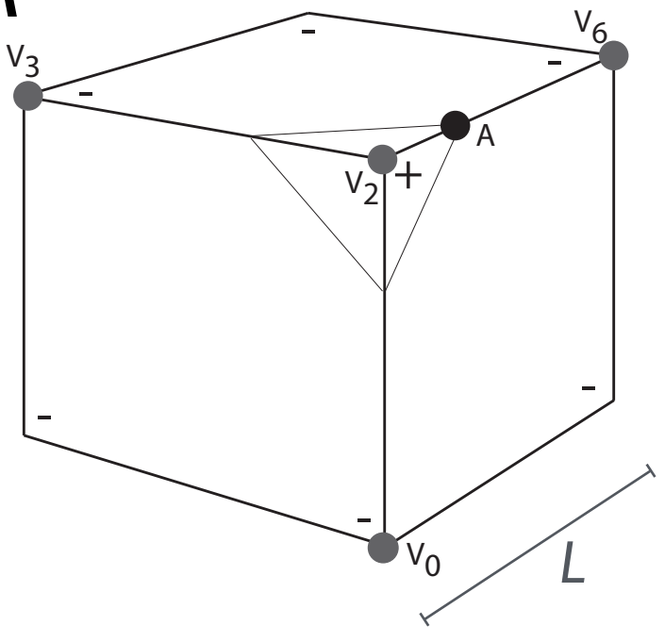
Source: GPU Gems III

Cube-Surface Intersections

Polygonization

- The surface intersects edge V_2-V_6 at A
- Use linear interpolation

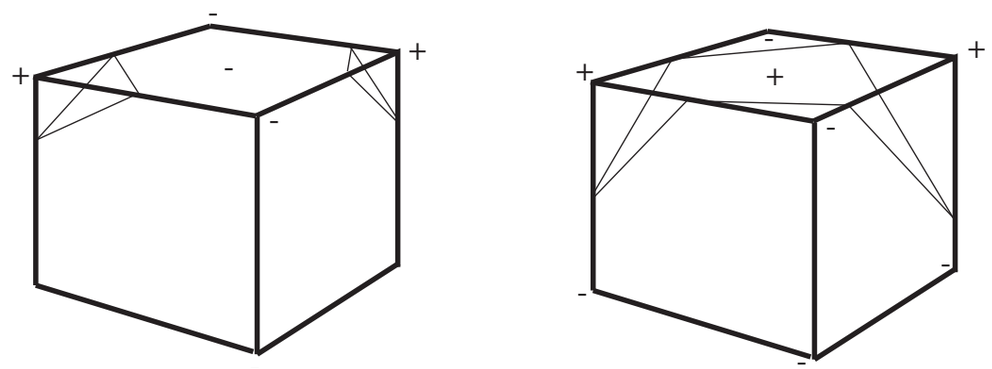
$$A = V_2 + L\frac{iso - f(V_2)}{f(V_6) - f(V_2)}$$



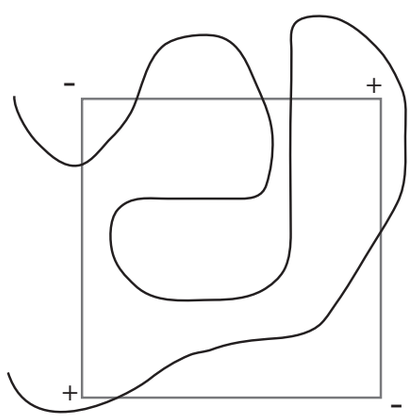
Sampling Problems

Polygonization

- **Ambiguous cases**: when opposite corners of a voxel have same sign and other corners have the other sign



- **Under-sampling**: function varies considerably within the voxel

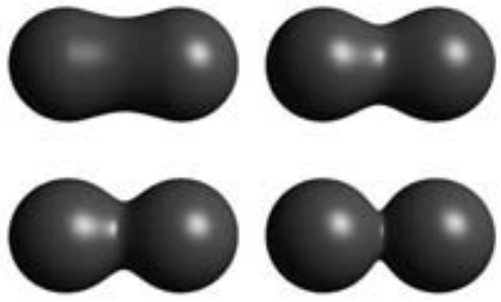


Super-elliptic Blending

- Given two functions f_A and f_B, denote a more general blending operator as $A \diamond B$
- The Ricchi blend (Ricchi, 1973) is defined as

$$f_{A \diamond B} = \left(f_A^n + f_B^n\right)^{\frac{1}{n}}$$

- Generalized blend varies from simple summation ($A + B$) to the union ($A \cup B$) as n varies from 1 to infinity



