コンピュータグラフィクス論

– モデリング (3) –

2025年5月1日

高山 健志

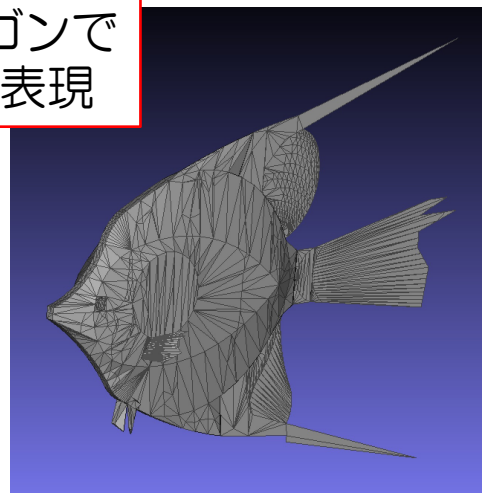
ソリッドモデリング

ソリッドモデルとは

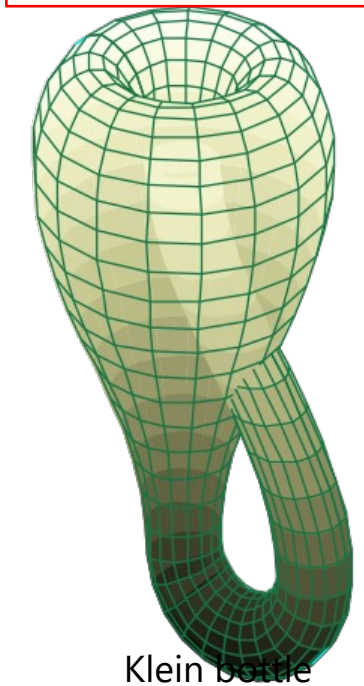
- 3D 空間の任意の位置で、モデルの“内側”と“外側”が定義できるもの

ソリッドでないケース

一枚のポリゴンで
薄い形状を表現

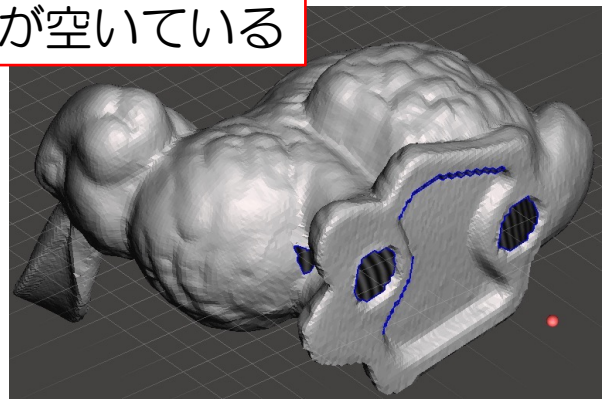


向き付け不可能

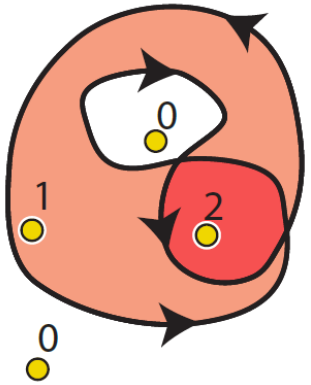


Klein bottle

穴が空いている



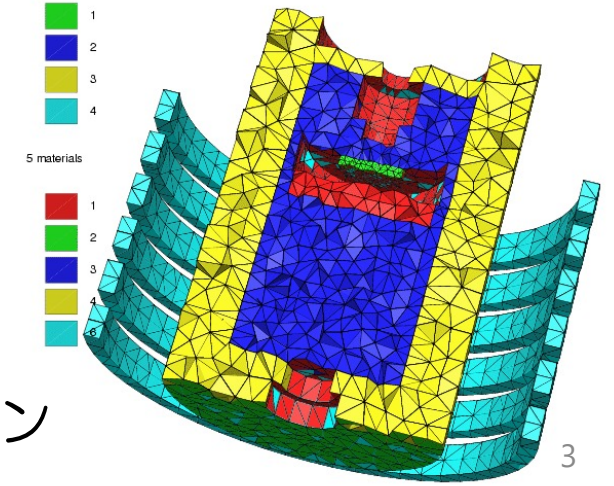
自己交差している



- 主な用途

3D プリント

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：

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

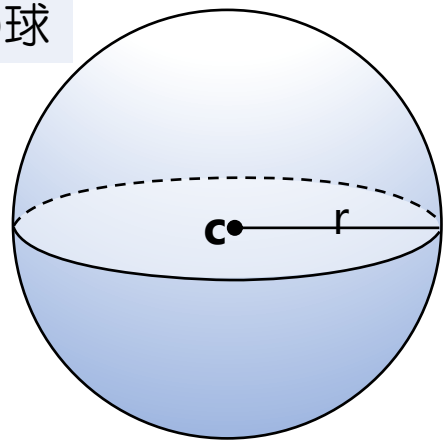
- モデル内部全体を表す集合：

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

- 例：

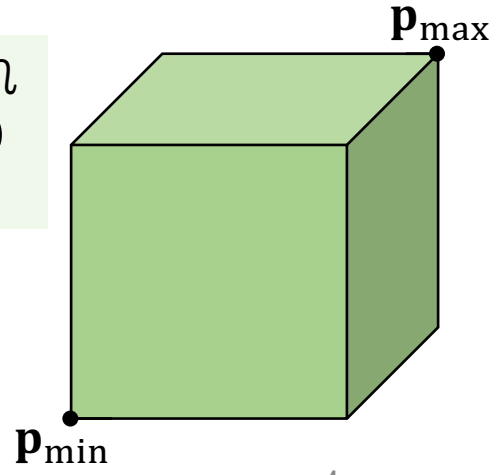
点 $\mathbf{c}$ を中心とした半径 r の球

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



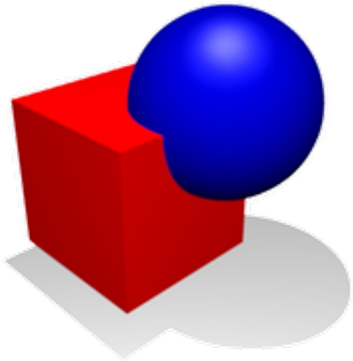
最小と最大の対角コーナーがそれぞれ $(x_{\min}, y_{\min}, z_{\min})$ と $(x_{\max}, y_{\max}, z_{\max})$ であるような直方体

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



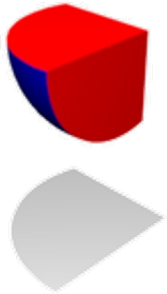
Constructive Solid Geometry (Boolean演算)

和



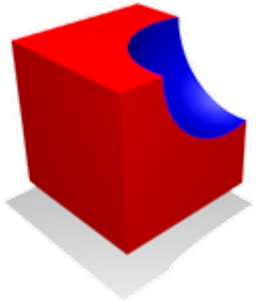
$$f_{A\cup B}(\mathbf{p}) \coloneqq f_A(\mathbf{p}) \vee f_B(\mathbf{p})$$

積



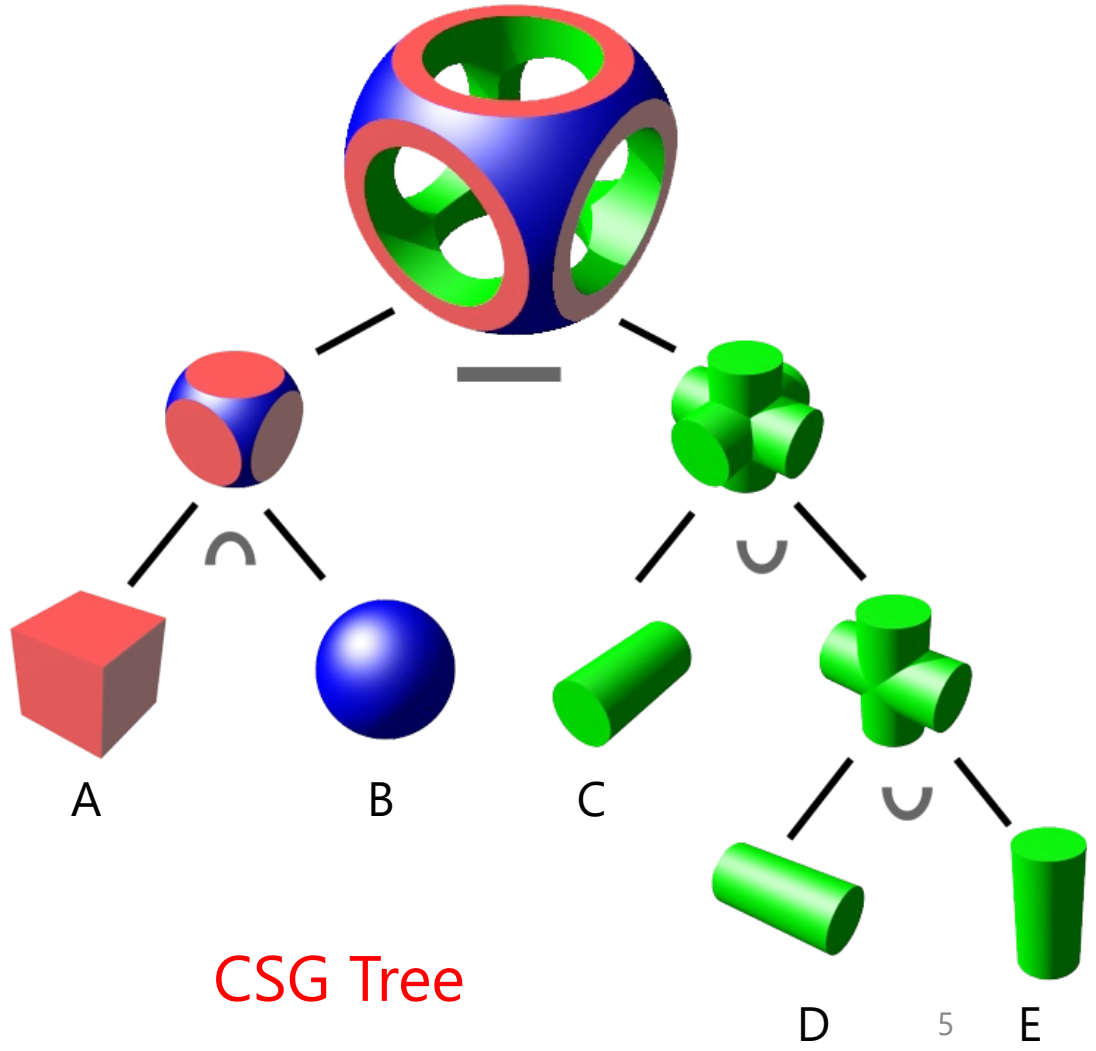
$$f_{A\cap B}(\mathbf{p}) \coloneqq f_A(\mathbf{p}) \wedge f_B(\mathbf{p})$$

差



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

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



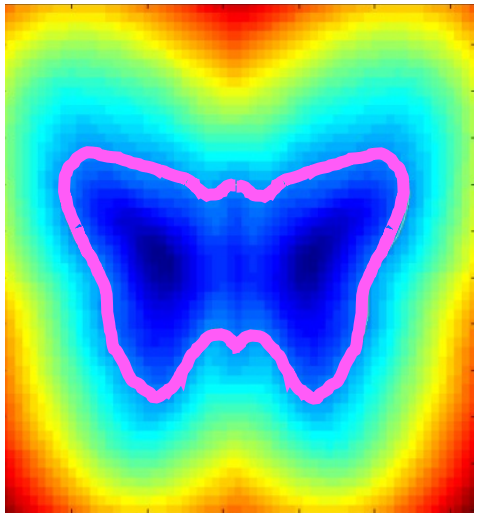
CSG Tree

符号付き距離場によるソリッドモデル表現

- **S**igned **D**istance **F**unction: 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:

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

- ゼロ等値面はモデル表面を表す：

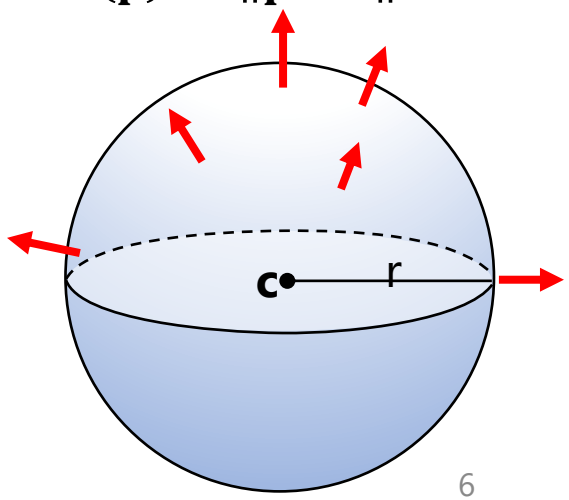
$$\{\mathbf{p} \mid d(\mathbf{p}) = 0\} \subset \mathbb{R}^3$$

- 「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

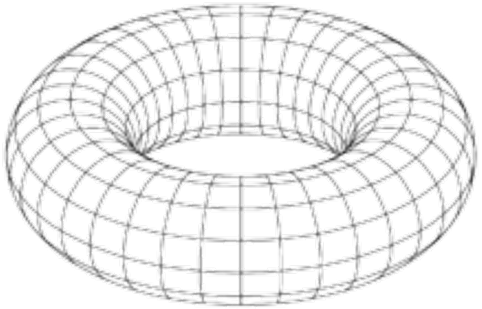
点 $\mathbf{c}$ を中心とした半径 r の球

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



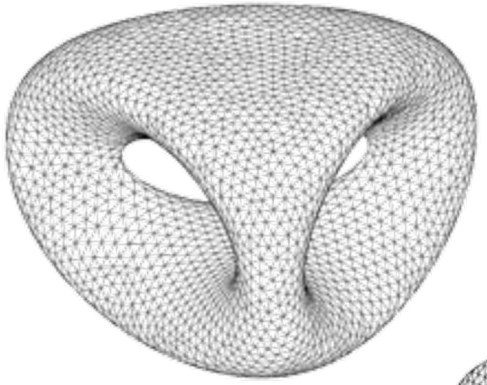
陰関数のデザイン例

必ずしも距離関数とは限らない

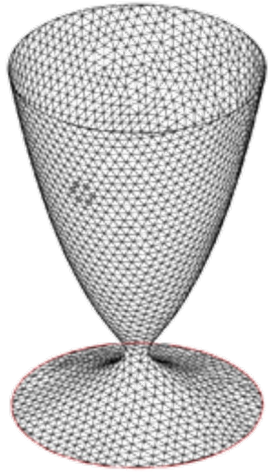


大半径 R, 小半径 a のトーラス

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



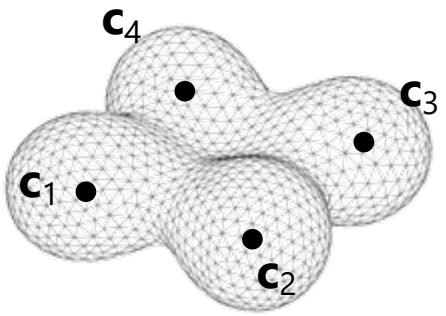
$$2y(y^2-3x^2)(1-z^2)+(x^2+y^2)^2-(9z^2-1)(1-z^2)=0$$



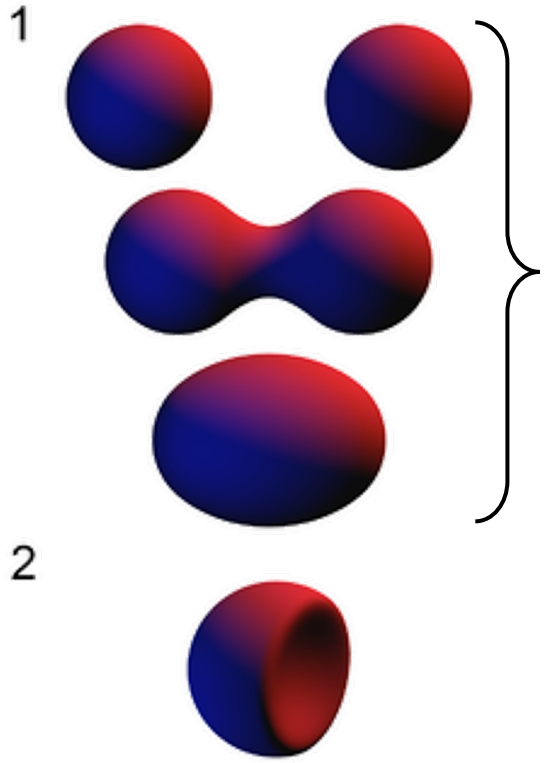
$$x^2+y^2-(\ln(z+3.2))^2-0.02=0$$

陰関数のデザイン例：等電位面 (Metaball)

