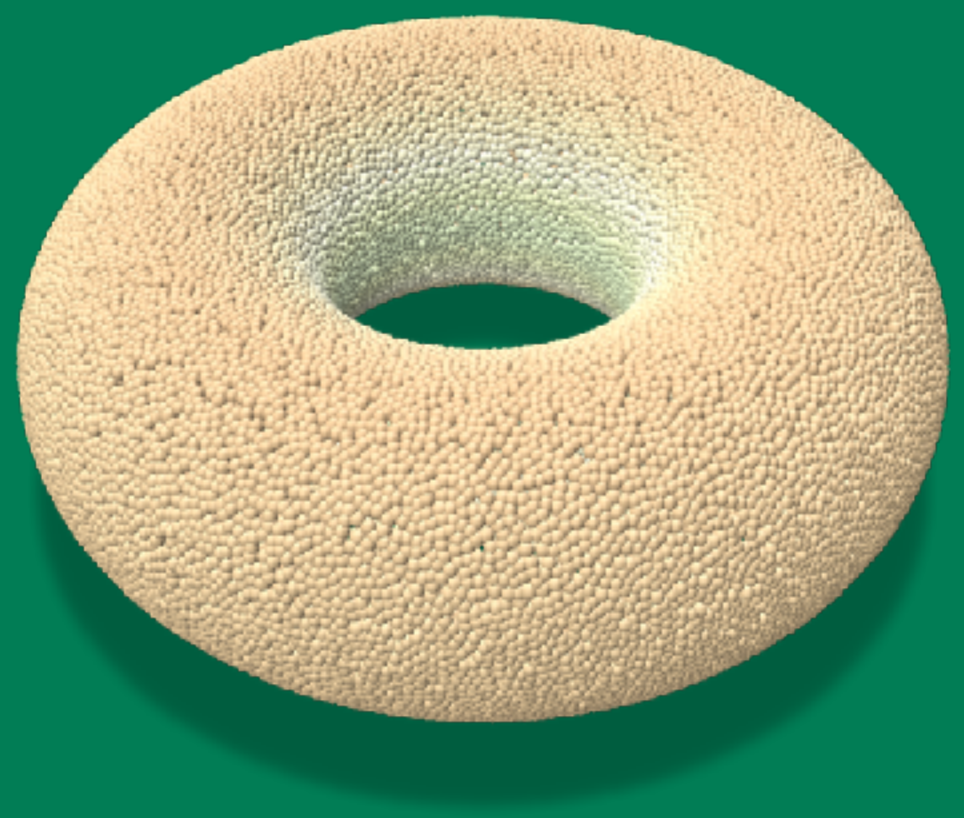
Estimating Discrete Total Curvature with Per Triangle Normal Variation

Crane He Chen

Week9

The Johns Hopkins University



Administration

Next week's meeting time

Problem Statement

The Output

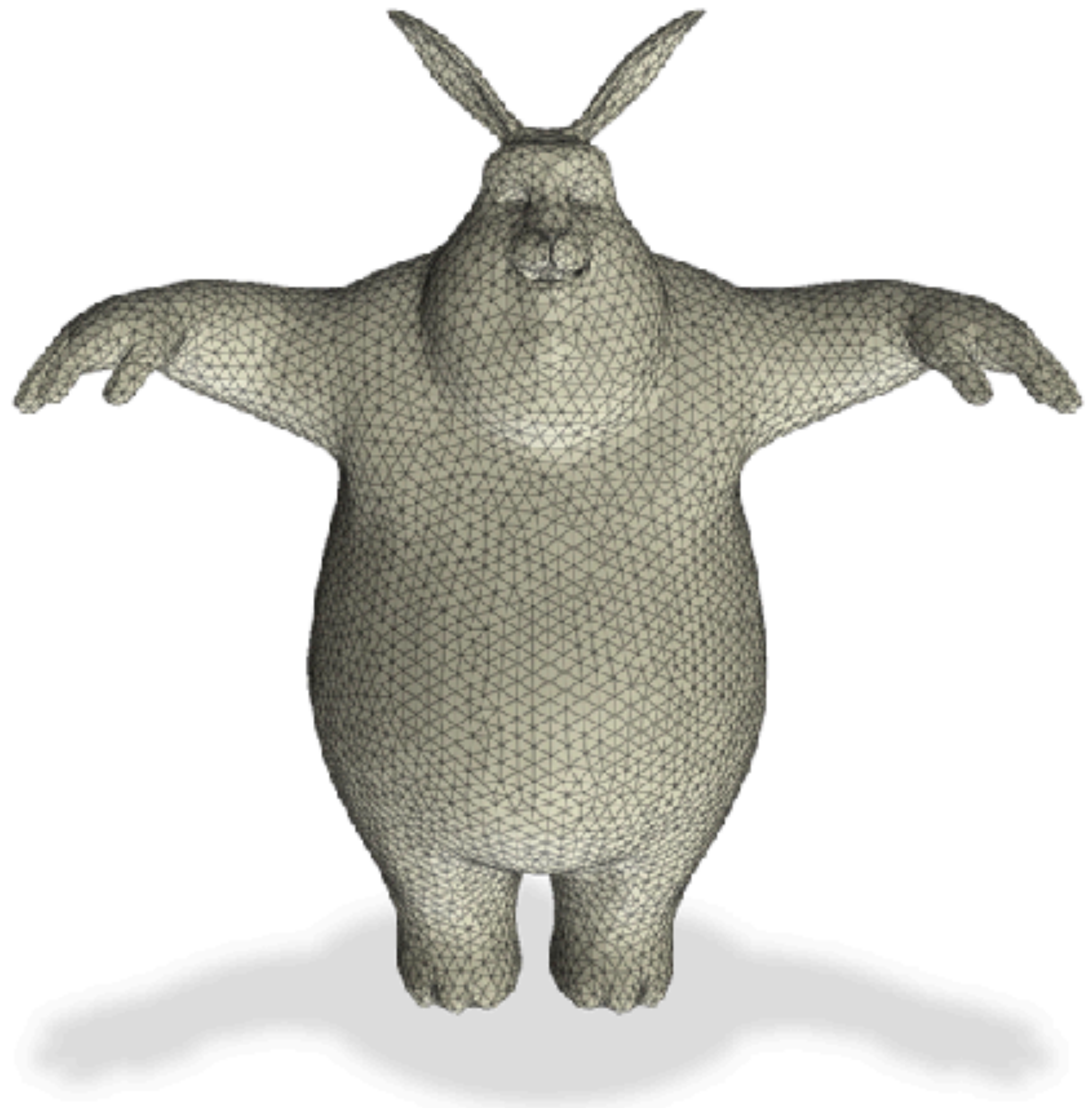
The Input

A method for computing total curvature of triangle meshes or point clouds, while avoiding the calculation of the shape operator

How the calculation works

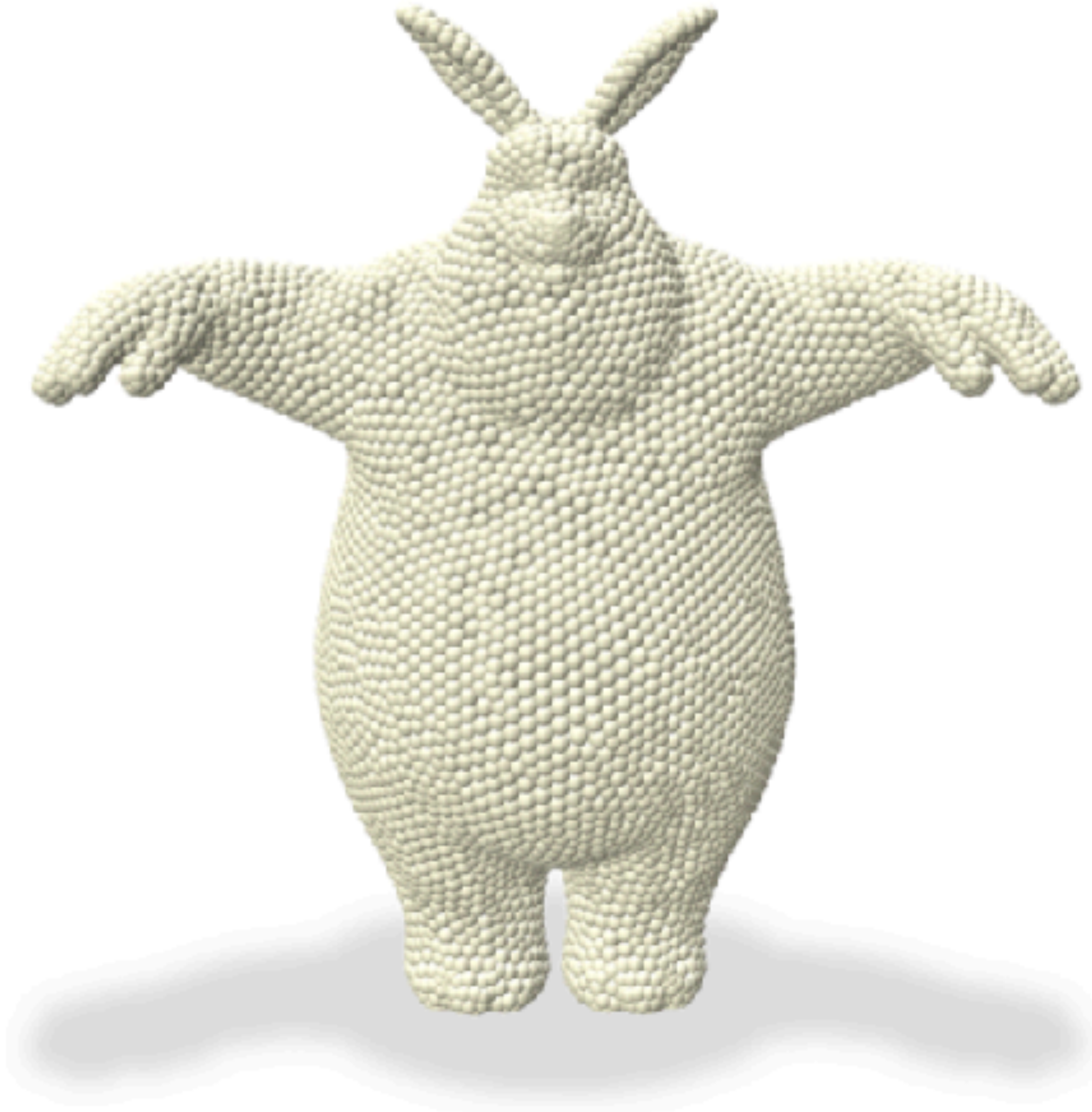
SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

Problem Statement

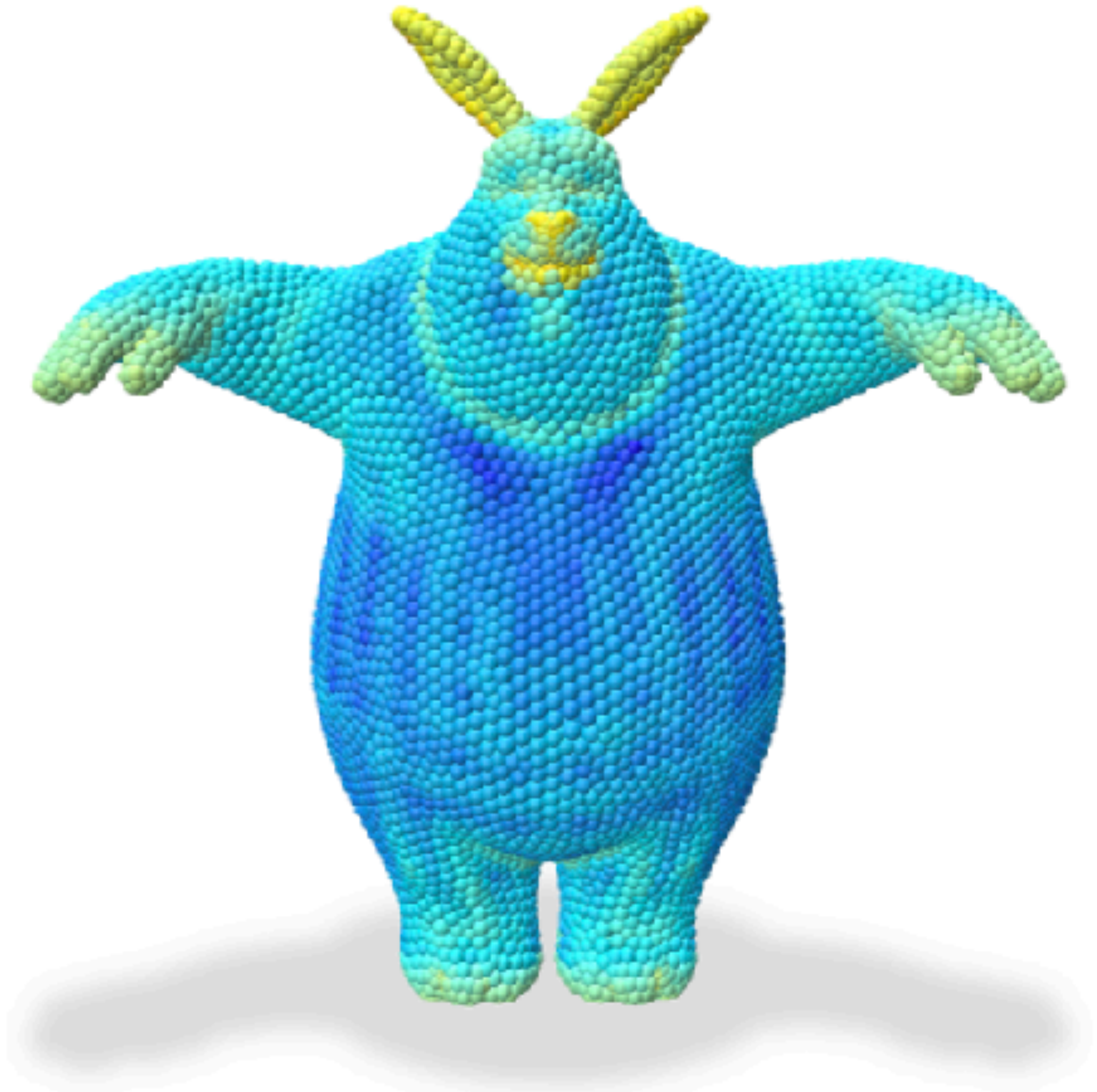


input
(triangle mesh)

output
(total curvature)



