Introduction to Computer Graphics

– Modeling (3) –

May 6, 2021

Kenshi Takayama

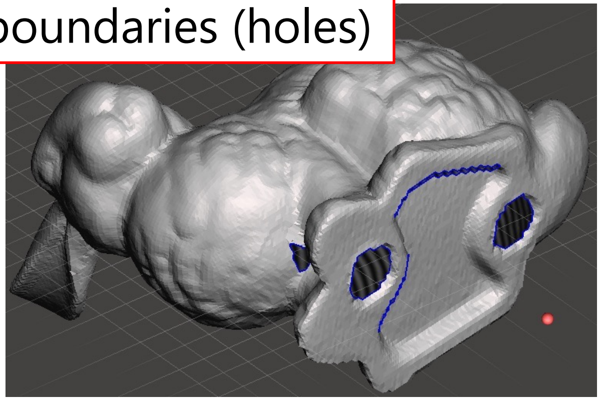
Solid modeling

Solid models

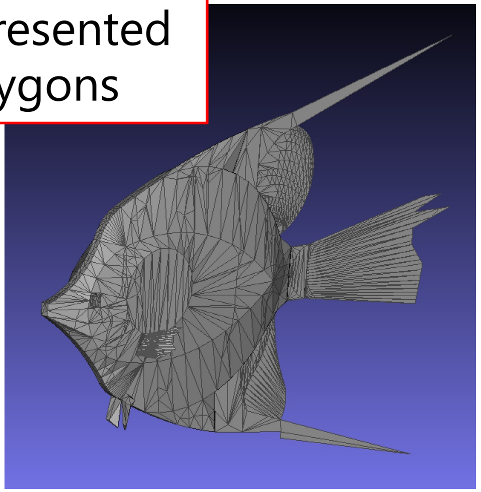
- Clear definition of “inside” & “outside” at any 3D point

Non-solid cases

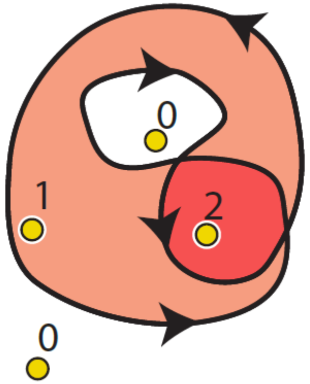
Open boundaries (holes)



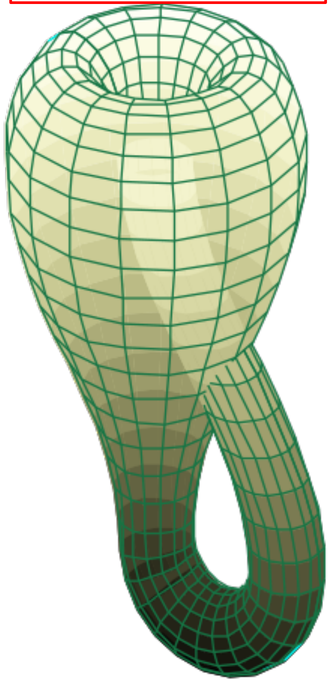
Thin shapes represented by single polygons



Self-intersections



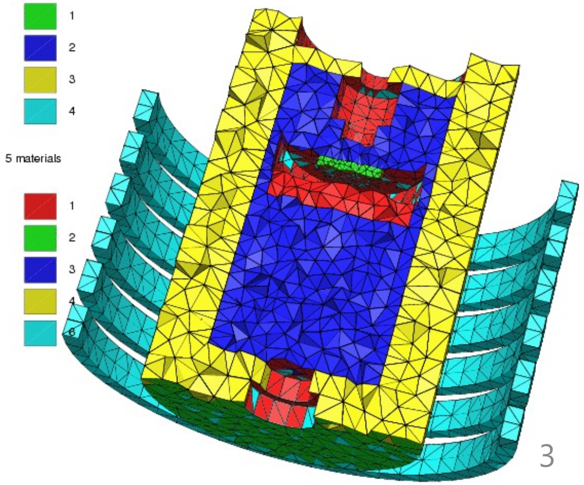
Unorientable



Klein bottle

- 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

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

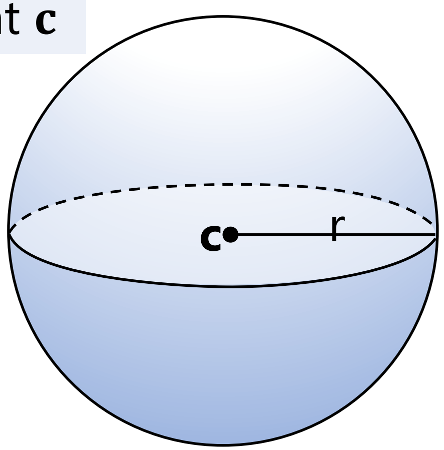
- The whole interior of the model:

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

- Examples:

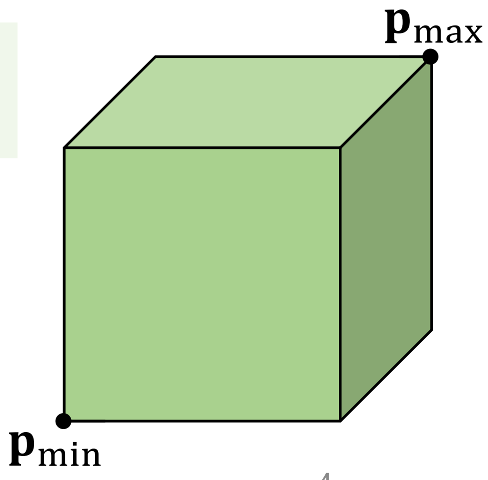
Sphere of radius r centered at $\mathbf{c}$

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



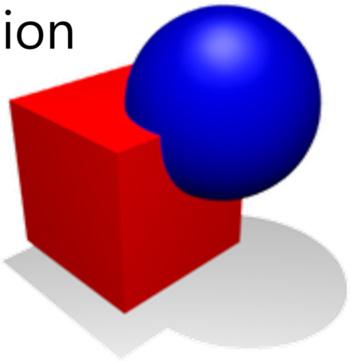
Box whose min & max corners are $(x_{\min}, y_{\min}, z_{\min})$ & $(x_{\max}, y_{\max}, z_{\max})$

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



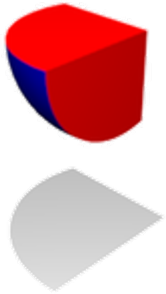
Constructive **S**olid **G**eometry (Boolean operations)

Union



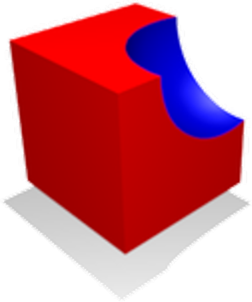
$$f_{A\cup B}(\mathbf{p}) \coloneqq f_A(\mathbf{p}) \vee f_B(\mathbf{p})$$

Intersection



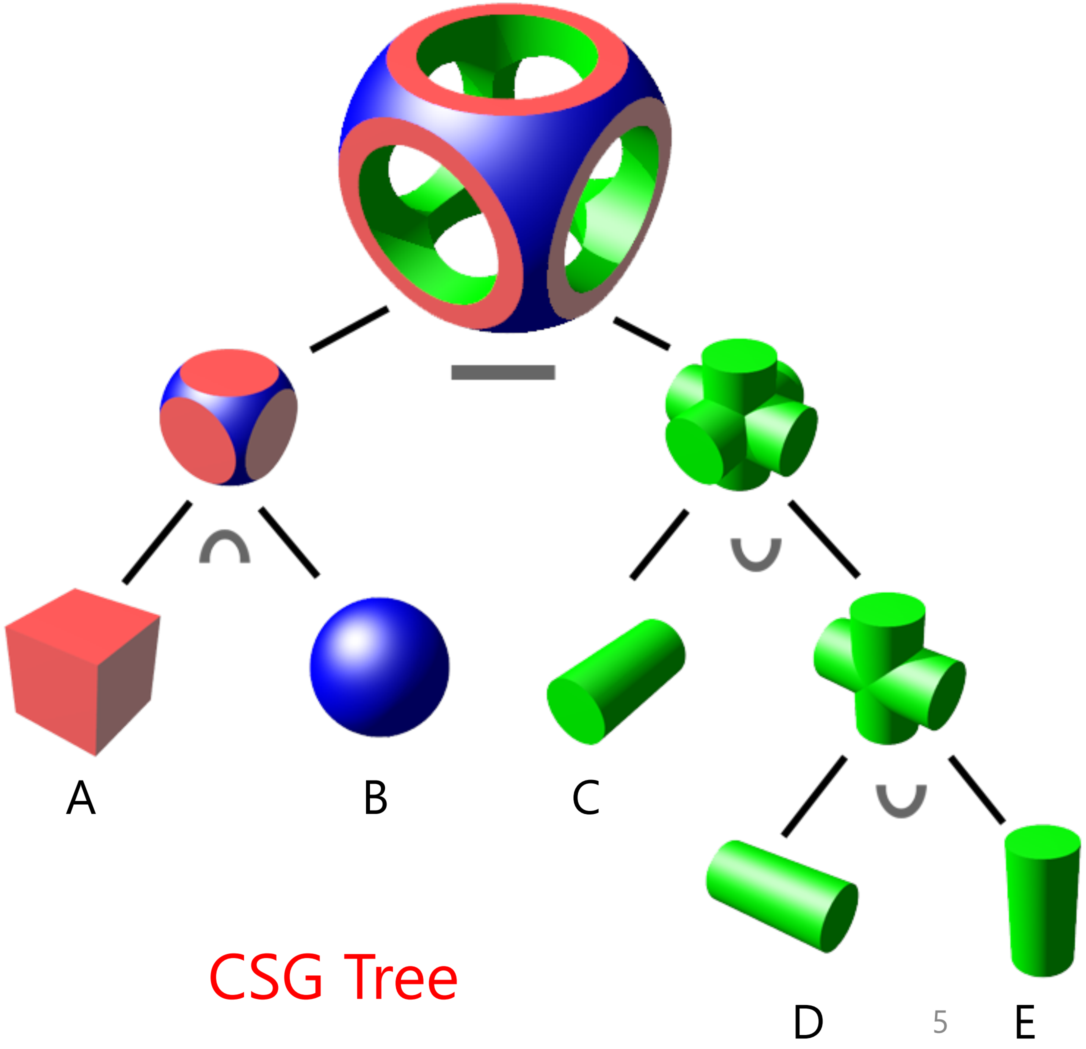
$$f_{A\cap B}(\mathbf{p}) \coloneqq f_A(\mathbf{p}) \wedge f_B(\mathbf{p})$$

Subtraction



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

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



CSG Tree

Solid model represented by **S**inged **D**istance **F**ield

- Shortest distance from 3D point to model surface:

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

  - Signed: positive ➔ outside, negative ➔ inside

- Corresponding predicate describing the solid:

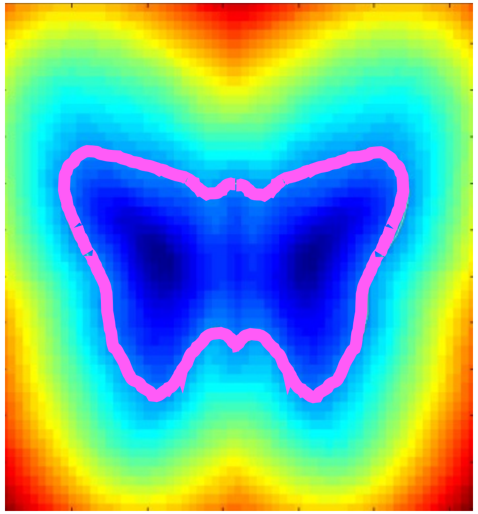
$$f(\mathbf{p}) \coloneqq d(\mathbf{p}) < 0$$

