

# コンピュータグラフィクス論

## - モデリング(3) -

2020年4月30日

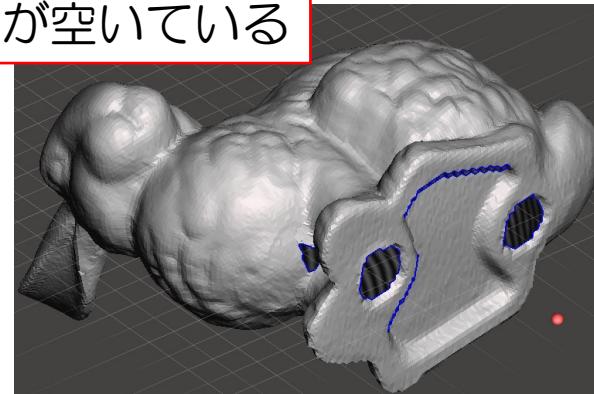
高山 健志

# ソリッドモデリング

# ソリッドモデルとは

- 3D 空間の任意の位置で、モデルの“内側”と“外側”が定義できるもの

ソリッドでないケース

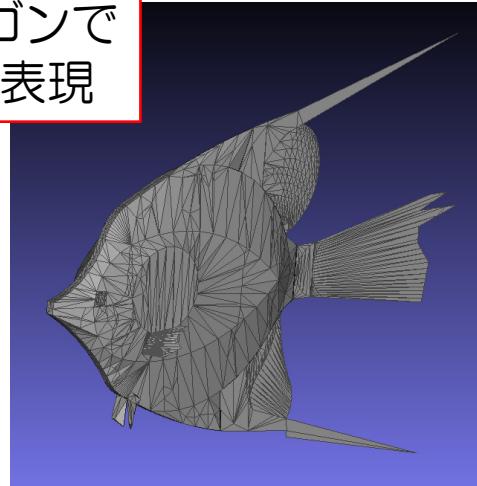


- 主な用途

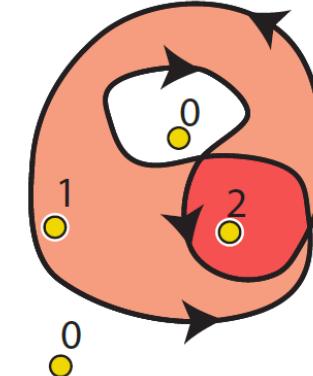
3D プリント



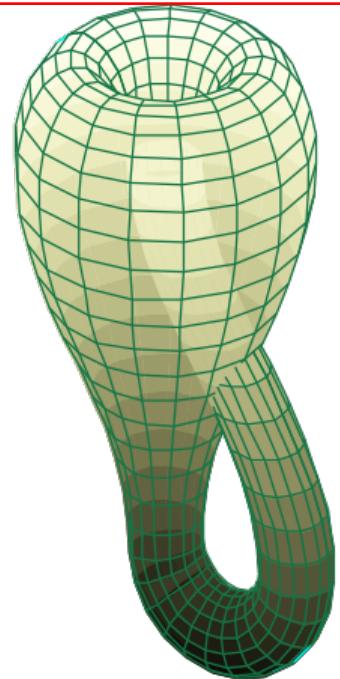
一枚のポリゴンで  
薄い形状を表現



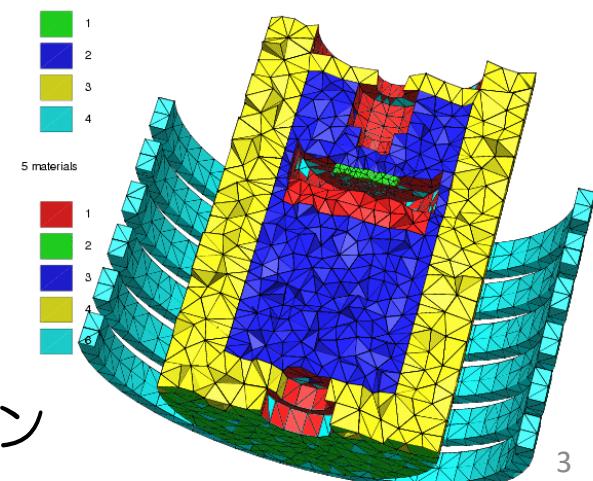
自己交差している



向き付け不可能



Klein bottle



物理シミュレーション

# ソリッドモデルの predicate 関数

- 3D 座標  $\mathbf{p} \in \mathbb{R}^3$  がソリッドモデルの内部であれば true を、そうでなければ false を返す関数：

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

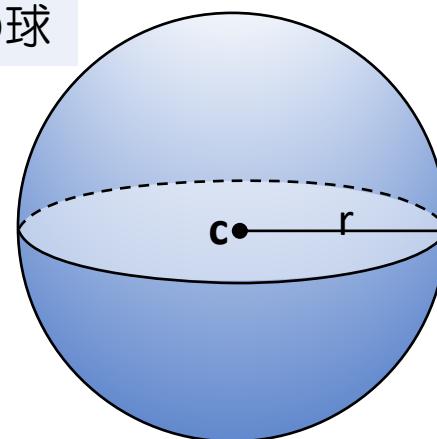
- モデル内部全体を表す集合：

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

- 例：

点  $\mathbf{c}$  を中心とした半径  $r$  の球

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

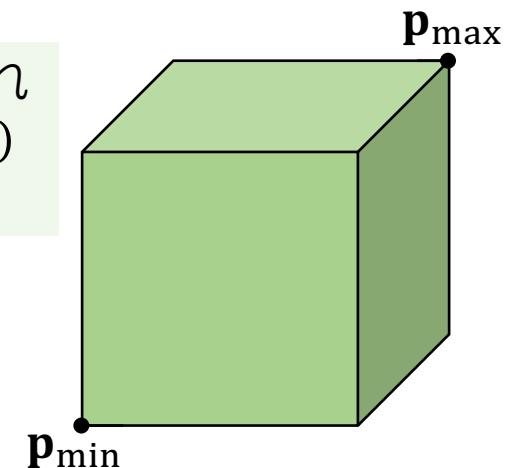


最小と最大の対角コーナーがそれぞれ  $(x_{\min}, y_{\min}, z_{\min})$  と  $(x_{\max}, y_{\max}, z_{\max})$  であるような直方体

$$f(x, y, z) := (x_{\min} < x < x_{\max})$$

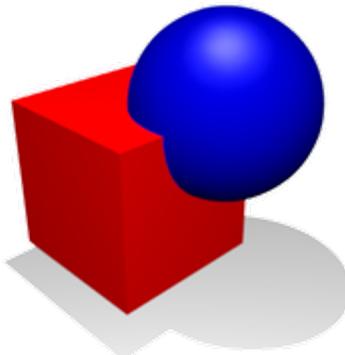
$$\wedge (y_{\min} < y < y_{\max})$$

$$\wedge (z_{\min} < z < z_{\max})$$



# Constructive Solid Geometry (Boolean演算)

和



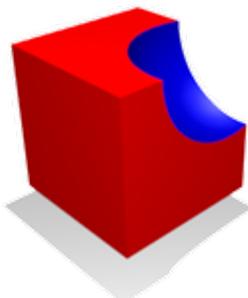
$$f_{A \cup B}(\mathbf{p}) := f_A(\mathbf{p}) \vee f_B(\mathbf{p})$$

積



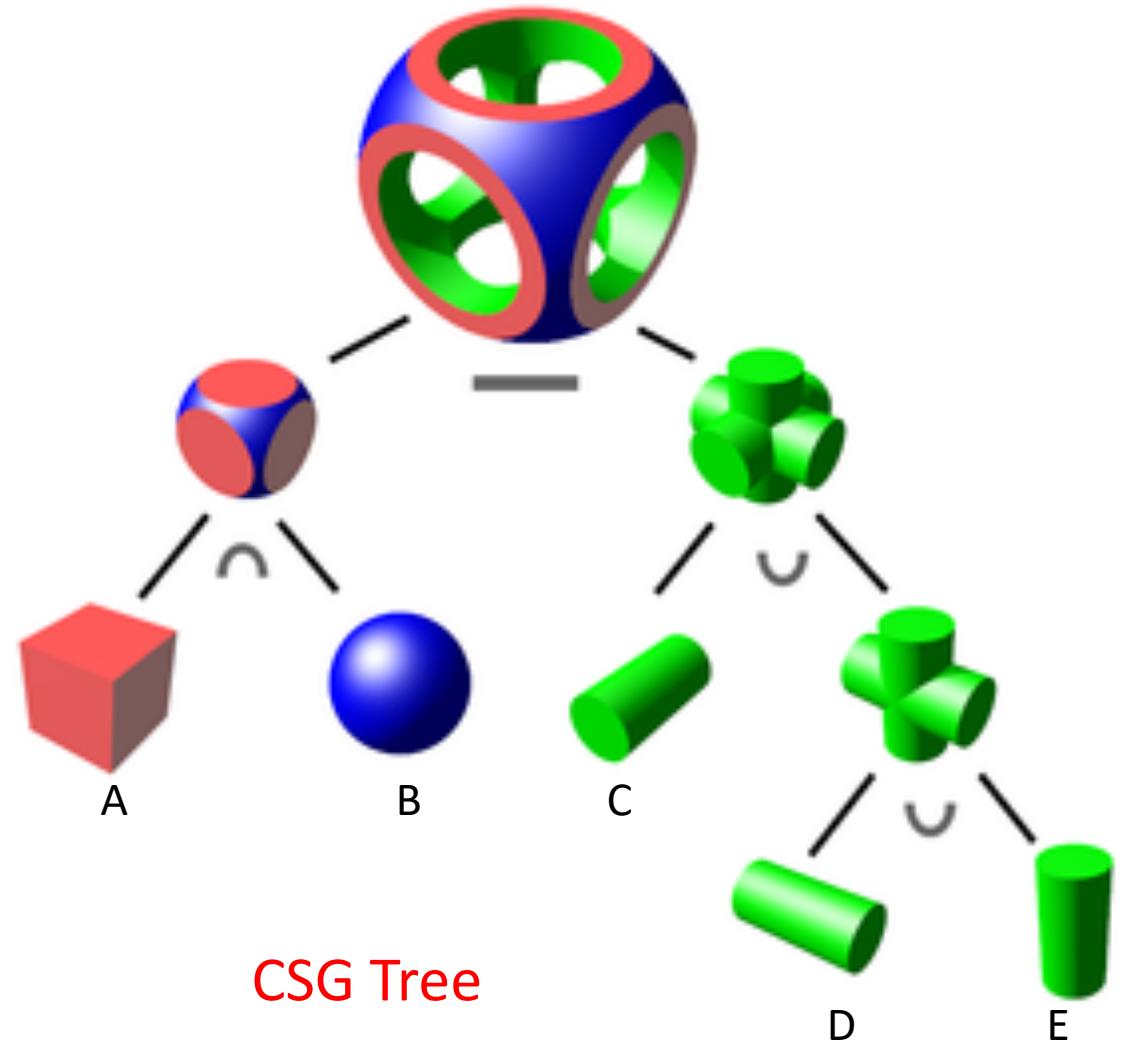
$$f_{A \cap B}(\mathbf{p}) := f_A(\mathbf{p}) \wedge f_B(\mathbf{p})$$