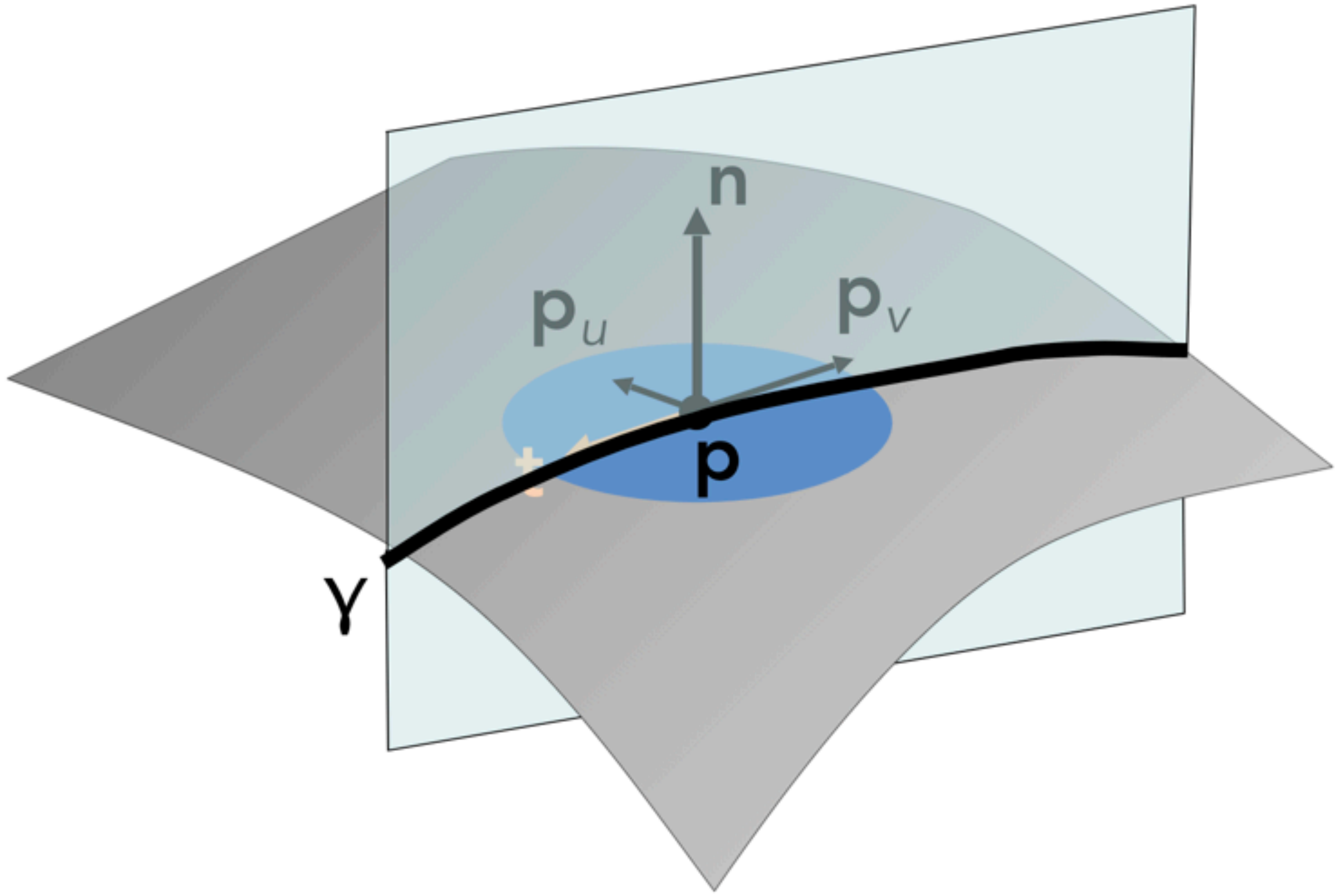
input
(point cloud)



output
(total curvature)

Problem Statement

Background: Surface Curvature



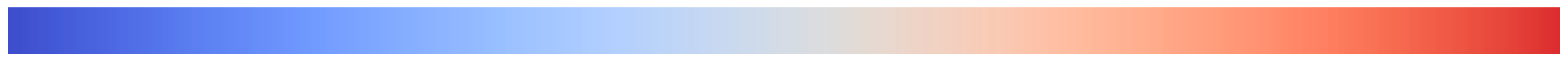
Minimal Curvature $\kappa_1 = \kappa_{min} = \min_{\phi} \kappa_n(\phi)$

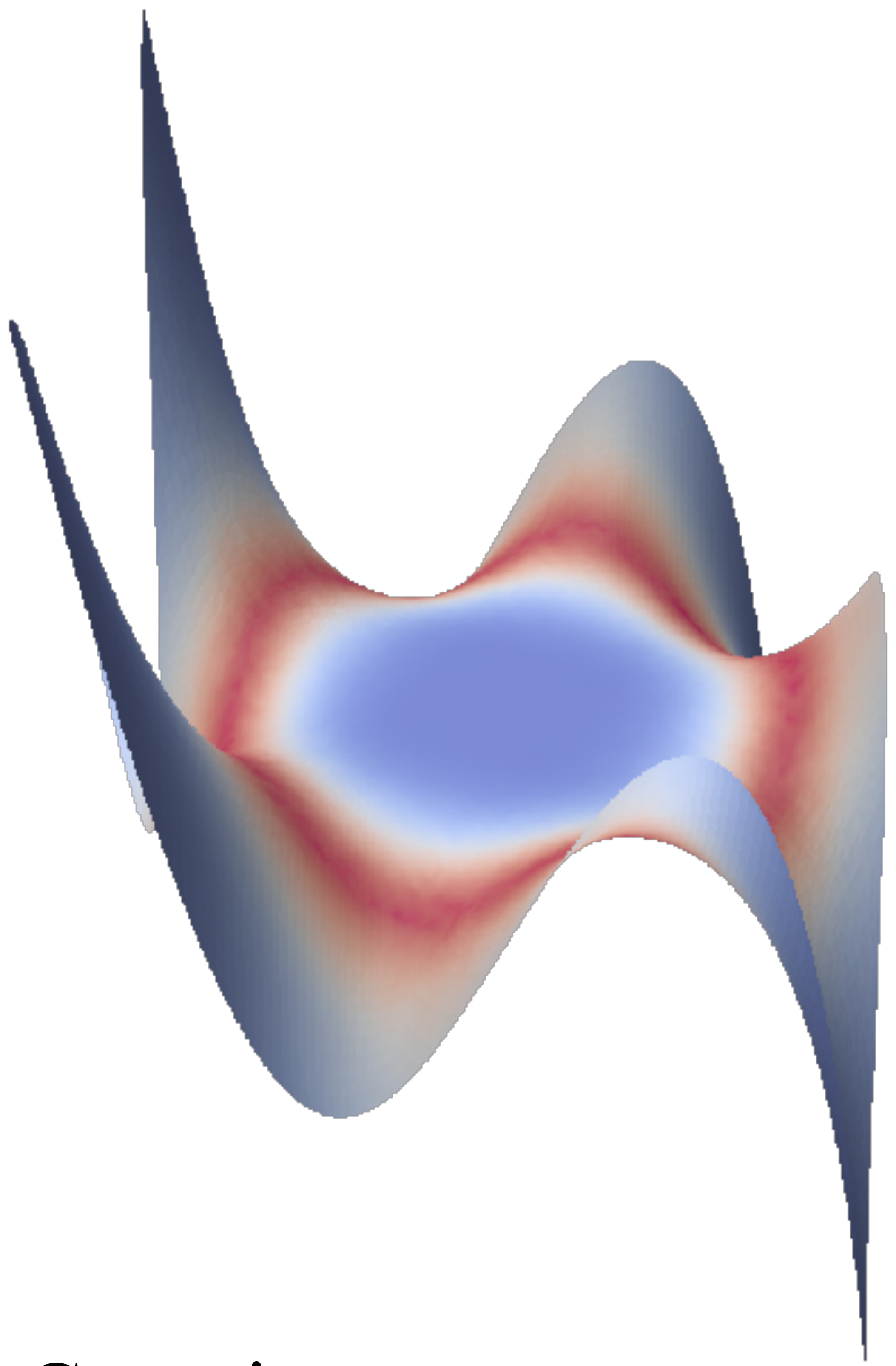
Maximal Curvature $\kappa_2 = \kappa_{max} = \max_{\phi} \kappa_n(\phi)$

* figure of normal curvature stolen from NYU lecture slides (Daniele's GP course)

