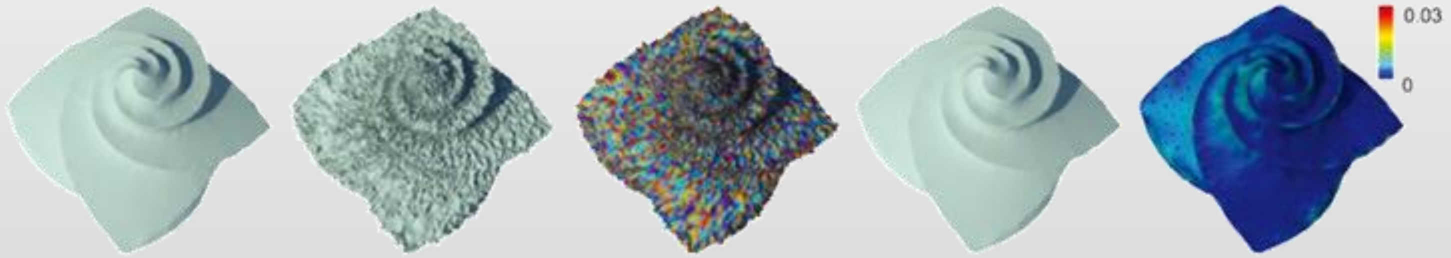
-形状分析-

Decoupling Noise and Features via Weighted l1-Analysis Compressed Sensing

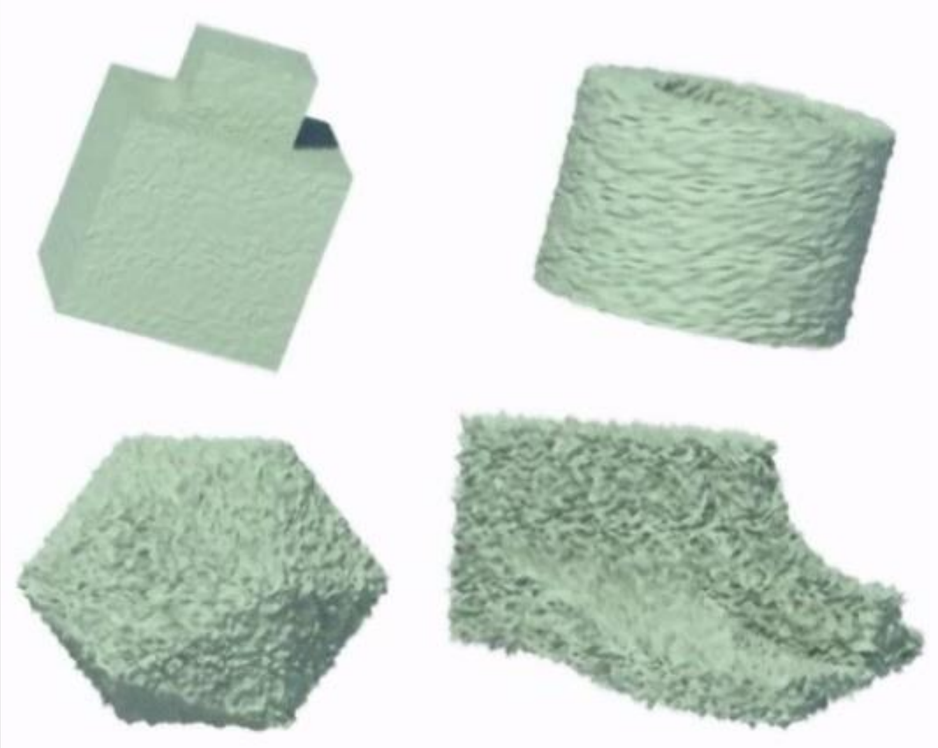
Ruimin Wang, Zhouwang Yang, Ligang Liu, Jiansong Deng, Falai Chen
(University of Science and Technology of China)



藤堂英樹
東京大学 / JST CREST

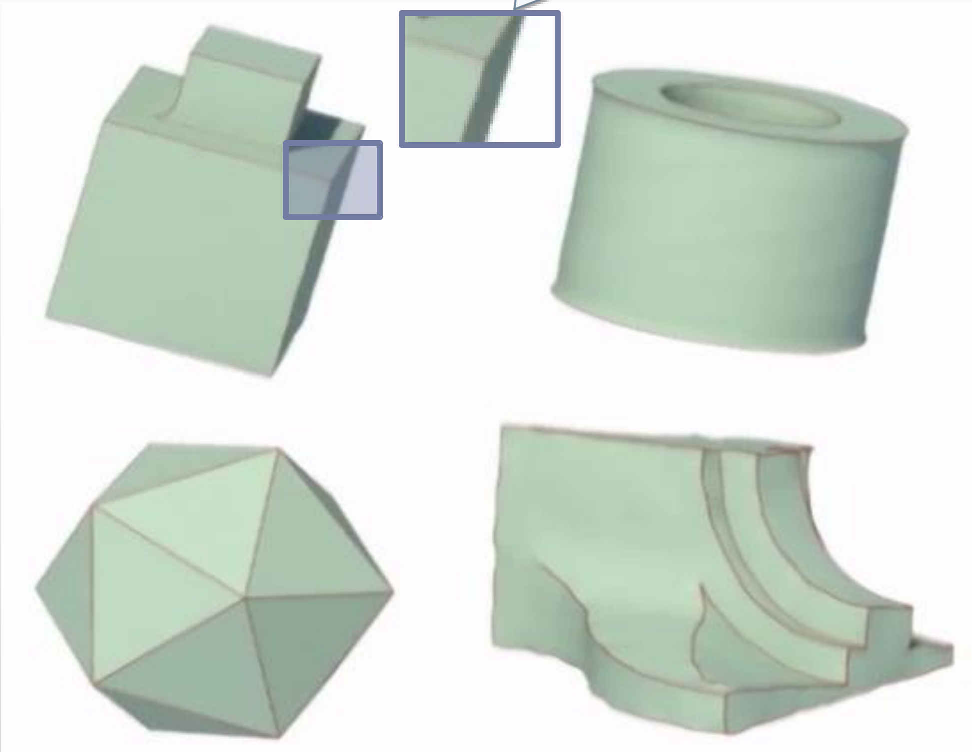
モチベーション

- 3Dスキャンデータの平滑化



3Dスキャンデータ

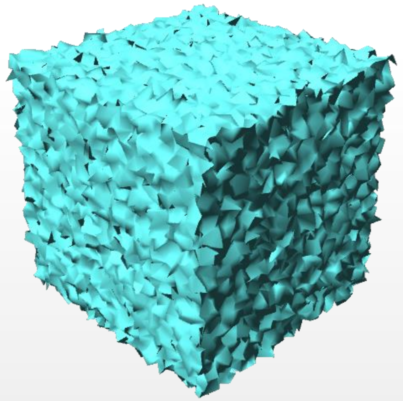
シャープな特徴が重要

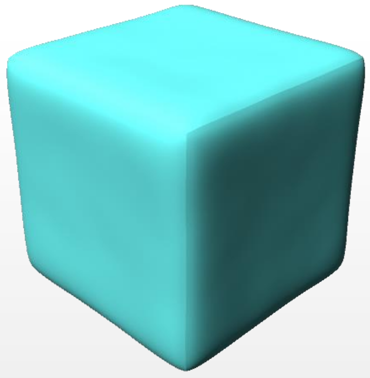


平滑化結果

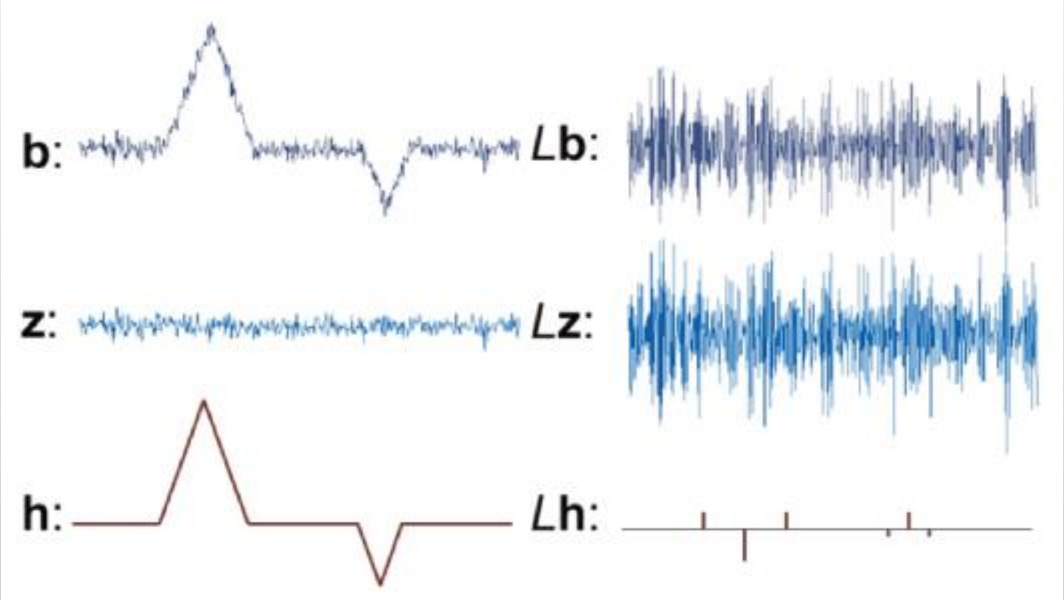
モチベーション

- 特徴とノイズを分けるのは難しい
  - 特徴はノイズがあると見つけにくい
  - ノイズを除去するとシャープな特徴がぼける





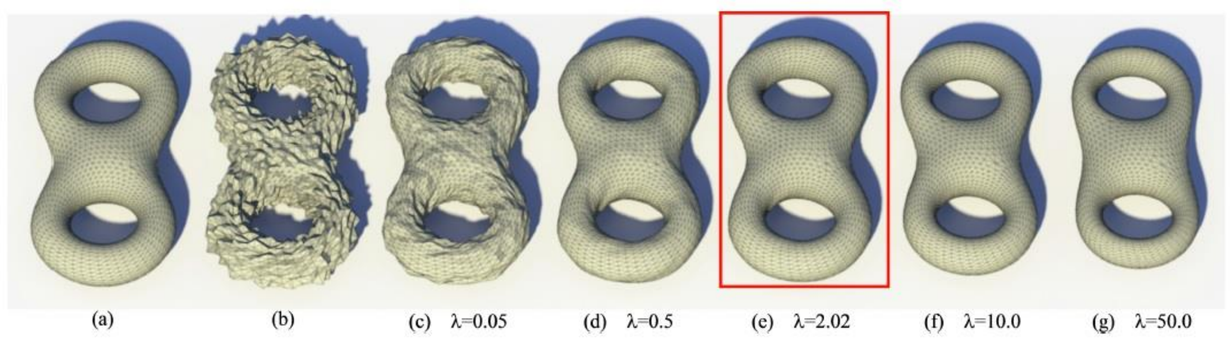
- 基本アイデア
  - シャープな特徴は疎
  - L1解析で疎な特徴を抽出できる



提案手法の概要

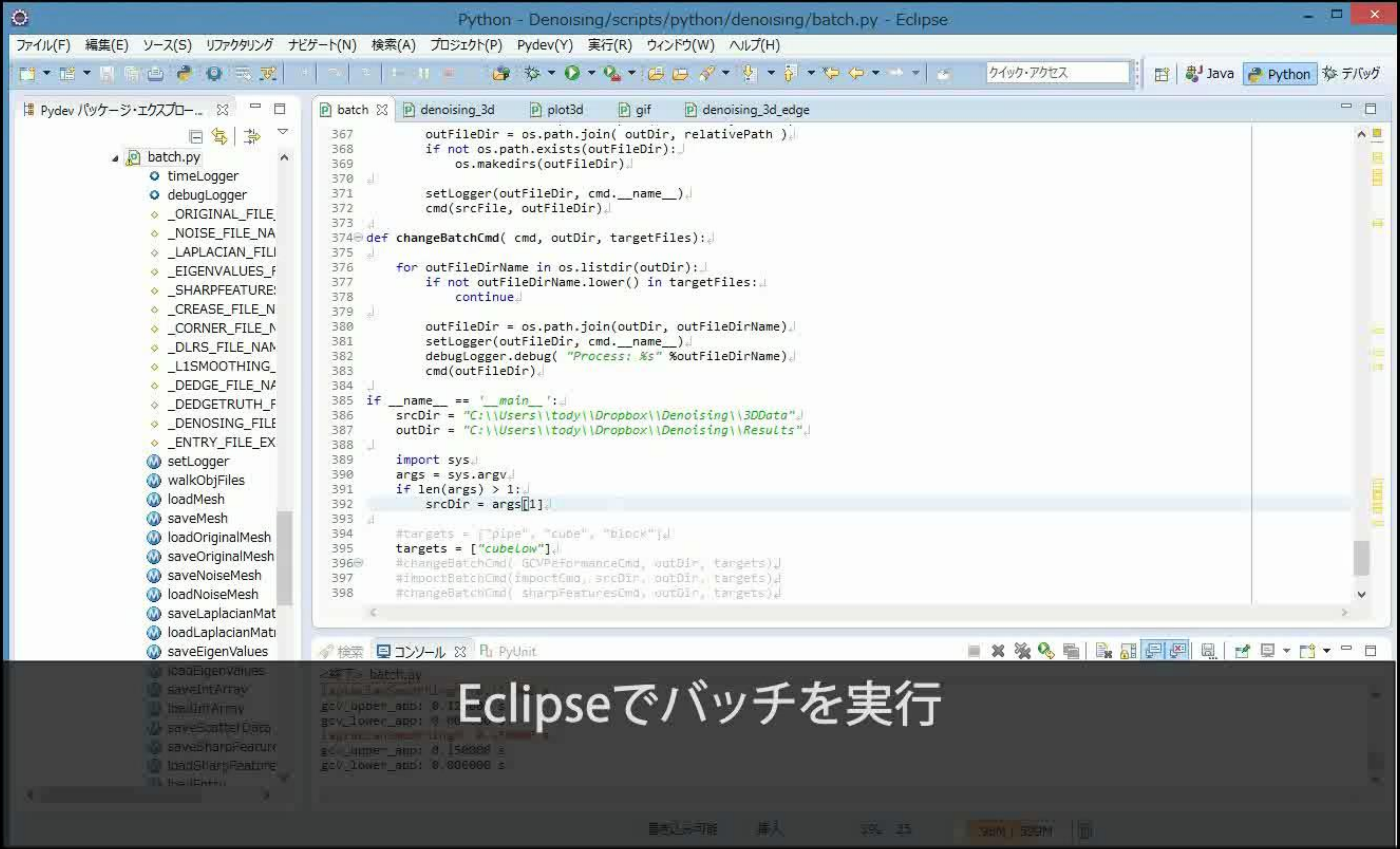
Phase I: GCVによる最適な平滑化

異なるノイズ設定に対して
最適なパラメータが自動的に決まる



Phase II: シャープさを考慮しつつ平滑化

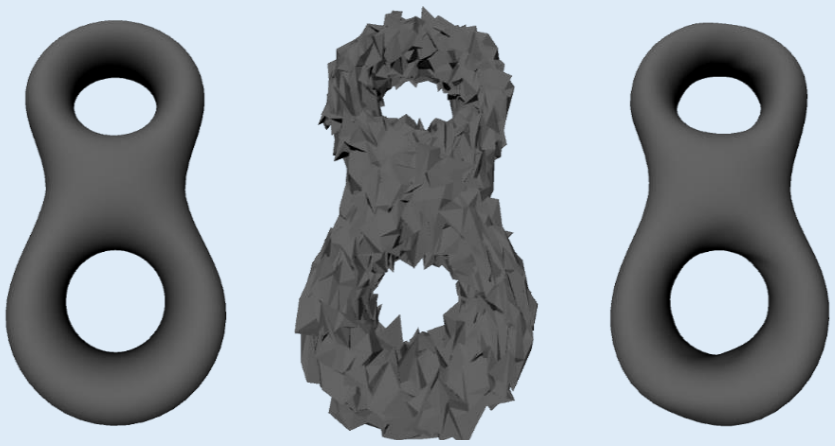
繰り返し：
・L1解析によるシャープな特徴の抽出
・シャープさを考慮しつつ平滑化


Python - Denoising/scripts/python/denoising/batch.py - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング ナビゲート(N) 検索(A) プロジェクト(P) Pydev(Y) 実行(R) ウィンドウ(W) ヘルプ(H)
batch denoising_3d plot3d gif denoising_3d_edge
outFileDir = os.path.join(outDir, relativePath)
if not os.path.exists(outFileDir):
os.makedirs(outFileDir)
setLogger(outFileDir, cmd.__name__)
cmd(srcFile, outFileDir)
def changeBatchCmd(cmd, outDir, targetFiles):
for outFileDirName in os.listdir(outDir):
if not outFileDirName.lower() in targetFiles:
continue
outFileDir = os.path.join(outDir, outFileDirName)
setLogger(outFileDir, cmd.__name__)
debugLogger.debug("Process: %s" %outFileDirName)
cmd(outFileDir)
if __name__ == '__main__':
srcDir = "C:\\Users\\tody\\Dropbox\\Denoising\\3DData"
outDir = "C:\\Users\\tody\\Dropbox\\Denoising\\Results"
import sys
args = sys.argv
if len(args) > 1:
srcDir = args[1]
targets = ["cubeLow"]
Eclipseでバッチを実行

前回までにできていた部分

Phase I: GCVによる最適な平滑化

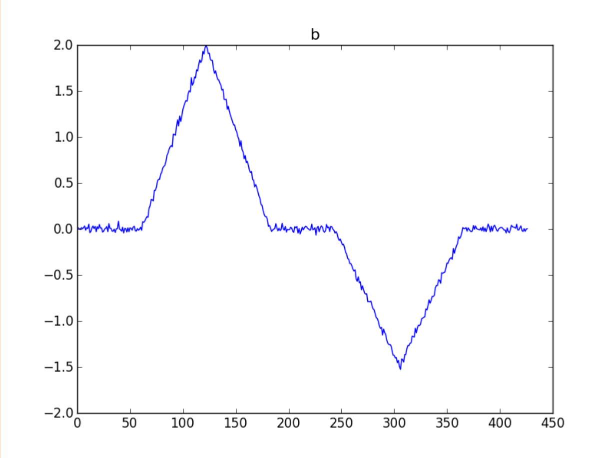
速度面を除いてPhase Iを再現



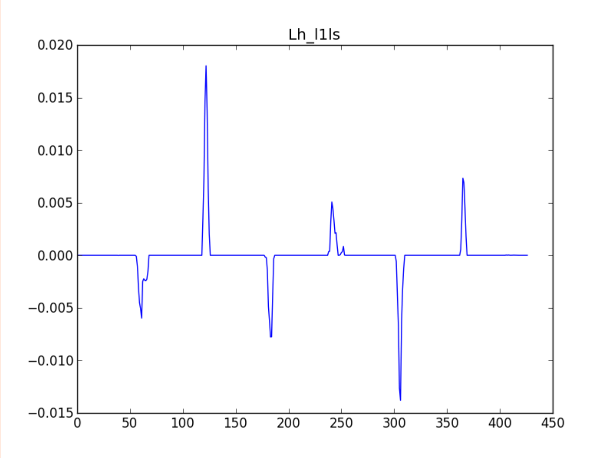
正解メッシュ　入力メッシュ（ノイズ）　$\lambda = 2.85$

Phase II: １次元信号でのシャープな特徴の解析

L1解析: $\min_{\mathbf{h}} \|\mathbf{h} - \mathbf{b}\|^2 + \tau \|W(L\mathbf{h})\|_1$



残差 $\mathbf{b}$



復元された特徴 $L\mathbf{h}$

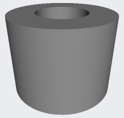
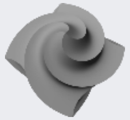
前回からの差分

- GCVの速度改善
- L1解析の精度改善
- シャープさを考慮した平滑化の実験
- 3次元メッシュでのL1特徴解析の実験
- メッシュ特徴に関する議論
- 他の特徴を使った特徴解析の試作

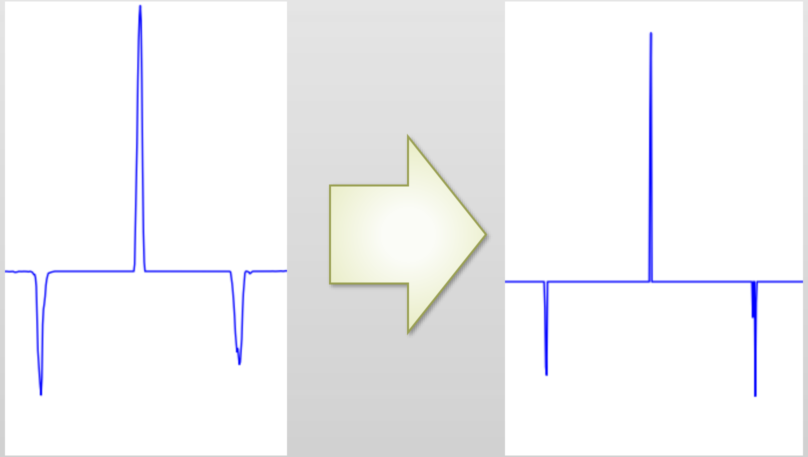
前回からの差分

- GCVの速度改善
- L1解析の精度改善
- シャープさを考慮した平滑化の実験
- 3次元メッシュでのL1特徴解析の実験
- メッシュ特徴に関する議論
- 他の特徴を使った特徴解析の試作

事前計算が1/3に

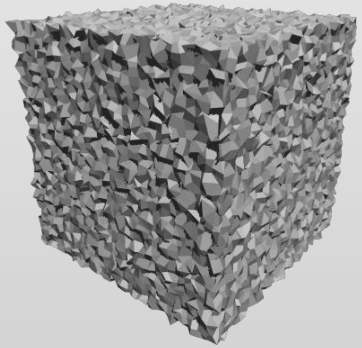
モデル		頂点数	固有値, 固有ベクトル	固有値のみ
Cube		9602	681s	277s
Pipe		9000	569s	234s
OctaFlower		7919	388s	169s

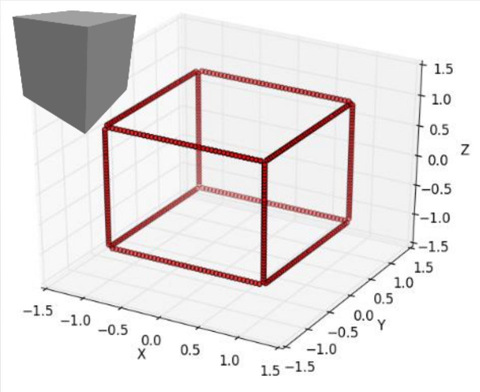
シャープな特徴のみを検出

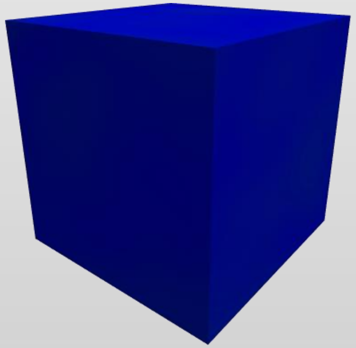


前回からの差分

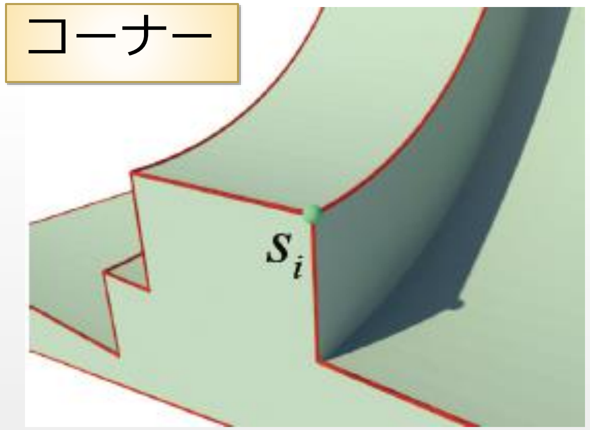
- GCVの速度改善
- L1解析の精度改善
- シャープさを考慮した平滑化の実験
- 3次元メッシュでのL1特徴解析の実験
- メッシュ特徴に関する議論
- 他の特徴を使った特徴解析の試作



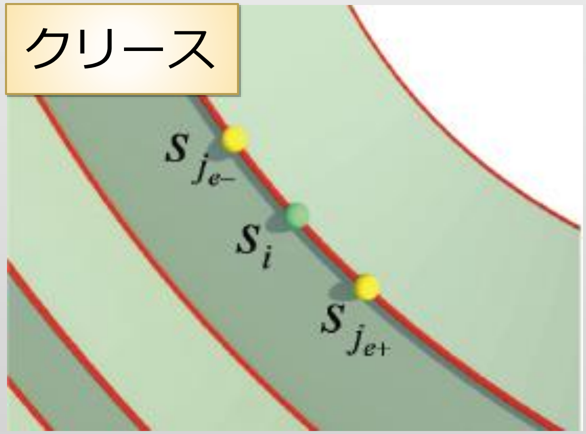




シャープさを考慮した平滑化

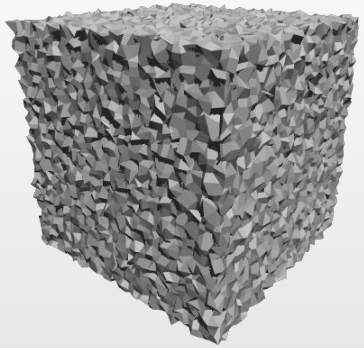


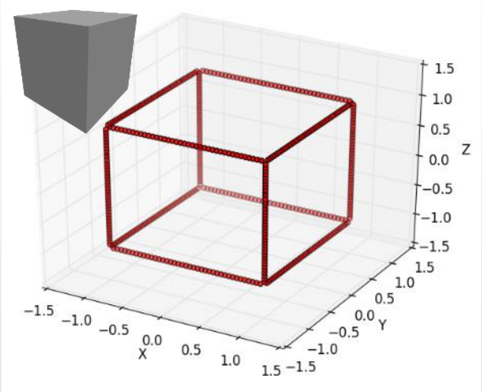
$$\mathbf{L}(s_i) = L_i(S) = 0$$

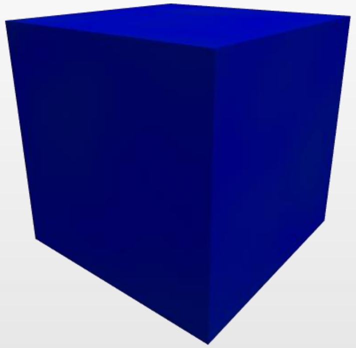


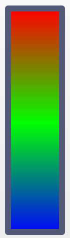
$$\mathbf{L}(s_i) = L_i(S) = s_{je-} + s_{je+} - 2s_i$$

