

Introduction to Computer Graphics

– Modeling (3) –

May 16, 2019

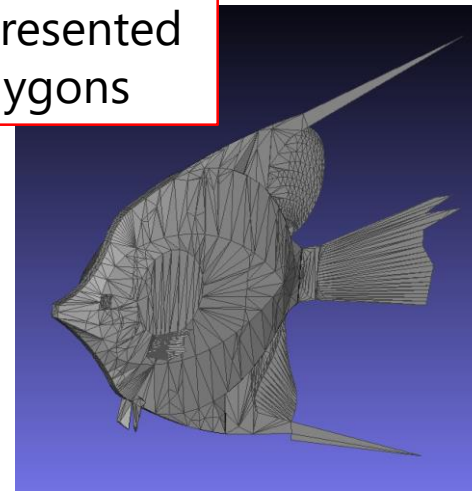
Kenshi Takayama

Solid modeling

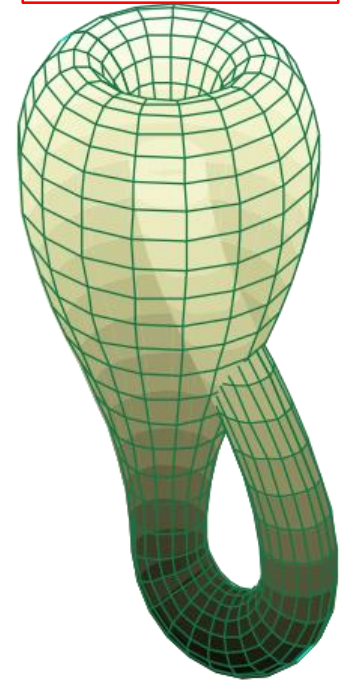
Solid models

- Clear definition of "inside" & "outside" at any 3D point

Thin shapes represented by single polygons

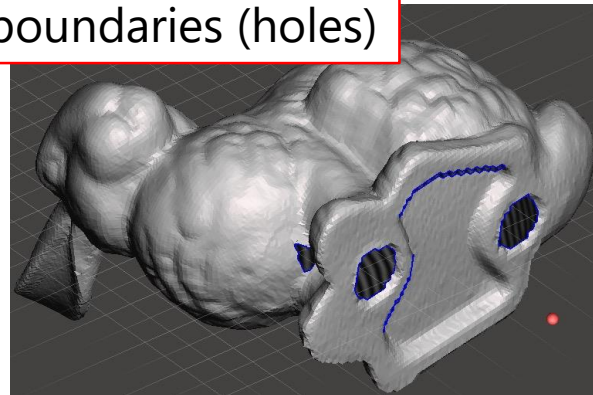


Unorientable



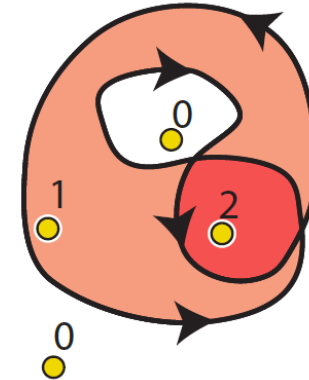
Klein bottle

Open boundaries (holes)



Non-solid cases

Self-intersections

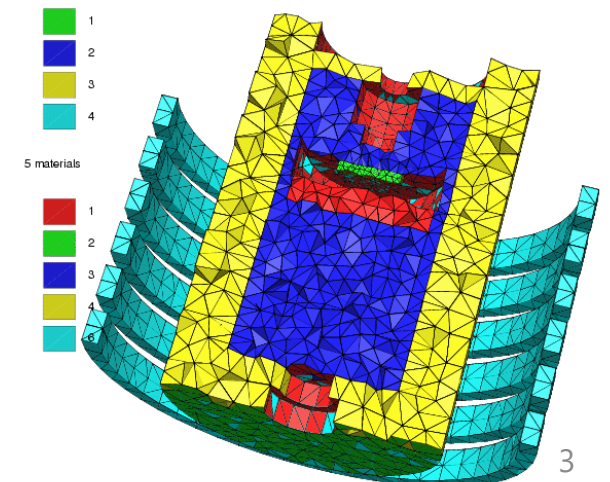


- Main usage:

3D printing



Physics simulation



Predicate function of a solid model

- Function that returns true/false if a 3D point $\mathbf{p} \in \mathbb{R}^3$ is inside/outside of the model

$$f(\mathbf{p}): \mathbb{R}^3 \mapsto \{ \text{true}, \text{false} \}$$

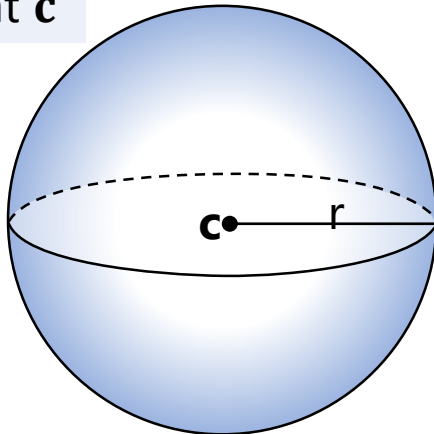
- The whole interior of the model:

$$\{ \mathbf{p} \mid f(\mathbf{p}) = \text{true} \} \subset \mathbb{R}^3$$

- Examples:

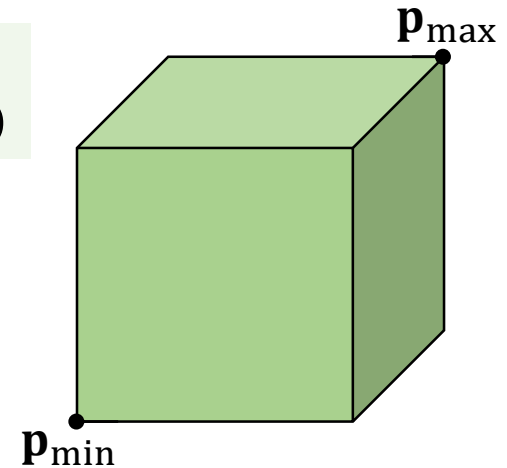
Sphere of radius r centered at \mathbf{c}

$$f(\mathbf{p}) := \|\mathbf{p} - \mathbf{c}\| < r$$



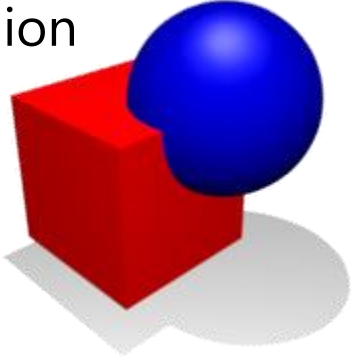
Box whose min & max corners are $(x_{\min}, y_{\min}, z_{\min})$ & $(x_{\max}, y_{\max}, z_{\max})$

$$\begin{aligned} f(x, y, z) := & (x_{\min} < x < x_{\max}) \\ & \wedge (y_{\min} < y < y_{\max}) \\ & \wedge (z_{\min} < z < z_{\max}) \end{aligned}$$



Constructive Solid Geometry (Boolean operations)

Union



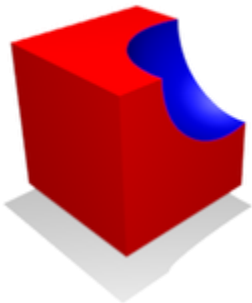
$$f_{A \cup B}(\mathbf{p}) := f_A(\mathbf{p}) \vee f_B(\mathbf{p})$$

Intersection



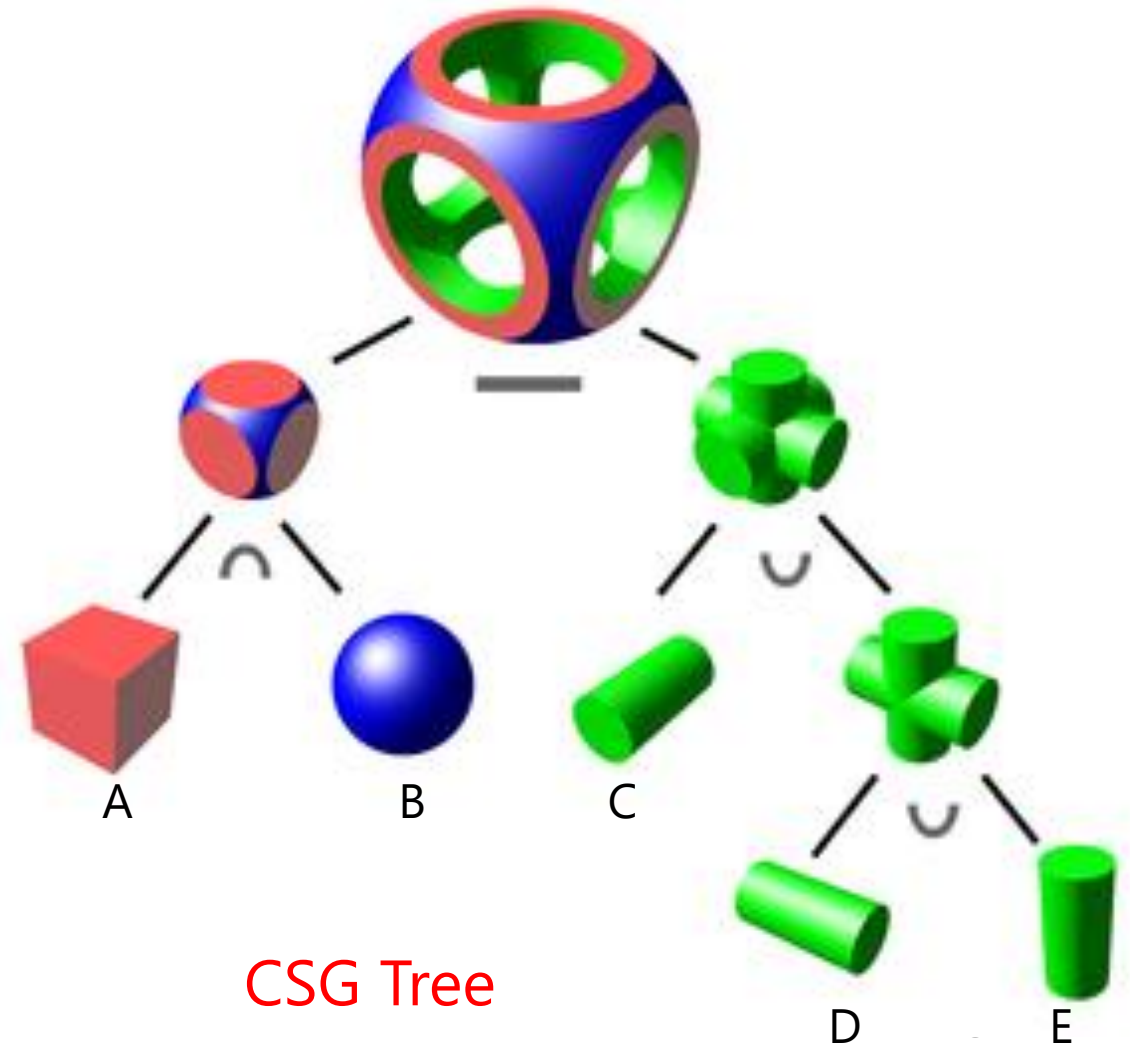
$$f_{A \cap B}(\mathbf{p}) := f_A(\mathbf{p}) \wedge f_B(\mathbf{p})$$

