Quadric Error Metrics for Variational Reconstruction and Learnable Shape Representation

Pierre Alliez

Inria center at Université Côte d'Azur

Joint work with…

TONG ZHAO

TAMY BOUBEKEUR

DAVID COHEN-STEINER

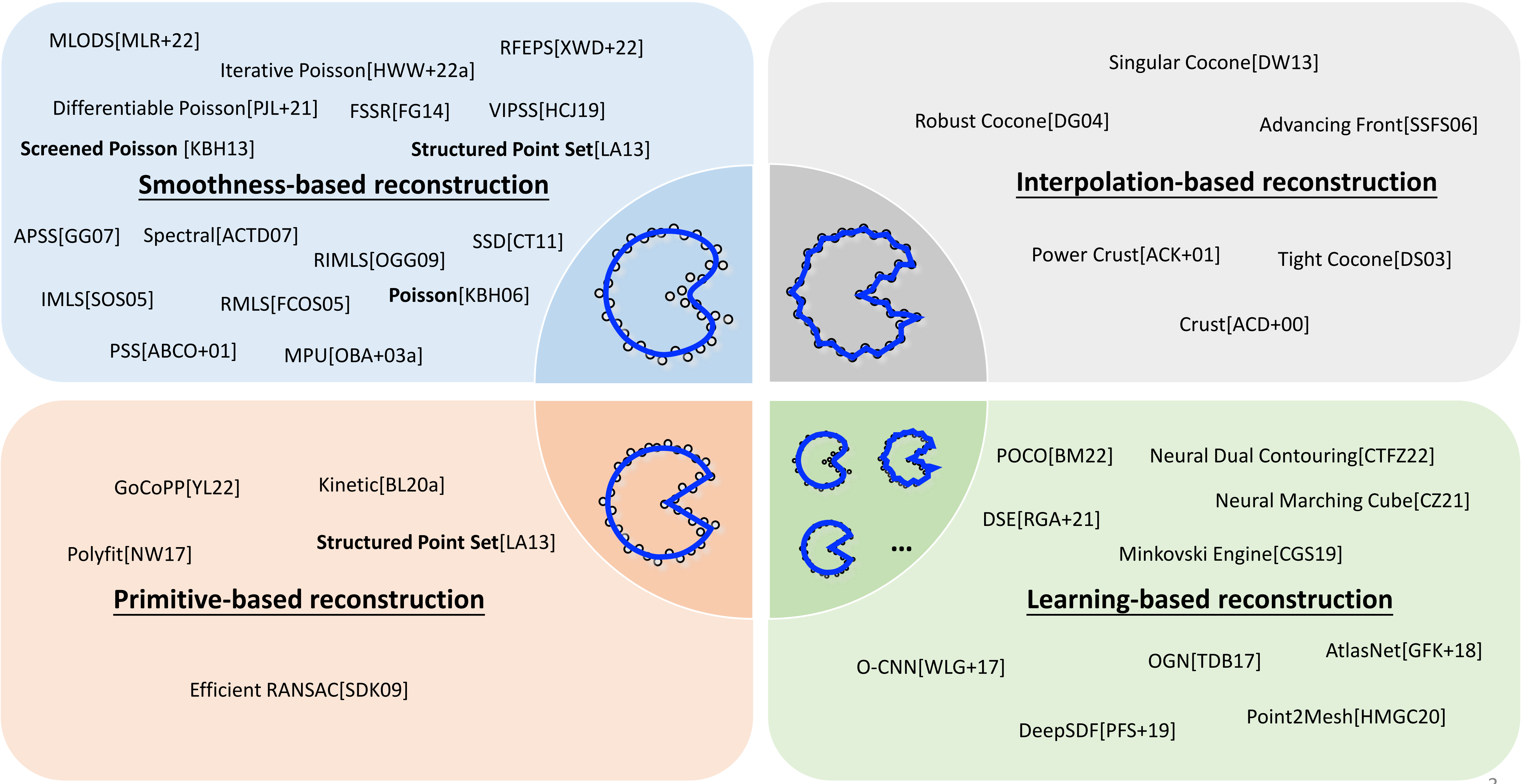
JEAN-MARC THIERY

LAURENT BUSÉ

Inria

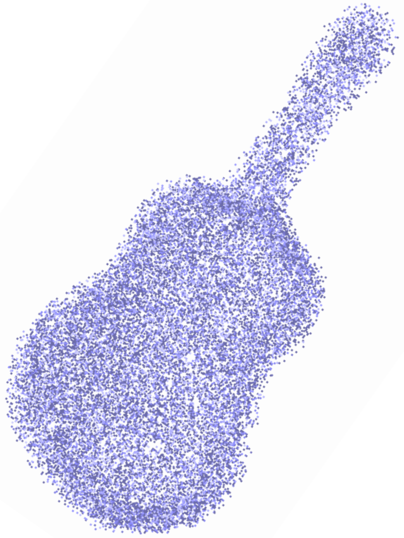