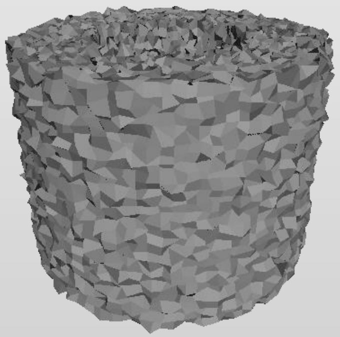
- シャープな特徴が事前に分かっている場合
  - 良好な平滑化結果が得られた

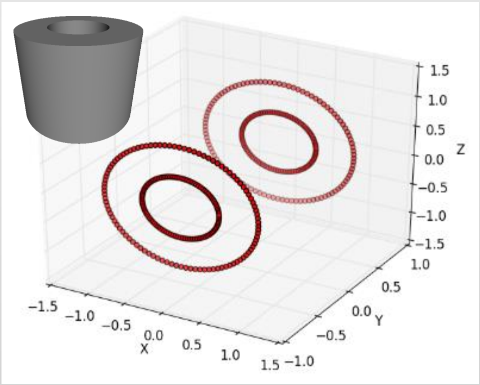


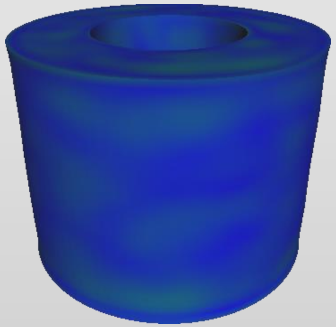




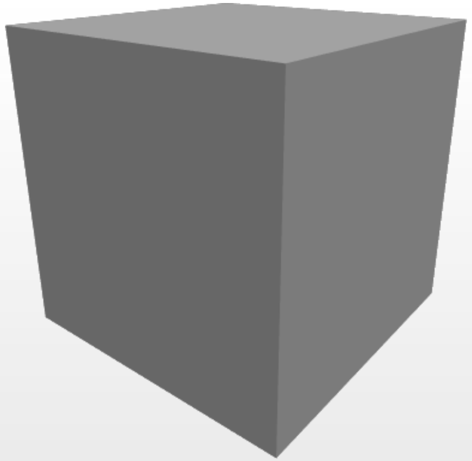




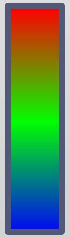


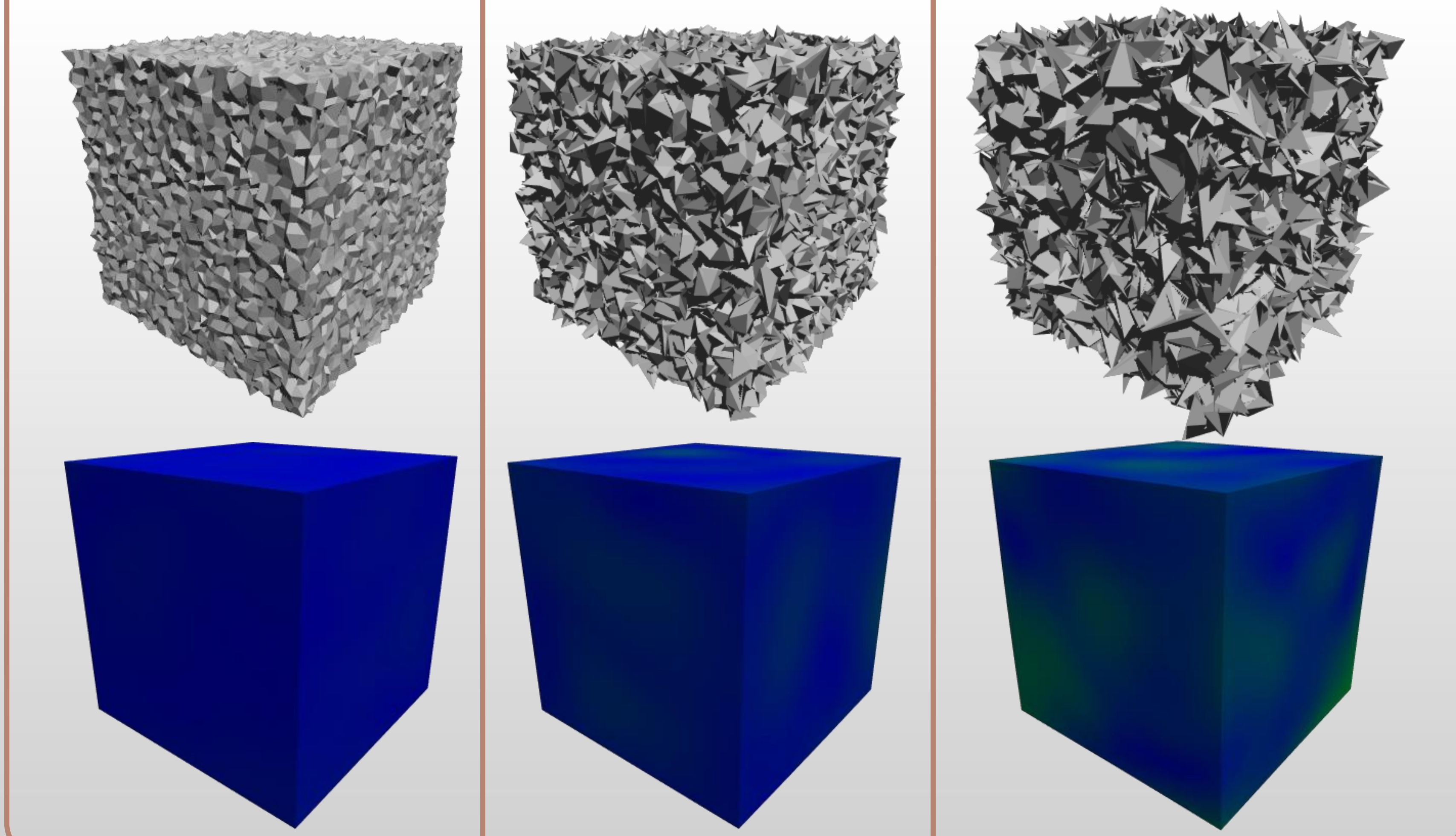


シャープさを考慮した平滑化

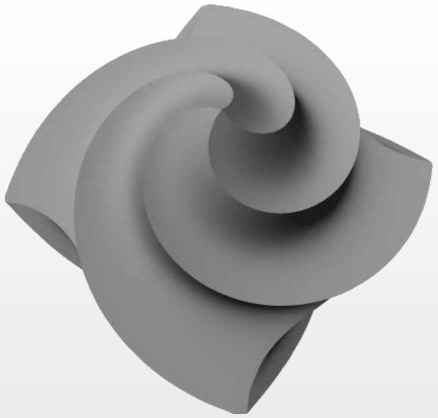


頂点数: 9602

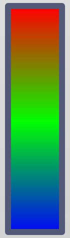




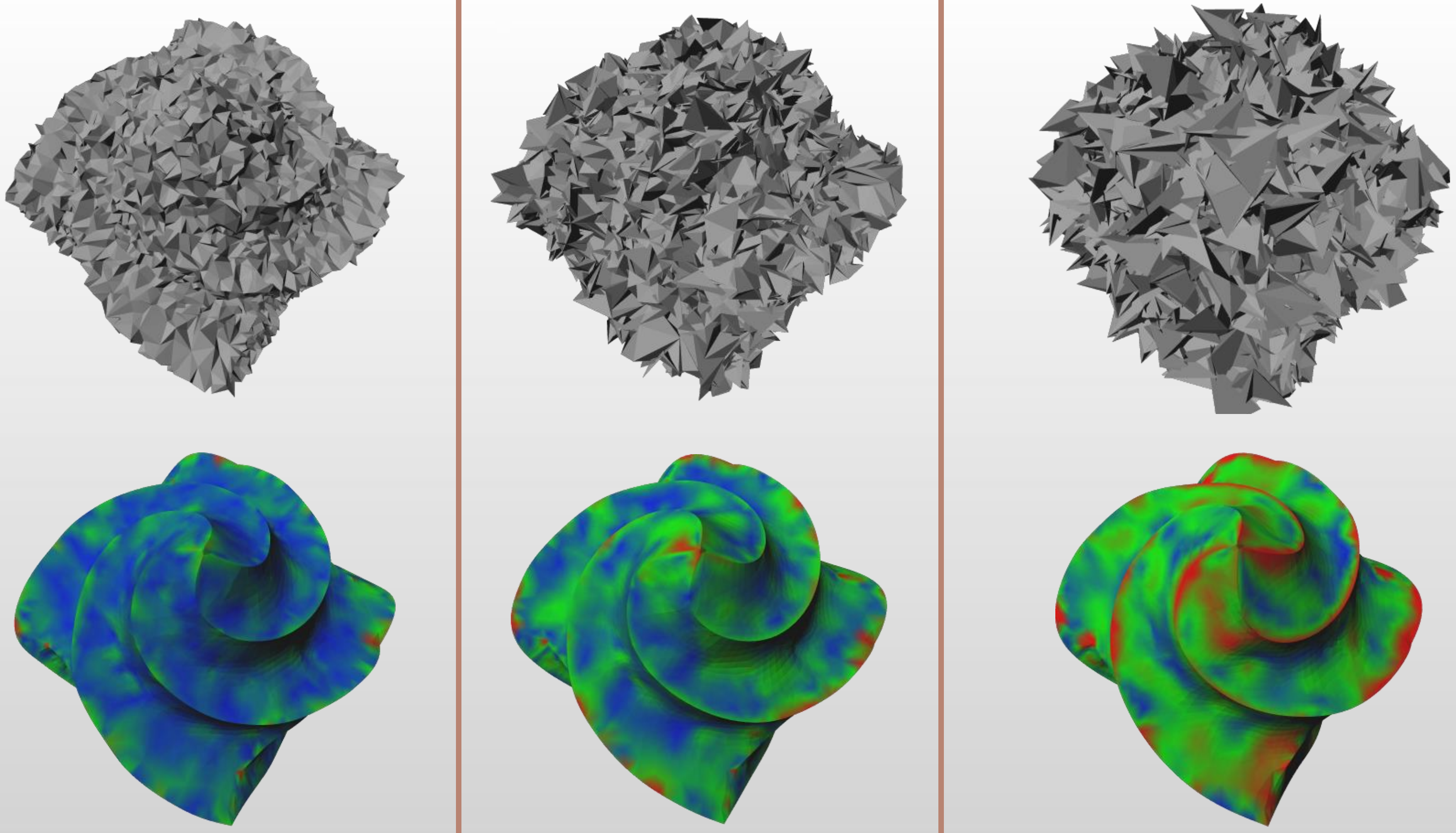
シャープさを考慮した平滑化



頂点数: 7919

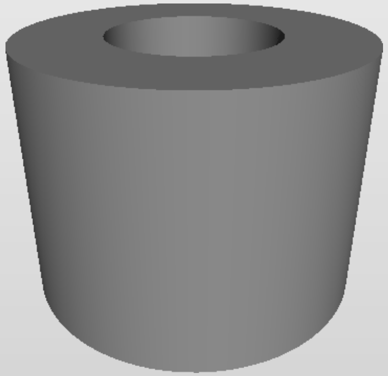


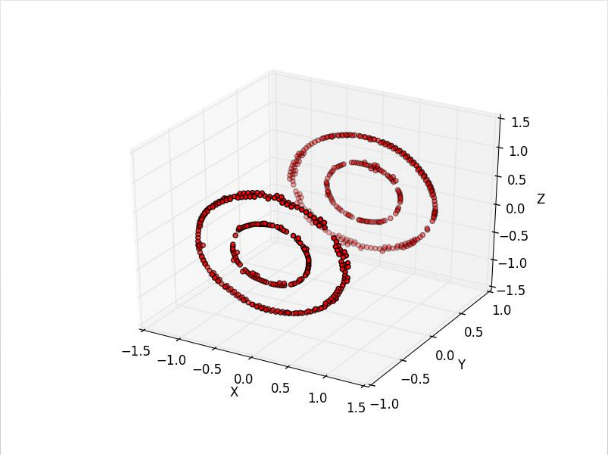
$\sigma = 0.01$ | $\sigma = 0.025$ | $\sigma = 0.05$



前回からの差分

- GCVの速度改善
- L1解析の精度改善
- シャープさを考慮した平滑化の実験
- 3次元メッシュでのL1特徴解析の実験
- メッシュ特徴に関する議論
- 他の特徴を使った特徴解析の試作

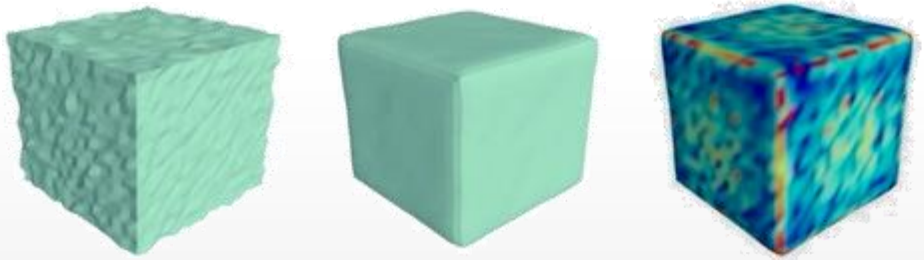




3次元メッシュでのL1特徴解析

- 法線方向の残差を計算

$$b_i = (\mathbf{p}_i - \hat{\mathbf{s}}_i)^T \hat{\mathbf{n}}_i ,$$

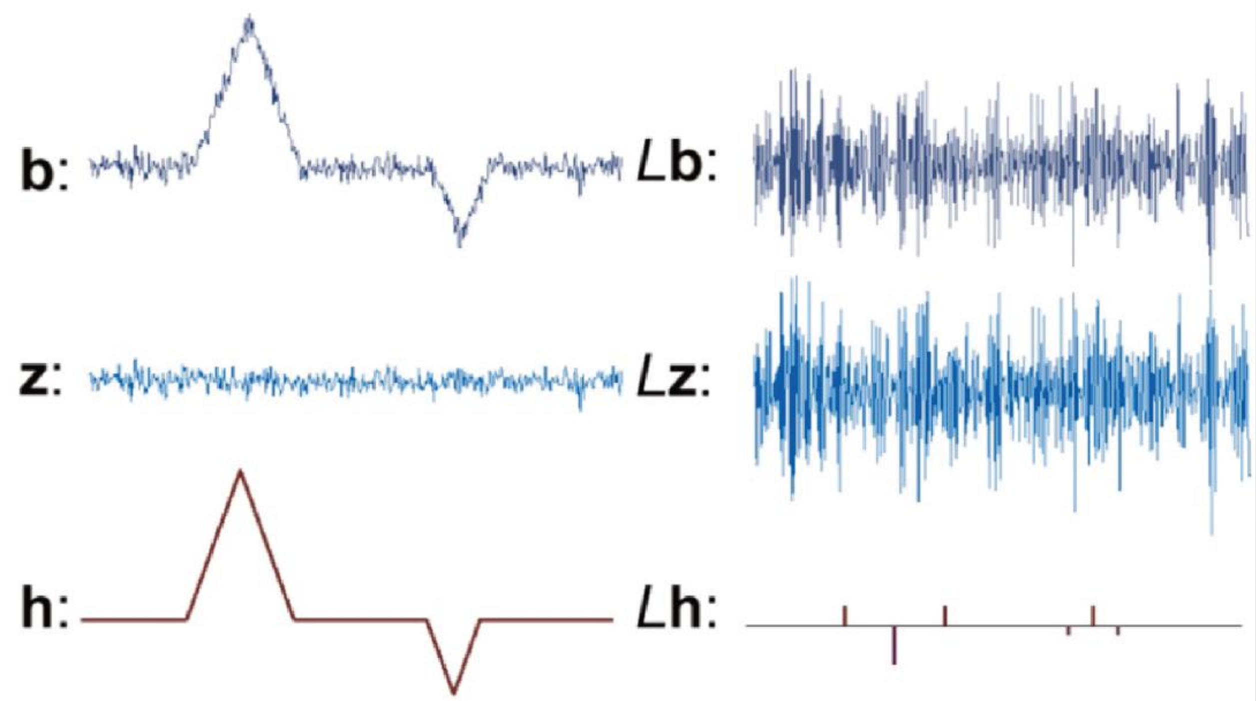


- Laplacian成分のL1解析でシャープな特徴を復元

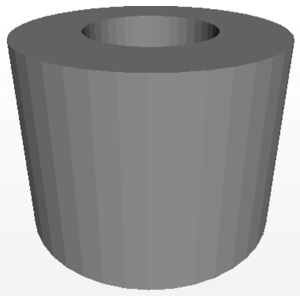
$$\min_{\mathbf{h}} \quad \|\mathbf{h} - \mathbf{b}\|_2^2$$

$$\text{s.t.} \quad \|W(L\mathbf{h})\|_1 = \sum_{i=1}^{n} w_i |L_i \mathbf{h}| \leq \tau$$

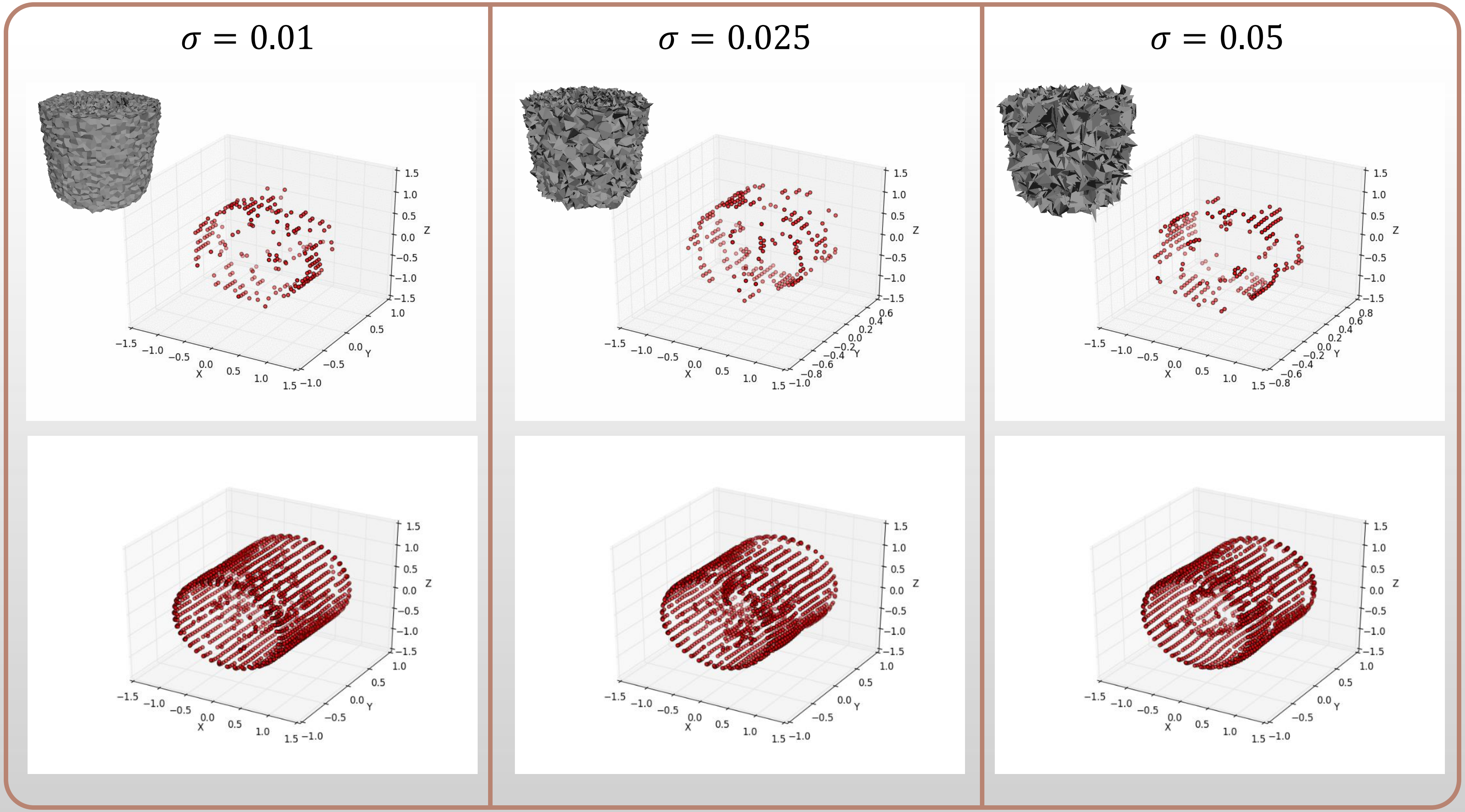
$$w_i = \frac{1}{\rho + \|L_i \hat{S}\|}$$



