コンピュータグラフィクス論

– モデリング (3) –

2015年4月30日

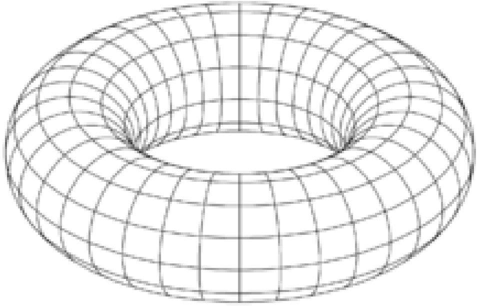
高山 健志

陰関数表現による形状モデリング

- 3D空間上のスカラー場 $F(x, y, z)$ に対し、$F(x, y, z) = 0$ で定義されるサーフェス
  - 別名：ゼロ等値面、陰関数 (ボリューム) 表現
    - ←→ サーフェス表現
- $F(\boldsymbol{p})$ が正 (負) のとき、$\boldsymbol{p}$ は物体の外部 (内部)
- 勾配 ∇F は法線方向に一致
  - $\|\nabla F(\boldsymbol{p})\| = 1\ \forall \boldsymbol{p}$ のとき、F は物体表面からの距離

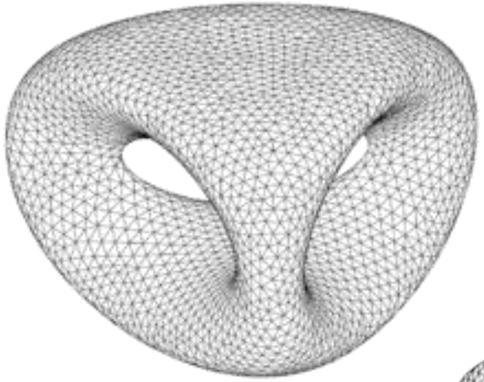
半径 2 の球
$F(x, y, z) = x^2 + y^2 + z^2 - 4 = 0$

陰関数の例

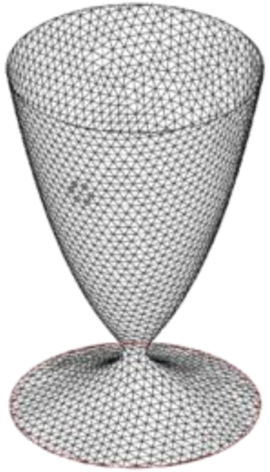


大半径 R, 小半径 a のトーラス

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



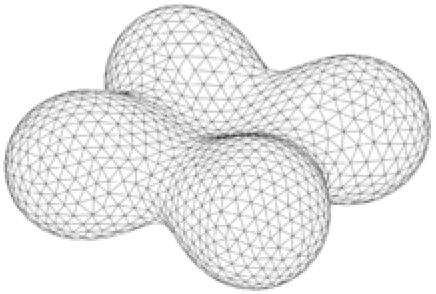
$$2y(y^2 - 3x^2)(1 - z^2) + (x^2 + y^2)^2 - (9z^2 - 1)(1 - z^2) = 0$$



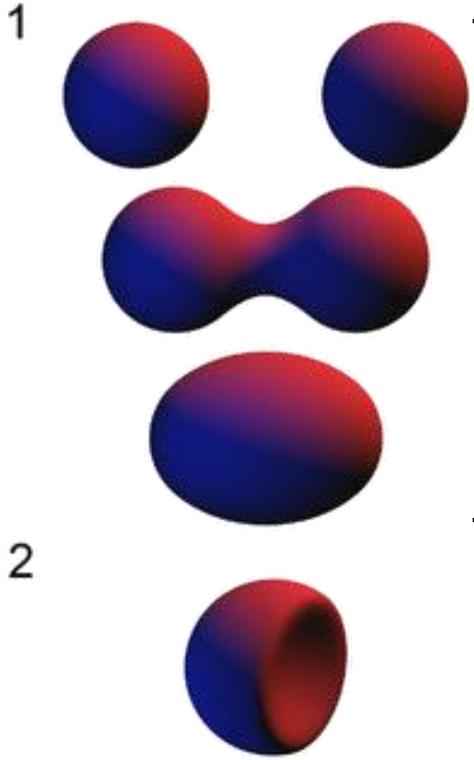
$$x^2 + y^2 - (\ln(z + 3.2))^2 - 0.02 = 0$$

陰関数の例：等電位面 (Metaball)

$$F_i(\boldsymbol{p}) = \frac{q_i}{\|\boldsymbol{p} - \boldsymbol{p}_i\|}$$



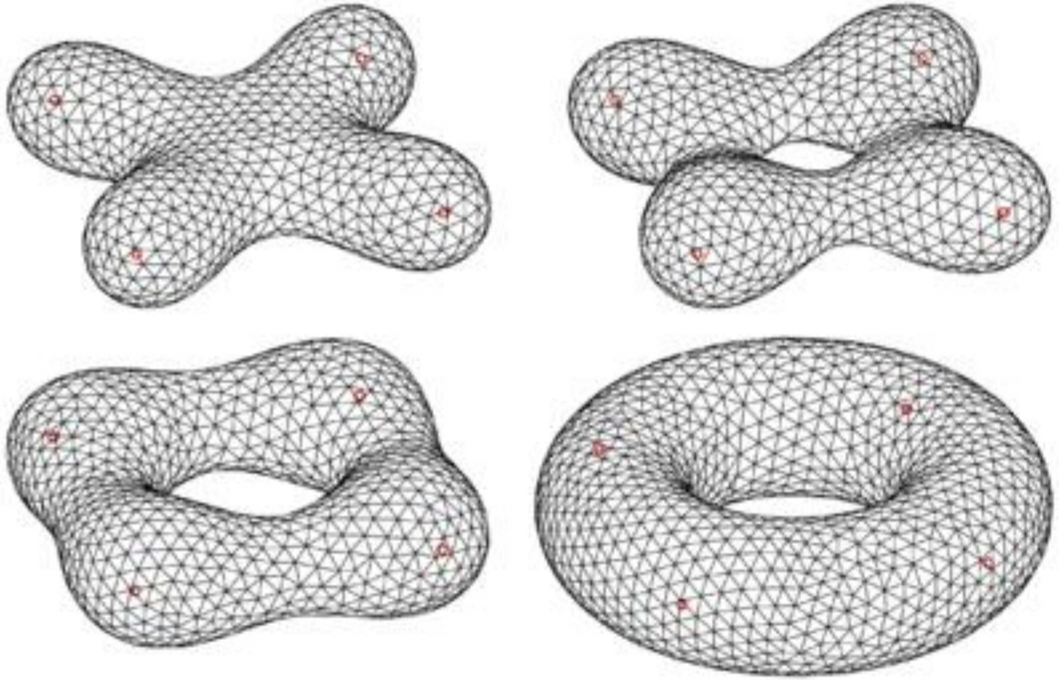
$$F(\boldsymbol{p}) = F_1(\boldsymbol{p}) + F_2(\boldsymbol{p}) + F_3(\boldsymbol{p}) + F_4(\boldsymbol{p})$$



$$F(\boldsymbol{p}) = F_1(\boldsymbol{p}) + F_2(\boldsymbol{p})$$

$$F(\boldsymbol{p}) = F_1(\boldsymbol{p}) - F_2(\boldsymbol{p})$$

陰関数の線形補間によるモーフィング



$$F(\boldsymbol{p}) = F_1(\boldsymbol{p})$$

$$F(\boldsymbol{p}) = \frac{2}{3}F_1(\boldsymbol{p}) + \frac{1}{3}F_2(\boldsymbol{p})$$

$$F(\boldsymbol{p}) = \frac{1}{3}F_1(\boldsymbol{p}) + \frac{2}{3}F_2(\boldsymbol{p})$$

$$F(\boldsymbol{p}) = F_2(\boldsymbol{p})$$

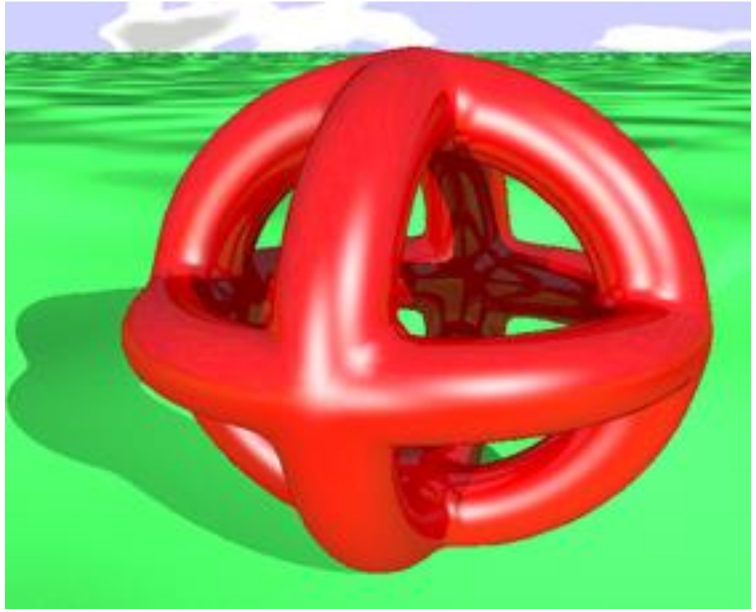
複数の陰関数を組み合わせたモデリング

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

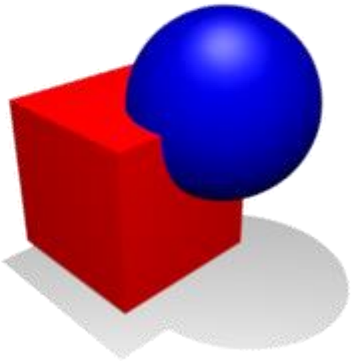
$$F_2 = (x^2 + y^2 + z^2 + R^2 - a^2)^2 - 4R^2(x^2 + z^2) = 0$$