差



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

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

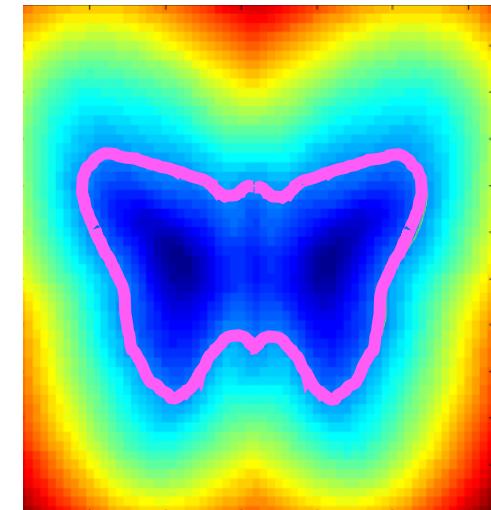


# 符号付き距離場によるソリッドモデル表現

- **Signed Distance Function:** 3D座標からモデル表面までの最短距離

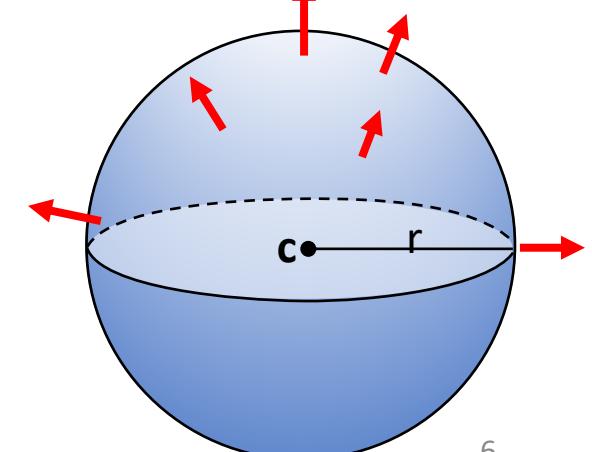
$$d(\mathbf{p}): \mathbb{R}^3 \mapsto \mathbb{R}$$

- 符号付き：内側では負、外側では正
- ソリッドモデルを表すpredicate:  
$$f(\mathbf{p}) := d(\mathbf{p}) < 0$$
- ゼロ等値面はモデル表面を表す：  
$$\{\mathbf{p} \mid d(\mathbf{p}) = 0\} \subset \mathbb{R}^3$$
- 「陰関数表現」 「ボリューム表現」
- 等値面の法線は勾配  $\nabla d(\mathbf{p})$  の方向と一致



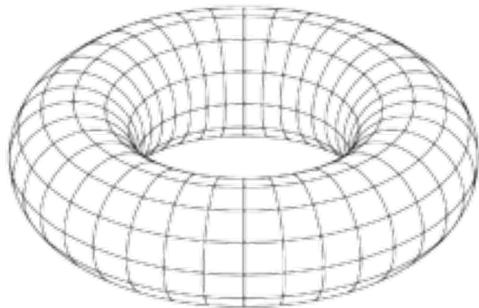
点  $\mathbf{c}$  を中心とした半径  $r$  の球

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



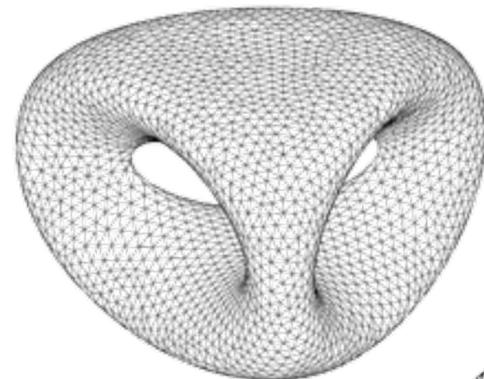
# 陰関数のデザイン例

必ずしも距離関数とは限らない

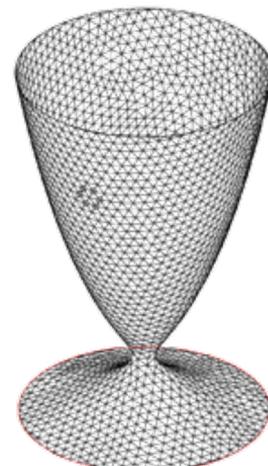


大半径  $R$ , 小半径  $a$  のトーラス

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



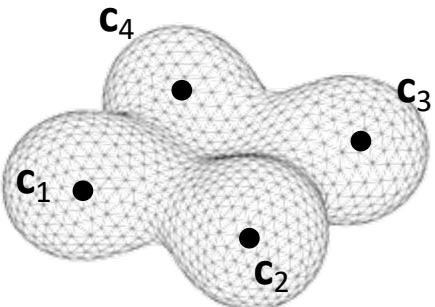
$$2y(y^2 - 3x^2)(1 - z^2) + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2) = 0$$



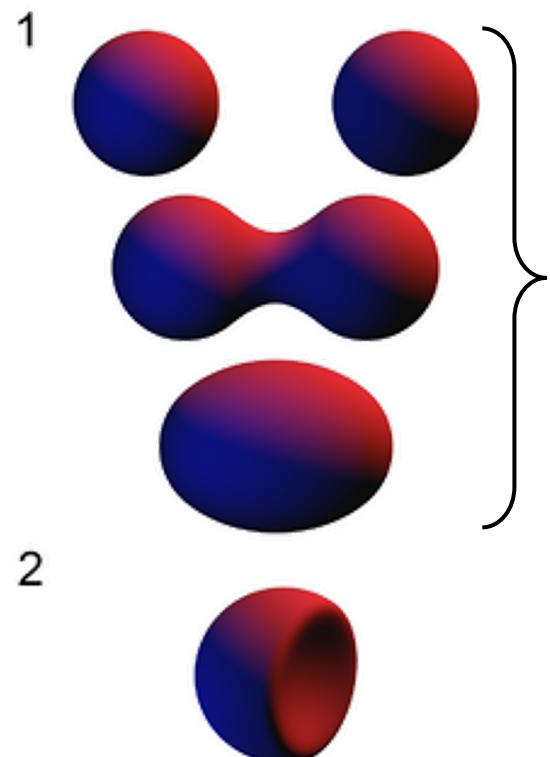
$$x^2 + y^2 - (\ln(z + 3.2))^2 - 0.02 = 0$$

# 陰関数のデザイン例：等電位面 (Metaball)

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



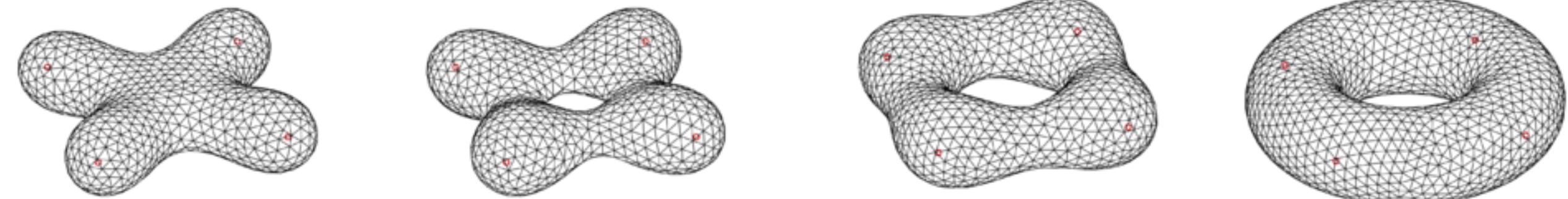
$$d(\mathbf{p}) = d_1(\mathbf{p}) + d_2(\mathbf{p}) + d_3(\mathbf{p}) + d_4(\mathbf{p})$$



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

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

# 陰関数の線形補間によるモーフィング



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

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

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

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

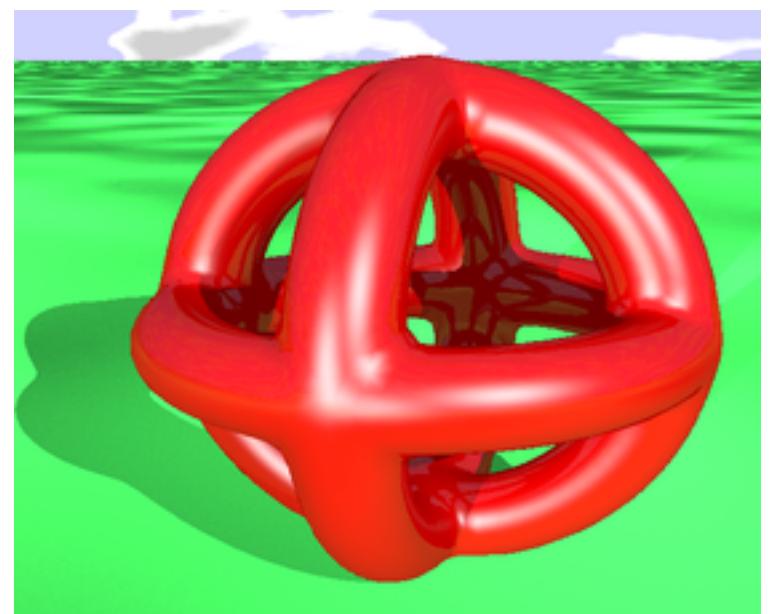
# 複数の陰関数を組み合わせたモデリング

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

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

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

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

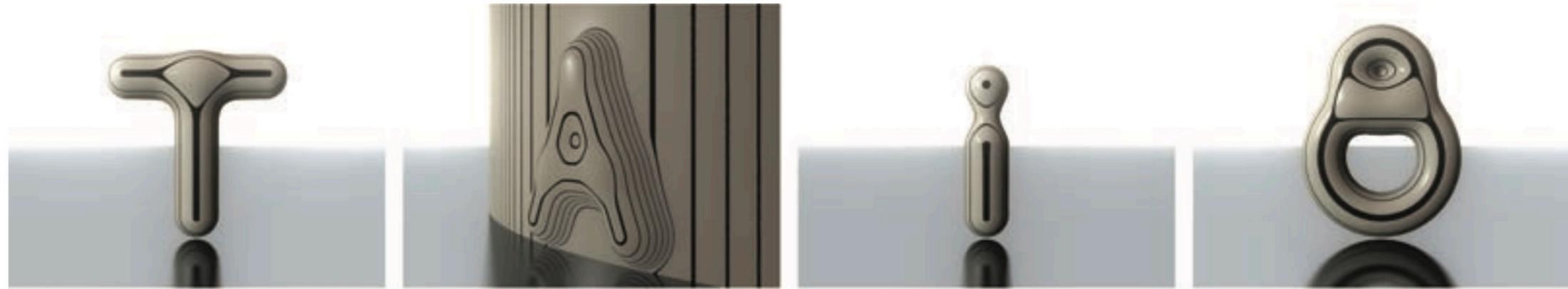


# より高度なブレンディング

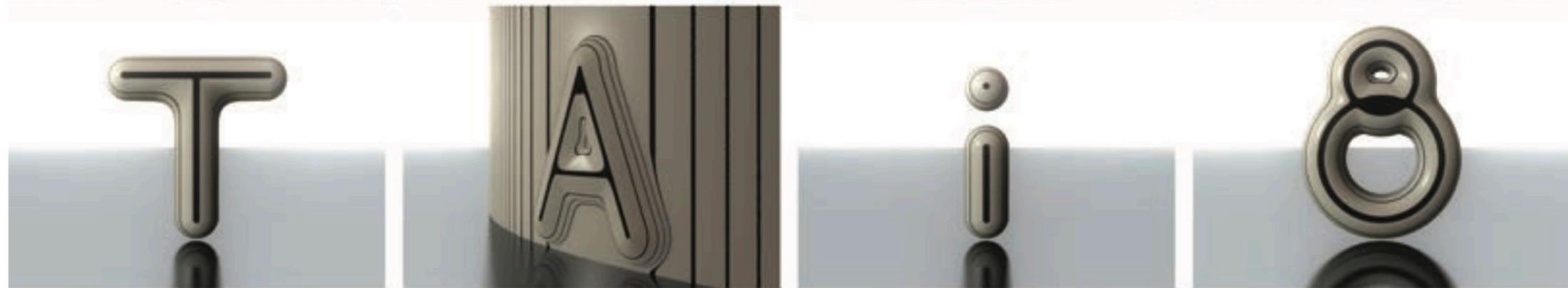
- 2つの陰関数をブレンドする際、それらの**勾配の向き**に応じてブレンド方法を変える