Problem Statement

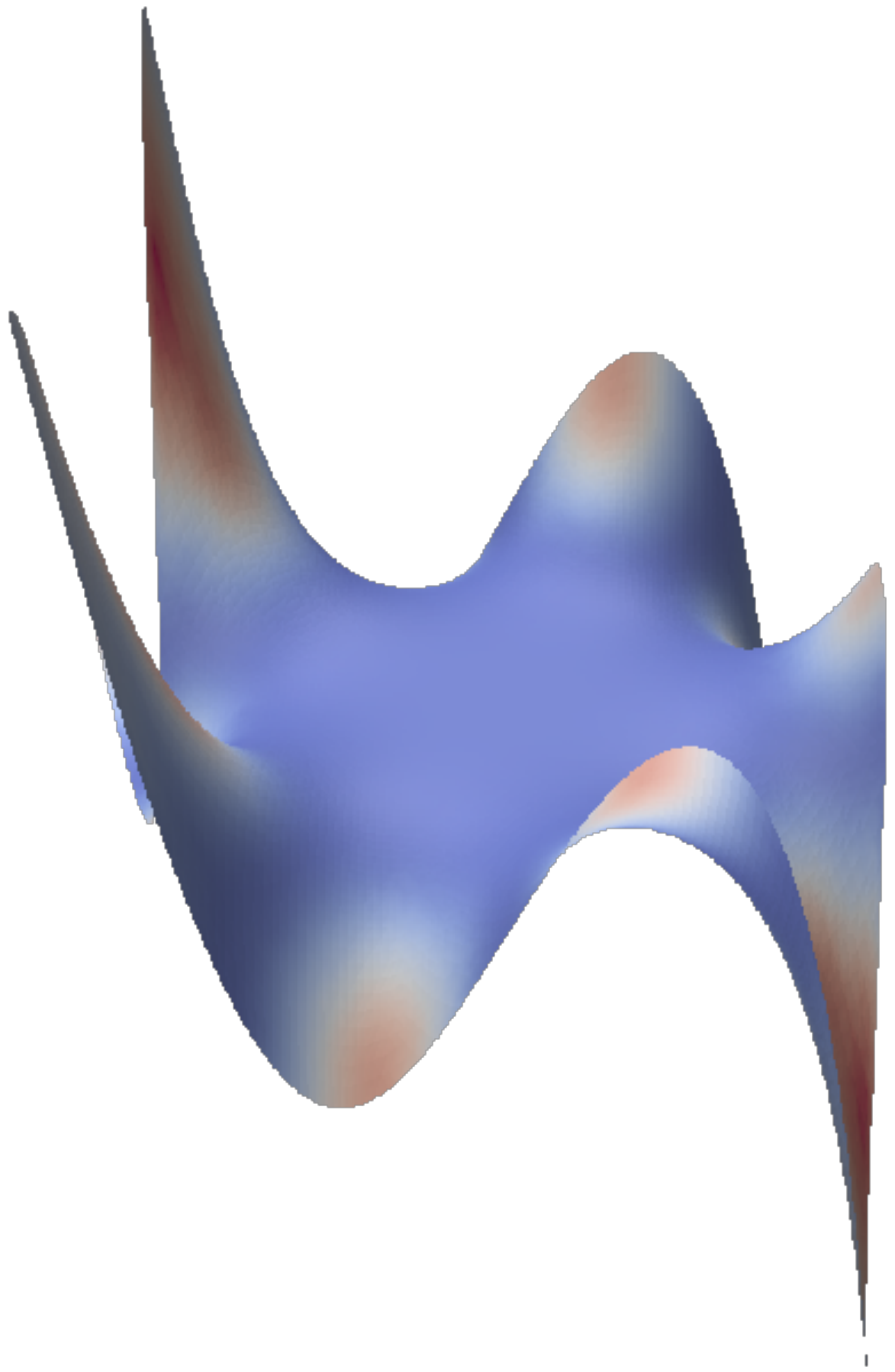
Background: Surface Curvature





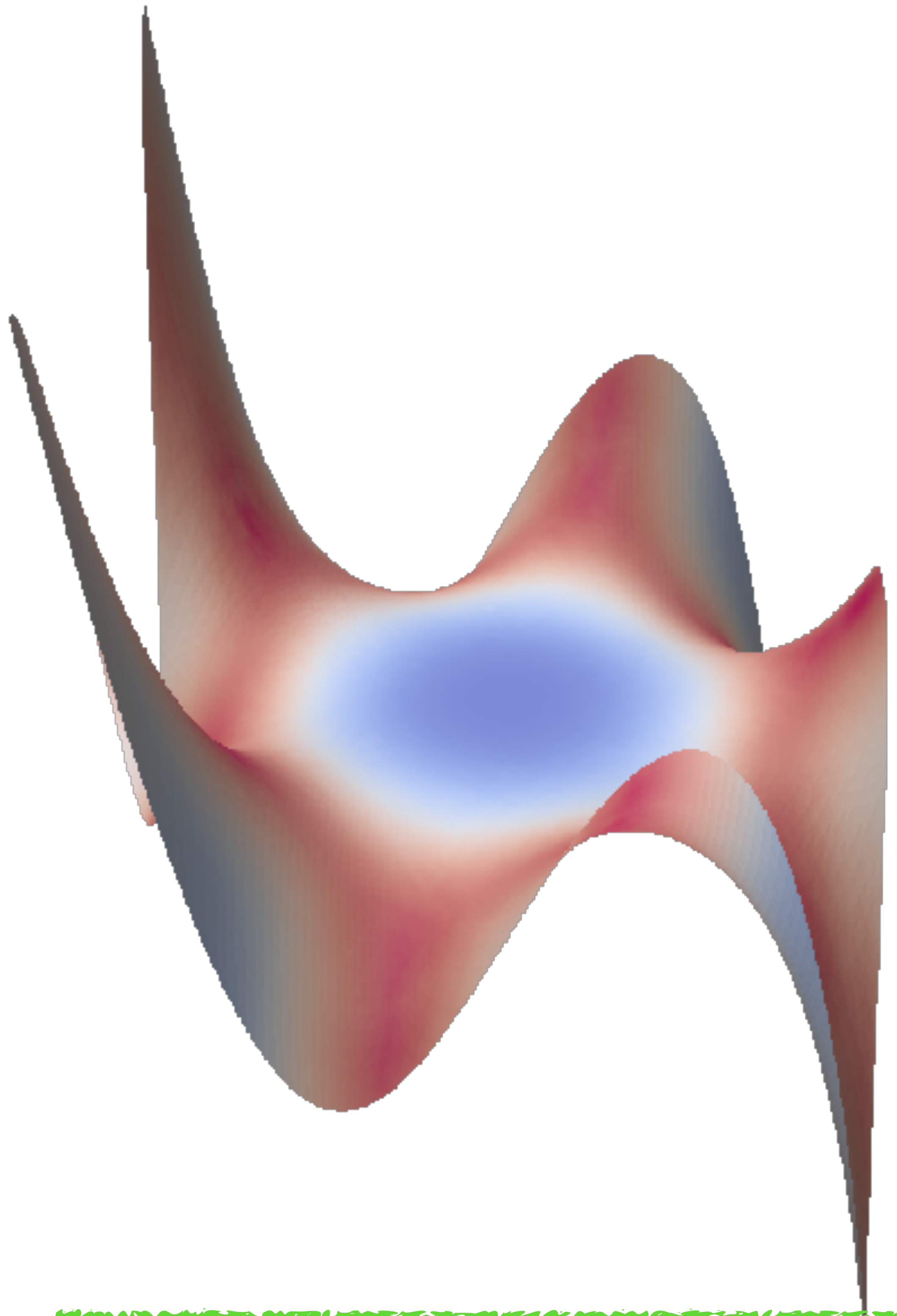
Gaussian curvature energy

$abs(K) = \|\kappa_1 \cdot \kappa_2\|$



bending energy

$$E_b = (\frac{k_1 + k_2}{2})^2$$



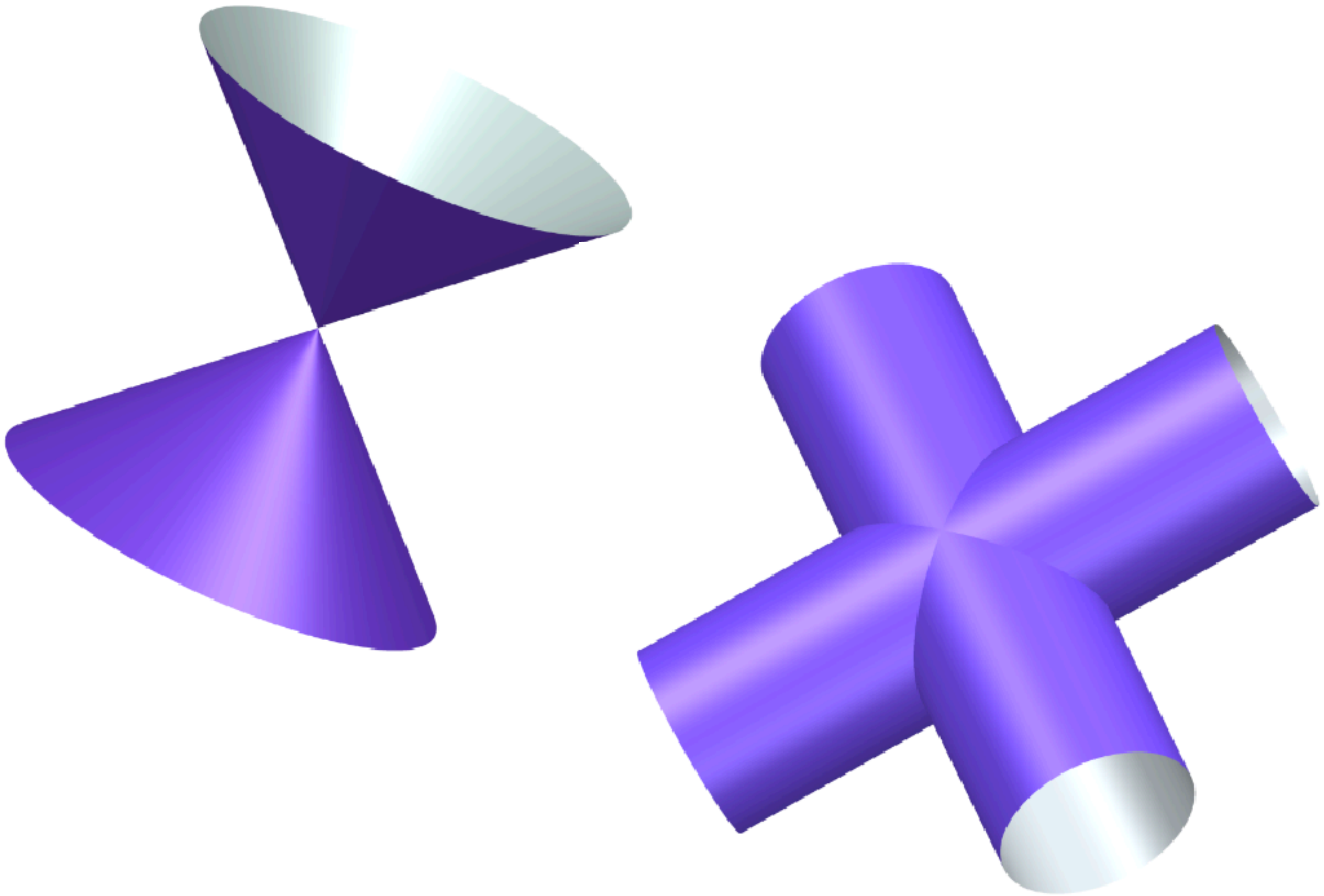
total curvature energy

$$\kappa_T = \kappa_1^2 + \kappa_2^2$$

locally, these two are not the same

Problem Statement

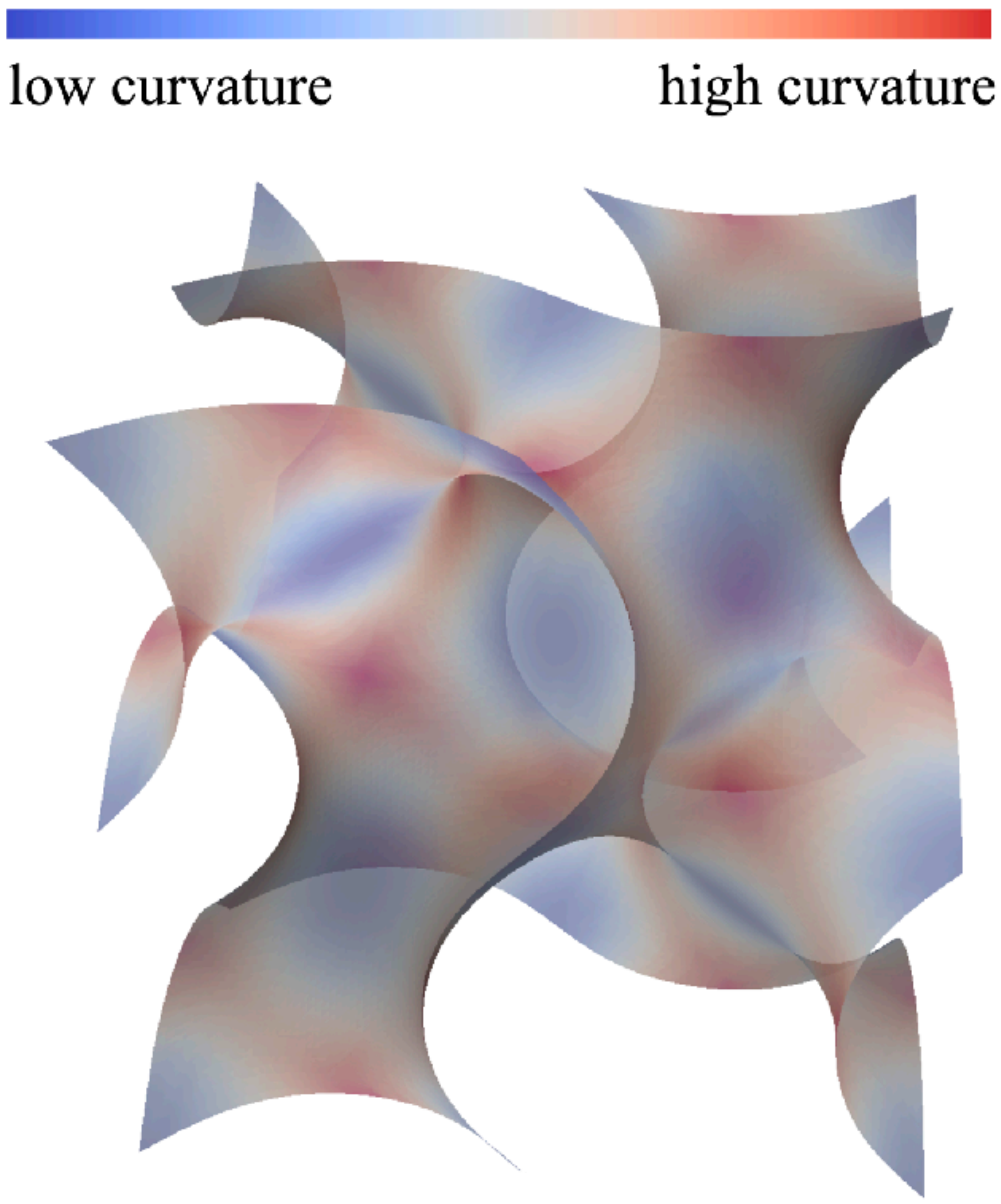
Background: Surface Curvature



Gaussian curvature vanishes on cones/cylinders

Problem Statement

Background: Surface Curvature



total curvature of the Gyroid

Mean curvature / bending energy vanishes on minimal surfaces

Total curvature is the winner, as it only vanishes on planes!

Previous Methods

Standard procedure for total curvature estimation……

1. (Fit a continuous surface.)
2. Estimate the shape operator.
3. Carry out eigen decomposition of the shape operator.
4. Take the sum of square

Previous Methods

Triangle Meshes

[Taubin 1995]

Taubin Matrix (a 3x3 matrix)

$$M_p = \frac{1}{2\pi}\int_{-\pi}^{\pi} \kappa_\theta \vec{t}_\theta \vec{t}_\theta^{\,T} d\theta$$

But we don't pre-know the normal curvatures. There is no way to accurately calculate the Taubin Matrix. Estimating the matrix is nontrivial and introduces errors.

Taubin's Observations:

Eigenvectors of the matrix are $\vec{n}$ $\vec{t_1}$ $\vec{t_2}$

Eigenvalues of the matrix are

$$\frac{3}{8}\kappa_{min} + \frac{1}{8}\kappa_{max} \qquad \frac{1}{8}\kappa_{min} + \frac{3}{8}\kappa_{max}$$

Previous Methods

Point Clouds

Using Covariance Matrix

For each sample in the point set:

- Find it's KNN
- Calculate the covariance matrix
- Perform PCA to the covariance matrix
- Normalize the eigenvalues

✓ if your samples are regularly distributed

✗ if your samples irregularly distributed

```
def compute_curvature(pcd, radius=0.5):

    points = np.asarray(pcd.points)

    from scipy.spatial import KDTree
    tree = KDTree(points)

    curvature = [ 0 ] * points.shape[0]

    for index, point in enumerate(points):
        indices = tree.query_ball_point(point, radius)

        # local covariance
        M = np.array([ points[i] for i in indices ]).T
        M = np.cov(M)

        # eigen decomposition
        V, E = np.linalg.eig(M)
        # h3 < h2 < h1
        h1, h2, h3 = V

        curvature[index] = h3 / (h1 + h2 + h3)

    return curvature
```