Subtraction



$$f_{A \setminus B}(\mathbf{p}) := f_A(\mathbf{p}) \wedge \neg f_B(\mathbf{p})$$

$$(A \cap B) \setminus (C \cup (D \cup E))$$

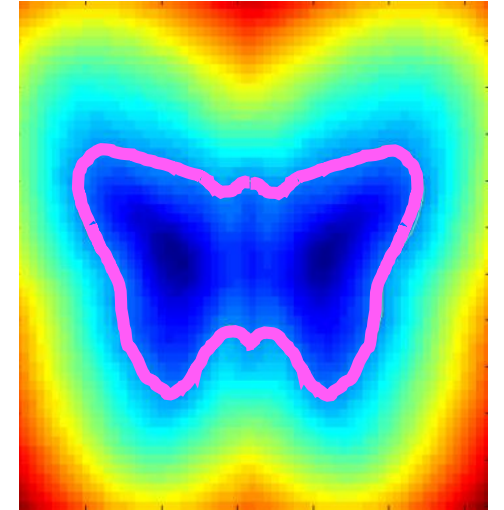


Solid model represented by Signed Distance Field

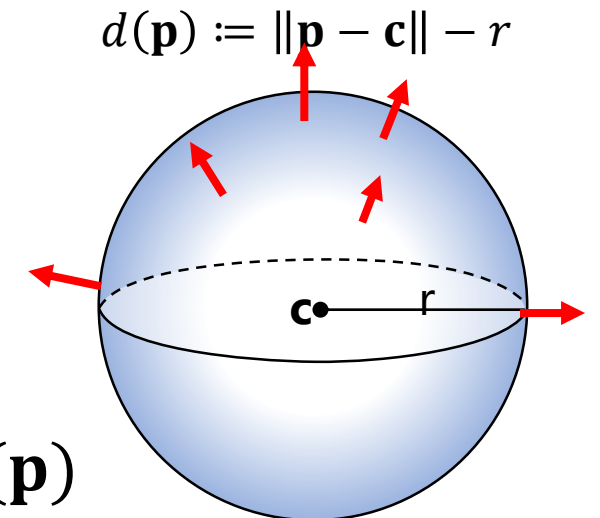
- Shortest distance from 3D point to model surface:

$$d(\mathbf{p}): \mathbb{R}^3 \mapsto \mathbb{R}$$

- Signed: positive \rightarrow outside, negative \rightarrow inside
- Corresponding predicate describing the solid:
$$f(\mathbf{p}) := d(\mathbf{p}) < 0$$
- Zero isosurface \rightarrow model surface:
$$\{\mathbf{p} \mid d(\mathbf{p}) = 0\} \subset \mathbb{R}^3$$
- Aka. "implicit" or "volumetric" representation
- Isosurface **normal** matches with direction of gradient $\nabla d(\mathbf{p})$

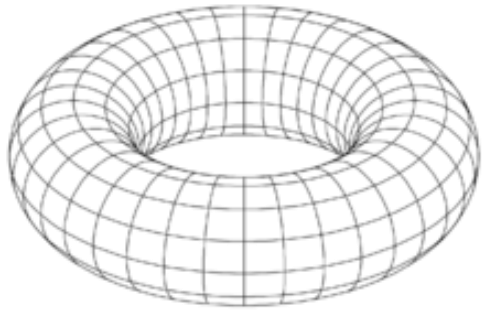


Sphere of radius r centered at \mathbf{c}



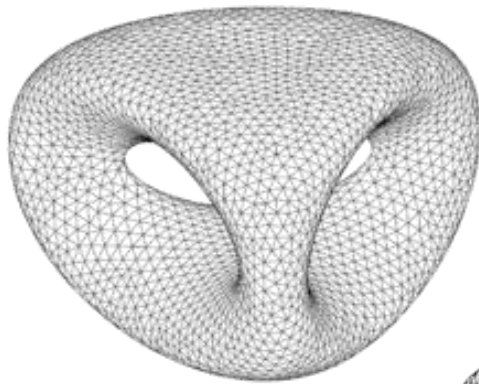
Examples of implicit functions

Not necessarily distance functions

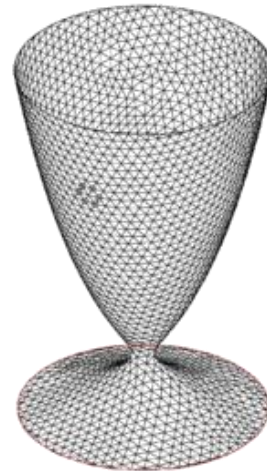


Torus with major & minor radii R & a

$$(x^2 + y^2 + z^2 + R^2 - a^2)^2 - 4R^2(x^2 + y^2) = 0$$

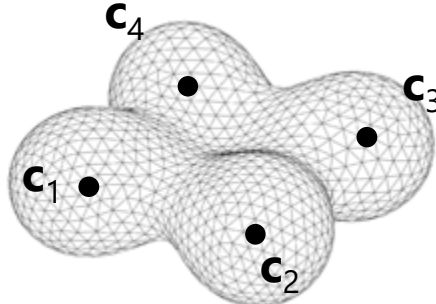


$$2y(y^2 - 3x^2)(1 - z^2) + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2) = 0$$



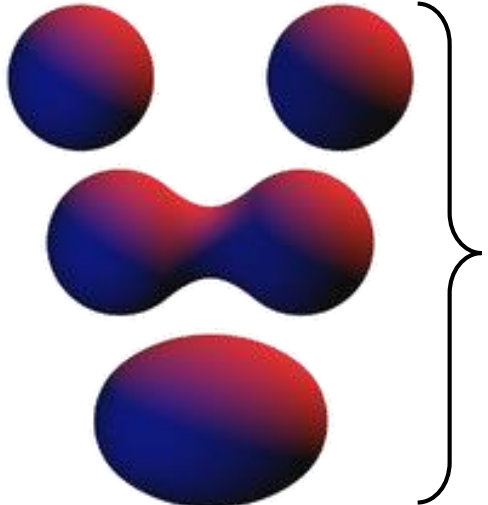
$$x^2 + y^2 - (\ln(z + 3.2))^2 - 0.02 = 0$$

Examples of implicit functions: Metaballs

$$d_i(\mathbf{p}) = \frac{q_i}{\|\mathbf{p} - \mathbf{c}_i\|} - r_i$$



$$d(\mathbf{p}) = d_1(\mathbf{p}) + d_2(\mathbf{p}) + d_3(\mathbf{p}) + d_4(\mathbf{p})$$

1



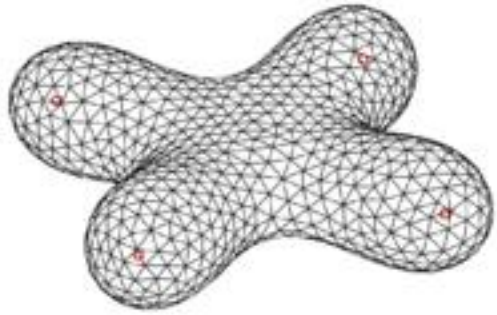
$$d(\mathbf{p}) = d_1(\mathbf{p}) + d_2(\mathbf{p})$$

2

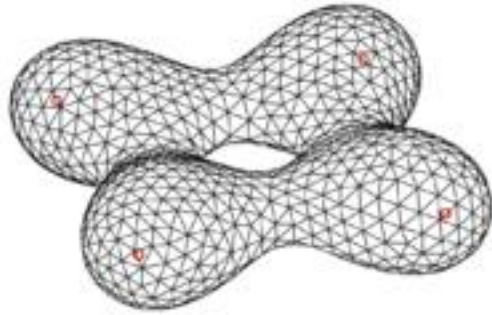


$$d(\mathbf{p}) = d_1(\mathbf{p}) - d_2(\mathbf{p})$$

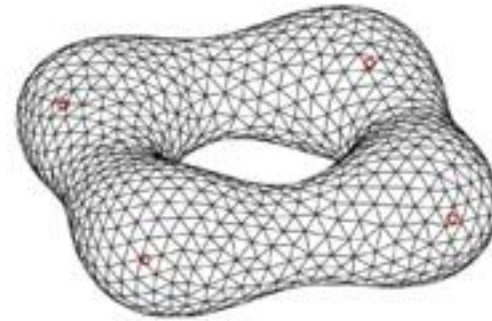
Morphing by interpolating implicit functions



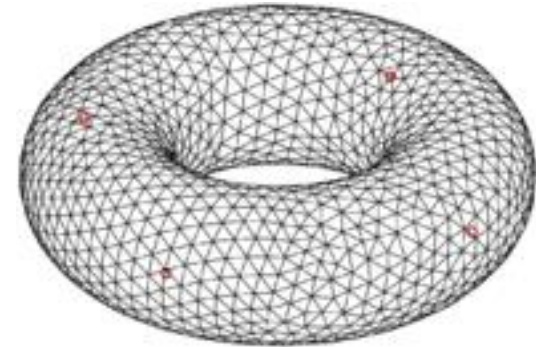
$$d_1(\mathbf{p}) = 0$$



$$\frac{2}{3}d_1(\mathbf{p}) + \frac{1}{3}d_2(\mathbf{p}) = 0$$



$$\frac{1}{3}d_1(\mathbf{p}) + \frac{2}{3}d_2(\mathbf{p}) = 0$$



$$d_2(\mathbf{p}) = 0$$

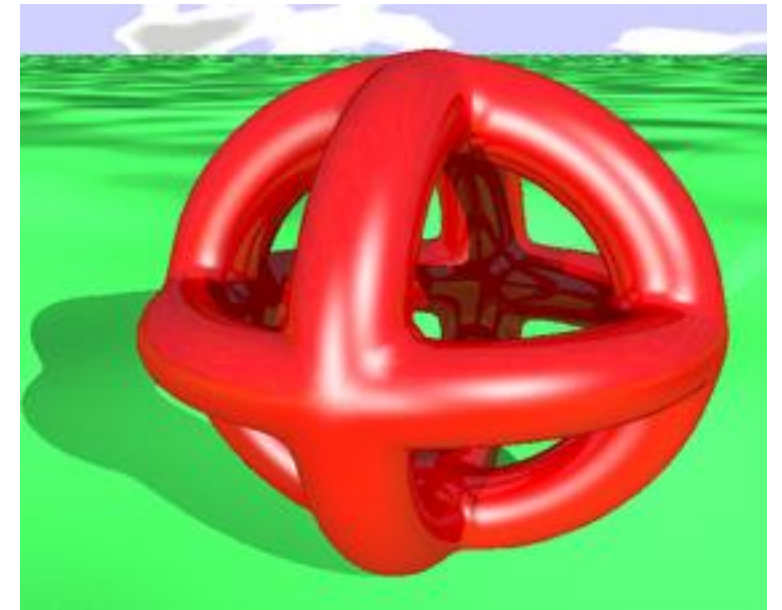
Modeling by combining implicit functions