従来法  
(ただの和)



提案法



# 陰関数の表示方法：Marching Cubes

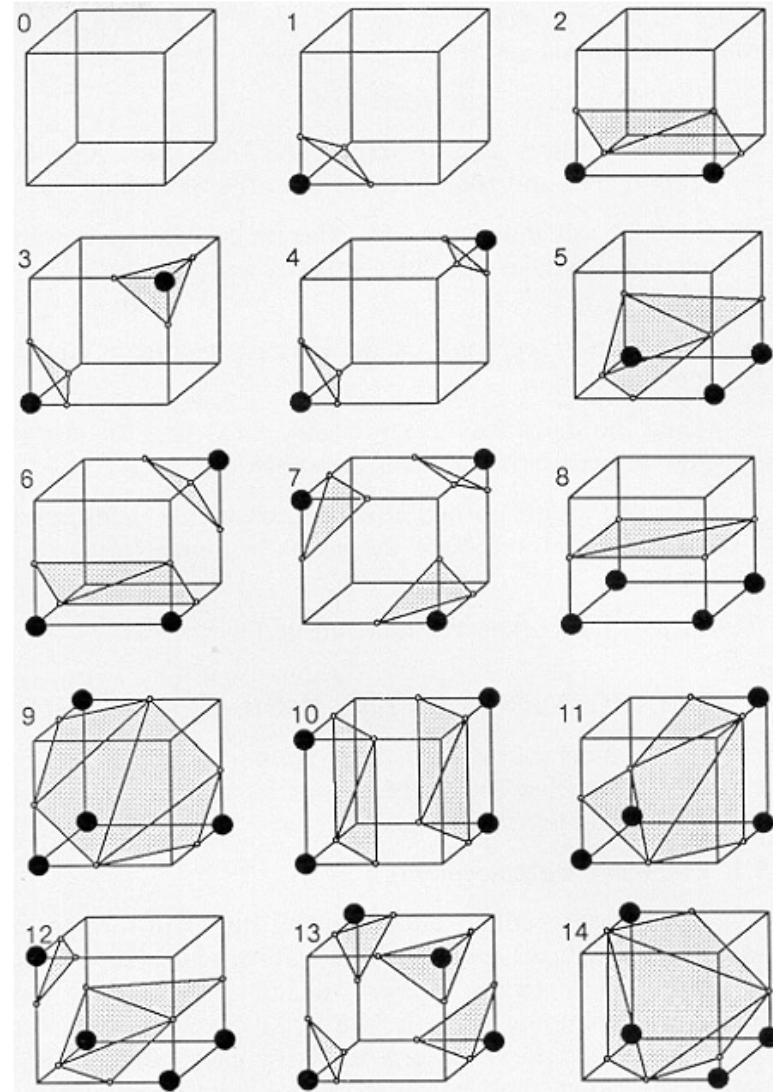
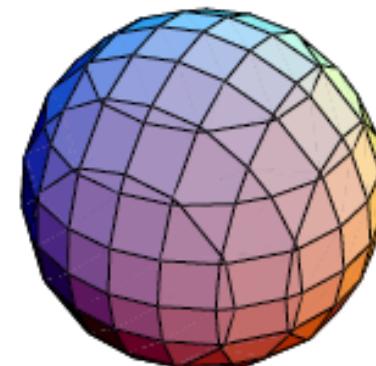
- 等値面を三角形メッシュとして抽出

- 立方体格子の各セルに対し、  
(1) 立方体の 8 頂点で関数値を計算

- (2) その正負のパターンから、  
生成する面のタイプを決定
  - 対称性から 15 通りに分類

- (3) 関数値の線形補間から  
面の位置を決定

(特許で縛られていたが、期限が切れた)



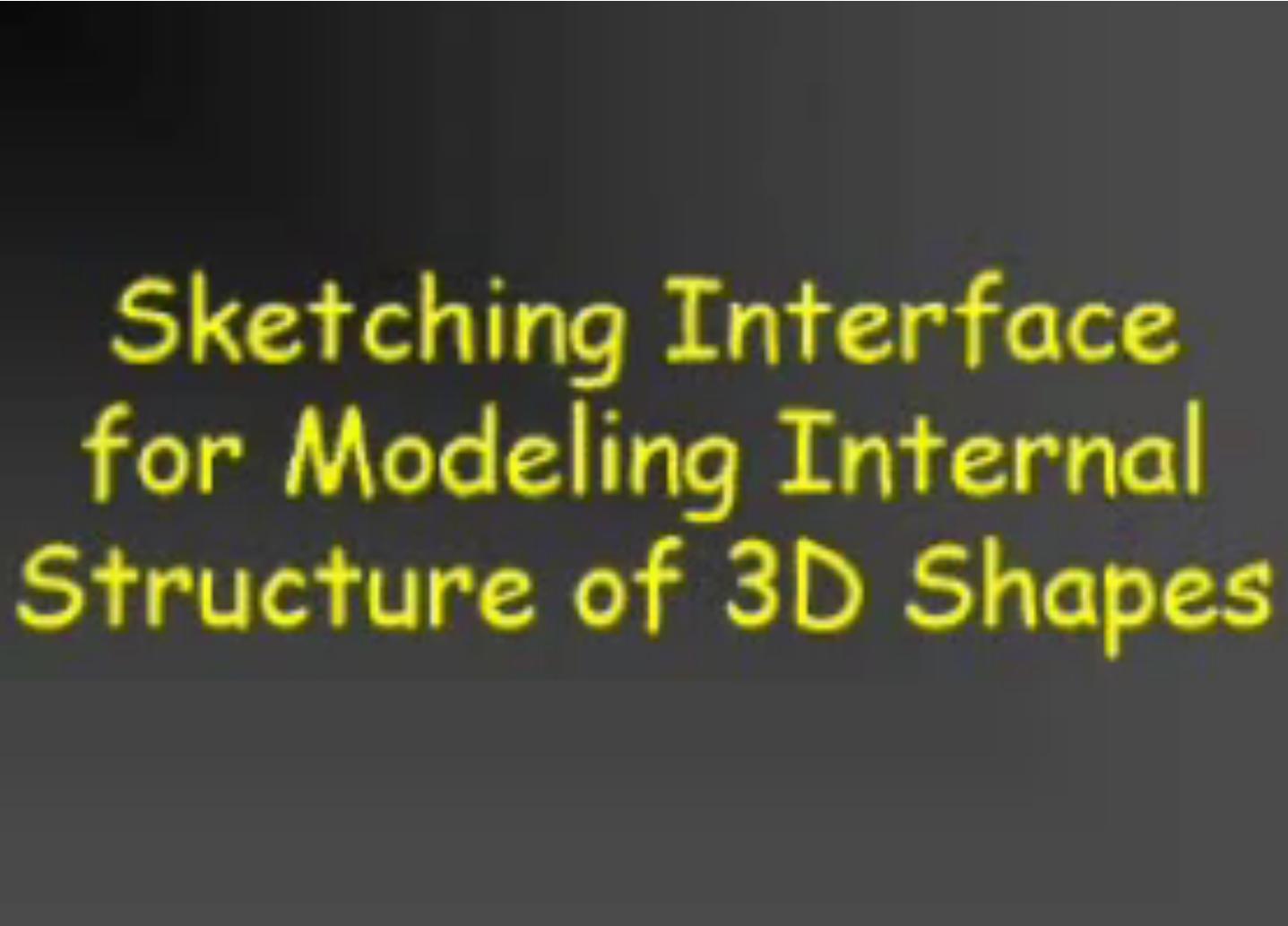
# 陰関数の性質を活用した3Dモデリングの例



ShapeShop v002

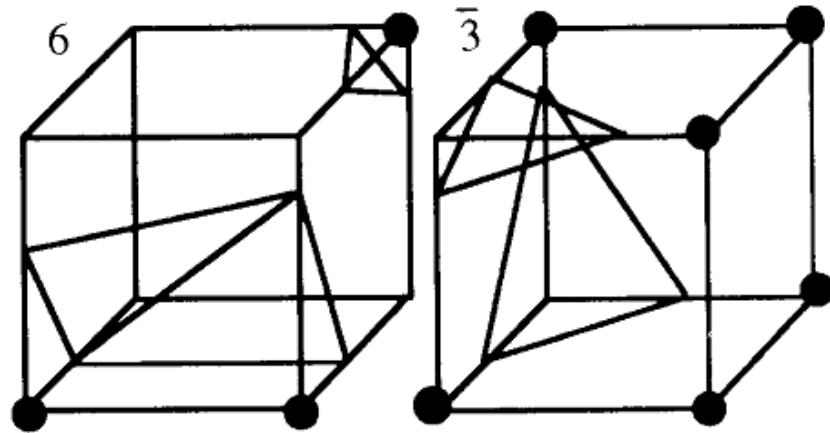
Demo Reel

# 陰関数の性質を活用した3Dモデリングの例

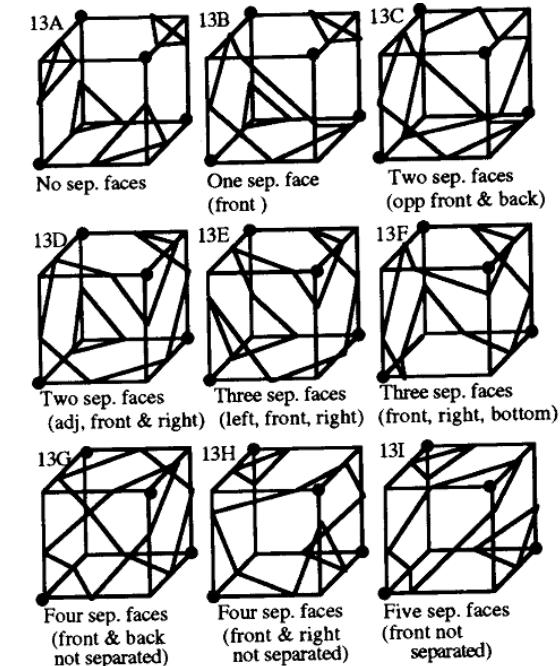


Sketching Interface  
for Modeling Internal  
Structure of 3D Shapes

# Marching Cubes の曖昧性



隣接するセルの間で面が整合しない

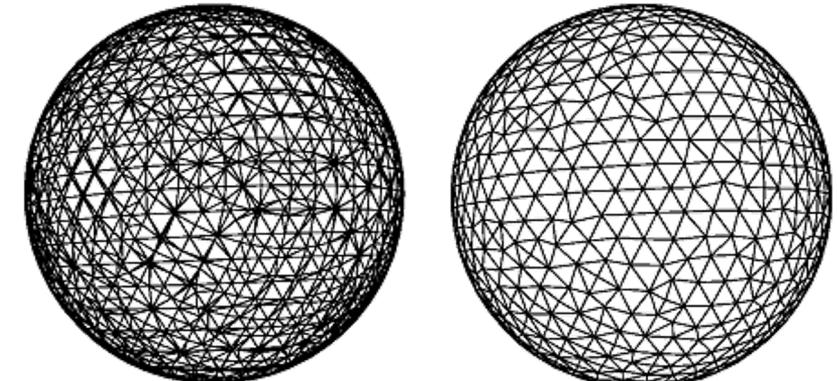
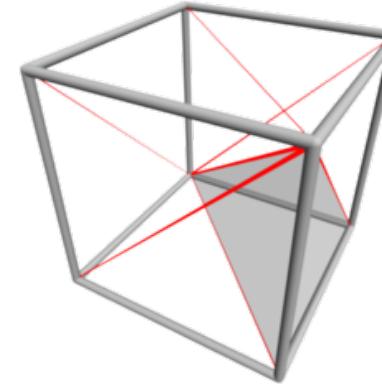
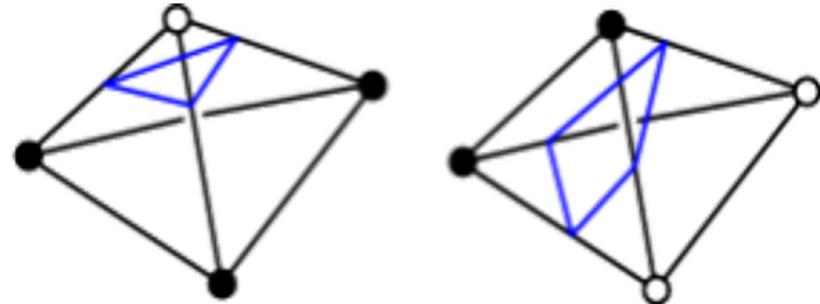