- Zero isosurface ➔ model surface:

$$\{\mathbf{p} \mid d(\mathbf{p}) = 0\} \subset \mathbb{R}^3$$

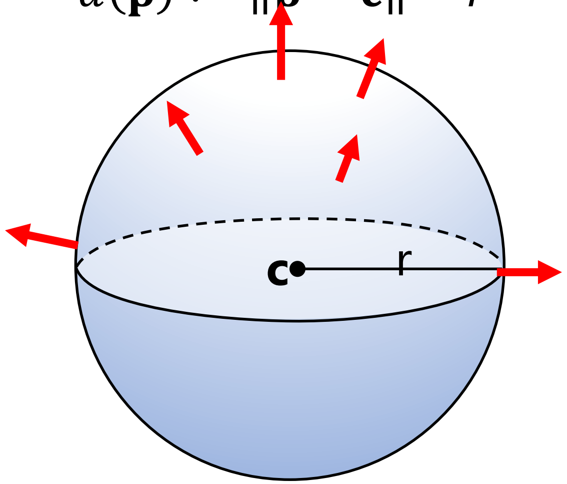
- Aka. "implicit" or "volumetric" representation

- Isosurface normal agrees with direction of gradient $\nabla d(\mathbf{p})$



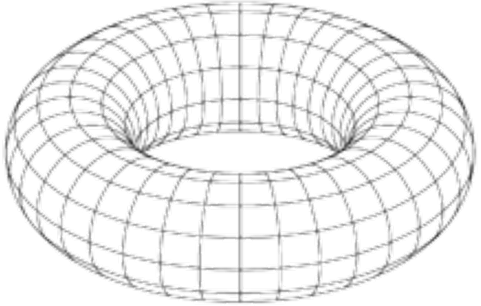
Sphere of radius r centered at $\mathbf{c}$

$$d(\mathbf{p}) \coloneqq \|\mathbf{p} - \mathbf{c}\| - r$$



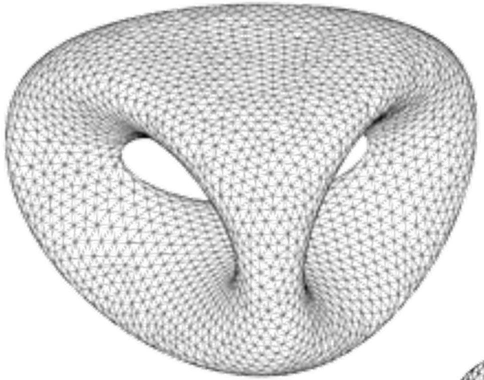
Examples of implicit functions

Not necessarily distance functions

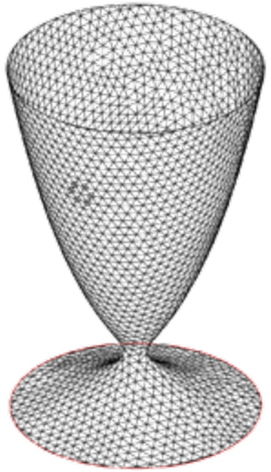


Torus with major & minor radii R & a

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



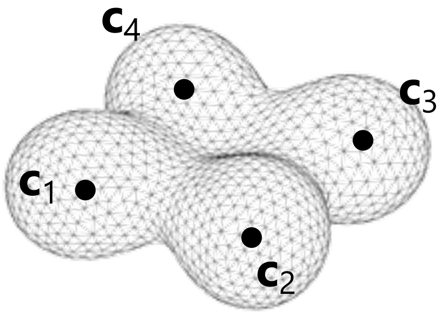
$$2y(y^2 - 3x^2)(1 - z^2) + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2) = 0$$



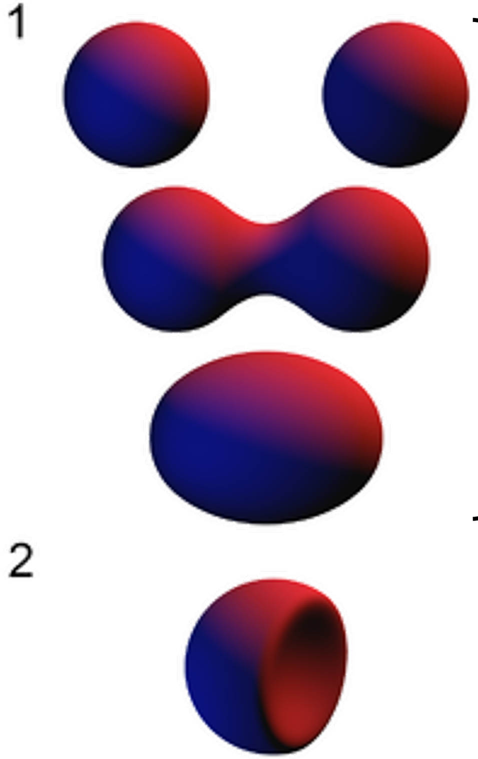
$$x^2 + y^2 - (\ln(z + 3.2))^2 - 0.02 = 0$$

Examples of implicit functions: Metaballs

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



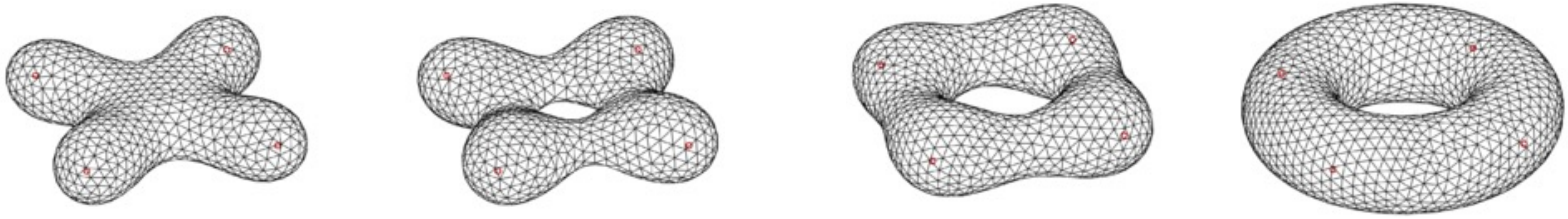
$$d(\mathbf{p}) = d_1(\mathbf{p}) + d_2(\mathbf{p}) + d_3(\mathbf{p}) + d_4(\mathbf{p})$$



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

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

Morphing by interpolating implicit functions



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

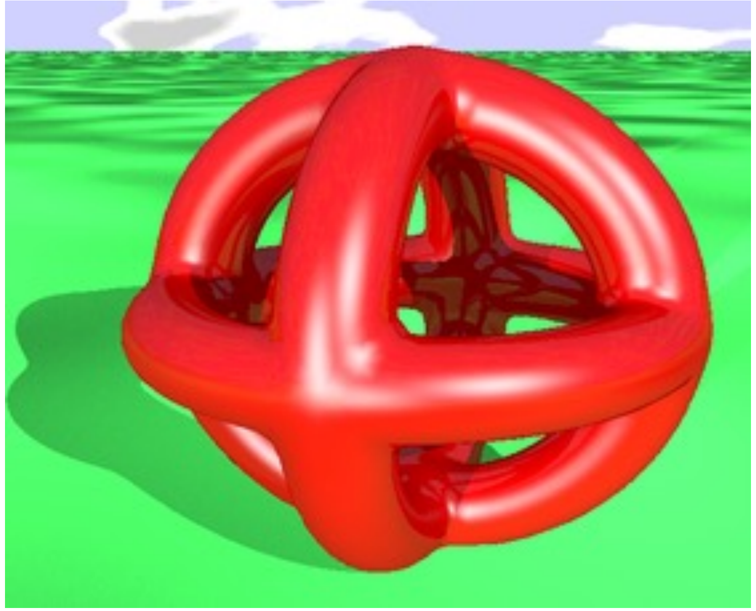
Modeling by combining implicit functions

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

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

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

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

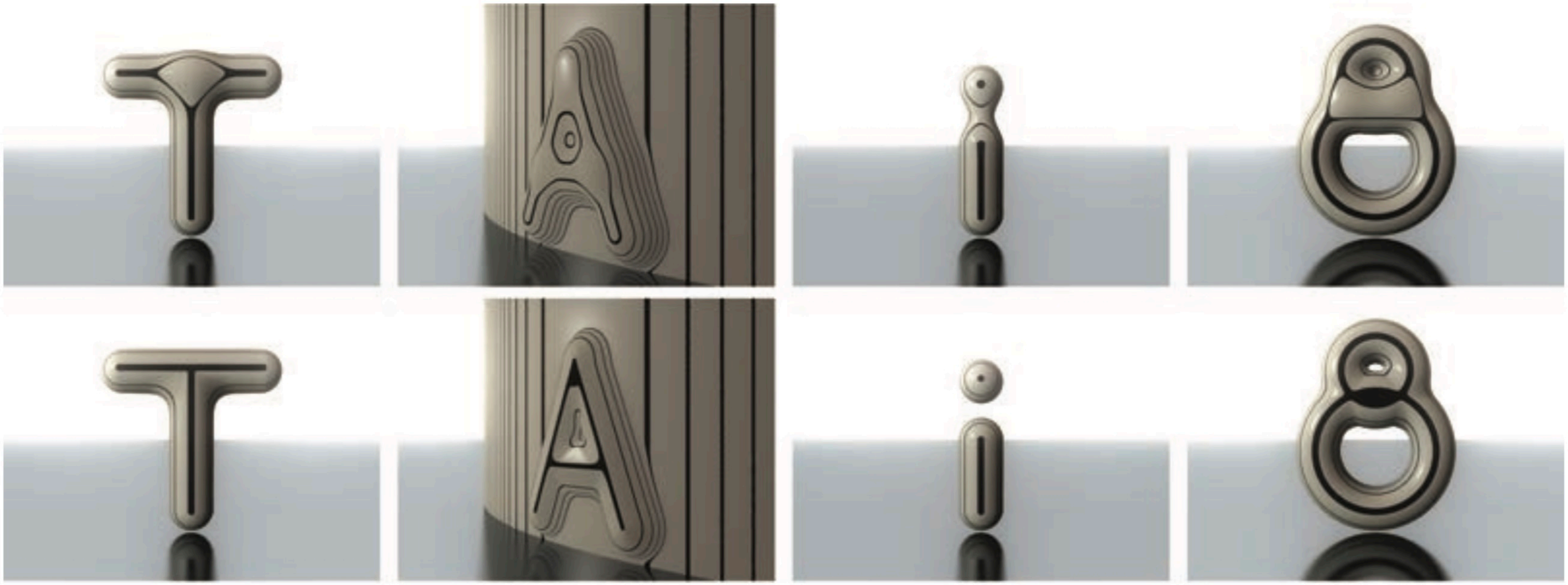


More advanced blending

- When blending two implicit functions, consider their gradie directions and choose different blending accordingly

Traditional (simple sum)