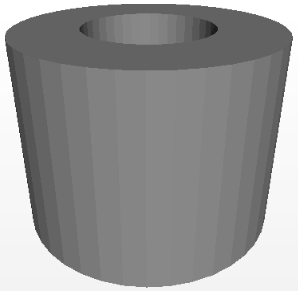
L1解析によるシャープな特徴の抽出



頂点数: 3600

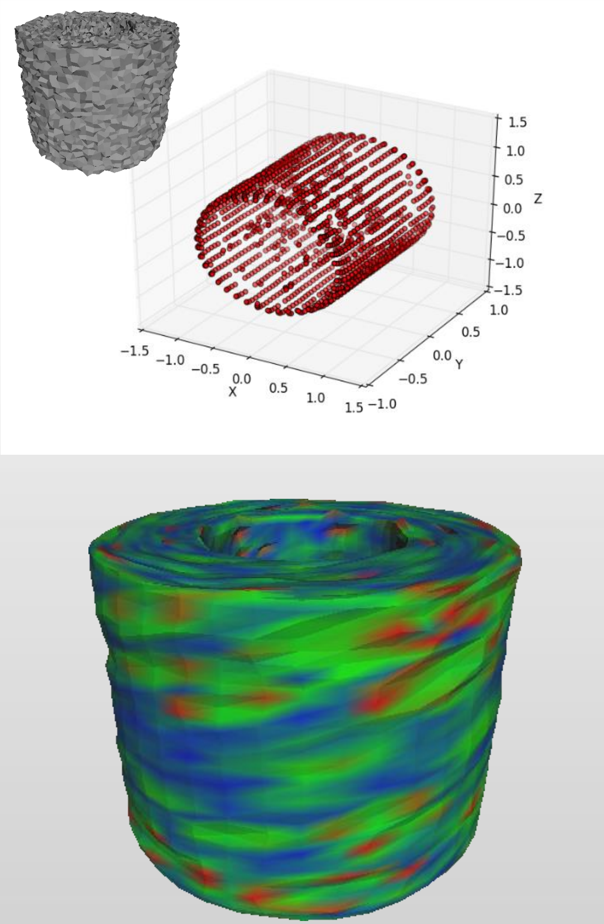


L1解析によるシャープな特徴の抽出

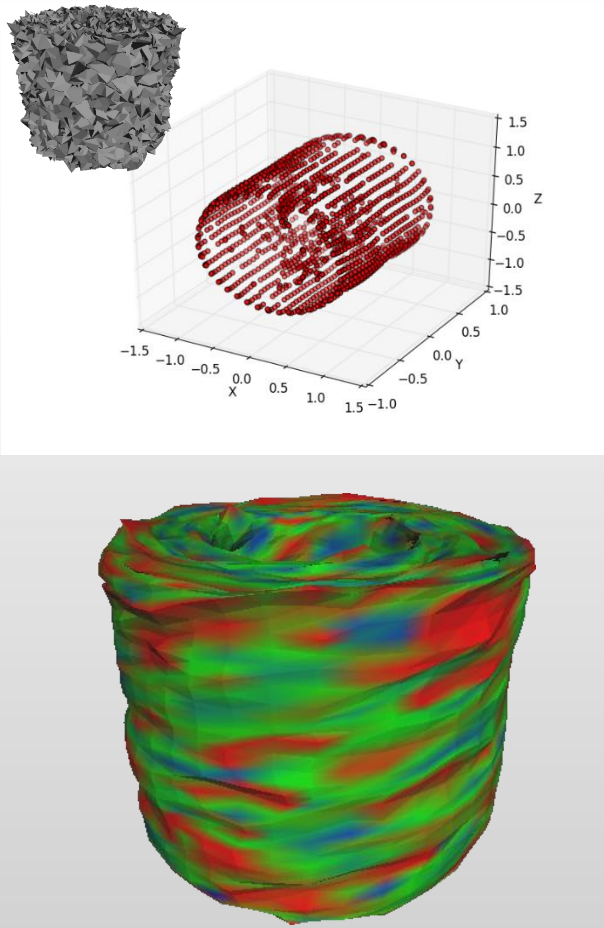


頂点数: 3600

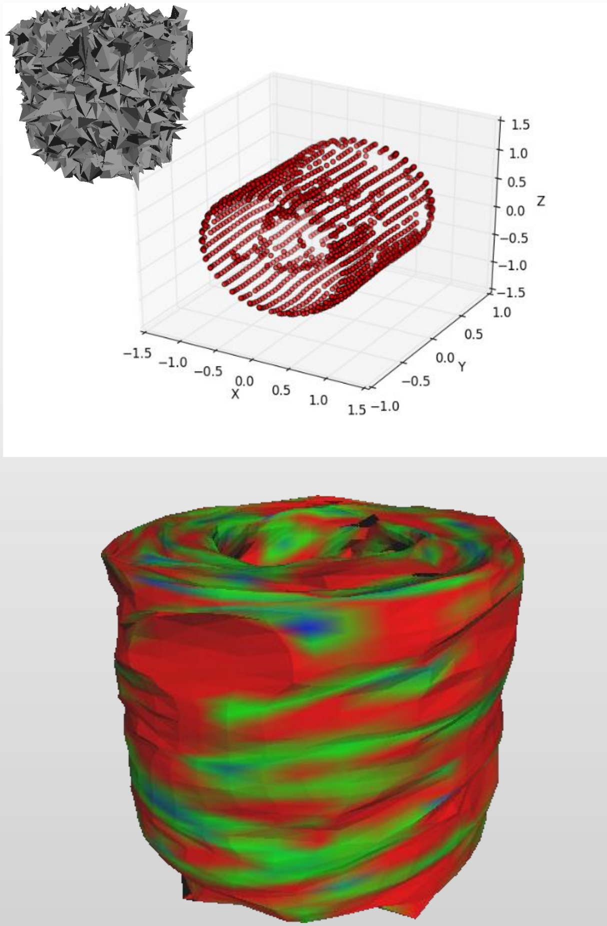
$\sigma = 0.01$



$\sigma = 0.025$



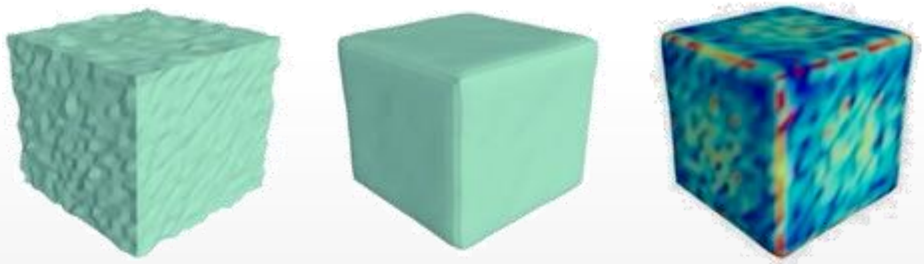
$\sigma = 0.05$



３次元メッシュでのL1特徴解析

- 法線方向の残差を計算

$$b_i = (\mathbf{p}_i - \hat{\mathbf{s}}_i)^T \hat{\mathbf{n}}_i ,$$



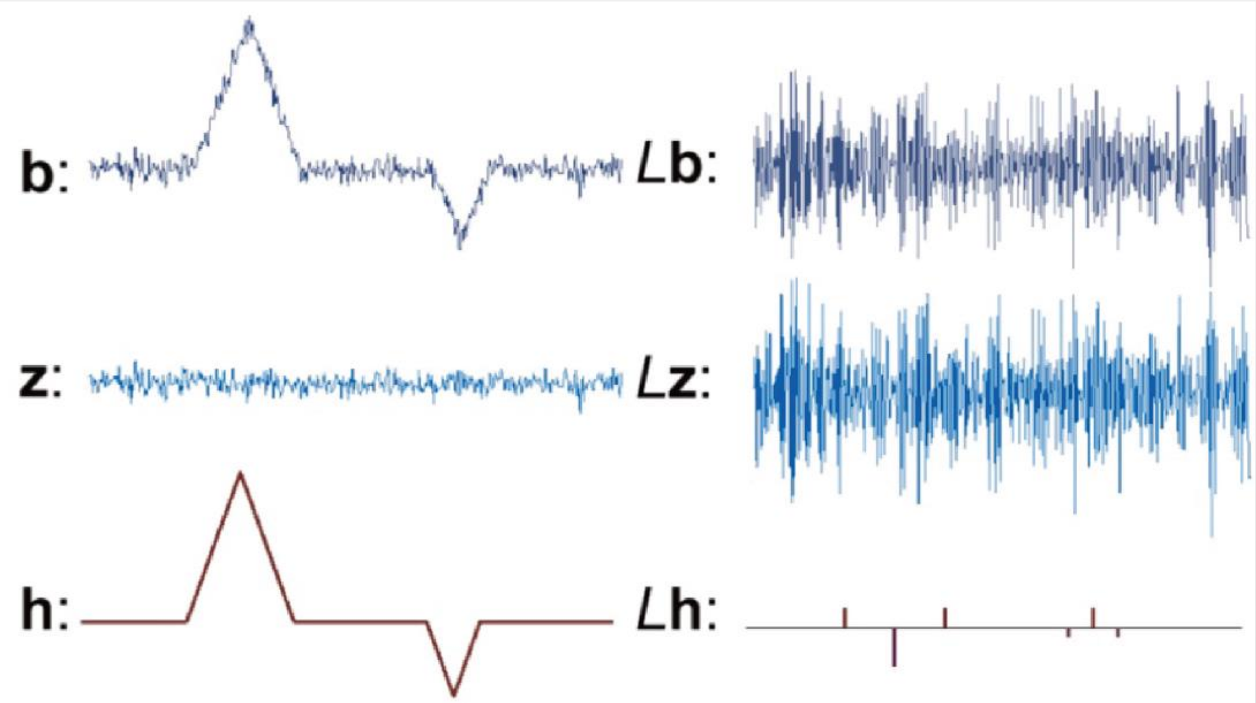
- Laplacian成分のL1解析でシャープな特徴を復元

$$\min_{\mathbf{h}} \quad \|\mathbf{h} - \mathbf{b}\|_2^2$$

$$\text{s.t.} \quad \|W(L\mathbf{h})\|_1 = \sum_{i=1}^{n} w_i |L_i \mathbf{h}| \leq \tau$$

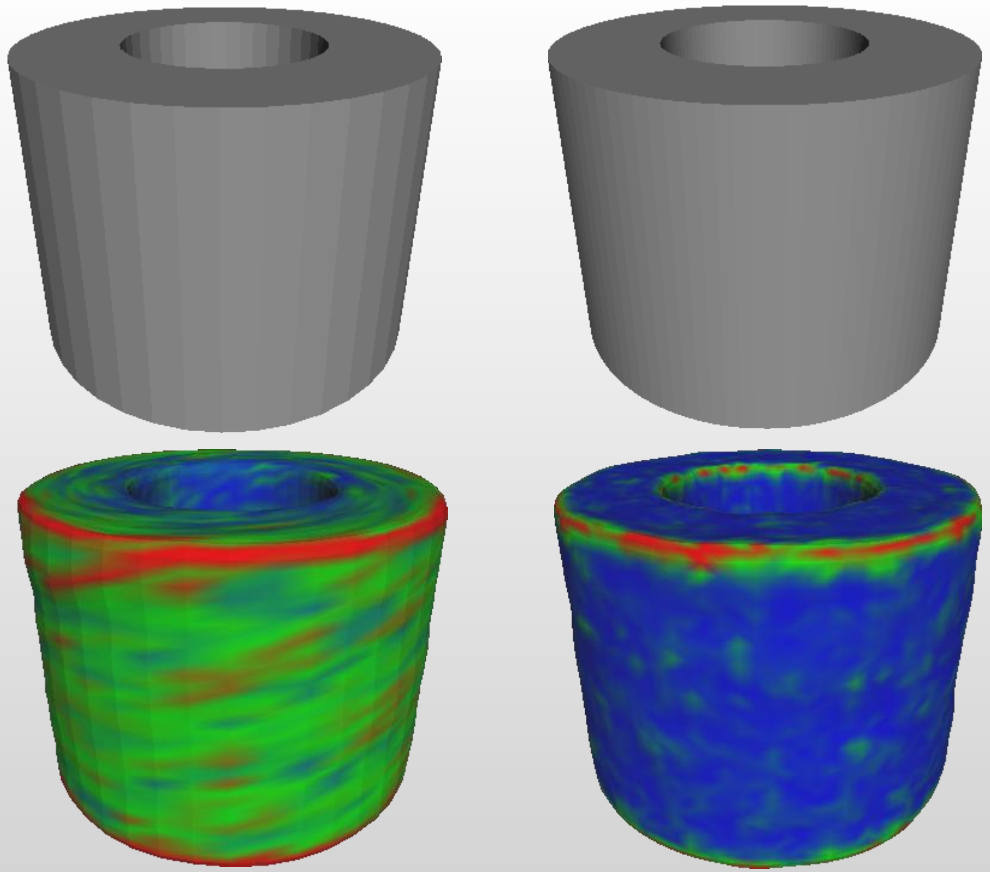
$$w_i = \frac{1}{\rho + \|L_i \hat{S}\|}$$

Laplacian成分が重要



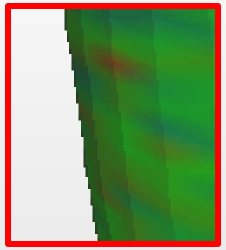
前回からの差分

- GCVの速度改善
- L1解析の精度改善
- シャープさを考慮した平滑化の実験
- 3次元メッシュでのL1特徴解析の実験
- メッシュ特徴に関する議論
- 他の特徴を使った特徴解析の試作

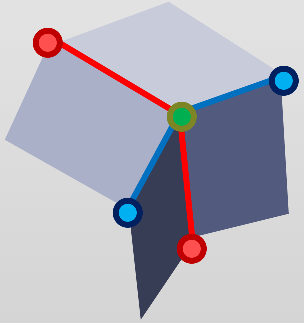


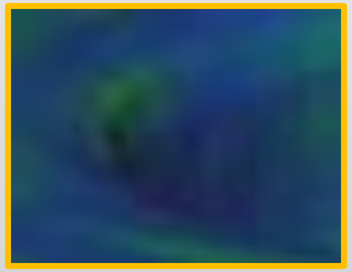
メッシュ解像度の重要性

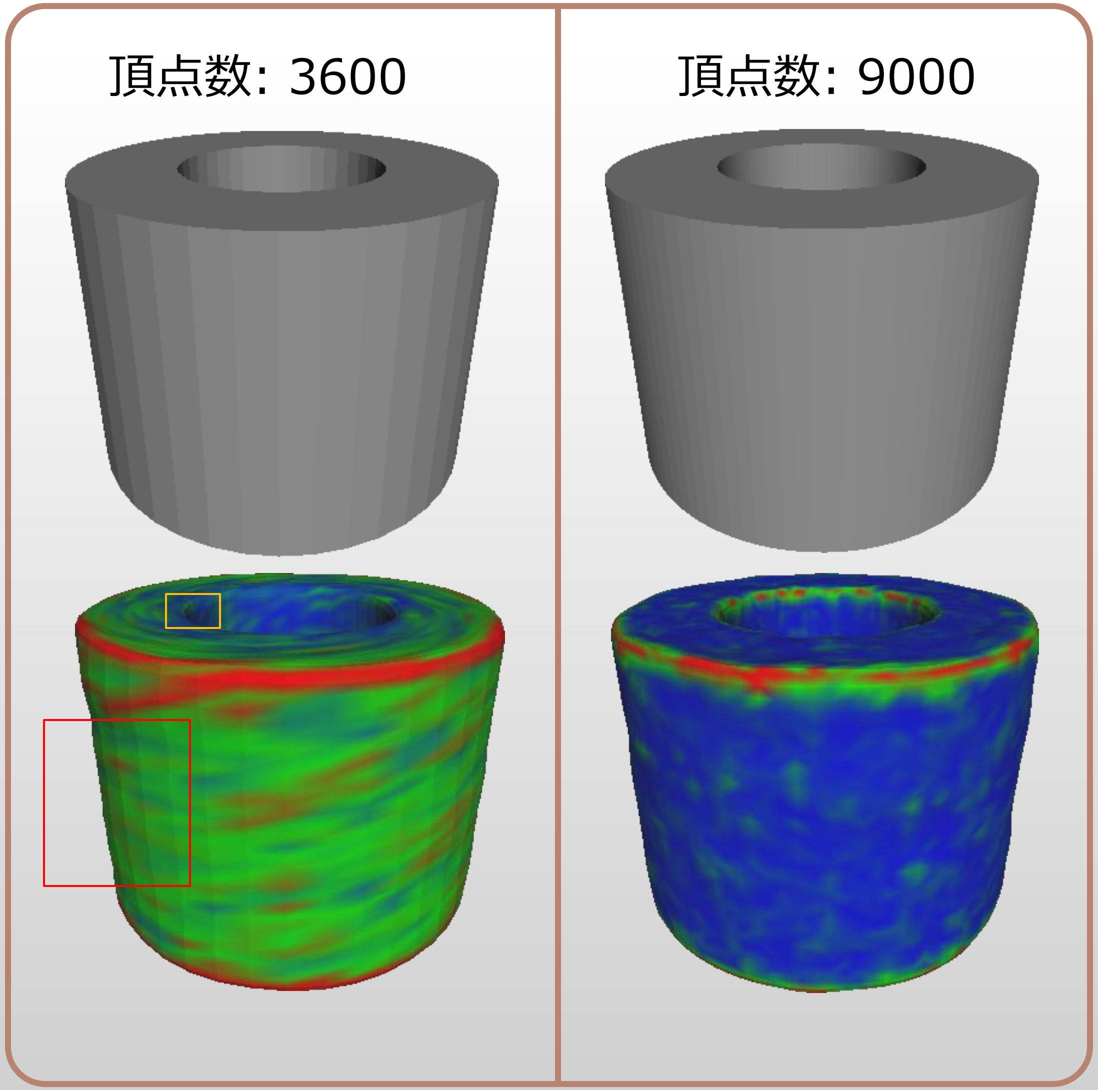
- 解像度が小さいと…
  - シャープさをLaplacianで検出できない
    - 曲面部の値が高くなる