不整合を解決する新たなルール

- 根本的な解決法は、格子を細かくすること

# Marching Tetrahedra

- 立方体の代わりに四面体を使う
  - パターンが少なく、曖昧性が無い  
→ 実装が簡単
- 各立方体セルを、6個の四面体に分割
  - (隣接セル間で分割の向きを合わせることに注意)
- きれいな三角形メッシュを取り出す工夫

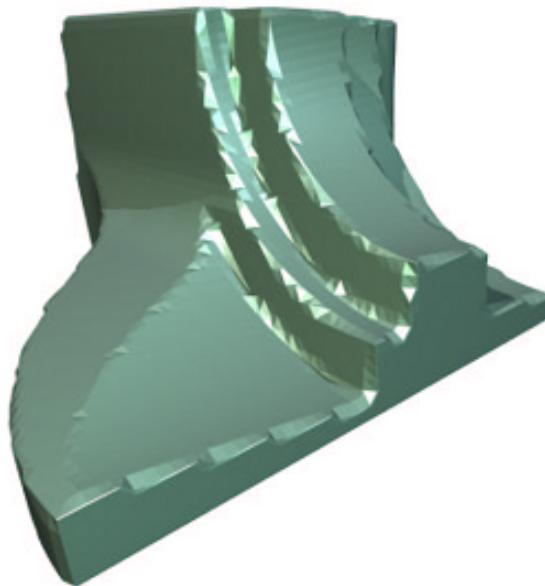


<http://paulbourke.net/geometry/polygonise/>

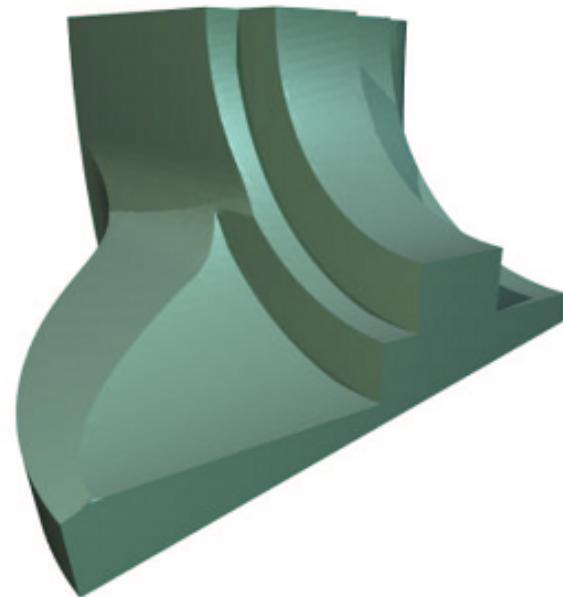
Regularised marching tetrahedra: improved iso-surface extraction [Treece C&G99]

# シャープなエッジを保持した等値面抽出

格子サイズ：65×65×65

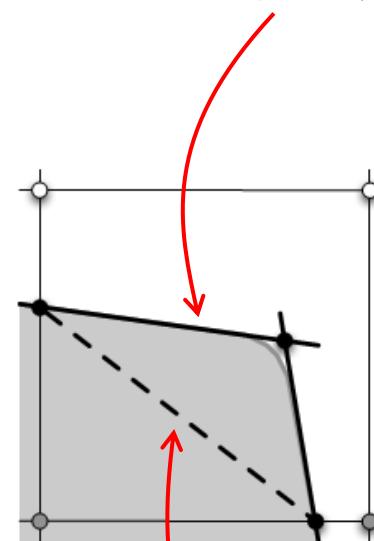


Marching Cubes

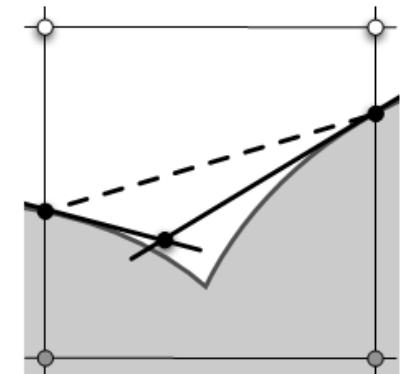


改善版

改善版 (陰関数の勾配も考慮)



改善版 (陰関数の値のみ考慮)



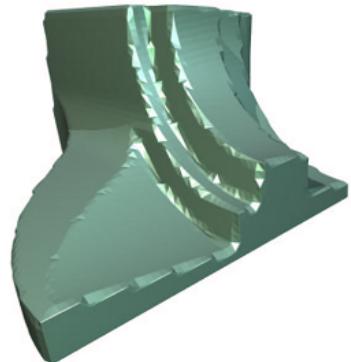
Feature Sensitive Surface Extraction from Volume Data [Kobbelt SIGGRAPH01]

Dual Contouring of Hermite Data [Ju SIGGRAPH02]