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



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



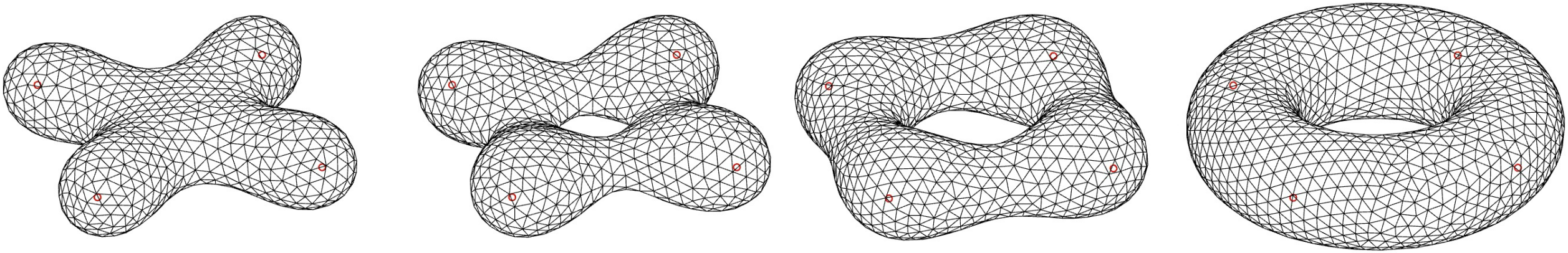
$$d(\mathbf{p}) = d_1(\mathbf{p}) + d_2(\mathbf{p})$$

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

Fast metaball rendering

GPU-based Fast Ray Casting for a Large Number of Metaballs [Kanamori,Szego,Nishita,EG08].mp4

陰関数の線形補間によるモーフィング



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

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

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

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

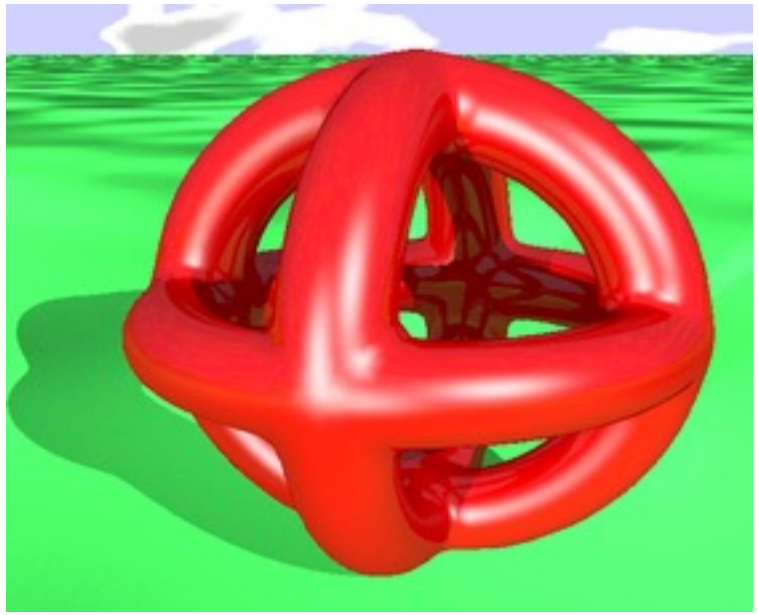
複数の陰関数を組み合わせたモデリング

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

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

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

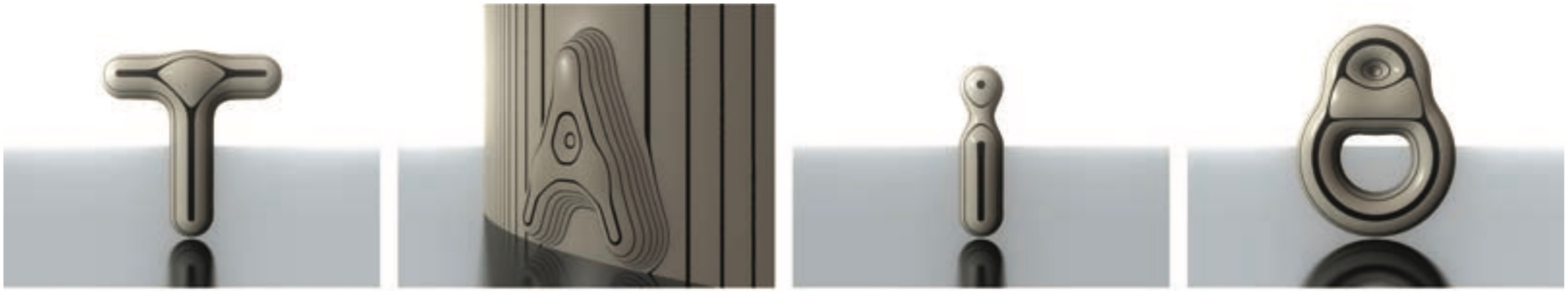
$$F(x, y, z) = F_1(x, y, z) \cdot F_2(x, y, z) \cdot F_3(x, y, z) - c = 0$$



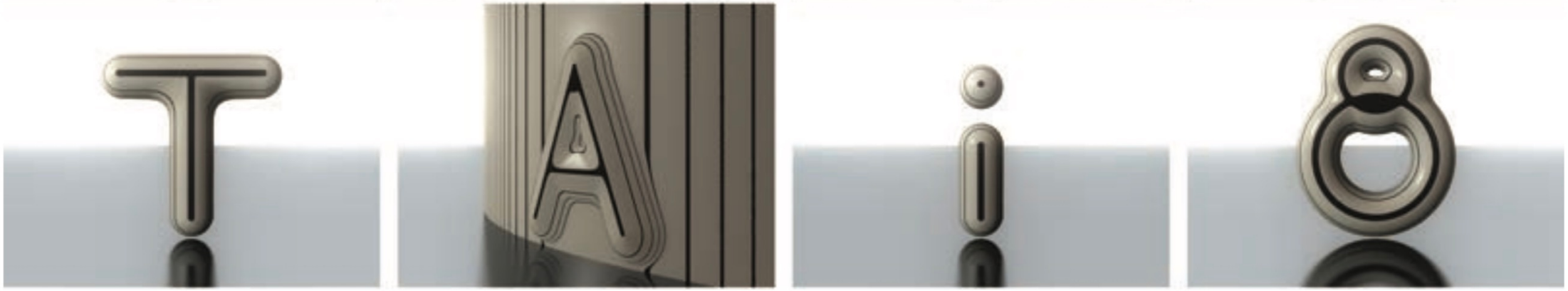
より高度なブレンディング

- 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

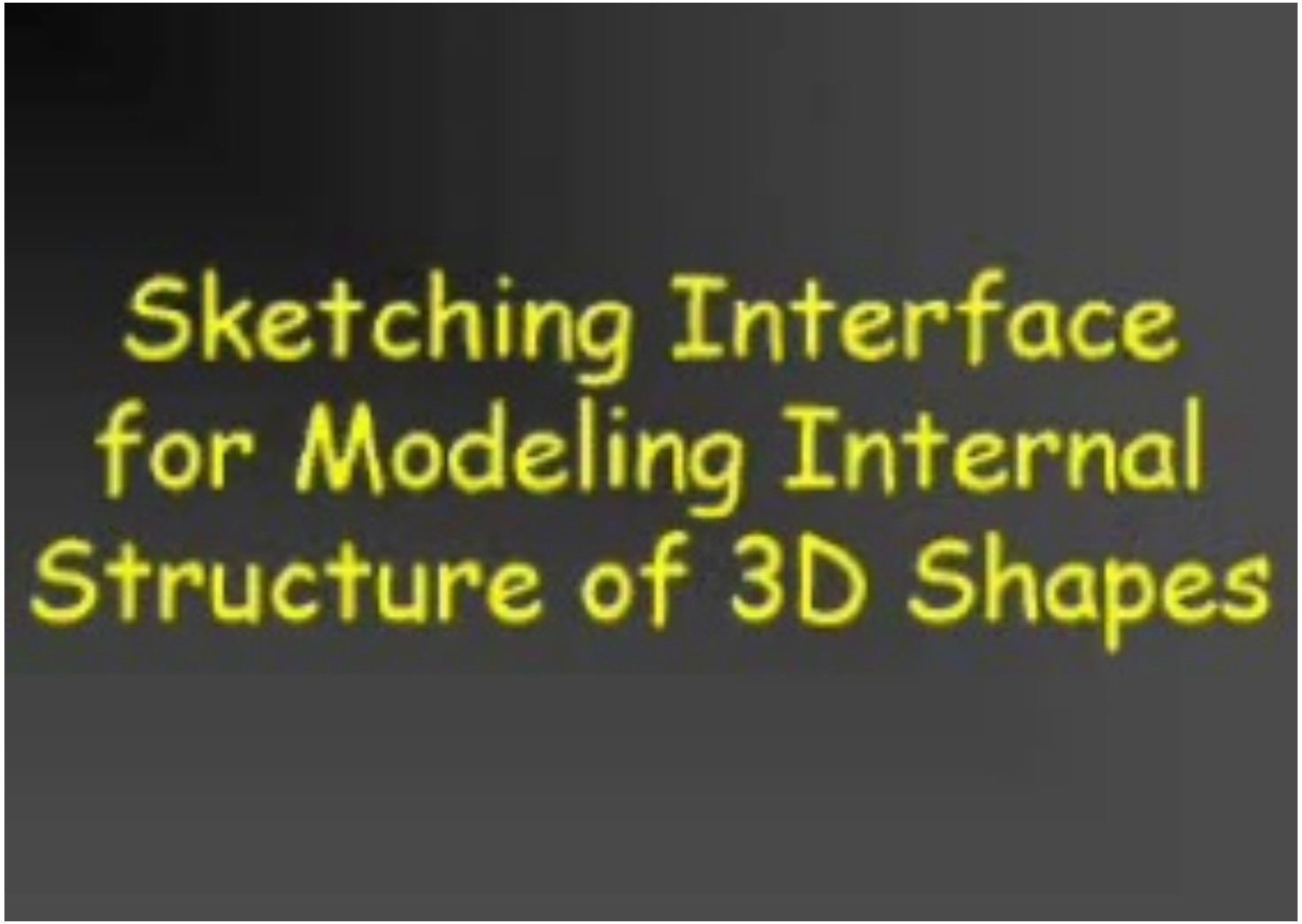


A gradient-based implicit blend [Gourmel,Barthe,Cani,Wyvill,Bernhardt,Grasberger,TOG13]

陰関数の性質を活用した3Dモデリングの例

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

陰関数の性質を活用した3Dモデリングの例



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

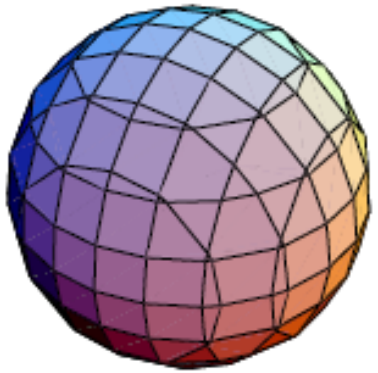
陰関数の表示方法：Marching Cubes

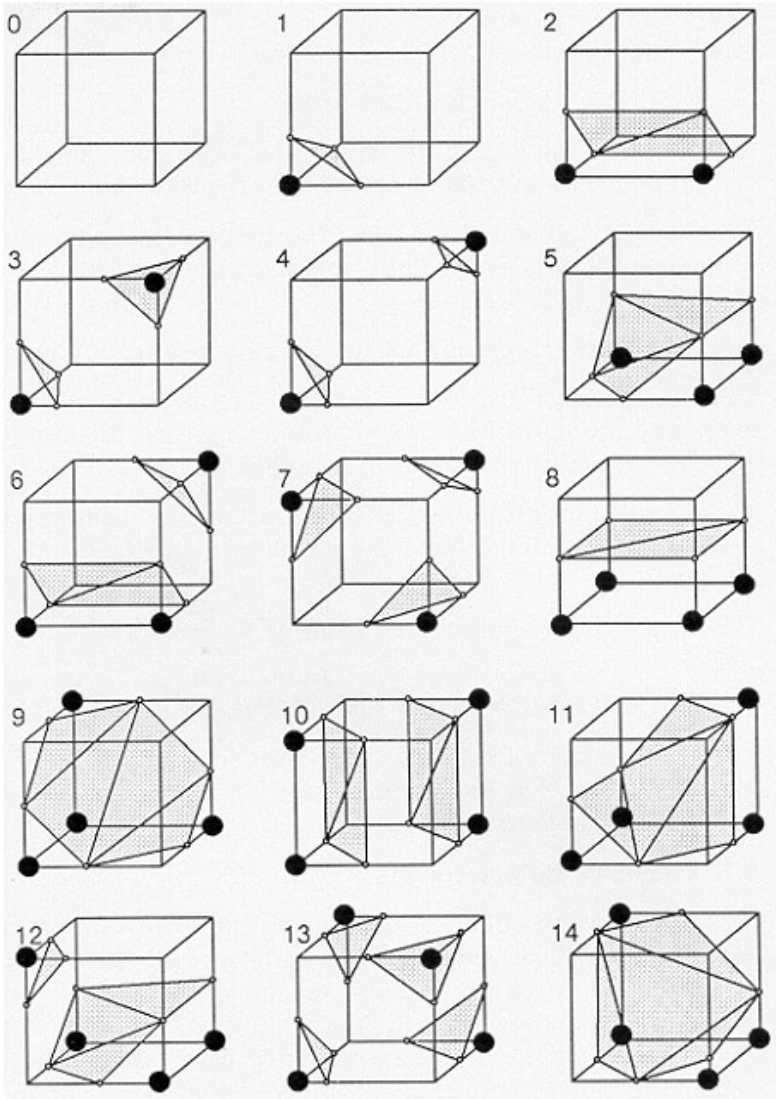
- 等値面を三角形メッシュとして抽出

- 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

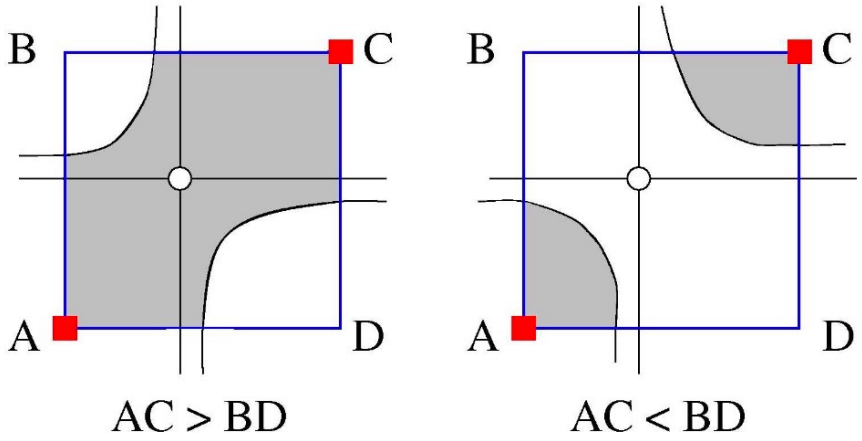




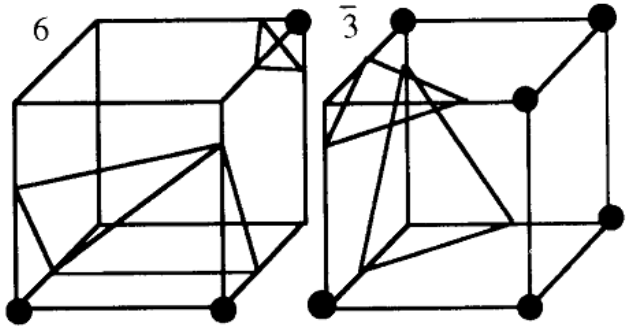
(特許で縛られていたが、期限が切れた)

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

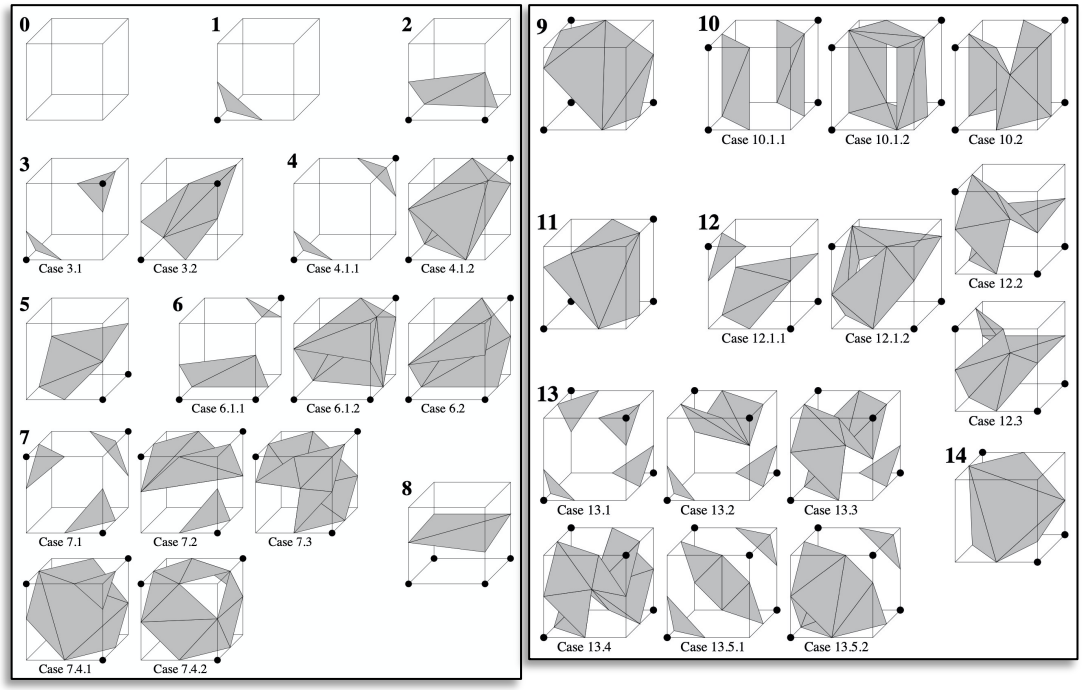
Marching Cubes の曖昧性



解決法：Bilinear/Trilinear補間に基づいてトポロジを決定



隣接するセルの間で面が整合しない

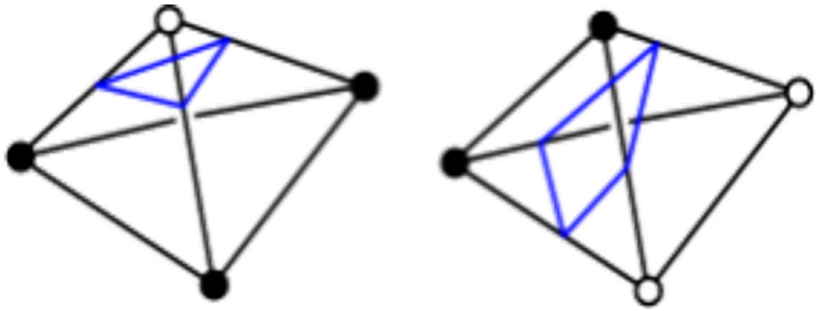


曖昧性を解決するための33種類のパターン
(正しく実装するのは結構大変・・・)