Super-elliptic Blending

- Generalized blend varies from simple summation ($A + B$) to the union ($A \cup B$) as n varies from 1 to infinity

$$\lim_{n \to +\infty} (f_A^n + f_B^n)^{\frac{1}{n}} = \max(f_A, f_B)$$

$$\lim_{n \to -\infty} (f_A^n + f_B^n)^{\frac{1}{n}} = \min(f_A, f_B)$$

Constructive Solid Geometry

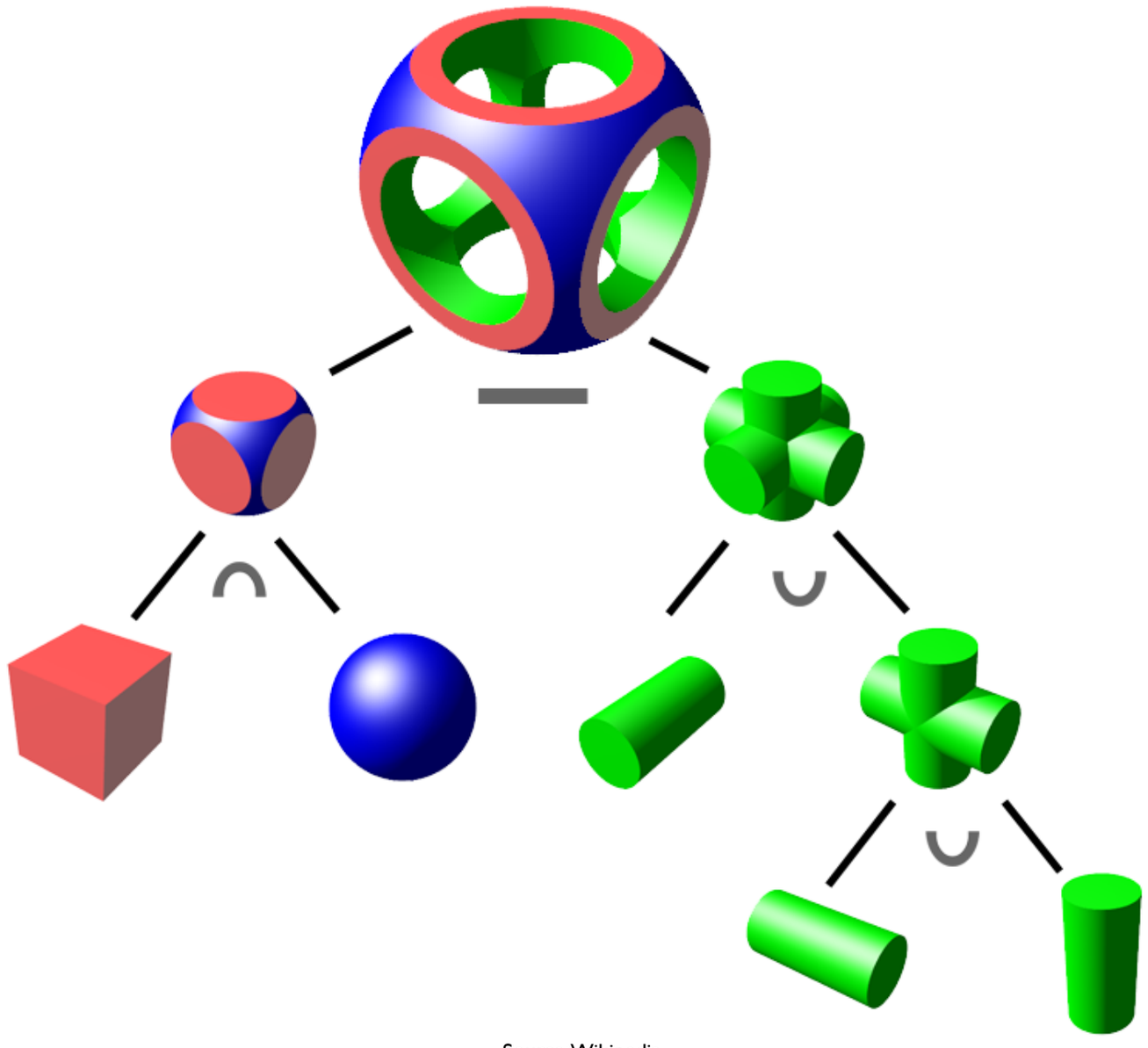
- Solid modelling for defining complex shapes from operations
- Uses union, intersection, difference, and blend upon primitives (Ricchi, 1973)
- The surface was considered as a boundary between half spaces $f(\mathbf{p}) < 1$ (inside) and $f(\mathbf{p}) > 1$ (outside)