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

# サーフェスマッシュ表現のみに基づく CSG

- ・ボリューム表現 (=Marching Cubesによる等値面抽出)  
→ 近似精度が格子の向きや解像度に依存 ☹

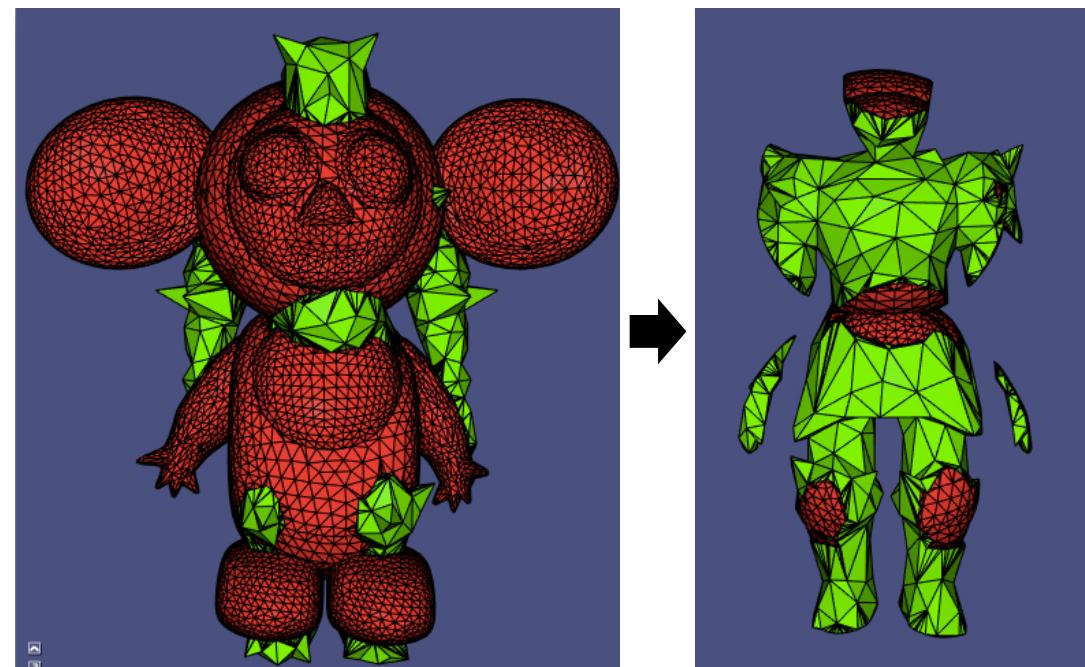


- ・サーフェスマッシュ表現による CSG  
→ 元のメッシュの形状を確実に保持 ☺

- ・ロバストで効率的な実装が難しい ☹
  - ・浮動小数の丸め誤差
  - ・厳密に同じ位置で重複する複数の三角形



- ・ここ数年で著しく進化



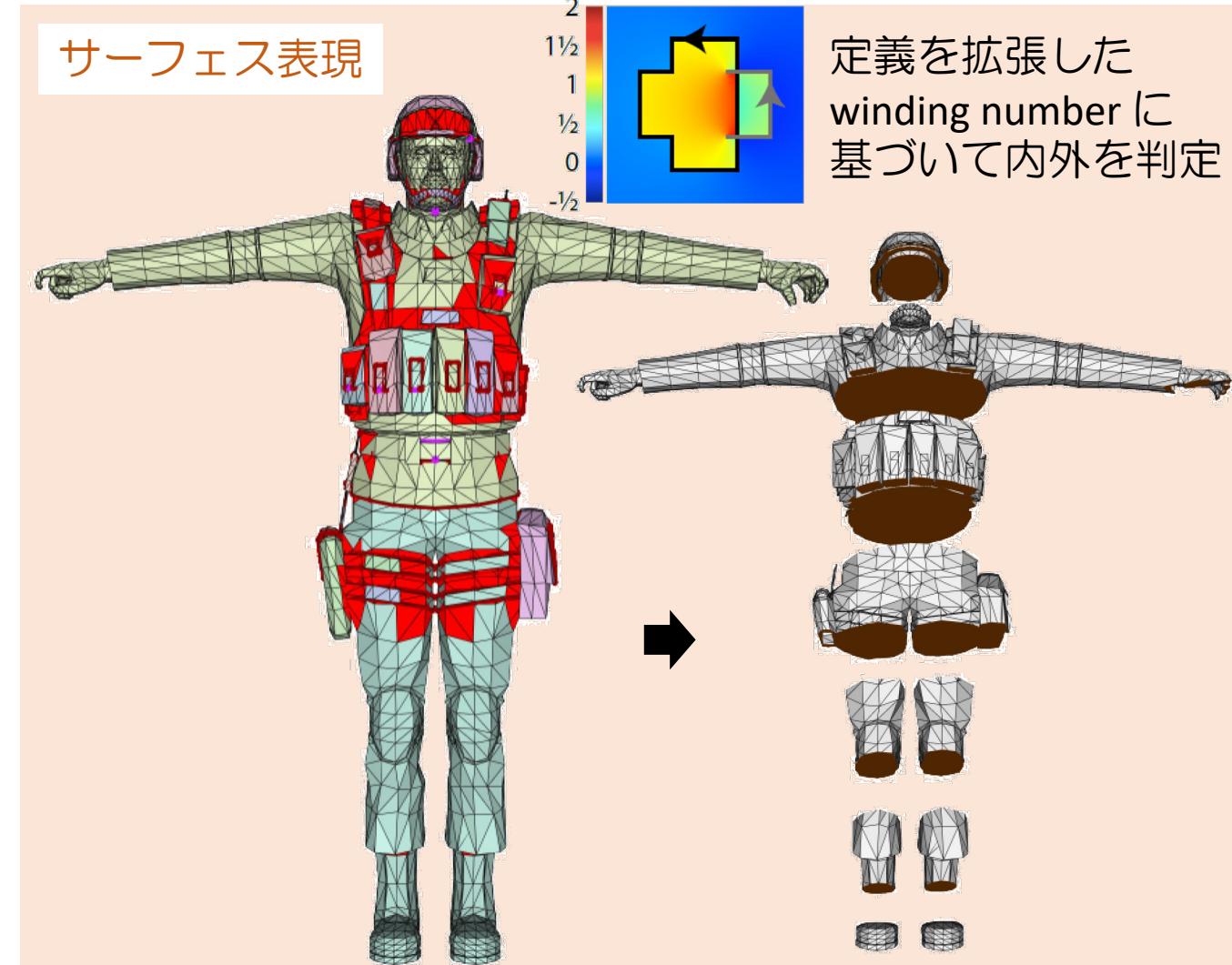
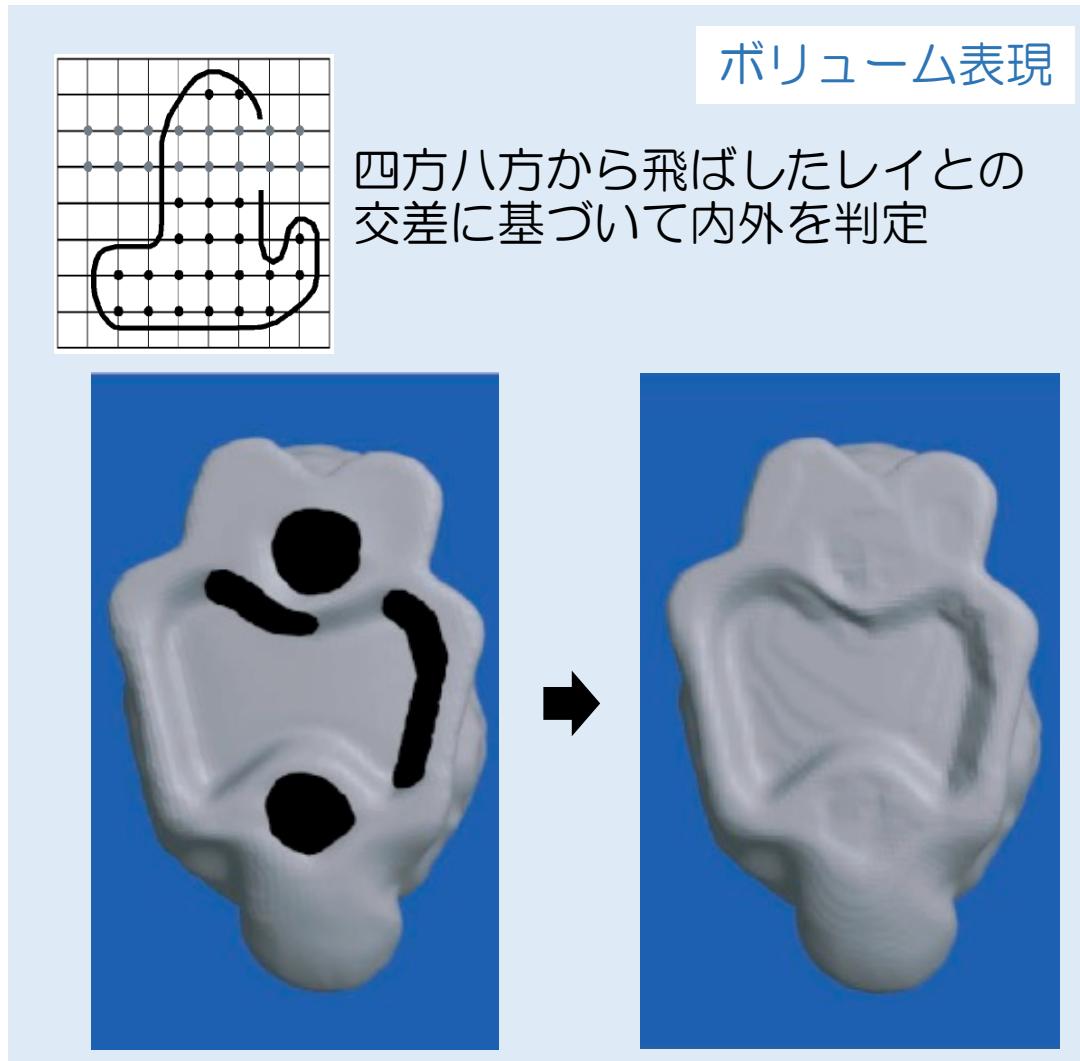
Fast, exact, linear booleans [Bernstein SGP09]

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

Mesh Arrangements for Solid Geometry [Zhou SIGGRAPH16]

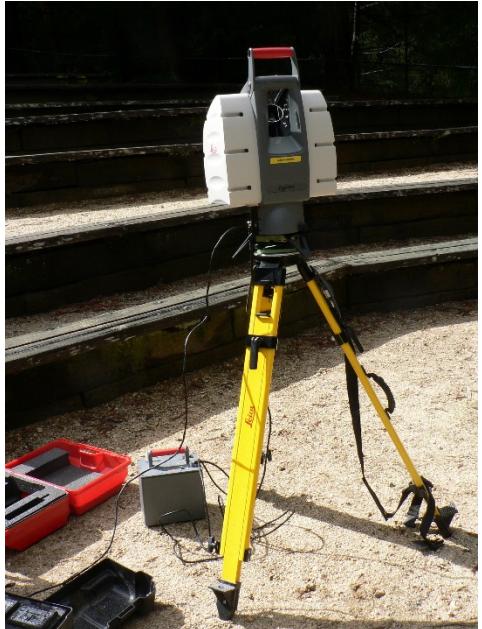
<https://libigl.github.io/libigl/tutorial/tutorial.html#booleanoperationsonmeshes>

# メッシュの補修 (mesh repair)



# 点群からのサーフェス再構成

# 3D 形状の計測

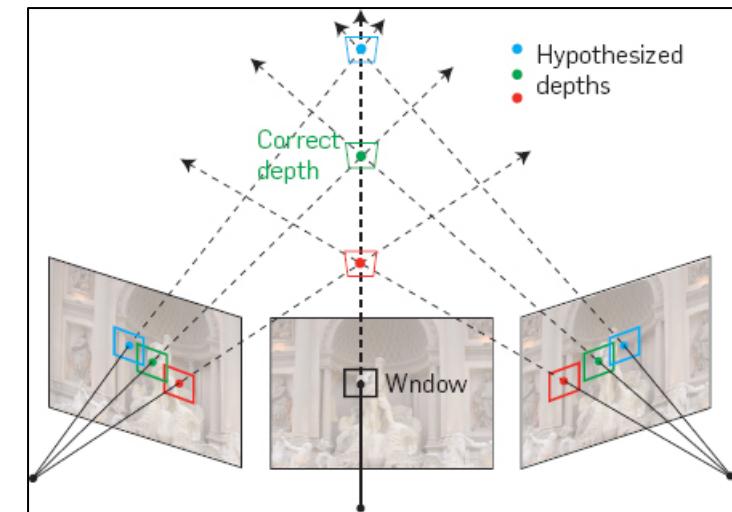
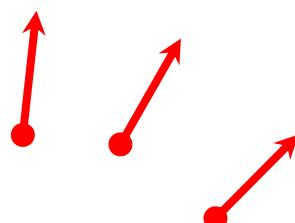


Range Scanner  
(LIDAR)



Depth Camera

- 得られるデータ：点群
  - 3D座標
  - 法線(面の向き)
- 法線が得られない場合 → 法線の推定
- ノイズが多くすぎる場合 → ノイズの除去

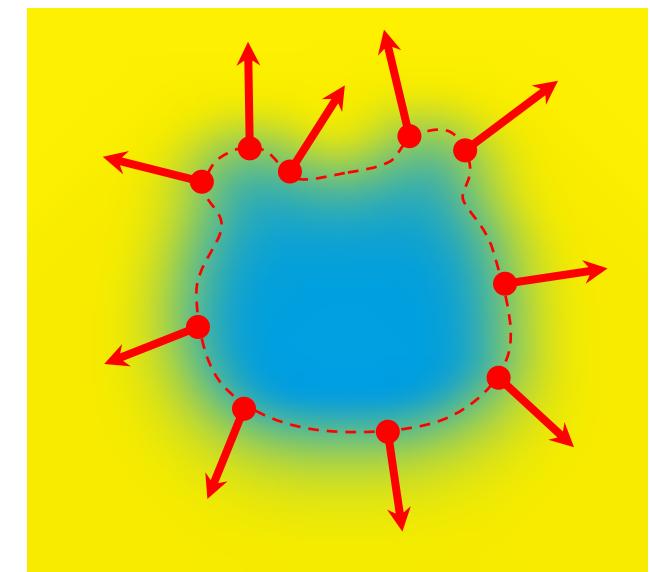


} Computer Vision の代表的なテーマ

# 点群からのサーフェス形状再構成

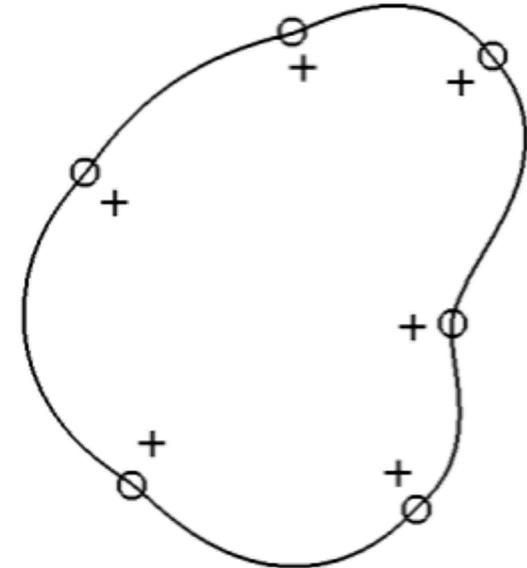
- 入力： $N$  個の点群データ
  - 座標  $\mathbf{x}_i = (x_i, y_i, z_i)$  と法線  $\mathbf{n}_i = (n_i^x, n_i^y, n_i^z), i \in \{1, \dots, N\}$
- 出力：関数  $f(\mathbf{x})$  で、値と勾配の制約を満たすもの
  - $f(\mathbf{x}_i) = f_i$
  - $\nabla f(\mathbf{x}_i) = \mathbf{n}_i$
  - 等値面  $f(\mathbf{x}) = 0$  が出力サーフェス形状
- “Scattered Data Interpolation” と呼ばれる問題
  - Moving Least Squares
  - Radial Basis Function