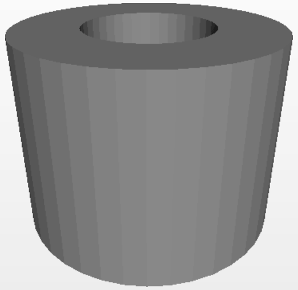
    - 較状のLaplacian成分が0に…



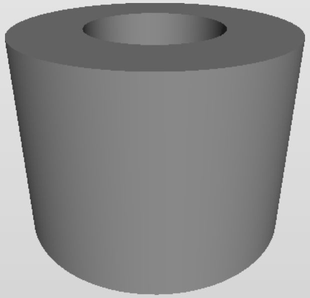




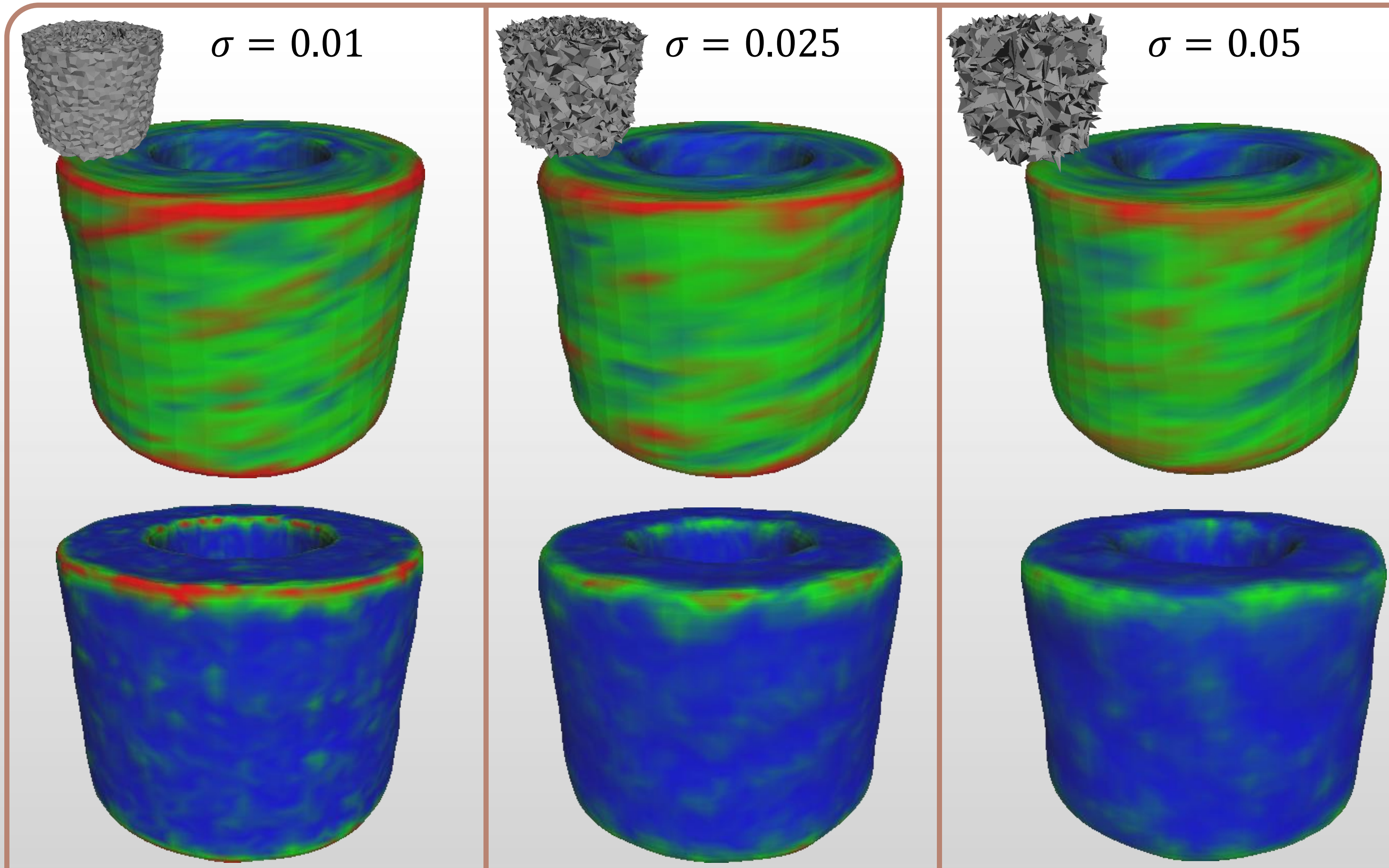
メッシュ解像度の重要性



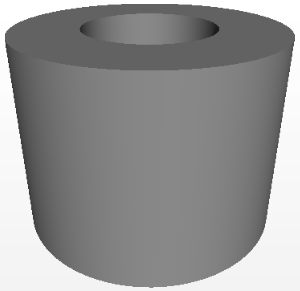
頂点数: 3600



頂点数: 9000

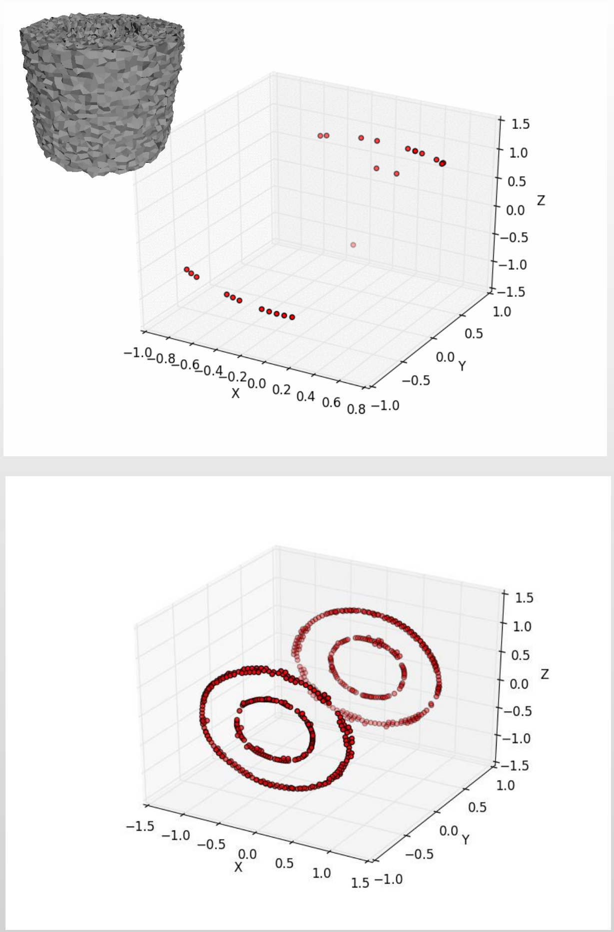


L1解析によるシャープな特徴の抽出

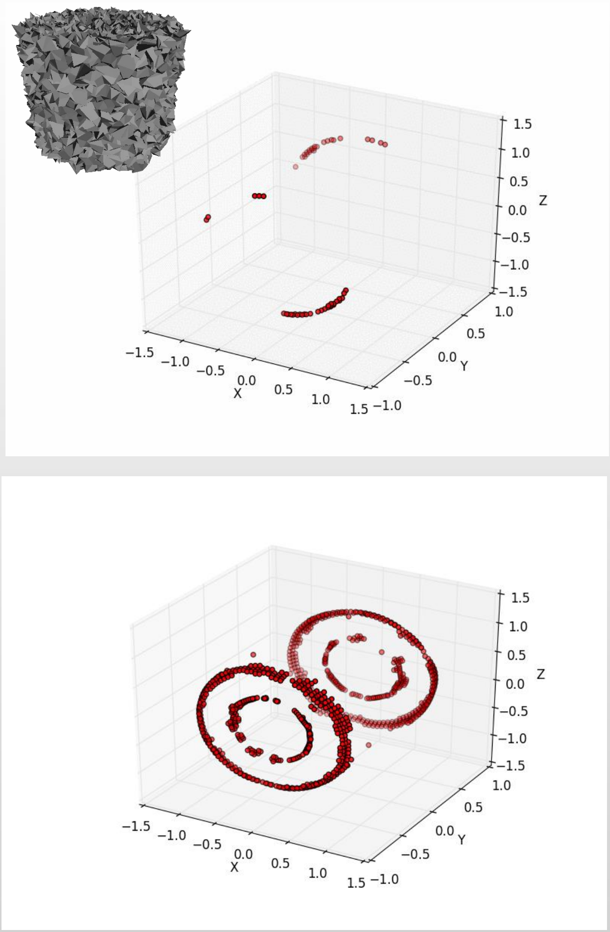


頂点数: 9000

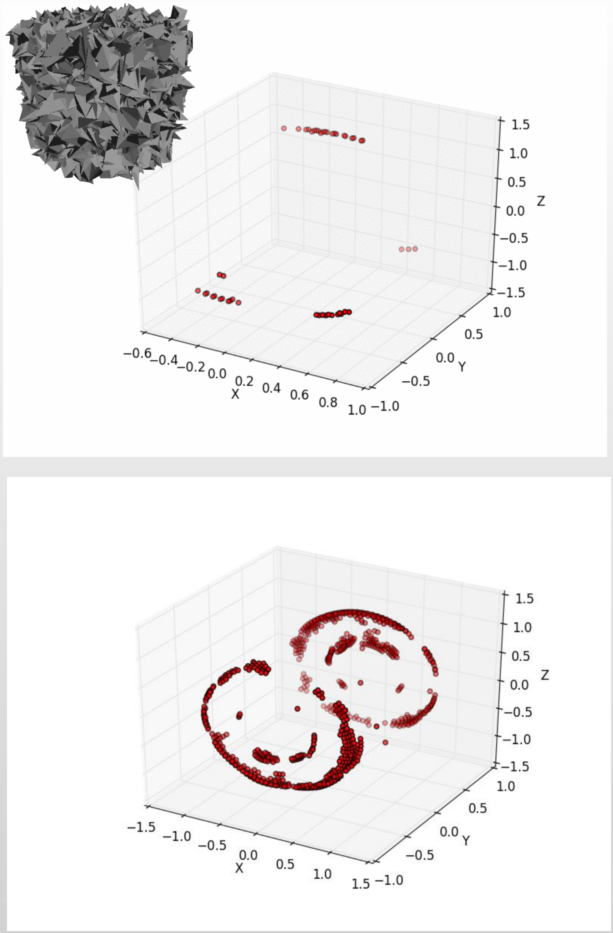
$\sigma = 0.01$



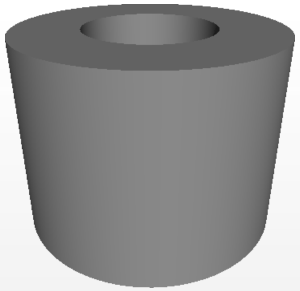
$\sigma = 0.025$



$\sigma = 0.05$

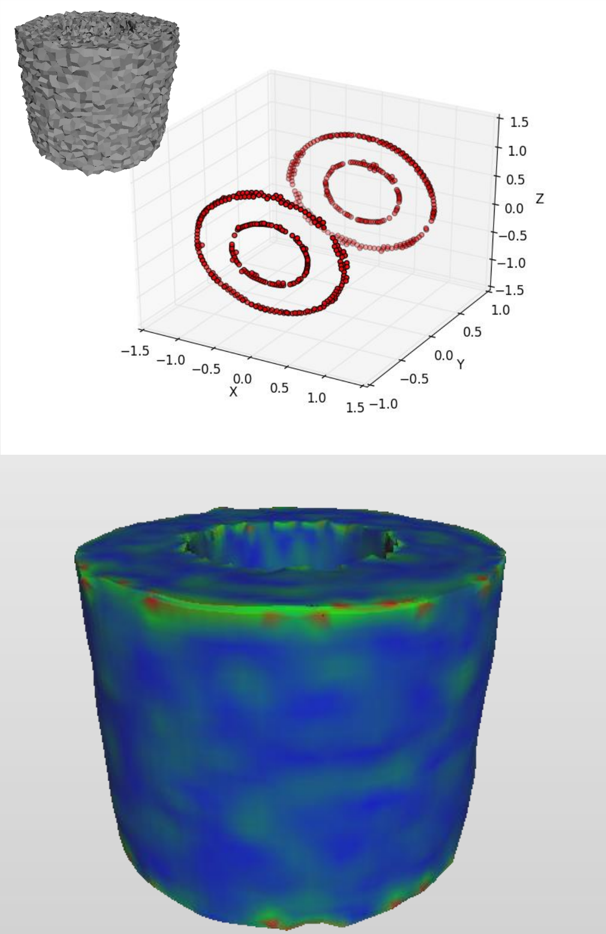


L1解析によるシャープな特徴の抽出

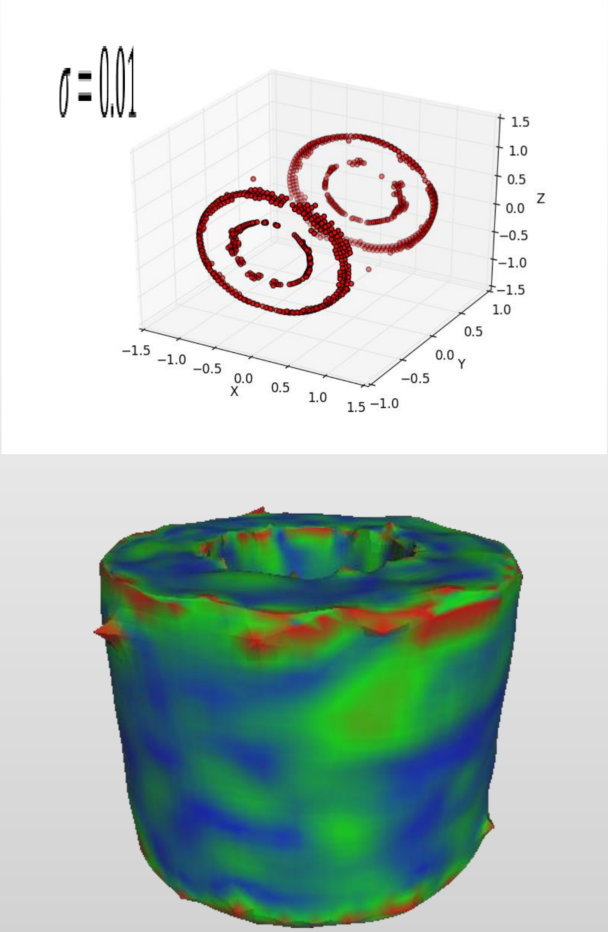


頂点数: 9000

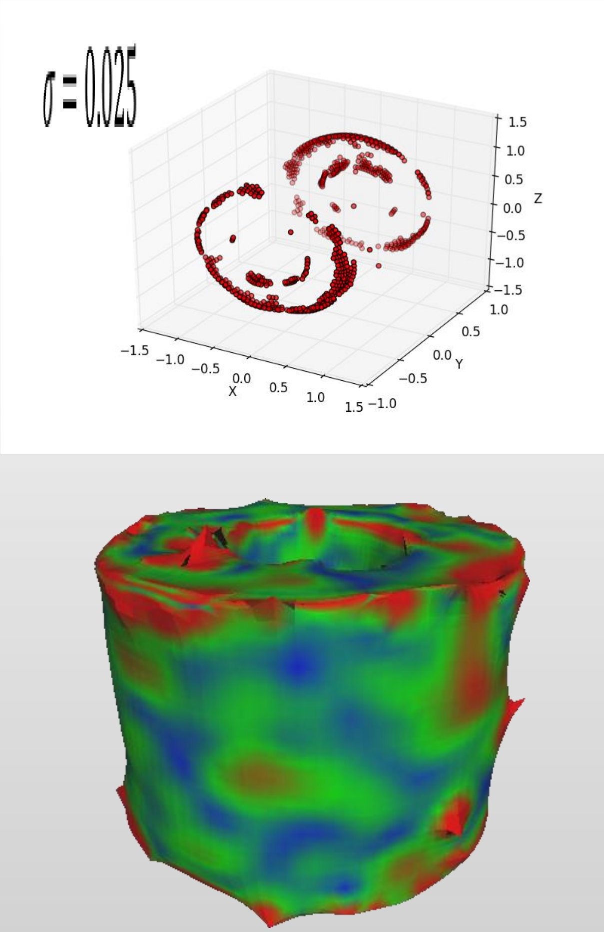
$\sigma = 0.01$



$\sigma = 0.025$



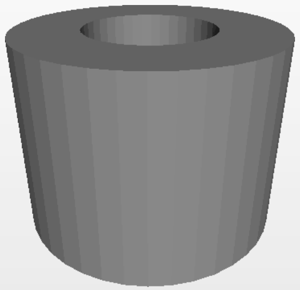
$\sigma = 0.05$

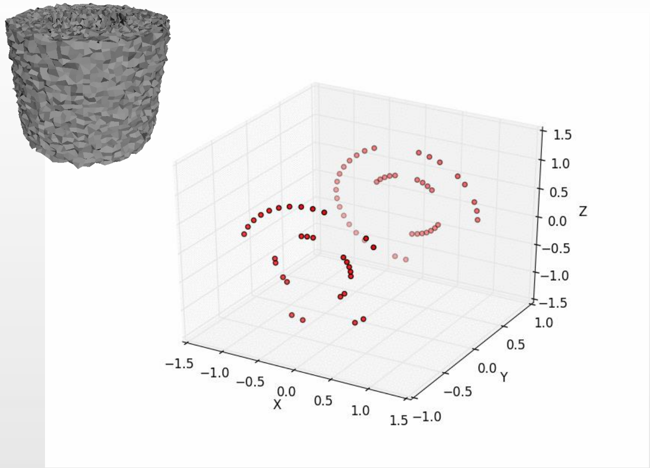


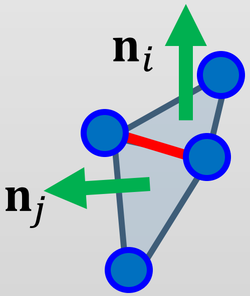
前回からの差分

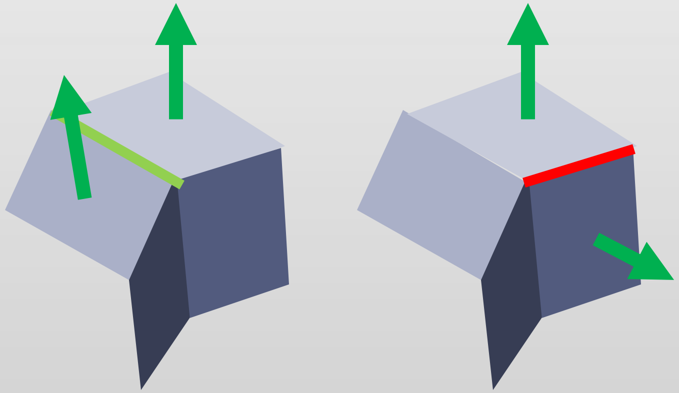
- GCVの速度改善
- L1解析の精度改善
- シャープさを考慮した平滑化の実験
- 3次元メッシュでのL1特徴解析の実験
- メッシュ特徴に関する議論
- 他の特徴を使った特徴解析の試作

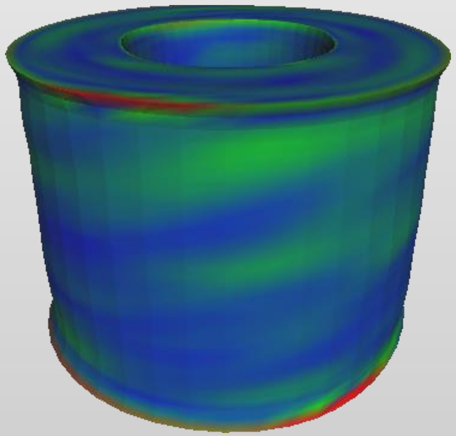
$\sigma = 0.01$





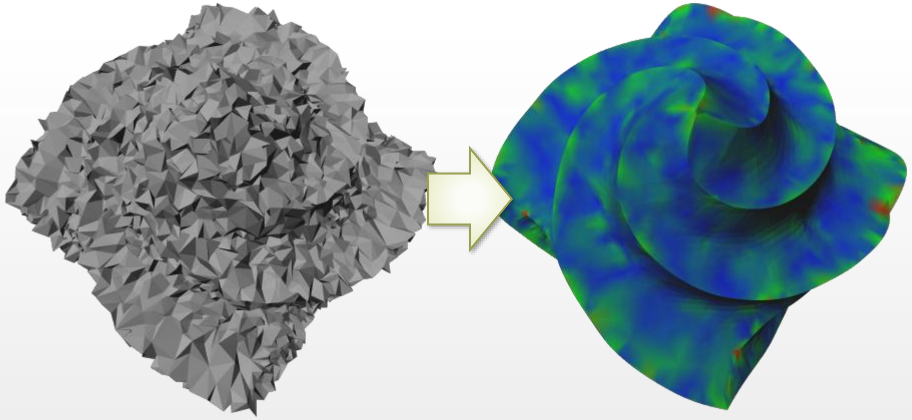






まとめ

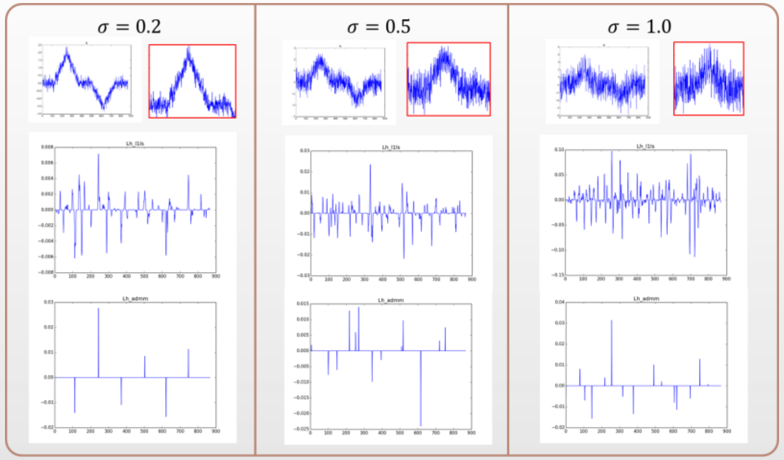
- 上手くいった点
  - GCVによる最適なメッシュ平滑化
  - シャープな特徴を考慮したLaplacian行列の修正
    ⇒別のノイズ付データにも使えそう

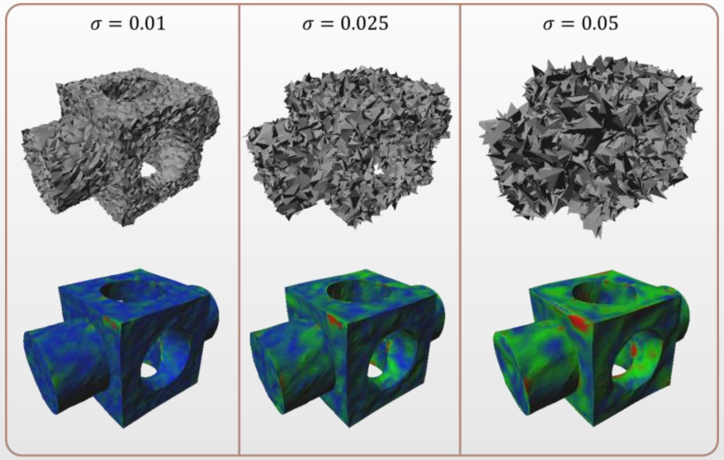


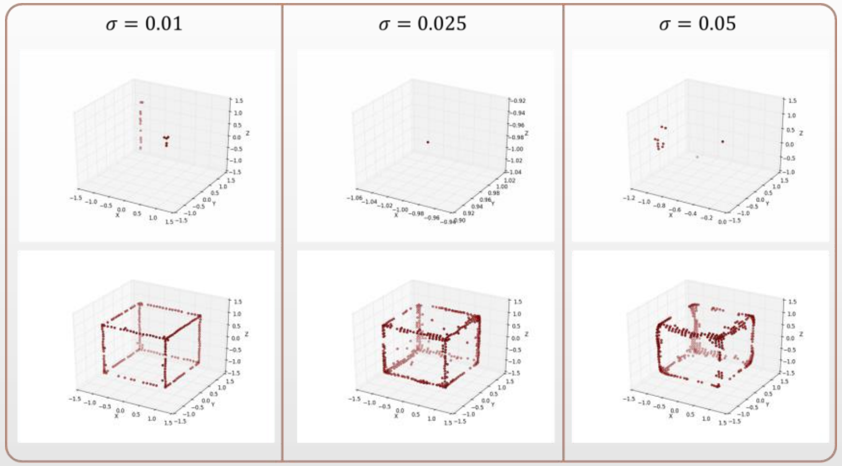
- 反省点
  - L1の解析精度は再現できなかった
    ⇑そうこうしている内に L1解析を利用したSIGGRAPHネタが…

ポスターセッションでは

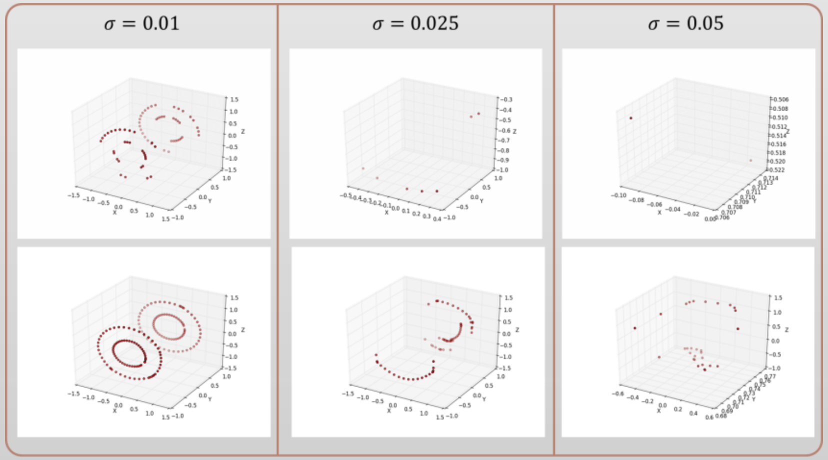
解析データ色々あります







モデル	頂点数	固有値, 固有ベクトル	固有値のみ	実行時
Eight	3070	25.0s	11.2s	7.5s
Cube	9602	681s	277s	45s
Block	12812	1660s	641s	76s
Pipe	9000	569s	234s	41s
OctaFlower	7919	388s	169s	33s



...

GCVの速度の改善

- 前回時の問題点
  - $\widehat{S_n}(\lambda)$の計算に固有ベクトルを使っていた

$$\mathrm{GCV}_n(\lambda) = \frac{\frac{1}{n}\|P - \hat{S}_n(\lambda)\|_F^2}{\left(1 - \frac{1}{n}\mathrm{tr}[A_n(\lambda)]\right)^2},$$

$$\hat{S}(\lambda) = A_n(\lambda)P$$

$$A_n(\lambda) = (I_n + \lambda M)^{-1}$$

逆行列の計算の負荷を
避けるため固有ベクトルで計算

$$\|P(I_n - A_n(\lambda))\|_F^2 = \sum_{j=1}^{n} \frac{\lambda^2\mu_j^2}{(1+\lambda\mu_j)^2}\mathbf{u}_j^T P$$

固有値

固有ベクトル

3070頂点のメッシュで25秒の計算

GCVの速度の改善

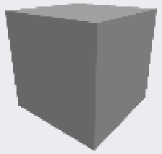
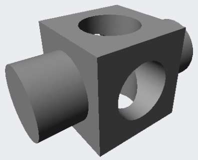
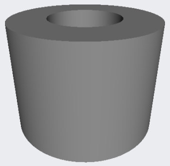
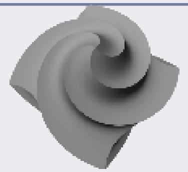
- $\widehat{S_n}(\lambda)$の計算方法の改善
  - 直接 $\hat{S}(\lambda) = (I_n + \lambda M)^{-1} P$ の線形方程式を解くだけ

scipy.sparse.linalg.spsolveを使用

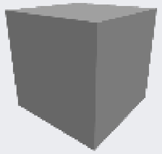
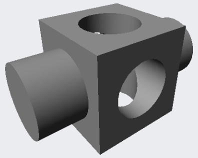
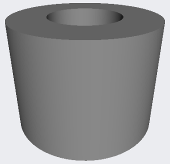
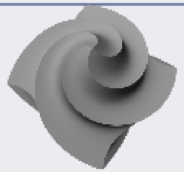
- 利点：
  - 事前計算の時間を1/3程度に短縮される
  - 解析データは固有値さえ保存しておけばよい
    - 固有値：頂点数12821で236 KB (頂点数分のデータ)
    - 固有ベクトル: 頂点数9000で1.4GB (頂点数×頂点数分のデータ)

- 欠点：
  - 各計算ステップは固有ベクトルの式よりも若干遅くなる

GCV付平滑化の計算時間

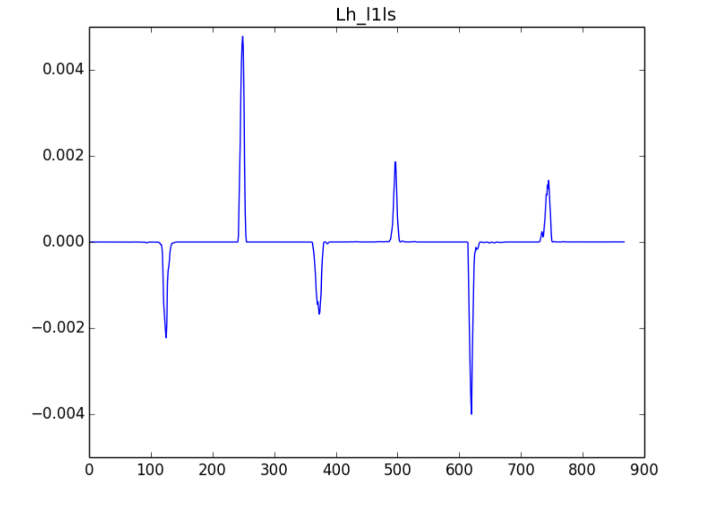
モデル		頂点数	固有値, 固有ベクトル	固有値のみ	実行時
Eight		3070	25.0s	11.2s	7.5s
Cube		9602	681s	277s	45s
Block		12812	1660s	641s	76s
Pipe		9000	569s	234s	41s
OctaFlower		7919	388s	169s	33s

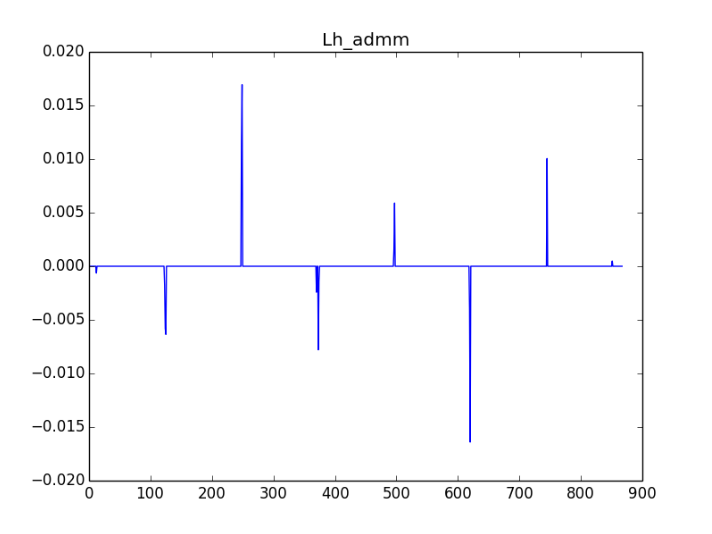
解析に必要なファイルのデータサイズ

モデル		頂点数	メッシュ	固有値	Laplacian行列
Eight		3070	356KB	78KB	255KB
Cube		9602	986KB	243KB	589KB
Block		12812	1.29MB	326KB	808KB
Pipe		9000	413KB	229KB	522KB
OctaFlower		7919	915KB	202KB	681KB

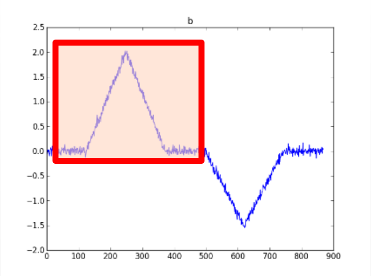
L1解析の改善

- 前回の解析の手法
  - $\min_{\mathbf{h}} \frac{1}{2}\|\mathbf{h}-\mathbf{b}\|^2 + \tau\|W(L\mathbf{h})\|_1$
  - $\min_{\mathbf{h}} \frac{1}{2}\|\mathbf{h}-\mathbf{b}\|^2 + \frac{\alpha}{2}\|W(L\mathbf{h})-\mathbf{y}\|^2 + \tau\|\mathbf{y}\|_1$
- ADMMによる定式化
  - $\min_{\mathbf{h}} \frac{1}{2}\|\mathbf{h}-\mathbf{b}\|^2 + \tau\|\mathbf{z}\|_1$, subject to $W(L\mathbf{h}) = \mathbf{z}$
  - $\min_{\mathbf{h}} \frac{1}{2}\|\mathbf{h}-\mathbf{b}\|^2 + \rho\mathbf{a}^T(W(L\mathbf{h})-\mathbf{z}) + \frac{\rho}{2}\|W(L\mathbf{h})-\mathbf{z}\|^2 + \tau\|\mathbf{z}\|_1$
  - $\min_{\mathbf{h}} \frac{1}{2}\|\mathbf{h}-\mathbf{b}\|^2 + \frac{\rho}{2}\|W(L\mathbf{h})-\mathbf{z}+\mathbf{a}\|^2 - \frac{\rho}{2}\|\mathbf{a}\|^2 + \tau\|\mathbf{z}\|_1$