Proposed

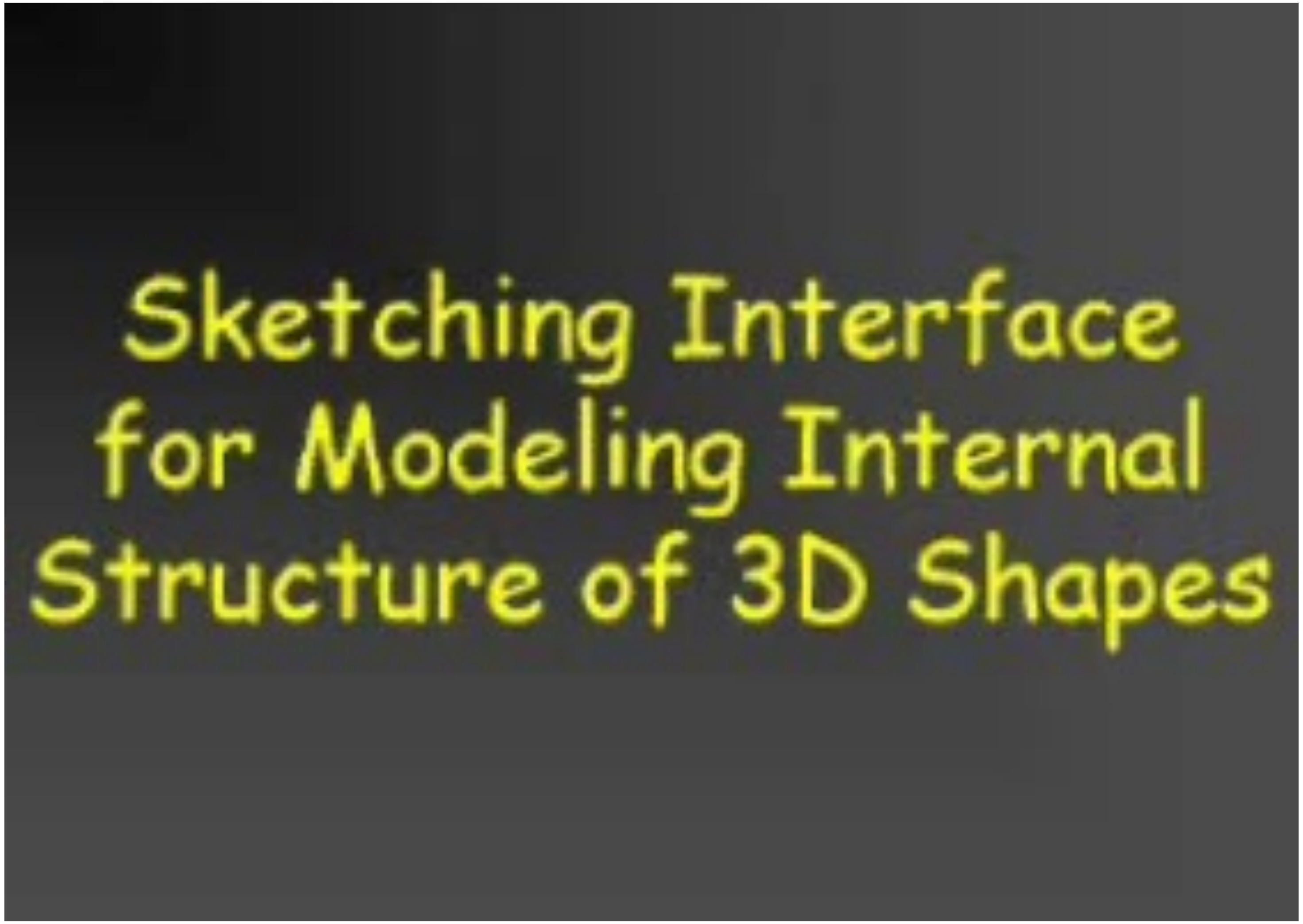


A gradient-based implicit blend [Gourmel,Barthe,Cani,Wyvill,Bernhardt,Grasberger,TOG13]

Example of 3D modeling tool using implicit surfaces

Shapeshop; Sketch-based solid modeling with blobtrees [Schmidt,Wyvill,Sousa,Jorge,SBIM05]

Example of 3D modeling tool using implicit surfaces

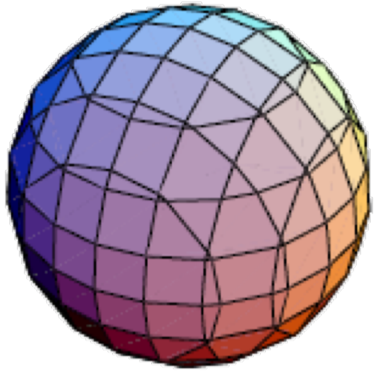


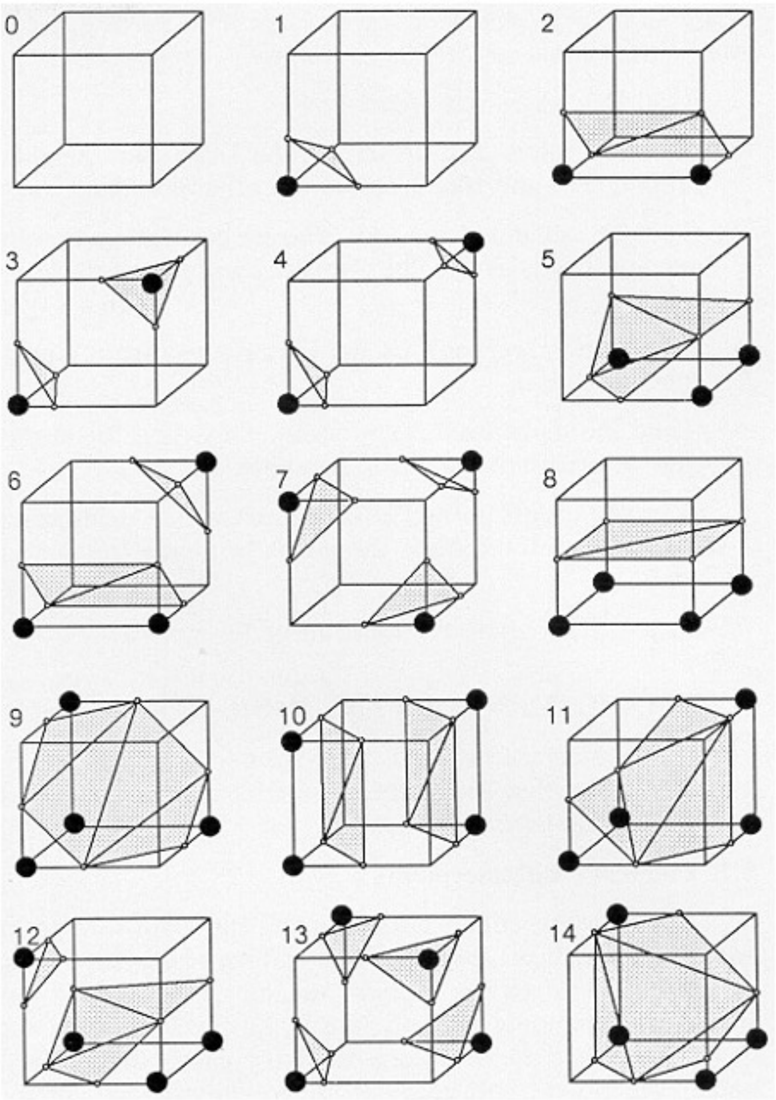
A sketching interface for modeling the internal structures of 3d shapes [Owada,Nielsen,Nakazawa,Igarashi,SmartGraphics03]

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

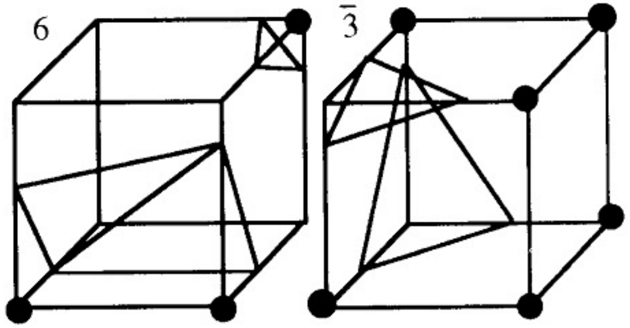




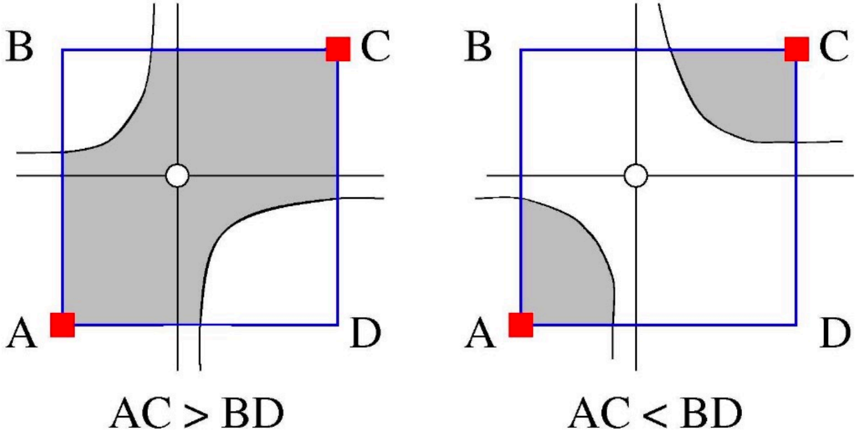
(Once patented☹, now expired☺)

Marching Cubes: A High Resolution 3D Surface Construction Algorithm [Lorensen SIGGRAPH87]

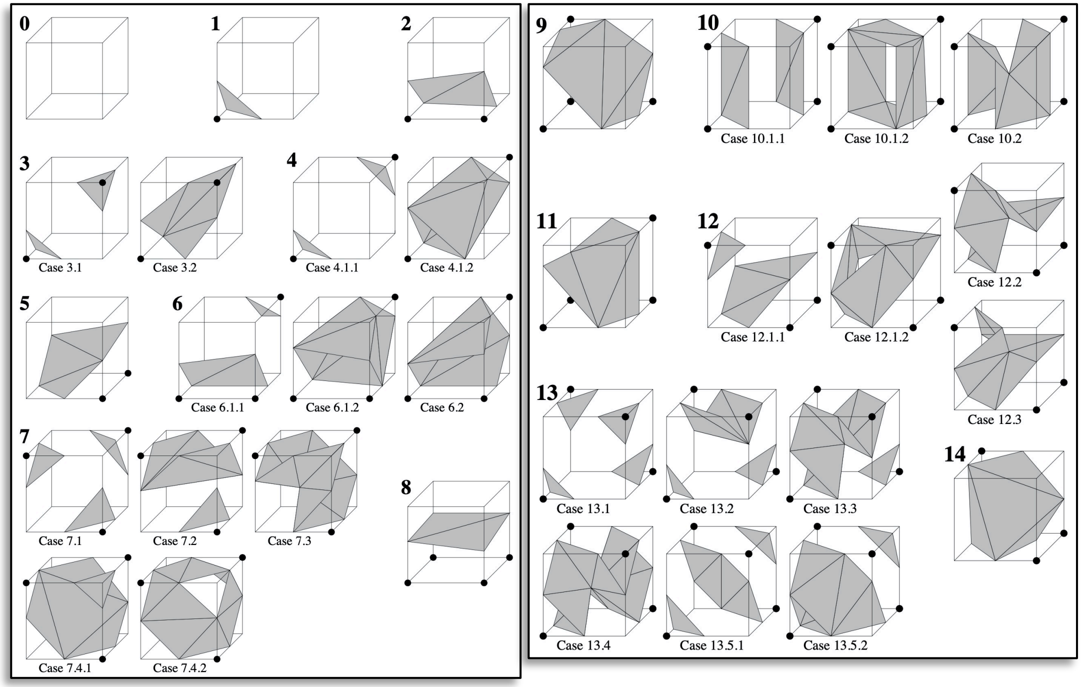
Ambiguity in Marching Cubes



Inconsistent faces between neighboring cubes



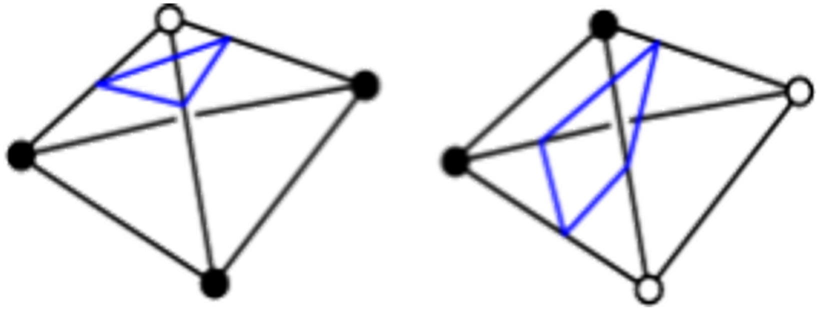
Solution: use bilinear/trilinear interpolation
to determine topology