CG以外の分野 (e.g. 機械学習) でも重要

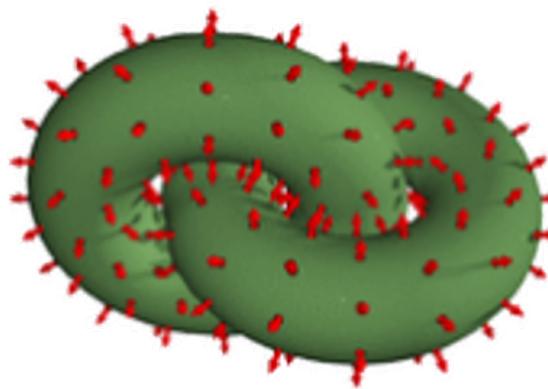


# 勾配を制約する二通りの方法

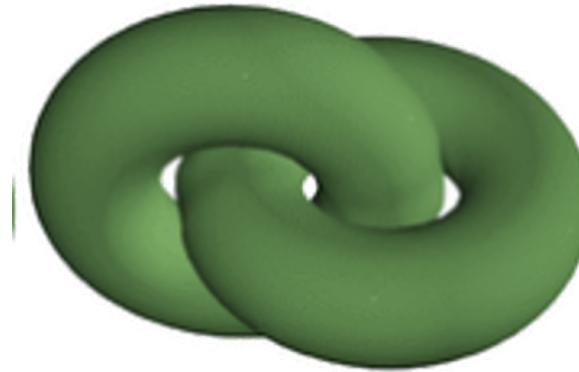
- 法線方向にオフセットした位置に値の制約を追加
  - 簡単



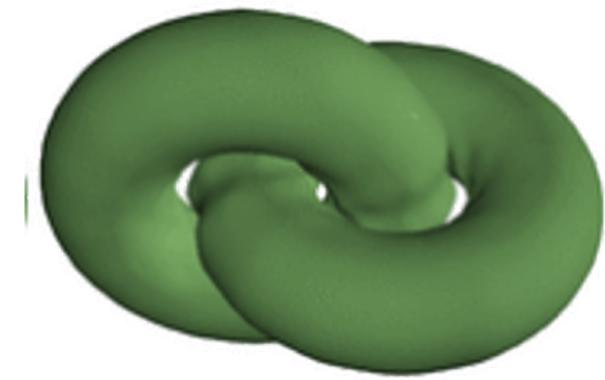
- 数学表現そのものに勾配制約を取り入れる(エルミート補間)
  - 高品質



値と勾配の制約



エルミート補間



オフセット法

# Moving Least Squares による補間 (移動最小二乗)

# 出発点：Least SQuares (最小二乗)

- 求めたい関数が線形だと仮定する： $f(\mathbf{x}) = ax + by + cz + d$ 
  - $a, b, c, d$  が未知係数

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

- データ点における値の制約

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

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

 $\vdots$  $\vdots$  $\vdots$ 

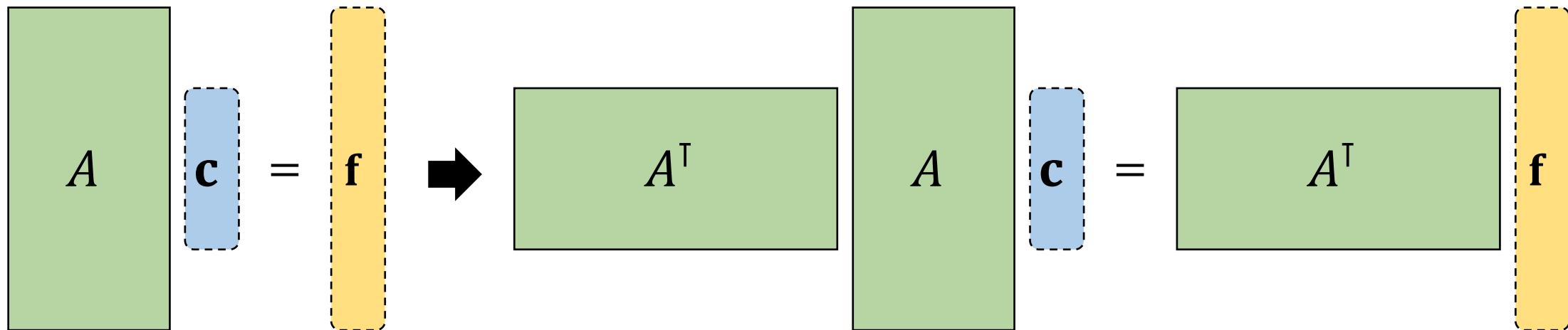
$$f(\mathbf{x}_N) = ax_N + by_N + cz_N + d = f_N$$

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

- (勾配制約は今は考えない)

# Overconstrained System

- #未知数 < #制約 (i.e. 縦長の行列) → 全ての制約を同時に満たせない



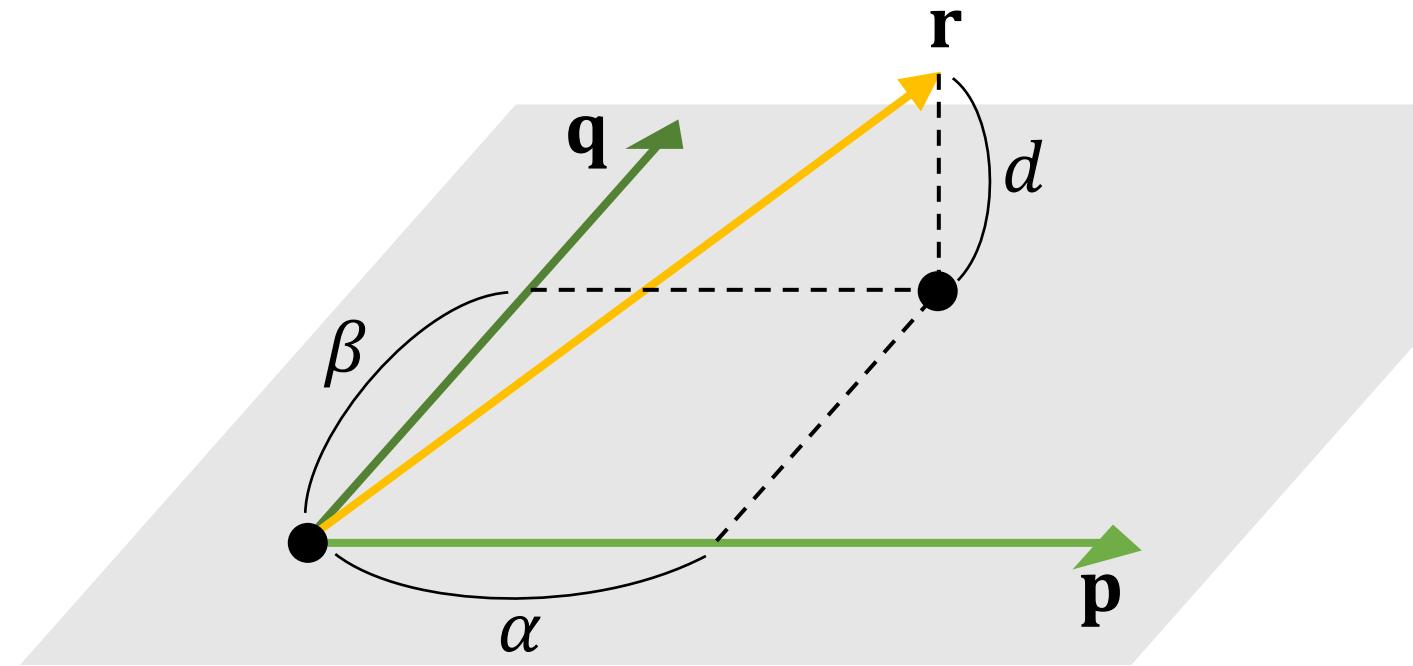
- fitting の誤差を最小化 :

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

$$\mathbf{c} = \begin{pmatrix} (A^\top A)^{-1} \\ A^\top \end{pmatrix} \begin{pmatrix} \mathbf{f} \end{pmatrix}$$

# LSQ の幾何的な解釈

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



- $\mathbf{p}$  と  $\mathbf{q}$  が張る空間中で  $\mathbf{r}$  に最も近い点を求める (投影する) ことに相当
  - fitting 誤差は投影距離に相当 :

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

# Weighted Least Squares (重み付き最小二乗)

- 各データ点ごとの誤差に、重み  $w_i$  をつける
  - 重要度、確信度
- 以下の誤差を最小化：

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

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

# Weighted Least Squares (重み付き最小二乗)

$$\begin{matrix} W \\ A \end{matrix} \quad \boxed{\mathbf{c}} = \begin{matrix} W \\ A \end{matrix} \quad \boxed{\mathbf{f}}$$

$$\Rightarrow \boxed{\mathbf{c}} = \begin{matrix} (A^\top W^2 A)^{-1} \\ A^\top W^2 \end{matrix} \quad \boxed{\mathbf{f}}$$

# Moving Least Squares (移動最小二乗)

- 重み  $w_i$  が、評価位置  $\mathbf{x}$  に依存：

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

- よく使われる関数 (Kernel) :

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

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

評価位置に近いほど  
大きな重み

- 重み行列  $W$  が  $\mathbf{x}$  に依存

→ 系数  $a, b, c, d$  が  $\mathbf{x}$  に依存

$$f(\mathbf{x}) = [x \ y \ z \ 1]$$

- 評価点ごとに  
方程式を解き直す

$$\begin{matrix} a(\mathbf{x}) \\ b(\mathbf{x}) \\ (A^\top W(\mathbf{x})^2 A)^{-1} \\ c(\mathbf{x}) \\ d(\mathbf{x}) \end{matrix} \quad A^\top W(\mathbf{x})^2 \quad \mathbf{f}$$

# 法線制約の導入

- 各データ点が表す1次式を考える：

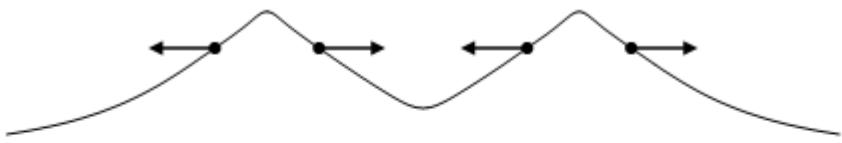
$$g_i(\mathbf{x}) = f_i + (\mathbf{x} - \mathbf{x}_i)^\top \mathbf{n}_i$$

- 各  $g_i$  を現在位置で評価したときの誤差を最小化：

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

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

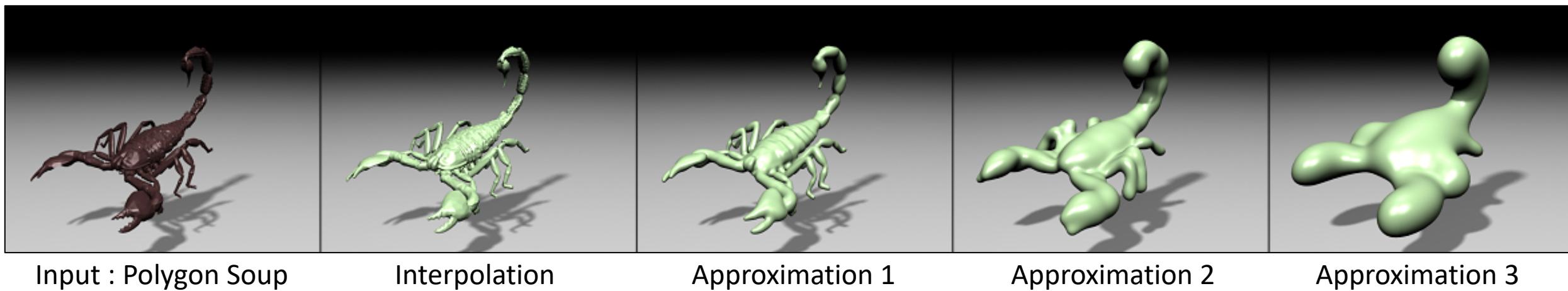
# 法線制約の導入



法線制約を利用



法線方向にオフセットして値を制約



Input : Polygon Soup

Interpolation

Approximation 1

Approximation 2

Approximation 3

# Radial Basis Function による補間 (放射基底関数)

# 基本的な考え方

- 関数  $f(\mathbf{x})$  を、基底関数  $\phi(\mathbf{x})$  の重み付き和として定義：

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

基底関数をデータ位置  
 $\mathbf{x}_i$  に平行移動

(Radial Basis Function)

- 放射基底関数  $\phi(\mathbf{x})$  :  $\mathbf{x}$  の長さのみに依存

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

- $\phi(\mathbf{x}) = \frac{1}{\sqrt{\|\mathbf{x}\|^2 + c^2}}$  (Inverse Multiquadric)