$$F_1 = (x^2 + y^2 + z^2 + R^2 - a^2)^2 - 4R^2(x^2 + y^2) = 0$$

$$F_2 = (x^2 + y^2 + z^2 + R^2 - a^2)^2 - 4R^2(x^2 + z^2) = 0$$

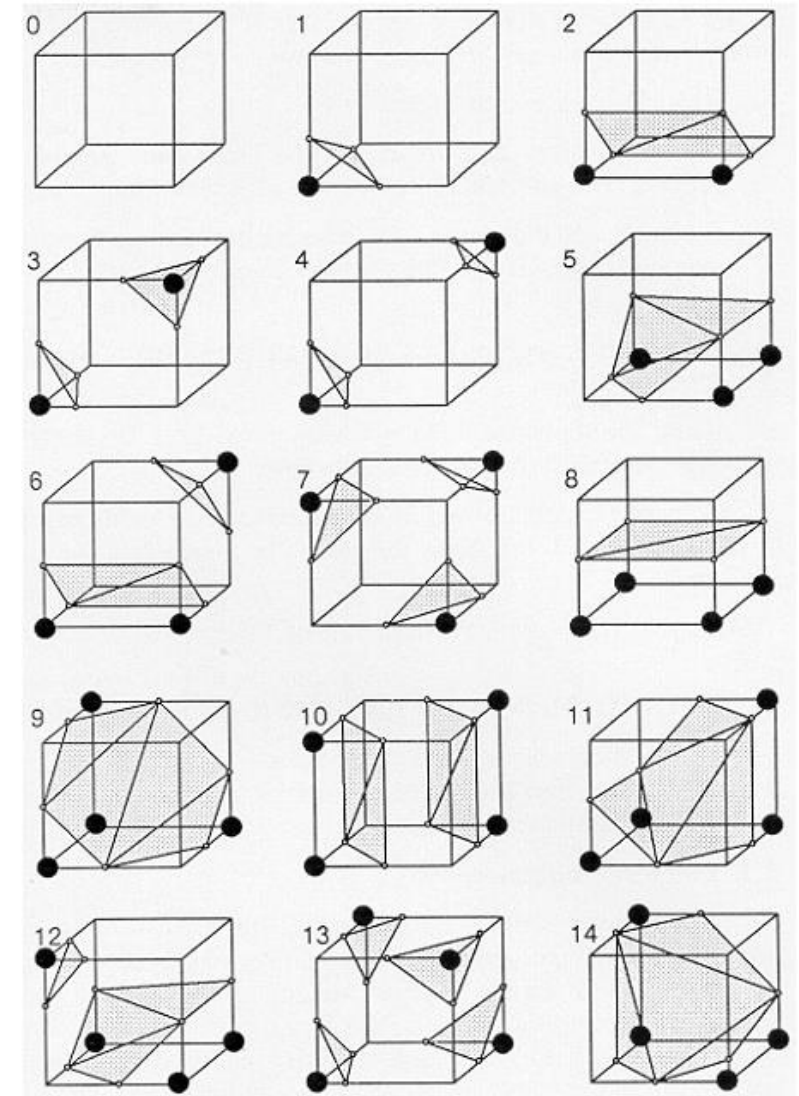
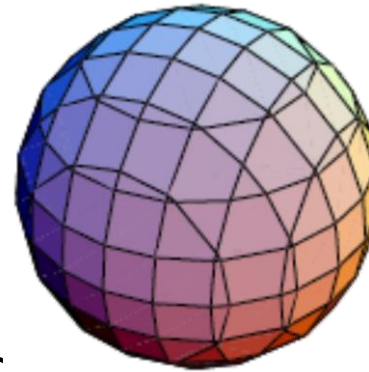
$$F_3 = (x^2 + y^2 + z^2 + R^2 - a^2)^2 - 4R^2(y^2 + z^2) = 0$$

$$F(x, y, z) = F_1(x, y, z) \cdot F_2(x, y, z) \cdot F_3(x, y, z) - c = 0$$



Visualizing implicit functions: Marching Cubes

- Extract isosurface as triangle mesh
- For every lattice cell:
 - (1) Compute function values at 8 corners
 - (2) Determine type of output triangles based on the sign pattern
 - Classified into 15 using symmetry
 - (3) Determine vertex positions by linearly interpolating function values



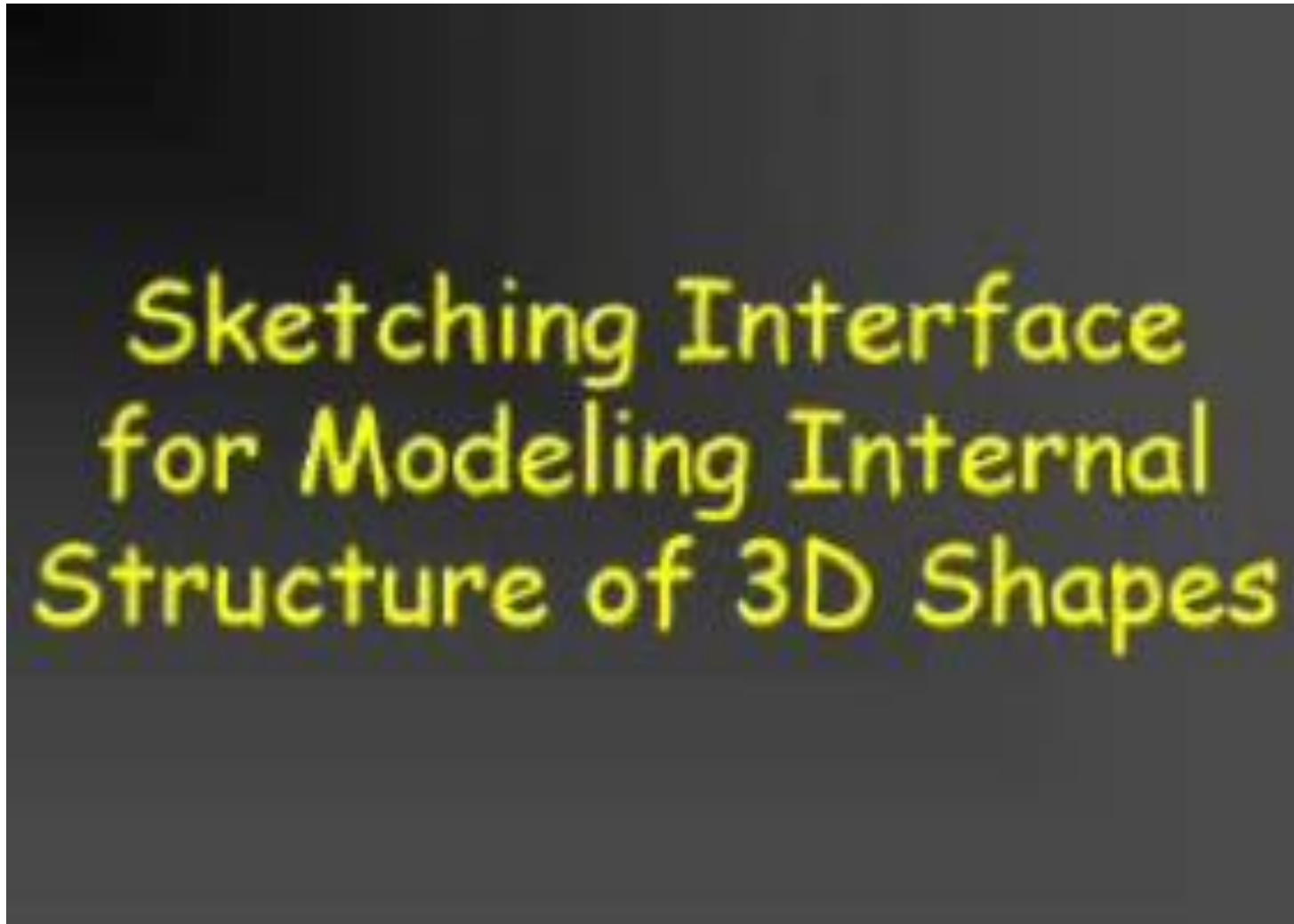
(Once patented☹, now expired☺)

Example of 3D modeling tool using implicit surfaces

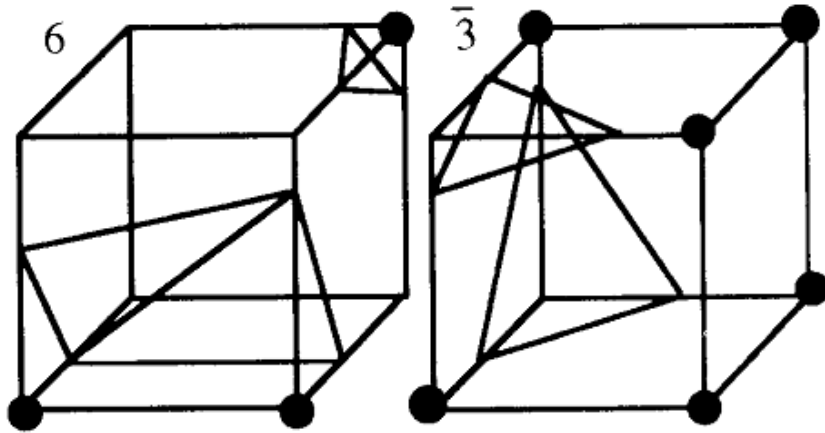
A black rectangular box containing white text. The text is centered and reads "ShapeShop v002" in a large, bold, sans-serif font, with "Demo Reel" in a smaller, bold, sans-serif font below it.