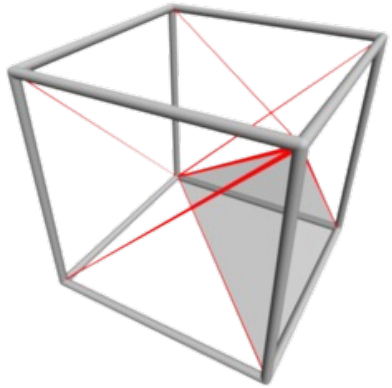
The asymptotic decider: resolving the ambiguity in marching cubes [Nielson VIS91]
Marching cubes 33: Construction of topologically correct isosurfaces [Chernyaev Tech.Rep. 95]
Topology Verification for Isosurface Extraction [Etiene TVCG12]
A Fast and Memory Saving Marching Cubes 33 Implementation with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

Marching Tetrahedra

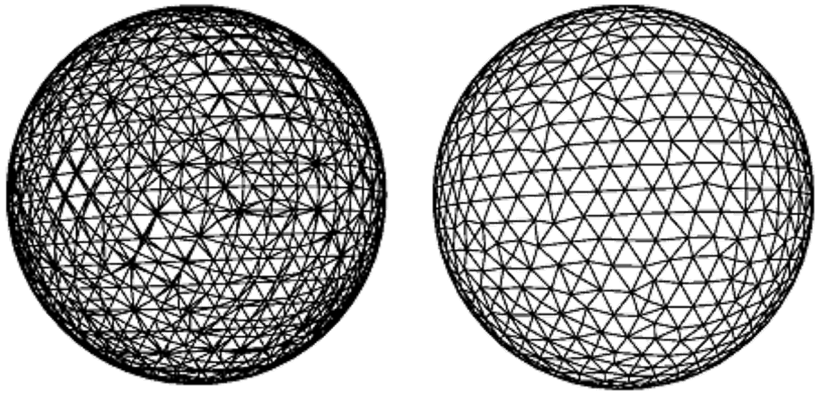
- 立方体の代わりに四面体を使う
  - パターンが少なく、曖昧性が無い ☺
    ➔ 実装が簡単
  - MCと比べて、三角形の数が多め ☹



- 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)



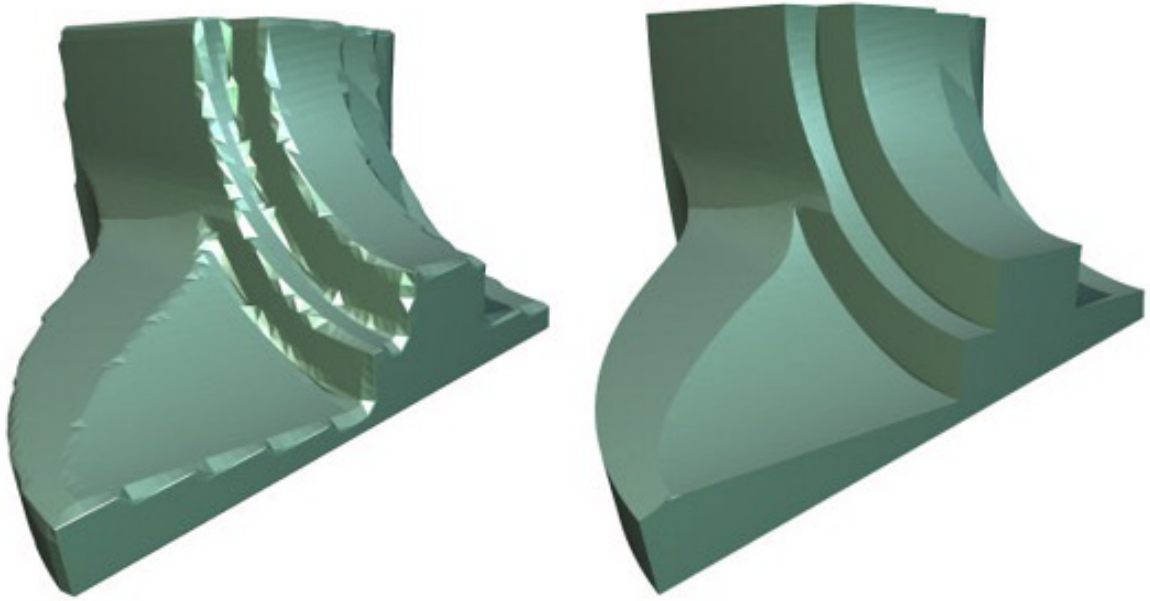
- きれいな三角形メッシュを取り出す工夫



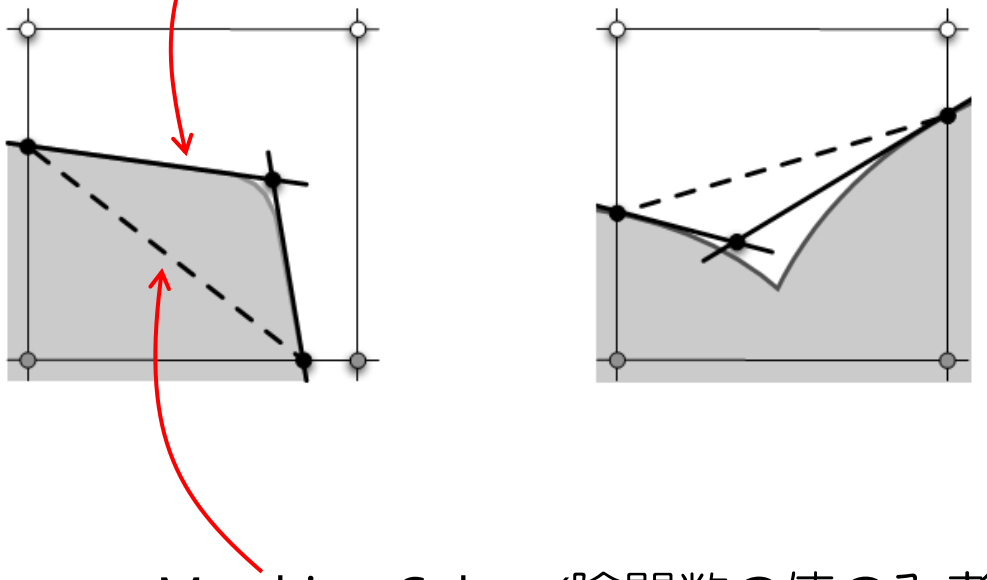
http://paulbourke.net/geometry/polygonise/
Regularised marching tetrahedra: improved iso-surface extraction [Treece C&G99]

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



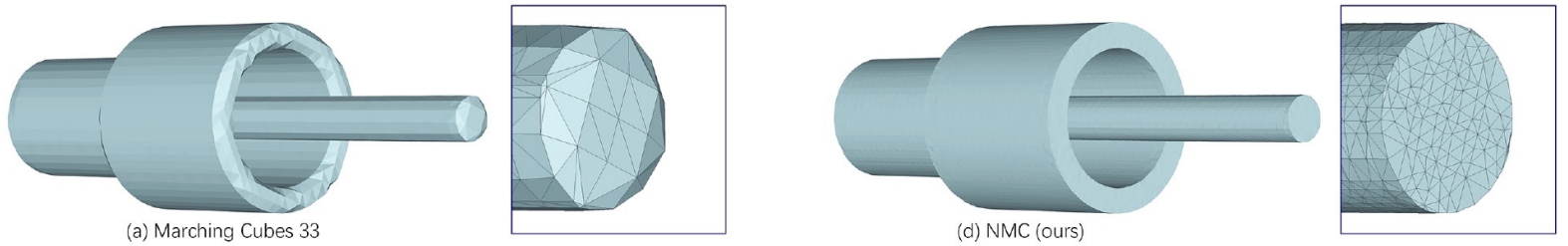
Marching Cubes　　改善版

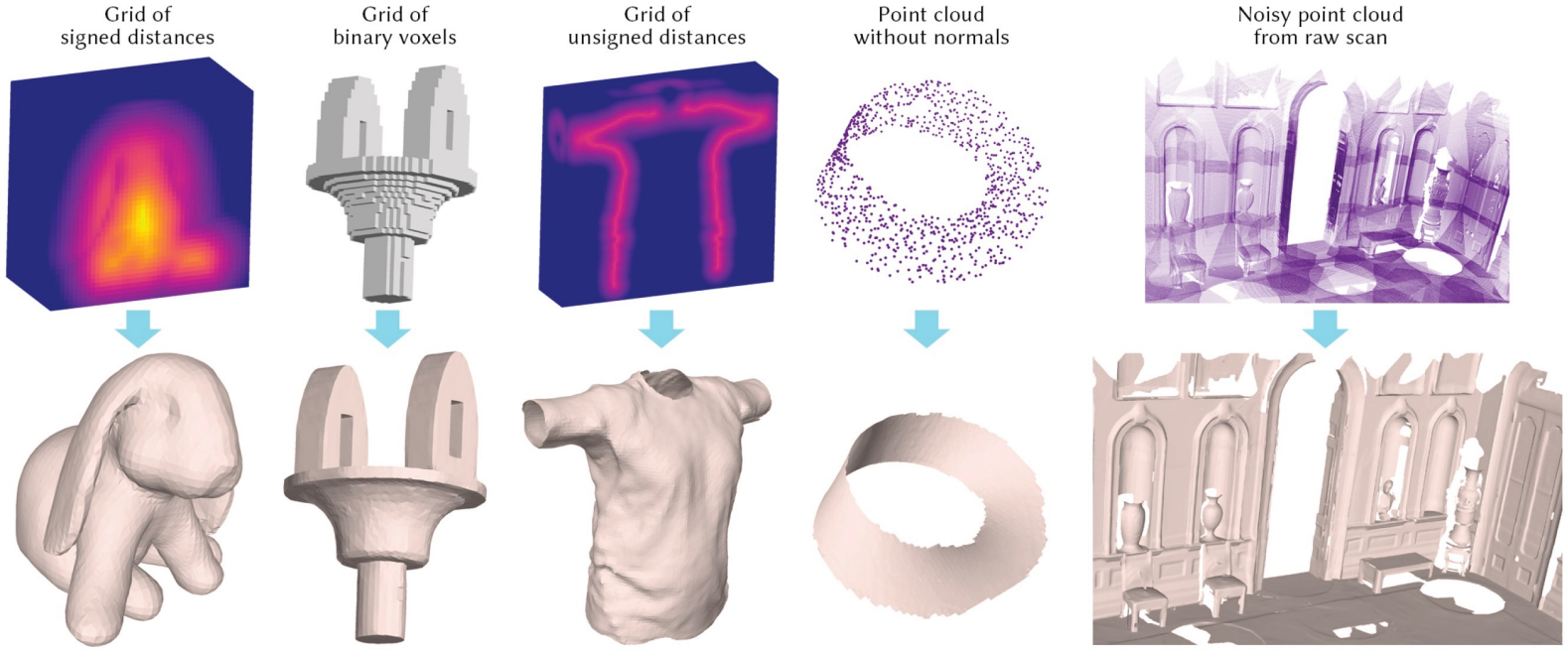


Feature Sensitive Surface Extraction from Volume Data [Kobbelt SIGGRAPH01]
Dual Contouring of Hermite Data [Ju SIGGRAPH02]
http://www.graphics.rwth-aachen.de/IsoEx/

深層学習を活用した最近の研究

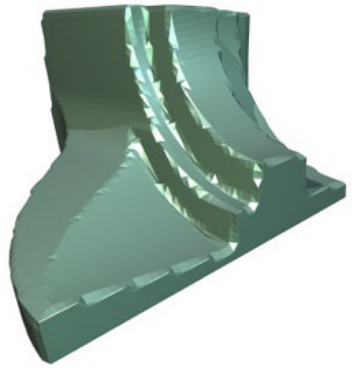
- Neural Marching Cubes
  - SIGGRAPH Asia 2021
- Neural Dual Contouring
  - SIGGRAPH 2022
- 両方とも同じ著者
  https://czq142857.github.io/





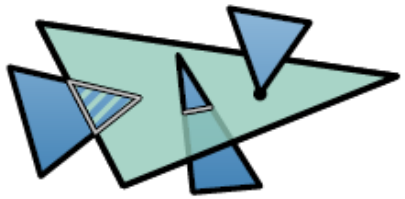
サーフェスメッシュ表現のみに基づく CSG

- 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 ☹

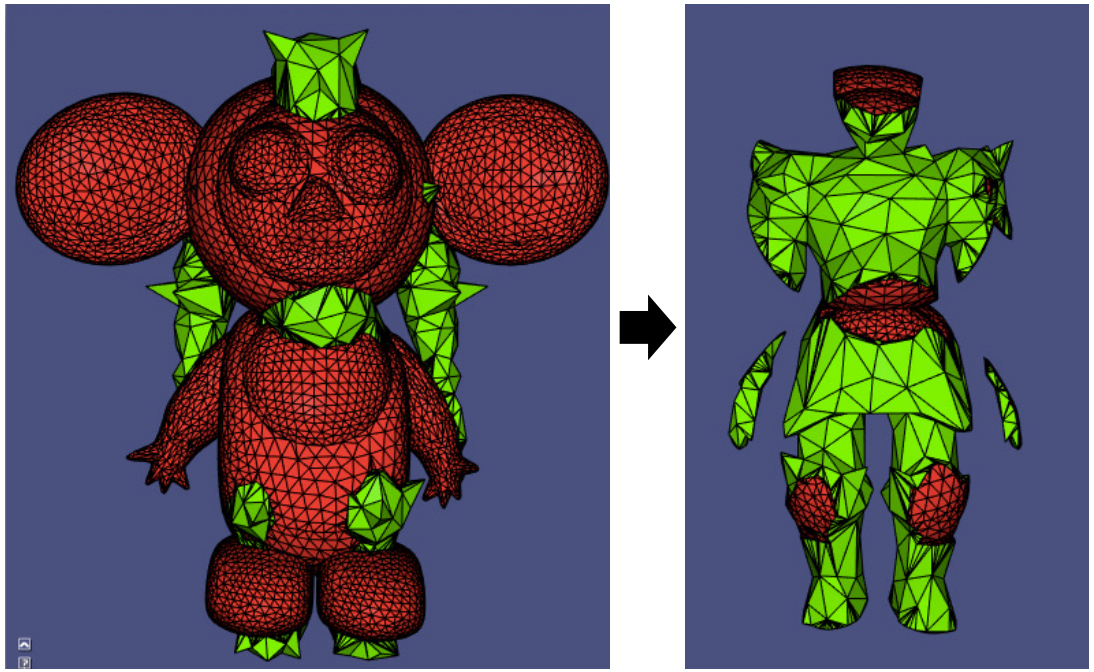


- サーフェスメッシュ表現による CSG
  ➔ 元のメッシュの形状を確実に保持 ☺

- 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



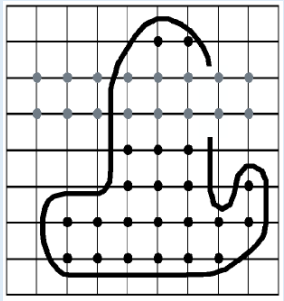
- 近年著しく進化



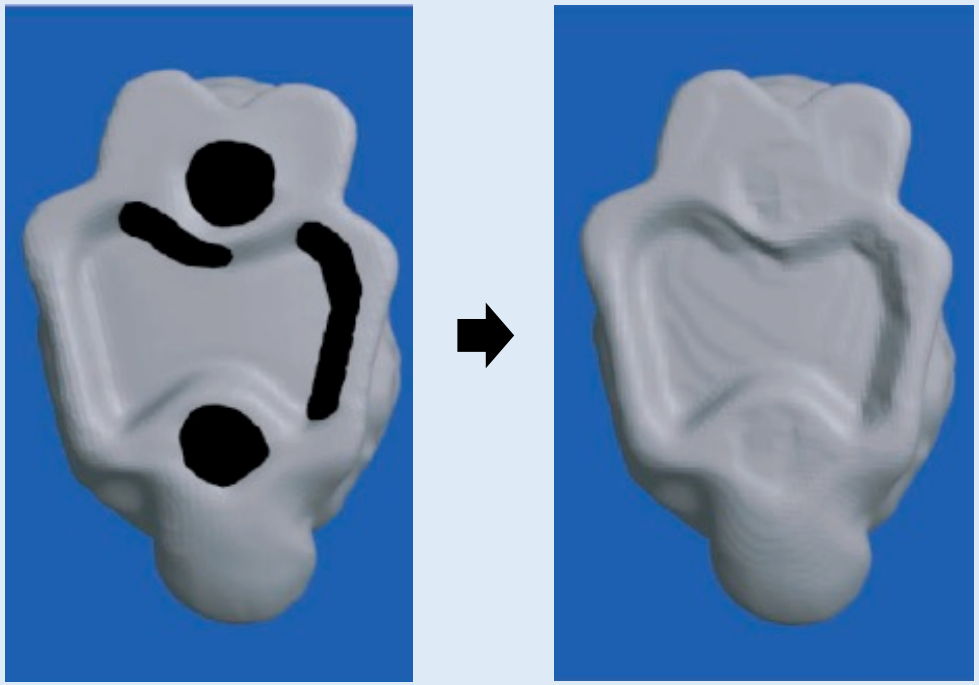
Fast, exact, linear booleans [Bernstein SGP09]
Exact and Robust (Self-)Intersections for Polygonal Meshes [Campen EG10]
Mesh Arrangements for Solid Geometry [Zhou SIGGRAPH16]
Exact and Efficient Intersection Resolution for Mesh Arrangements [Guo SIGGRAPHAsia24]
https://libigl.github.io/tutorial/#boolean-operations-on-meshes