- 各データ点における制約  $f(\mathbf{x}_i) = f_i$  から、重み係数  $w_i$  を求める

# 基本的な考え方

$\phi_{i,j} = \phi(\mathbf{x}_i - \mathbf{x}_j)$  と表記する

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

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

•  
•  
•

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

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

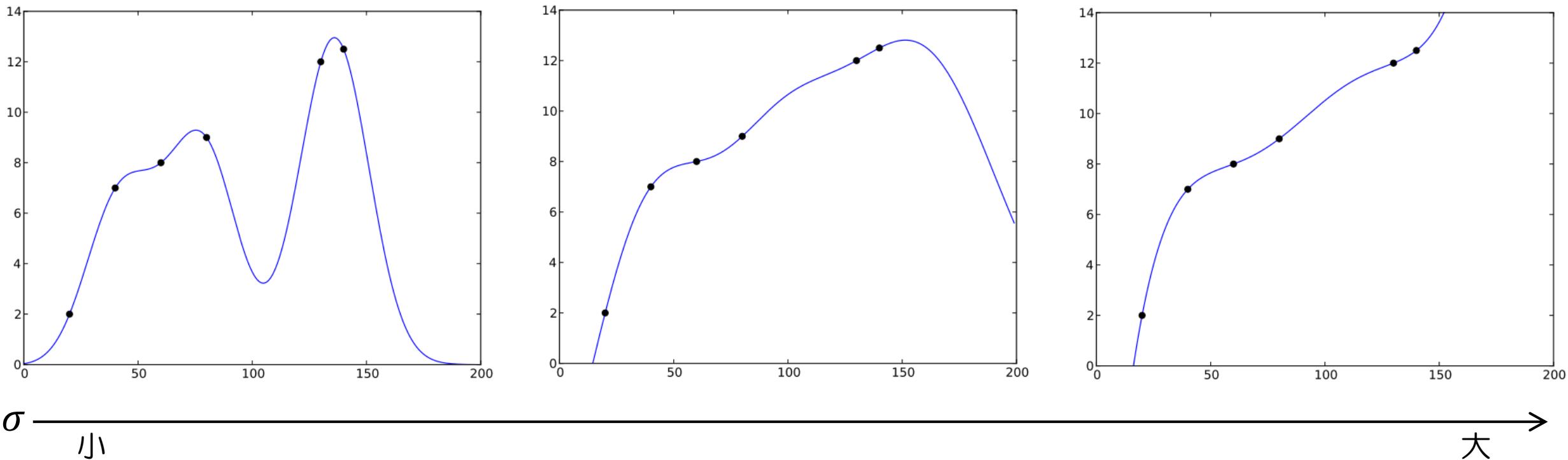
$$\begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,N} \\ \phi_{2,1} & \phi_{2,2} & \phi_{2,N} \\ \vdots & \ddots & \vdots \\ \phi_{N,1} & \phi_{N,2} & \phi_{N,N} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \mathbf{w} \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \mathbf{f} \\ \vdots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

これを解けば良い

# Gaussian 基底関数を使う場合

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

- パラメタ  $\sigma$  の選び方によって、結果が大きく変わる！



- なるべく滑らかな結果を得るには？

# 関数の滑らかさの尺度

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

2D

$$\nabla^2 f(x, y) := (f_{xx}, f_{xy}, f_{yx}, f_{yy})$$

$$E_2[f] := \int_{\mathbb{R}^2} f_{xx}^2 + f_{yy}^2 + 2f_{xy}^2 \quad \text{"Thin-plate" energy}$$

$m = 2$

3D

$$\nabla^2 f(x, y, z) := (f_{xx}, f_{xy}, f_{xz}, f_{yx}, f_{yy}, f_{yz}, f_{zx}, f_{zy}, f_{zz})$$

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

$m = 3$

$$\nabla^3 f(x, y) := (f_{xxx}, f_{xxy}, f_{xyx}, f_{xyy}, f_{yxx}, f_{yxy}, f_{yyx}, f_{yyy})$$

$$E_3[f] := \int_{\mathbb{R}^2} f_{xxx}^2 + f_{yyy}^2 + 2(f_{xxy}^2 + f_{yyx}^2)$$

$$\nabla^3 f(x, y) := \begin{pmatrix} f_{xxx}, f_{xxy}, f_{xxz}, f_{xyx}, f_{xyy}, f_{xyz}, f_{xzx}, f_{xzy}, f_{xzz}, \\ f_{yxx}, f_{yxy}, f_{yxz}, f_{yyx}, f_{yyy}, f_{yyz}, f_{yzx}, f_{yzy}, f_{yzz}, \\ f_{zxx}, f_{zxy}, f_{zxz}, f_{zyx}, f_{zyy}, f_{zyz}, f_{zzx}, f_{zzy}, f_{zzz} \end{pmatrix}$$

$$E_3[f] := \int_{\mathbb{R}^3} f_{xxx}^2 + f_{yyy}^2 + f_{zzz}^2 + 3(f_{xxy}^2 + f_{yyz}^2 + f_{zzx}^2 + f_{xyy}^2 + f_{yzz}^2 + f_{zxx}^2) + f_{xyz}^2$$

# 偉大な発見 (Duchon 1977)

- 制約  $\{f(\mathbf{x}_i) = f_i\}$  を満たす関数全体のうち、 $E_m[f]$  を最小化するものは以下のRBF基底を使って表せる！
  - 空間次元数が奇数の場合 :  $\phi(\mathbf{x}) = \|\mathbf{x}\|^{2m-3}$
  - 空間次元数が偶数の場合 :  $\phi(\mathbf{x}) = \|\mathbf{x}\|^{2m-2} \log \|\mathbf{x}\|$ 
    - ただし  $\phi(0) = 0$  とする
- よく使われるもの：
  - 2Dの場合 :  $\phi(\mathbf{x}) = \|\mathbf{x}\|^2 \log \|\mathbf{x}\|$  ( $E_2$ を最小化)
  - 3Dの場合 :  $\phi(\mathbf{x}) = \|\mathbf{x}\|^3$  ( $E_3$ を最小化)

# 線形項の追加

- $E_m[f]$  (ただし  $m \geq 2$ ) は 2 階以上の微分を使って定義される  
→ 任意の線形項  $p(\mathbf{x}) = \textcolor{blue}{a}x + \textcolor{blue}{b}y + \textcolor{blue}{c}z + \textcolor{blue}{d}$  を加えても不变：

$$E_m[f + p] = E_m[f] \quad \forall f$$

- 線形項を未知数に含めることで、関数を一意に定める：

$$f(\mathbf{x}) = \sum_{i=1}^N \textcolor{blue}{w}_i \phi(\mathbf{x} - \mathbf{x}_i) + \textcolor{blue}{a}x + \textcolor{blue}{b}y + \textcolor{blue}{c}z + \textcolor{blue}{d}$$

# 線形項の追加

$$f(\mathbf{x}_1) = \textcolor{blue}{w_1}\phi_{1,1} + \textcolor{blue}{w_2}\phi_{1,2} + \cdots + \textcolor{blue}{w_N}\phi_{1,N} + \textcolor{blue}{a}x_1 + \textcolor{blue}{b}y_1 + \textcolor{blue}{c}z_1 + \textcolor{blue}{d} = f_1$$

$$f(\mathbf{x}_2) = \textcolor{blue}{w_1}\phi_{2,1} + \textcolor{blue}{w_2}\phi_{2,2} + \cdots + \textcolor{blue}{w_N}\phi_{2,N} + \textcolor{blue}{a}x_2 + \textcolor{blue}{b}y_2 + \textcolor{blue}{c}z_2 + \textcolor{blue}{d} = f_2$$

•  
•  
•

$$f(\mathbf{x}_N) = \textcolor{blue}{w_1}\phi_{N,1} + \textcolor{blue}{w_2}\phi_{N,2} + \cdots + \textcolor{blue}{w_N}\phi_{N,N} + \textcolor{blue}{a}x_N + \textcolor{blue}{b}y_N + \textcolor{blue}{c}z_N + \textcolor{blue}{d} = f_N$$

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

4 個の未知数  $a, b, c, d$   
が追加されたので、  
4 個の制約を追加する  
必要がある

# 追加の制約条件：線形関数の再現性

- 「全てのデータ点の制約  $(\mathbf{x}_i, f_i)$  がある線形関数からのサンプリングであるとき、RBF による補間結果はその線形関数と一致する」

- これを満たすための条件：

- $\sum_{i=1}^N w_i = 0$

- $\sum_{i=1}^N x_i w_i = 0$

- $\sum_{i=1}^N y_i w_i = 0$

- $\sum_{i=1}^N z_i w_i = 0$

- 行列がちょうど対称になる

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

# 勾配制約の導入

- 基底関数の勾配  $\nabla\phi$  の重み付き和を導入：

$$f(\mathbf{x}) = \sum_{i=1}^N \left\{ \mathbf{w}_i \phi(\mathbf{x} - \mathbf{x}_i) + \mathbf{v}_i^\top \nabla\phi(\mathbf{x} - \mathbf{x}_i) \right\} + \mathbf{a}x + \mathbf{b}y + \mathbf{c}z + \mathbf{d}$$

未知数の3Dベクトル

- $f$  の勾配：

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

- 勾配の制約  $\nabla f(\mathbf{x}_i) = \mathbf{n}_i$  を追加

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

Hessian 行列 (関数)

# 勾配制約の導入

- 1番目のデータ点について：

値の制約：

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

勾配の制約：

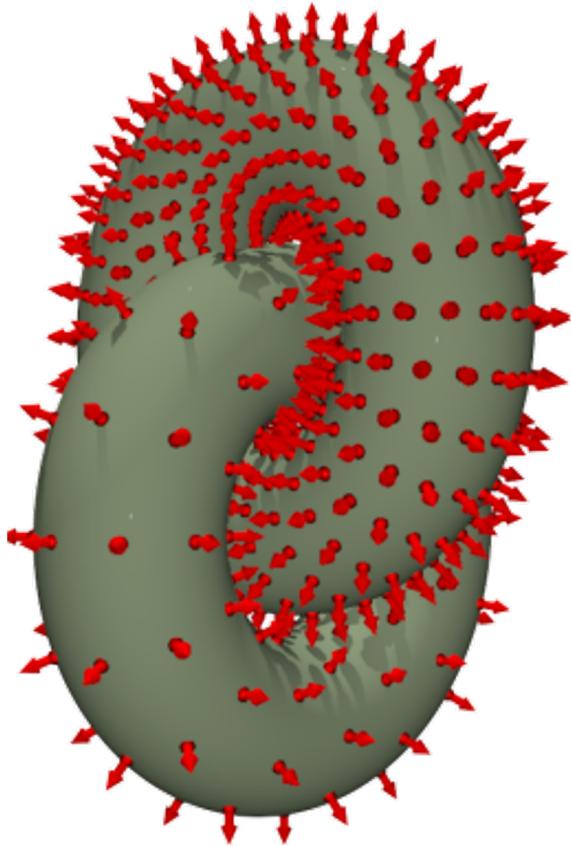
$$\nabla f(\mathbf{x}_1) = \mathbf{w}_1 \nabla \phi_{1,1} + H_\phi^{1,1} \mathbf{v}_1 + \mathbf{w}_2 \nabla \phi_{1,2} + H_\phi^{1,2} \mathbf{v}_2 + \cdots + \mathbf{w}_N \nabla \phi_{1,N} + H_\phi^{1,N} \mathbf{v}_N$$

$$\left[ \begin{array}{c|c} \phi_{1,1} & (\nabla \phi_{1,1})^\top \\ \hline \nabla \phi_{1,1} & \Phi_{1,1} \end{array} \quad \begin{array}{c|c} \phi_{1,2} & (\nabla \phi_{1,2})^\top \\ \hline \nabla \phi_{1,2} & \Phi_{1,2} \end{array} \quad \cdots \quad \begin{array}{c|c} \phi_{1,N} & (\nabla \phi_{1,N})^\top \\ \hline \nabla \phi_{1,N} & \Phi_{1,N} \end{array} \quad \begin{array}{c|c} x_1 & y_1 & z_1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right]$$