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

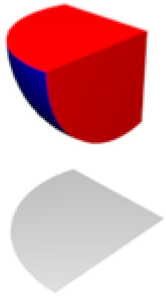
$$F(x, y, z) = F_1(x, y, z) \cdot F_2(x, y, z) \cdot F_3(x, y, z) - c = 0$$



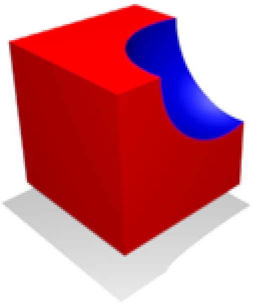
Constructive **S**olid **G**eometry (Boolean演算)



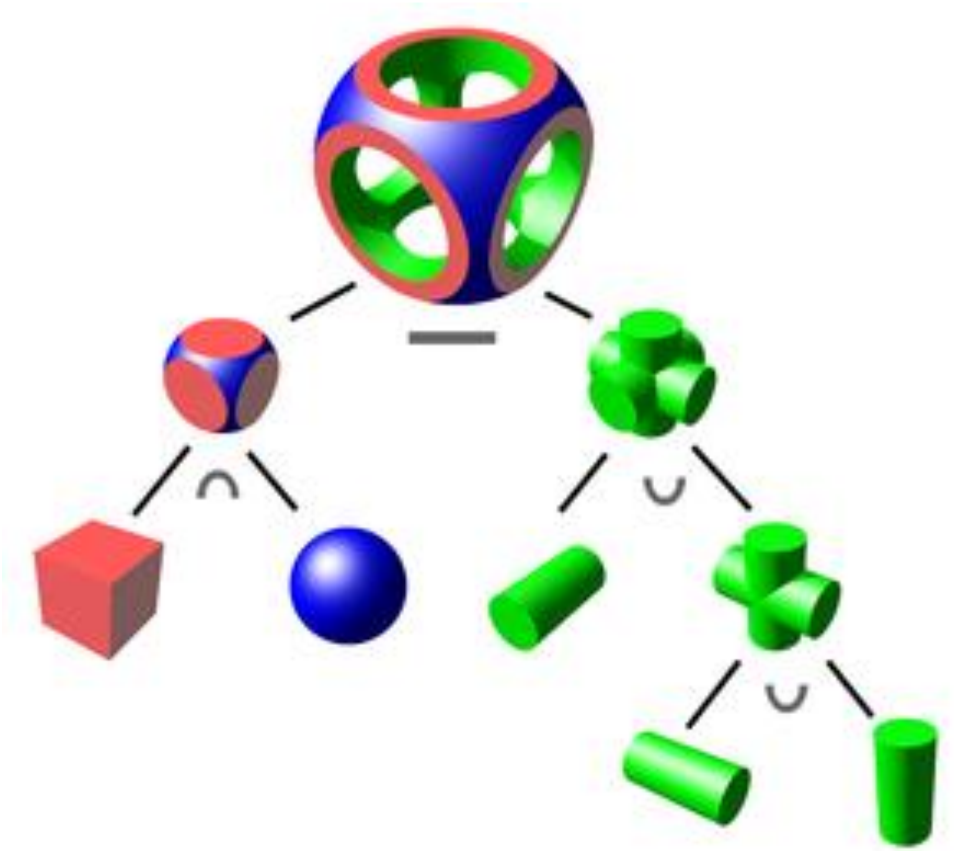
和：$F_{A\cup B}(\boldsymbol{p}) = \max\left(F_A(\boldsymbol{p}), F_B(\boldsymbol{p})\right)$



積：$F_{A\cap B}(\boldsymbol{p}) = \min\left(F_A(\boldsymbol{p}), F_B(\boldsymbol{p})\right)$



差：$F_{A-B}(\boldsymbol{p}) = \min\left(F_A(\boldsymbol{p}), -F_B(\boldsymbol{p})\right)$



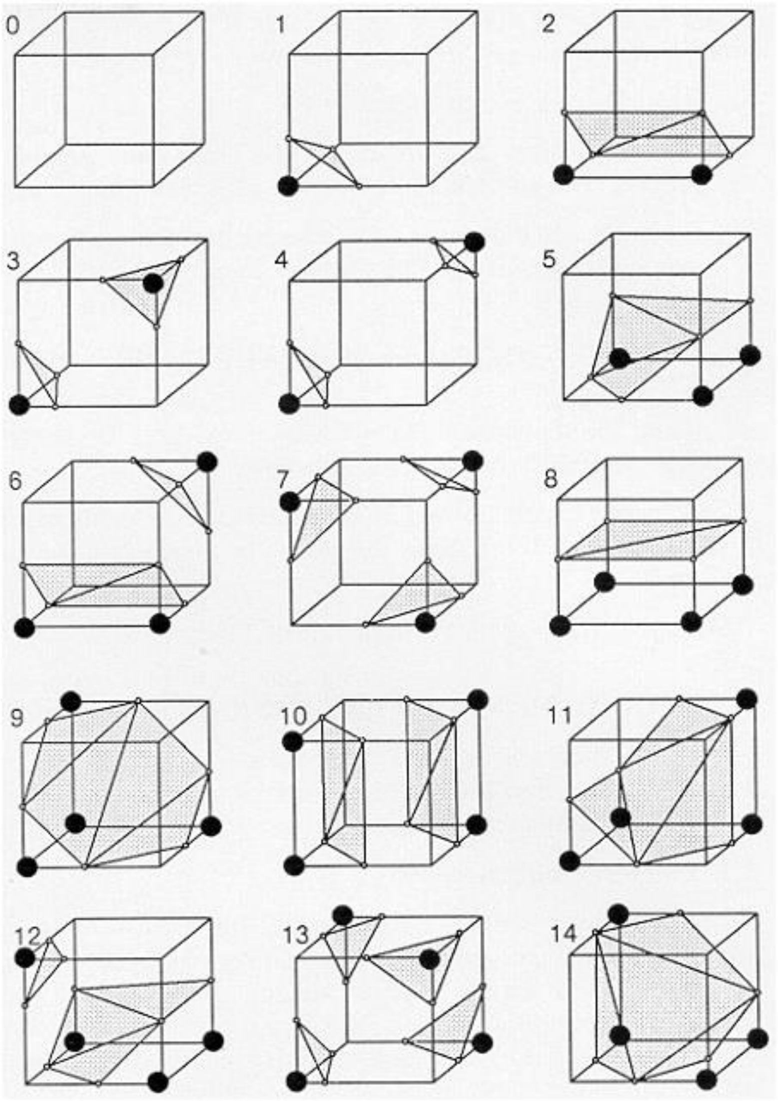
陰関数の表示方法：Marching Cubes

- 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

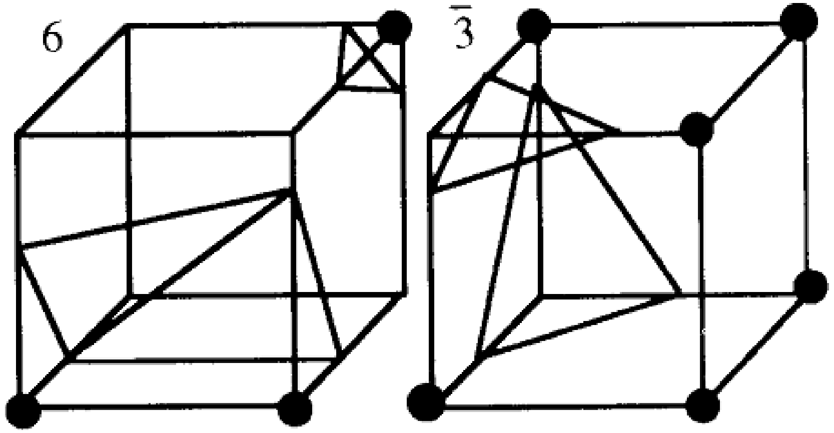
  (3) 関数値の線形補間から
  面の位置を決定

- 最も有名 (特許問題でも ☹)