メッシュの補修 (mesh repair)

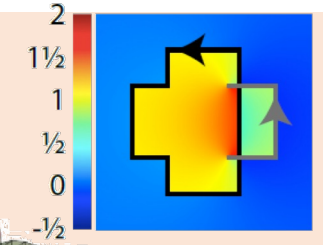
ボリューム表現



四方八方から飛ばしたレイとの
交差に基づいて内外を判定



サーフェス表現



定義を拡張した
winding number に
基づいて内外を判定

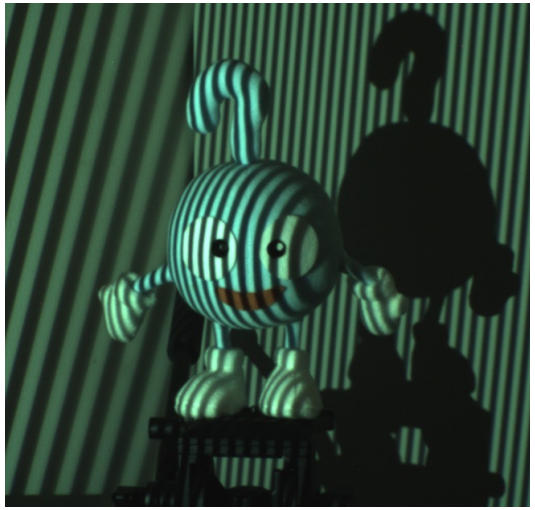
Simplification and Repair of Polygonal Models Using Volumetric Techniques [Nooruddin TVCG03]
Robust Inside-Outside Segmentation using Generalized Winding Numbers [Jacobson SIGGRAPH13]

点群からのサーフェス再構成

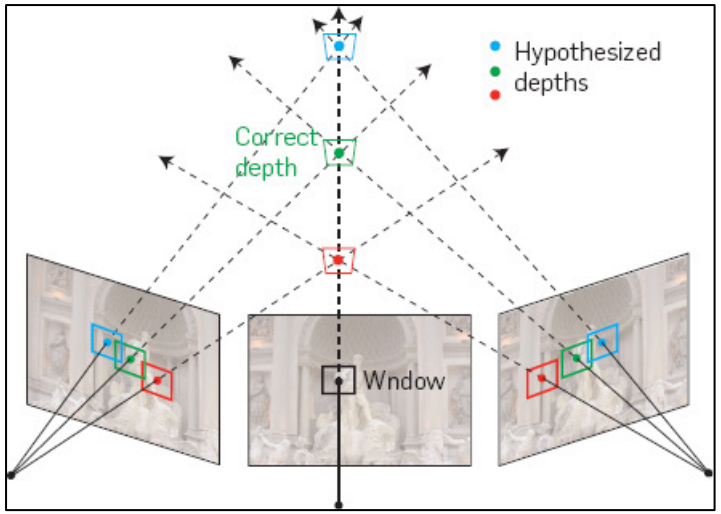
3D 形状の計測

Range Scanner
(LIDAR)

Depth Camera

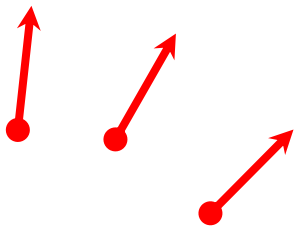


Structured Light



Multi-View Stereo

- 得られるデータ：点群
  - 3D座標
  - 法線 (面の向き)

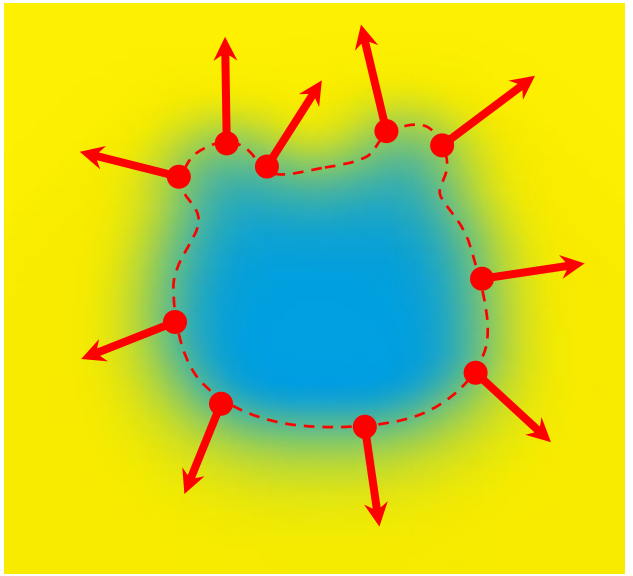


- 法線が得られない場合 ➔ 法線の推定
- ノイズが多すぎる場合 ➔ ノイズの除去

} 今回は取り上げない

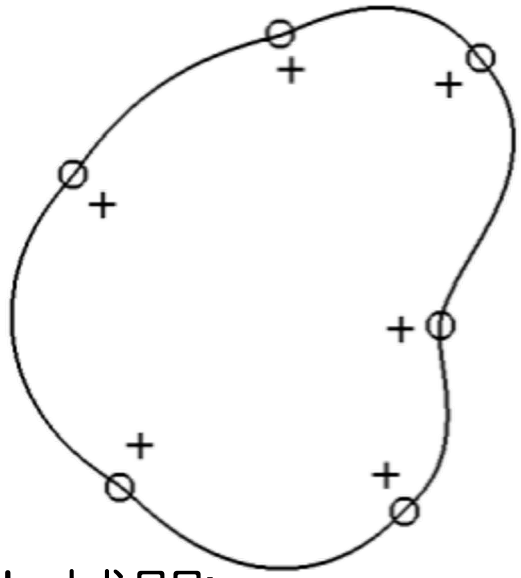
点群からのサーフェス形状再構成

- 入力：N 個の点群データ
  - 座標 $\mathbf{x}_i = (x_i, y_i, z_i)$ と法線 $\mathbf{n}_i = \left(n_i^{\mathrm{x}}, n_i^{\mathrm{y}}, n_i^{\mathrm{z}}\right),\ i \in \{1, \ldots, N\}$
- 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

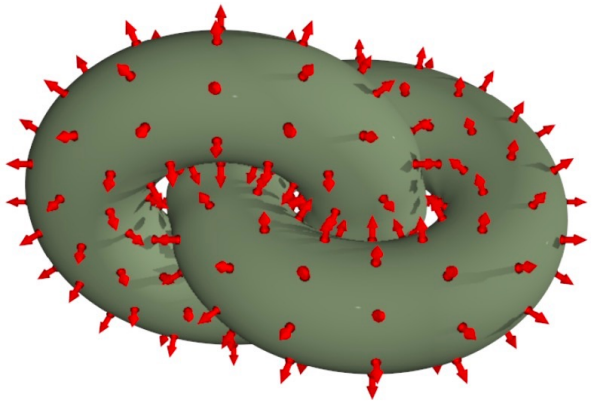


勾配を制約する二通りの方法

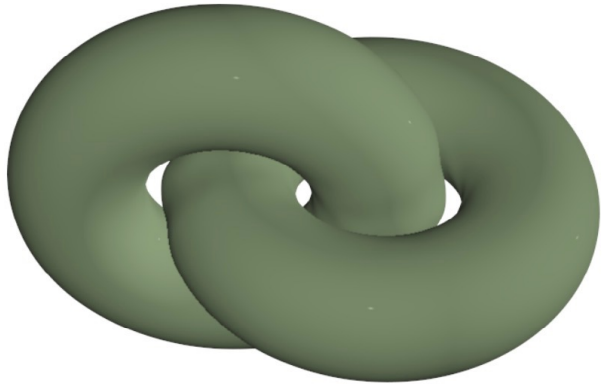
- 法線方向にオフセットした位置に値の制約を追加
  - 簡単



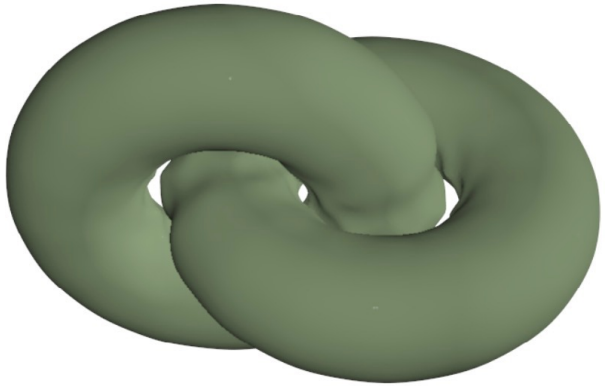
- 数学表現そのものに勾配制約を取り入れる (エルミート補間)
  - 高品質



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：**L**east **SQ**uares (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

- データ点における値の制約

$\mathbf{x} \coloneqq (x, y, z)$

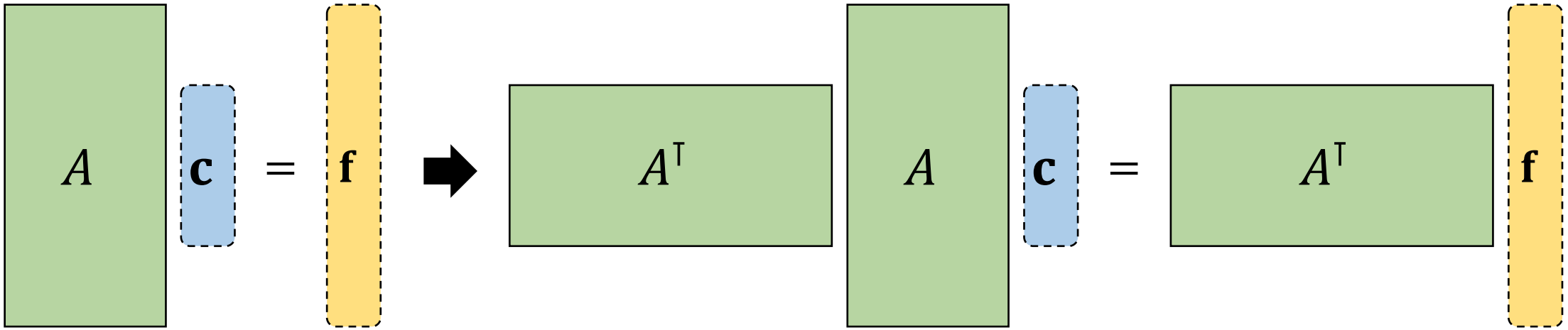
$$f(\mathbf{x}_1) = ax_1 + by_1 + cz_1 + d = f_1$$
$$f(\mathbf{x}_2) = ax_2 + by_2 + cz_2 + d = f_2$$
$$\vdots$$
$$f(\mathbf{x}_N) = ax_N + by_N + cz_N + d = f_N$$

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

- (勾配制約は今は考えない)

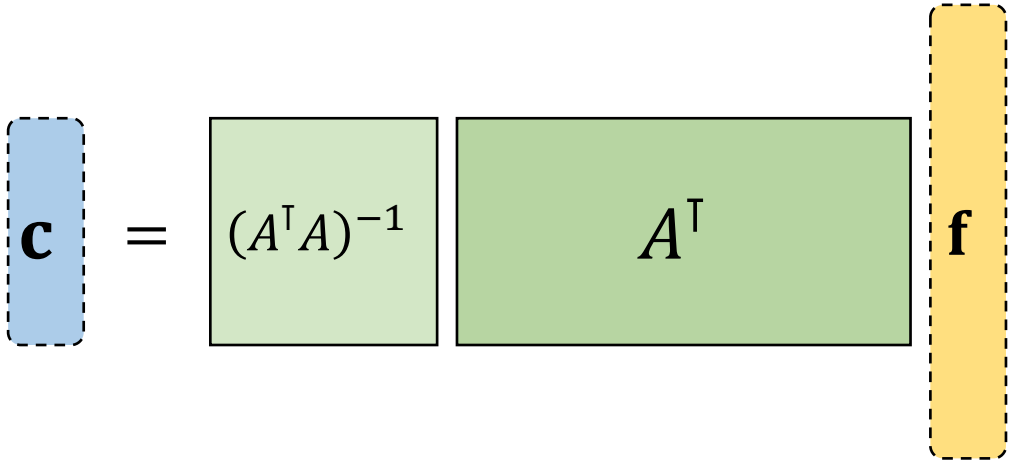
Overconstrained System

- #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



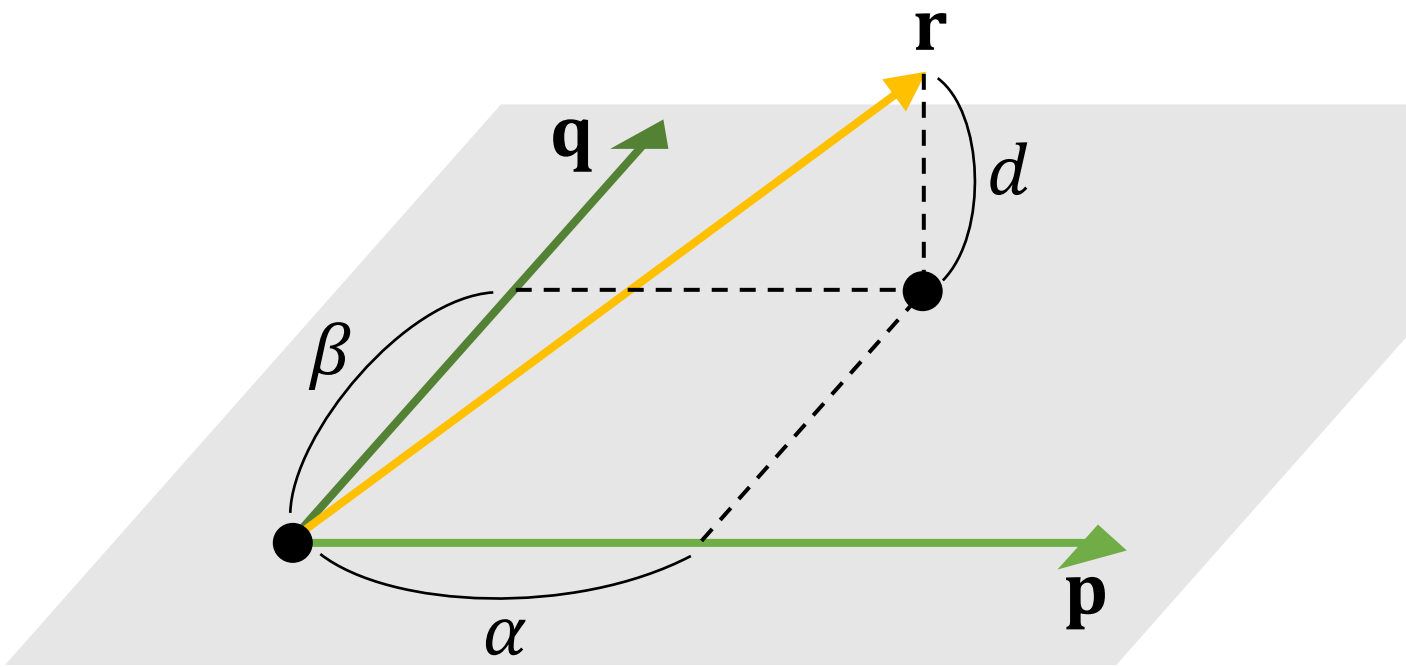
- fitting の誤差を最小化：

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



LSQ の幾何的な解釈

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



- **p** と **q** が張る空間中で **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：

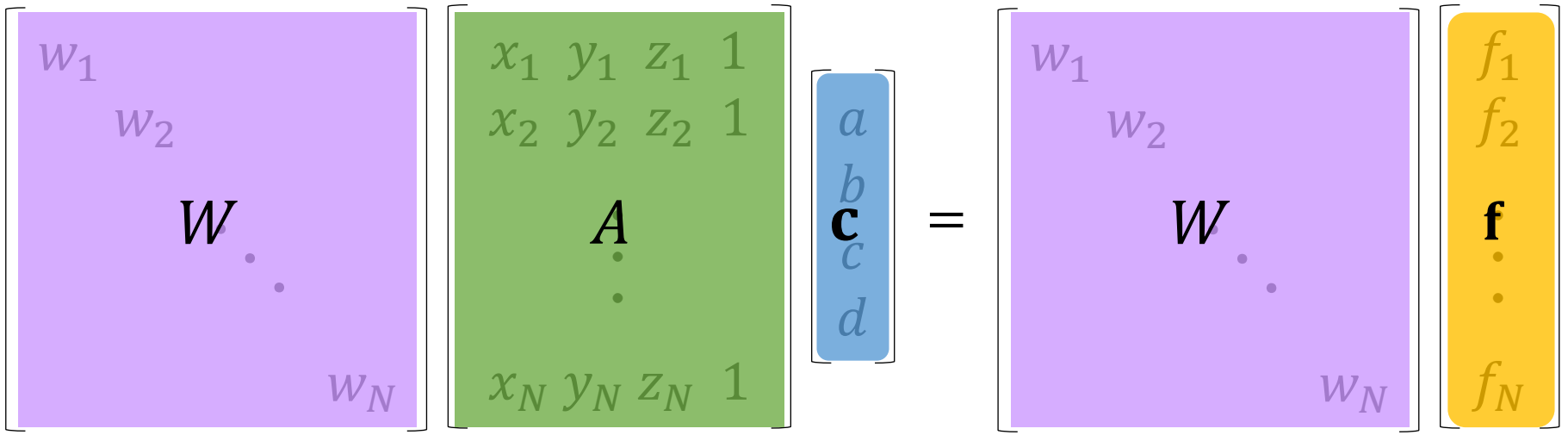
$$d^2 = \|\alpha\mathbf{p} + \beta\mathbf{q} - \mathbf{r}\|^2$$