ShapeShop v002
Demo Reel

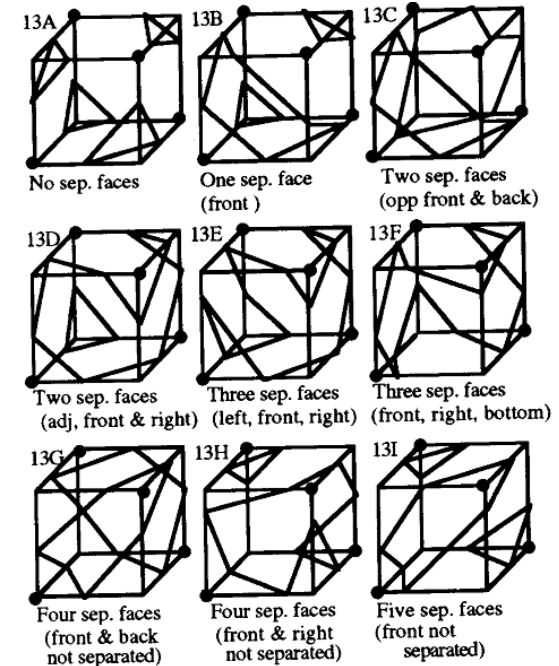
Example of 3D modeling tool using implicit surfaces



Ambiguity in Marching Cubes



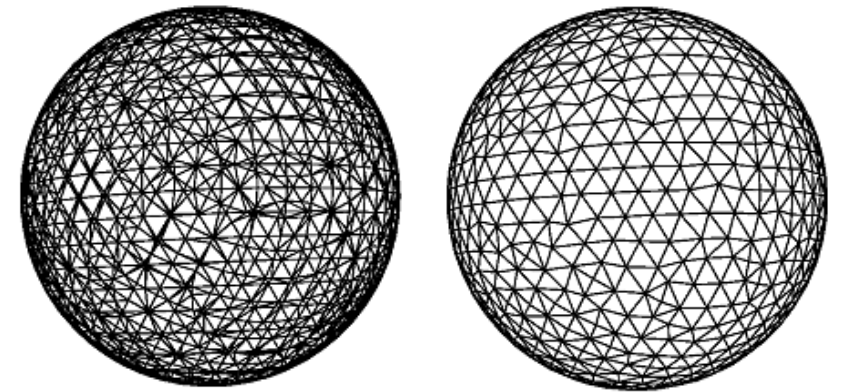
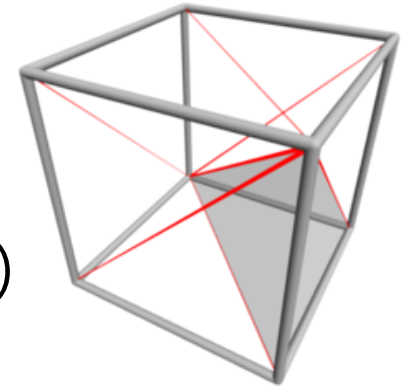
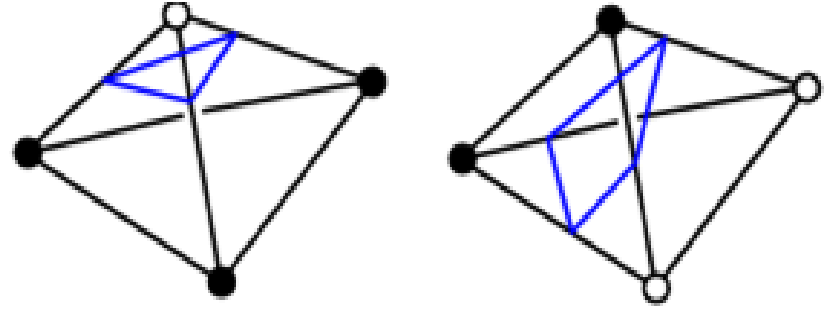
Discontinuous faces across neighboring cells



New rules to resolve ambiguity

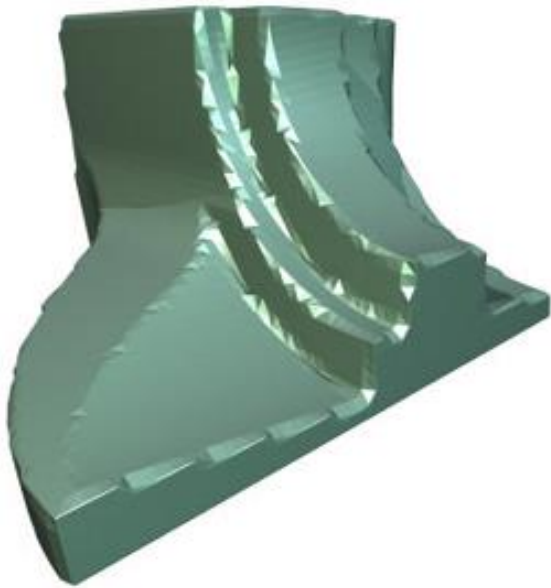
Marching Tetrahedra

- Use tetrahedra instead of cubes
 - Fewer patterns, no ambiguity
→ Simpler implementation
- A cube split into 6 tetrahedra
 - (Make sure consistent splitting across neighboring cubes)
- Some techniques to improve mesh quality

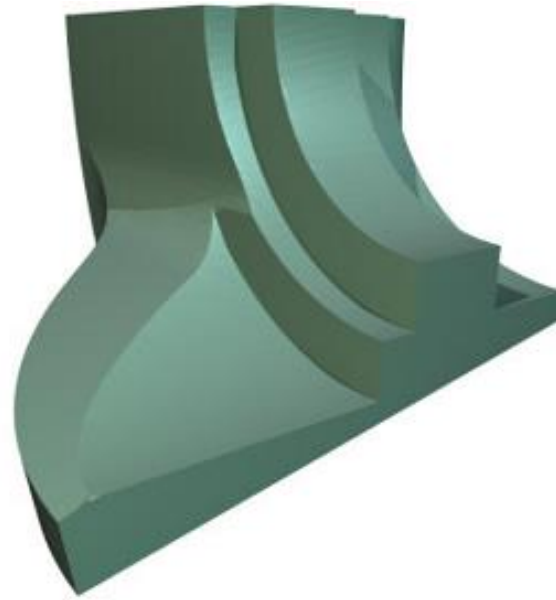


Isosurface extraction preserving sharp edges

Grid size: $65 \times 65 \times 65$

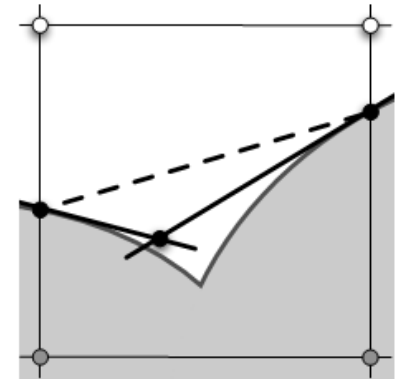
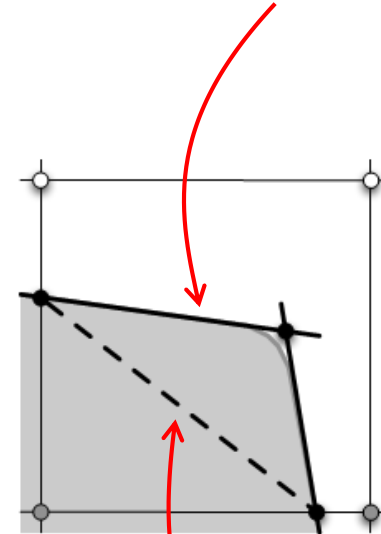


Marching Cubes



Improved version

Improved version (uses function *gradient* as well)



Marching Cubes (only uses function values)

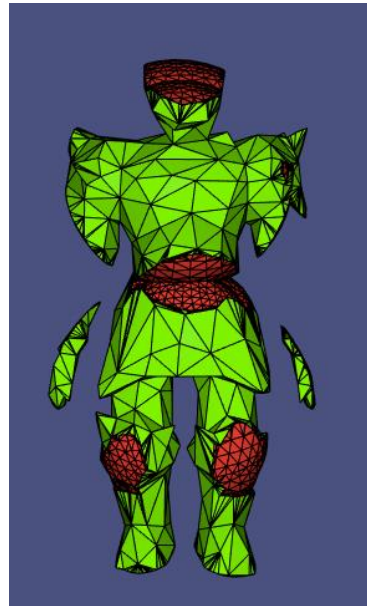
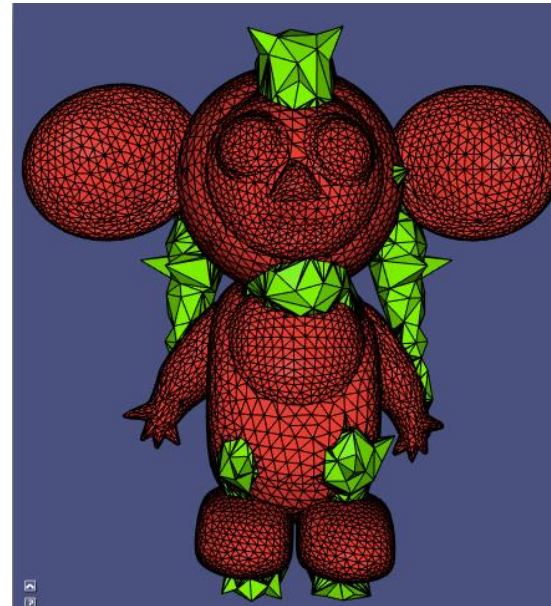
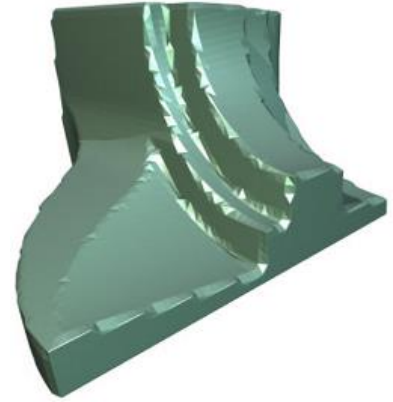
Feature Sensitive Surface Extraction from Volume Data [Kobbelt SIGGRAPH01]

Dual Contouring of Hermite Data [Ju SIGGRAPH02]

<http://www.graphics.rwth-aachen.de/IsoEx/>

CSG with surface representation only

- Volumetric representation (=isosurface extraction using MC)
 - ➔ Approximation accuracy depends on grid resolution ☹️
- CSG with surface representation only
 - ➔ Exactly keep original mesh geometry 😊
- Difficult to implement robust & efficient ☹️
 - Floating point error
 - Exactly coplanar faces
- Notable advances in recent years



Fast, exact, linear booleans [Bernstein SGP09]

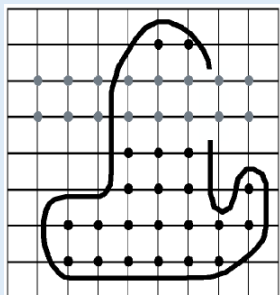
Exact and Robust (Self-)Intersections for Polygonal Meshes [Campen EG10]

Mesh Arrangements for Solid Geometry [Zhou SIGGRAPH16]

<https://libigl.github.io/libigl/tutorial/tutorial.html#booleanoperationsonmeshes>