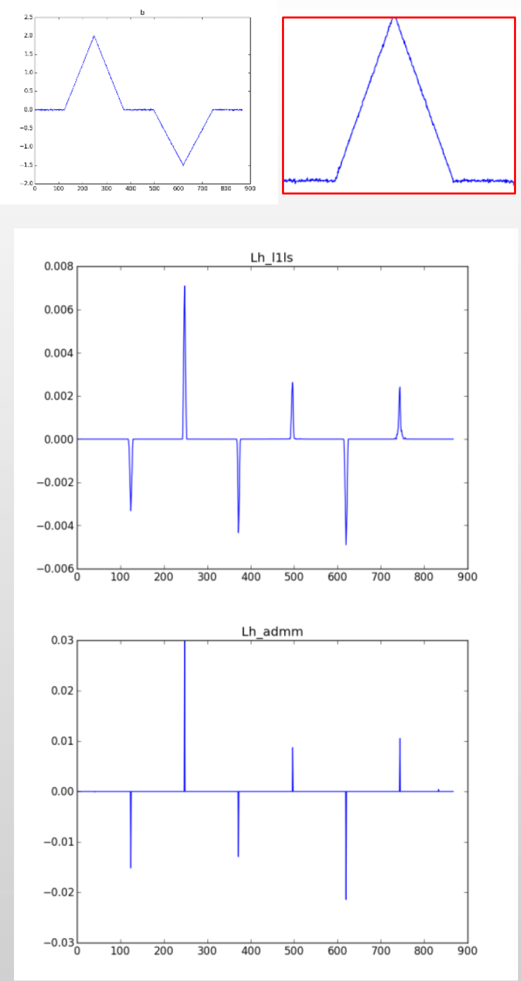


L1解析の改善

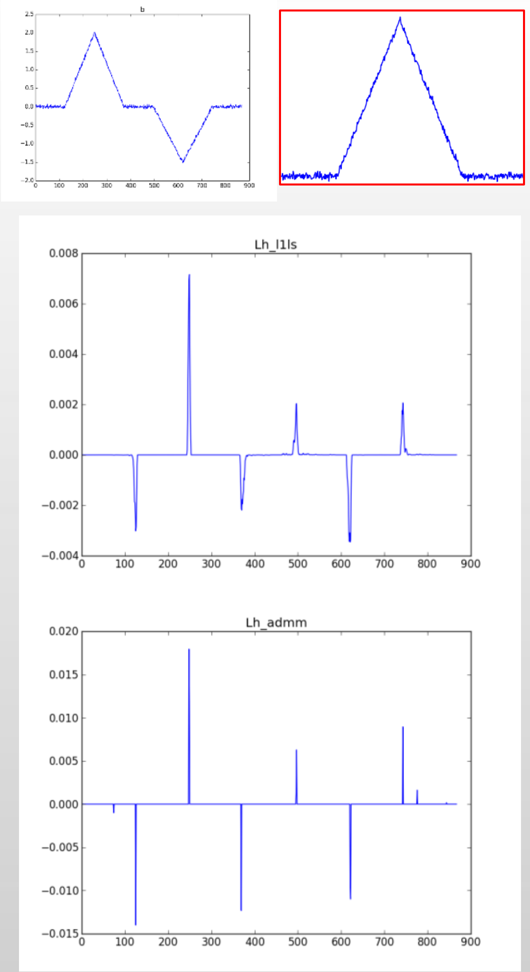


Input data

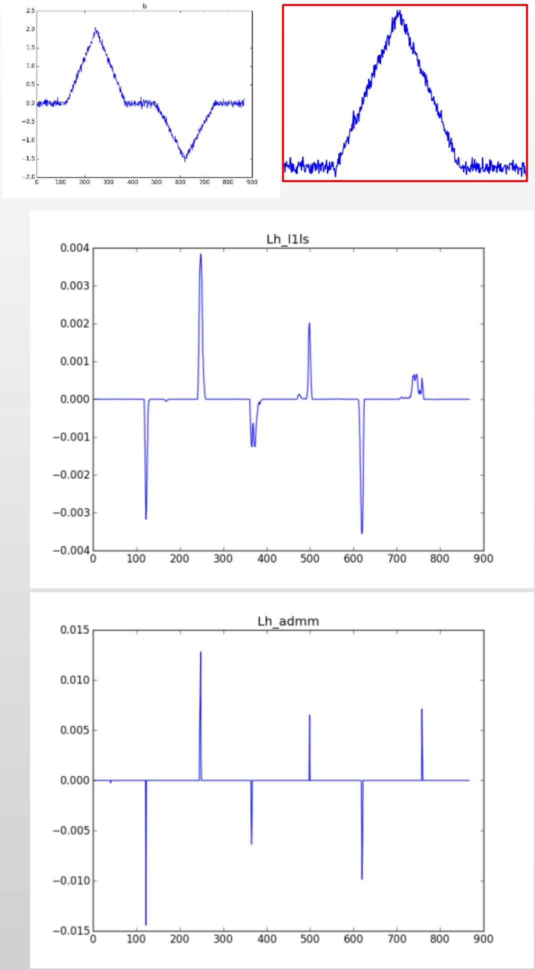
$\sigma = 0.01$



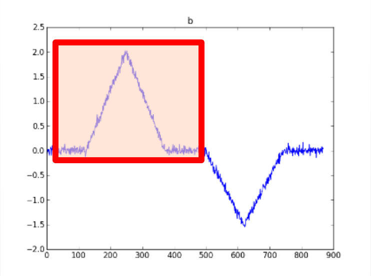
$\sigma = 0.025$



$\sigma = 0.05$

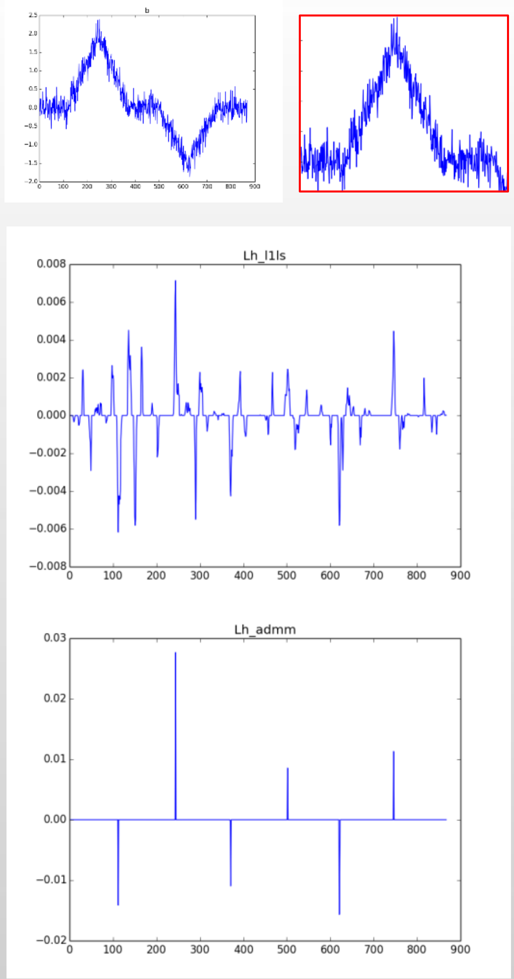


L1解析の改善

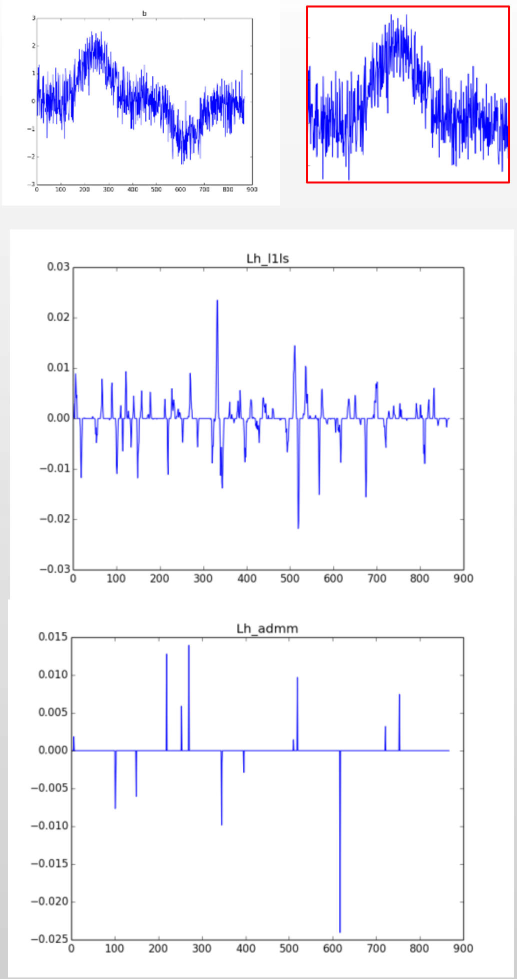


Input data

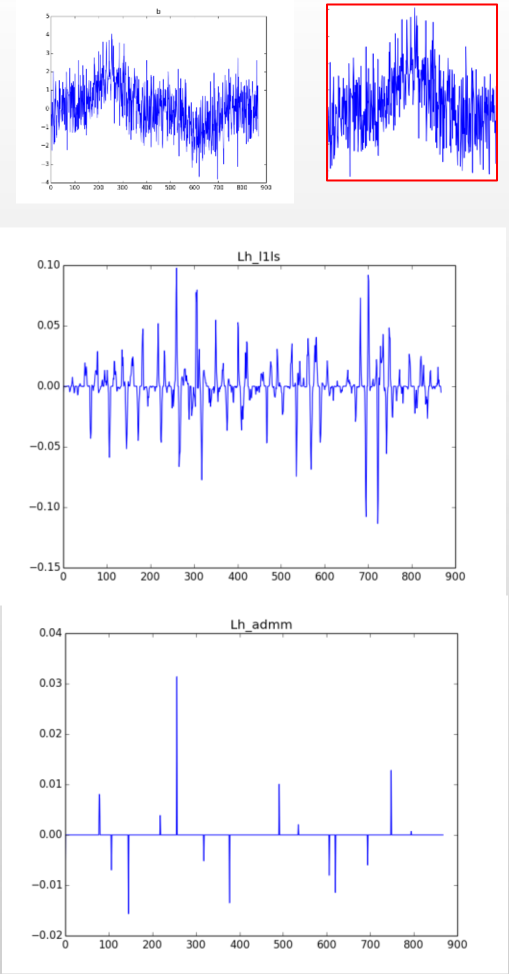
$\sigma = 0.2$



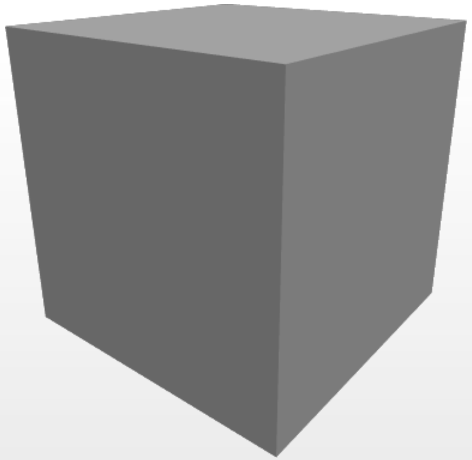
$\sigma = 0.5$



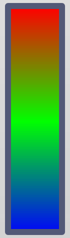
$\sigma = 1.0$

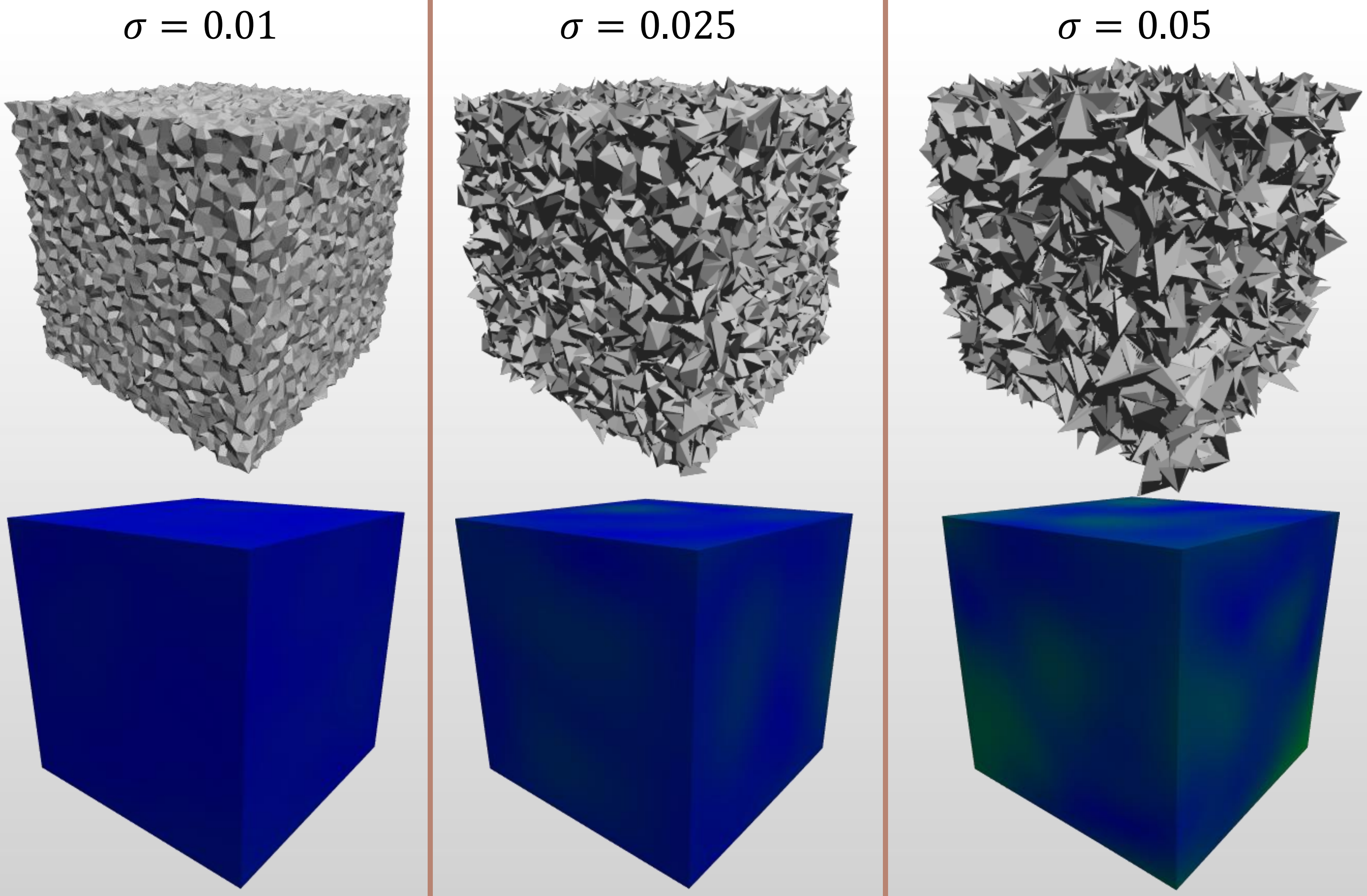


シャープさを考慮した平滑化

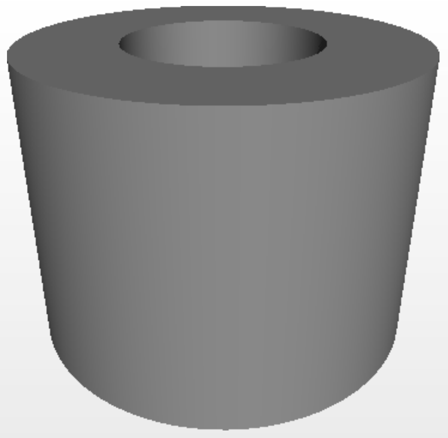


頂点数: 9602

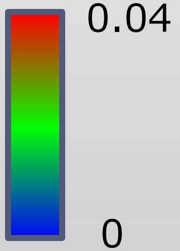


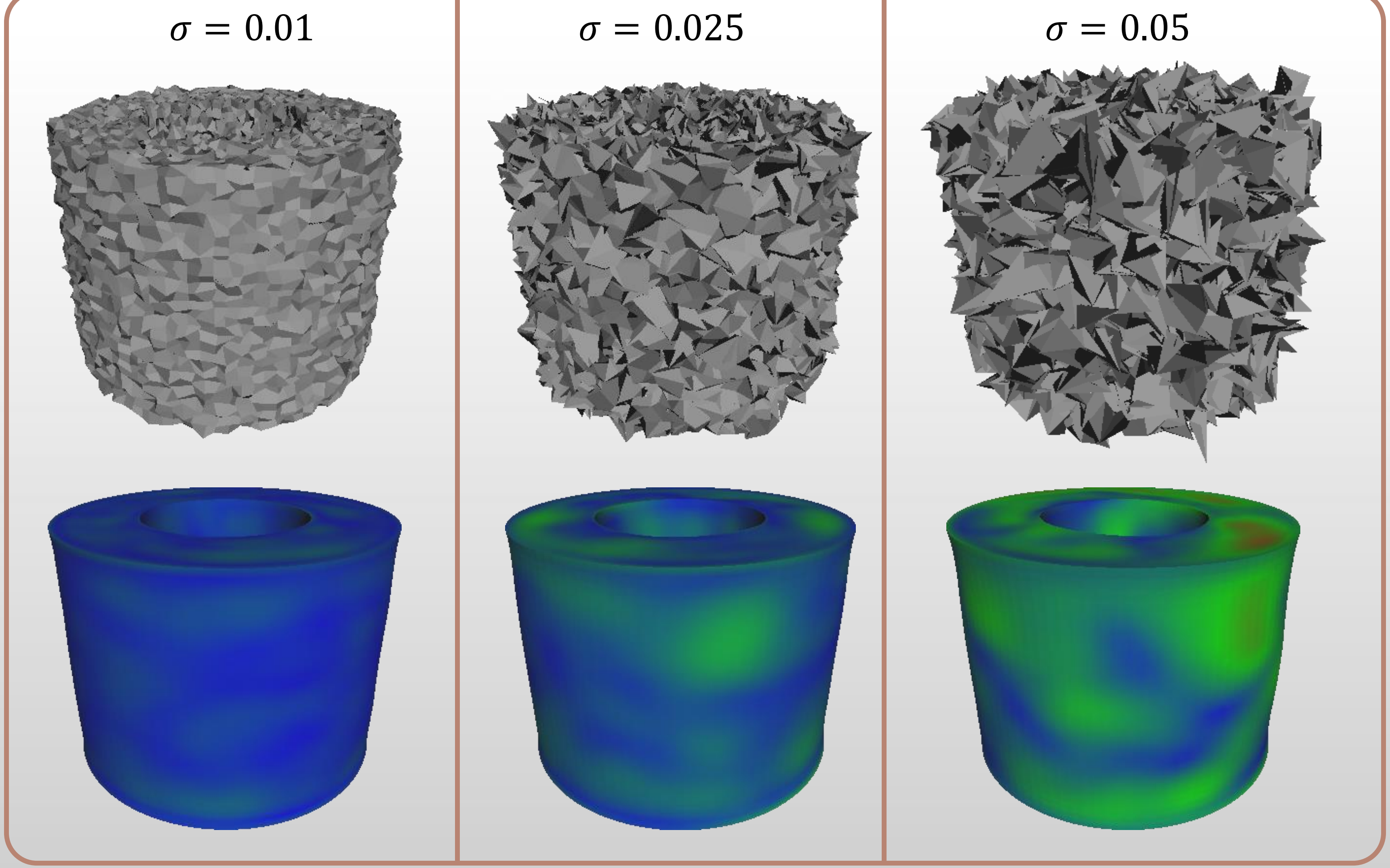


シャープさを考慮した平滑化

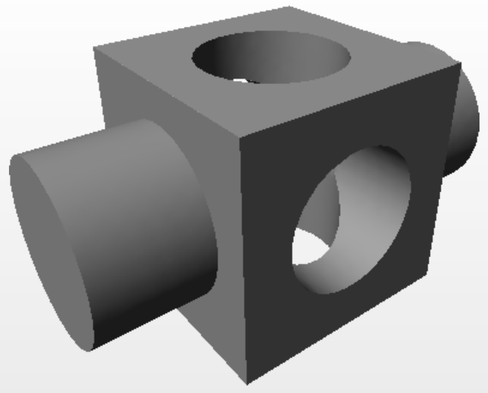


頂点数: 9000

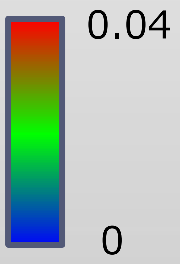


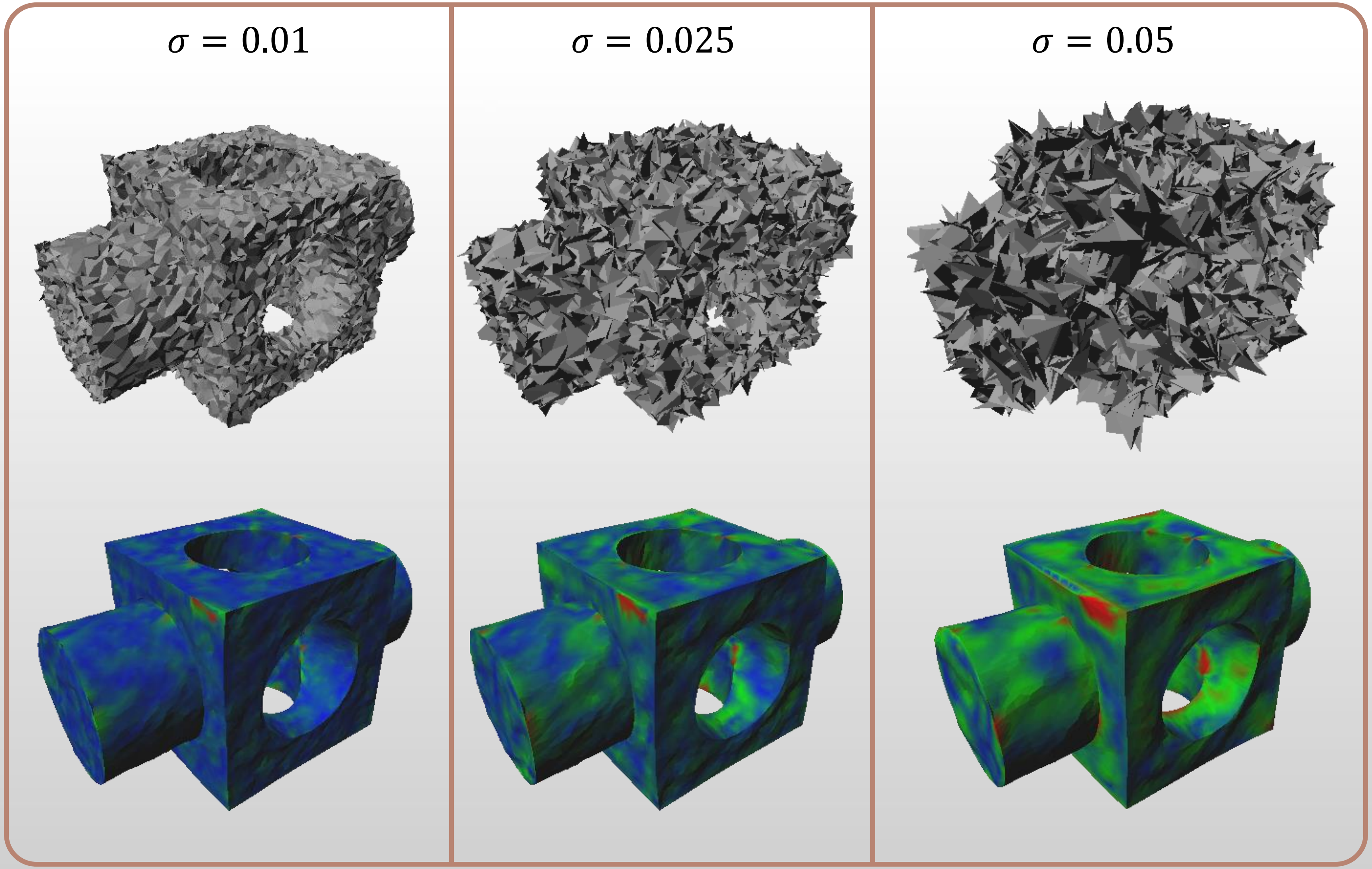


シャープさを考慮した平滑化

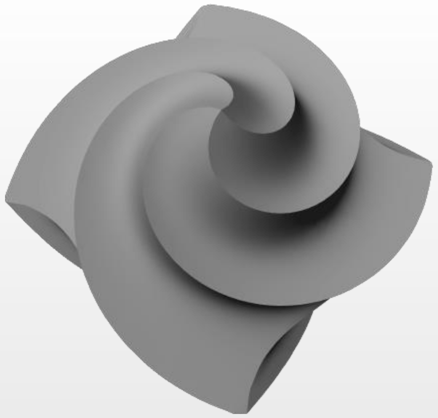


頂点数: 12812

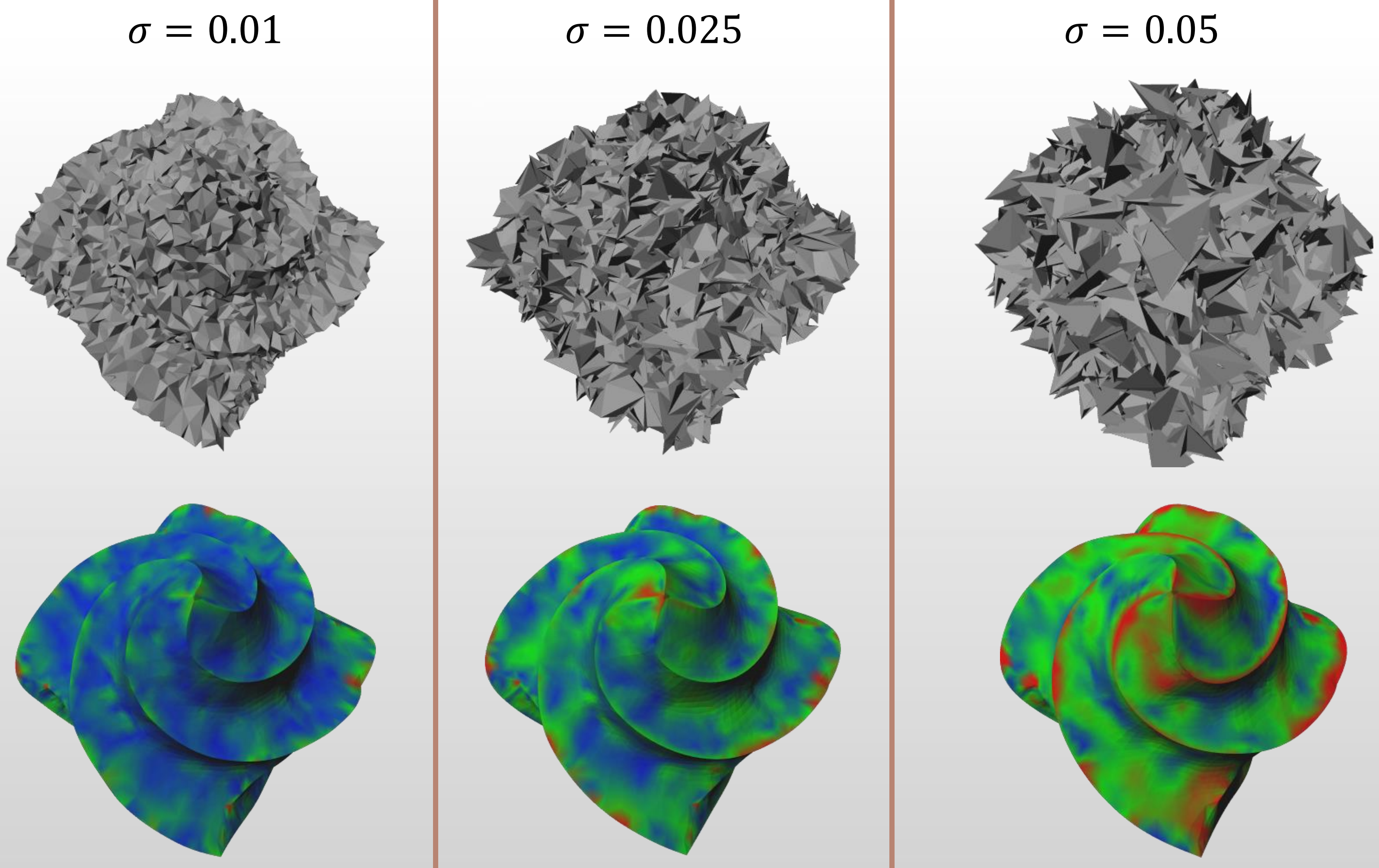


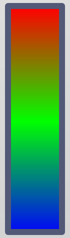


シャープさを考慮した平滑化

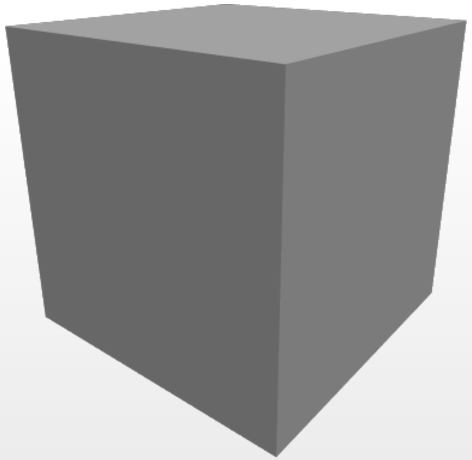


頂点数: 7919

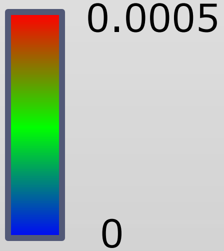


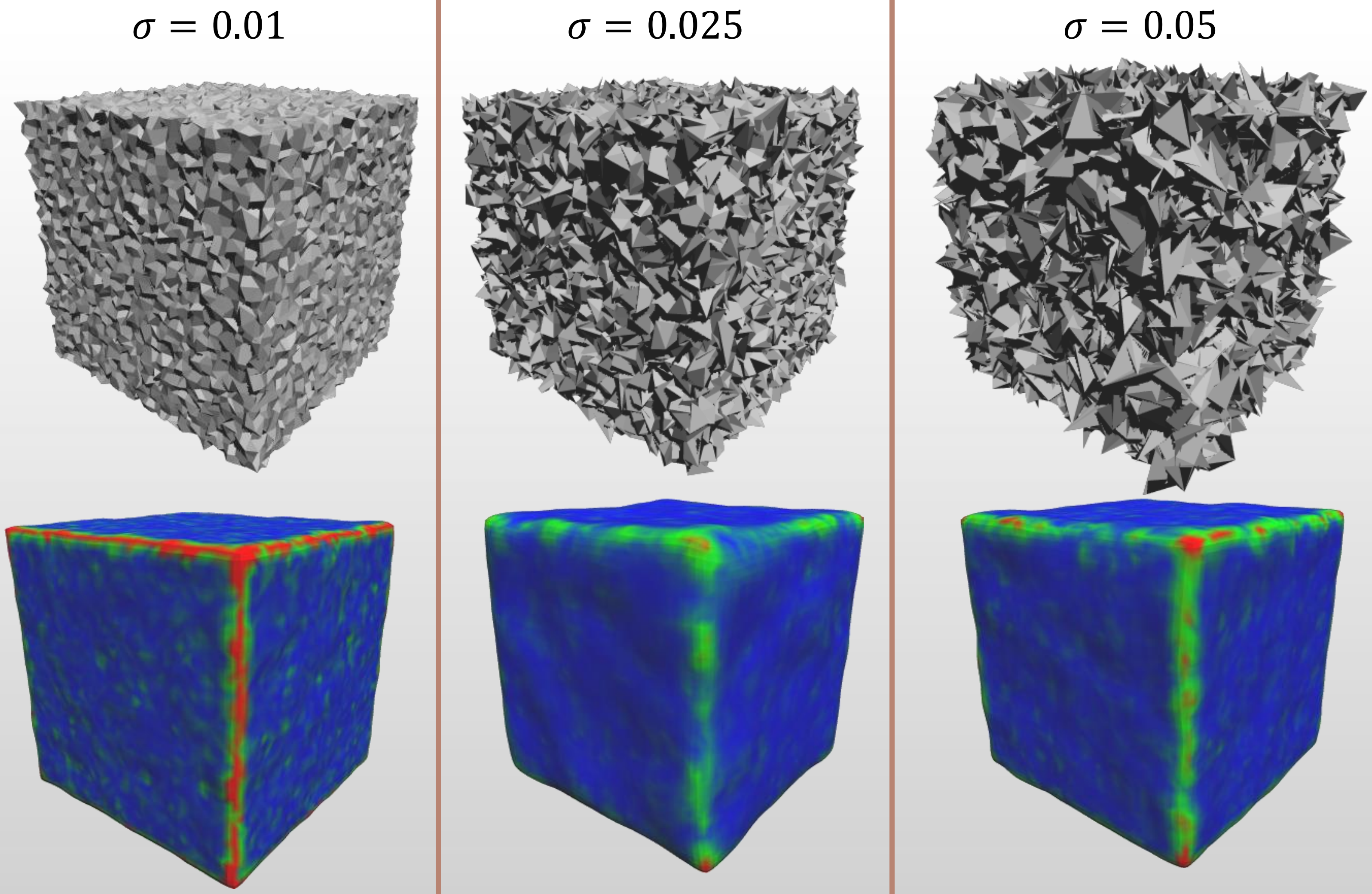


平滑化後のLaplacian成分の可視化

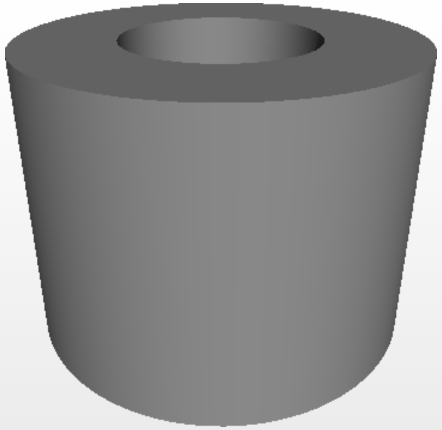


頂点数: 9602

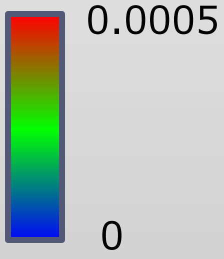




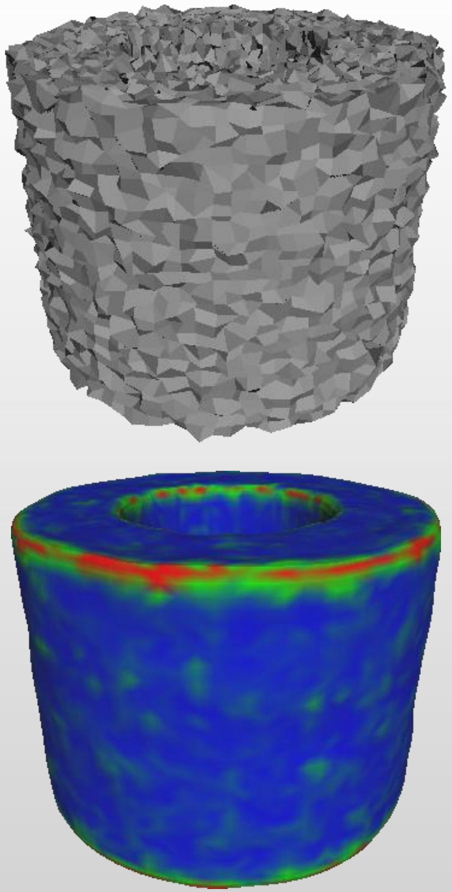
平滑化後のLaplacian成分の可視化



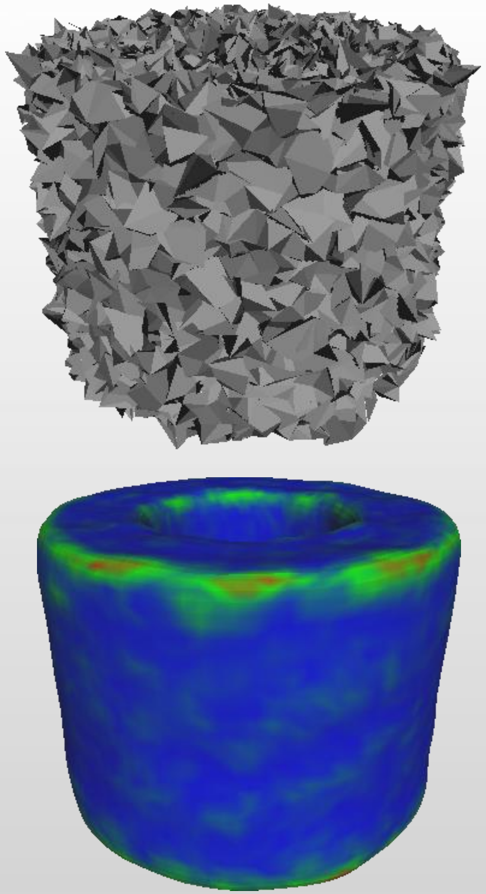
頂点数: 9000



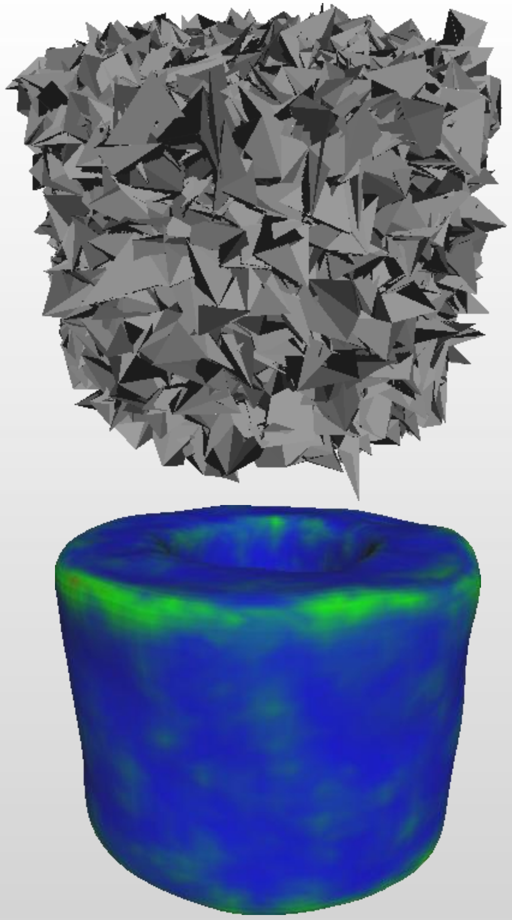
$\sigma = 0.01$