Adobe Research

Surface Reconstruction

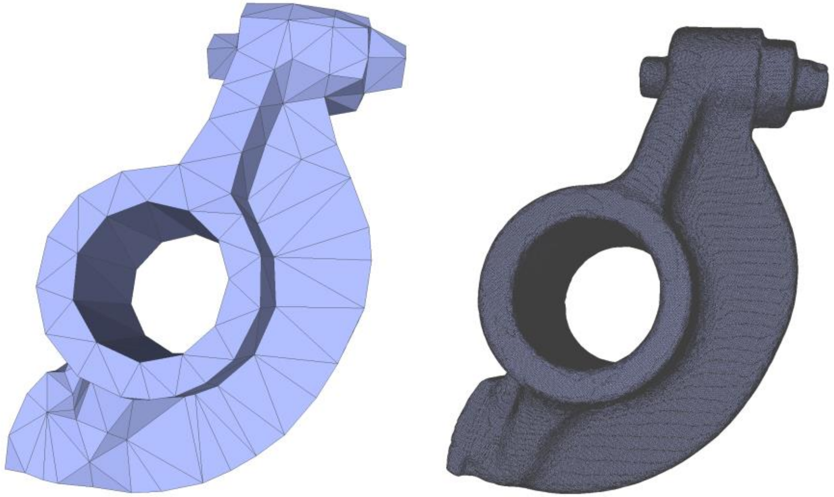


Challenges

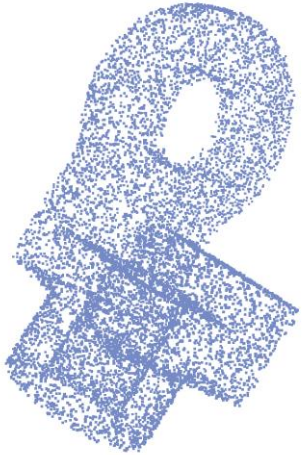
Missing data



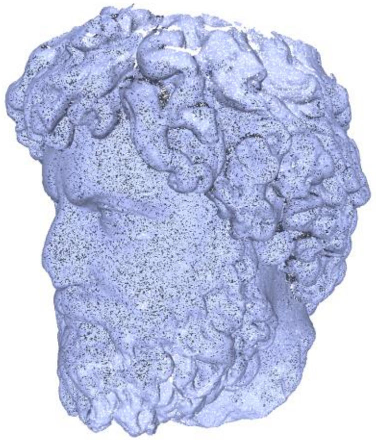
Noise



Complexity-distortion tradeoff



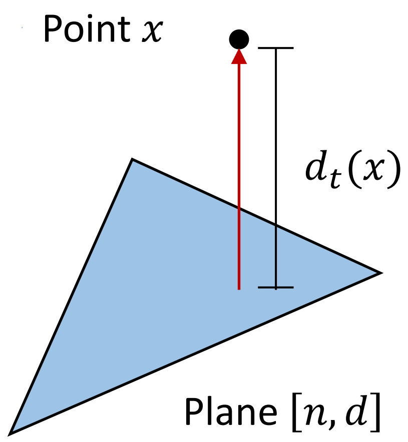
Sharp features



Multi-scale features

...