Weighted **L**east **S**quares (重み付き最小二乗)

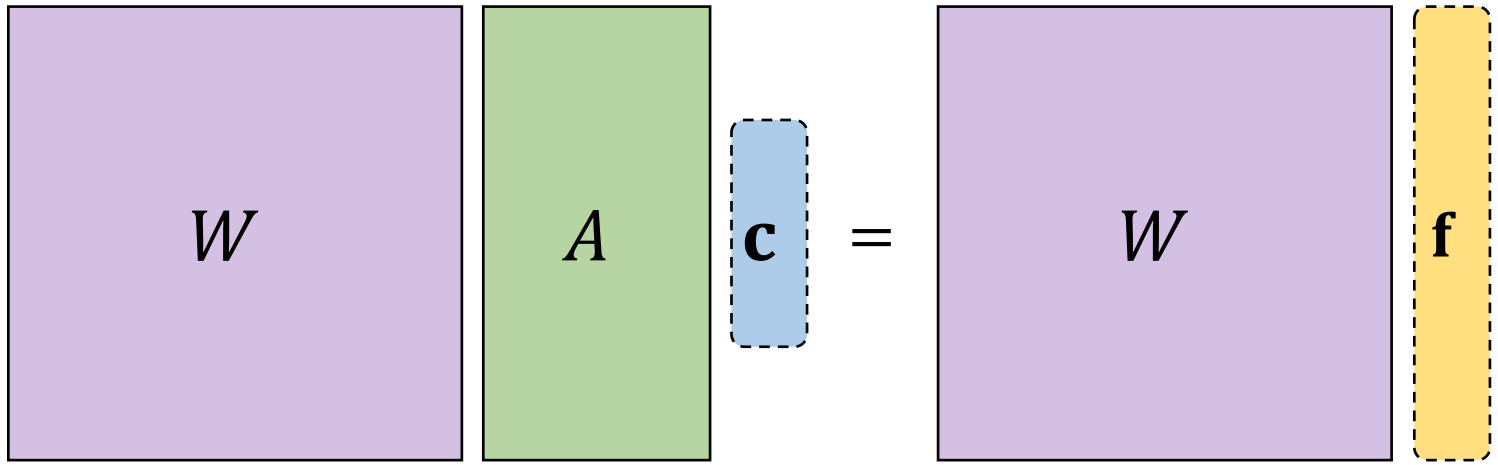
- 各データ点ごとの誤差に、重み w_i をつける
  - 重要度、確信度

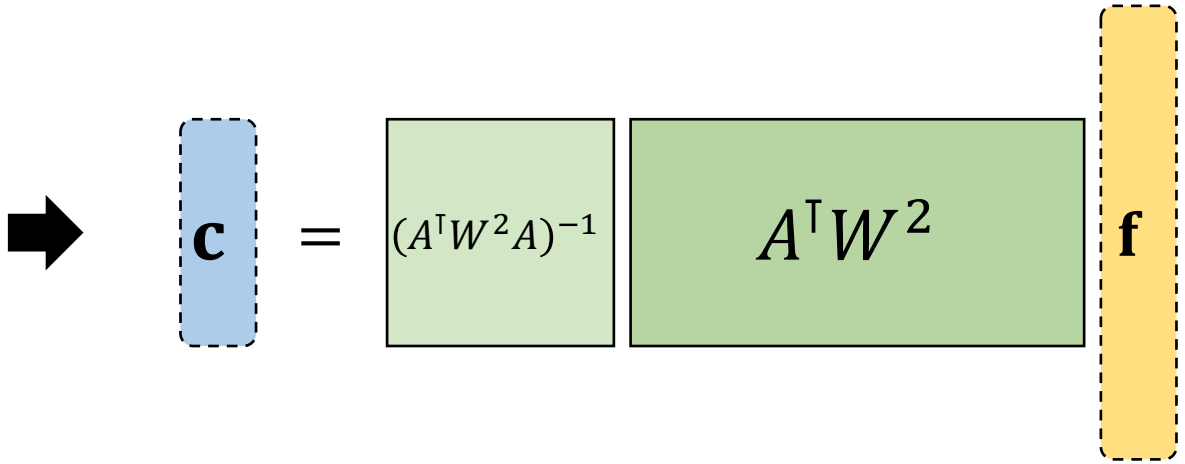
- 以下の誤差を最小化：

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



Weighted **L**east **S**quares (重み付き最小二乗)





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

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

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

- 評価点ごとに
  方程式を解き直す

法線制約の導入

- 各データ点が表す 1 次式を考える：

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

- 各 g_i を現在位置で評価したときの誤差を最小化：

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

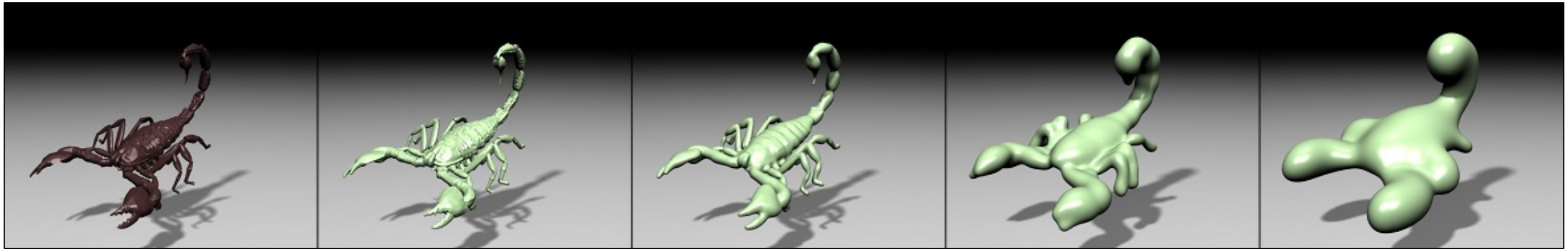
$$\begin{bmatrix} w_1(\mathbf{x}) & & & \\ & w_2(\mathbf{x}) & & \\ & & \ddots & \\ & & & w_N(\mathbf{x}) \end{bmatrix} \begin{bmatrix} x & y & z & 1 \\ x & y & z & 1 \\ & & \vdots & \\ x & y & z & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} w_1(\mathbf{x}) & & &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

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)

注意：数学的な原理について私自身が理解できていません 😢

RBFの式

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

- $\phi(\cdot)$ は基底関数
  - $\phi(\mathbf{x}) = \|\mathbf{x}\|^3$
- $N + 4$個の係数を求めるための制約条件
  - $f(\mathbf{x}_i) = f_i \quad (i = 1,2,\dots,N)$
  - $\sum w_i = \sum x_i w_i = \sum y_i w_i = \sum z_i w_i = 0$
- よく言われているが、理解できていない:
  - 2Dの場合 $\phi(\mathbf{x}) = \|\mathbf{x}\|^2 \log\|\mathbf{x}\|$ の方が良い (?)
  - Biharmonicエネルギー $\int_{\mathbb{R}^d} \|\nabla^2 f(\mathbf{x})\|^2 d\mathbf{x}$ を最小化する (?)
    - $\nabla^2 f(x,y) \coloneqq (f_{xx}, f_{xy}, f_{yx}, f_{yy})$

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

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

$\boldsymbol{\Phi}$ $\mathbf{P}$ $\mathbf{P}^\intercal$ $\mathbf{w}$ $\mathbf{c}$ $\mathbf{f}$

https://en.wikipedia.org/wiki/Polyharmonic_spline
Scattered Data Approximation (Wendland, 2004)

- 基底関数として他の選択肢もあり得る (e.g. $e^{-\|\mathbf{x}\|^2/\sigma^2}$)
  - ただし $\|\mathbf{x}\|^2$ は行列が特異になるのでダメ!

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：

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

未知数の3Dベクトル

- f の勾配：

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

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

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

Hessian 行列 (関数)

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：

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

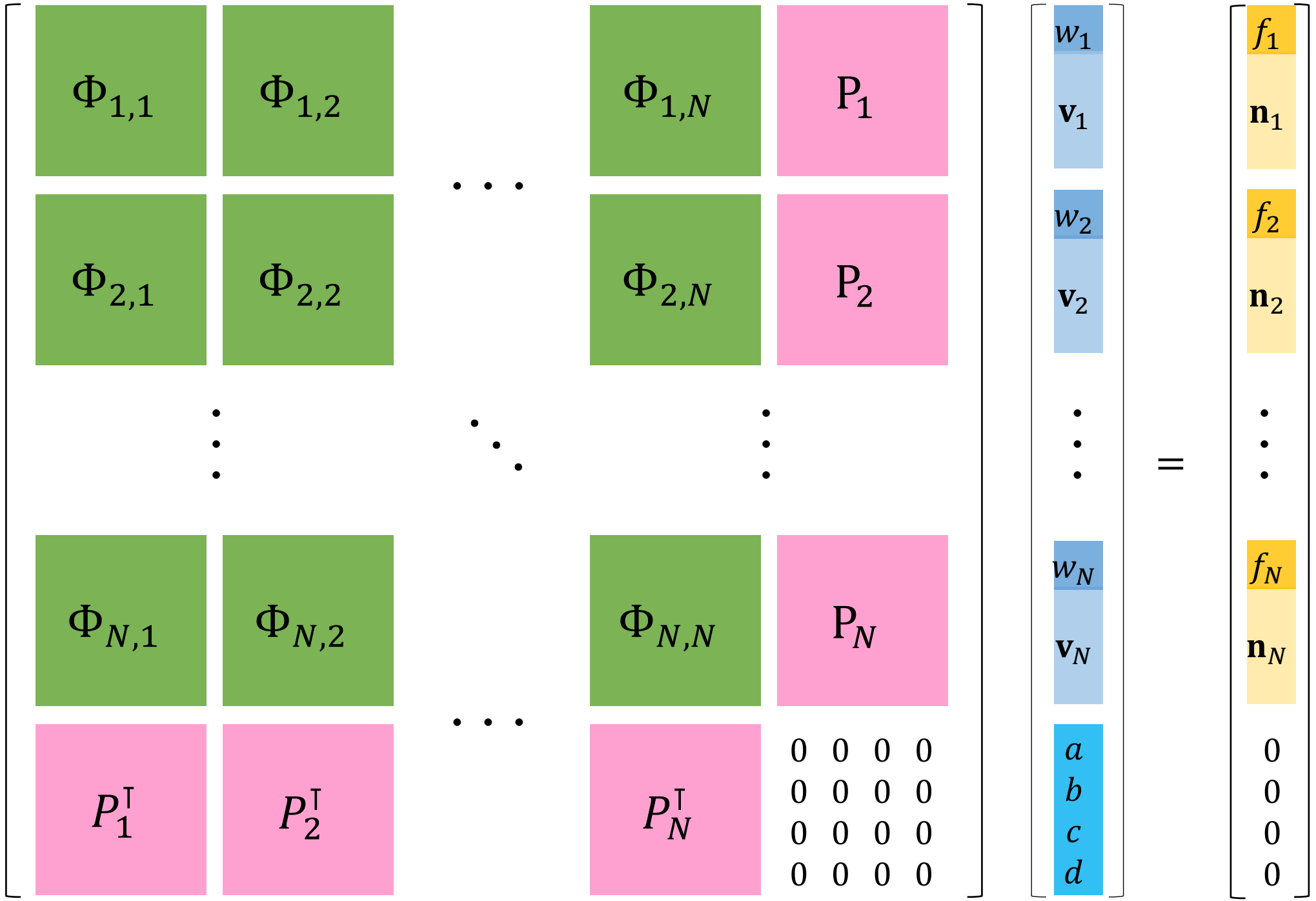
勾配の制約：

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

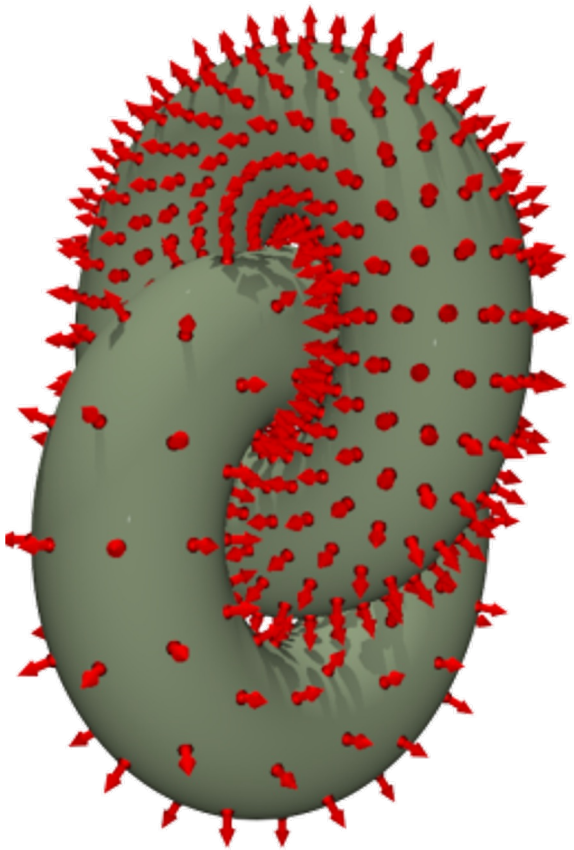
$$\begin{bmatrix} \phi_{1,1} & (\nabla\phi_{1,1})^{\mathsf{T}} \\ \nabla\phi_{1,1} & \mathrm{H}_{\phi}^{1,1} \end{bmatrix} \begin{bmatrix} \phi_{1,2} & (\nabla\phi_{1,2})^{\mathsf{T}} \\ \nabla\phi_{1,2} & \mathrm{H}_{\phi}^{1,2} \end{bmatrix} \cdots \begin{bmatrix} \phi_{1,N} & (\nabla\phi_{1,N})^{\mathsf{T}} \\ \nabla\phi_{1,N} & \mathrm{H}_{\phi}^{1,N} \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ \mathbf{v}_1 \\ w_2 \\ \mathbf{v}_2 \\ \vdots \\ w_N \\ \mathbf{v}_N \\ a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f_1 \\ \mathbf{n}_1 \end{bmatrix}$$

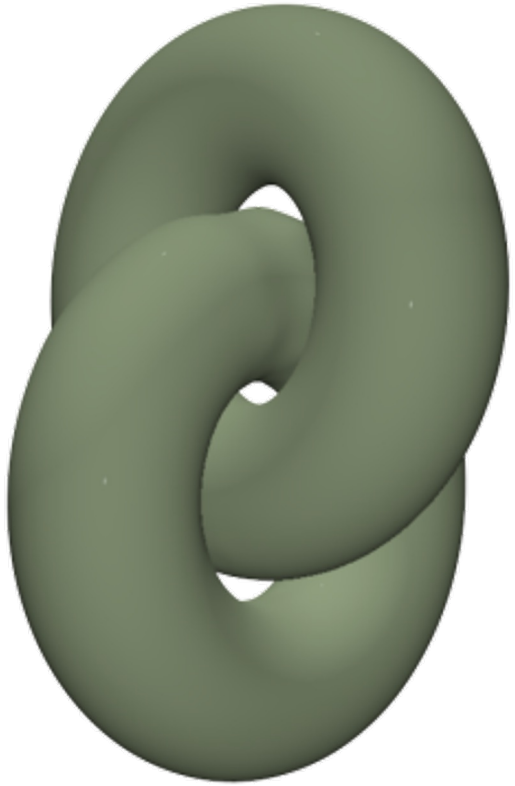
$\Phi_{1,1}$ $\Phi_{1,2}$ $\cdots$ $\Phi_{1,N}$ P_1