Mesh repair

Volumetric representation

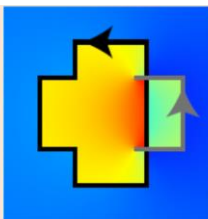


Decide inside/outside by shooting rays from outside

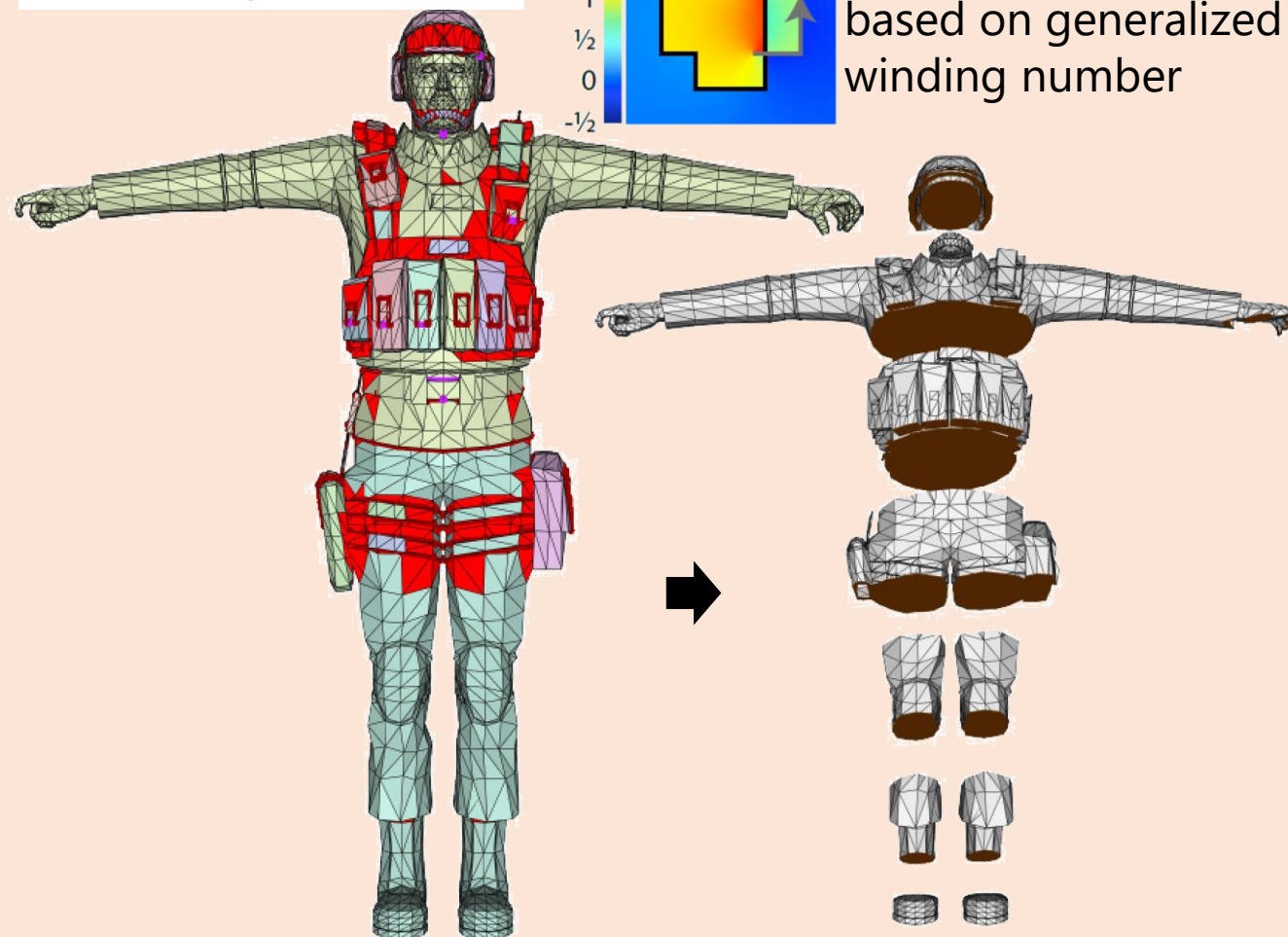


Surface representation

2
1½
1
½
0
-½

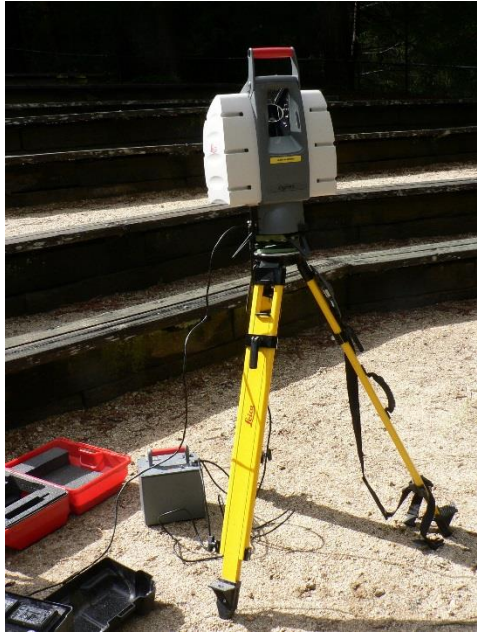


Decide inside/outside based on generalized winding number



Surface reconstruction from point cloud

Measuring 3D shapes



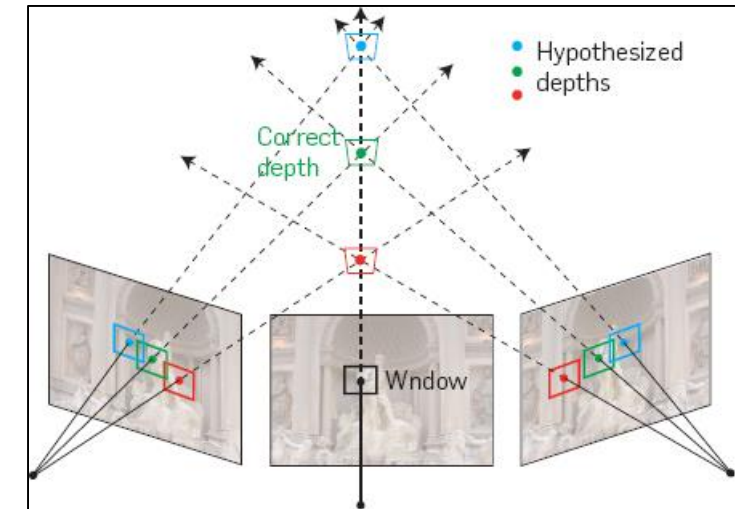
Range Scanner
(LIDAR)



Structured Light



Depth Camera



Multi-View Stereo

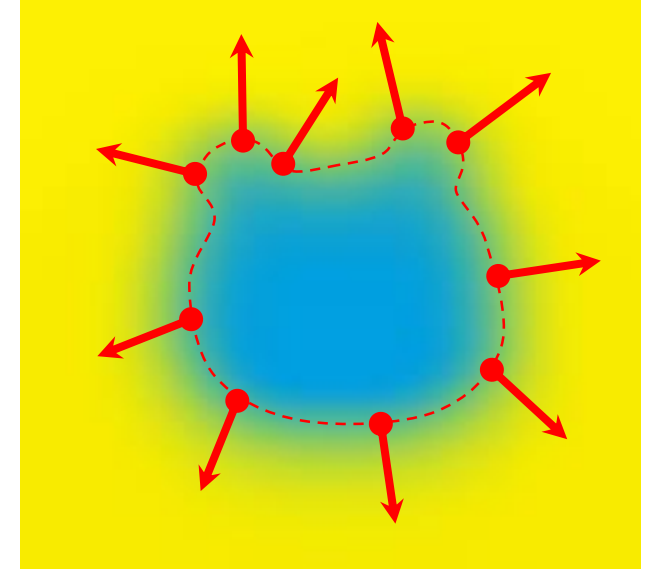
- Obtained data: point cloud
 - 3D coordinate
 - Normal (surface orientation)

- Normals not available? → Normal estimation
- Too noisy? → Denoising

} Typical Computer Vision problems

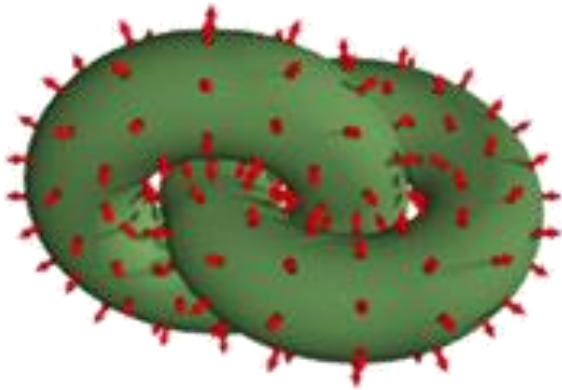
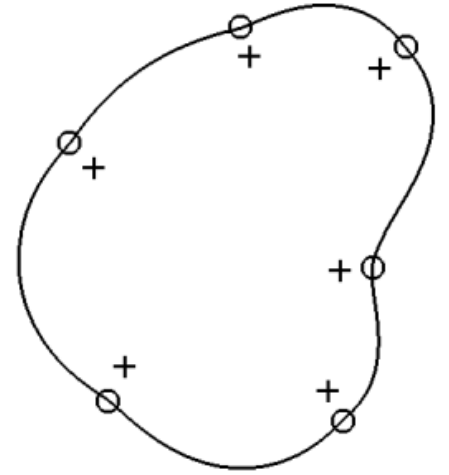
Surface reconstruction from point cloud

- Input: N points
 - Coordinate $\mathbf{x}_i = (x_i, y_i, z_i)$ & normal $\mathbf{n}_i = (n_i^x, n_i^y, n_i^z)$, $i \in \{1, \dots, N\}$
- Output: function $f(\mathbf{x})$ satisfying value & gradient constraints
 - $f(\mathbf{x}_i) = f_i$
 - $\nabla f(\mathbf{x}_i) = \mathbf{n}_i$
 - Zero isosurface $f(\mathbf{x}) = 0 \rightarrow$ output surface
- "Scattered Data Interpolation"
 - **M**oving **L**east **S**quares
 - **R**adial **B**asis **F**unction
 - Important to other fields (e.g. Machine Learning) as well

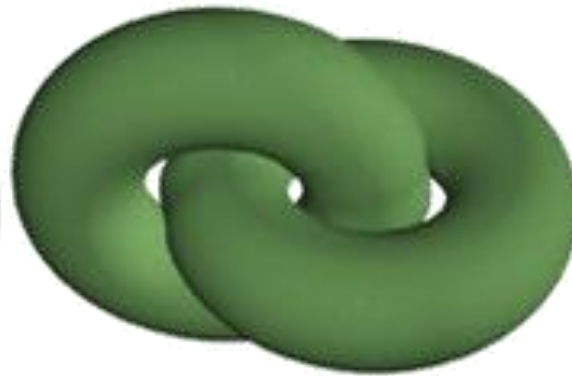


Two ways for controlling gradients

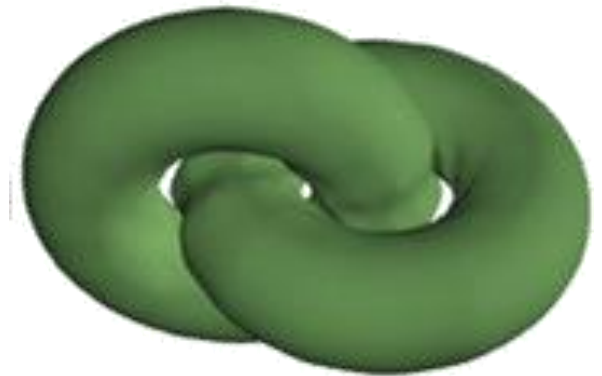
- Additional value constraints at offset locations
 - Simple
- Directly include gradient constraint in the mathematical formulation (Hermite interpolation)
 - High-quality