Quadric Error Metric



Squared distance to plane

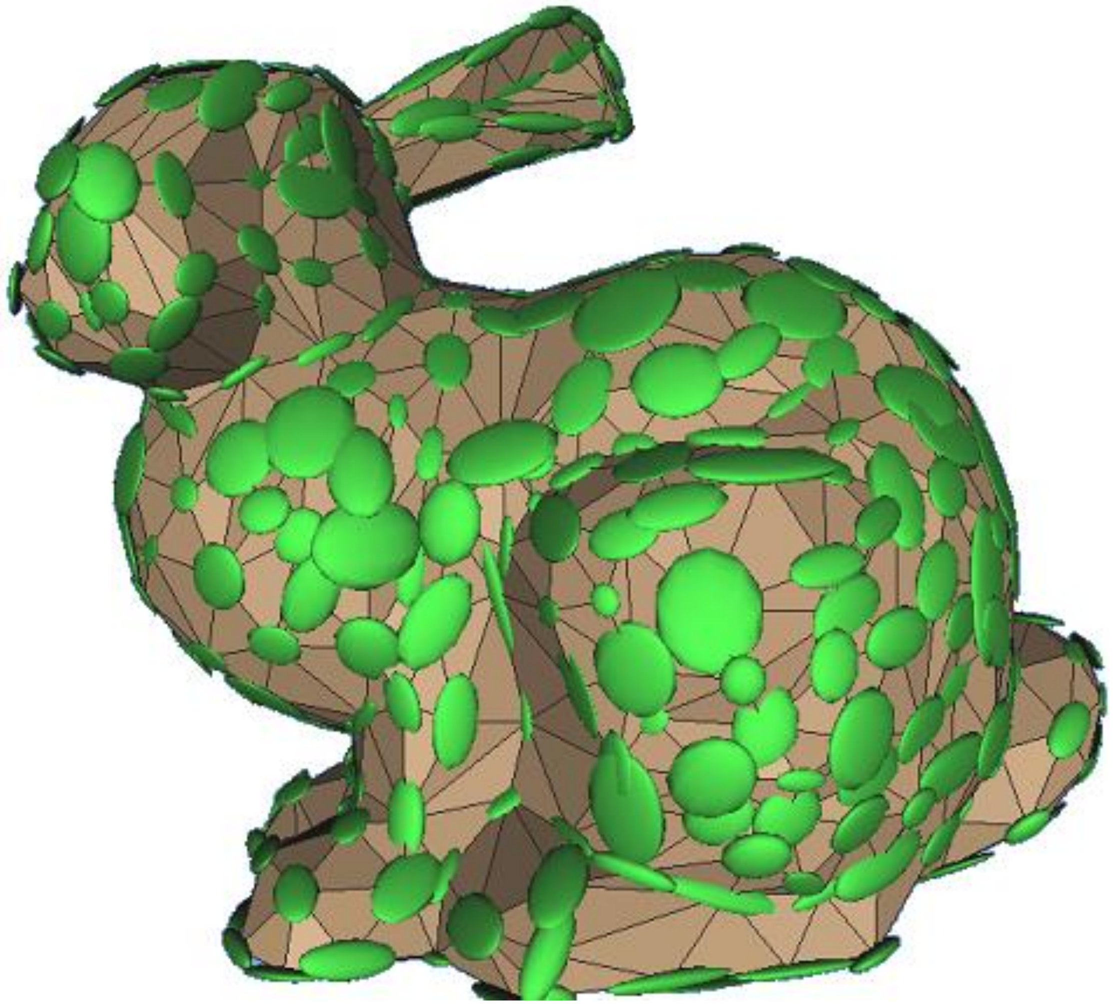
$$p = (x, y, z, 1)^T, \quad q = (a, b, c, d)^T$$

$$dist(q, p)^2 = (q^T p)^2 = p^T (q q^T) p =: p^T Q_q p$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & b^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

[Garland, Heckbert 97]