Constructive Solid Geometry

- Boolean set operations defined for implicit functions
- Typically evaluated bottom-up according to binary tree
  - Leaf nodes: low degree polynomial primitives
  - Internal nodes: boolean set operations
- Supported boolean operations include:

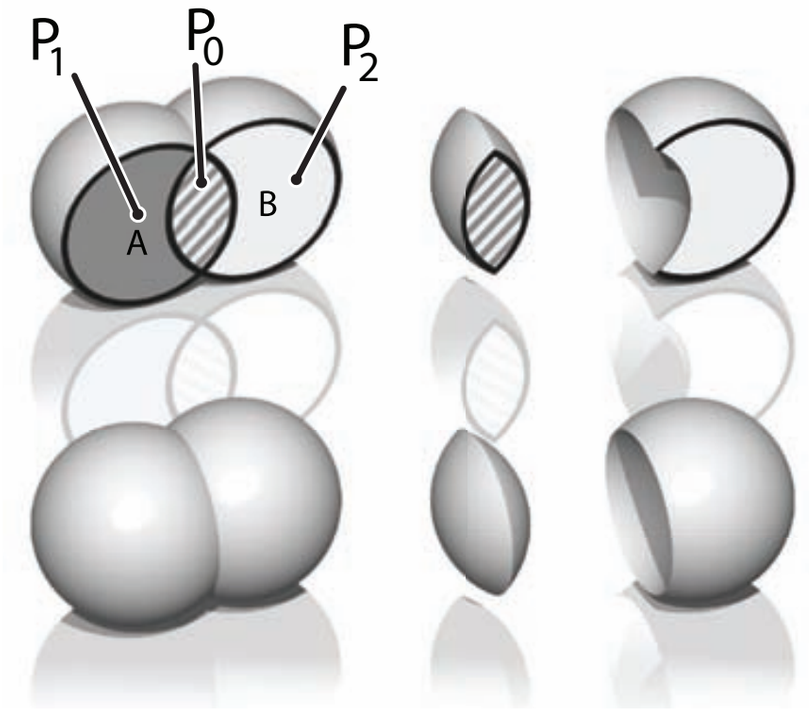
  union $\cup_{max}$, intersection $\cap_{min}$, and difference $\setminus_{min\ max}$

Constructive Solid Geometry



Source: Wikipedia

Constructive Solid Geometry



Union $\cup_{\max} f = \max_{i=0}^{k-1} (f_i)$,

Intersection $\cap_{\min} f = \min_{i=0}^{k-1} (f_i)$,

Difference $\setminus_{\min \max} f = \min \left(f_0, 2 * \text{iso} - \max_{j=1}^{k-1} (f_j) \right)$

- CSG operators create creases, i.e., C^1 discontinuities

Warping

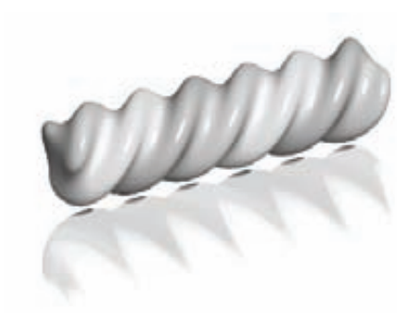
- Distorting shape of a surface by warping the space in its neighbourhood
- A warp is a continuous function $w(x, y, z) : \mathbb{R}^3 \mapsto \mathbb{R}^3$
- Define warped element by applying $w(\mathbf{p})$ to the implicit equation:

$$f_i(\mathbf{p}) = g_i \circ d_i \circ w_i(\mathbf{p})$$

Warping - Twist

- Twist around an axis by θ
- Twist around z-axis is expressed as

$$w(x, y, z) = \begin{Bmatrix} x\cos(\theta(z)) - y\sin(\theta(z)) \\ x\sin(\theta(z)) + y\cos(\theta(z)) \\ z \end{Bmatrix}$$