Previous Methods

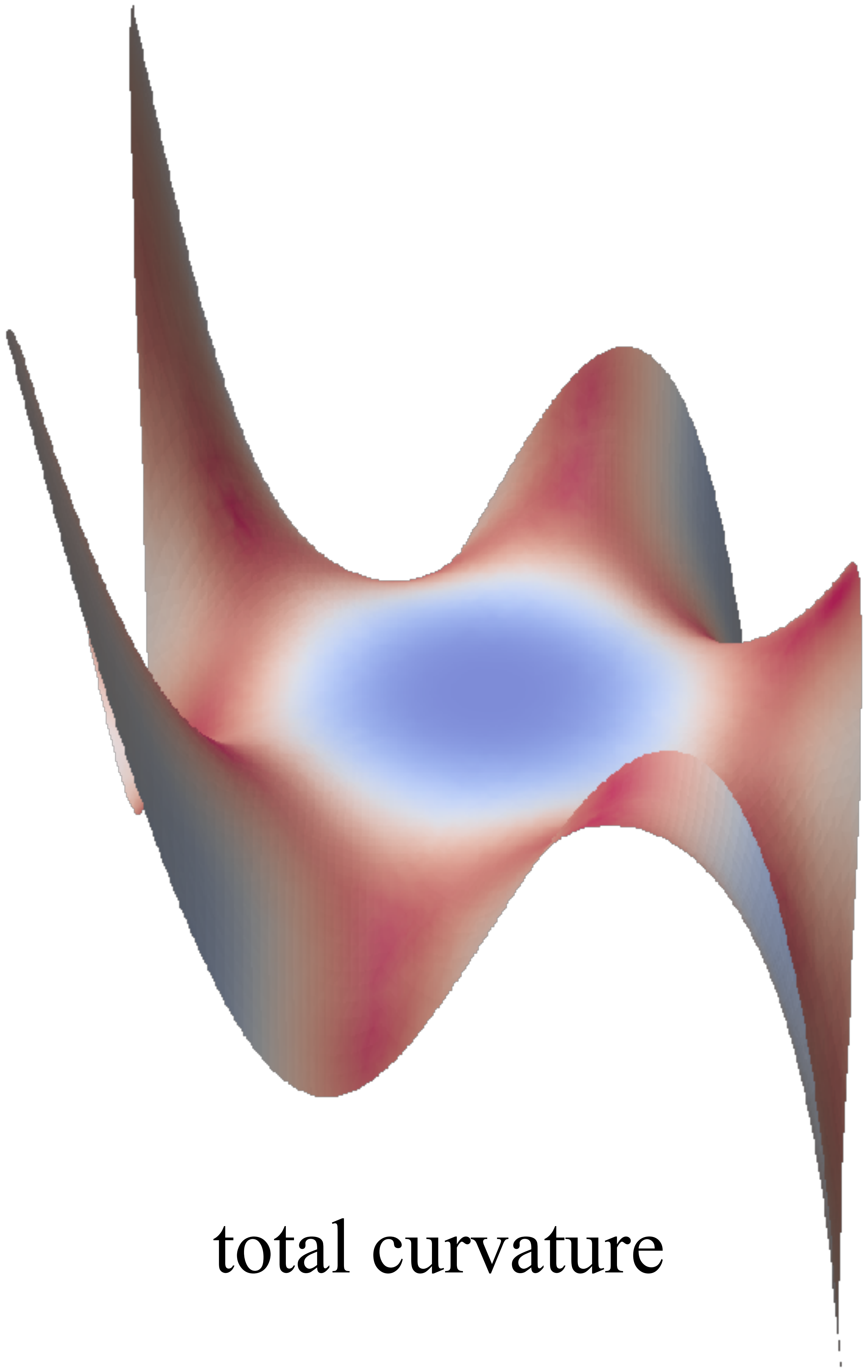
Previous methods are less desirable……

- Estimating the shape operator is error-prone.
- Normalization is non-trivial.

Our objective is simpler……

- Our goal is simpler, just the total curvature.
- We don’t really need to know the exact values of the principal curvatures.

Our Method



total curvature

$$\cancel{\kappa_T = \int_T (\kappa_1^2 + \kappa_2^2)dp}$$

$$\kappa_T = \int_T \|dN\|^2$$

Our Method

normal field (Gauss Map) gradient

Curvature can be considered as how quickly does the surface normal change.

In mathematics, Dirichlet energy is a measure of how variable a function is.

$$\kappa_T = \int_T \|\nabla N\|^2$$

Dirichlet energy of Gauss Map

Our Method

$$\kappa_T = \int_T ||\nabla N||^2$$

Dirichlet energy of Gauss Map

We love Dirichlet energy here. Because we know exactly how to calculate it, and that would be with the stiffness matrix (cotangent Laplacian).

$$\kappa_T = trace(N^T \cdot S \cdot N)$$

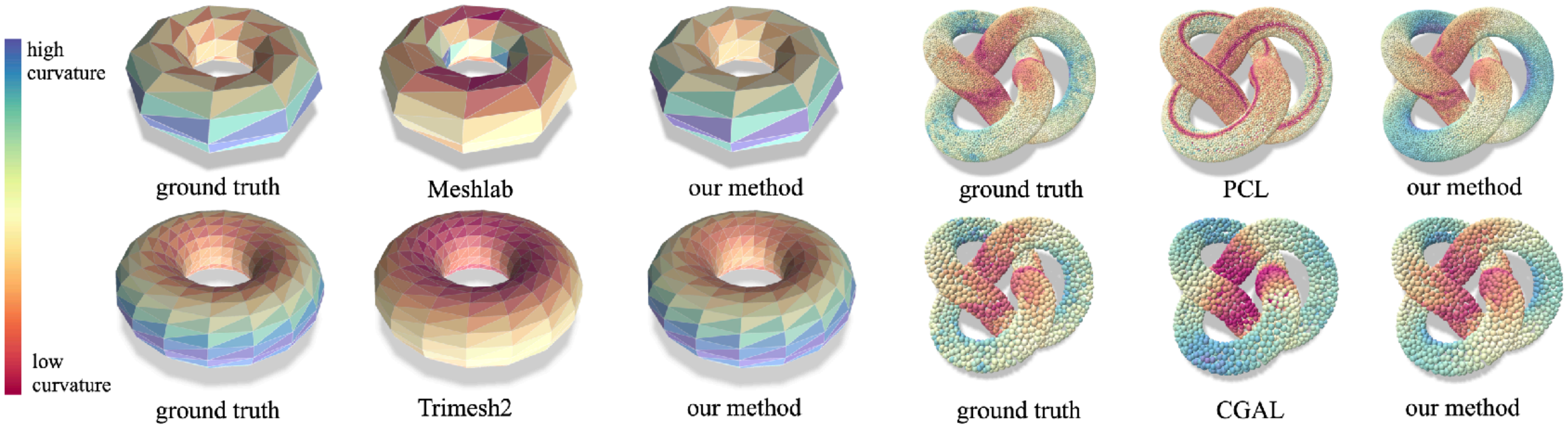
Our Method

$$\kappa_T = trace(N \cdot S \cdot N^T)$$

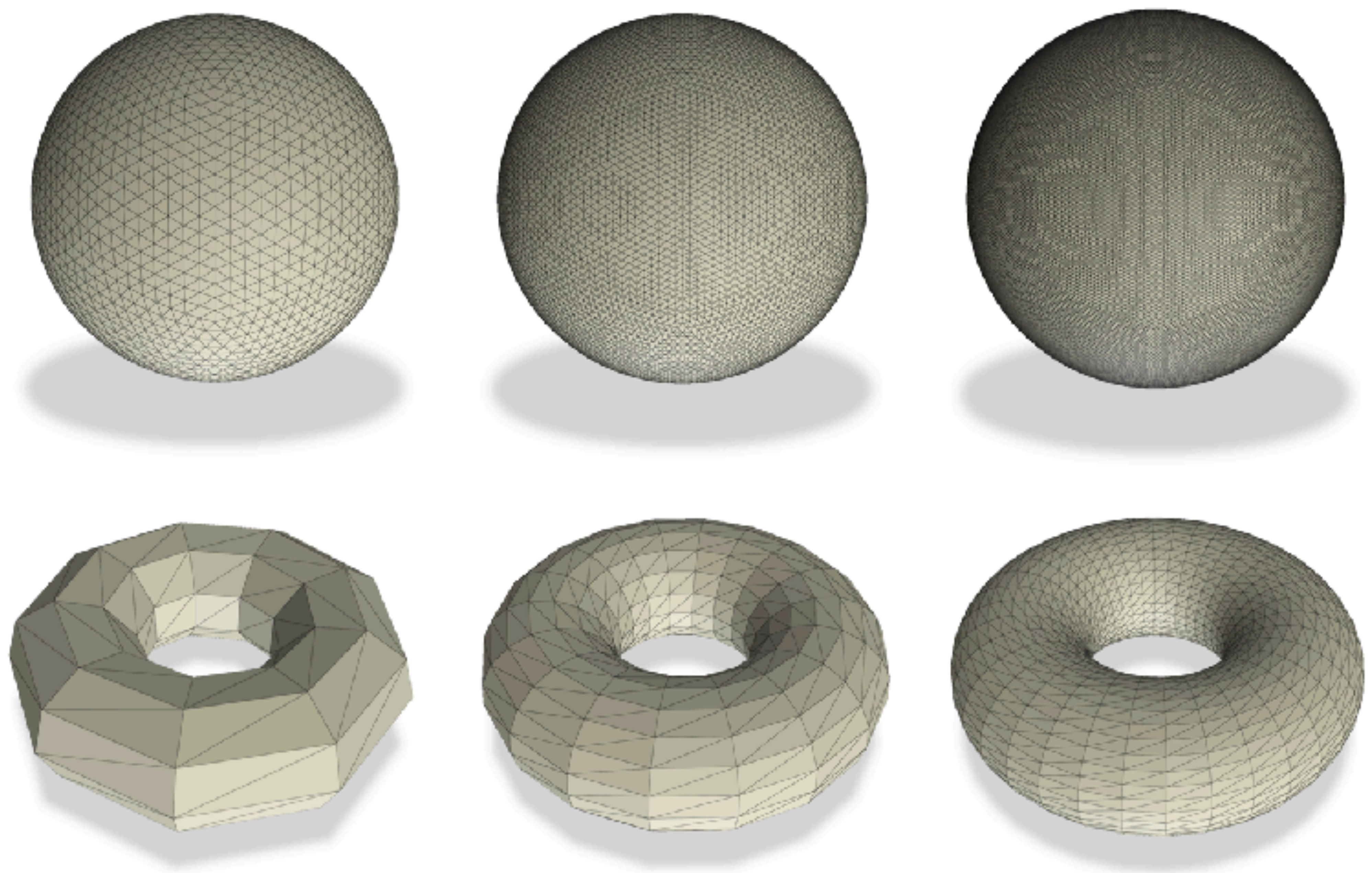
The estimation of discrete total curvature boils down to two questions:

- How to calculate the Laplacian?
- How to estimate the normal?

Performance



Performance

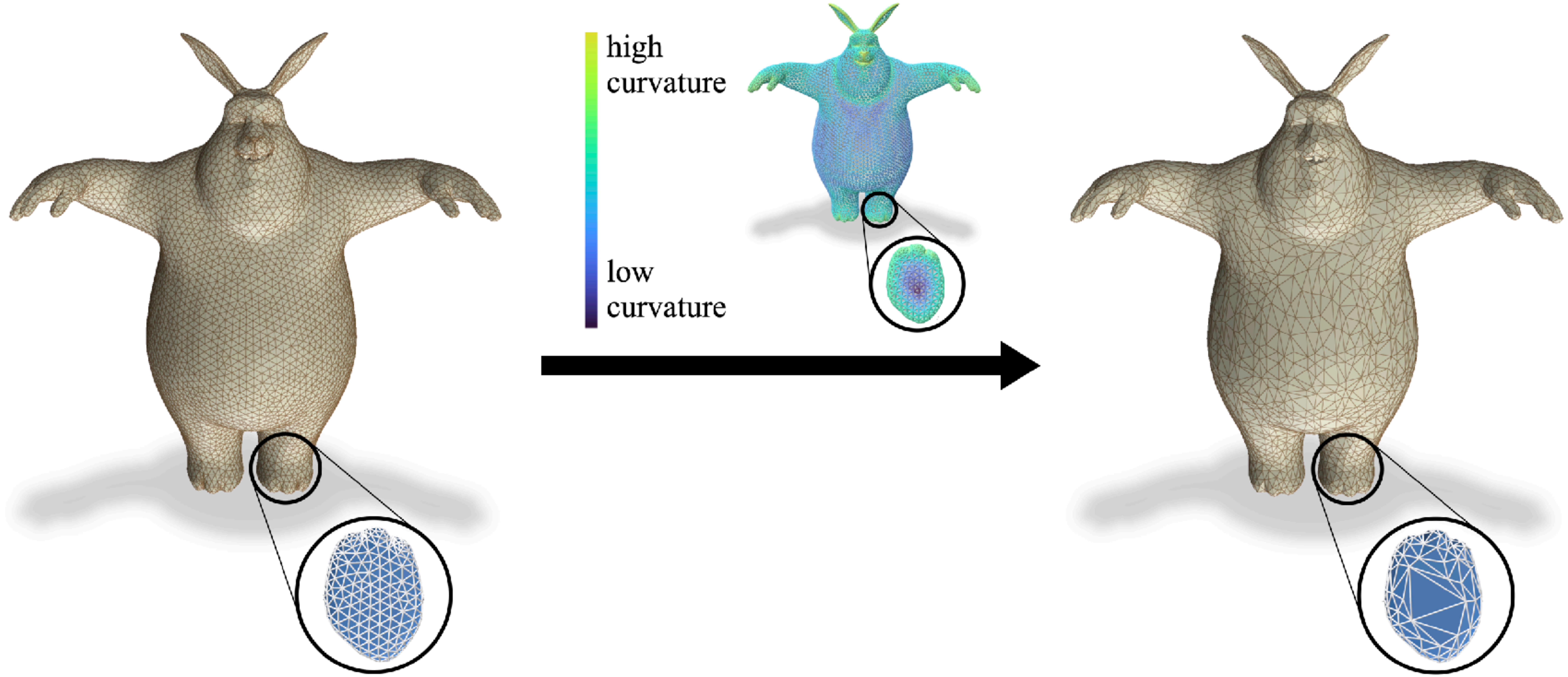


RMSE between ground truth and estimation of total curvature on regular triangulations of the sphere and torus at different resolutions.

resolution	Libigl [Panozzo et al. 2010]	Meshlab [Taubin 1995]	Trimesh2 [Rusinkiewicz 2004]	Ours
icosahedron-subdivided spheres				
4-subdivision	0.1104	0.0308	0.0155	**0.0000**
5-subdivision	0.0271	0.0353	0.0155	**0.0000**
6-subdivision	0.0067	0.0382	0.0155	**0.0000**
polyhedral torus				
9 x 9 grid	19.2708	2.5869	1.6643	**0.4759**
18 x 18 grid	3.5917	2.6976	1.1838	**0.1425**
36 x 36 grid	1.28	2.7072	1.0621	**0.0372**