Value+gradient constraints



Hermite interpolation



Simple offsetting

Interpolation using **M**oving **L**east **S**quares

Starting point: Least **S**quares

- For now, assume the function as linear: $f(\mathbf{x}) = ax + by + cz + d$

- Unknowns: a, b, c, d

$$\mathbf{x} := (x, y, z)$$

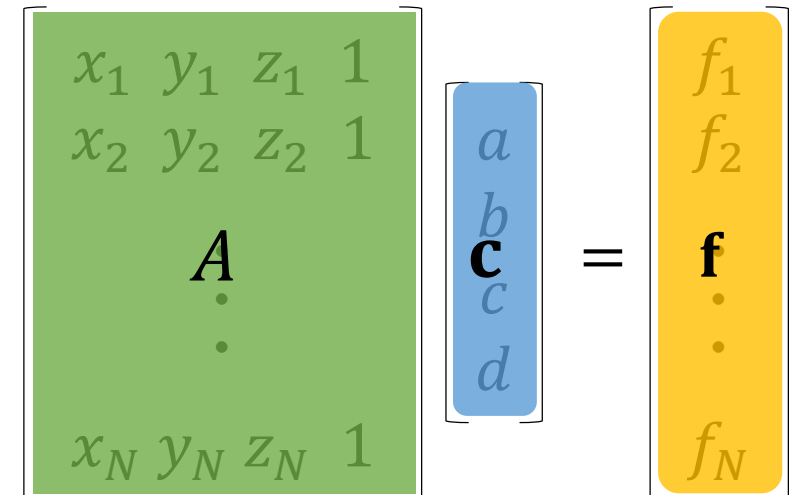
- Value constraints at data points

$$f(\mathbf{x}_1) = ax_1 + by_1 + cz_1 + d = f_1$$

$$f(\mathbf{x}_2) = ax_2 + by_2 + cz_2 + d = f_2$$

\vdots

$$f(\mathbf{x}_N) = ax_N + by_N + cz_N + d = f_N$$

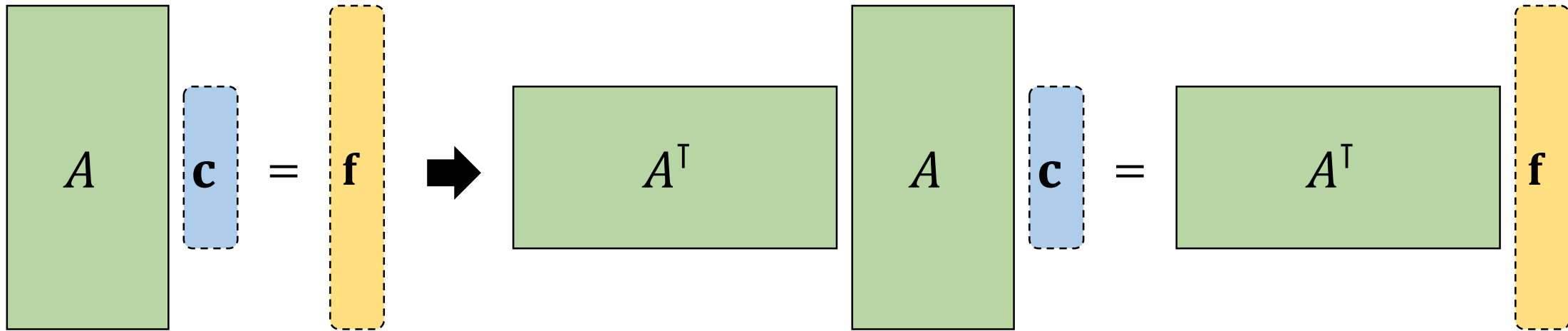


The diagram illustrates the matrix equation $A\mathbf{c} = \mathbf{f}$. On the left, a green matrix A is shown with rows $[x_1 \ y_1 \ z_1 \ 1]$, $[x_2 \ y_2 \ z_2 \ 1]$, and $[x_N \ y_N \ z_N \ 1]$, with a vertical ellipsis between the second and last rows. To its right is a blue column vector \mathbf{c} containing the unknowns a , b , c , and d . An equals sign follows, and to the right is a yellow column vector \mathbf{f} containing the values f_1 , f_2 , and f_N , with a vertical ellipsis between f_2 and f_N .

- (Forget about gradient constraints for now)

Overconstrained System

- #unknowns < #constraints (i.e. taller matrix)
→ cannot exactly satisfy all the constraints



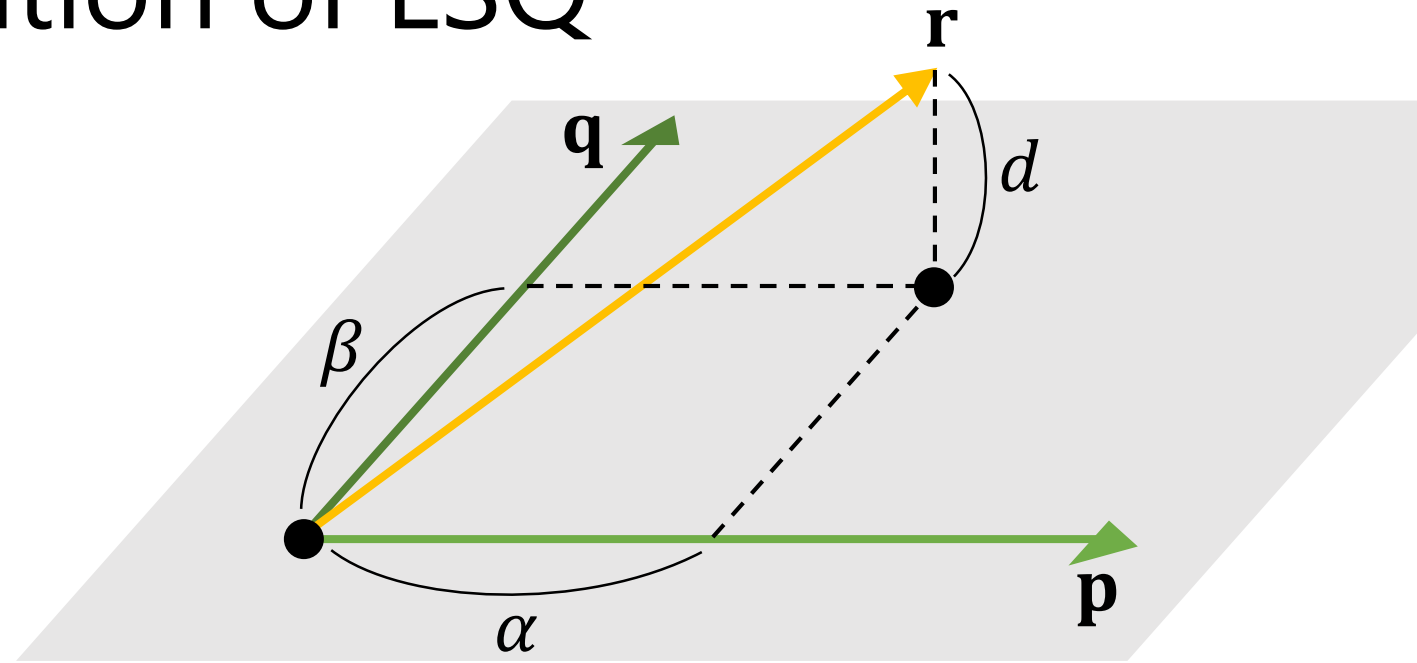
- Minimizing fitting error

$$\|A \mathbf{c} - \mathbf{f}\|^2 = \sum_{i=1}^N \|f(\mathbf{x}_i) - f_i\|^2$$

$$\mathbf{c} = (A^T A)^{-1} A^T \mathbf{f}$$

Geometric interpretation of LSQ

$$\begin{bmatrix} p_x & q_x \\ p_y & q_y \\ p_z & q_z \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}$$



- Project \mathbf{r} onto a plane spanned by \mathbf{p} & \mathbf{q}
 - Fitting error = projection distance

$$d^2 = \|\alpha\mathbf{p} + \beta\mathbf{q} - \mathbf{r}\|^2$$

Weighted Least Squares

- Each data point is weighted by w_i
 - Importance, confidence, ...

- Minimize the following fitting error:

$$\sum_{i=1}^N \|w_i(f(\mathbf{x}_i) - f_i)\|^2$$

The diagram illustrates the matrix formulation of the Weighted Least Squares problem. It shows the equation $W A \mathbf{c} = W \mathbf{f}$, where:

- W (purple square) is a diagonal matrix of weights w_1, w_2, \dots, w_N .
- A (green square) is the design matrix with rows $[x_1 \ y_1 \ z_1 \ 1], [x_2 \ y_2 \ z_2 \ 1], \dots, [x_N \ y_N \ z_N \ 1]$.
- \mathbf{c} (blue column) is the vector of parameters $[a \ b \ c \ d]^T$.
- \mathbf{f} (yellow column) is the vector of observed values $[f_1 \ f_2 \ \dots \ f_N]^T$.

Weighted Least Squares

The diagram illustrates the Weighted Least Squares (WLS) problem and its solution. It is divided into two parts by a large black arrow pointing from left to right.

Top part: Shows the WLS equation. On the left, a large purple square labeled W is followed by a green rectangle labeled A , which is then followed by a small blue dashed rectangle labeled \mathbf{c} . An equals sign follows. To the right of the equals sign is another large purple square labeled W , followed by a tall yellow dashed rectangle labeled \mathbf{f} .

Bottom part: Shows the solution for \mathbf{c} . A large black arrow points from the left to a small blue dashed rectangle labeled \mathbf{c} . This is followed by an equals sign. To the right of the equals sign are two green rectangles: the first is labeled $(A^T W^2 A)^{-1}$ and the second is labeled $A^T W^2$. These two green rectangles are followed by a tall yellow dashed rectangle labeled \mathbf{f} .