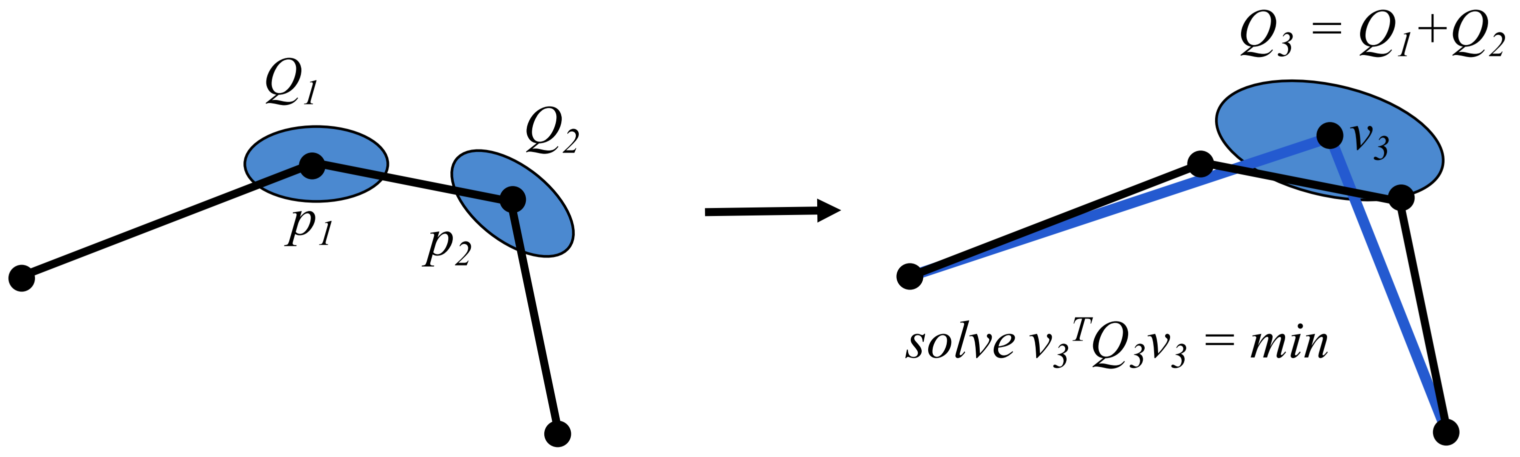
Quadric Error Metrics

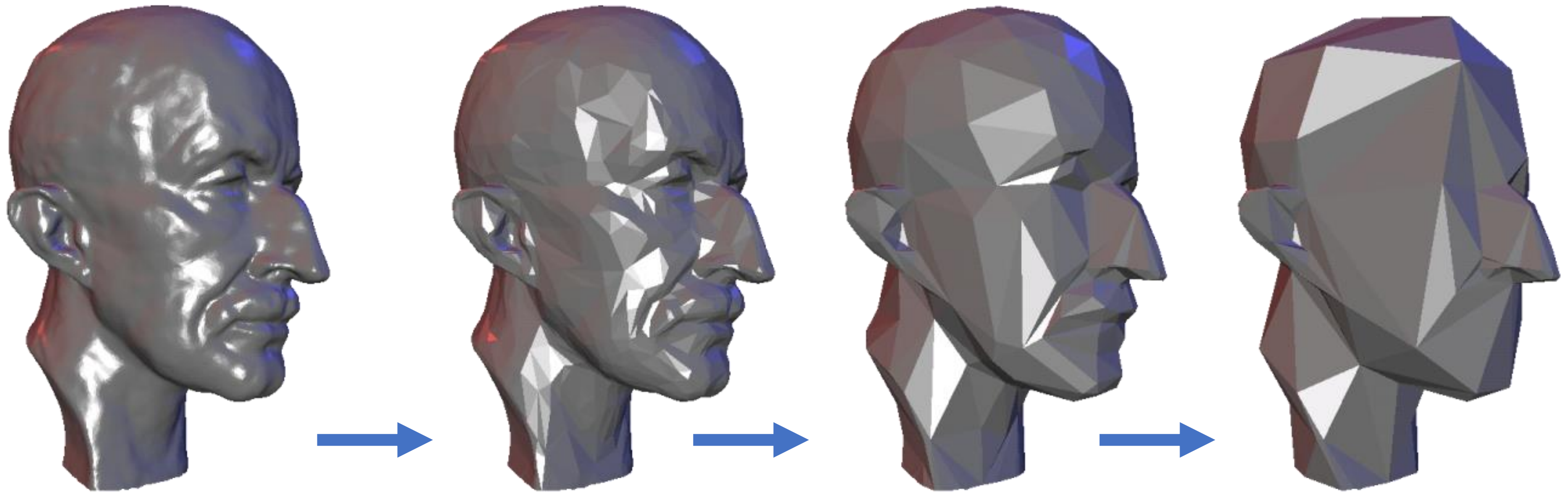


Ellipsoid = error isolevel

[Garland, Heckbert 97]