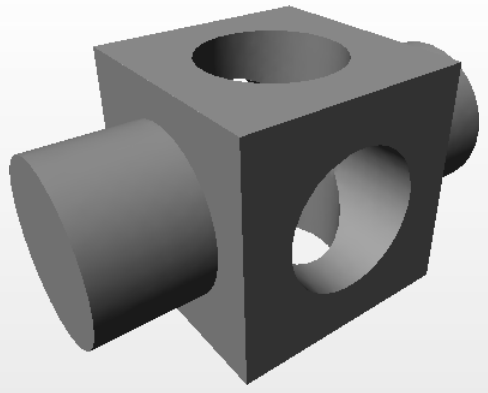
$\sigma = 0.025$



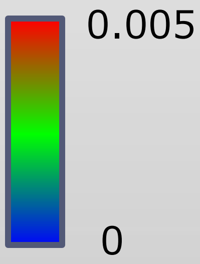
$\sigma = 0.05$

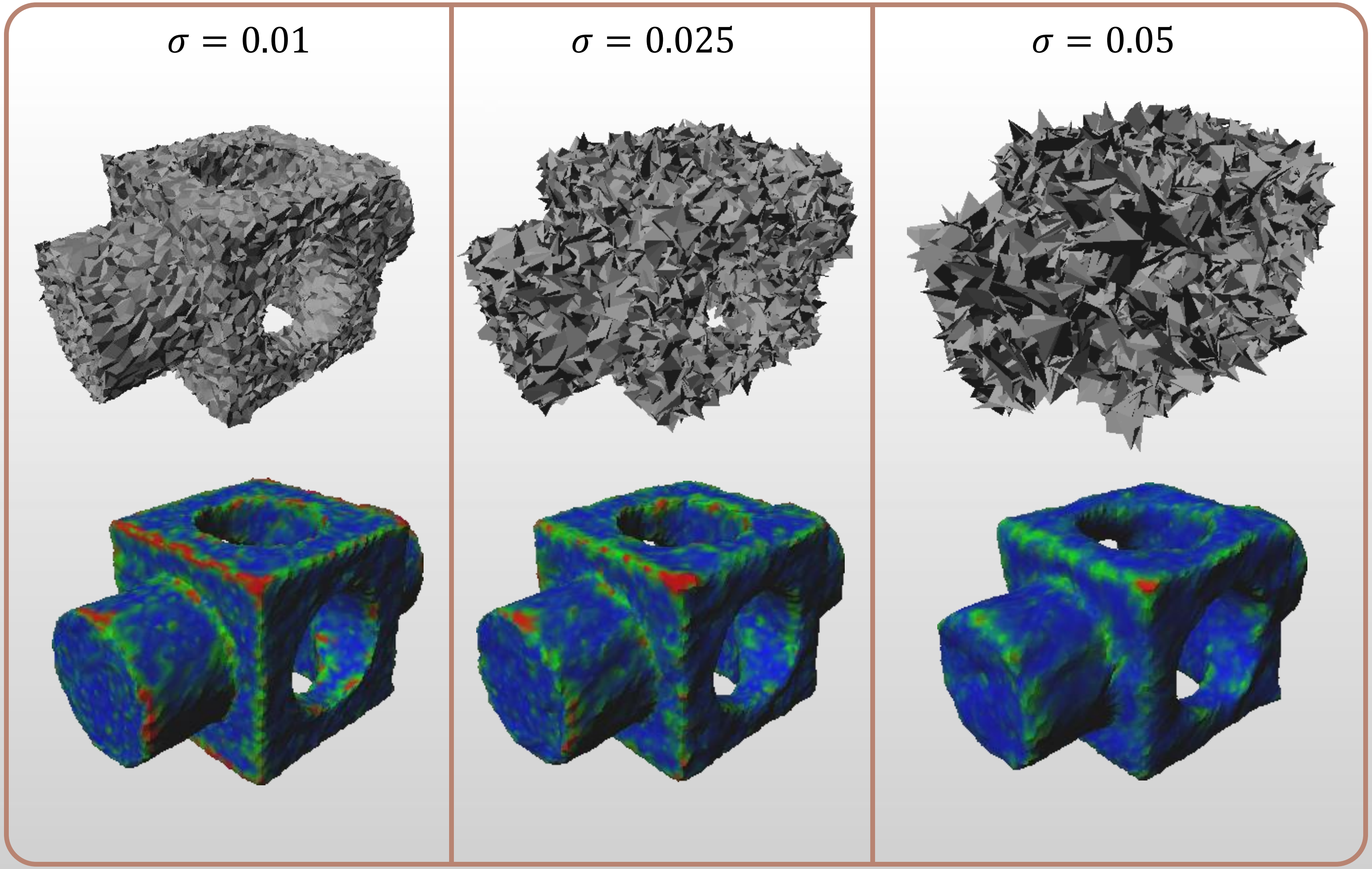


平滑化後のLaplacian成分の可視化

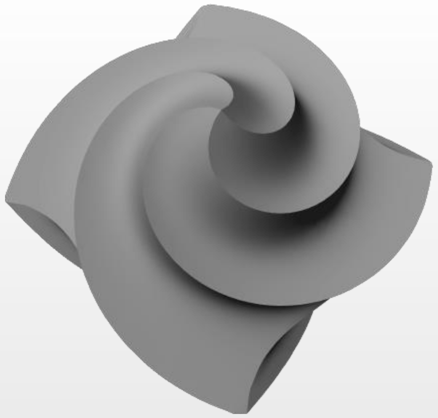


頂点数: 12812

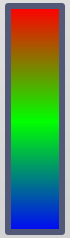


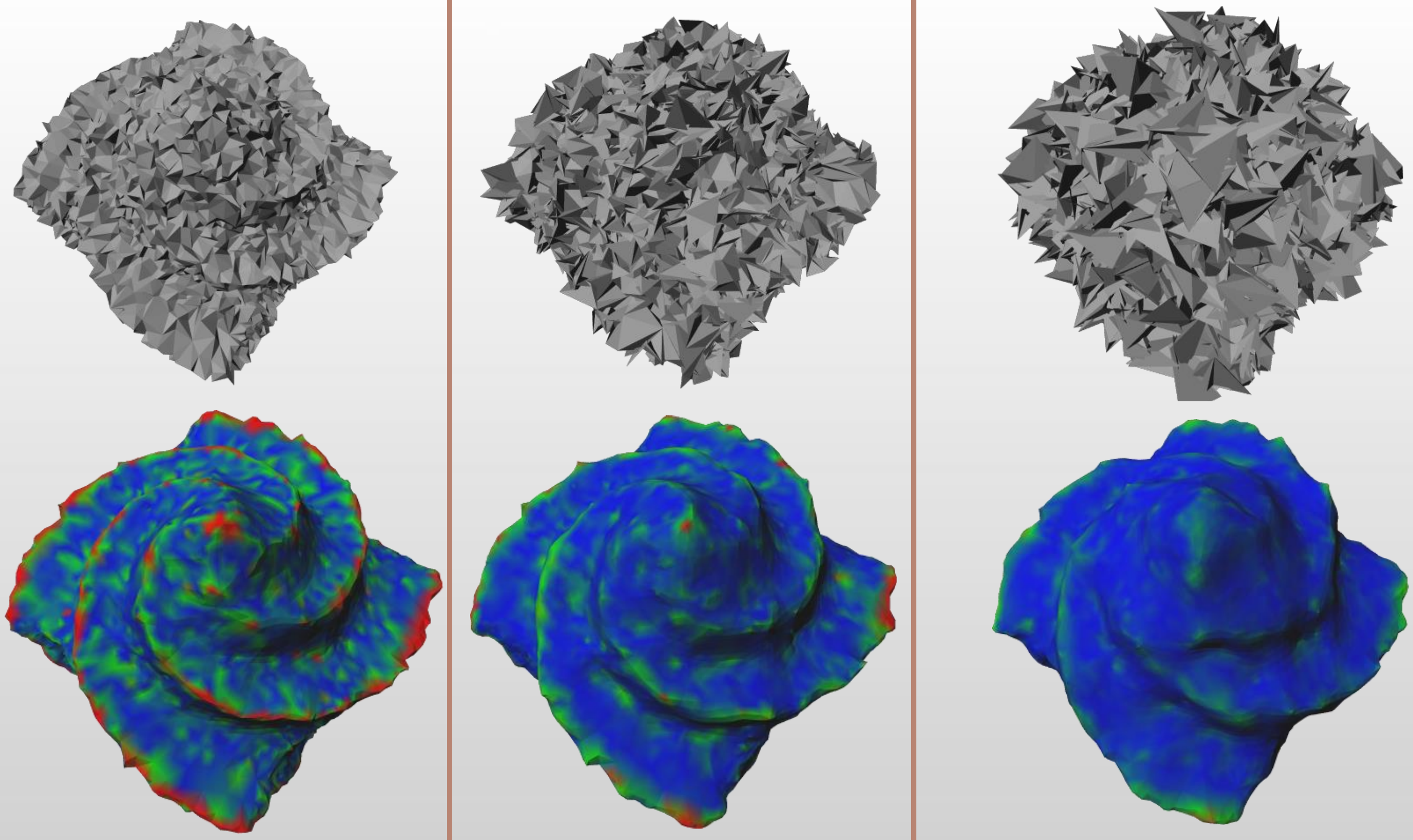


平滑化後のLaplacian成分の可視化

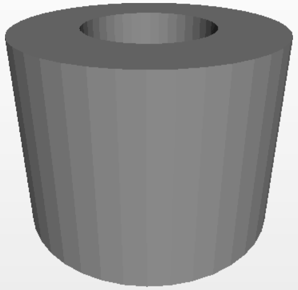


頂点数: 7919

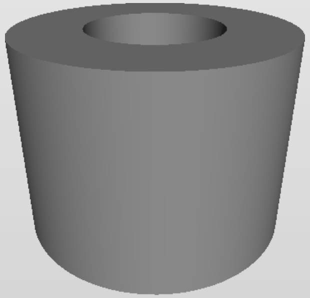




メッシュ解像度の重要性

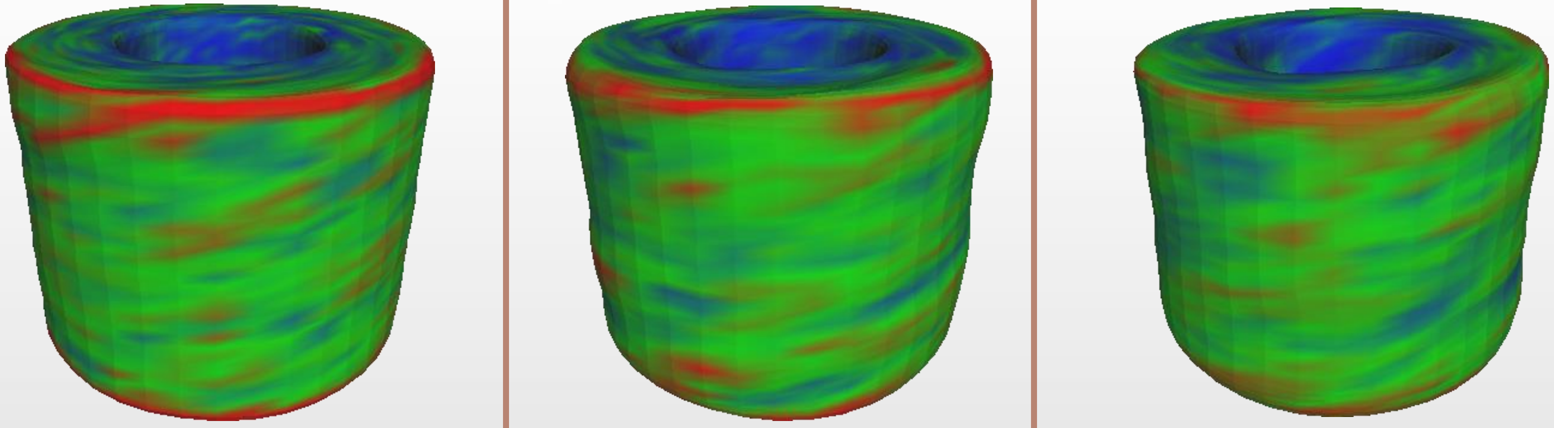


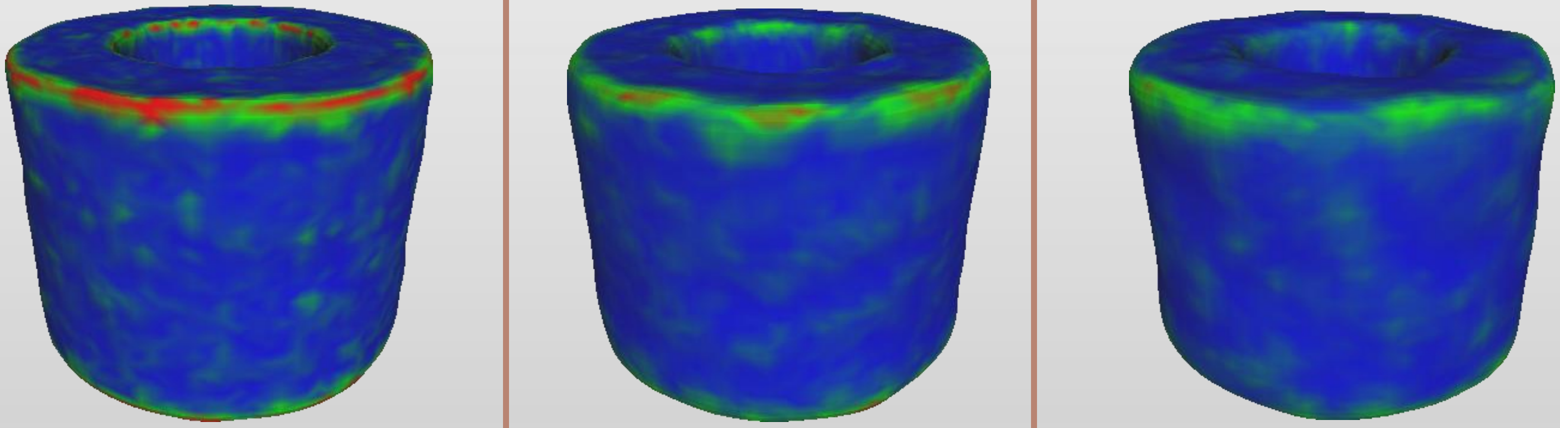
頂点数: 3600



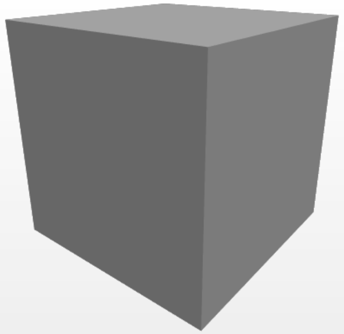
頂点数: 9000

$\sigma = 0.01$ | $\sigma = 0.025$ | $\sigma = 0.05$

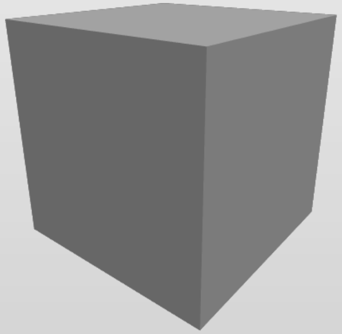




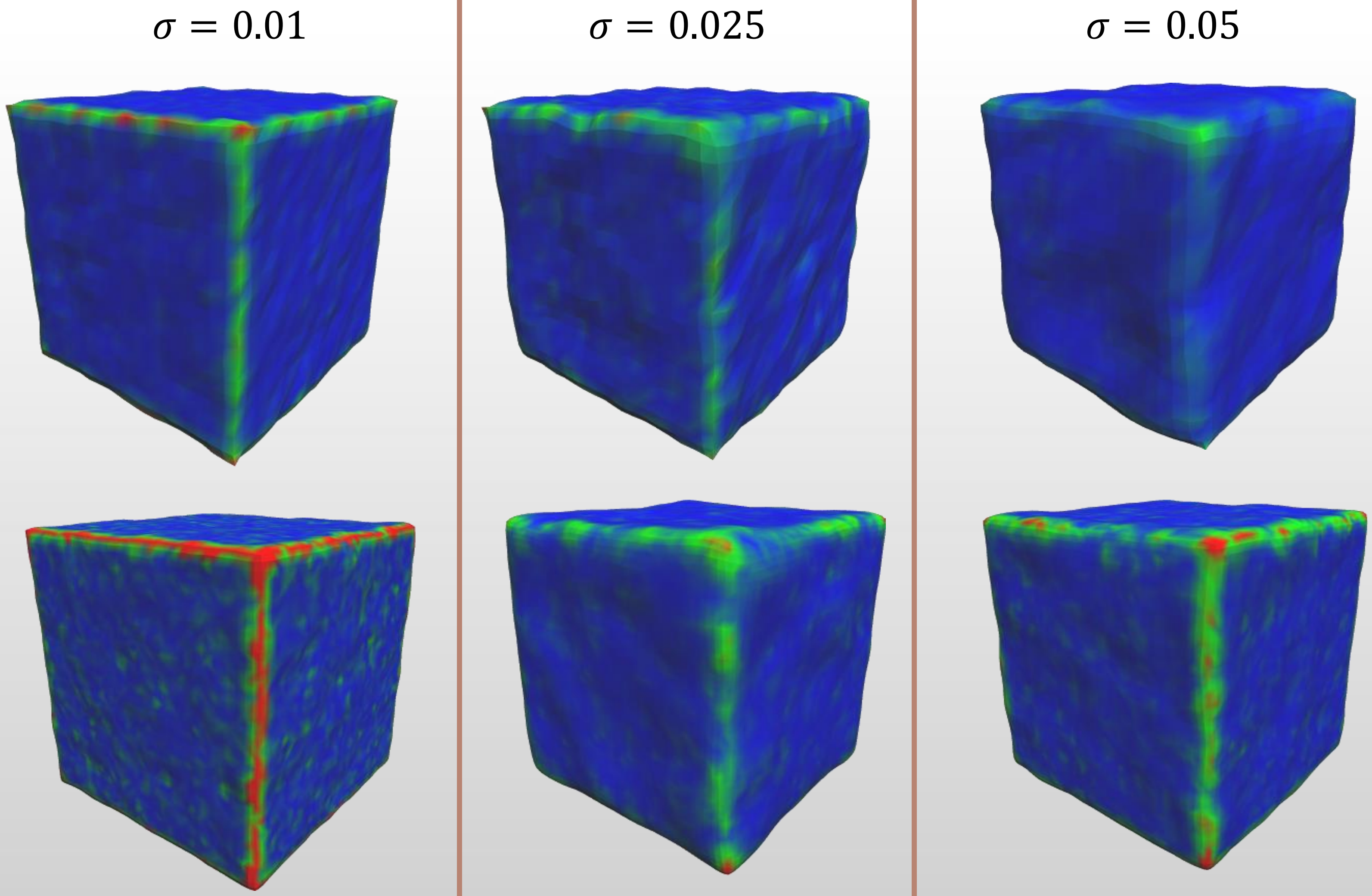
メッシュ解像度の重要性



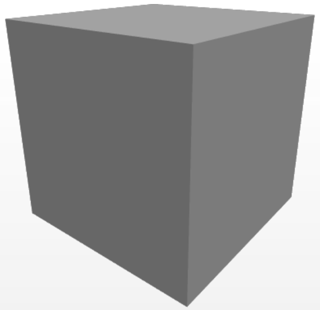
頂点数: 2402



頂点数: 9602

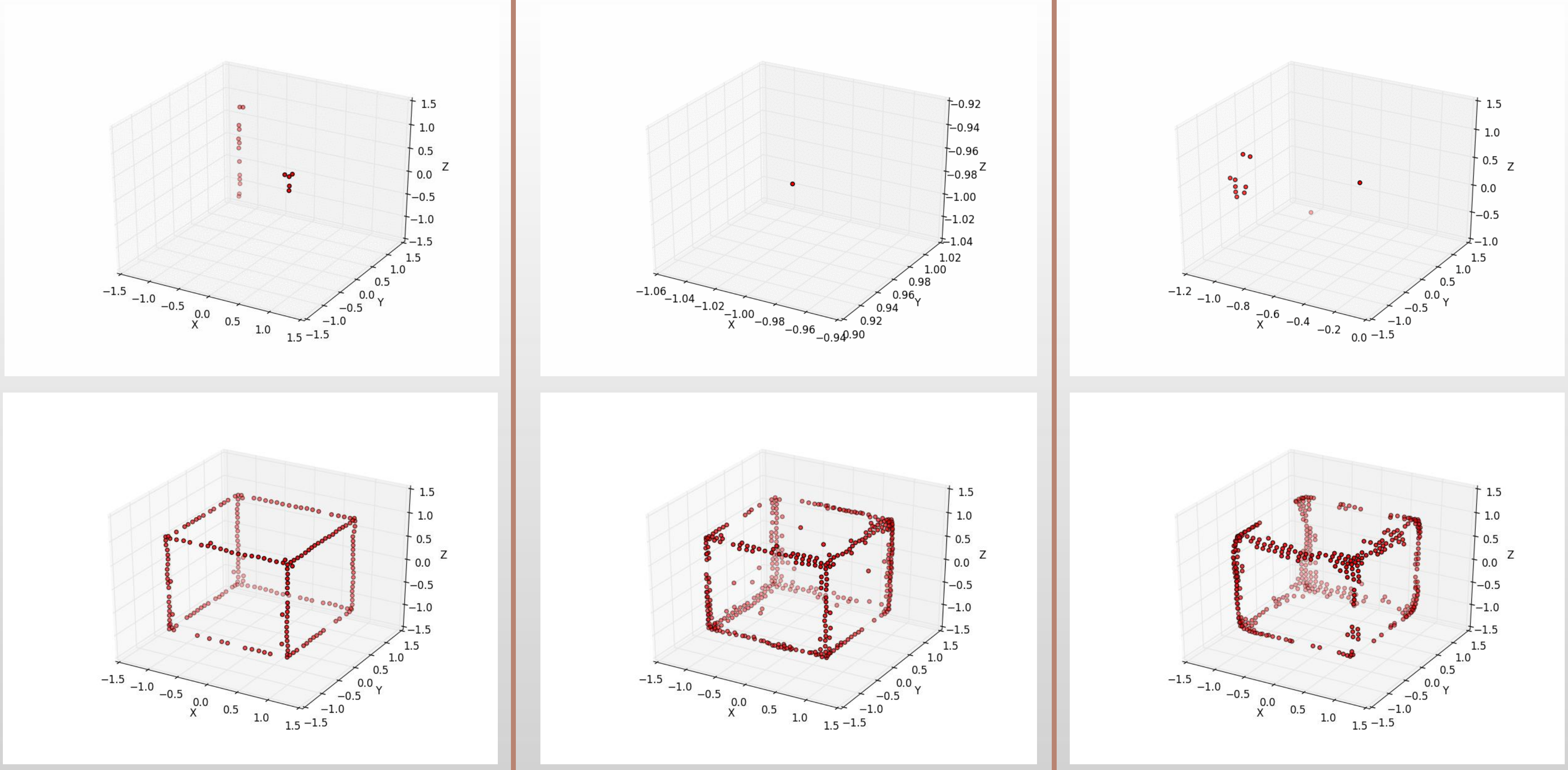


L1解析によるシャープな特徴の抽出

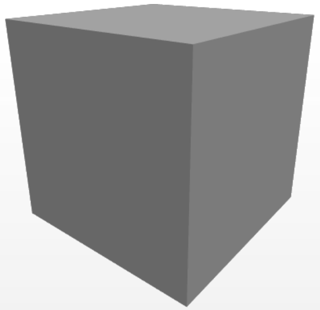


頂点数: 2402

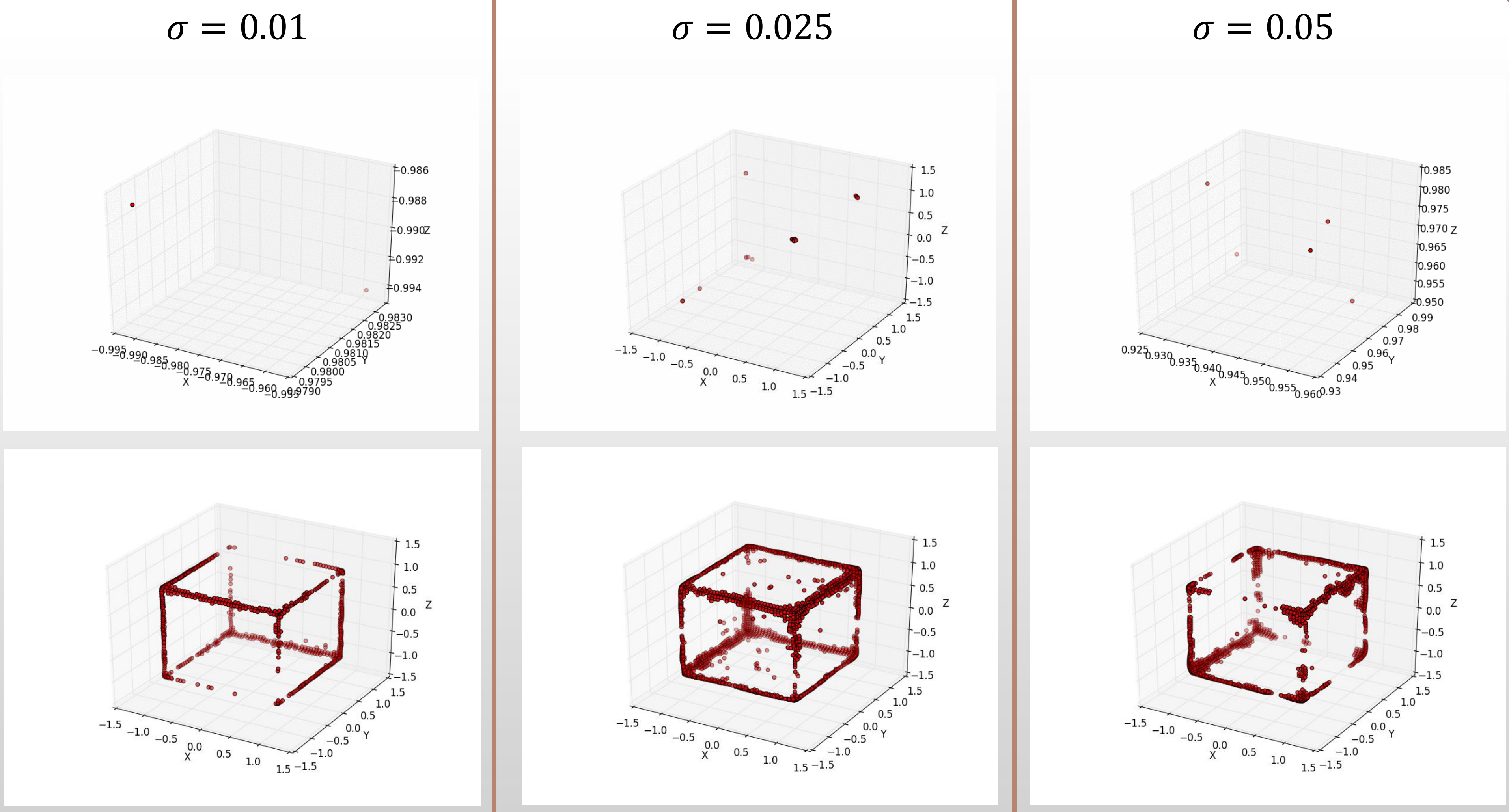
$\sigma = 0.01$ | $\sigma = 0.025$ | $\sigma = 0.05$



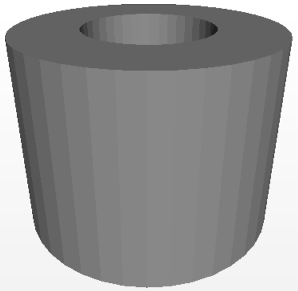
L1解析によるシャープな特徴の抽出



頂点数: 9602

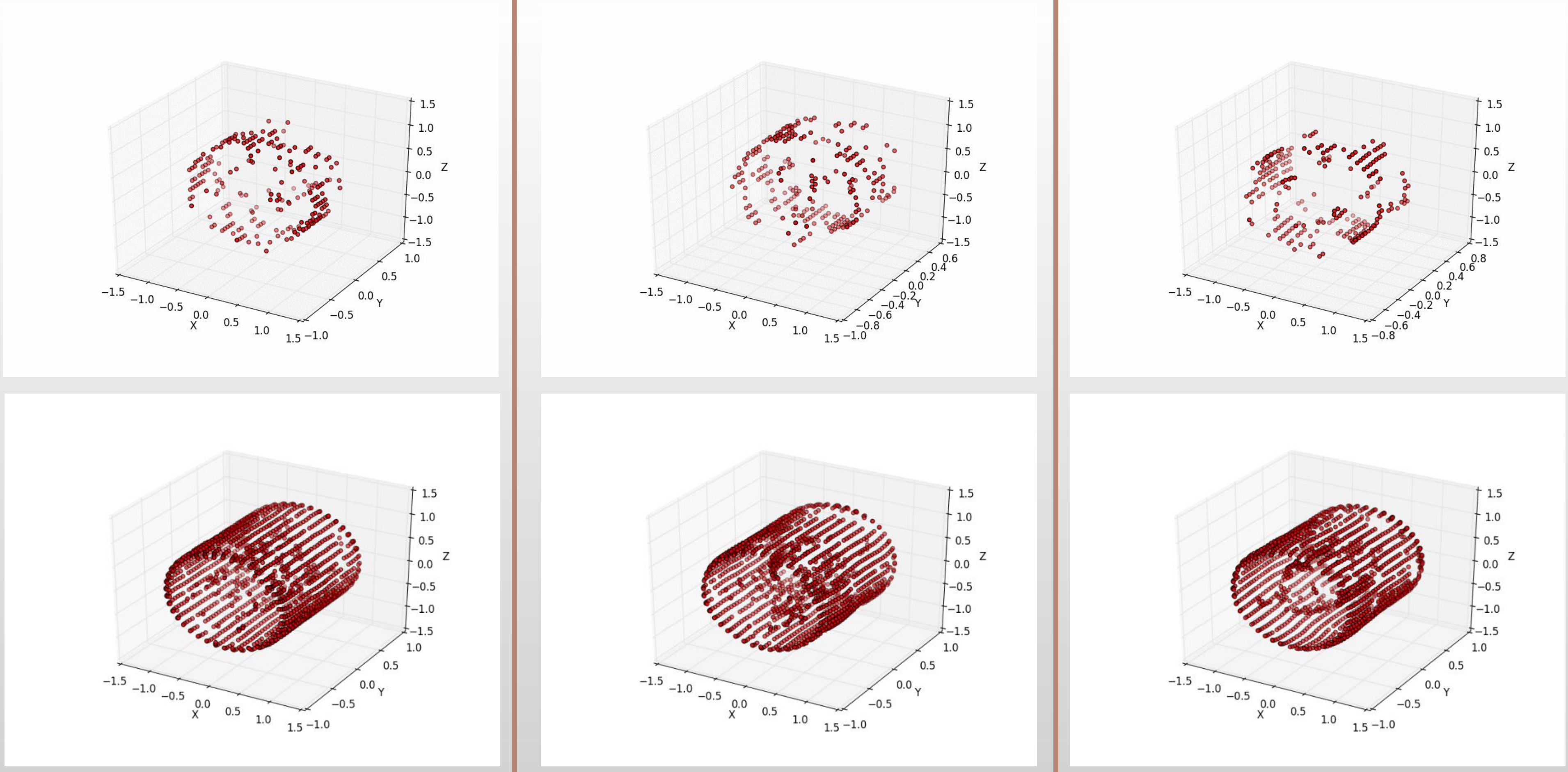


L1解析によるシャープな特徴の抽出

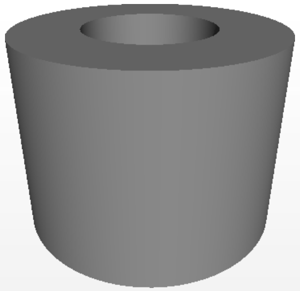


頂点数: 3600

$\sigma = 0.01$ $\sigma = 0.025$ $\sigma = 0.05$

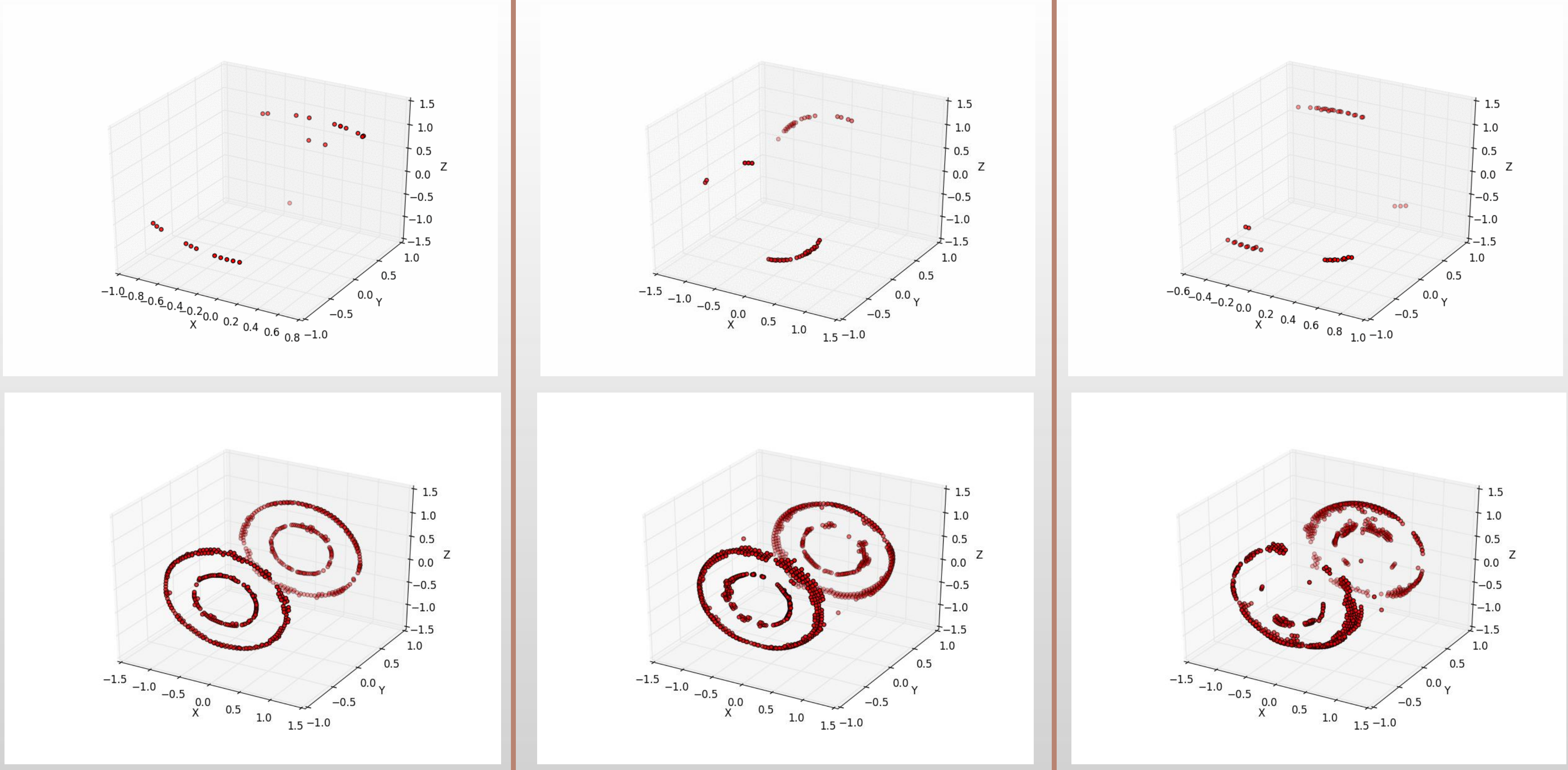


L1解析によるシャープな特徴の抽出

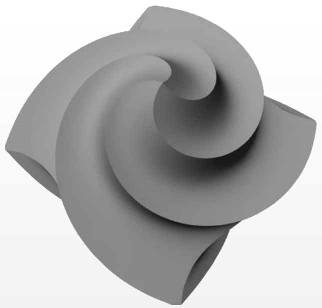


頂点数: 9000

$\sigma = 0.01$ | $\sigma = 0.025$ | $\sigma = 0.05$

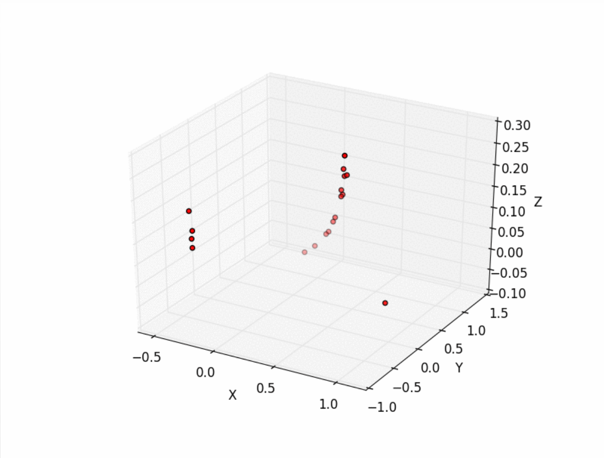