Moving Least Squares

- Weight w_i is a function of evaluation point \mathbf{x} :

$$w_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|)$$

- Popular choices for the function (kernel):

- $w(r) = e^{-r^2/\sigma^2}$

- $w(r) = \frac{1}{r^2 + \epsilon^2}$

Larger the weight as \mathbf{x} is closer to \mathbf{x}_i

- Weighting matrix W is a function of \mathbf{x}

→ Coeffs a, b, c, d are functions of \mathbf{x}

$$f(\mathbf{x}) = [x \ y \ z \ 1] \begin{bmatrix} a(\mathbf{x}) \\ b(\mathbf{x}) \\ c(\mathbf{x}) \\ d(\mathbf{x}) \end{bmatrix} (A^T W(\mathbf{x})^2 A)^{-1} A^T W(\mathbf{x})^2 \mathbf{f}$$

Introducing gradient (normal) constraints

- Consider linear function represented by each data point:

$$g_i(\mathbf{x}) = f_i + (\mathbf{x} - \mathbf{x}_i)^\top \mathbf{n}_i$$

- Minimize fitting error to each g_i evaluated at \mathbf{x} :

$$\sum_{i=1}^N \|w_i(\mathbf{x})(f(\mathbf{x}) - g_i(\mathbf{x}))\|^2$$

$$\begin{bmatrix} w_1(\mathbf{x}) & & & \\ & w_2(\mathbf{x}) & & \\ & & \ddots & \\ & & & w_N(\mathbf{x}) \end{bmatrix} \begin{bmatrix} x & y & z & 1 \\ x & y & z & 1 \\ & \vdots & & \\ & & & x & y & z & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} w_1(\mathbf{x}) & & & \\ & w_2(\mathbf{x}) & & \\ & & \ddots & \\ & & & w_N(\mathbf{x}) \end{bmatrix} \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_N(\mathbf{x}) \end{bmatrix}$$

Introducing gradient (normal) constraints



Normal constraints



Simple offsetting



Input : Polygon Soup

Interpolation

Approximation 1

Approximation 2

Approximation 3

Interpolation using **Radial Basis Functions**

Basic idea

- Define $f(\mathbf{x})$ as weighted sum of basis functions $\phi(\mathbf{x})$:

$$f(\mathbf{x}) = \sum_{i=1}^N \underset{\text{Unknown}}{w_i} \phi(\mathbf{x} - \mathbf{x}_i)$$

Basis function translated to each data point \mathbf{x}_i

- **Radial Basis Function** $\phi(\mathbf{x})$: only depends on the length of \mathbf{x}
 - $\phi(\mathbf{x}) = e^{-\|\mathbf{x}\|^2/\sigma^2}$ (Gaussian)
 - $\phi(\mathbf{x}) = \frac{1}{\sqrt{\|\mathbf{x}\|^2 + c^2}}$ (Inverse Multiquadric)
- Determine weights w_i from constraints at data points $f(\mathbf{x}_i) = f_i$

Basic idea

Notation: $\phi_{i,j} = \phi(\mathbf{x}_i - \mathbf{x}_j)$

$$f(\mathbf{x}_1) = w_1 \phi_{1,1} + w_2 \phi_{1,2} + \cdots + w_N \phi_{1,N} = f_1$$

$$f(\mathbf{x}_2) = w_1 \phi_{2,1} + w_2 \phi_{2,2} + \cdots + w_N \phi_{2,N} = f_2$$

•
•
•

$$f(\mathbf{x}_N) = w_1 \phi_{N,1} + w_2 \phi_{N,2} + \cdots + w_N \phi_{N,N} = f_N$$

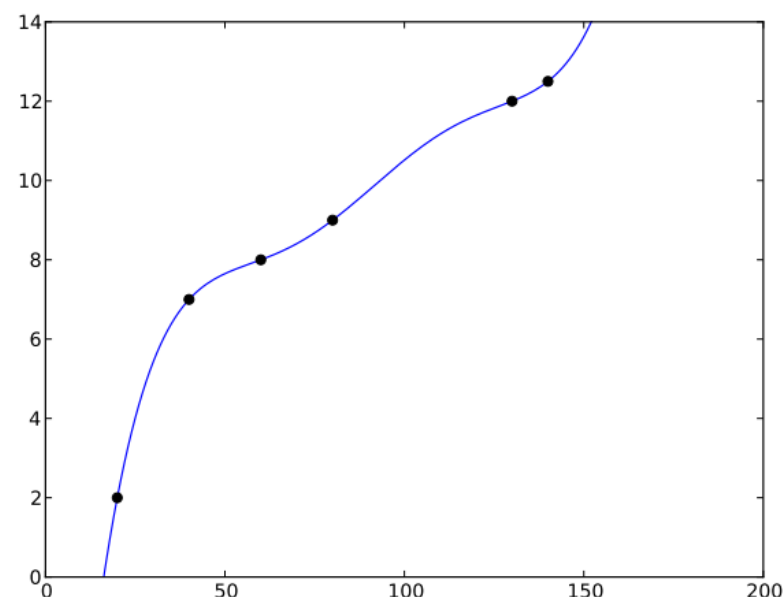
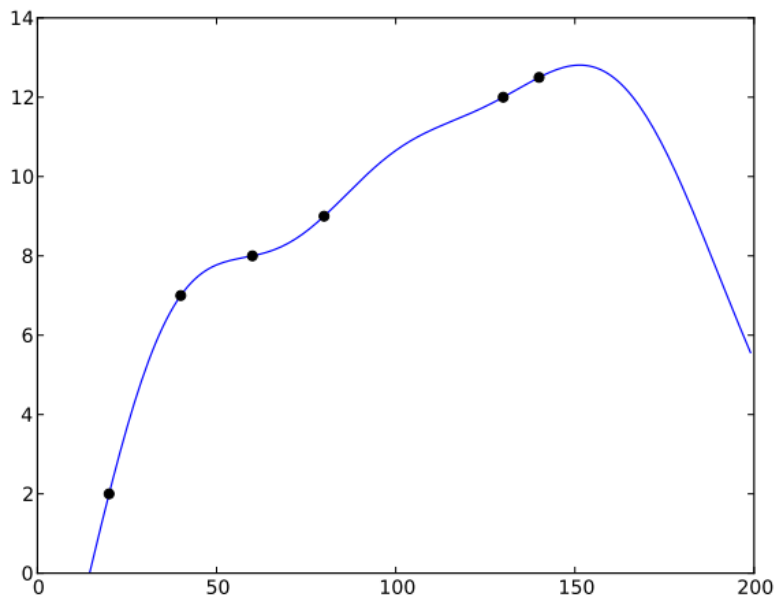
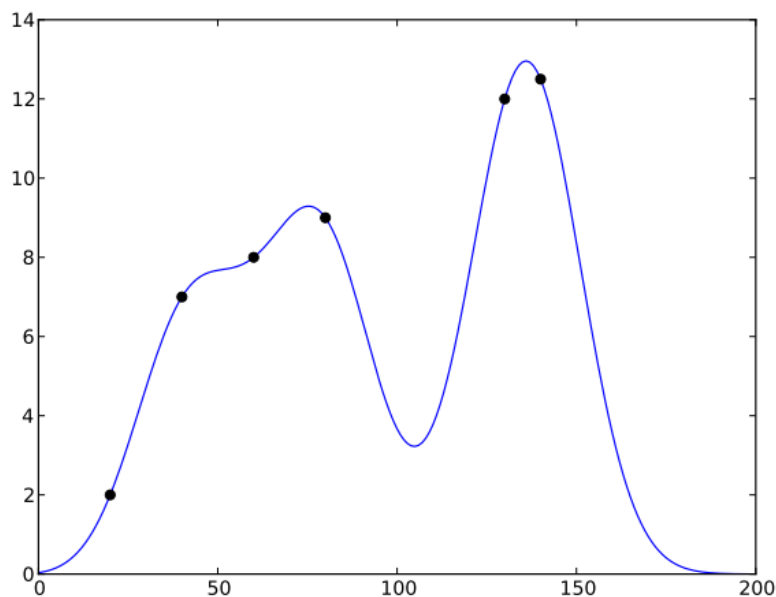
A diagram showing a matrix equation. On the left is a green square matrix Φ with elements $\phi_{1,1}, \phi_{1,2}, \phi_{1,N}$ in the first row, $\phi_{2,1}, \phi_{2,2}, \phi_{2,N}$ in the second row, and $\phi_{N,1}, \phi_{N,2}, \phi_{N,N}$ in the last row. To its right is a blue vertical vector \mathbf{w} with elements w_1, w_2, \dots, w_N . These are followed by an equals sign and a yellow vertical vector \mathbf{f} with elements f_1, f_2, \dots, f_N .

Solve this!

When using Gaussian RBF

$$\phi(\mathbf{x}) = e^{-\|\mathbf{x}\|^2/\sigma^2}$$

- Results highly dependent on the choice of parameter σ ☹️



σ —————→
小 大

- How to obtain the as-smooth-as-possible result?

Measuring function's "bend": Thin-Plate Energy

- 2nd derivative (=curvature) magnitude integrated over the whole domain

$$E_2[f] = \int_{\mathbf{x} \in \mathbb{R}^d} \|\Delta f(\mathbf{x})\|^2 d\mathbf{x}$$

Laplacian operator

- 1D case:

$$E_2[f] = \int_{x \in \mathbb{R}} f''(x)^2 dx$$

- 2D case:

$$E_2[f] = \int_{\mathbf{x} \in \mathbb{R}^2} (f_{xx}(\mathbf{x})^2 + 2f_{xy}(\mathbf{x})^2 + f_{yy}(\mathbf{x})^2) d\mathbf{x}$$

- 3D case:

$$E_2[f] = \int_{\mathbf{x} \in \mathbb{R}^3} (f_{xx}(\mathbf{x})^2 + f_{yy}(\mathbf{x})^2 + f_{zz}(\mathbf{x})^2 + 2f_{xy}(\mathbf{x})^2 + 2f_{yz}(\mathbf{x})^2 + 2f_{zx}(\mathbf{x})^2) d\mathbf{x}$$

Known theory in the math literature

- Of all functions satisfying $\{ f(\mathbf{x}_i) = f_i \}$, the minimizer of E_2 is represented as RBFs with the following basis:
 - 1D case: $\phi(x) = |x|^3$
 - 2D case: $\phi(\mathbf{x}) = \|\mathbf{x}\|^2 \log \|\mathbf{x}\|$
 - 3D case: $\phi(\mathbf{x}) = \|\mathbf{x}\|$
- FYI
 - Finite Element Method: Find f minimizing E_2 discretized over mesh
 - RBF: Find f minimizing E_2 analytically

Additional linear term

- $E_2[f]$ is defined using 2nd derivative
→ Any additional linear term $p(\mathbf{x}) = ax + by + cz + d$ has no effect:

$$E_2[f + p] = E_2[f]$$

- Make f unique by regarding linear term as additional unknowns:

$$f(\mathbf{x}) = \sum_{i=1}^N w_i \phi(\mathbf{x} - \mathbf{x}_i) + ax + by + cz + d$$

With linear term

$$f(\mathbf{x}_1) = w_1\phi_{1,1} + w_2\phi_{1,2} + \cdots + w_N\phi_{1,N} + ax_1 + by_1 + cz_1 + d = f_1$$

$$f(\mathbf{x}_2) = w_1\phi_{2,1} + w_2\phi_{2,2} + \cdots + w_N\phi_{2,N} + ax_2 + by_2 + cz_2 + d = f_2$$

•
•
•

$$f(\mathbf{x}_N) = w_1\phi_{N,1} + w_2\phi_{N,2} + \cdots + w_N\phi_{N,N} + ax_N + by_N + cz_N + d = f_N$$

$$\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,N} & x_1 & y_1 & z_1 & 1 \\ \phi_{2,1} & \phi_{2,2} & \phi_{2,N} & x_2 & y_2 & z_2 & 1 \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ \phi_{N,1} & \phi_{N,2} & \phi_{N,N} & x_N & y_N & z_N & 1 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \\ a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}$$