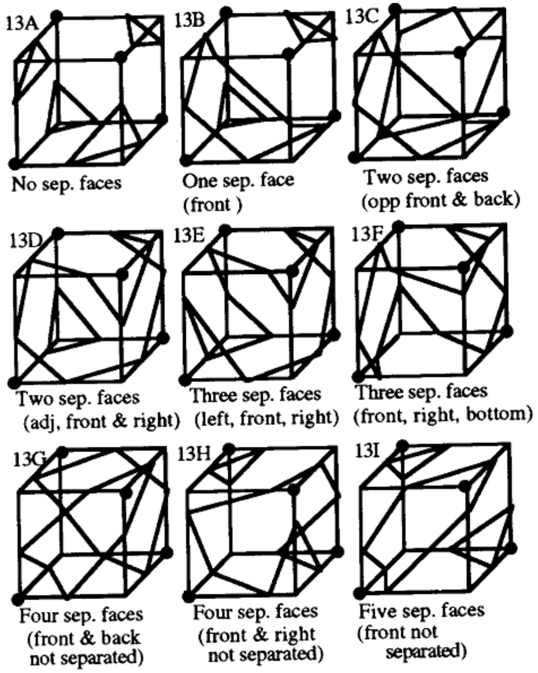


Marching Cubes: A High Resolution 3D Surface Construction Algorithm [Lorensen SIGGRAPH87]

Marching Cubes の曖昧性



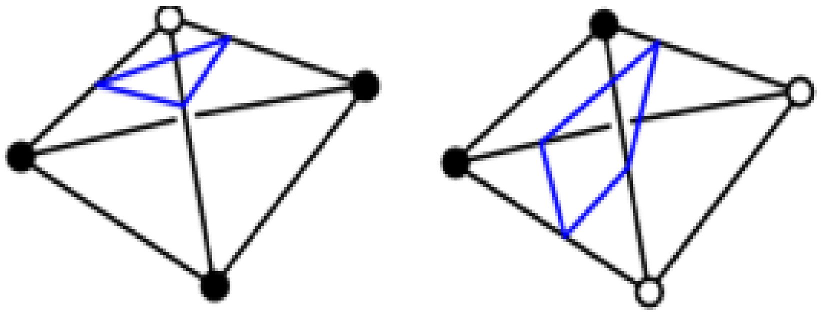
隣接するセルの間で面が整合しない



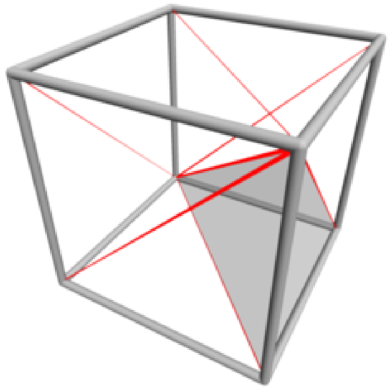
不整合を解決する新たなルール

The asymptotic decider: resolving the ambiguity in marching cubes [Nielson VIS91]

Marching Tetrahedra

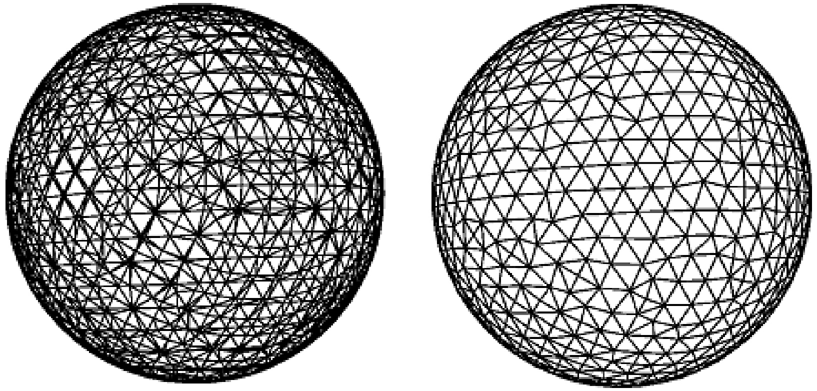


- 立方体の代わりに四面体を使う
  - パターンが少なく、曖昧性が無い
    ➔ 実装が簡単



- 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)