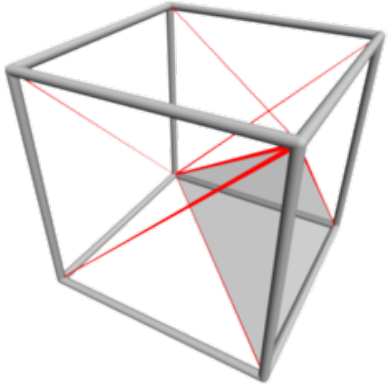
33 patterns for resolving topological ambiguity
(Implementing them correctly can be tricky…)

The asymptotic decider: resolving the ambiguity in marching cubes [Nielson VIS91]
Marching cubes 33: Construction of topologically correct isosurfaces [Chernyaev Tech.Rep. 95]
Topology Verification for Isosurface Extraction [Etiene TVCG12]
A Fast and Memory Saving Marching Cubes 33 Implementation with the Correct Interior Test [Vega JCGT19]

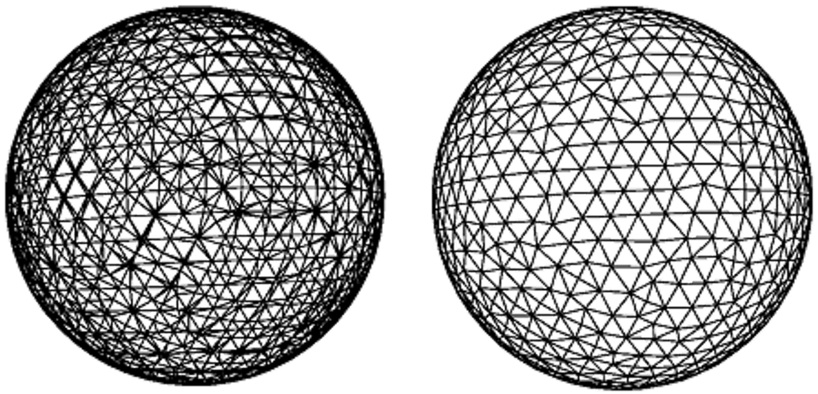
Marching Tetrahedra



- Use tetrahedra instead of cubes
  - Fewer patterns, no ambiguity ☺
    ➔ Simpler implementation
  - More triangles compared to marching cubes ☹



- A cube split into 6 tetrahedra
  - (Make sure consistent splitting across neighboring cubes)

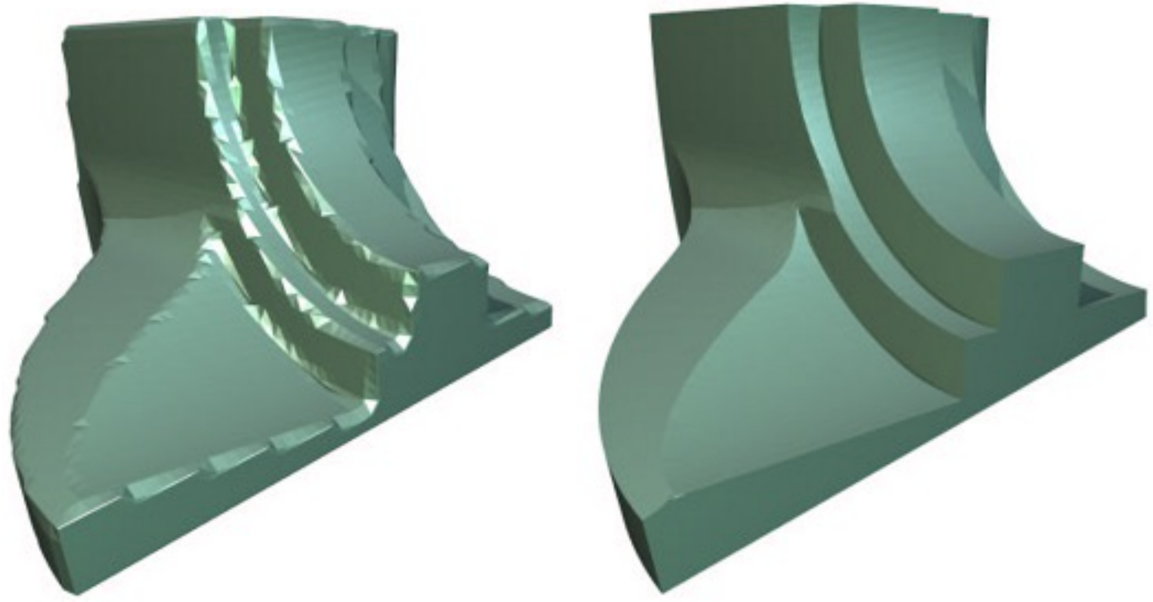


- Some techniques to improve mesh quality

http://paulbourke.net/geometry/polygonise/
Regularised marching tetrahedra: improved iso-surface extraction [Treece C&G99]

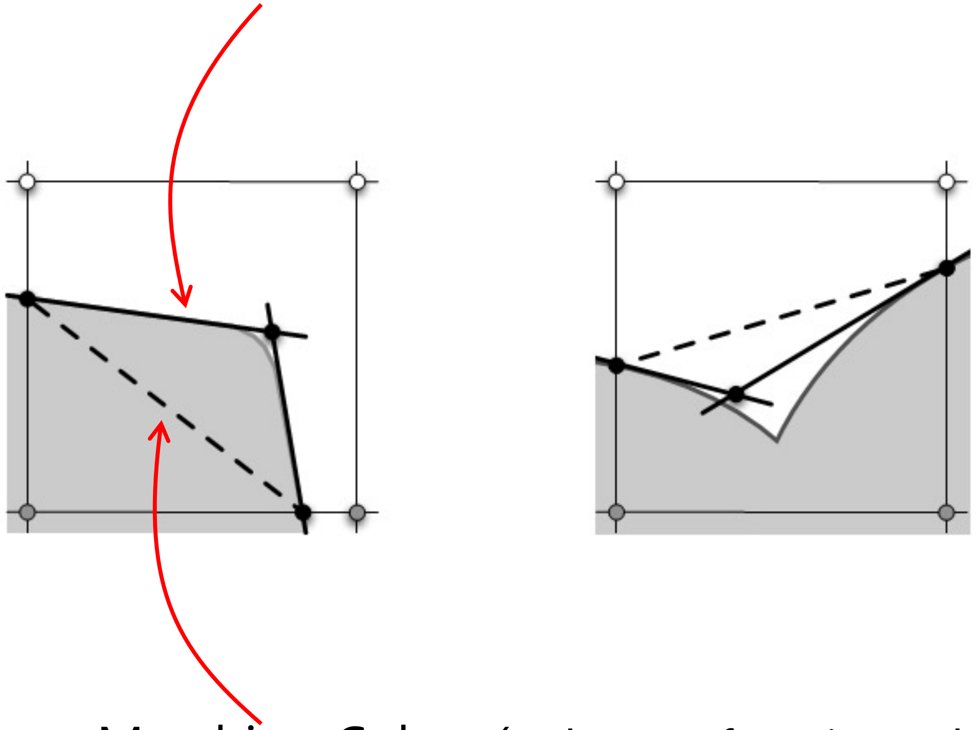
Isosurface extraction preserving sharp edges

Grid size: 65×65×65



Marching Cubes　　Improved version

Improved version (uses function *gradient* as well)

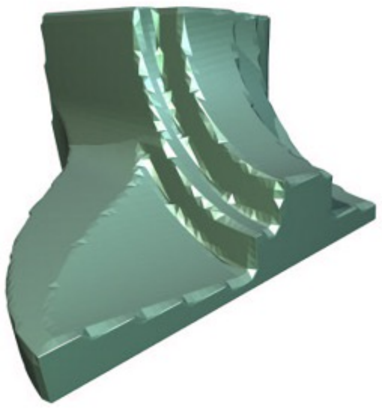


Marching Cubes (only uses function values)

Feature Sensitive Surface Extraction from Volume Data [Kobbelt SIGGRAPH01]
Dual Contouring of Hermite Data [Ju SIGGRAPH02]
http://www.graphics.rwth-aachen.de/IsoEx/

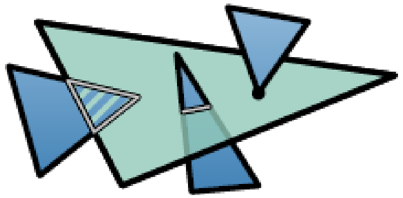
CSG with surface representation only

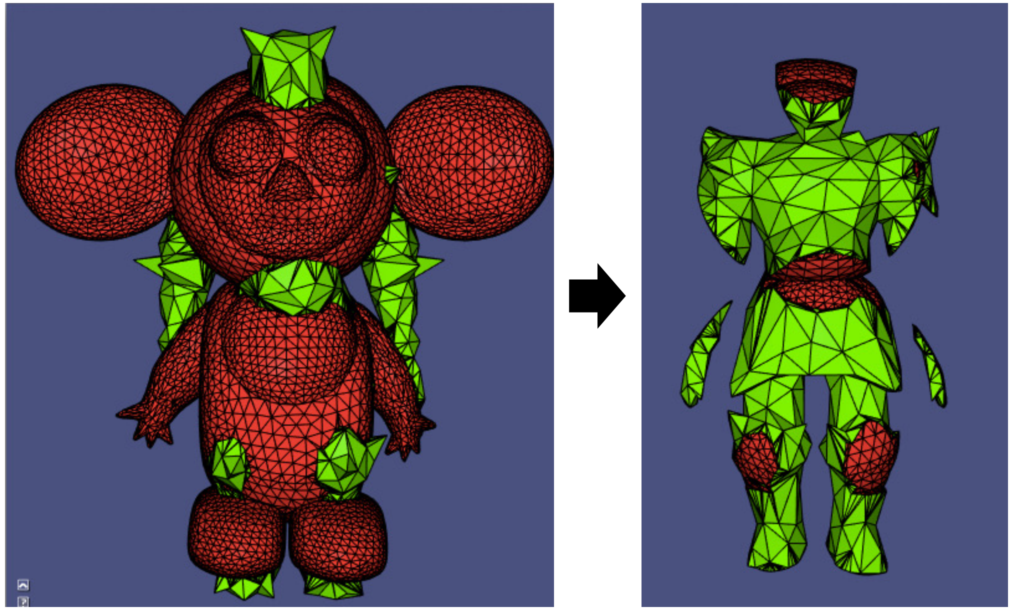
- Volumetric representation (=isosurface extraction using MC)
  ➔ Approximation accuracy depends on grid resolution ☹