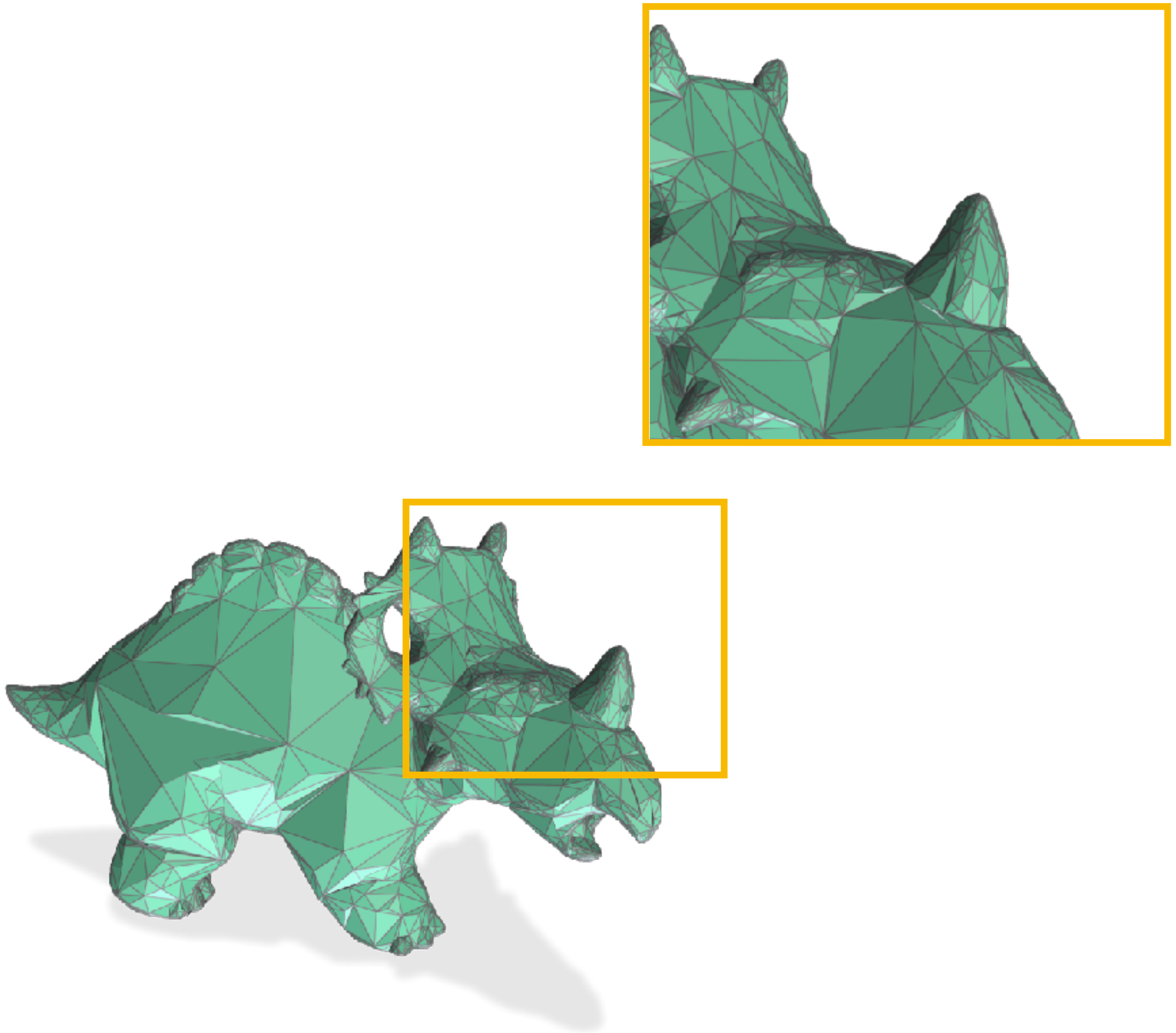
Performance

feature-preserving mesh decimation

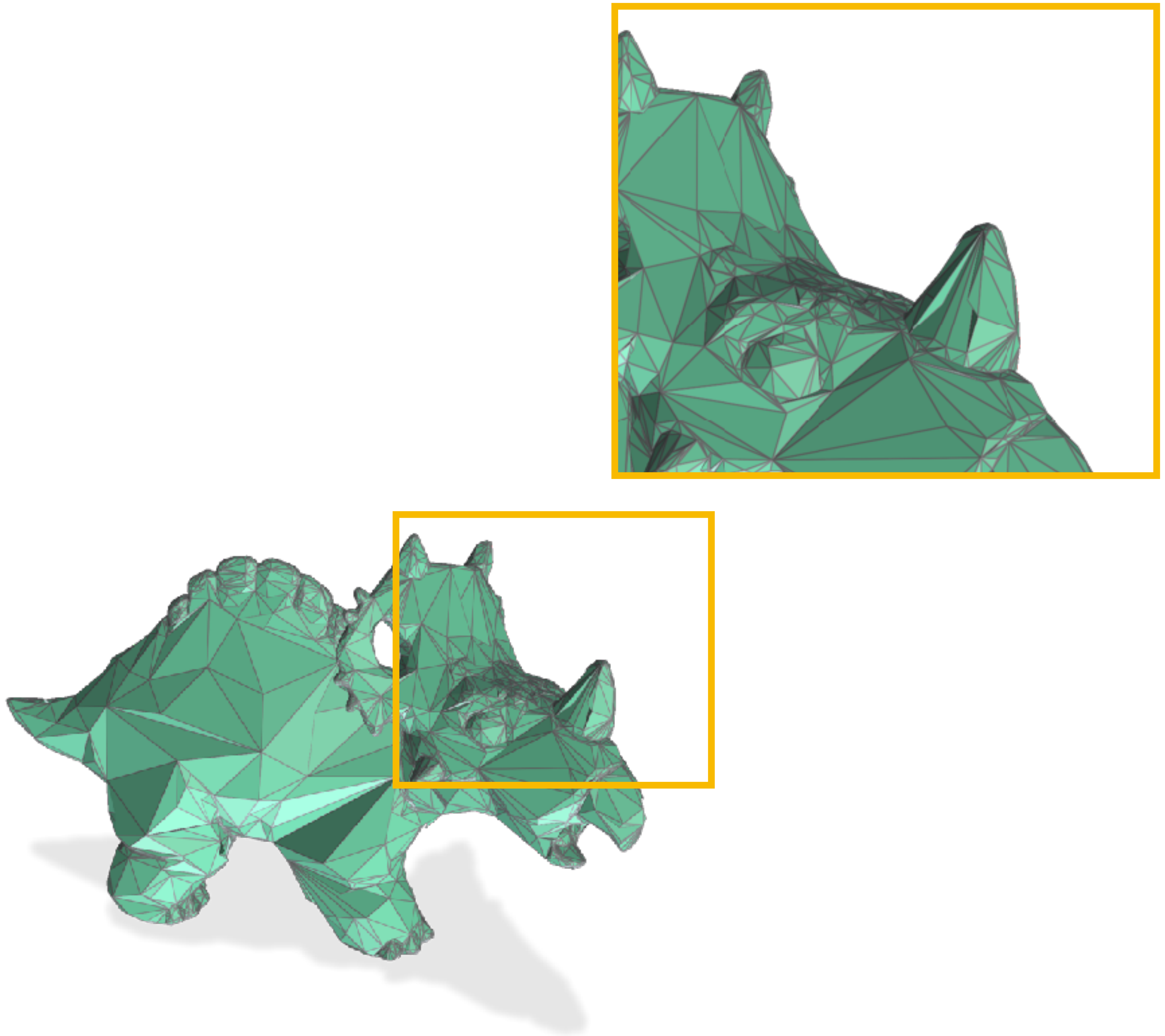


Applications

feature-preserving mesh decimation



using libigl curvature



using our curvature

Performance

feature-preserving mesh decimation

Hausdorff distance between feature-aware decimated mesh and the original mesh for the bunny (top), cow (middle), and armadillo man (bottom) models.

metric	Libigl [Panozzo et al. 2010]	Meshlab [Taubin 1995]	Trimesh2 [Rusinkiewicz 2004]	Ours
RMS	0.0066	0.0062	0.0056	**0.0054**
Max	0.0542	0.0608	0.0533	**0.0385**
RMS	0.0073	0.0071	0.0085	**0.0069**
Max	0.0731	0.0427	0.0459	**0.0385**
RMS	0.0031	**0.0027**	0.0031	**0.0027**
Max	0.0370	0.0233	0.0324	**0.0174**

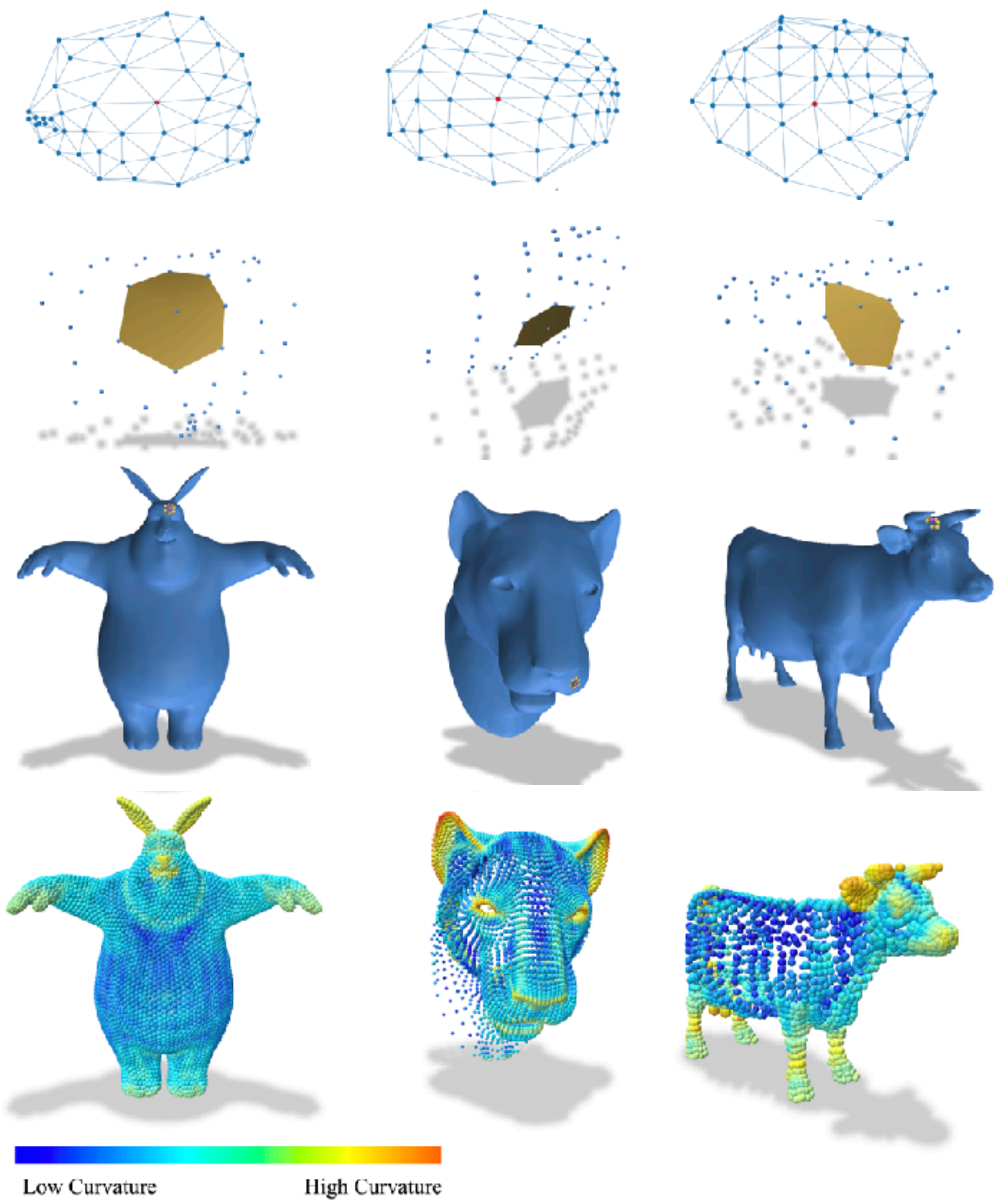
Performance

Aforementioned experiments are handling triangle meshes

To handle point clouds,

- We use Open3D to calculate the normals of the point cloud.
- We use [Belkin et. al. 2009] to calculate the Laplacian

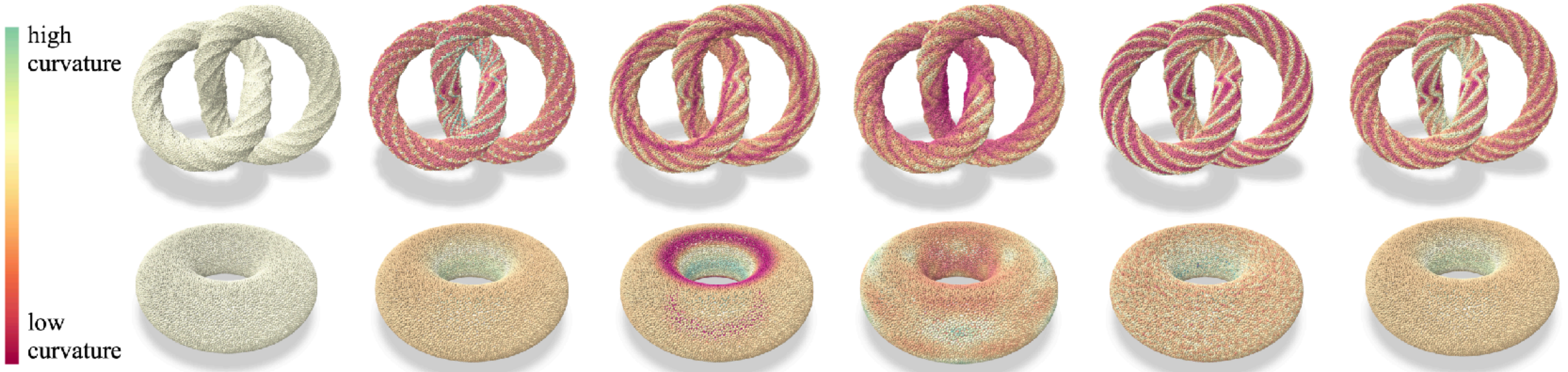
It should be noted that, local triangulation is enough for our total curvature-estimation algorithm.