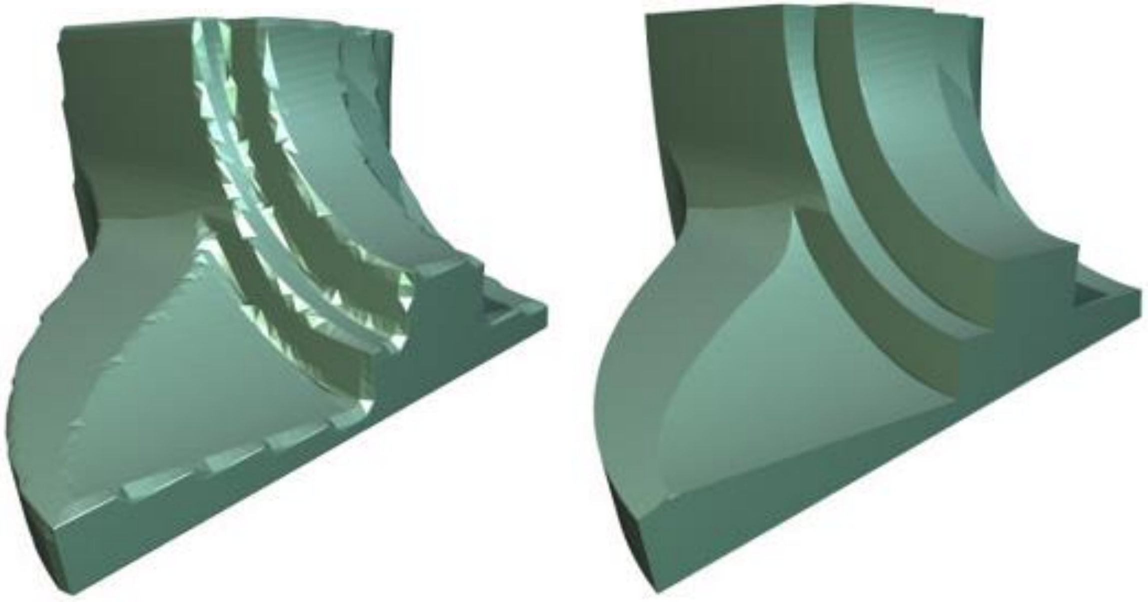
- きれいな三角形メッシュを取り出す工夫



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

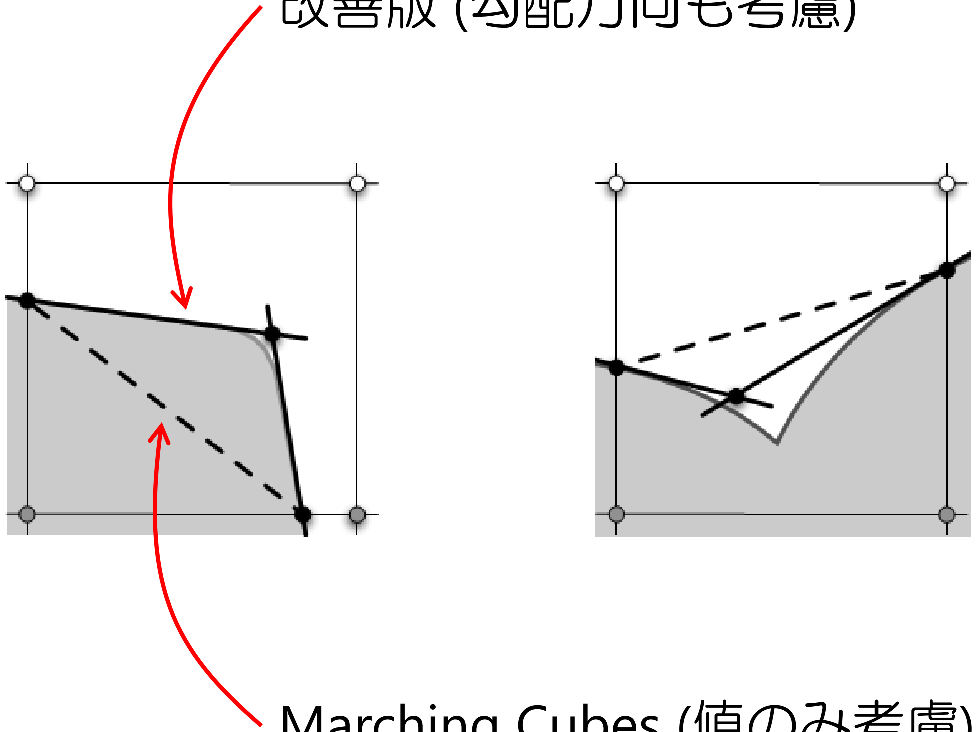
シャープなエッジを保持した等値面抽出

格子サイズ：65×65×65



Marching Cubes　　改善版

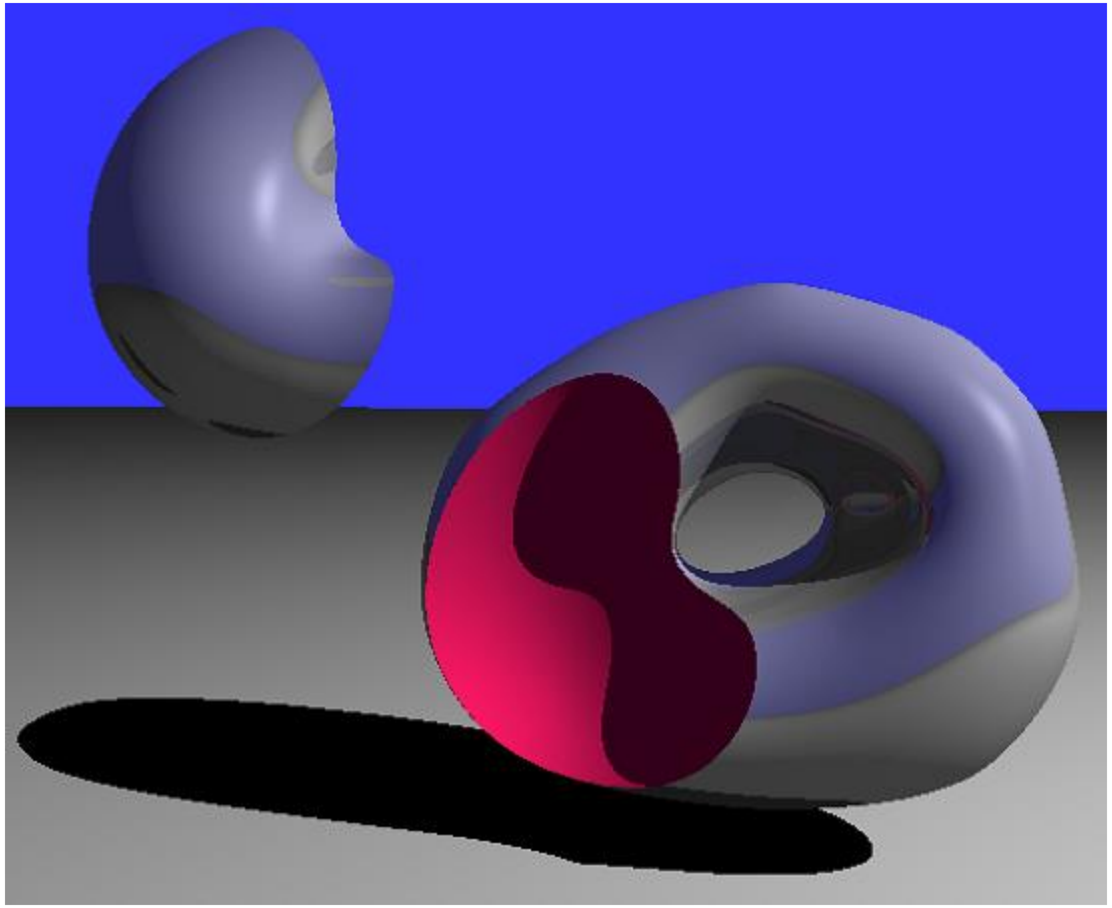
改善版 (勾配方向も考慮)

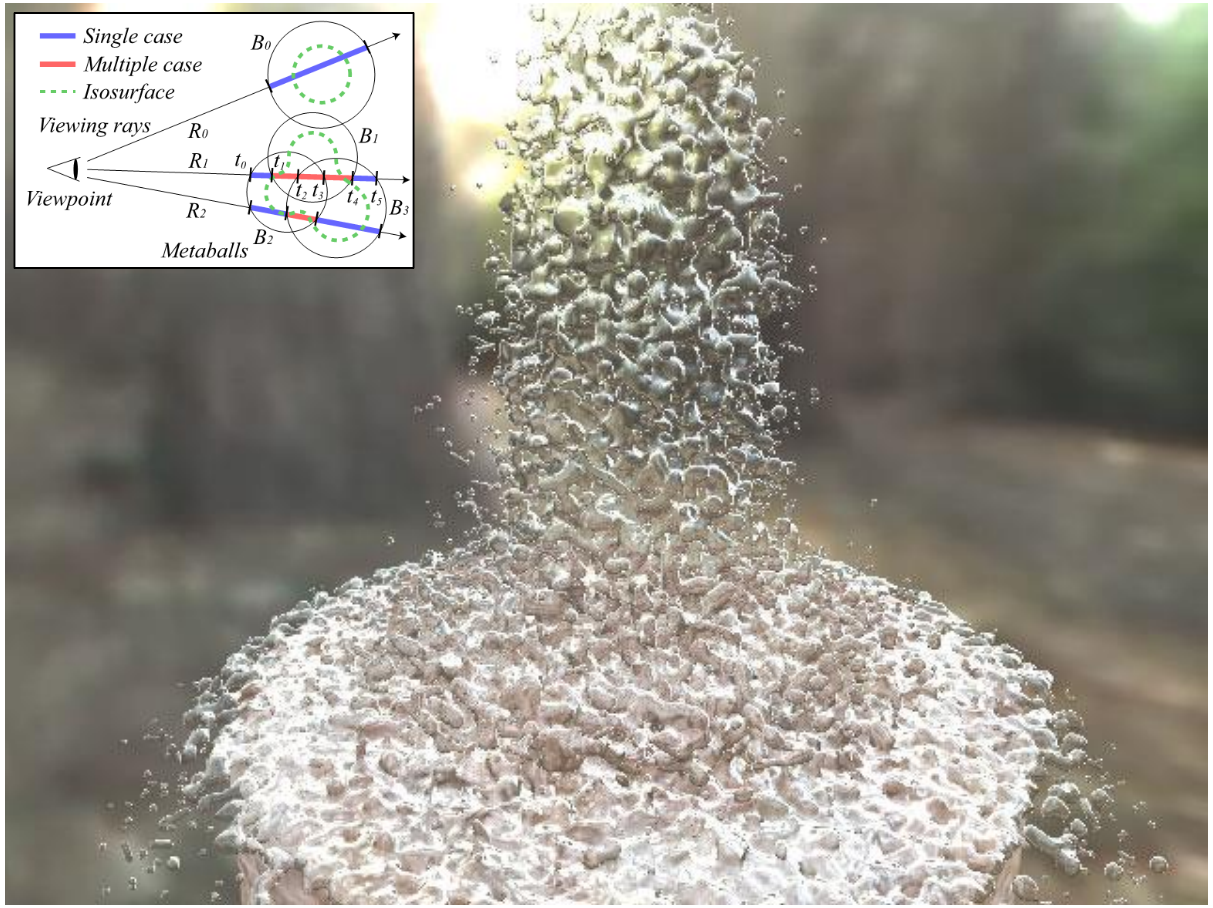


Marching Cubes (値のみ考慮)

Feature Sensitive Surface Extraction from Volume Data [Kobbelt SIGGRAPH01]
http://www.graphics.rwth-aachen.de/IsoEx/

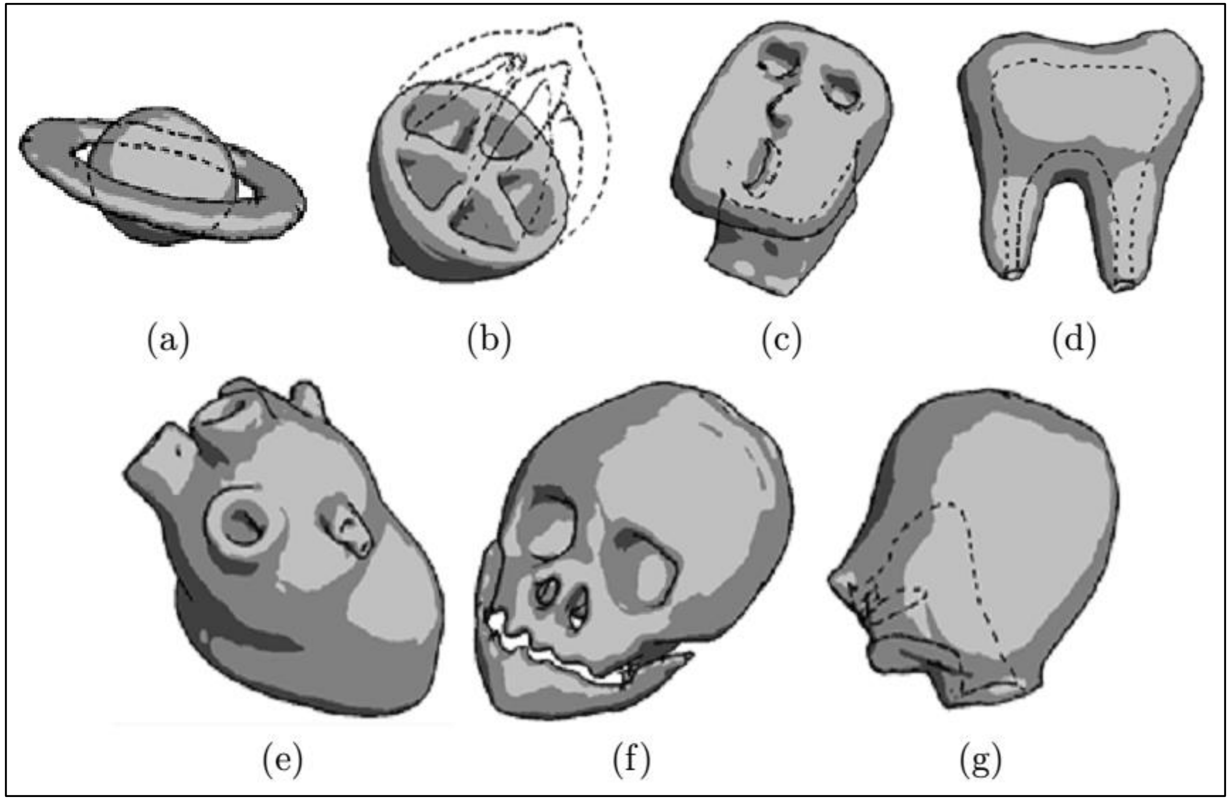
陰関数の表示方法：レイトレーシング





Modelling with implicit surfaces that interpolate [Turk TOG02]
GPU-based Fast Ray Casting for a Large Number of Metaballs [Kanamori EG08]