Warping - Taper

- Taper is applied along one major axis
- Examples include linear, quadratic and cubic tapers
- A linear taper along y-axis could be:

$$s(y) = \frac{y_{max} - y}{y_{max} - y_{min}}, w(x, y, z) = \begin{Bmatrix} s(y)x \\ y \\ s(y)z \end{Bmatrix}$$

Warping - Bend

- Bend is applied along one major axis
- Bending rate *k* measured in radians per unit length
- A linear taper along y-axis could be:

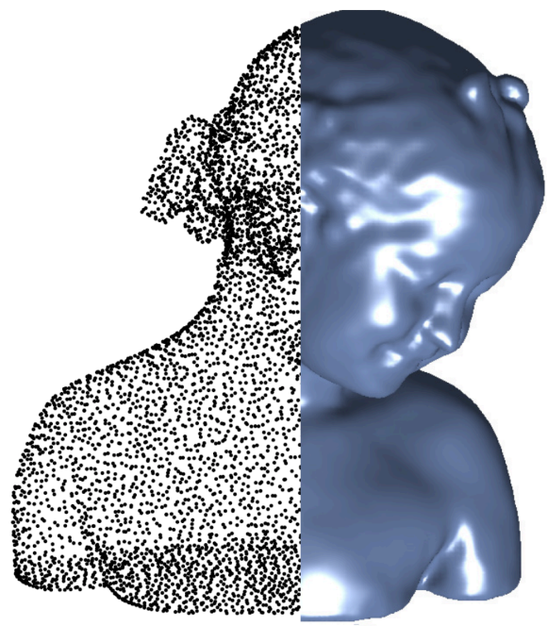
$$w(x, y, z) = \begin{Bmatrix} -\sin(\theta)(y - 1/k) + x_0 \\ \cos(\theta)(y - 1/k) + 1/k \\ z \end{Bmatrix}$$

Surface Reconstruction

Given a set of points in $\mathbb{R}^2$ or $\mathbb{R}^3$, find a surface passing through the set of points.

Otherwise, known as the *scattered data interpolation* problem,

- RBF reconstruction ✓
- Poisson reconstruction



Source: Mohammad Rouhani, https://in.mathworks.com/matlabcentral/fileexchange/44654-surface-reconstruction-using-implicit-b-splines

Gaussian RBF

Given values $f_i = f(\mathbf{p}_i)$ of an unknown function f for data points $\mathbf{p}_i$ find a function:

$$s(\mathbf{p}) = \sum_i \lambda_i \phi(\|\mathbf{p} - \mathbf{p}_i\|)$$

such that $s(\mathbf{p}_i) = f(\mathbf{p}_i)$ where $\phi(r) = \exp(-\alpha r^2)$.
This gives exact interpolation, if we solve the system:

$$f_i = \sum_j \lambda_j \phi(\|\mathbf{p}_i - \mathbf{p}_j\|)$$

Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

General RBF

In general, we need a polynomial term

$$f_i = \sum_j \lambda_j \phi(\|\mathbf{p}_i - \mathbf{p}_j\|) + P(\mathbf{p}_i)$$

where $P(\mathbf{p}) = c + c_x p_x + c_y p_y + c_z p_z$.
Leads to linear system below