Hoppe, Hugues, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. "Surface reconstruction from unorganized points." In *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, pp. 71-78. 1992.

Metzer, Gal, Rana Hanocka, Denis Zorin, Raja Giryes, Daniele Panozzo, and Daniel Cohen-Or. "Orienting point clouds with dipole propagation." *ACM Transactions on Graphics (TOG)* 40, no. 4 (2021): 1-14.

Performance



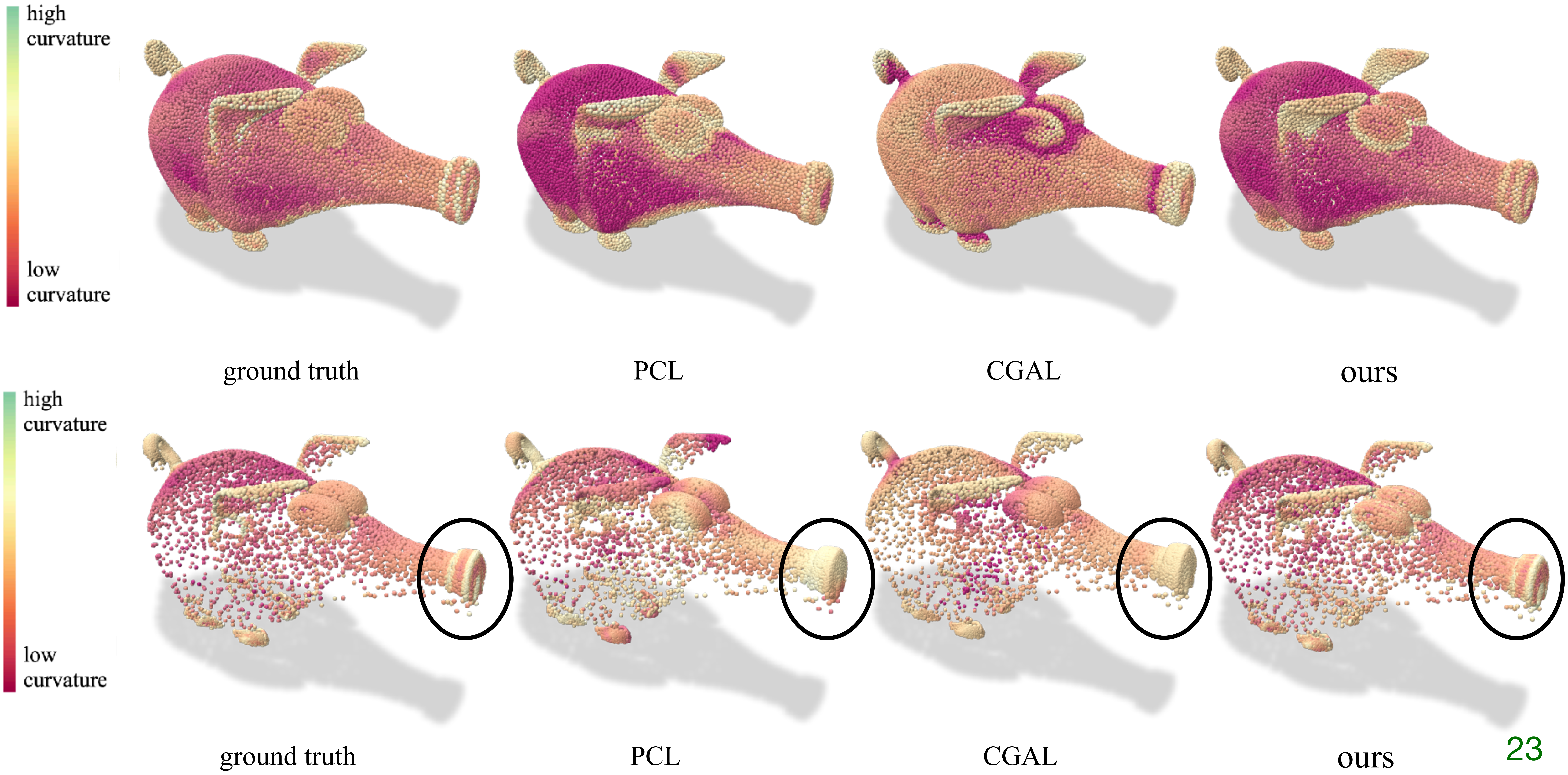
The shape GT PCL CGAL Ours (Estimated normal) Ours (GT normal)

Performance

RMSE between ground truth curvature and estimated curvature on point clouds for the knot (top) and torus (bottom) models.

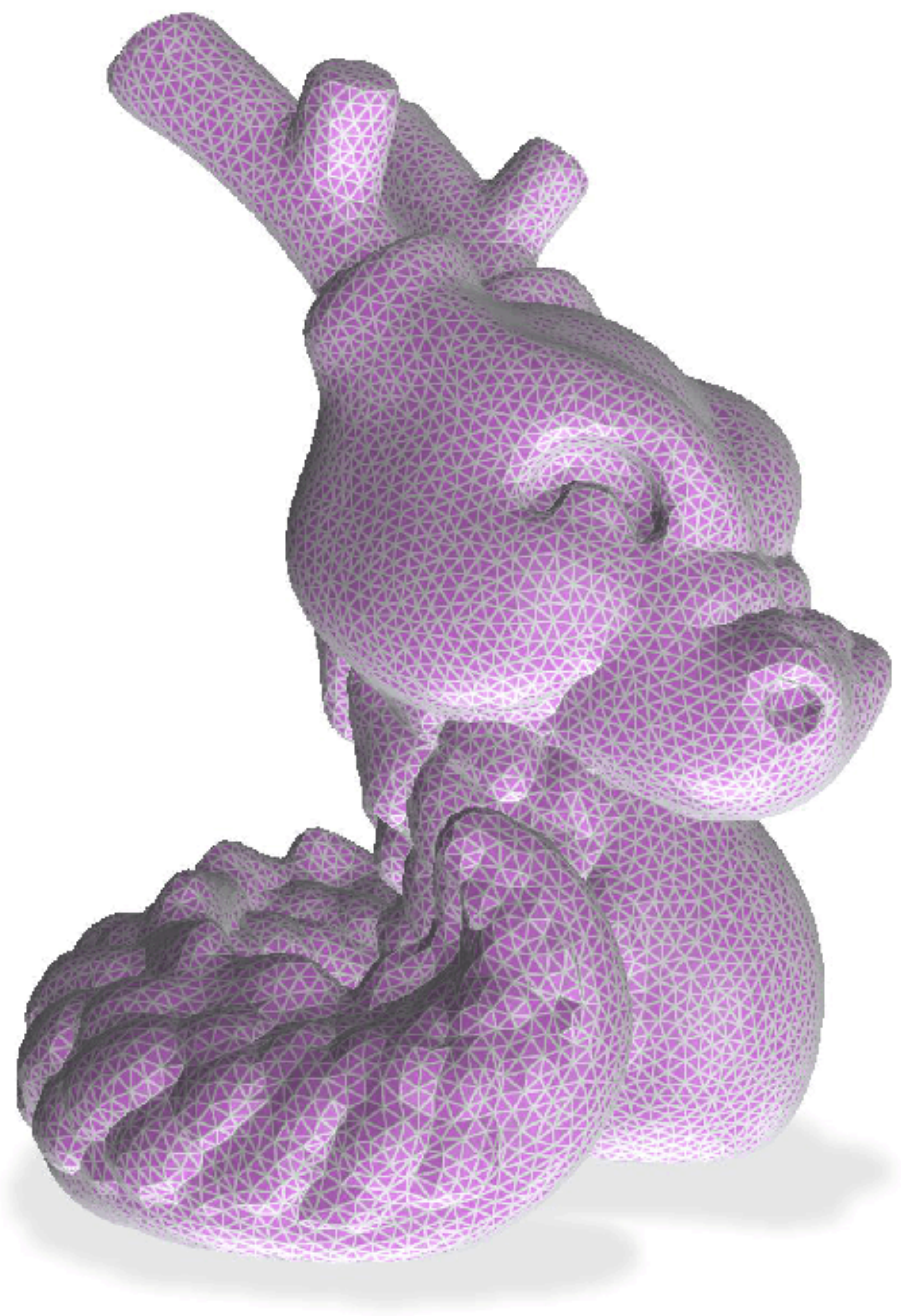
sampling	PCL	CGAL [Mérigot et al. 2010]	Ours (N est.)	Ours (N gt)
uniform	292.8847	342.6716	237.3573	**197.5912**
nonuniform	309.8654	345.6605	295.0542	**221.0447**
sparse	387.7908	438.9999	**315.5943**	315.7218
uniform	1.4364	1.9893	0.8138	**0.0219**
nonuniform	1.5057	2.0118	1.3447	**0.0367**
sparse	1.5792	2.4791	0.6501	**0.0548**