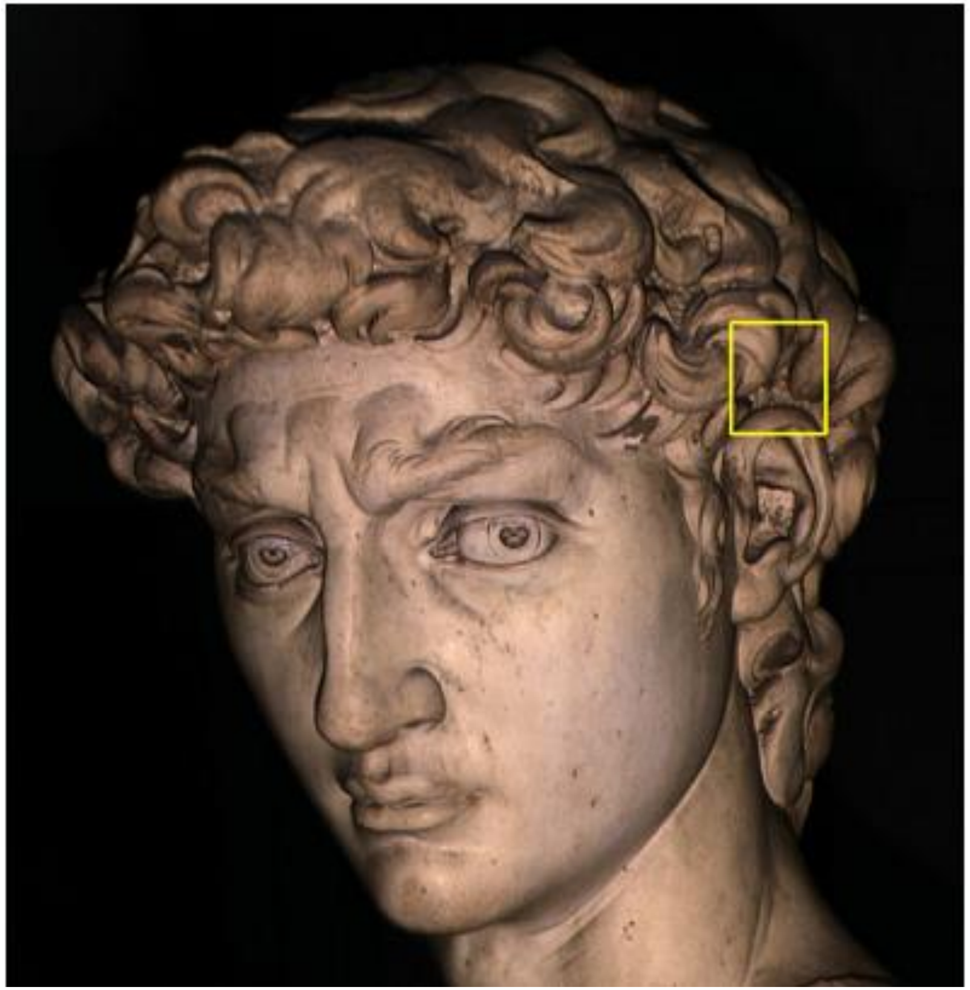
応用例：スケッチベース 3D モデリング

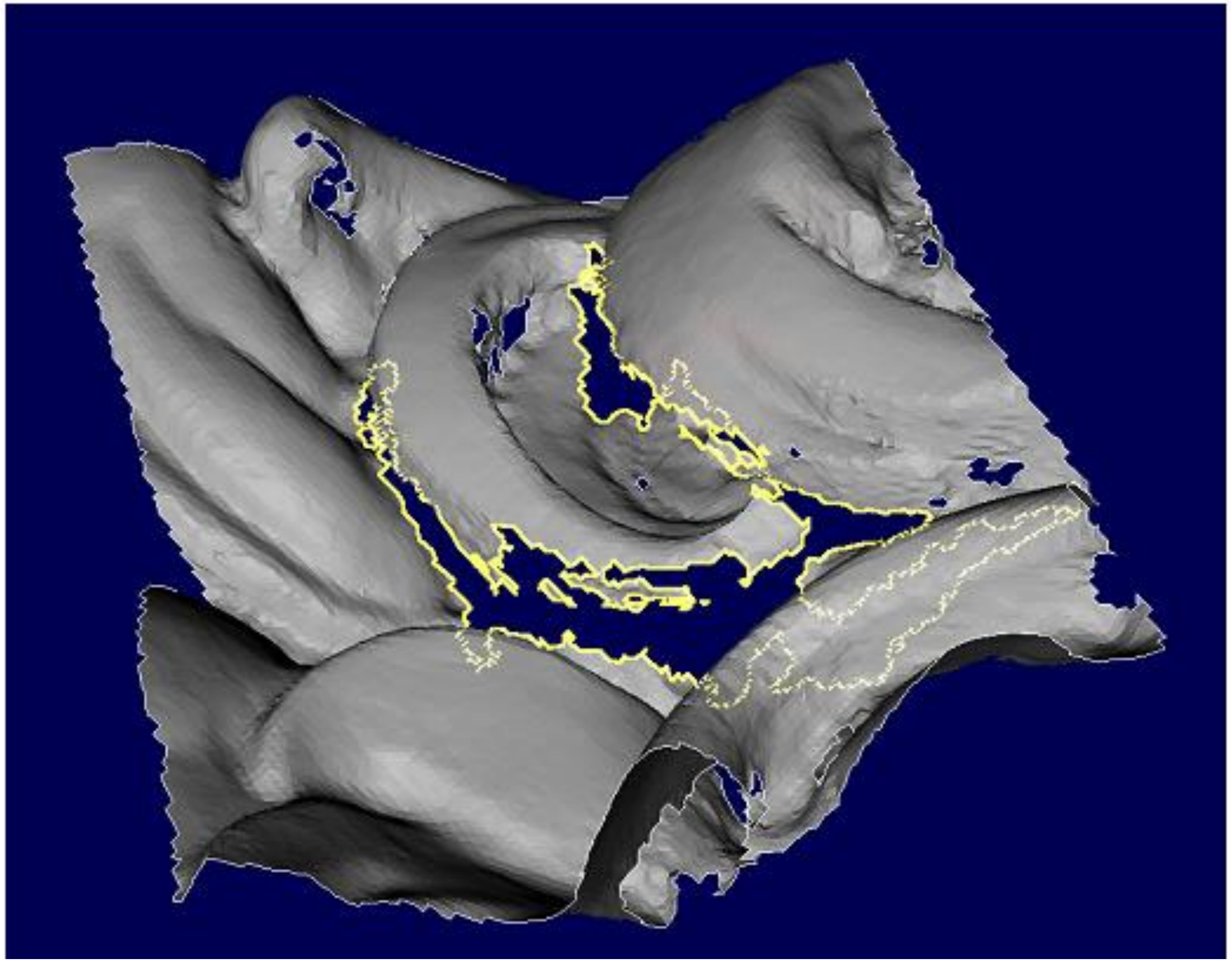


https://www.youtube.com/watch?v=GyJUG2VSvqw

A sketching interface for modeling the internal structures of 3d shapes [Owada SmartGraphics03]
ShapeShop: Sketch-Based Solid Modeling with BlobTrees [Schmidt SBIM05]