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}$$

Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

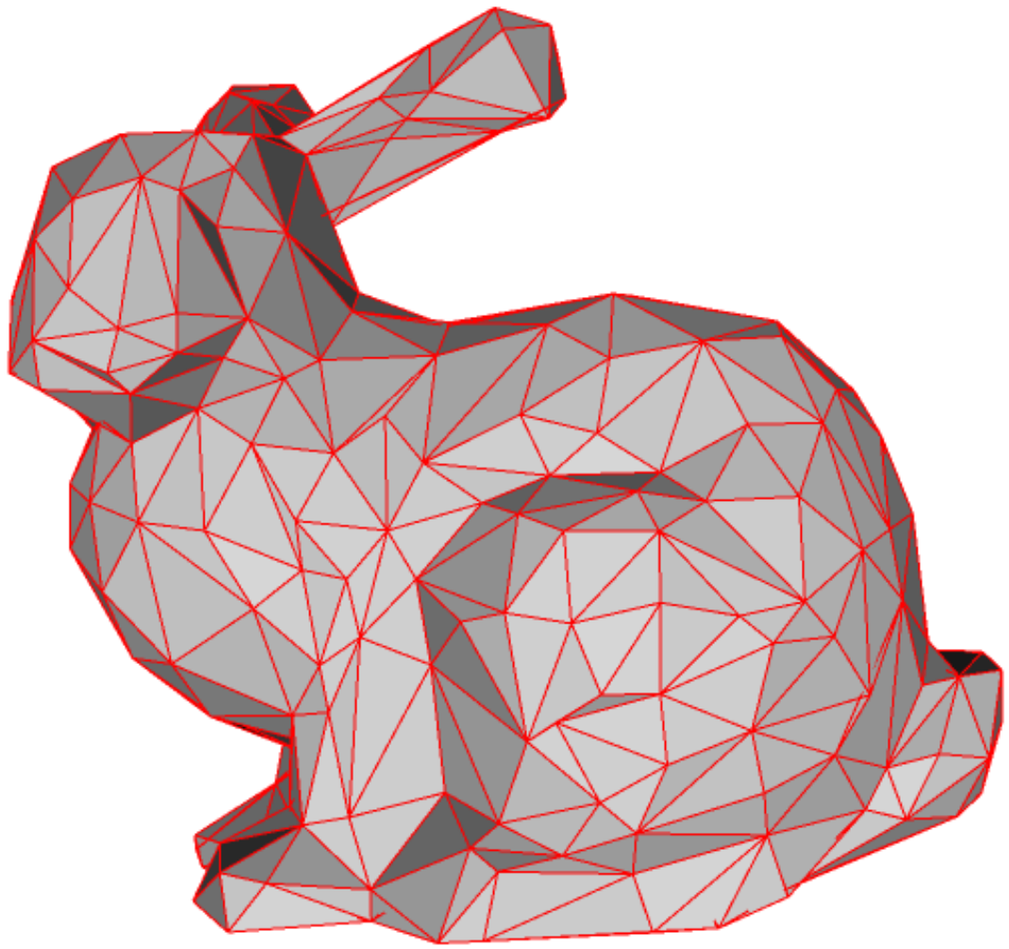
Radial Basis Functions

A few commonly used RBFs

Name	$\phi(r)$	polynomial
Gaussian	$\phi(r) = \exp(-\alpha r^2)$	N/A
Linear	$\phi(r) = r$	constant
Thin Plate Spline	$\phi(r) = r^2 \log(r)$	linear
Cubic	$\phi(r) = r^3$	linear