- CSG with surface representation only
  ➔ Exactly keep original mesh geometry ☺

- Difficult to implement robust & efficient ☹
  - Floating point error
  - Exactly coplanar faces



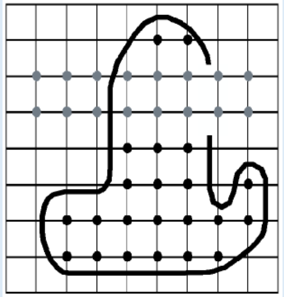


- Notable advances in recent years

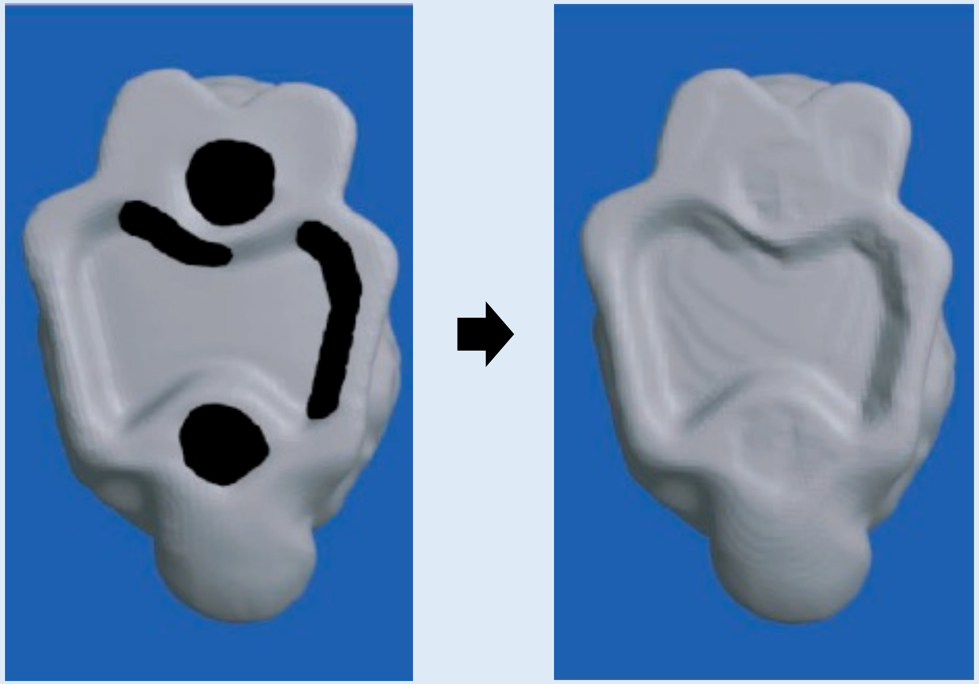
Fast, exact, linear booleans [Bernstein SGP09]
Exact and Robust (Self-)Intersections for Polygonal Meshes [Campen EG10]
Mesh Arrangements for Solid Geometry [Zhou SIGGRAPH16]
https://libigl.github.io/libigl/tutorial/tutorial.html#booleanoperationsonmeshes

Mesh repair

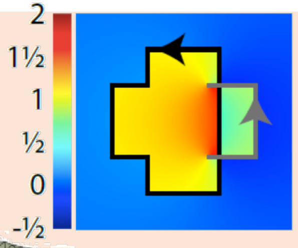
Volumetric representation



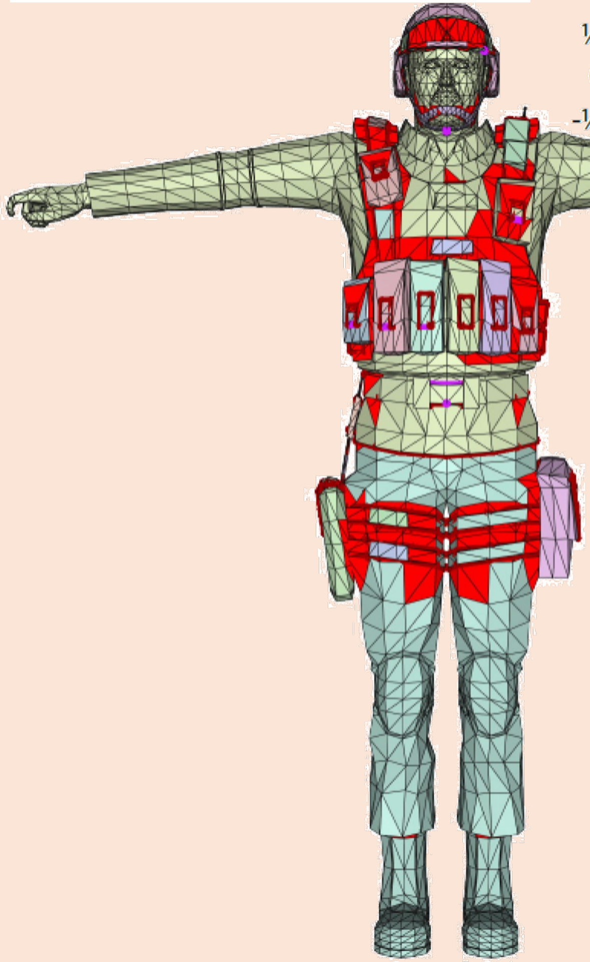
Decide inside/outside by shooting rays from outside

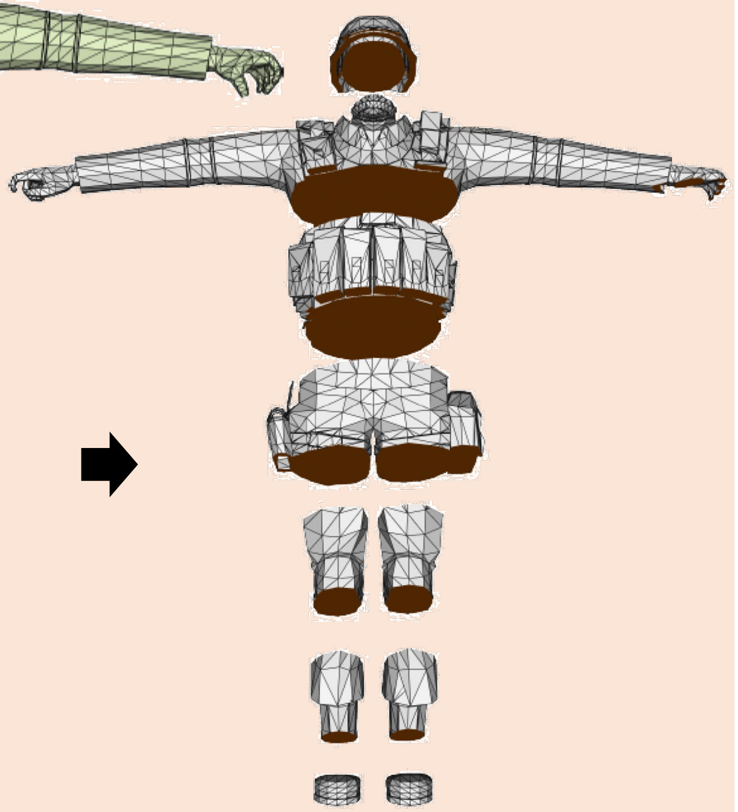


Surface representation



Decide inside/outside based on generalized winding number





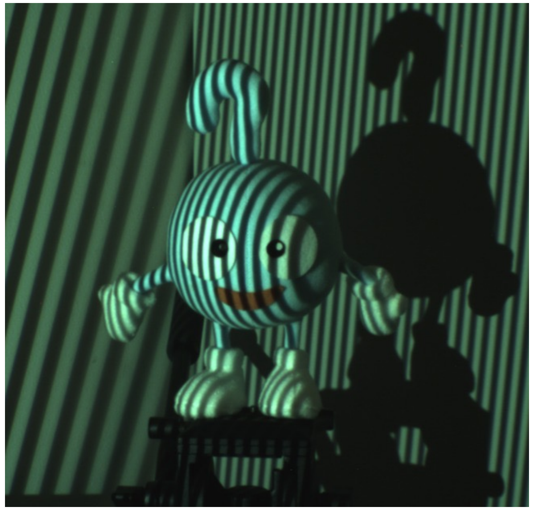
Simplification and Repair of Polygonal Models Using Volumetric Techniques [Nooruddin TVCG03]
Robust Inside-Outside Segmentation using Generalized Winding Numbers [Jacobson SIGGRAPH13]

Surface reconstruction from point cloud

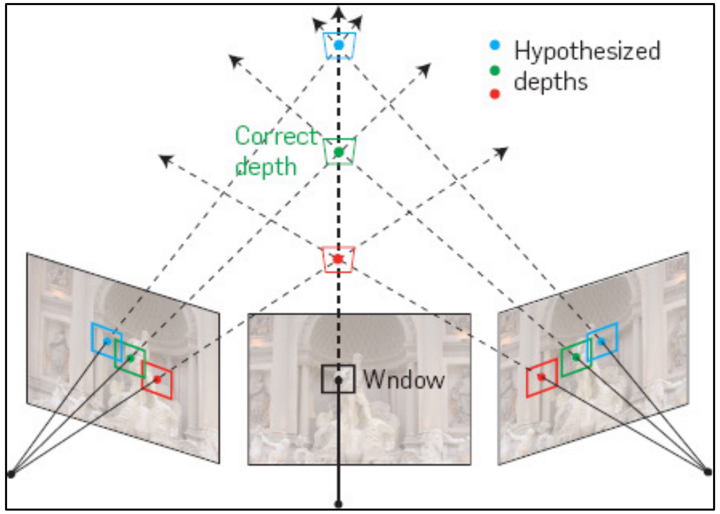
Measuring 3D shapes

Range Scanner (LIDAR)

Depth Camera

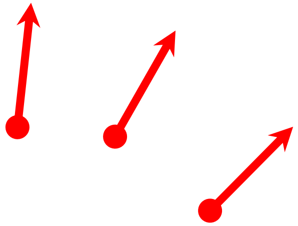


Structured Light



Multi-View Stereo

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



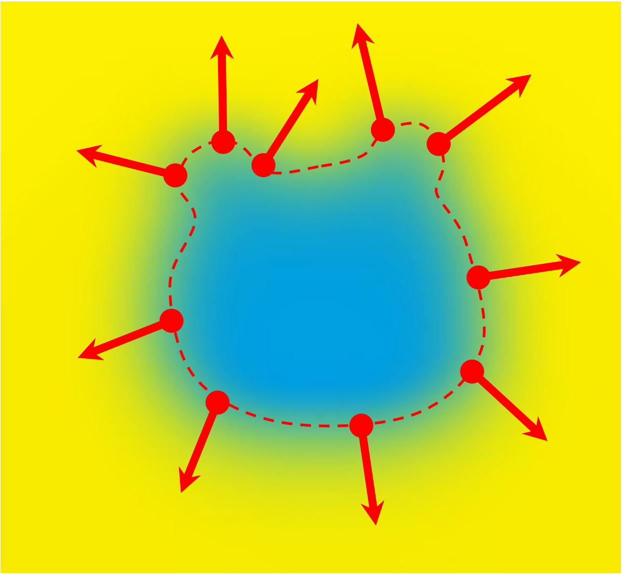
- Normals not available? ➔ Normal estimation
- Too noisy? ➔ Denoising