応用例：メッシュの穴埋め

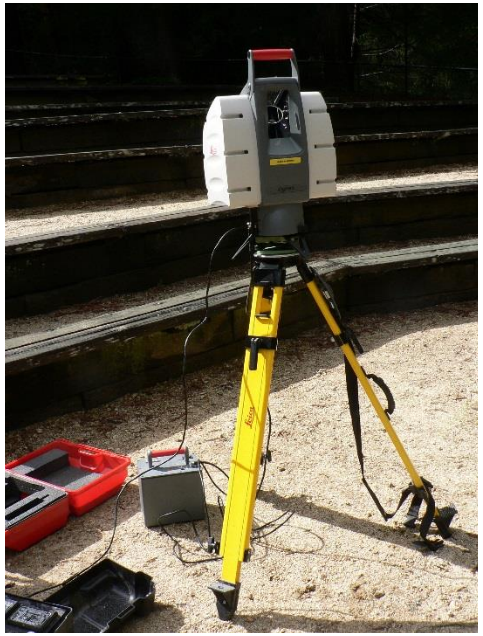




Filling holes in complex surfaces using volumetric diffusion [Davis 3DPVT02]

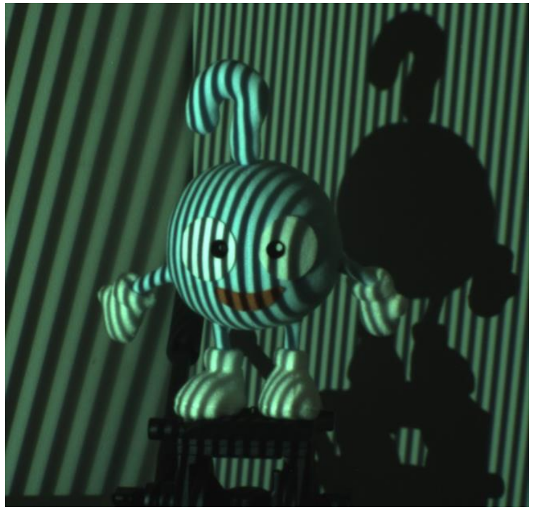
点群からのサーフェス再構成

3D 形状の計測



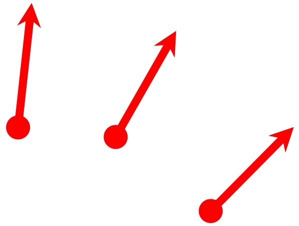
Range Scanner
(LIDAR)

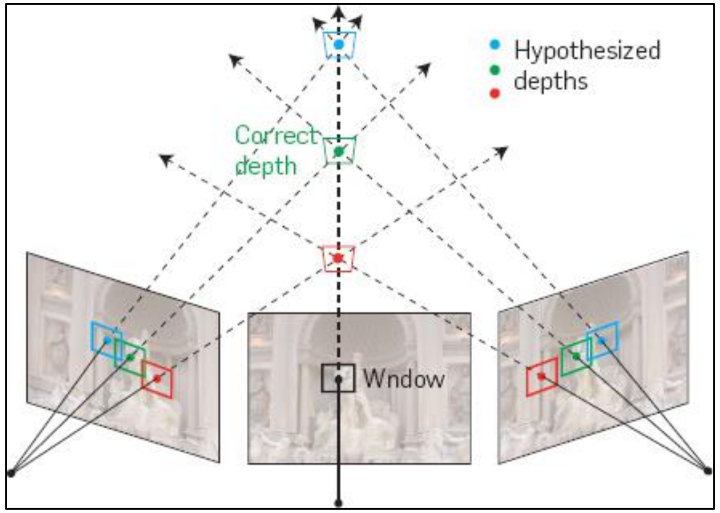
Depth Camera



Structured Light

- 得られるデータ：点群
  - 3D座標
  - 法線 (面の向き)
    - 得られない場合もある
    - ノイズが多すぎる場合もある



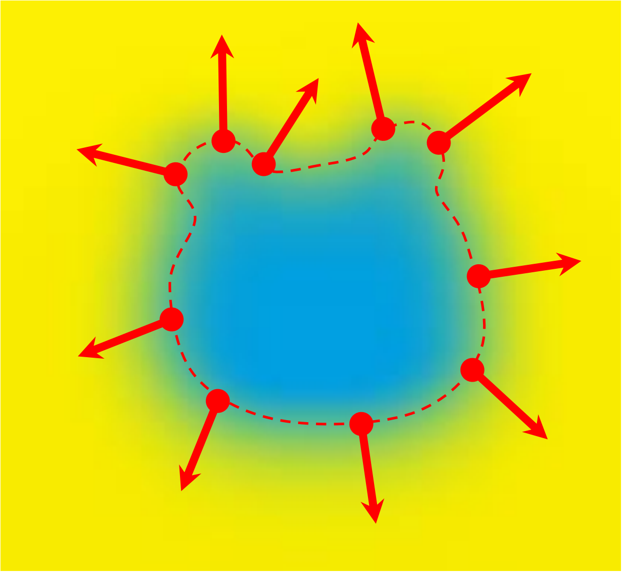


Multi-View Stereo

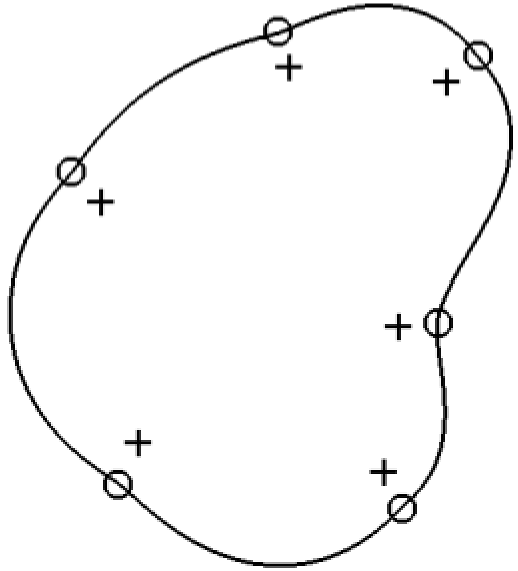
点群からのサーフェス形状再構成

- 入力：N 個の点群データ
  - 座標 $\mathbf{x}_i = (x_i, y_i, z_i)$ と法線 $\mathbf{n}_i = (n_i^x, n_i^y, n_i^z)$, $i \in \{1, \ldots, N\}$
- 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
  - **M**oving **L**east **S**quares
  - **R**adial **B**asis **F**unction

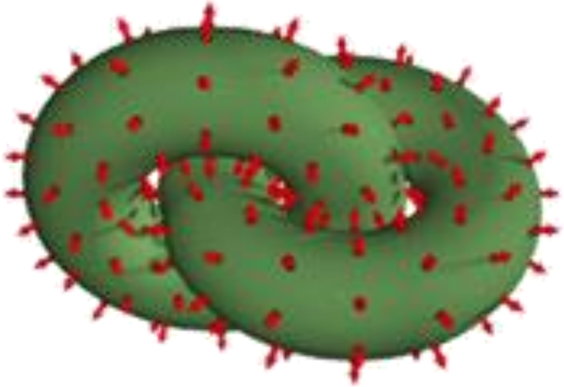
    CG以外の分野 (e.g. 機械学習) でも重要



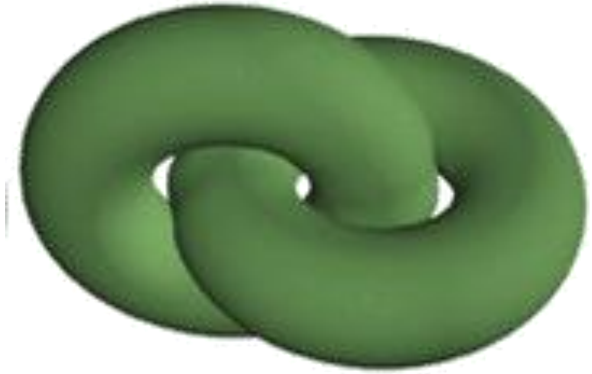
勾配を制約する二通りの方法



- 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

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

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

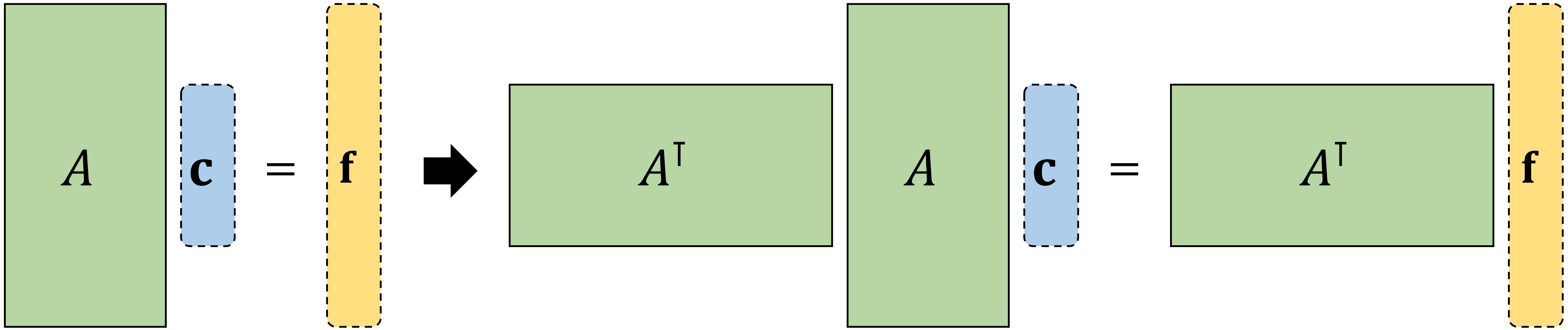
$$
\begin{aligned}
f(\mathbf{x}_1) &= ax_1 + by_1 + cz_1 + d = f_1 \\
f(\mathbf{x}_2) &= ax_2 + by_2 + cz_2 + d = f_2 \\
&\vdots \\
f(\mathbf{x}_N) &= ax_N + by_N + cz_N + d = f_N
\end{aligned}
$$