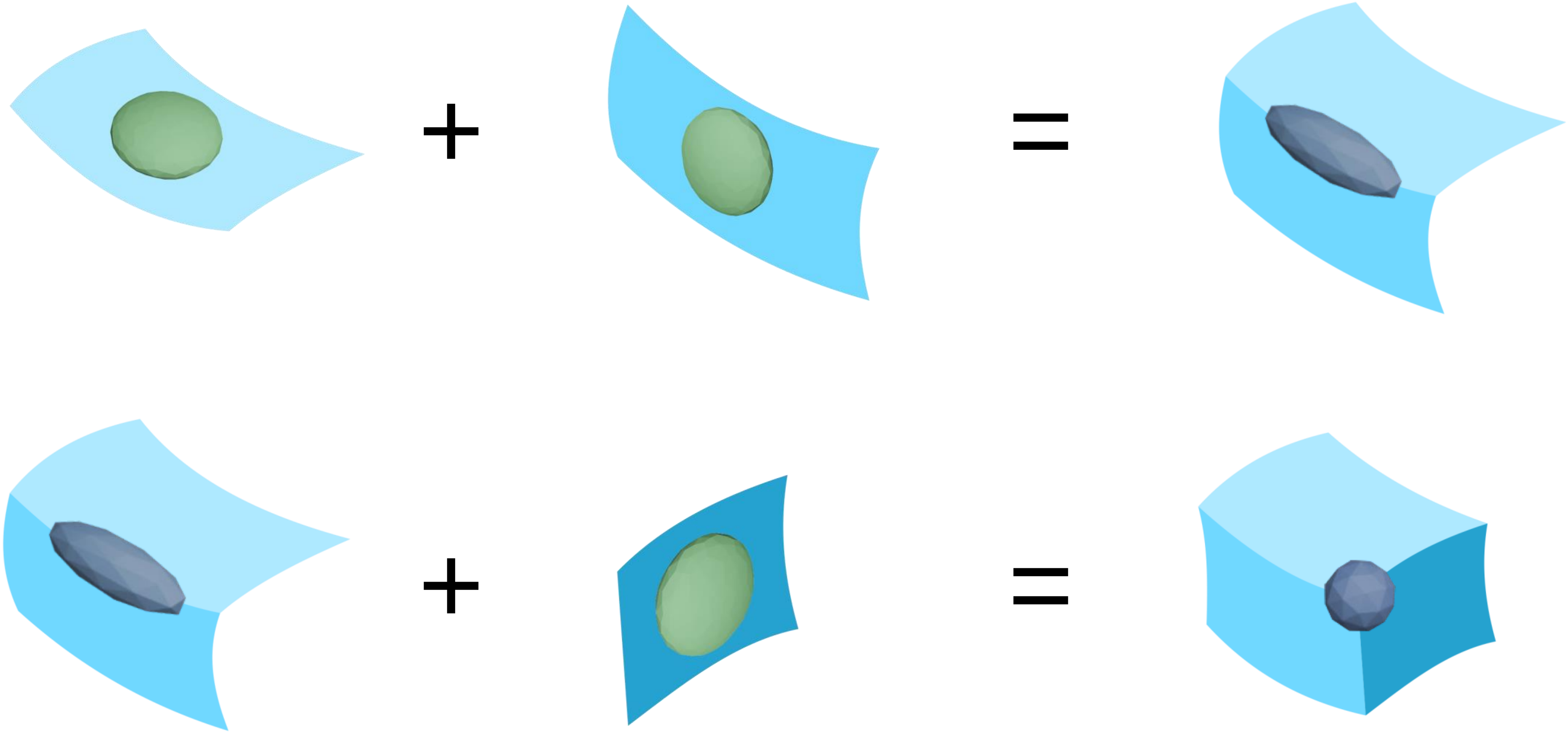
Quadric Error Metrics for Mesh Decimation