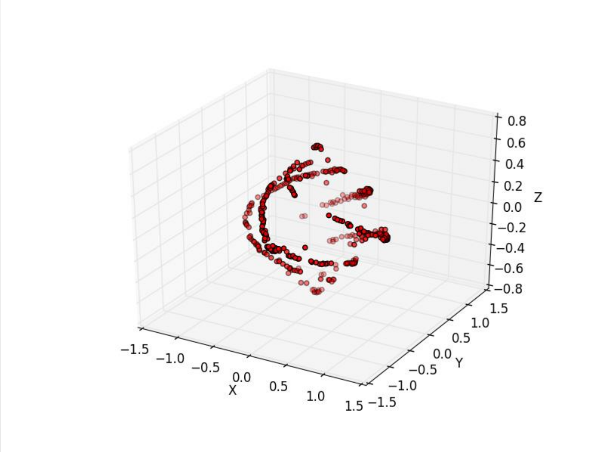
L1解析によるシャープな特徴の抽出



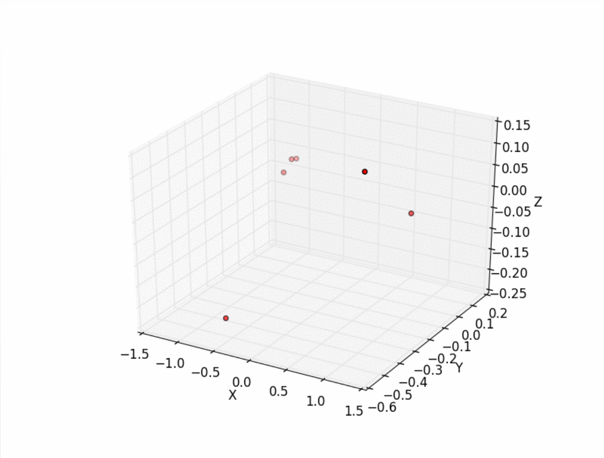
頂点数: 7919

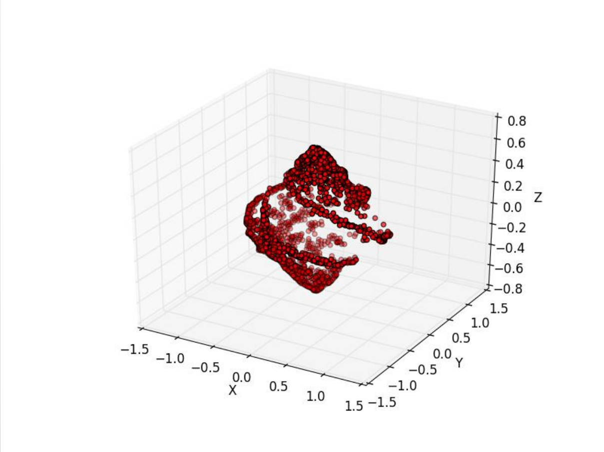
$\sigma = 0.01$



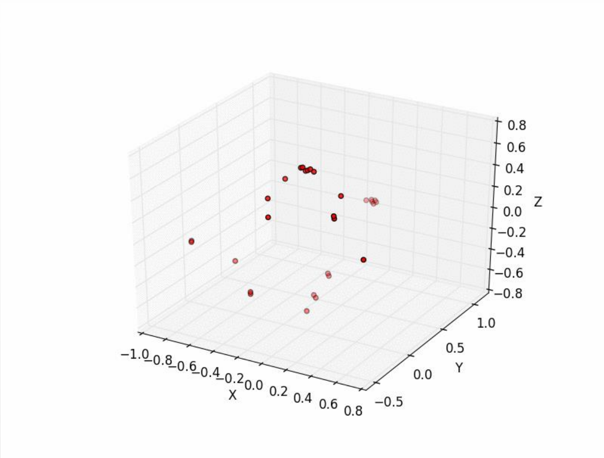


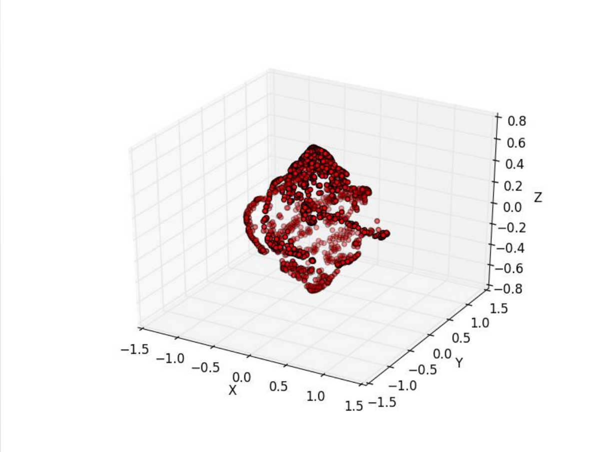
$\sigma = 0.025$





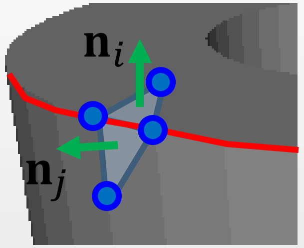
$\sigma = 0.05$

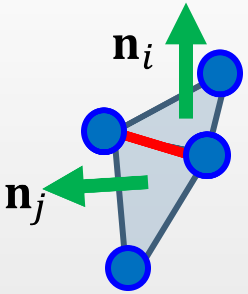




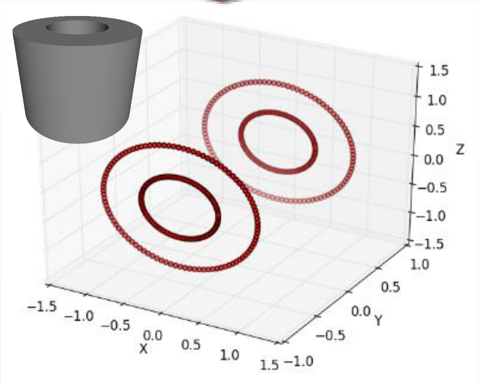
二面角によるシャープな特徴の検出

- 一般的なエッジのクリース判定
  - 二面角による判定: $\mathbf{n}_i \cdot \mathbf{n}_j < \beta$

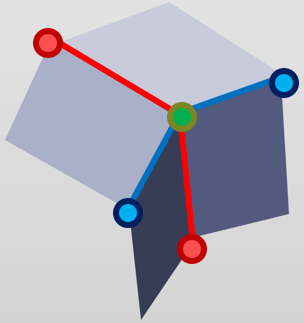


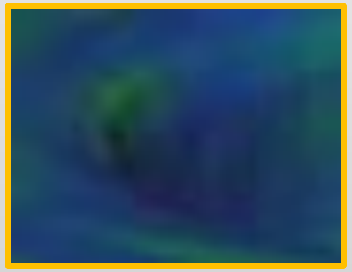


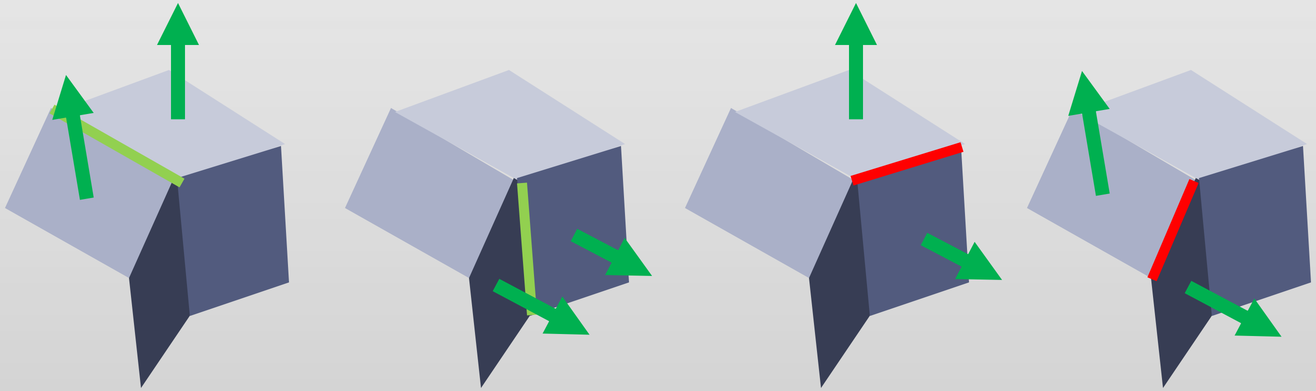
正解メッシュのクリース検出



- 鞍状でもシャープなエッジを特定可能

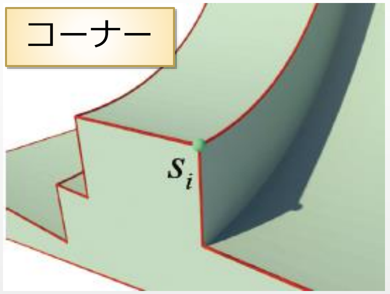




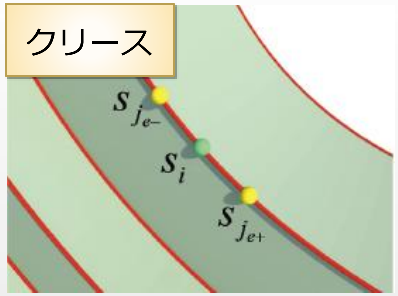


二面角度を利用した平滑化の試作

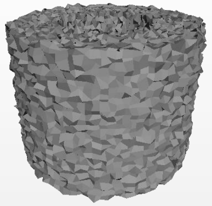
- シャープさを考慮した平滑化手法は有用

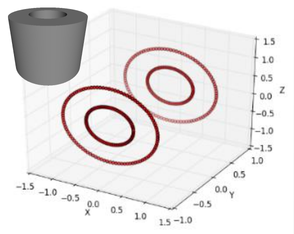


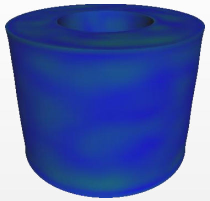
$\mathbf{L}(s_i) = L_i(S) = 0$



$\mathbf{L}(s_i) = L_i(S) = s_{je-} + s_{je+} - 2s_i$

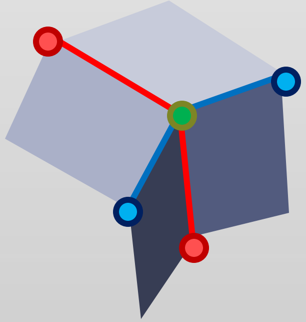






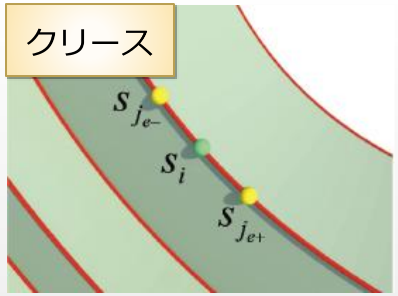
  - Laplacian行列の修正+GCVを利用した平滑化

- 二面角を利用してLaplacian行列を繰り返し処理で正解に近づける
  - エッジの２面角を重みとした Laplacian行列の修正

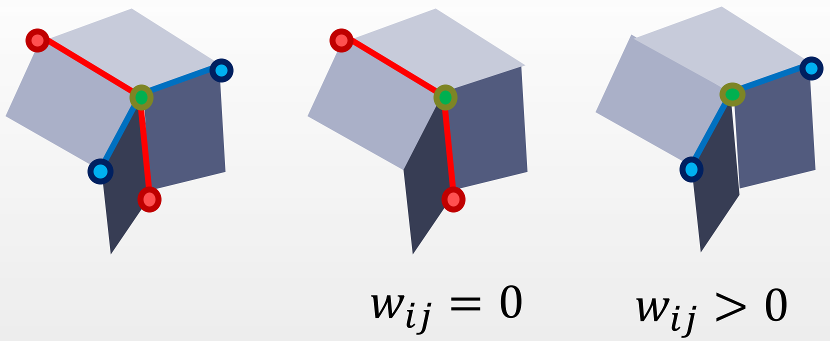


エッジの二面角を利用したLaplacian行列の修正

- クリースのLaplacian行列に注目
  - 二面角が大きい方向は$w_{ij} = 0$
  - 二面角が小さい方向は$w_{ij} > 0$



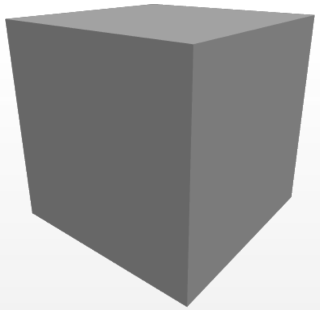
$\mathbf{L}(s_i) = L_i(S) = s_{je-} + s_{je+} - 2s_i$