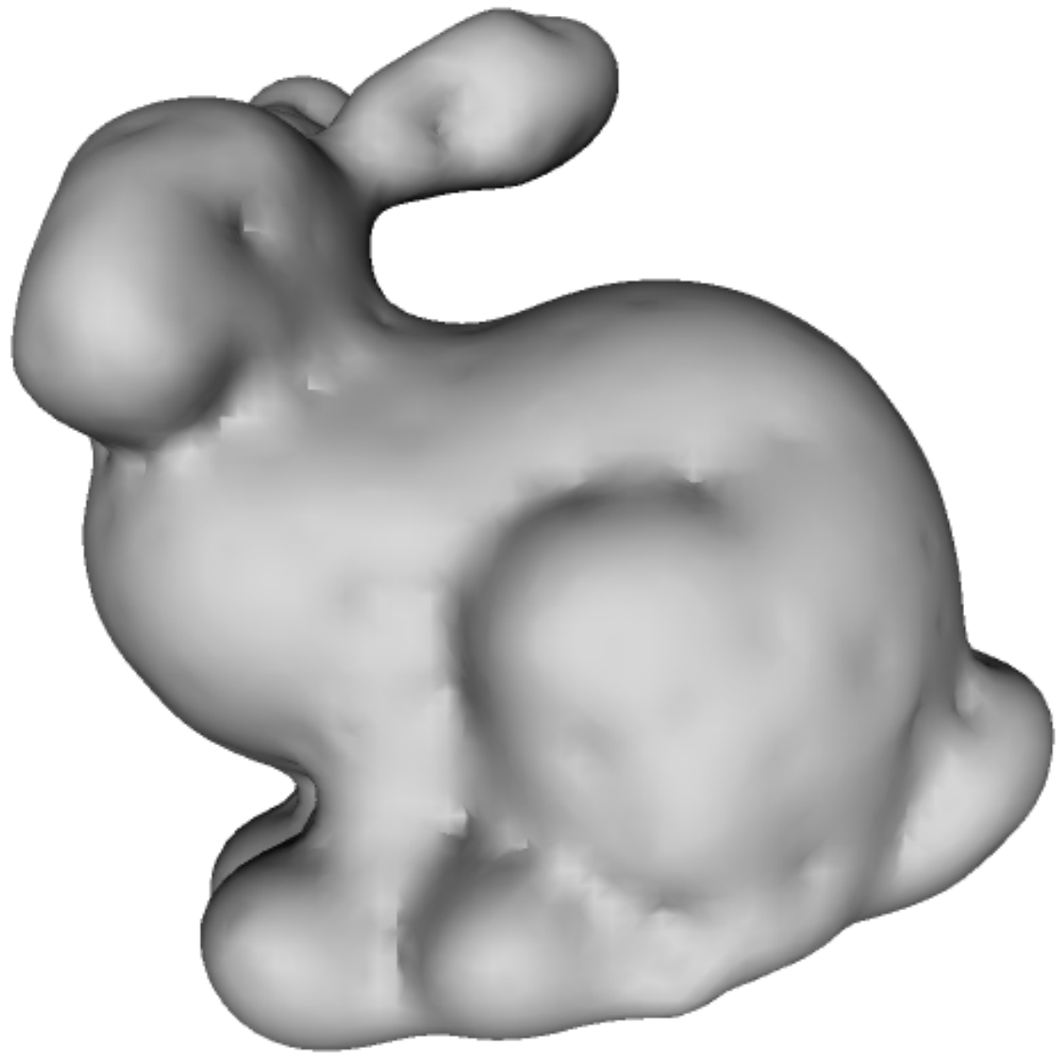
Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Original Mesh



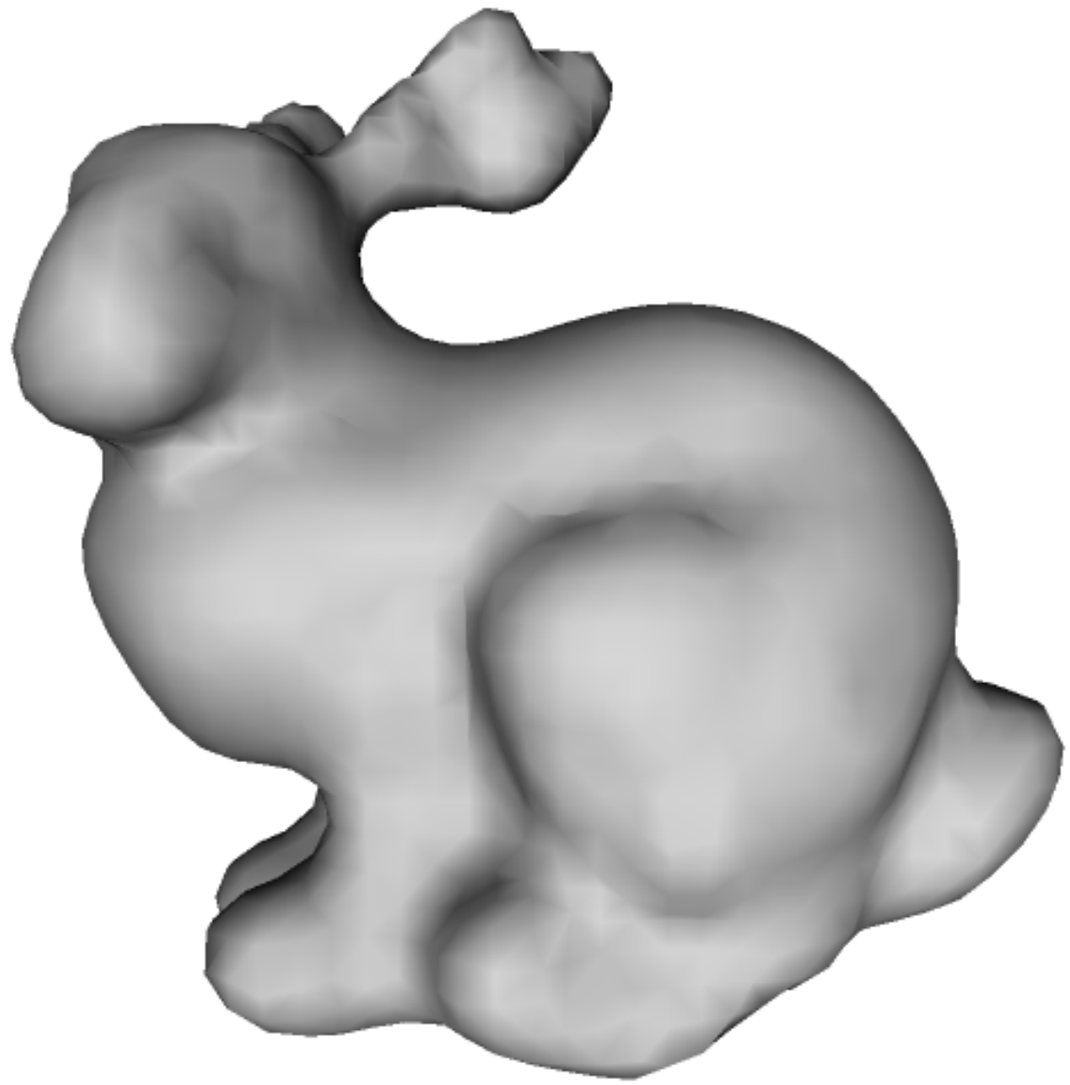
Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Linear RBF, $\phi(r) = r$