Hermite Radial Basis Functions Implicits [Macedo CGF10]

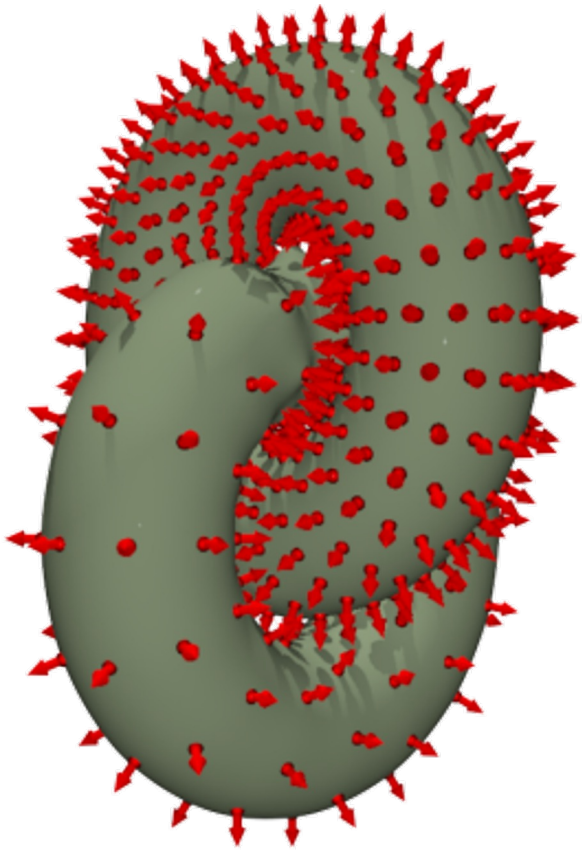

$\Phi_{1,1}$ $\Phi_{1,2}$ $\Phi_{1,N}$ P_1
$\Phi_{2,1}$ $\Phi_{2,2}$ $\Phi_{2,N}$ P_2
$\Phi_{N,1}$ $\Phi_{N,2}$ $\Phi_{N,N}$ P_N
$P_1^\intercal$ $P_2^\intercal$ $P_N^\intercal$
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
w_1 $\mathbf{v}_1$
w_2 $\mathbf{v}_2$
w_N $\mathbf{v}_N$
a b c d
=
f_1 $\mathbf{n}_1$
f_2 $\mathbf{n}_2$
f_N $\mathbf{n}_N$
0 0 0 0

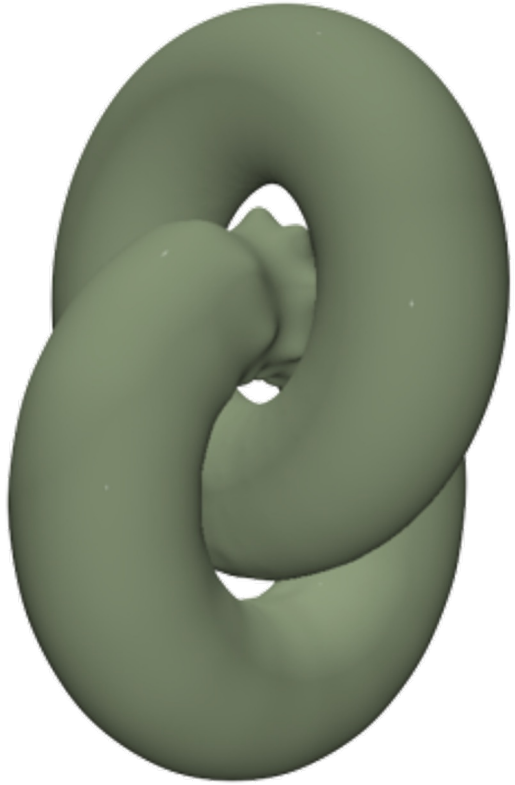
比較





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：

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

$$= \mathbf{n}^{\mathsf{T}} H\, \mathbf{n}$$

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

行列 H は $\{\mathbf{x}_i\}$ のみに依存して定まる

- 2次制約付き2次計画問題として定式化：

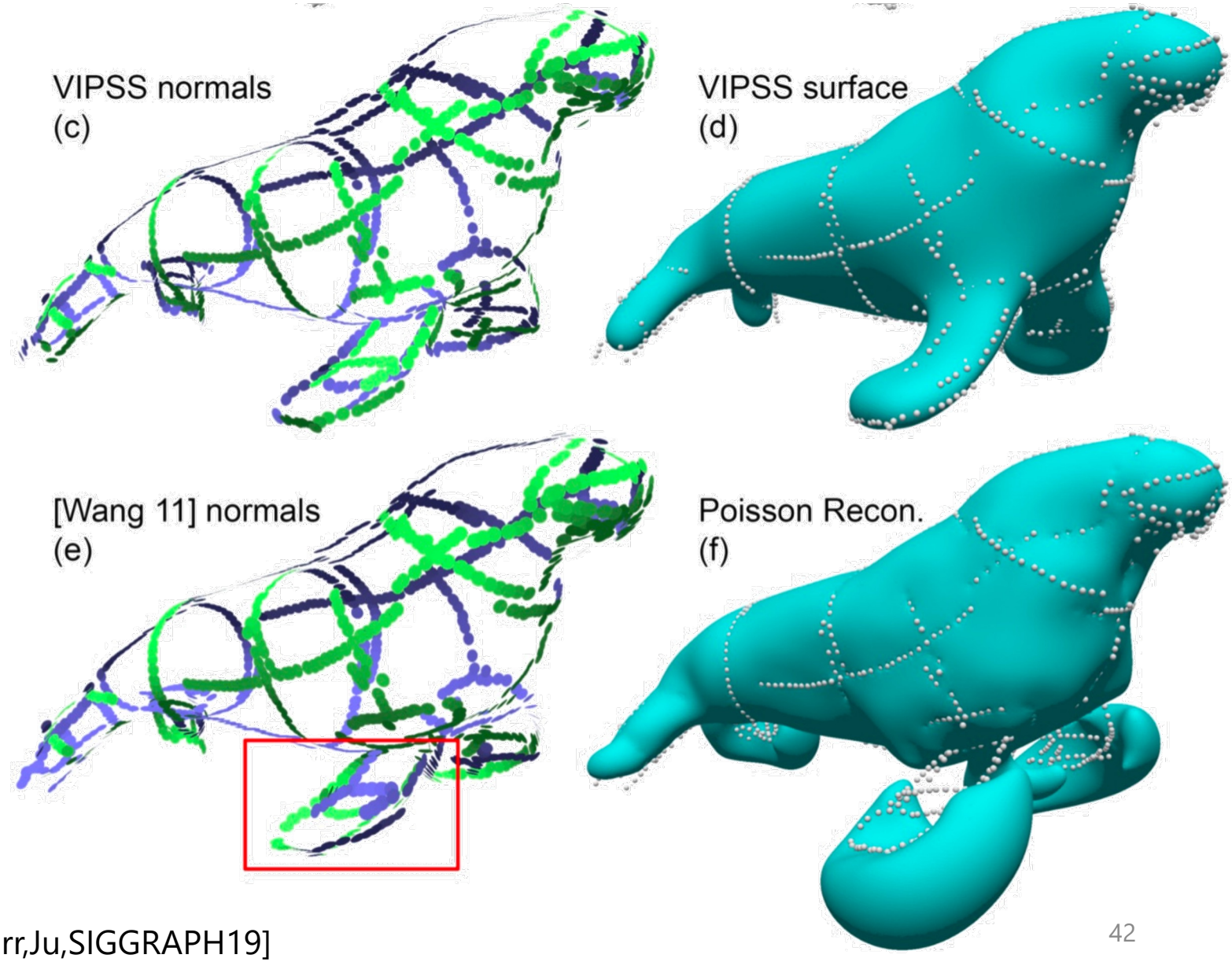
$$\text{minimize } \mathbf{n}^{\mathsf{T}} H\, \mathbf{n}$$

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

Variational Implicit Point Set Surfaces [Huang,Carr,Ju,SIGGRAPH19]

最近の研究：グローバルな法線推定

入力点群



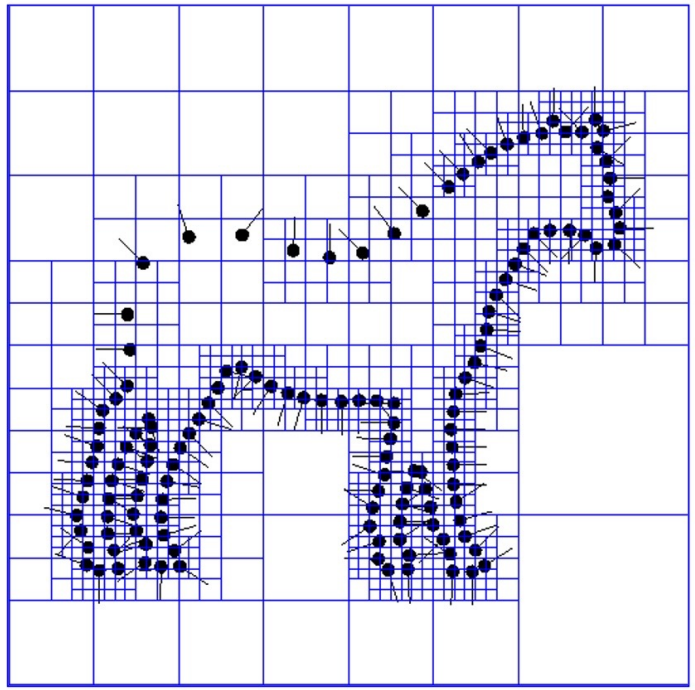
Variational Implicit Point Set Surfaces [Huang,Carr,Ju,SIGGRAPH19]

Poisson Surface Reconstruction

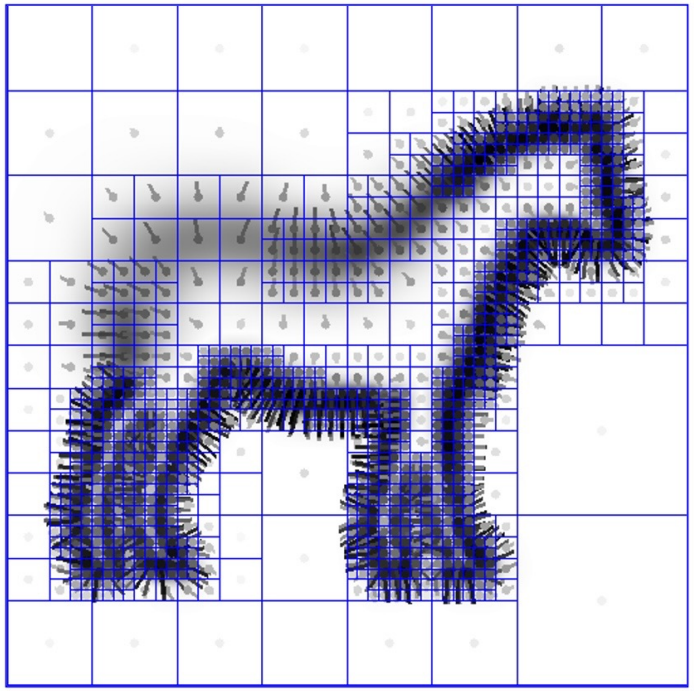
Poisson Surface Reconstruction (PSR)

Kazhdan et al., Symposium on Geometry Processing 2006

Cited by 3437



点群に基づいて空間を
適応的に分割する



法線ベクトルを補間して
ベクトル場 $\vec{V}$ を作る

陰関数 f を以下のように求める：

$$f^* = \arg\min_f \int_B \left\| \nabla f - \vec{V} \right\|^2 dx$$

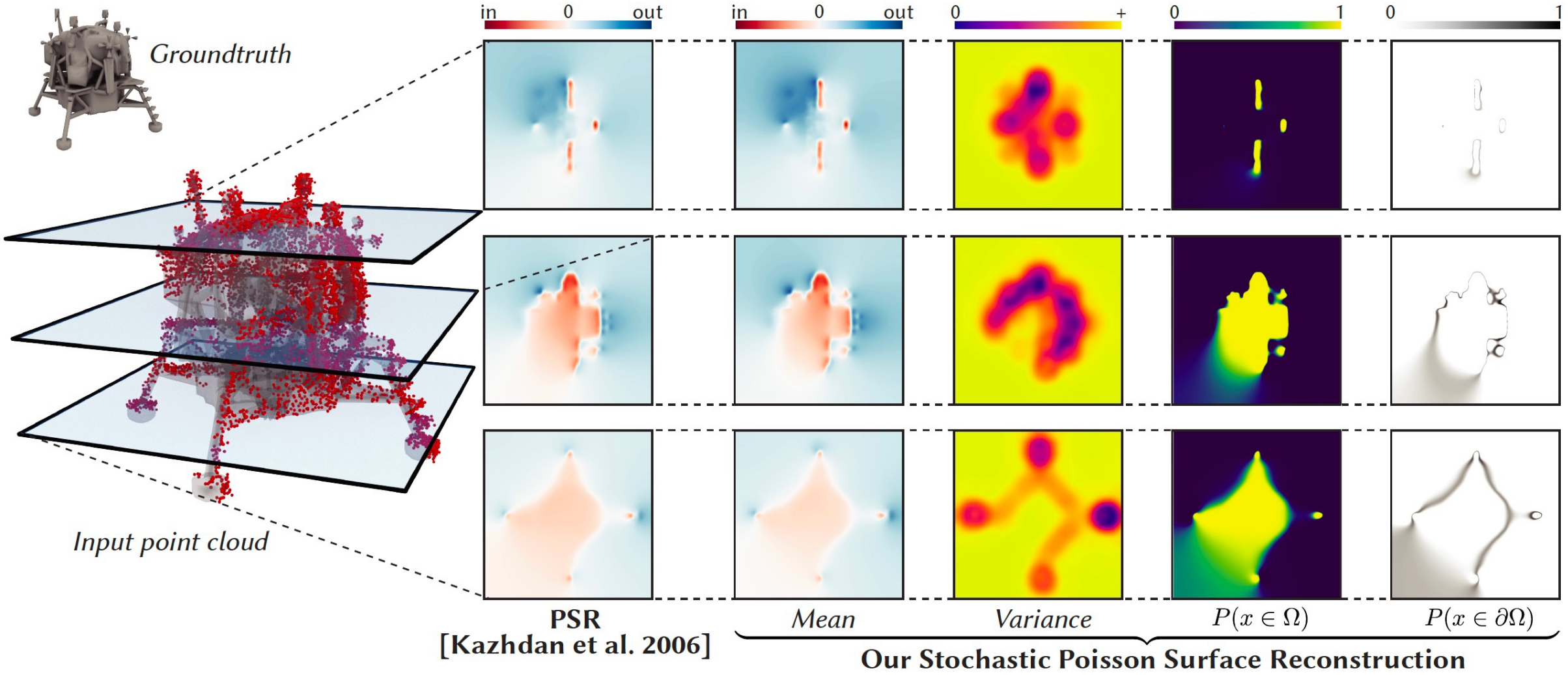
⬇️

$$\Delta f^* = \nabla \cdot \vec{V}$$

Poisson equation

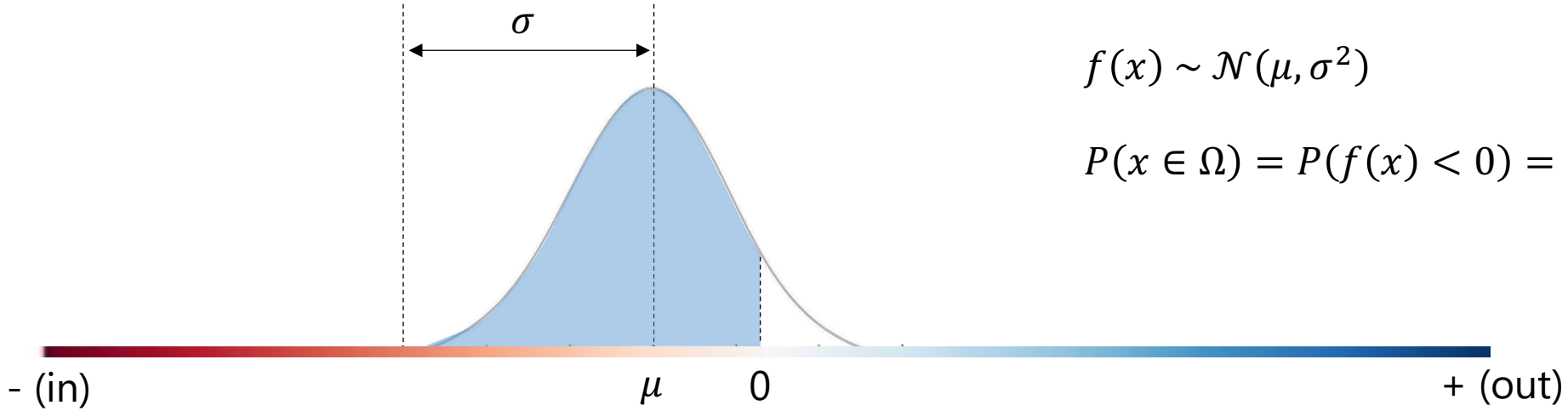
- その後も同じ著者よりいくつか改善版が出ている (2013, 2018, 2020)
- 非常に実用的、業界のデファクトスタンダード
- https://github.com/mkazhdan/PoissonRecon