Φ \mathbf{P} \mathbf{w} \mathbf{c} \mathbf{f}

4 unknowns a, b, c, d
added \rightarrow 4 new
constraints needed

Additional constraints: reproduction of all linear functions

- “If all data points (\mathbf{x}_i, f_i) are sampled from a linear function, RBF should reproduce the original function”
- Additional constraints:
 - $\sum_{i=1}^N w_i = 0$
 - $\sum_{i=1}^N x_i w_i = 0$
 - $\sum_{i=1}^N y_i w_i = 0$
 - $\sum_{i=1}^N z_i w_i = 0$
- Makes the matrix symmetric

$$\begin{bmatrix}
 \begin{matrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,N} \\ \phi_{2,1} & \phi_{2,2} & \phi_{2,N} \\ & \Phi & \\ & & \ddots \\ \phi_{N,1} & \phi_{N,2} & \phi_{N,N} \end{matrix} &
 \begin{matrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ & P & \\ & & \ddots \\ x_N & y_N & z_N & 1 \end{matrix} \\
 \begin{matrix} x_1 & x_2 & x_N \\ y_1 & y_2 & y_N \\ z_1 & z_2 & z_N \\ 1 & 1 & 1 \end{matrix} &
 \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}
 \end{bmatrix}
 \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \\ a \\ b \\ c \\ d \end{bmatrix}
 =
 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Introducing gradient constraints

- Introduce weighted sum of basis' gradient $\nabla\phi$:

$$f(\mathbf{x}) = \sum_{i=1}^N \{ \mathbf{w}_i \phi(\mathbf{x} - \mathbf{x}_i) + \mathbf{v}_i^T \nabla \phi(\mathbf{x} - \mathbf{x}_i) \} + ax + by + cz + d$$

Unknown 3D vector

- Gradient of f :

$$\nabla f(\mathbf{x}) = \sum_{i=1}^N \{ \mathbf{w}_i \nabla \phi(\mathbf{x} - \mathbf{x}_i) + \mathbf{H}_\phi(\mathbf{x} - \mathbf{x}_i) \mathbf{v}_i \} + \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

- Incorporate gradient constraints $\nabla f(\mathbf{x}_i) = \mathbf{n}_i$

$$\mathbf{H}_\phi(\mathbf{x}) = \begin{pmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} \\ \phi_{yx} & \phi_{yy} & \phi_{yz} \\ \phi_{zx} & \phi_{zy} & \phi_{zz} \end{pmatrix}$$

Hessian matrix (function) 41

Introducing gradient constraints

- 1st data point:

Value constraint:

$$f(\mathbf{x}_1) = w_1 \phi_{1,1} + \mathbf{v}_1^\top \nabla \phi_{1,1} + w_2 \phi_{1,2} + \mathbf{v}_2^\top \nabla \phi_{1,2} + \cdots + w_N \phi_{1,N} + \mathbf{v}_N^\top \nabla \phi_{1,N}$$

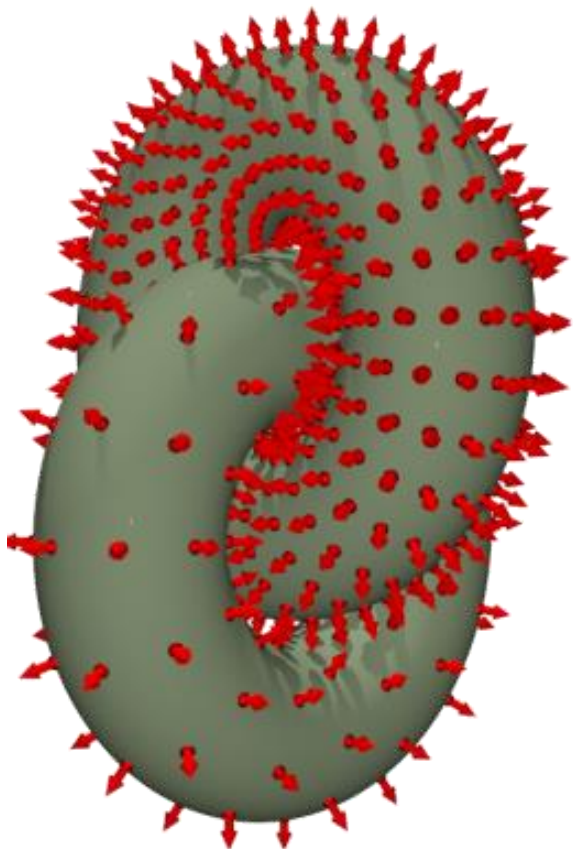
Gradient constraint:

$$\nabla f(\mathbf{x}_1) = w_1 \nabla \phi_{1,1} + H_{\phi}^{1,1} \mathbf{v}_1 + w_2 \nabla \phi_{1,2} + H_{\phi}^{1,2} \mathbf{v}_2 + \dots + w_N \nabla \phi_{1,N} + H_{\phi}^{1,N} \mathbf{v}_N$$

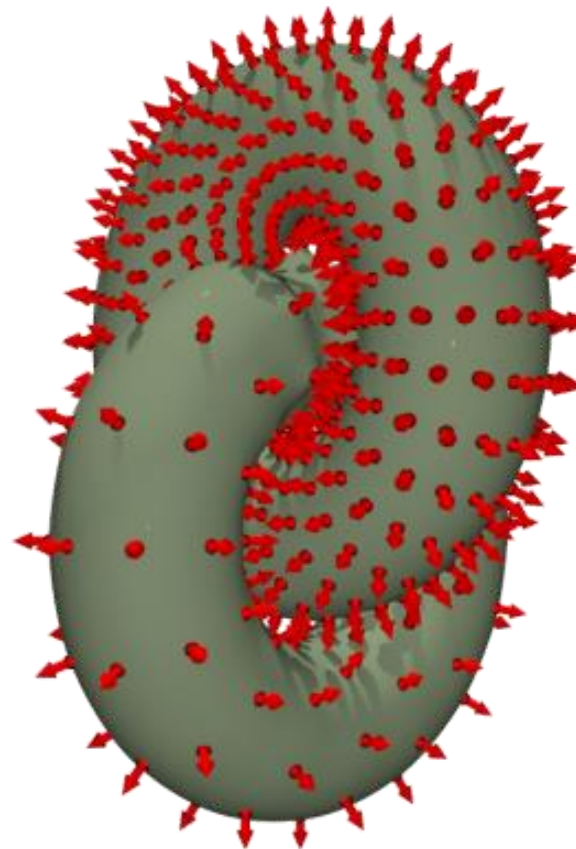
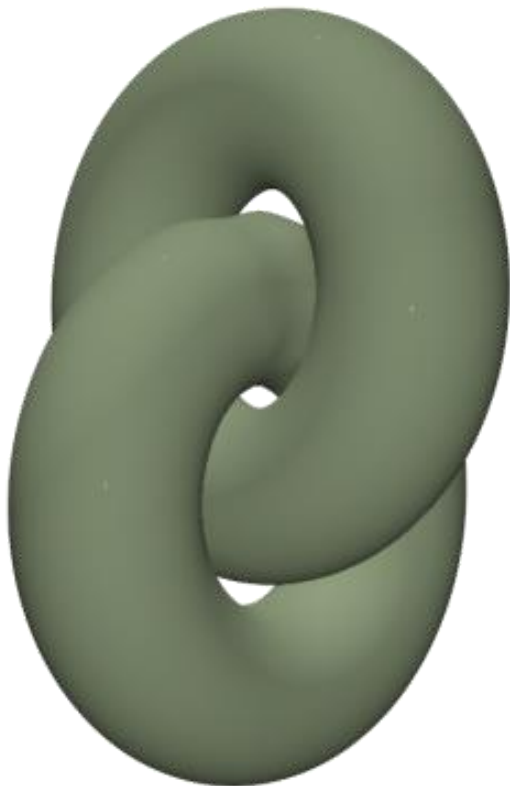
Radial Basis Functions Implicits [Macedo CGF10]

$$\begin{bmatrix}
 \begin{bmatrix} \Phi_{1,1} & \Phi_{1,2} \end{bmatrix} & \dots & \begin{bmatrix} \Phi_{1,N} \end{bmatrix} & \begin{bmatrix} P_1 \end{bmatrix} \\
 \begin{bmatrix} \Phi_{2,1} & \Phi_{2,2} \end{bmatrix} & \dots & \begin{bmatrix} \Phi_{2,N} \end{bmatrix} & \begin{bmatrix} P_2 \end{bmatrix} \\
 \vdots & \ddots & \vdots & \vdots \\
 \begin{bmatrix} \Phi_{N,1} & \Phi_{N,2} \end{bmatrix} & \dots & \begin{bmatrix} \Phi_{N,N} \end{bmatrix} & \begin{bmatrix} P_N \end{bmatrix} \\
 \begin{bmatrix} P_1^\top & P_2^\top \end{bmatrix} & \dots & \begin{bmatrix} P_N^\top \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
 \end{bmatrix}
 \begin{bmatrix}
 \begin{bmatrix} w_1 \\ \mathbf{v}_1 \end{bmatrix} \\
 \begin{bmatrix} w_2 \\ \mathbf{v}_2 \end{bmatrix} \\
 \vdots \\
 \begin{bmatrix} w_N \\ \mathbf{v}_N \end{bmatrix} \\
 \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}
 \end{bmatrix}
 =
 \begin{bmatrix}
 \begin{bmatrix} f_1 \\ \mathbf{n}_1 \end{bmatrix} \\
 \begin{bmatrix} f_2 \\ \mathbf{n}_2 \end{bmatrix} \\
 \vdots \\
 \begin{bmatrix} f_N \\ \mathbf{n}_N \end{bmatrix} \\
 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
 \end{bmatrix}$$

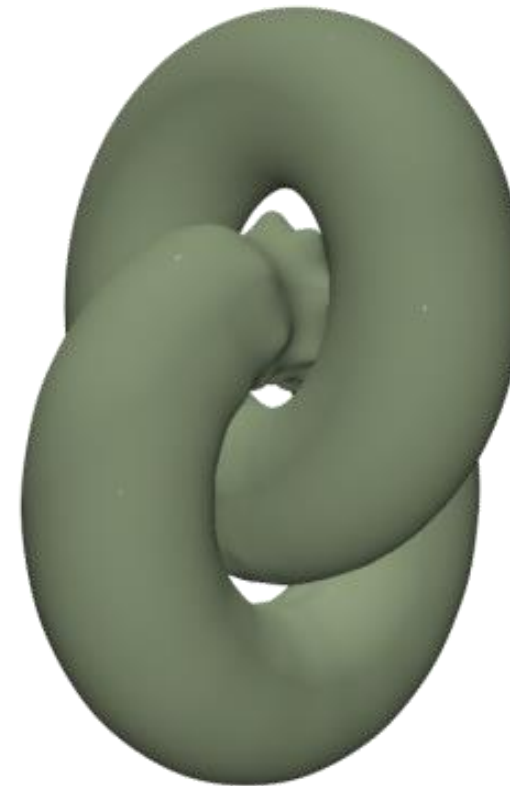
Comparison



Gradient constraints



Simple offsetting with
value constraints only



References

- State of the Art in Surface Reconstruction from Point Clouds [Berger EG14 STAR]
- A survey of methods for moving least squares surfaces [Cheng PBG08]
- Scattered Data Interpolation for Computer Graphics [Anjyo SIGGRAPH14 Course]
- An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares for scattered data approximation and interpolation [Nealen TechRep04]

References

- http://en.wikipedia.org/wiki/Implicit_surface
- http://en.wikipedia.org/wiki/Radial_basis_function
- http://en.wikipedia.org/wiki/Thin_plate_spline
- http://en.wikipedia.org/wiki/Polyharmonic_spline