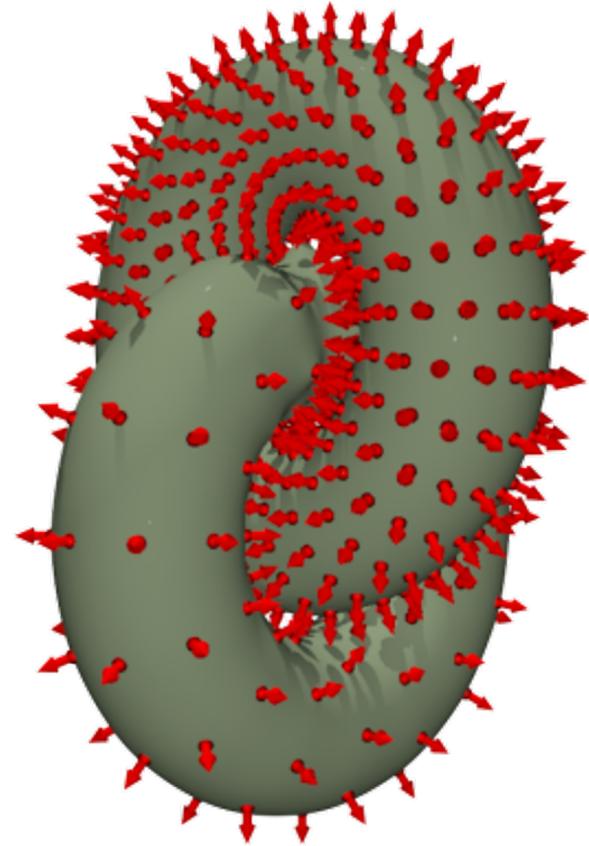
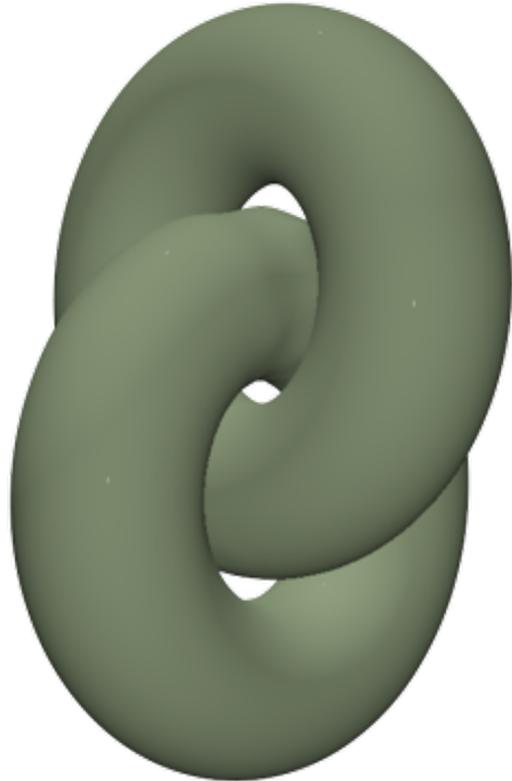
$$\begin{matrix}
 w_1 \\
 \vdots \\
 w_N \\
 \hline
 \mathbf{v}_1 \\
 \vdots \\
 \mathbf{v}_N \\
 \hline
 a \\
 b \\
 c \\
 d
 \end{matrix}
 \leftarrow
 \begin{matrix}
 b y_1 + c z_1 + d = f_1 \\
 \vdots \\
 \mathbf{n}_1
 \end{matrix}
 = 
 \begin{bmatrix}
 f_1 \\
 \mathbf{n}_1
 \end{bmatrix}$$

$$\left[ \begin{array}{cc|cc|cc}
\Phi_{1,1} & \Phi_{1,2} & \cdots & \Phi_{1,N} & P_1 & \\ \hline
\Phi_{2,1} & \Phi_{2,2} & & \Phi_{2,N} & P_2 & \\ \hline
& & \ddots & & \ddots & \\ \hline
\Phi_{N,1} & \Phi_{N,2} & & \Phi_{N,N} & P_N & \\ \hline
P_1^\top & P_2^\top & \cdots & P_N^\top & \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} &
\end{array} \right] = \left[ \begin{array}{c|c}
\begin{matrix} w_1 \\ \mathbf{v}_1 \end{matrix} & f_1 \\ \hline
\begin{matrix} w_2 \\ \mathbf{v}_2 \end{matrix} & f_2 \\ \hline
\vdots & \vdots \\ \hline
\begin{matrix} w_N \\ \mathbf{v}_N \end{matrix} & f_N \\ \hline
\begin{matrix} a \\ b \\ c \\ d \end{matrix} & \mathbf{n}_N \\ \hline
0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0
\end{array} \right]$$

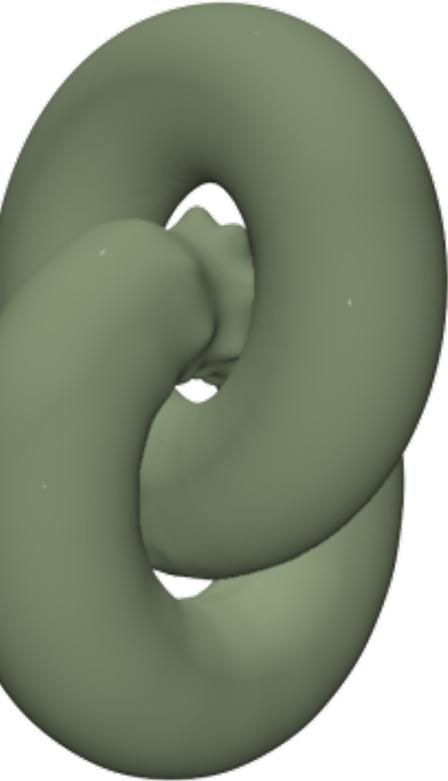
# 比較



勾配を制約



オフセットして値を制約



# 最近の研究：グローバルな法線推定

- サンプル点  $\{\mathbf{x}_i\}$  における法線  $\{\mathbf{n}_i\}$  が未知のとき、

値の制約  $f(\mathbf{x}_i) = 0$

勾配の制約  $\nabla f(\mathbf{x}_i) = \mathbf{n}_i$

を満たすような関数  $f$  を、RBFを使って一意に定めることができる

→ つまり関数は、未知数の法線  $\{\mathbf{n}_i\}$  に依存して定まる： $f_{\{\mathbf{n}_i\}}$

→ その滑らかさを測るエネルギーも、 $\{\mathbf{n}_i\}$  に依存して定まる：

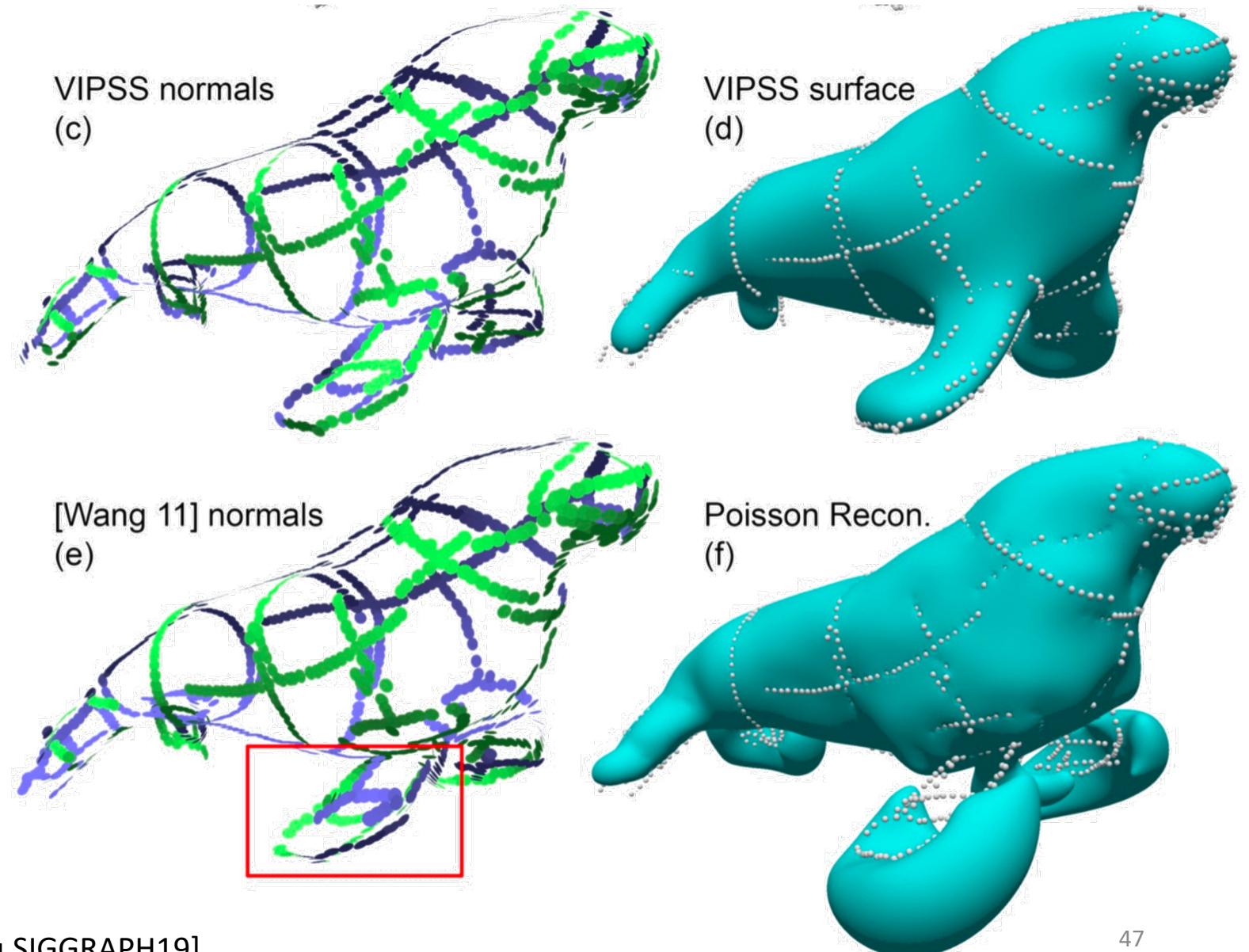
$$\begin{aligned} E_{\{\mathbf{n}_i\}} &:= E[f_{\{\mathbf{n}_i\}}] \\ &= \mathbf{n}^\top H \mathbf{n} \end{aligned} \quad \mathbf{n} = \begin{pmatrix} \vdots \\ \mathbf{n}_i^\top \\ \vdots \end{pmatrix}$$

- 2次制約付き2次計画問題として定式化：

$$\begin{aligned} &\text{minimize } \mathbf{n}^\top H \mathbf{n} \\ \text{s. t. } &\mathbf{n}_i^\top \mathbf{n}_i = 1 \quad \forall i \end{aligned}$$

行列  $H$  は  $\{\mathbf{x}_i\}$  のみに依存して定まる

# 最近の研究：グローバルな法線推定



# 参考サーバイ等

- 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]

# 参考ページ

- [http://en.wikipedia.org/wiki/Implicit\\_surface](http://en.wikipedia.org/wiki/Implicit_surface)
- [http://en.wikipedia.org/wiki/Radial\\_basis\\_function](http://en.wikipedia.org/wiki/Radial_basis_function)
- [http://en.wikipedia.org/wiki/Thin\\_plate\\_spline](http://en.wikipedia.org/wiki/Thin_plate_spline)
- [http://en.wikipedia.org/wiki/Polyharmonic\\_spline](http://en.wikipedia.org/wiki/Polyharmonic_spline)