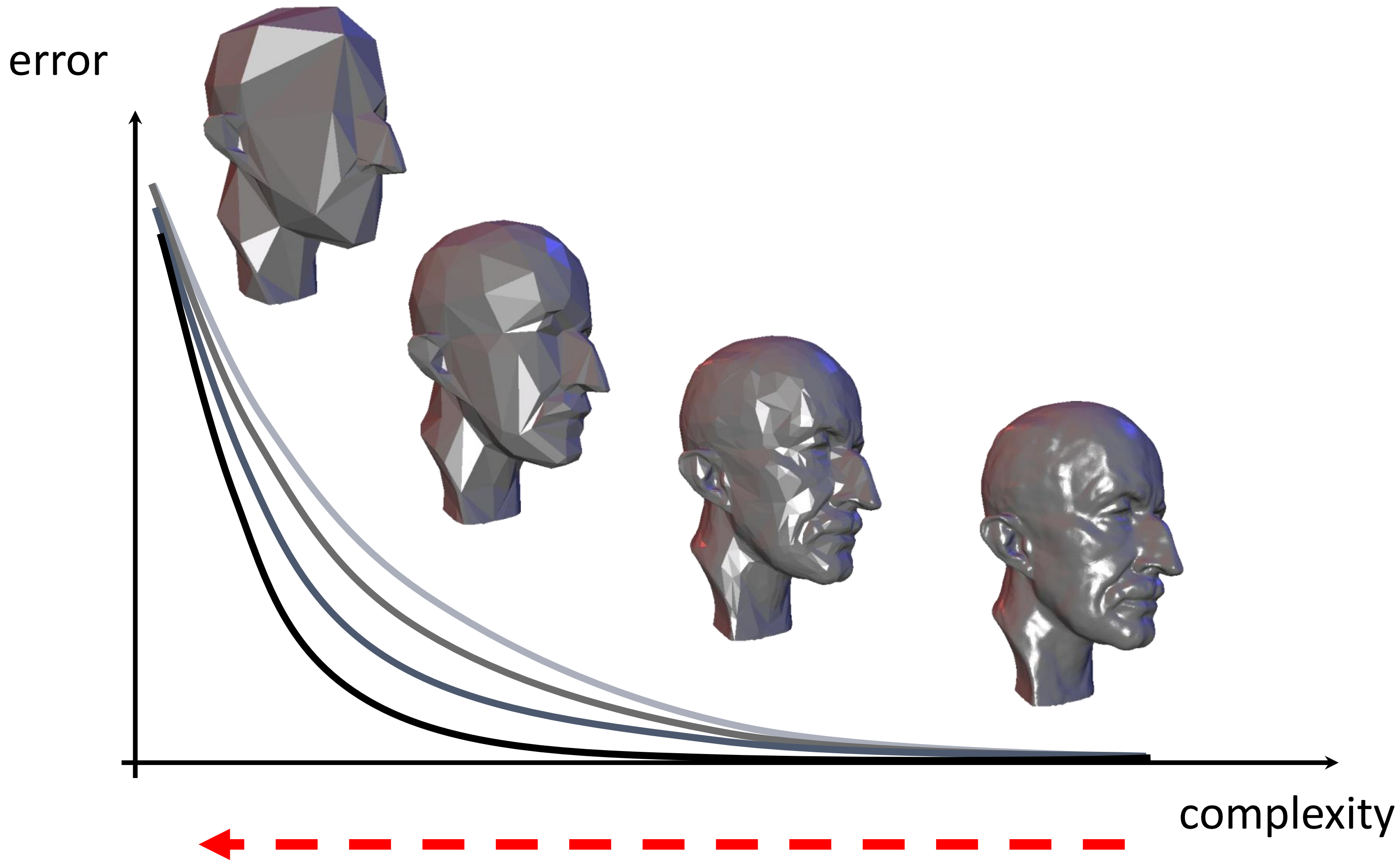


[Garland, Heckbert 97]