最近の研究：Stochastic PSR [SIGGRAPH Asia 2022]



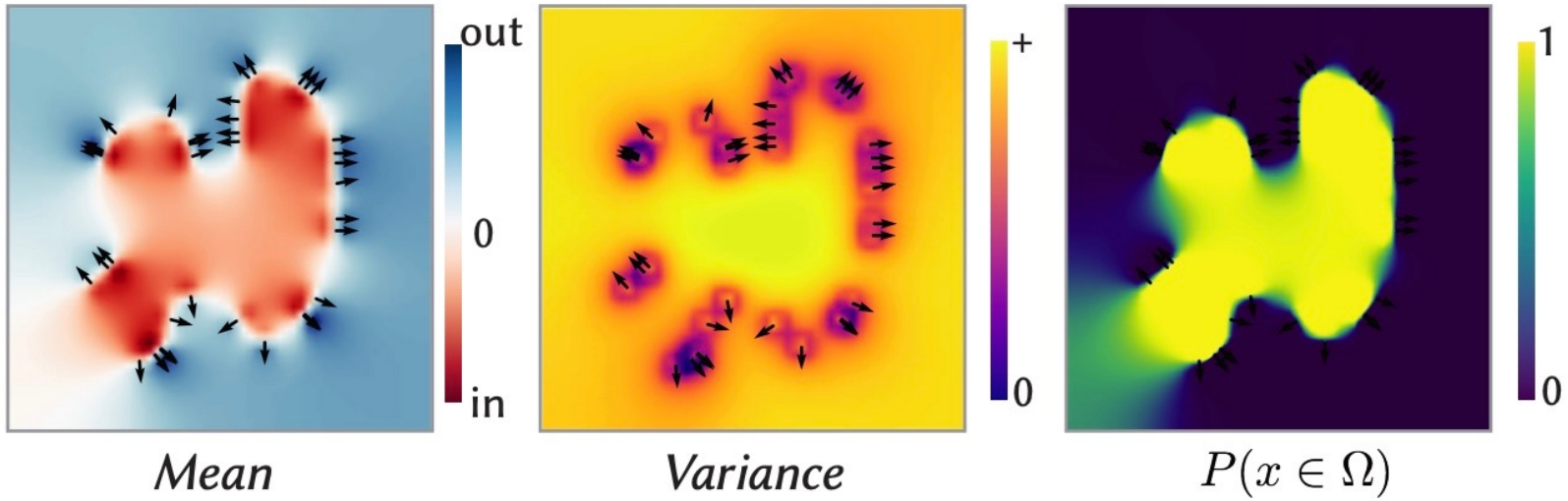
各点において、陰関数の値そのものではなく、値の平均と分散 (ガウス分布) を求める！
➔ 統計量が分かると色々嬉しい

メリット1：点が物体内部にあるかどうかの確率の計算



$$f(x) \sim \mathcal{N}(\mu, \sigma^2)$$

$$P(x \in \Omega) = P(f(x) < 0) = CDF_{\mu,\sigma^2}(0)$$

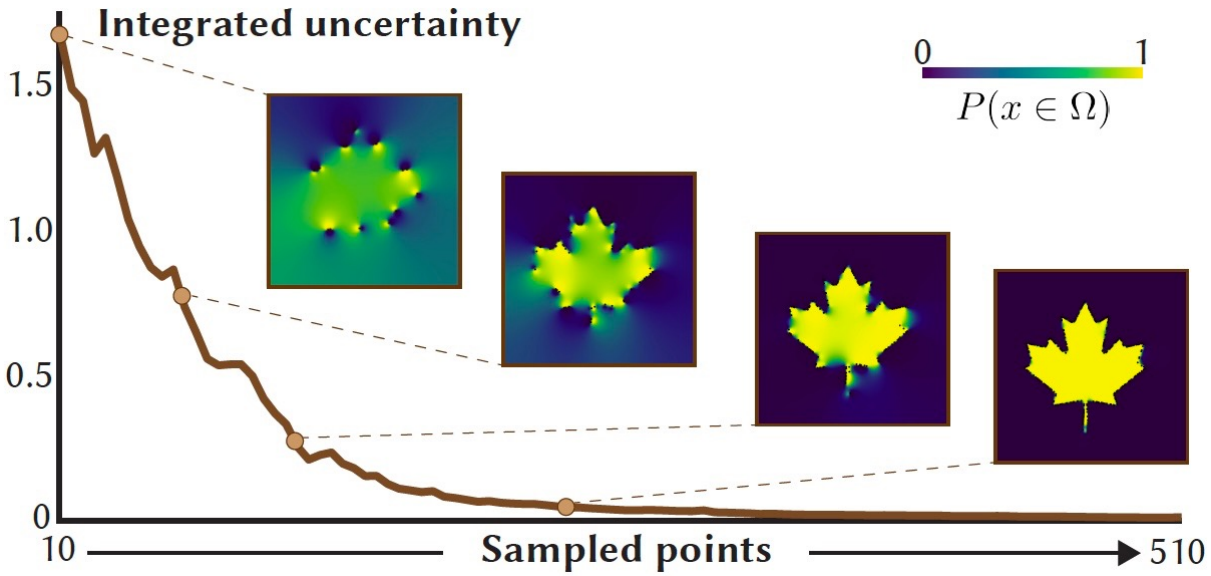


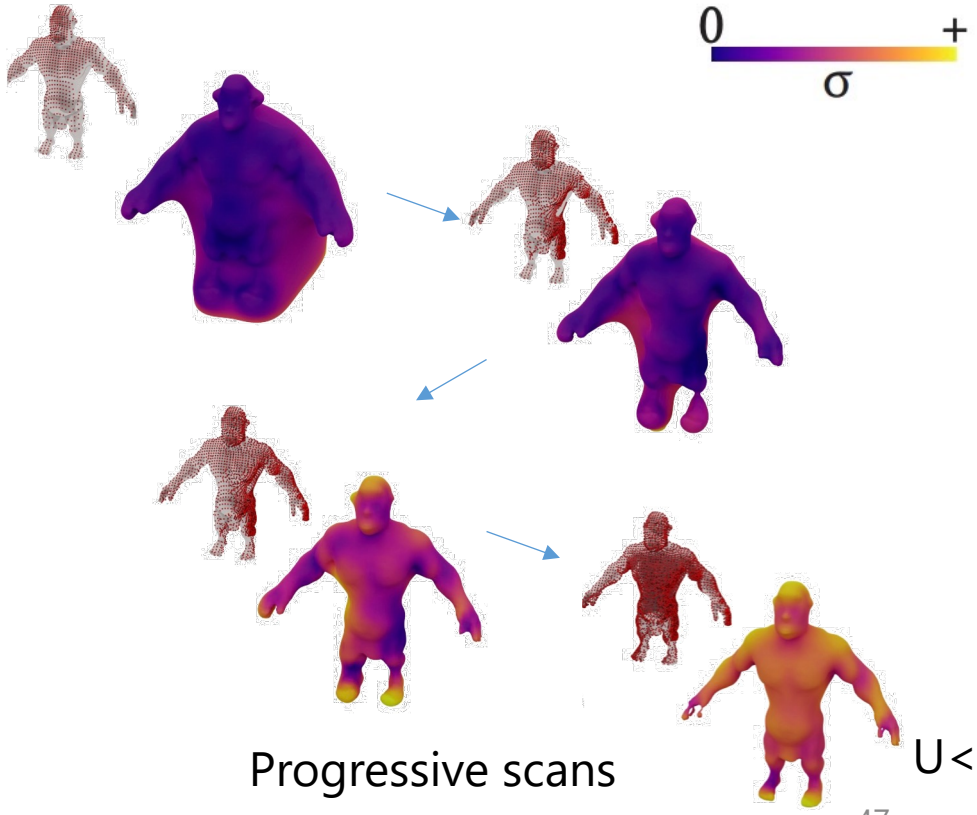
メリット1：点が物体内部にあるかどうかの確率の計算

➔ 「Total Uncertainty」という量を計算できる

$$U_{SPSR} = \int_B (0.5 - |P(x \in \Omega) - 0.5|)\ dx$$

➔ スキャン追加の必要の有無を判定するのに使える





Progressive scans

U<0.1

メリット2：同時確率を用いた、確率的な衝突判定

$$R_s = \{r_1, \ldots, r_s\} \subset R$$

領域Rからランダムにs個のサンプル点を取る

↓

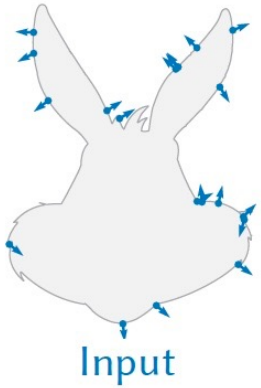
$$\mathbf{f}(r_1, \ldots, r_s) \,|\mathcal{S} \sim \mathcal{N}(\mathbf{W}\mathbf{f}_{SPSR}, \mathbf{W}^\top \mathbf{K}_\mathbf{f} \mathbf{W})$$

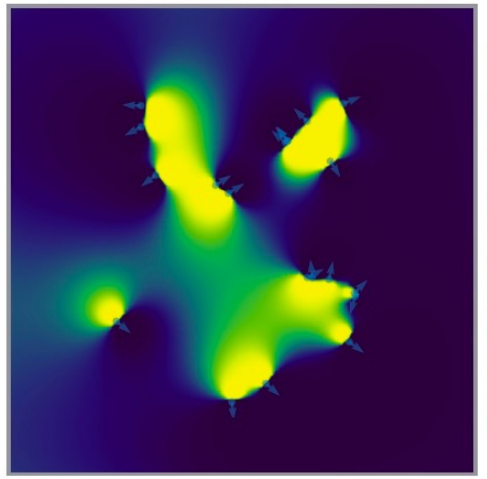
各サンプル点における確率分布を計算

↓

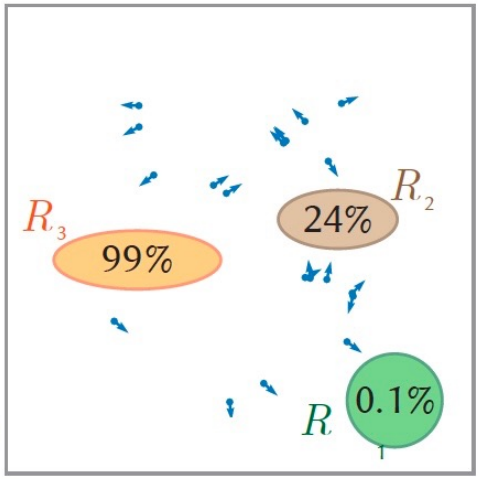
$$p(\mathbf{f}(r_1) > 0, \ldots, \mathbf{f}(r_s) > 0))$$

全サンプル点が物体外部である確率を計算





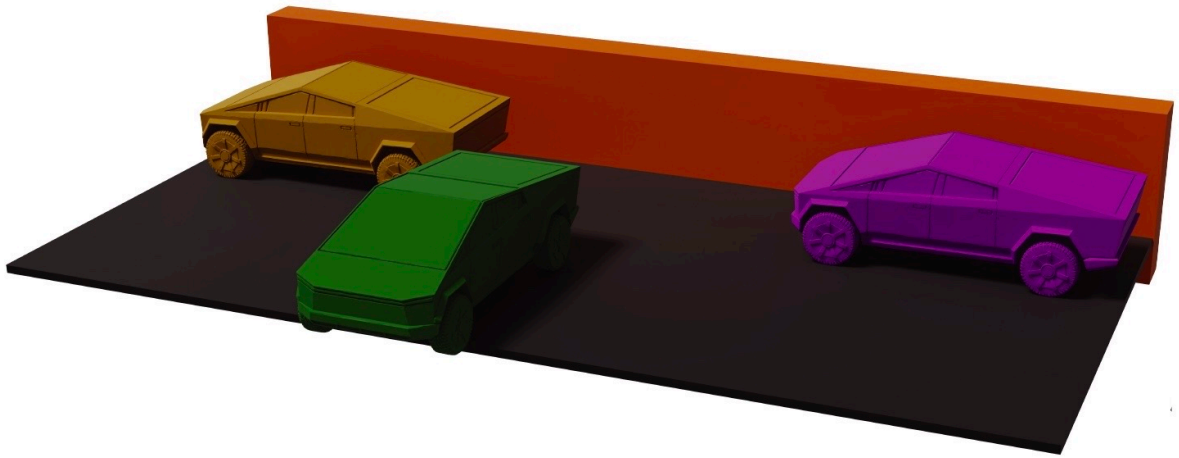
Individual probability
$P(x \in \Omega)$



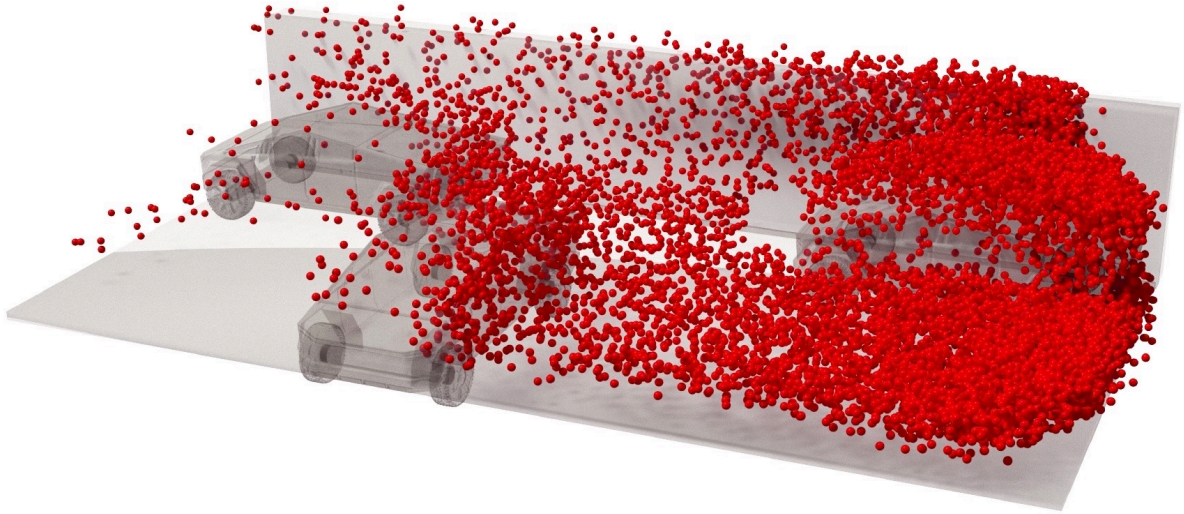
Regional probability
$P(R \cap \Omega \neq \emptyset)$