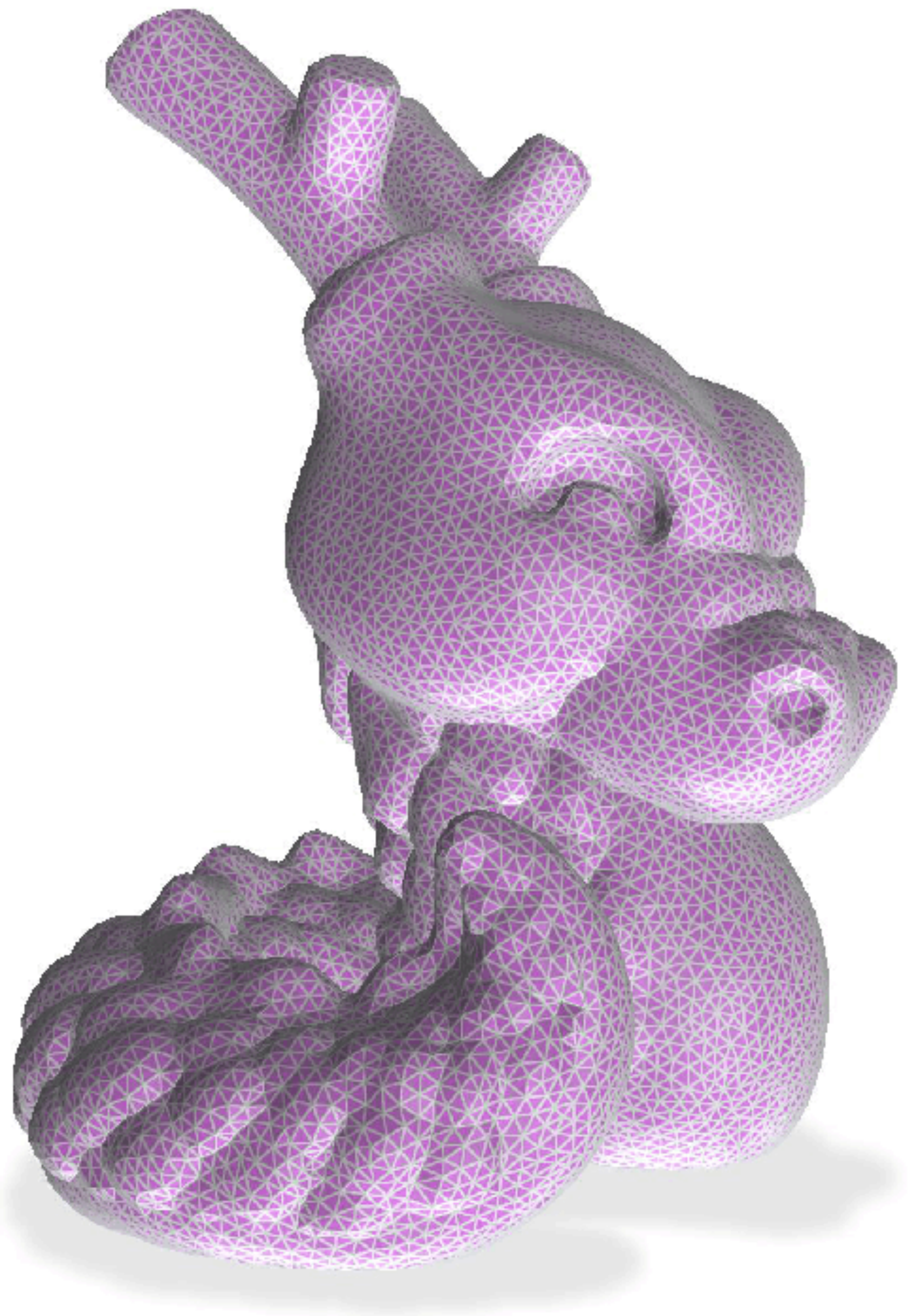
Performance



Applications

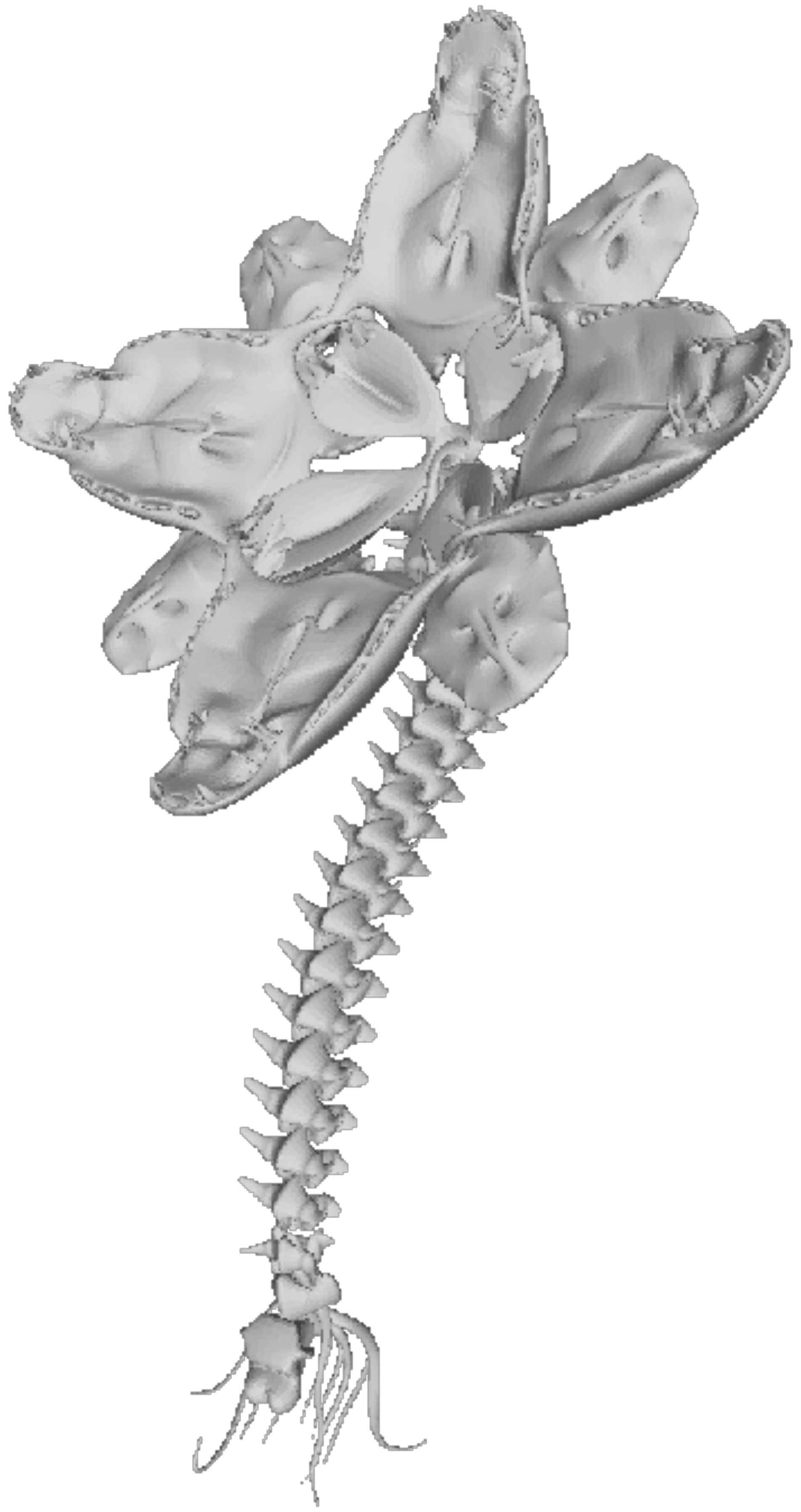
feature-preserving mesh decimation



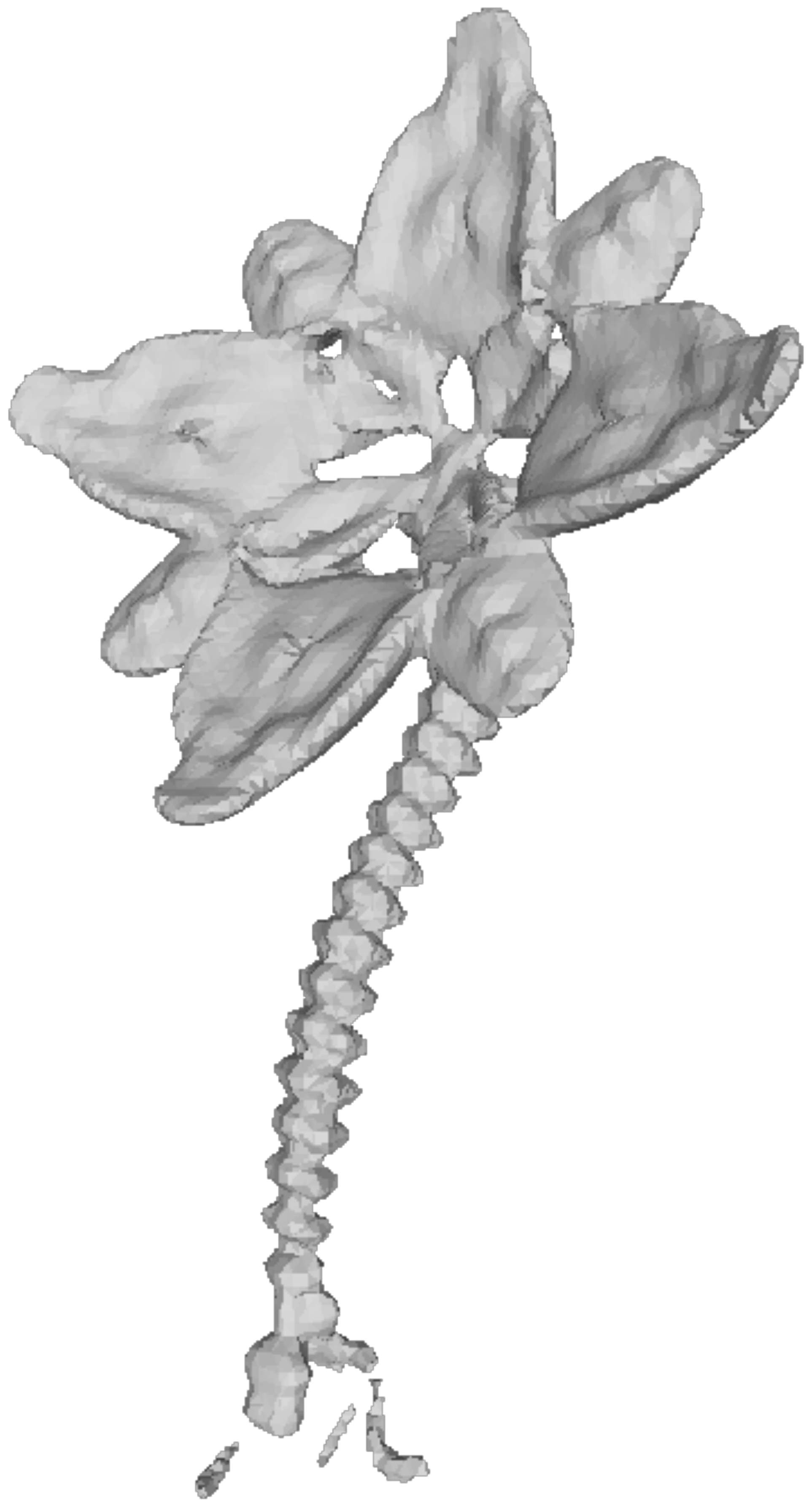


Applications

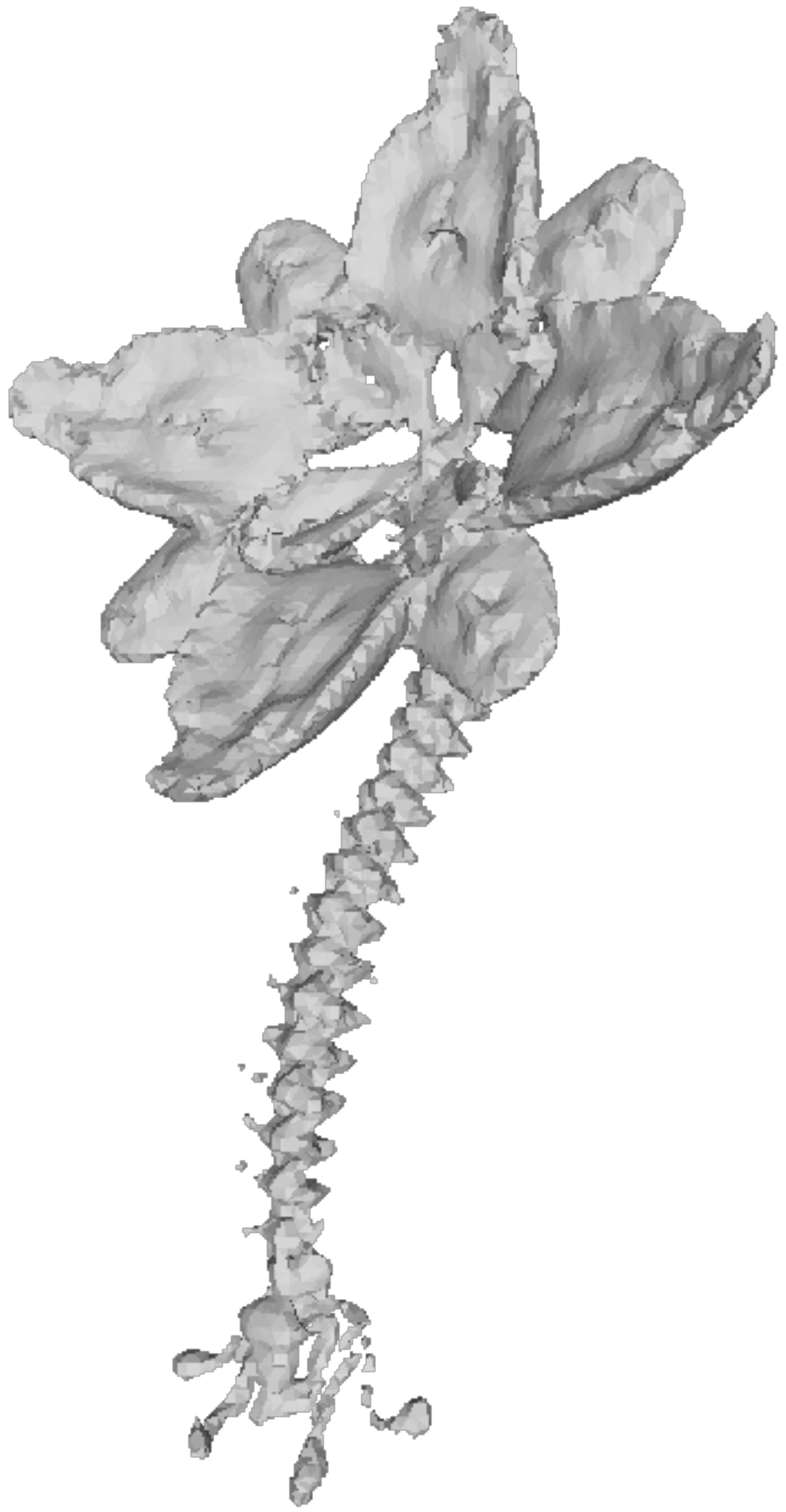
feature-preserving surface reconstruction