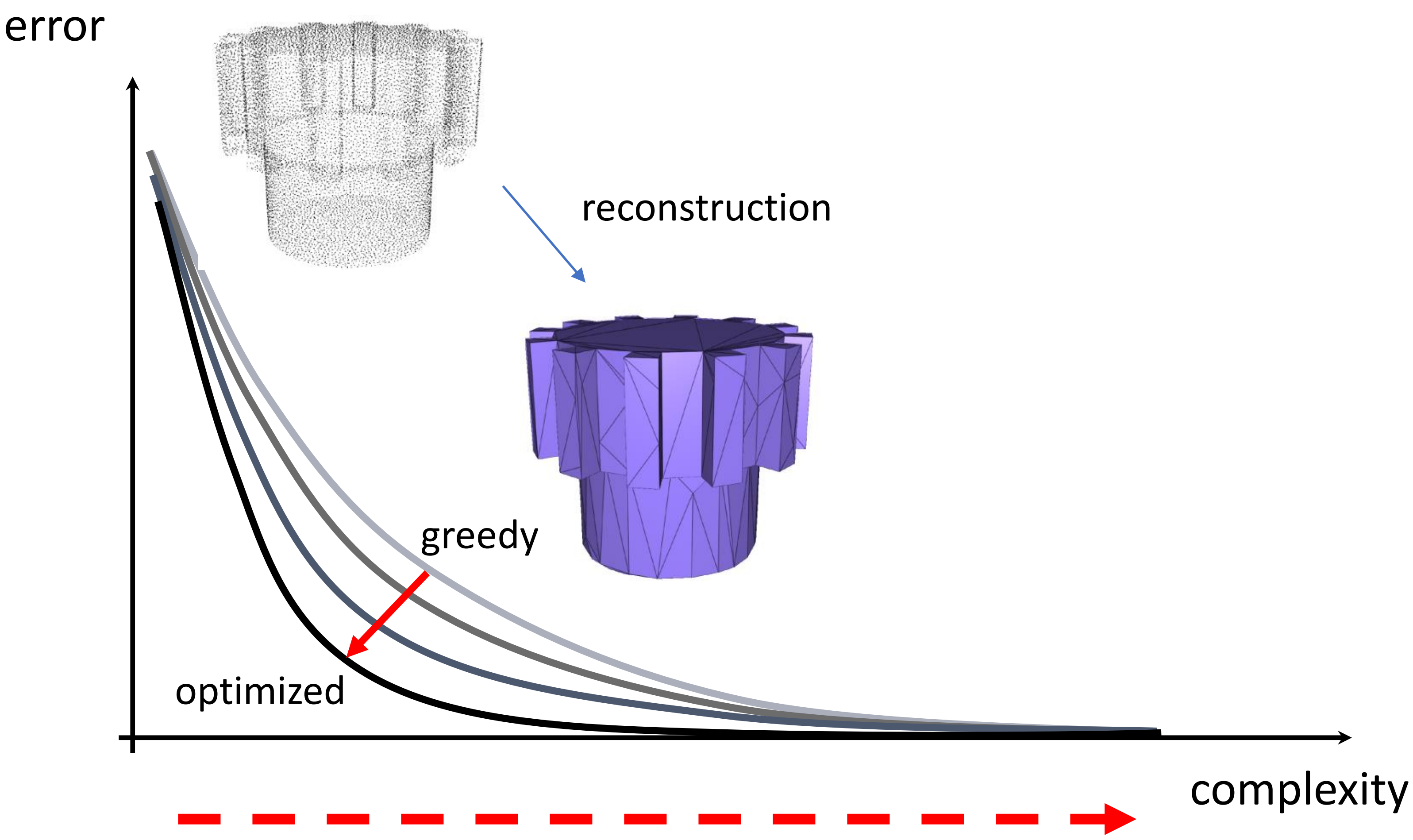
Quadric Error Metrics



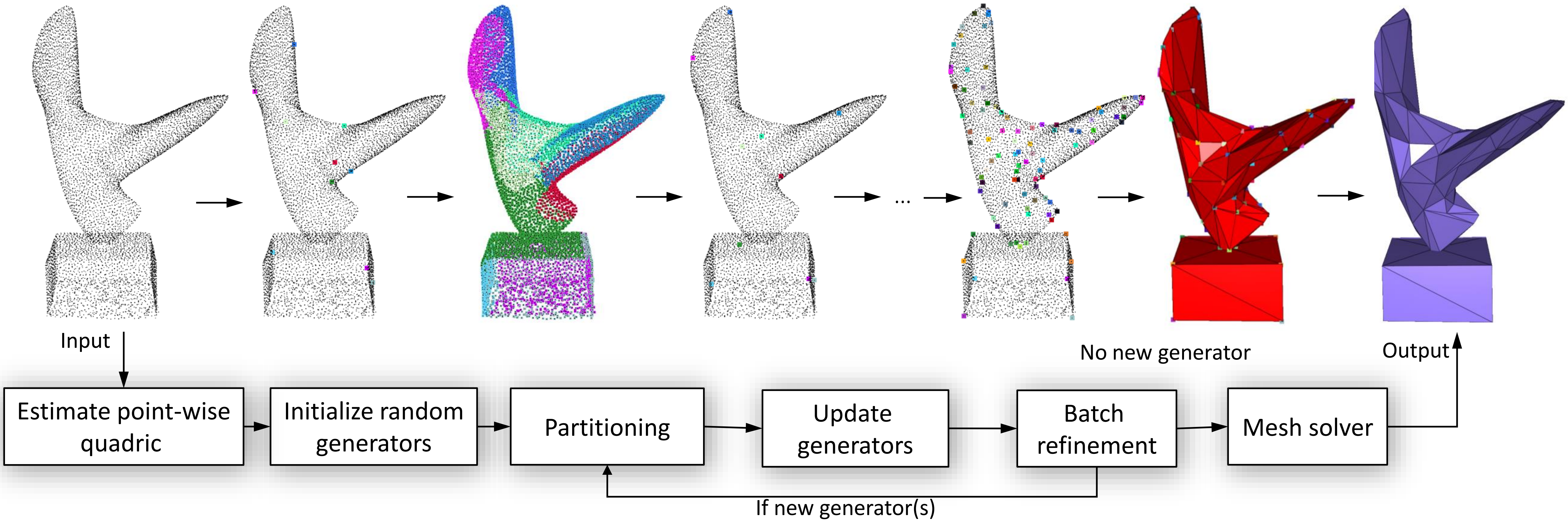
Complexity-Error Tradeoff



Objective



Pipeline

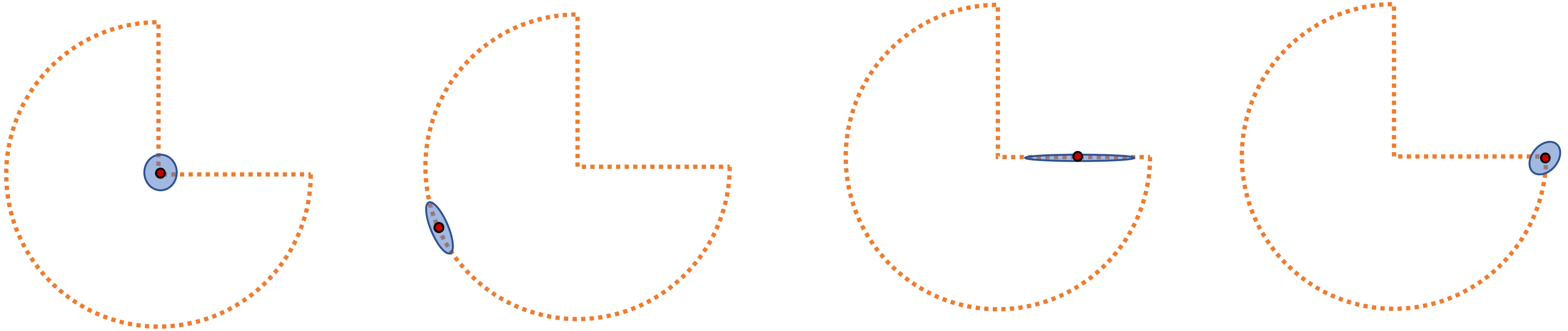

Input
...
No new generator
Output
Estimate point-wise quadric
Initialize random generators
Partitioning