Typical Computer Vision problems

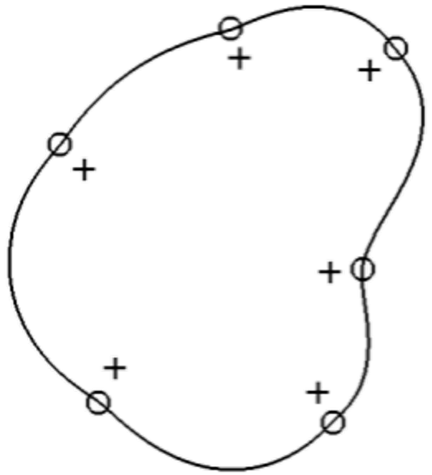
Surface reconstruction from point cloud

- Input: N points
  - Coordinate $\mathbf{x}_i = (x_i, y_i, z_i)$ & normal $\mathbf{n}_i = \left(n_i^{\mathrm{x}}, n_i^{\mathrm{y}}, n_i^{\mathrm{z}}\right)$, $i \in \{1, \ldots, N\}$
- 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 $f(\mathbf{x}) = 0$ ➔ output surface
- “Scattered Data Interpolation”
  - **M**oving **L**east **S**quares
  - **R**adial **B**asis **F**unction

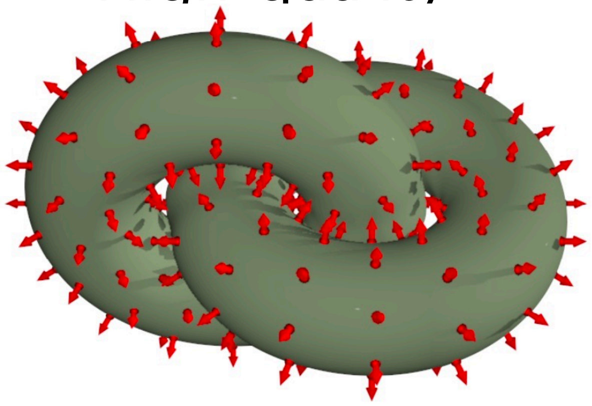
    Important to other fields (e.g. Machine Learning) as well



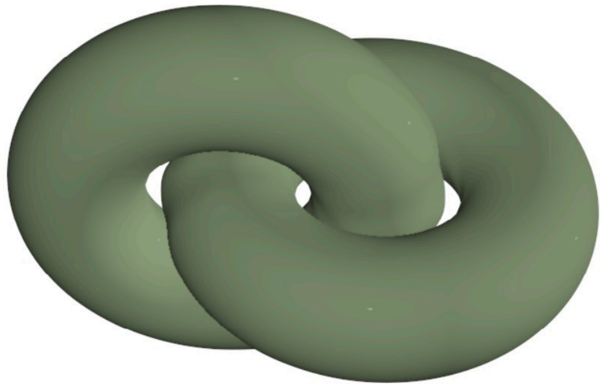
Two ways for controlling gradients



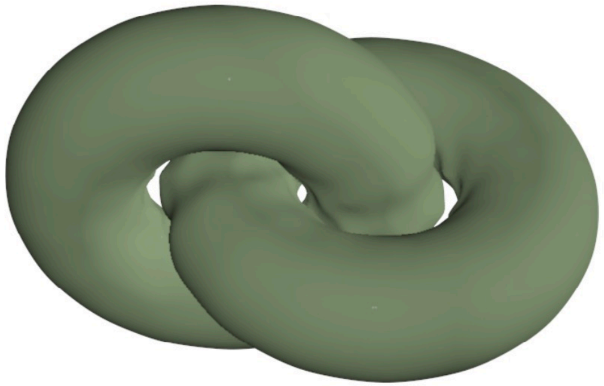
- Additional value constraints at offset locations
  - Simple
- Directly include gradient constraint in the mathematical formulation (Hermite interpolation)
  - High-quality



Value+gradient constraints



Hermite interpolation



Simple offsetting

Modelling with implicit surfaces that interpolate [Turk TOG02]
Hermite Radial Basis Functions Implicits [Macedo CGF10]

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

Starting point: **Least SQuares**

- 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

$$
\begin{aligned}
f(\mathbf{x}_1) &= ax_1 + by_1 + cz_1 + d = f_1 \\
f(\mathbf{x}_2) &= ax_2 + by_2 + cz_2 + d = f_2 \\
&\vdots \\
f(\mathbf{x}_N) &= ax_N + by_N + cz_N + d = f_N
\end{aligned}
$$

$$
\underbrace{\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ & \vdots & & \\ x_N & y_N & z_N & 1 \end{bmatrix}}_{A}
\underbrace{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}_{\mathbf{c}}
=
\underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\mathbf{f}}
$$

- (Forget about gradient constraints for now)

Overconstrained System

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

"normal equation"

$$A\,\mathbf{c} = \mathbf{f} \;\Rightarrow\; A^{\top} A\,\mathbf{c} = A^{\top}\mathbf{f}$$

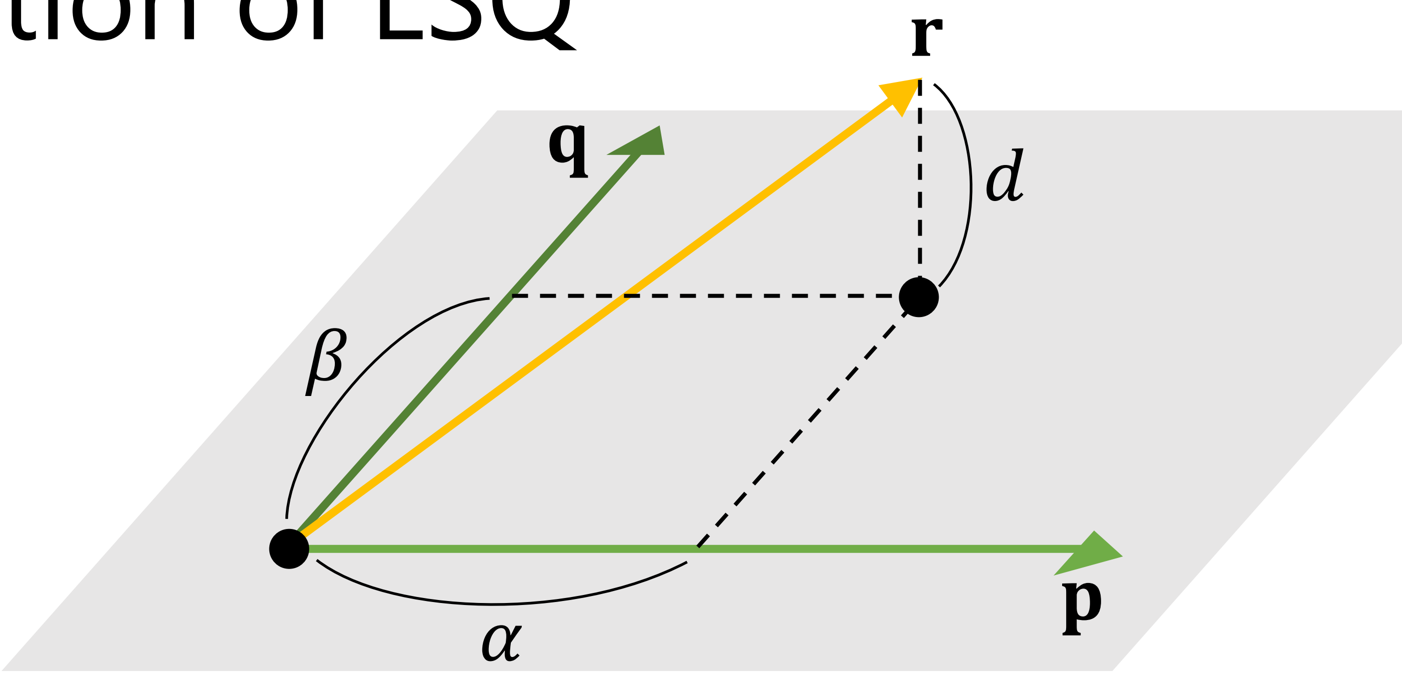
- Minimizing fitting error

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

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

Geometric interpretation of LSQ

$$\begin{bmatrix} p_{\mathrm{x}} & q_{\mathrm{x}} \\ p_{\mathrm{y}} & q_{\mathrm{y}} \\ p_{\mathrm{z}} & q_{\mathrm{z}} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} r_{\mathrm{x}} \\ r_{\mathrm{y}} \\ r_{\mathrm{z}} \end{bmatrix}$$



- 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 **L**east **S**quares

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

- Minimize the following fitting error:

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

$$\underbrace{\begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_N \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ & \vdots & & \\ x_N & y_N & z_N & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}_{\mathbf{c}} = \underbrace{\begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_N \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\mathbf{f}}$$

Weighted **L**east **S**quares

$$W \, A \, \mathbf{c} = W \, \mathbf{f}$$

$$\Rightarrow \quad \mathbf{c} = (A^\intercal W^2 A)^{-1} \, A^\intercal W^2 \, \mathbf{f}$$

Moving **L**east **S**quares

- 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 $\mathbf{x}$

$$\begin{bmatrix} a(\mathbf{x}) \\ b(\mathbf{x}) \\ c(\mathbf{x}) \\ d(\mathbf{x}) \end{bmatrix}$$

$$f(\mathbf{x}) = [\,x\ \ y\ \ z\ \ 1\,]\ (A^\top W(\mathbf{x})^2 A)^{-1}\ A^\top W(\mathbf{x})^2\ \mathbf{f}$$

Introducing gradient (normal) constraints

- Consider linear function represented by each data point:

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

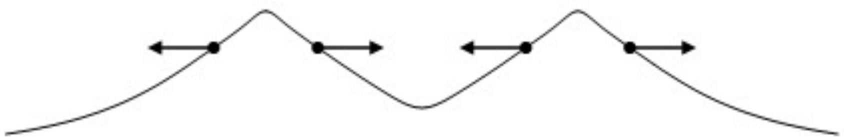
- Minimize fitting error to each g_i evaluated at $\mathbf{x}$:

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

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

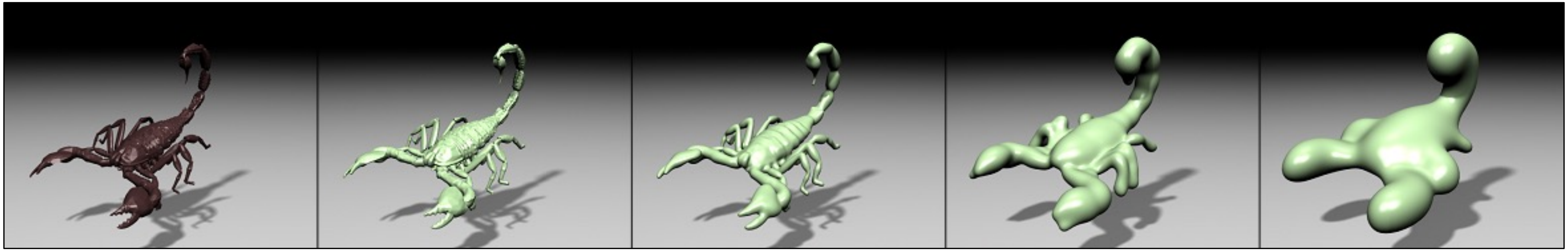
Interpolating and Approximating Implicit Surfaces from Polygon Soup [Shen SIGGRAPH04]