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

- (勾配制約は今は考えない)

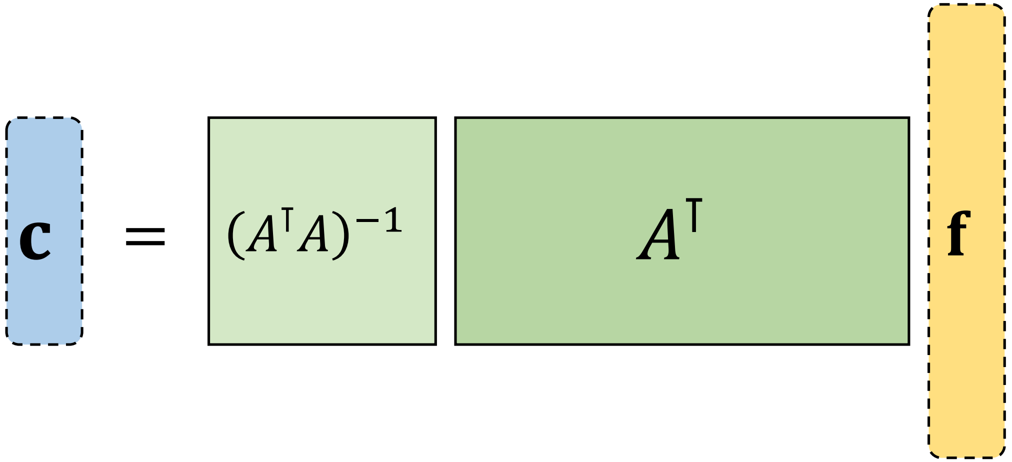
Overconstrained System

- #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



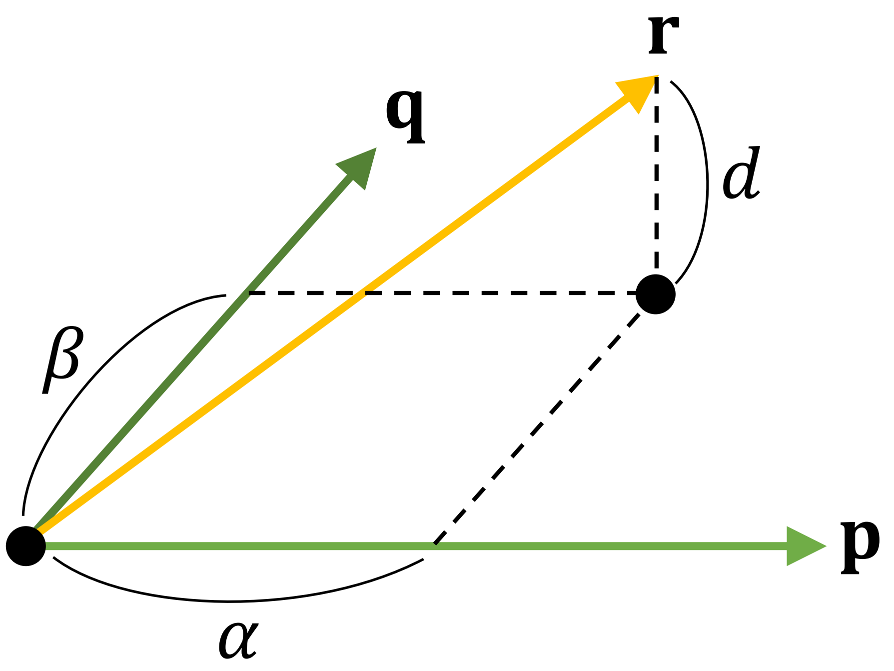
- fitting の誤差を最小化：

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



LSQ の幾何的な解釈

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



- $\mathbf{p}$ 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：

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

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

Weighted Least Squares (重み付き最小二乗)

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

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

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
  ➔ 係数 a, b, c, d が $\mathbf{x}$ に依存

$$f(\mathbf{x}) = [\, x \;\; y \;\; z \;\; 1 \,] \begin{bmatrix} a(\mathbf{x}) \\ b(\mathbf{x}) \\ c(\mathbf{x}) \\ d(\mathbf{x}) \end{bmatrix}$$

$$(A^\intercal W(\mathbf{x})^2 A)^{-1} \quad A^\intercal W(\mathbf{x})^2 \quad \mathbf{f}$$

法線制約の導入

- 各データ点が表す 1 次式を考える：

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

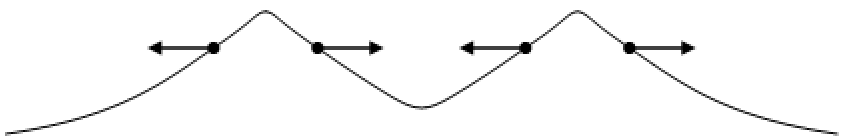
- 各 g_i を現在位置で評価したときの誤差を最小化：

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

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

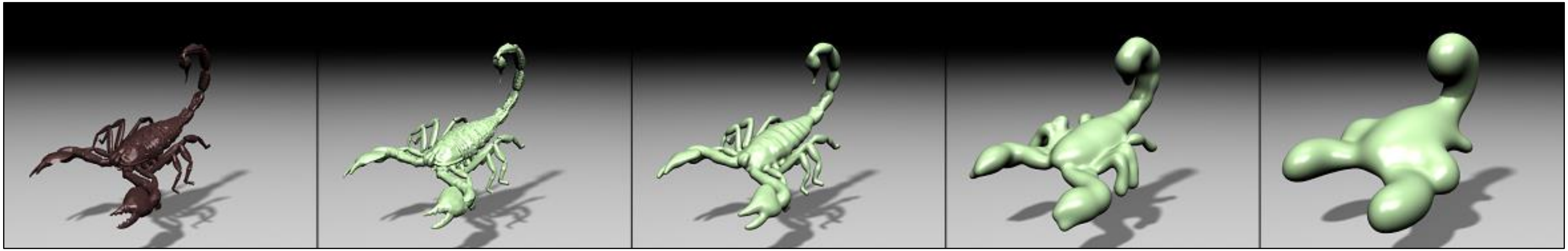
Interpolating and Approximating Implicit Surfaces from Polygon Soup [Shen SIGGRAPH04]

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

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

Radial **B**asis **F**unction 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

- 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

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

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

$$\vdots$$

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

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

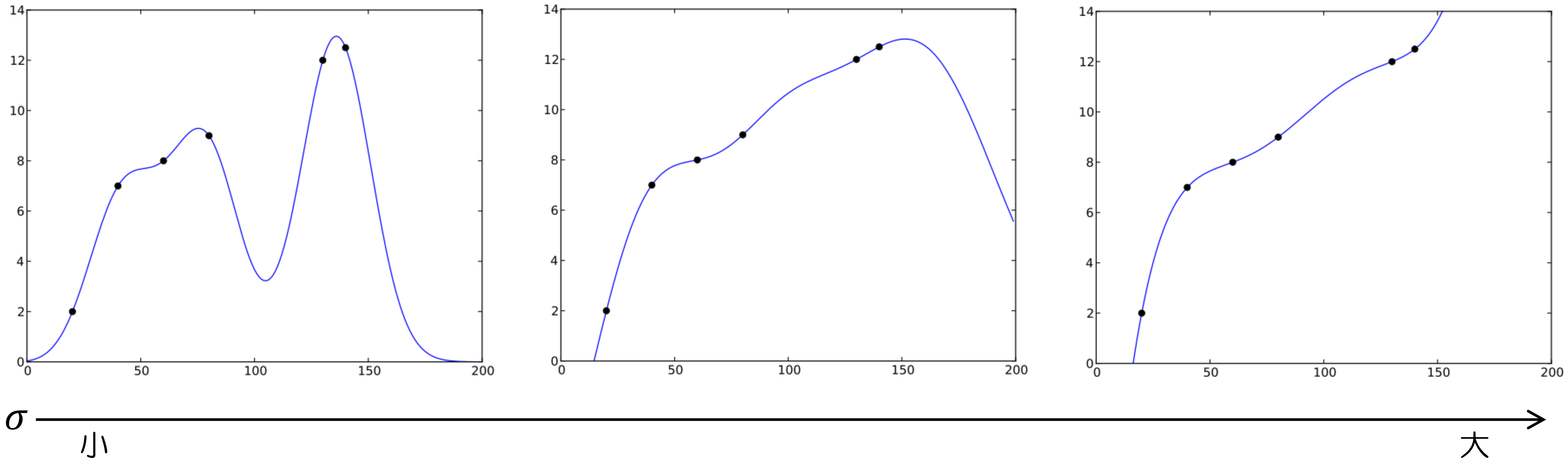
$$\Phi \mathbf{w} = \mathbf{f}$$

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 σ 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 "曲がり具合" の尺度 (Thin-Plate Energy)