メリット2：同時確率を用いた、確率的な衝突判定

自動運転のシナリオ

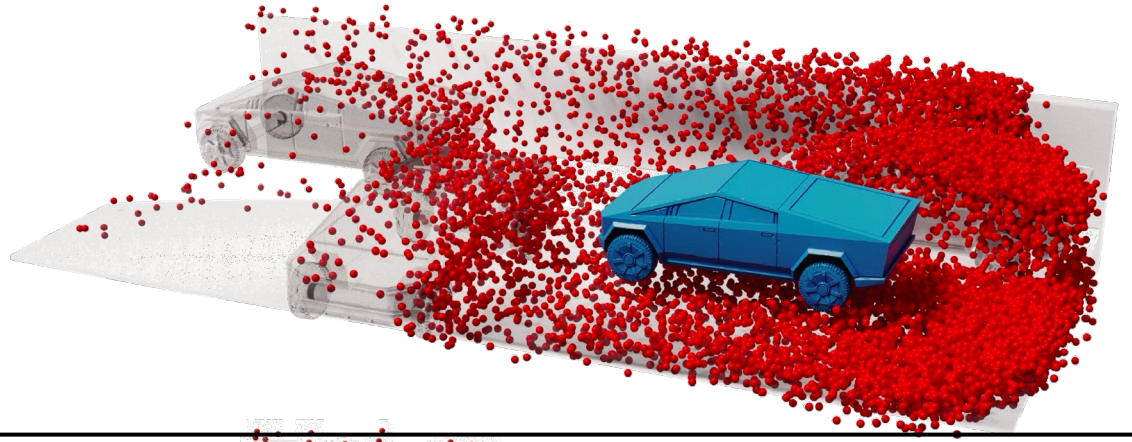


3Dシーン

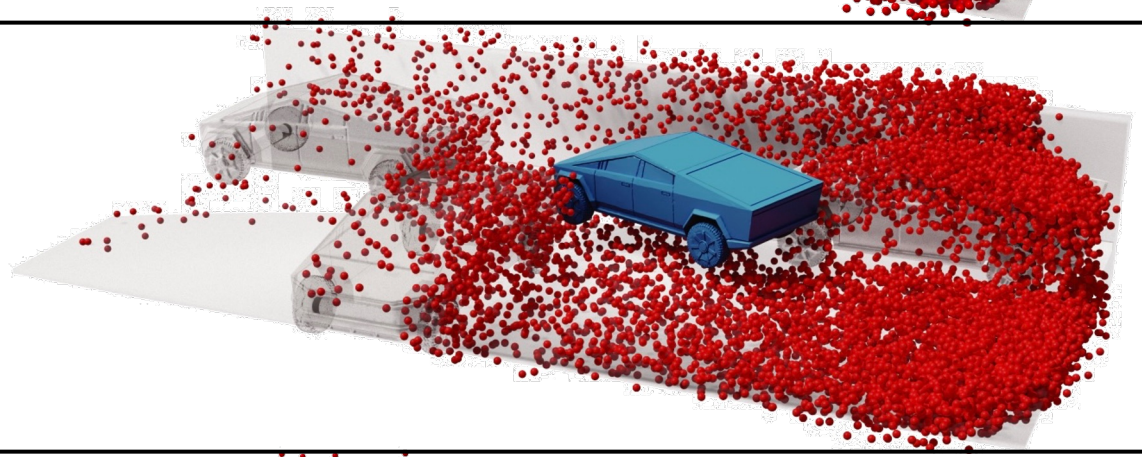


スキャンを模した点群データ

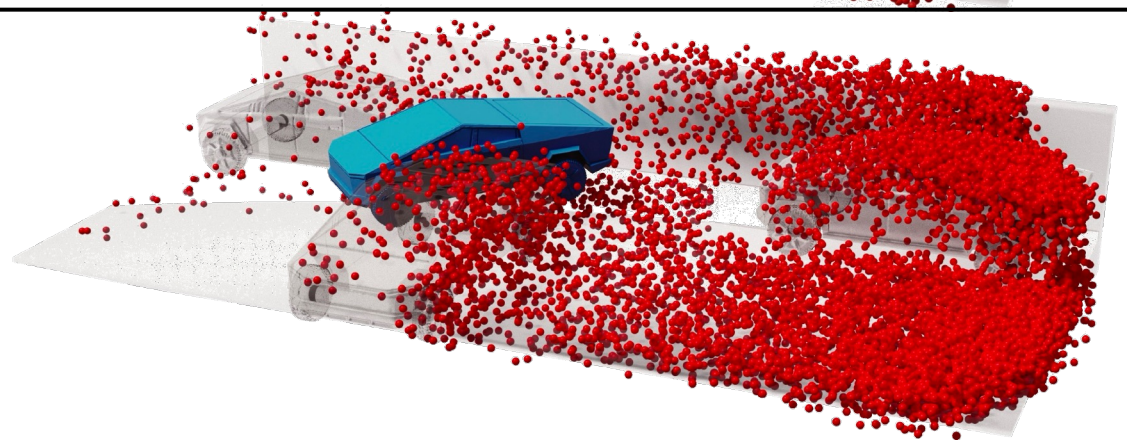
メリット2：同時確率を用いた、確率的な衝突判定



PSRによる判定	SPSRによる判定
衝突無し	0.56%の確率で衝突



衝突無し	36%の確率で衝突

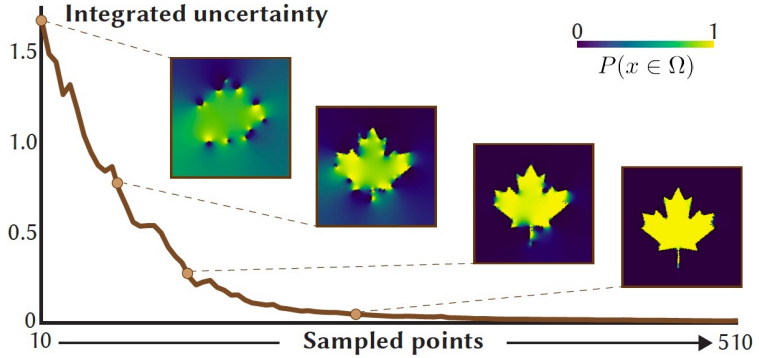


衝突有り	99%の確率で衝突

最近の研究: SPSRを応用した法線方向の推定

[Sellan22] が提案した
"uncertainty"指標:

$$U_{SPSR} = \int_{\Omega} \frac{1}{2} - |P(\mathbf{x} \in M) - \frac{1}{2}| d\mathbf{x}$$



点群の欠損/ノイズが減るほど、uncertaintyが下がる

提案: 符号付きのuncertainty指標

$$F = \int_{\Omega} \phi_k(\mathbf{x})[P(\mathbf{x} \in M) - \frac{1}{2}] d\mathbf{x}$$

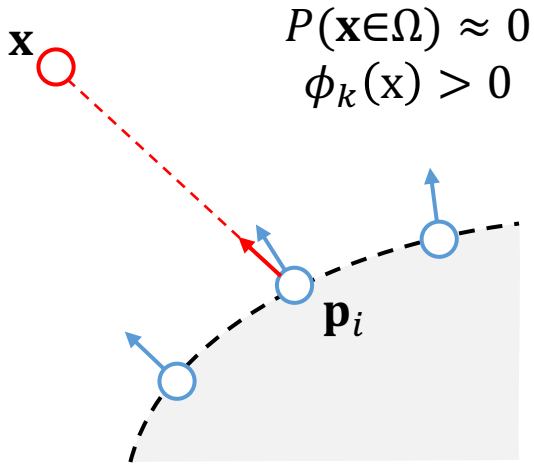
$$\phi_k(\mathbf{x}) = \sum_{n=1}^{k} \frac{(\mathbf{x} - \mathbf{p}_i) \cdot \mathbf{n}_i}{\|\mathbf{x} - \mathbf{p}_i\|_2}$$

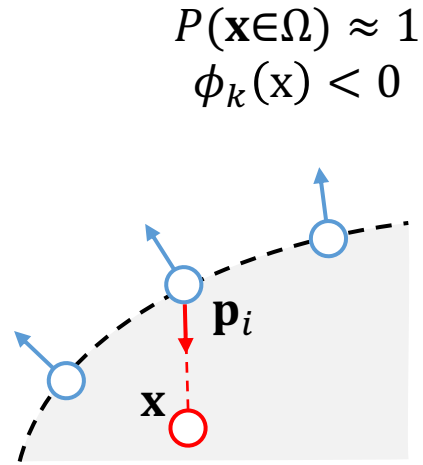
点$\mathbf{x}$のk近傍

$$G = \frac{1}{|S|} \sum_{\mathbf{p}_i \in S} (\mu_i - \hat{\mu})^2, \quad \hat{\mu} = \frac{1}{|S|} \sum_{\mathbf{p}_i \in S} \mu_i$$

正規化項

$\mathbf{p}_i$における
indicator関数の値





L-BFGSで最適化

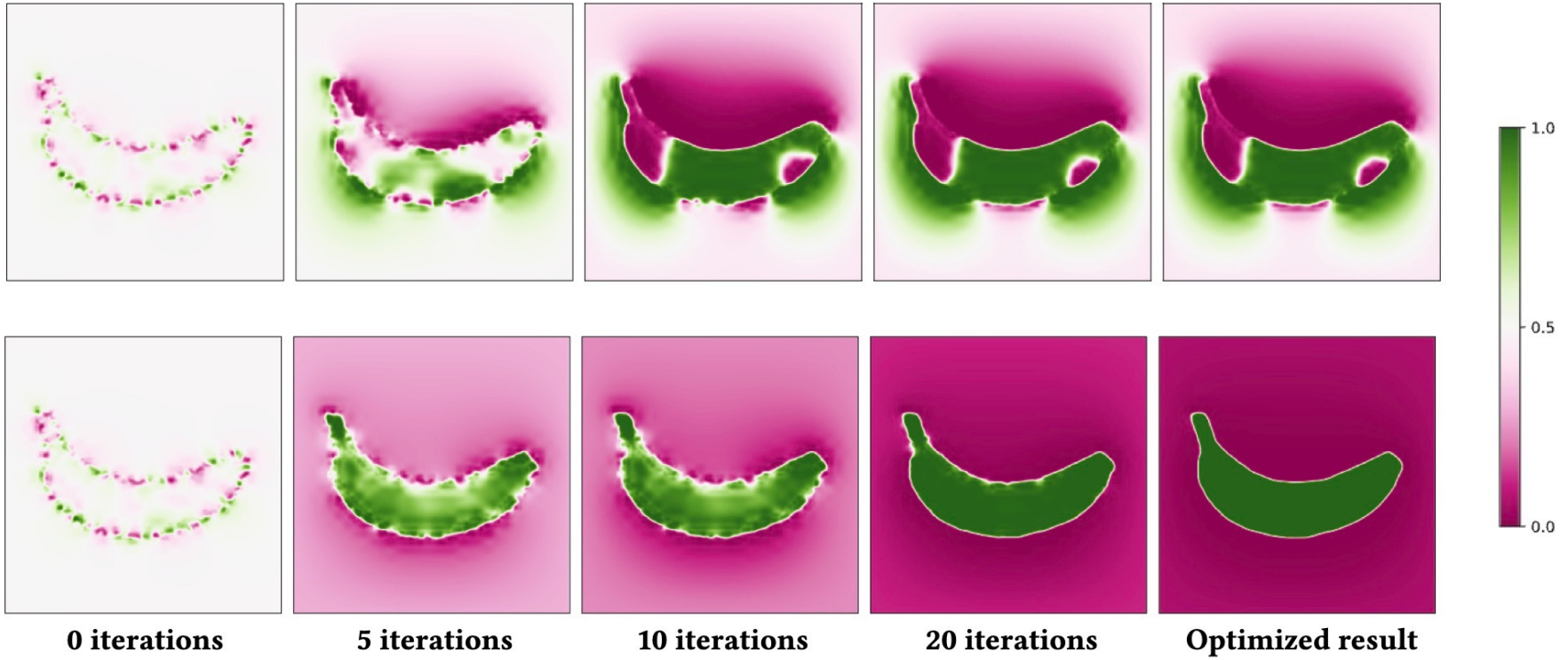
- 法線方向を極座標で表す

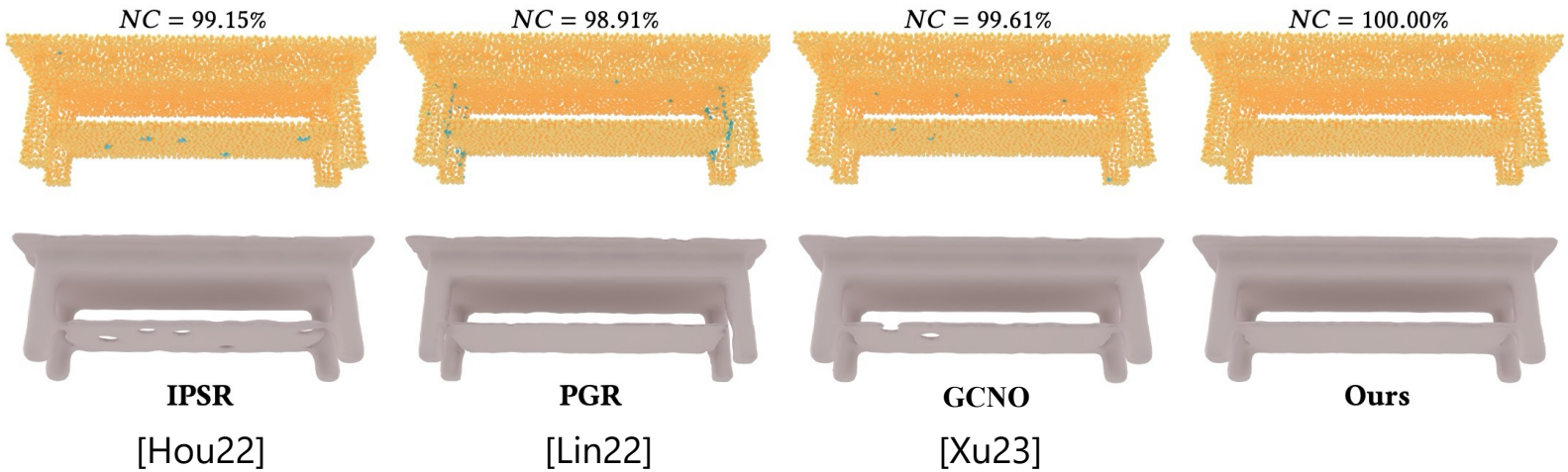
Stochastic Normal Orientation for Point Clouds [Huang,Fang,Zhang,Liu,Fu,SIGGRAPHAsia24]

U_{SPSR}の最小化

Random normals

提案法





[Hou22] [Lin22] [Xu23]

Stochastic Normal Orientation for Point Clouds [Huang,Fang,Zhang,Liu,Fu,SIGGRAPHAsia24]

最近の研究: 点群を直接エッジで繋ぐアプローチ

第1回Spatial AI勉強会

Surface Reconstruction Using Rotation Systems

(SIGGRAPH Asia 2024 Journal Track)

Ruiqi Cui
Technical University of Denmark Anker Engelunds Vej 101Lyngby Denmark
ruicu@dtu.dk

Emil Toftegaard Gæde
Technical University of Denmark Anker Engelunds Vej 101Lyngby Denmark
etoga@dtu.dk

Eva Rotenberg
Technical University of Denmark Anker Engelunds Vej 101Lyngby Denmark
erot@dtu.dk

Leif Kobbelt
Visual Computing Institute, RWTH Aachen University 1 Thørvald CircleAachen Germany
kobbelt@cs.rwth-aachen.de

J. Andreas Bærentzen
Technical University of Denmark Anker Engelunds Vej 101Lyngby Denmark
janba@dtu.dk

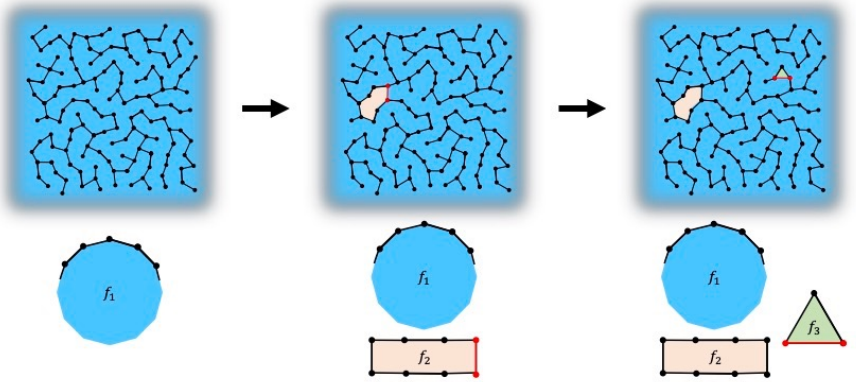
Project: https://cuirq3.github.io/projects/siga_24/
Code: https://github.com/cuirq3/RsR
arXiv: https://arxiv.org/abs/2402.01893
YouTube: https://youtu.be/9DEfhN3pzng

高山 健志 (CyberAgent)

2024年12月21日

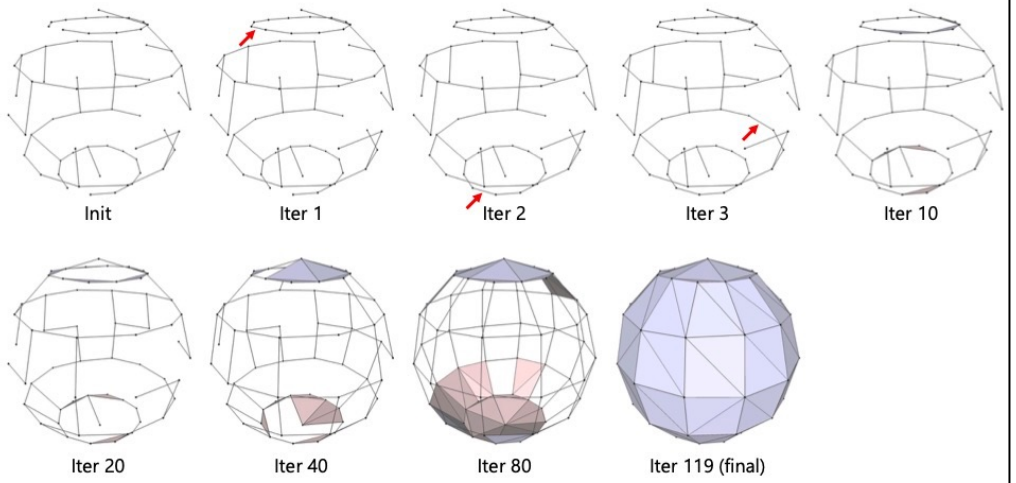
基本アイディア

- このポリゴンメッシュにエッジを挿入し、面を分割する
- 全ての面が三角形になるまで繰り返す



球に対する実行の様子

挿入候補エッジは、初期のk近傍グラフの中から短い順に選ぶ



Spatial AI勉強会スライド: https://speakerdeck.com/spatial_ai_network/surface-reconstruction-using-rotation-systems-siggraph-asia-2024

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

- https://en.wikipedia.org/wiki/Metaballs
- 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