Introducing gradient (normal) constraints



Normal constraints

Simple offsetting



Input : Polygon Soup　Interpolation　Approximation 1　Approximation 2　Approximation 3

Interpolating and Approximating Implicit Surfaces from Polygon Soup [Shen SIGGRAPH04]

Interpolation using **R**adial **B**asis **F**unctions

Basic idea

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

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

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

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

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

$$\vdots$$

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

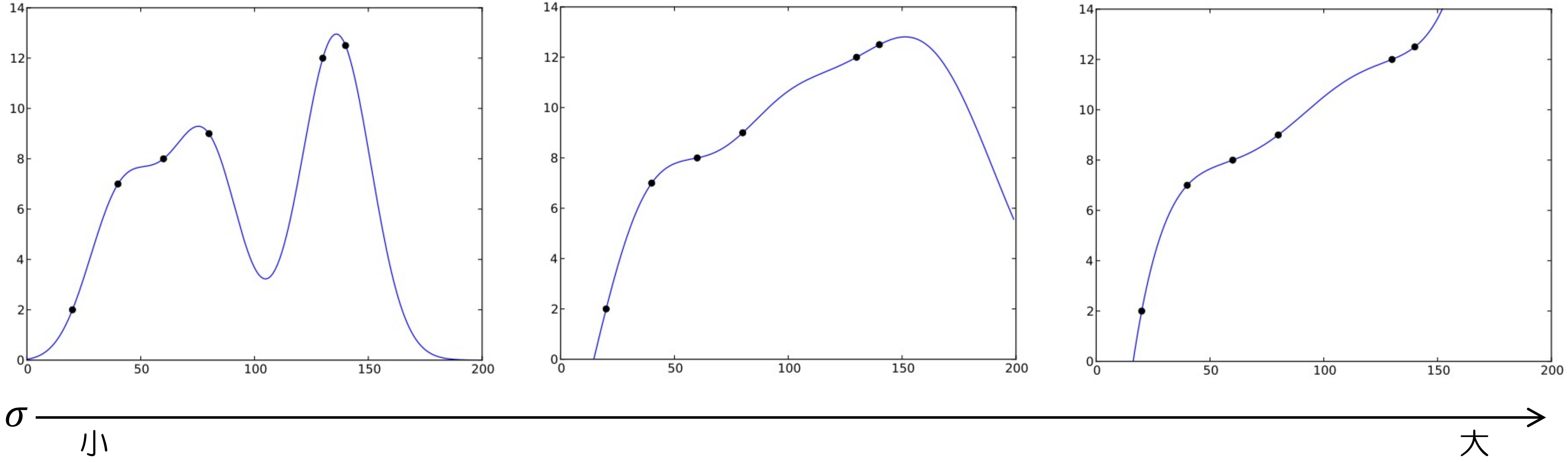
$$\underbrace{\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & & \phi_{1,N} \\ \phi_{2,1} & \phi_{2,2} & & \phi_{2,N} \\ & & \ddots & \\ \phi_{N,1} & \phi_{N,2} & & \phi_{N,N} \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}}_{\mathbf{w}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}}_{\mathbf{f}}$$

Solve this!

When using Gaussian RBF

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

- Results highly dependent on the choice of parameter σ ☹



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

Scattered Data Interpolation for Computer Graphics [Anjyo SIGGRAPH14 Course]

Measure of function's smoothness

$$E_m[f] := \int_{\mathbb{R}^d} \|\boldsymbol{\nabla}^m f(\mathbf{x})\|^2 d\mathbf{x}$$

	2D	3D
$m = 2$	$\boldsymbol{\nabla}^2 f(x,y) := \left(f_{\mathrm{xx}}, f_{\mathrm{xy}}, f_{\mathrm{yx}}, f_{\mathrm{yy}}\right)$ $E_2[f] := \int_{\mathbb{R}^2} f_{\mathrm{xx}}^2 + f_{\mathrm{yy}}^2 + 2f_{\mathrm{xy}}^2$ "Thin-plate" energy	$\boldsymbol{\nabla}^2 f(x,y,z) := \left(f_{\mathrm{xx}}, f_{\mathrm{xy}}, f_{\mathrm{xz}}, f_{\mathrm{yx}}, f_{\mathrm{yy}}, f_{\mathrm{yz}}, f_{\mathrm{zx}}, f_{\mathrm{zy}}, f_{\mathrm{zz}}\right)$ $E_2[f] := \int_{\mathbb{R}^3} f_{xx}^2 + f_{yy}^2 + f_{zz}^2 + 2\left(f_{xy}^2 + f_{yz}^2 + f_{zx}^2\right)$
$m = 3$	$\boldsymbol{\nabla}^3 f(x,y) := \left(f_{xxx}, f_{xxy}, f_{xyx}, f_{xyy}, f_{yxx}, f_{yxy}, f_{yyx}, f_{yyy}\right)$ $E_3[f] := \int_{\mathbb{R}^2} f_{\mathrm{xxx}}^2 + f_{\mathrm{yyy}}^2 + 2\left(f_{\mathrm{xxy}}^2 + f_{\mathrm{yyx}}^2\right)$	$\boldsymbol{\nabla}^3 f(x,y) := \begin{pmatrix} f_{xxx}, f_{xxy}, f_{xxz}, f_{xyx}, f_{xyy}, f_{xyz}, f_{xzx}, f_{xzy}, f_{xzz}, \\ f_{yxx}, f_{yxy}, f_{yxz}, f_{yyx}, f_{yyy}, f_{yyz}, f_{yzx}, f_{yzy}, f_{yzz}, \\ f_{zxx}, f_{zxy}, f_{zxz}, f_{zyx}, f_{zyy}, f_{zyz}, f_{zzx}, f_{zzy}, f_{zzz} \end{pmatrix}$ $E_3[f] := \int_{\mathbb{R}^3} f_{\mathrm{xxx}}^2 + f_{\mathrm{yyy}}^2 + f_{\mathrm{zzz}}^2 + 3\left(f_{\mathrm{xxy}}^2 + f_{\mathrm{yyz}}^2 + f_{\mathrm{zzx}}^2 + f_{\mathrm{xyy}}^2 + f_{\mathrm{yzz}}^2 + f_{\mathrm{zxx}}^2\right) + f_{\mathrm{xyz}}^2$

Great discovery (Duchon 1977)

- Of all functions satisfying $\{ f(\mathbf{x}_i) = f_i \}$, the minimizer of $E_m[f]$ is represented as RBFs with the following basis:
  - When the space dimension is odd: $\phi(\mathbf{x}) = \|\mathbf{x}\|^{2m-3}$
  - When the space dimension is even: $\phi(\mathbf{x}) = \|\mathbf{x}\|^{2m-2} \log\|\mathbf{x}\|$
    - Assume $\phi(0) = 0$
- Popular choice:
  - For 2D: $\phi(\mathbf{x}) = \|\mathbf{x}\|^2 \log\|\mathbf{x}\|$ (minimizes E_2)
  - For 3D: $\phi(\mathbf{x}) = \|\mathbf{x}\|^3$ (minimizes E_3)

Splines minimizing rotation-invariant semi-norms in Sobolev spaces [Duchon, 1977]

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

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

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

$$\vdots$$

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

$$\left[\begin{array}{cccc|cccc} \phi_{1,1} & \phi_{1,2} & & \phi_{1,N} & x_1 & y_1 & z_1 & 1 \\ \phi_{2,1} & \phi_{2,2} & & \phi_{2,N} & x_2 & y_2 & z_2 & 1 \\ & \Phi & \ddots & & & \mathrm{P} \; \vdots & & \\ \phi_{N,1} & \phi_{N,2} & & \phi_{N,N} & x_N & y_N & z_N & 1 \end{array}\right] \left[\begin{array}{c} w_1 \\ w_2 \\ \mathbf{w} \; \vdots \\ w_N \\ \hline a \\ b \\ \mathbf{c} \; c \\ d \end{array}\right] = \left[\begin{array}{c} f_1 \\ f_2 \\ \mathbf{f} \; \vdots \\ f_N \end{array}\right]$$

4 unknowns a, b, c, d added ➔ 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

$$\begin{bmatrix} \Phi & \mathrm{P} \\ \mathrm{P}^\intercal & \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \end{bmatrix}$$

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

Introducing gradient constraints

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

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

Unknown 3D vector

- Gradient of f :

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

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

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

Hessian matrix (function)

Hermite Radial Basis Functions Implicits [Macedo CGF10]

Introducing gradient constraints

- 1st data point:

Value constraint:

$$f(\mathbf{x}_1) = w_1\phi_{1,1} + \mathbf{v}_1^\intercal \nabla\phi_{1,1} + w_2\phi_{1,2} + \mathbf{v}_2^\intercal \nabla\phi_{1,2} + \cdots + w_N\phi_{1,N} + \mathbf{v}_N^\intercal \nabla\phi_{1,N} \ldots by_1 + cz_1 + d = f_1$$

Gradient constraint:

$$\nabla f(\mathbf{x}_1) = w_1\nabla\phi_{1,1} + \mathrm{H}_\phi^{1,1}\mathbf{v}_1 + w_2\nabla\phi_{1,2} + \mathrm{H}_\phi^{1,2}\mathbf{v}_2 + \cdots + w_N\nabla\phi_{1,N} + \mathrm{H}_\phi^{1,N} \ldots \mathbf{n}_1$$