- Laplacian重みの修正
  - $w_{ij} = \lfloor \mathbf{n}_i \cdot \mathbf{n}_j - \beta \rfloor$

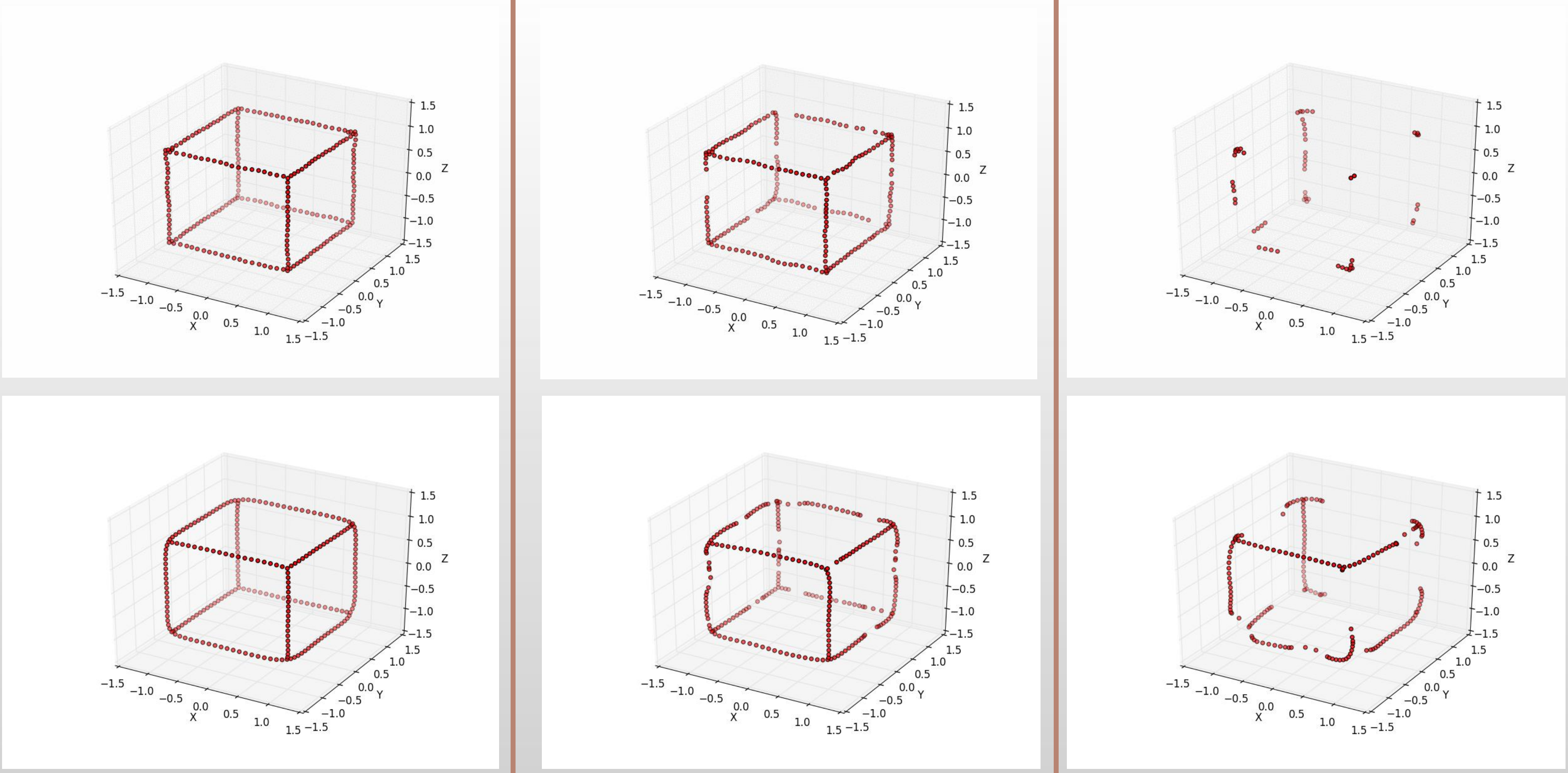
- 繰り返しLaplacian行列を修正しながら平滑化していく

二面角によるシャープな特徴の検出

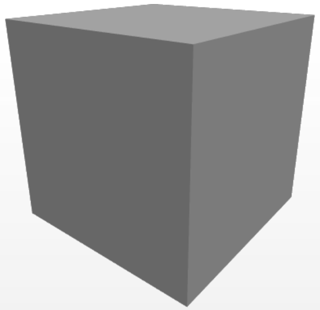


頂点数: 2402

$\sigma = 0.01$ $\sigma = 0.025$ $\sigma = 0.05$

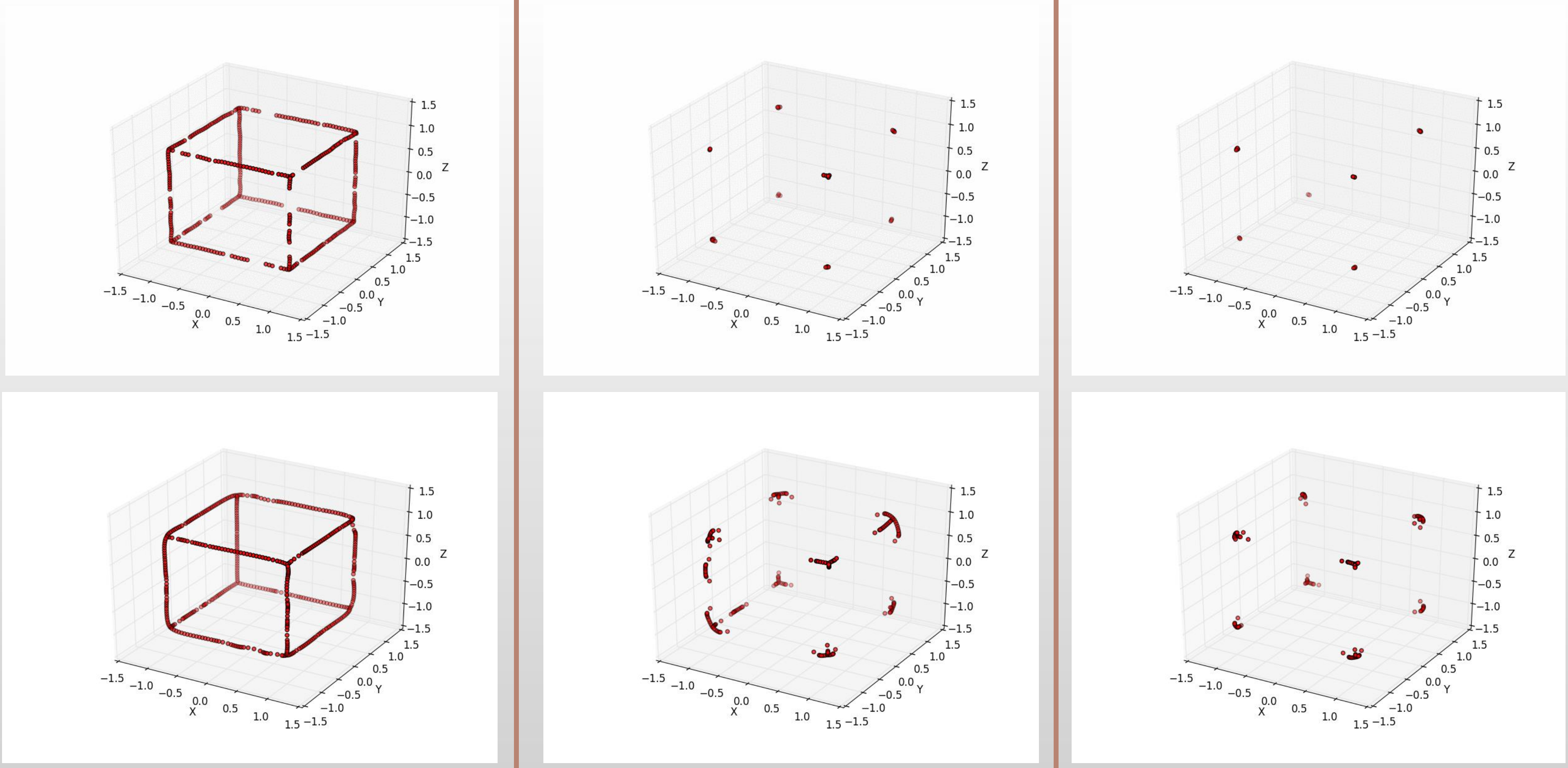


二面角によるシャープな特徴の検出

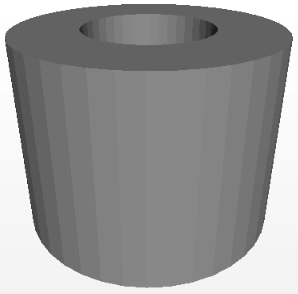


頂点数: 9602

$\sigma = 0.01$ $\sigma = 0.025$ $\sigma = 0.05$

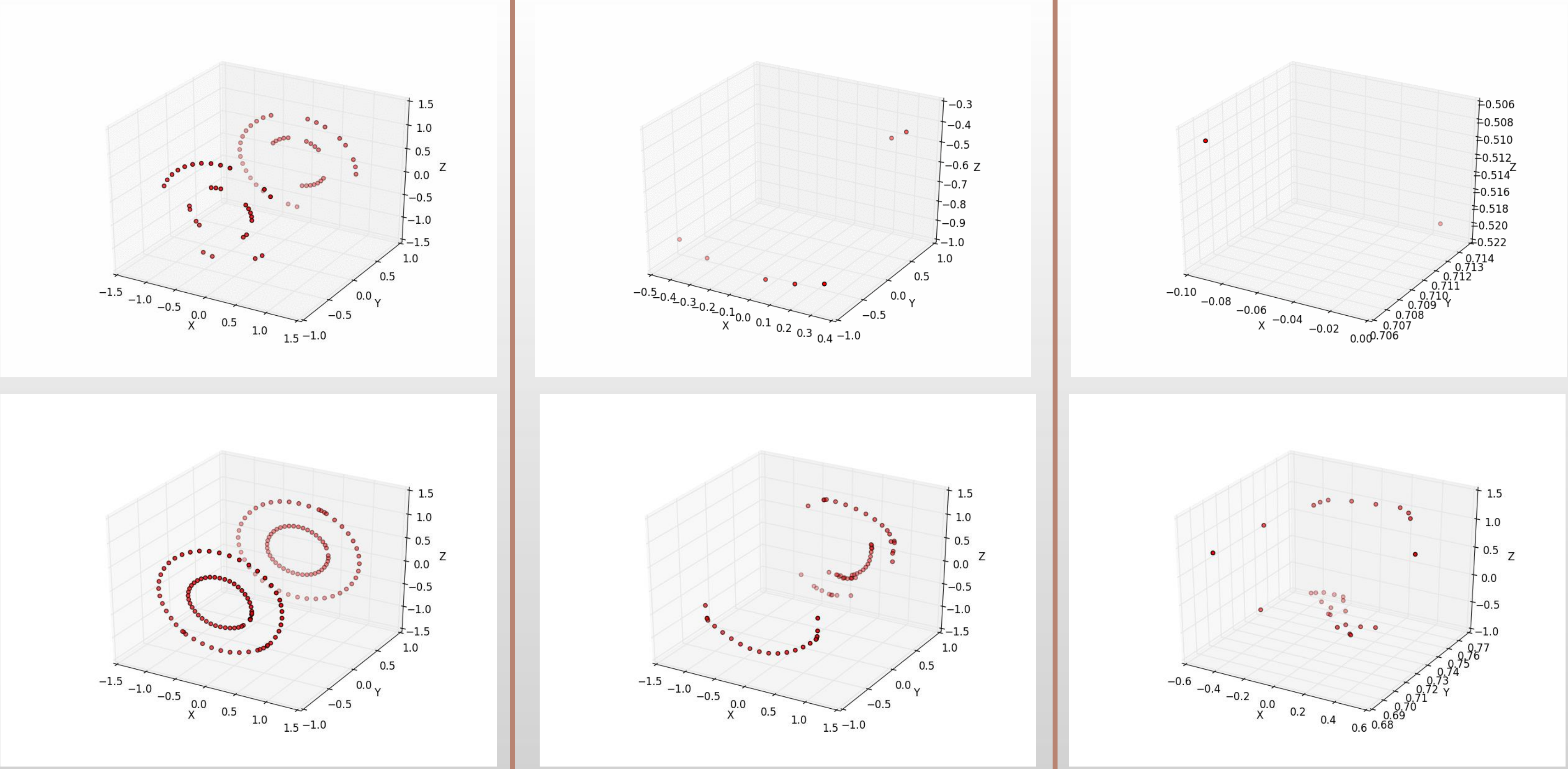


二面角によるシャープな特徴の検出

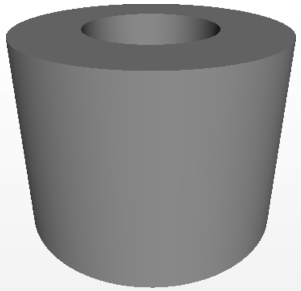


頂点数: 3600

$\sigma = 0.01$	$\sigma = 0.025$	$\sigma = 0.05$

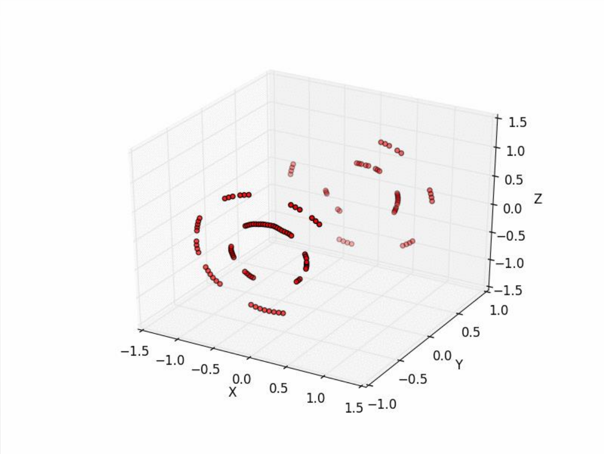


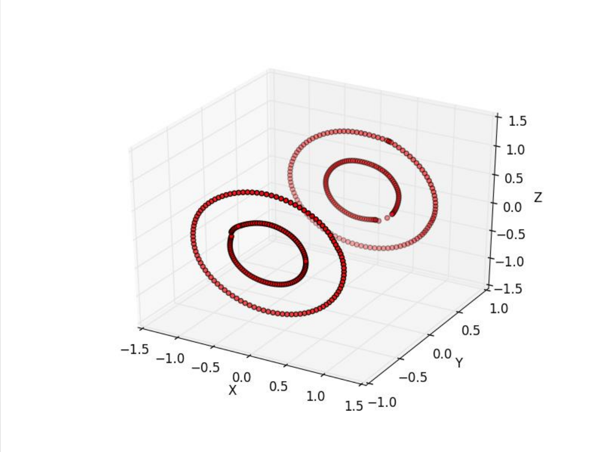
二面角によるシャープな特徴の検出



頂点数：9000

$\sigma = 0.01$

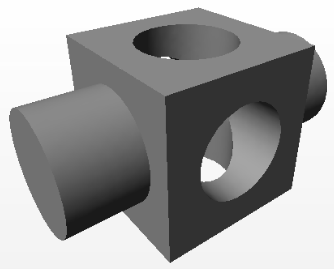




$\sigma = 0.025$

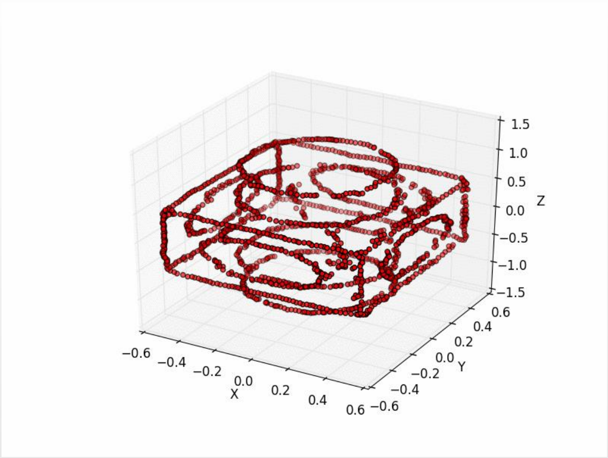
$\sigma = 0.05$

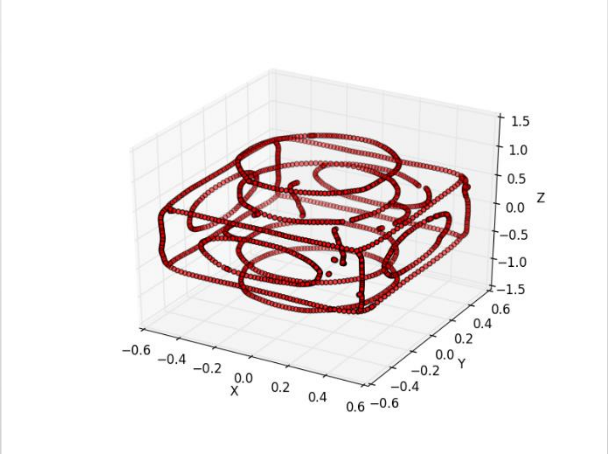
L1解析によるシャープな特徴の抽出



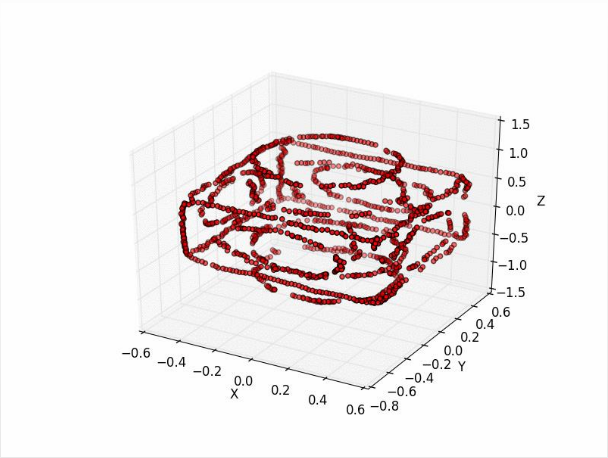
頂点数: 12812

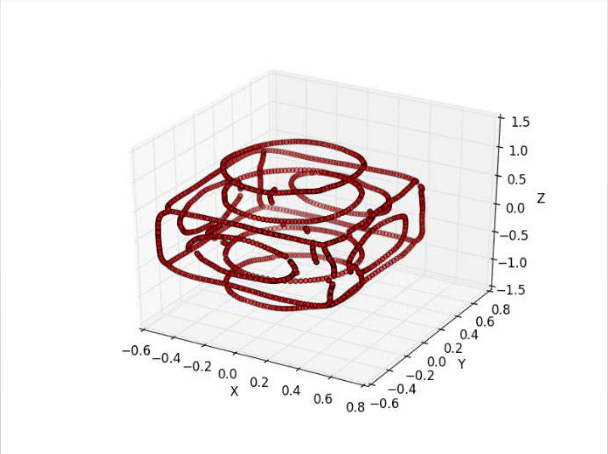
$\sigma = 0.01$



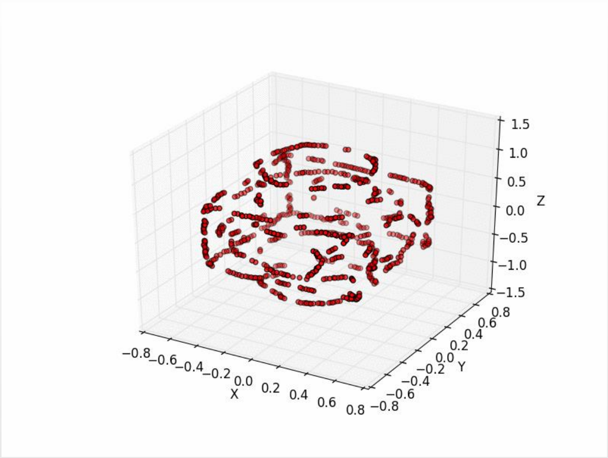


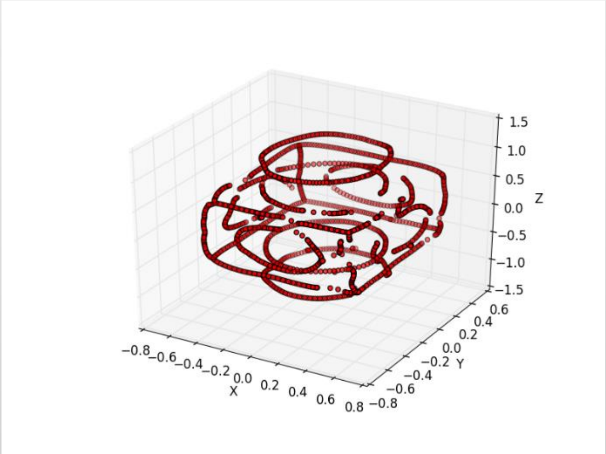
$\sigma = 0.025$



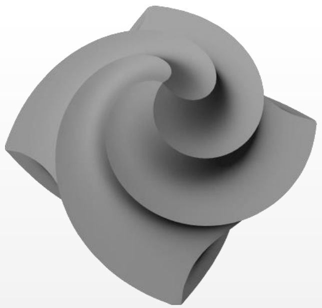


$\sigma = 0.05$



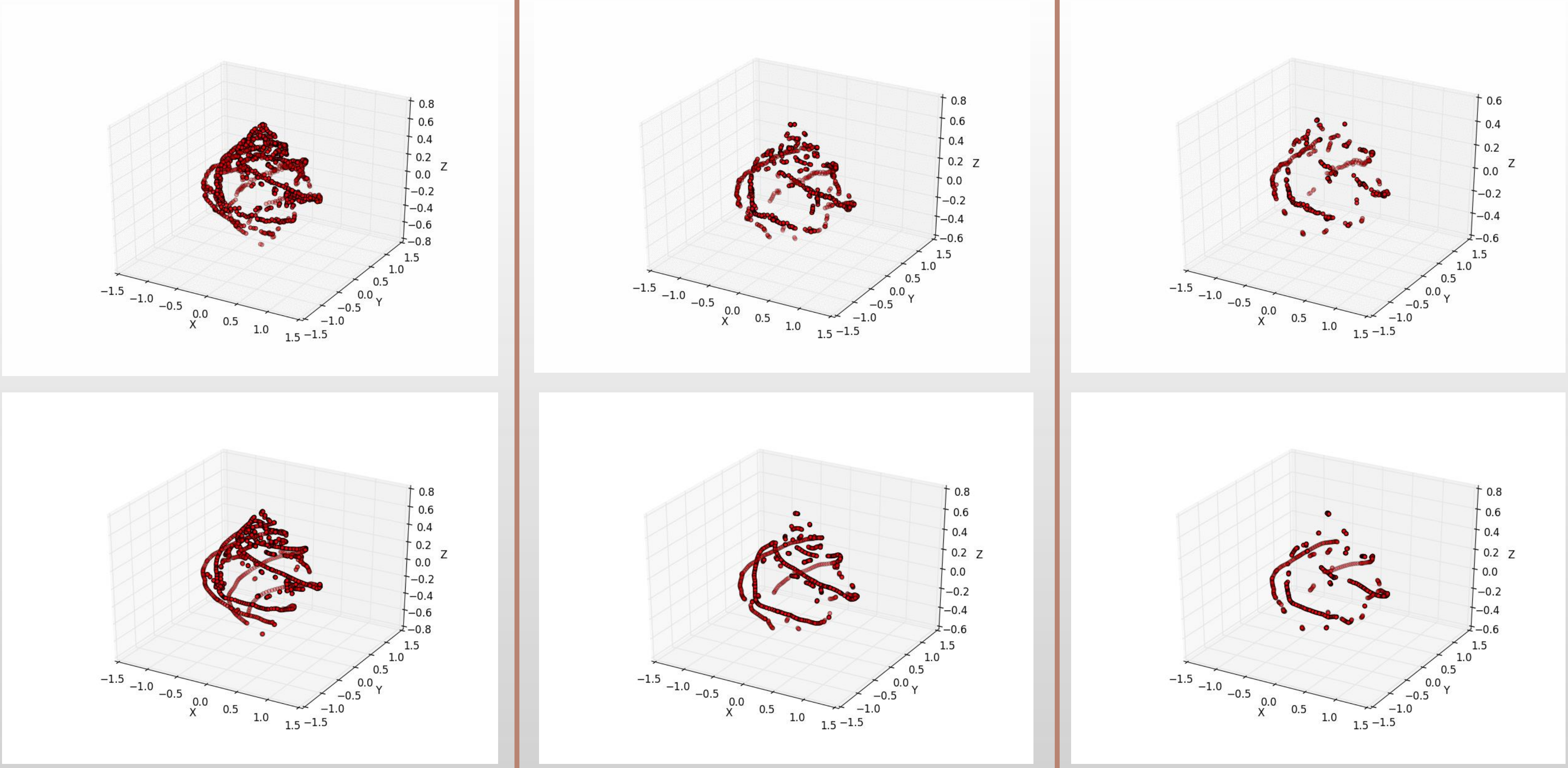


二面角によるシャープな特徴の検出



頂点数: 7919

$\sigma = 0.01$ | $\sigma = 0.025$ | $\sigma = 0.05$



実装について

- Python: 各種アルゴリズムの実装，テスト
  - Numpy: 密行列演算
  - Scipy: 疎行列演算
  - Matplotlib: グラフの表示
- Maya Plugin: DLRS のデモ
  - Eigen: 行列演算
- ※： 並列化計算，GPU処理による最適化は一切行っていない

マシンスペック

デスクトップPC	
CPU	Intel Core i7 3.4GHz
メモリ	16.0 GB
GPU	GeForce GTX 770 （メモリ 2048MB）

ノートPC	
CPU	Intel Core i3 2.5GHz
メモリ	4.0 GB
GPU	GeForce GT 640 M LE