- 2 階微分 (≒曲率) の大きさを空間全体で積分したもの：

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

  - 1 次元空間の場合：

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

  - 2 次元空間の場合：

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

  - 3 次元空間の場合：

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

数学分野の知見

- 制約 $\{ f(\mathbf{x}_i) = f_i \}$ を満たす関数全体のうち、E_2 を最小化する関数は以下の基底を使った RBF として表せる：
  - 1 次元空間の場合：$\phi(x) = |x|^3$
  - 2 次元空間の場合：$\phi(\mathbf{x}) = \|\mathbf{x}\|^2 \log \|\mathbf{x}\|$
  - 3 次元空間の場合：$\phi(\mathbf{x}) = \|\mathbf{x}\|$
- 参考
  - 有限要素法の場合：離散化した領域上で E_2 を最小化する f を近似的に求める
  - RBF の場合：グリーン関数を使って E_2 を最小化する f を解析的に求める

Scattered Data Interpolation for Computer Graphics [Anjyo SIGGRAPH14 Course]

線形項の追加

- $E_2[f]$ は 2 階微分を使って定義される
  ➔ 任意の線形項 $p(\mathbf{x}) = ax + by + cz + d$ を加えても不変：

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

- 線形項を未知数に含めることで、関数を一意に定める：

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

線形項の追加

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

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

$$\vdots$$

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

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

4 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：
  - $\sum_{i=1}^{N} w_i = 0$
  - $\sum_{i=1}^{N} x_i w_i = 0$
  - $\sum_{i=1}^{N} y_i w_i = 0$
  - $\sum_{i=1}^{N} z_i w_i = 0$

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

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

勾配制約の導入

- 基底関数の勾配 $\boldsymbol{\nabla}\phi$ の重み付き和を導入：

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

- f の勾配：

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

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

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

Hessian 行列

Hermite Radial Basis Functions Implicits [Macedo CGF10]

勾配制約の導入

- 1 番目のデータ点について：

値の制約：

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

$$by_1 + cz_1 + d = f_1$$

勾配の制約：

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

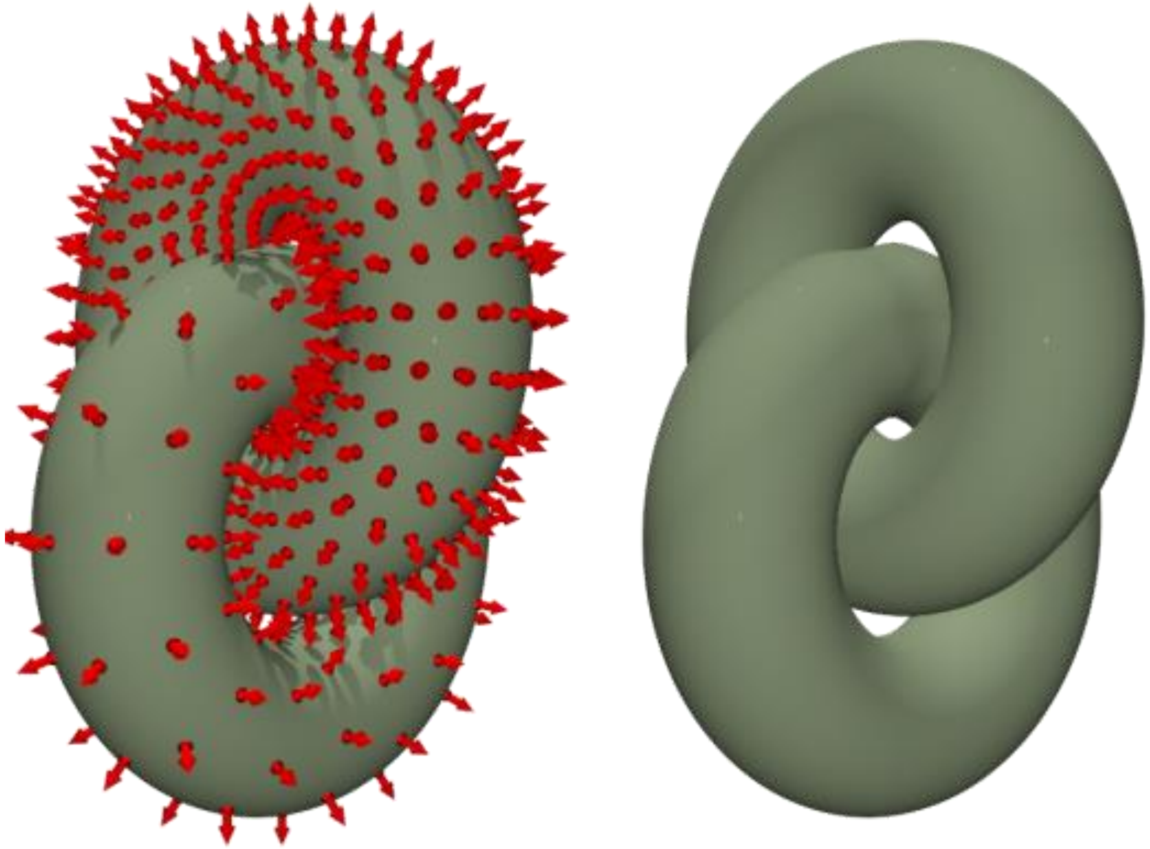
$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{n}_1$$

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

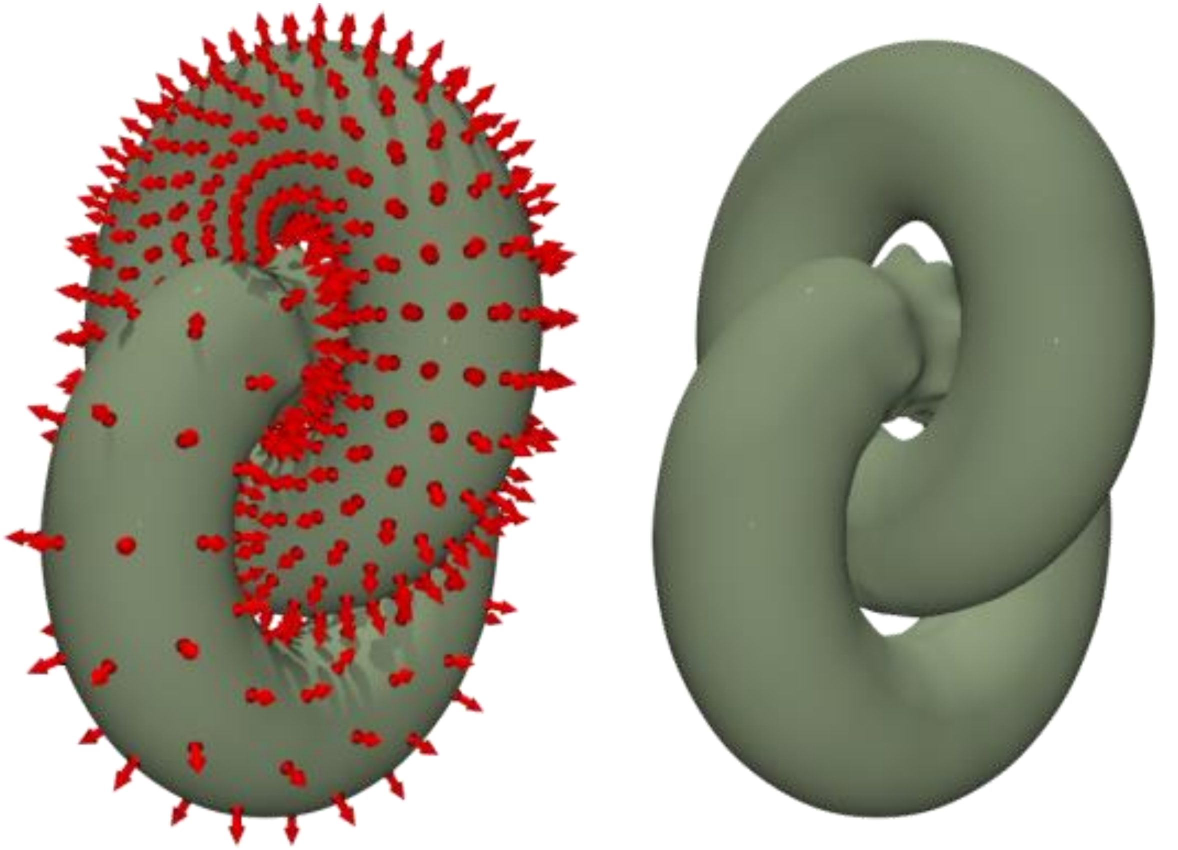
Hermite Radial Basis Functions Implicits [Macedo CGF10]

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

比較



勾配を制約



オフセットして値を制約

参考サーベイ等

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