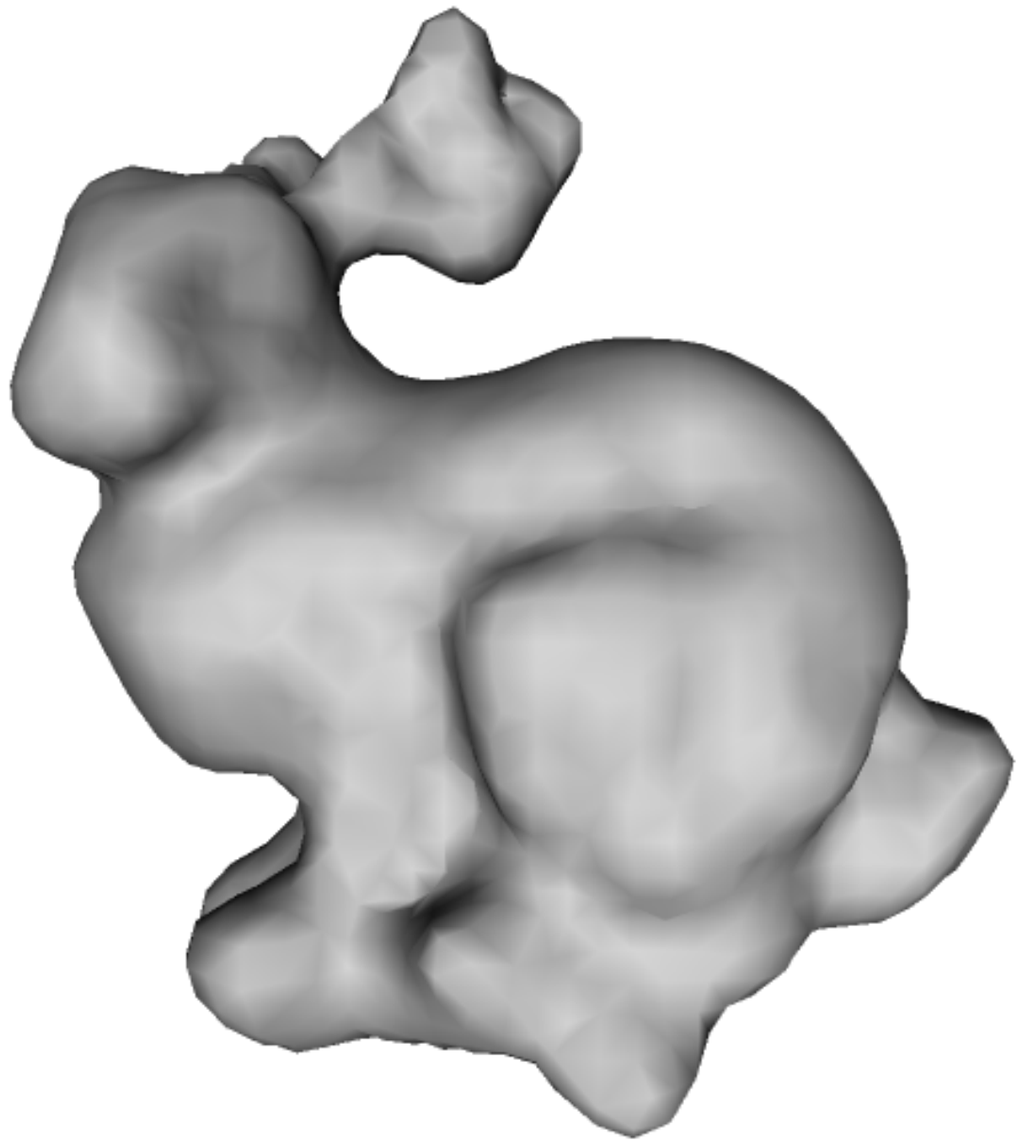
Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Cubic RBF, $\phi(r) = r^3$



Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Gaussian RBF, $\phi(r)=exp(-\alpha r^2)$



Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Regularization

Regularization is used to make the system less ill-conditioned:

$$f_i = \sum_j \lambda_j \phi(\|\mathbf{p}_i - \mathbf{p}_j\|) + P(\mathbf{p}_i) + k\lambda_i$$

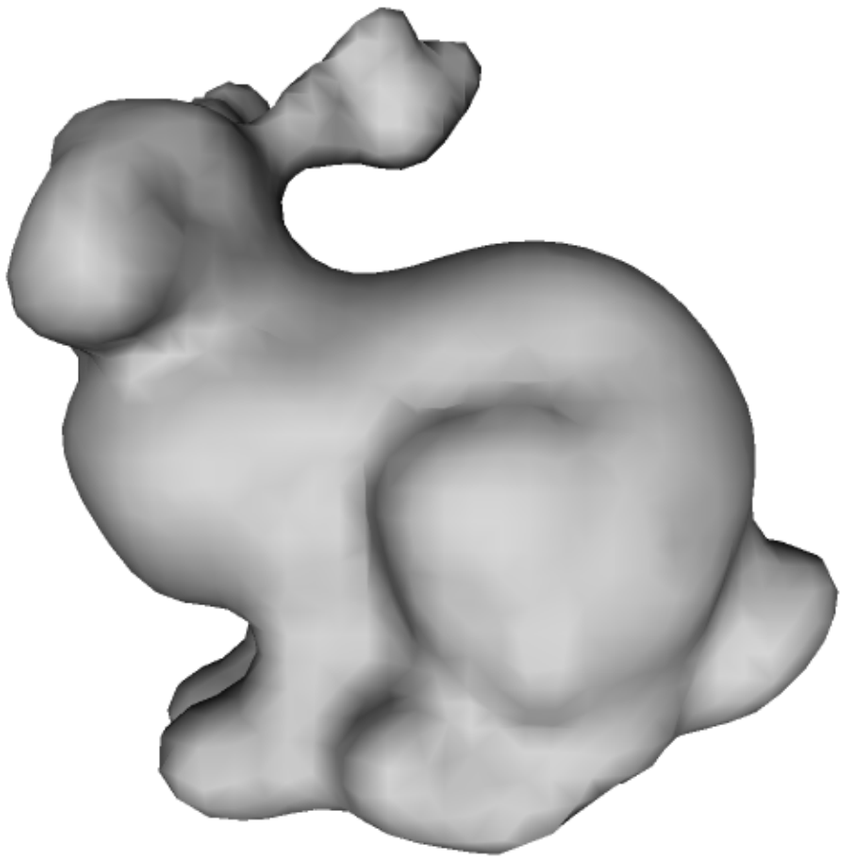
Leads to linear system below

$$\begin{bmatrix} \mathbf{\Phi} + k\mathbf{I} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}$$

for regularization constant k.

Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Regularization



Cubic RBF
Without regularisation

Cubic RBF
With regularisation

Slides courtesy: A. Baerentzen, Computational Geometry Processing, 2007.

Reading

- FCG: 16

ICG: Interactive Computer Graphics, E. Angel, and D. Shreiner, 6th ed.
FCG: Fundamentals of Computer Graphics, P. Shirley, M. Ashikhmin, and S. Marschner, 3rd ed.