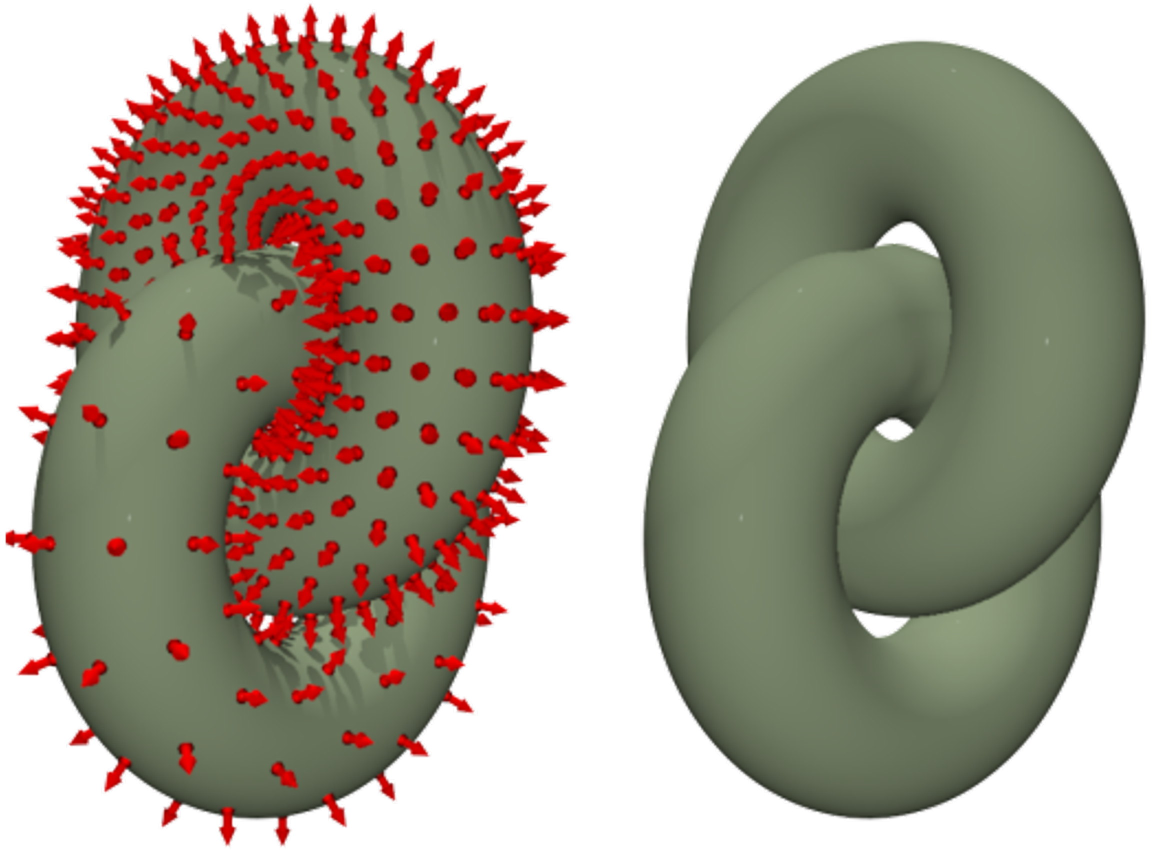
$$\begin{bmatrix} \Phi_{1,1} & \Phi_{1,2} & \cdots & \Phi_{1,N} & \mathrm{P}_1 \end{bmatrix} \begin{bmatrix} w_1 \\ \mathbf{v}_1 \\ w_2 \\ \mathbf{v}_2 \\ \vdots \\ w_N \\ \mathbf{v}_N \\ a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f_1 \\ \mathbf{n}_1 \end{bmatrix}$$

$$\Phi_{1,1} = \begin{bmatrix} \phi_{1,1} & (\nabla\phi_{1,1})^\intercal \\ \nabla\phi_{1,1} & \mathrm{H}_\phi^{1,1} \end{bmatrix}, \quad \mathrm{P}_1 = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

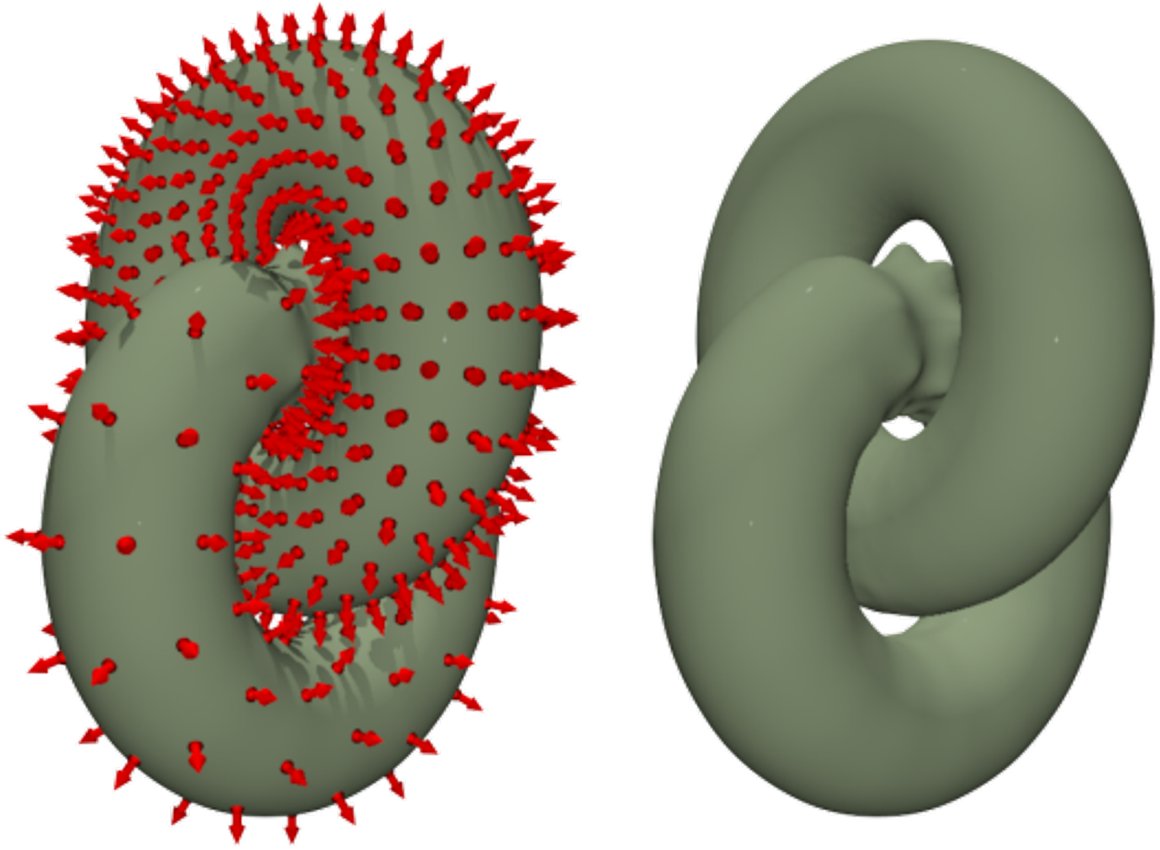
Hermite Radial Basis Functions Implicits [Macedo CGF10]

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

Comparison



Gradient constraints



Simple offsetting with value constraints only

Recent work: global normal estimation

- With known locations $\{\mathbf{x}_i\}$ and unknown normals $\{\mathbf{n}_i\}$, a function f satisfying

  value constraints $\quad f(\mathbf{x}_i) = 0$

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

  can be uniquely specified by using RBF

  ➔ Unknown normals $\{\mathbf{n}_i\}$ determine the function: $f_{\{\mathbf{n}_i\}}$

  ➔ Unknown normals $\{\mathbf{n}_i\}$ determine the function's smoothness:

$$E_{\{\mathbf{n}_i\}} := E\left[f_{\{\mathbf{n}_i\}}\right]$$

$$= \mathbf{n}^\top H\, \mathbf{n}$$

$$\mathbf{n} = \begin{pmatrix} \vdots \\ \mathbf{n}_i^\top \\ \vdots \end{pmatrix}$$

Matrix H depends only on $\{\mathbf{x}_i\}$

- Formulated as a quadratically-constrained quadratic programming:

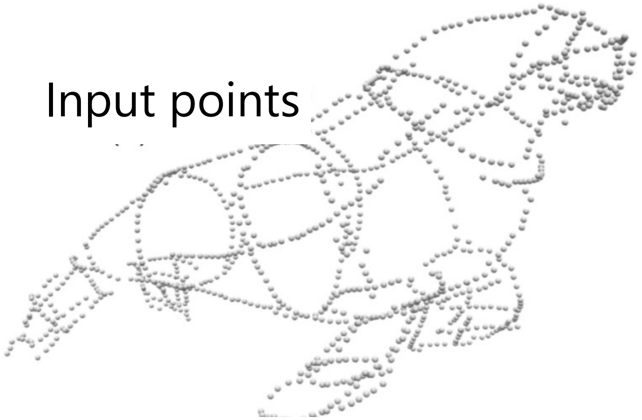
$$\text{minimize } \mathbf{n}^\top H\, \mathbf{n}$$

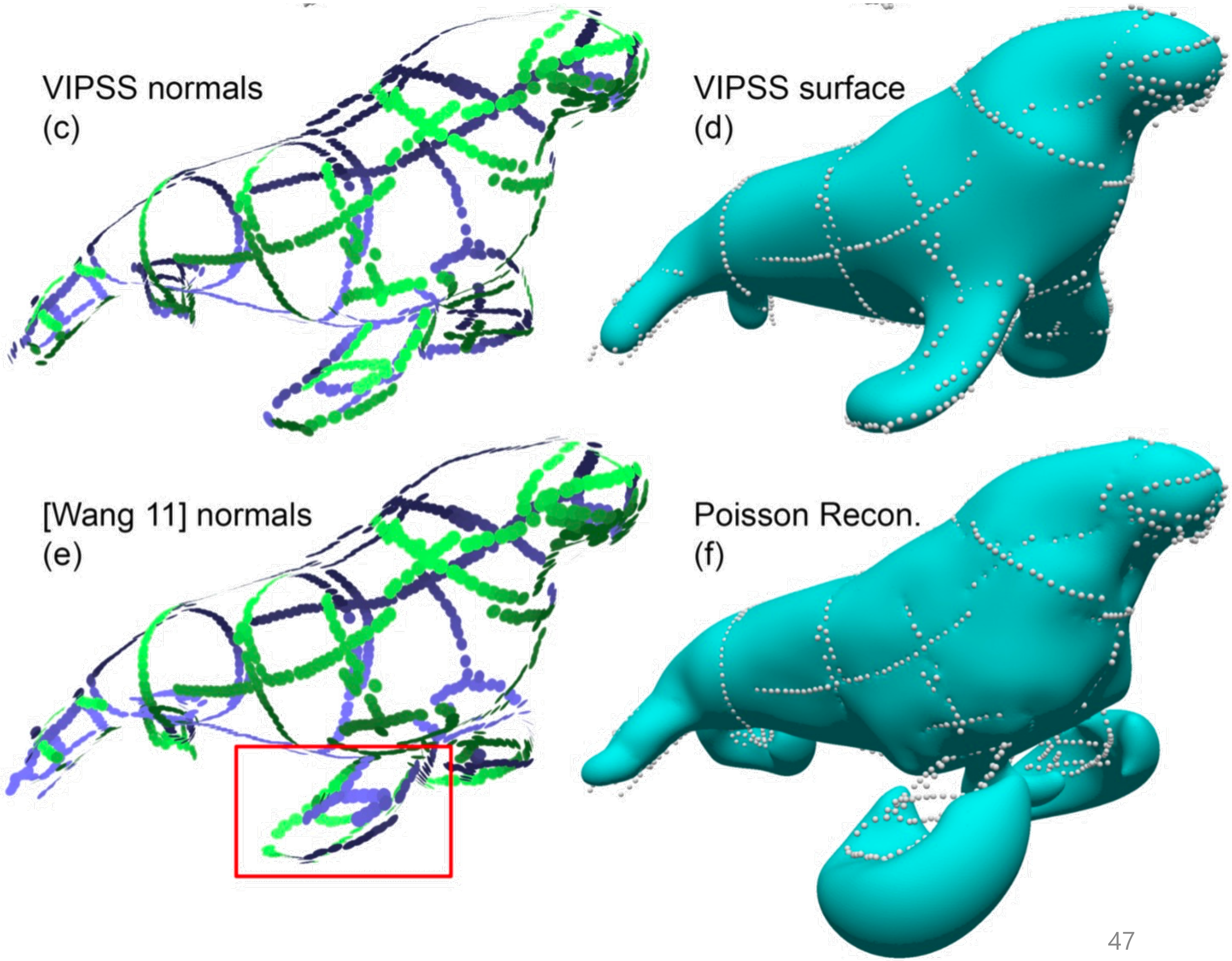
$$\text{s.t. } \mathbf{n}_i^\top \mathbf{n}_i = 1 \quad \forall i$$

Variational Implicit Point Set Surfaces

Recent work: global normal estimation

Input points





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