ground truth



Screened PoissonRecon



Screened PoissonRecon with
curvature-modulated weights

Applications

feature-preserving surface reconstruction

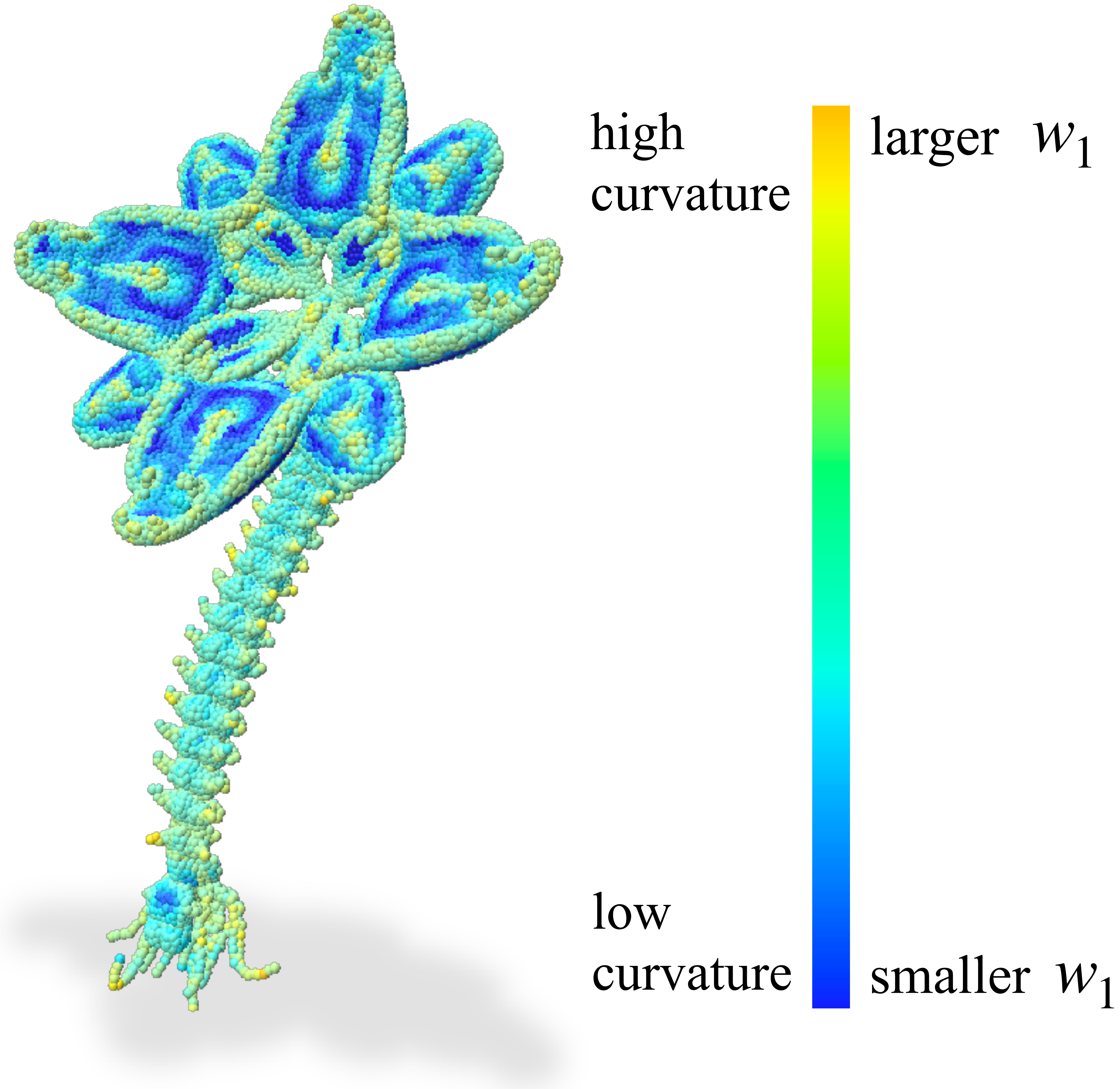
Screened PoissonRecon

Input: oriented point cloud

Output: watertight surface

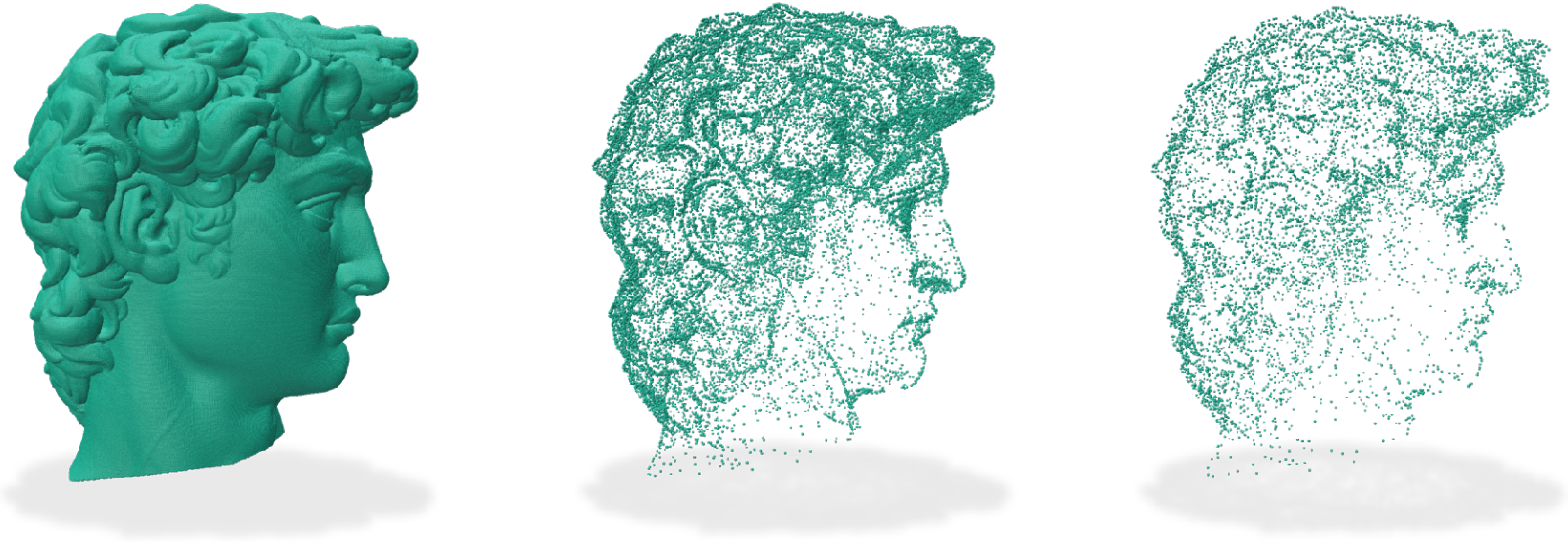
The energy:

$$E(\chi) = \underbrace{\int \|\vec{V}(p) - \nabla\chi(p)\|^2 dp}_{\text{normal/gradient fit}} + w_1 \underbrace{\sum_{p\in P} \chi(p)^2}_{\text{value fit}}$$



Applications

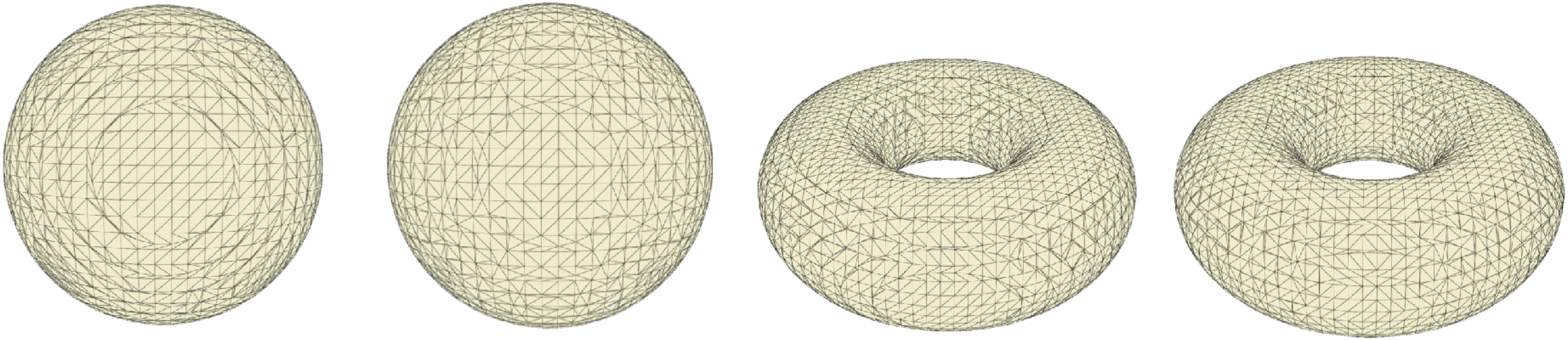
feature-preserving point cloud simplification



all the points | 10 percent of the points | 3 percent of the points

Challenges & Opportunities

skinny triangles



before edge flipping | after edge flipping | before edge flipping | after edge flipping

Sharp, Nicholas, and Keenan Crane. "A laplacian for nonmanifold triangle meshes." *Computer Graphics Forum*. Vol. 39. No. 5. 2020.

Challenges & Opportunities

skinny triangles

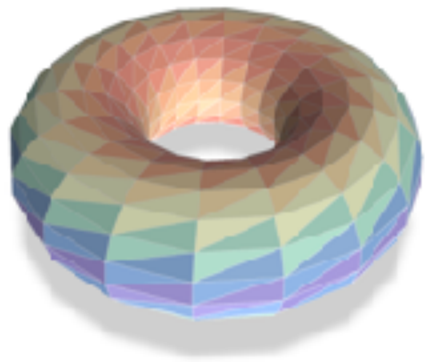
mesh	min	max	RMSE
sphere (estimated normal)			
before edge flipping	1.148	108.992	10.025
after edge flipping	1.538	25.621	6.178
sphere (ground truth normal)			
before edge flipping	2.000	2.000	0.000
after edge flipping	2.000	2.000	0.000
torus (estimated normal)			
before edge flipping	0.813	510.986	62.996
after edge flipping	1.978	423.171	49.722
torus (ground truth normal)			
before edge flipping	2.592	2.592	0.832
after edge flipping	8.724	8.724	0.754

Summary

✓ A **simple** method

Meshlab

construct Taubin matrix → eigen decomposition → derive principal curvatures from eigenvalues → calculate total curvature



TotalCurvature Calculator

Cotangent Laplacian → matrix multiplication from linear algebra

Gauss map → matrix multiplication from linear algebra

✓ Can calculate total curvature for both **triangle meshes** and **point clouds**

✓ Can benefit multiple **applications**, including **decimation** and **reconstruction**

Software

libigl — style source code

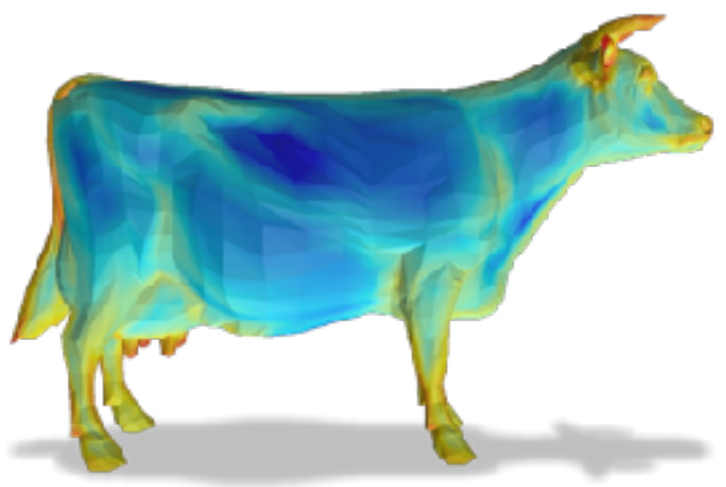
Dependencies:

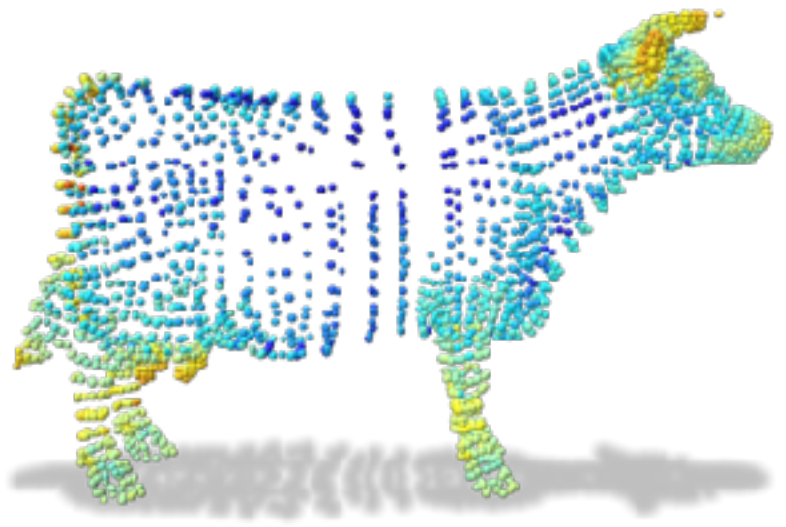
libigl OpenMP

Compilation verified on:

macOS Ubuntu

```
if (format == "mesh"){
  igl::total_curvature_mesh(V, F, N, k_S);
  VisTriangleMesh(k_S, V, F);
}
if (format == "point_cloud"){
  igl::total_curvature_point_cloud(V, N, k_S, 20);
  VisPointCloud(k_S, V);
}
```





OPEN3D — style source code

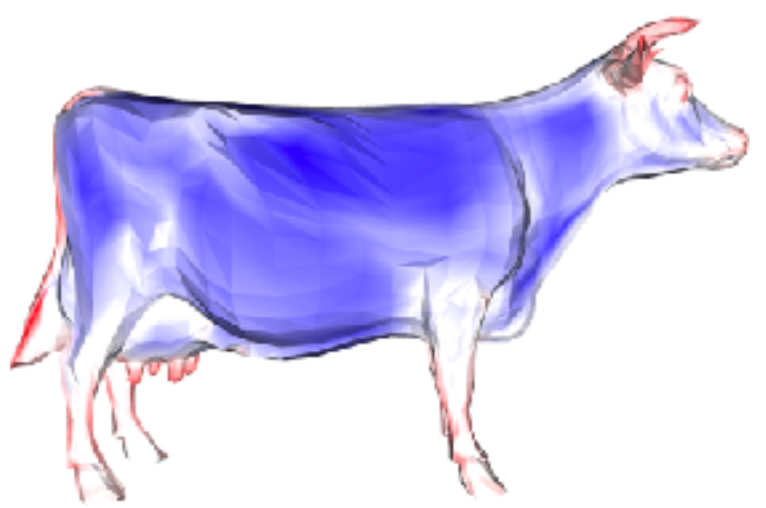
Dependencies:

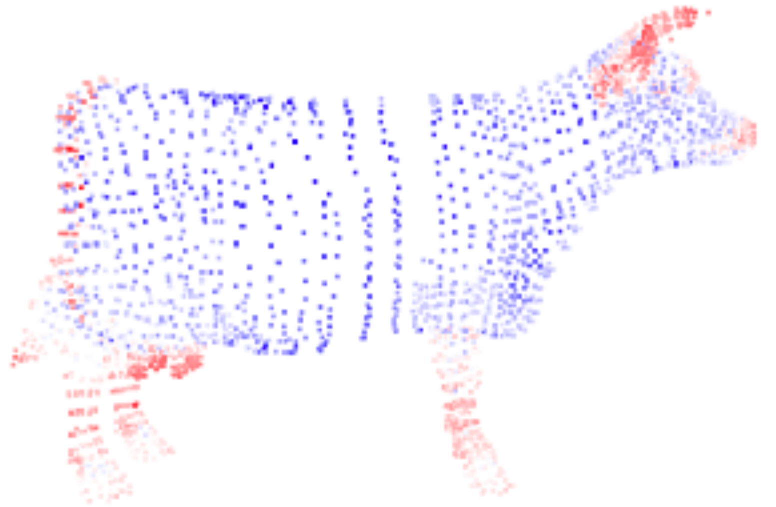
OPEN3D OpenMP

Compilation verified on: macOS

```
// calculate total curvature on triangle mesh
open3d::geometry::TotalCurvature::TotalCurvatureMesh(V, F, N, k_S);

// calculate total curvature on point cloud
open3d::geometry::TotalCurvaturePointCloud::TotalCurvaturePCD(V_PCD, N_PCD, k_S_PCD, 20);
```





Thank You!