Update generators
Batch refinement
Mesh solver
If new generator(s)

QEM Initialization

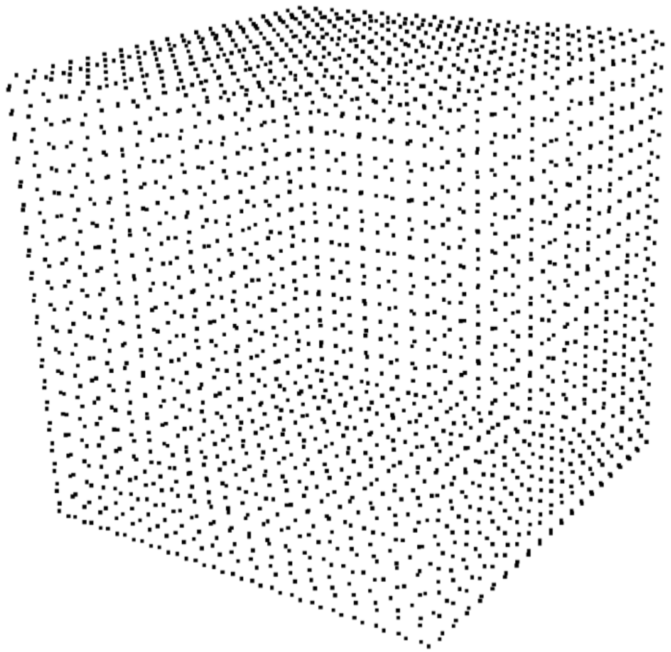
1. Point QEM: $Q_{p_i} = [n_i, p_i] \cdot [n_i, p_i]^T$

2. Point area: $a_{p_i} = \frac{1}{2k^2} \cdot \left(\sum_{p_j | (p_i, p_j) \in KNN(\mathcal{P})} \| p_i - p_j \| \right)^2$

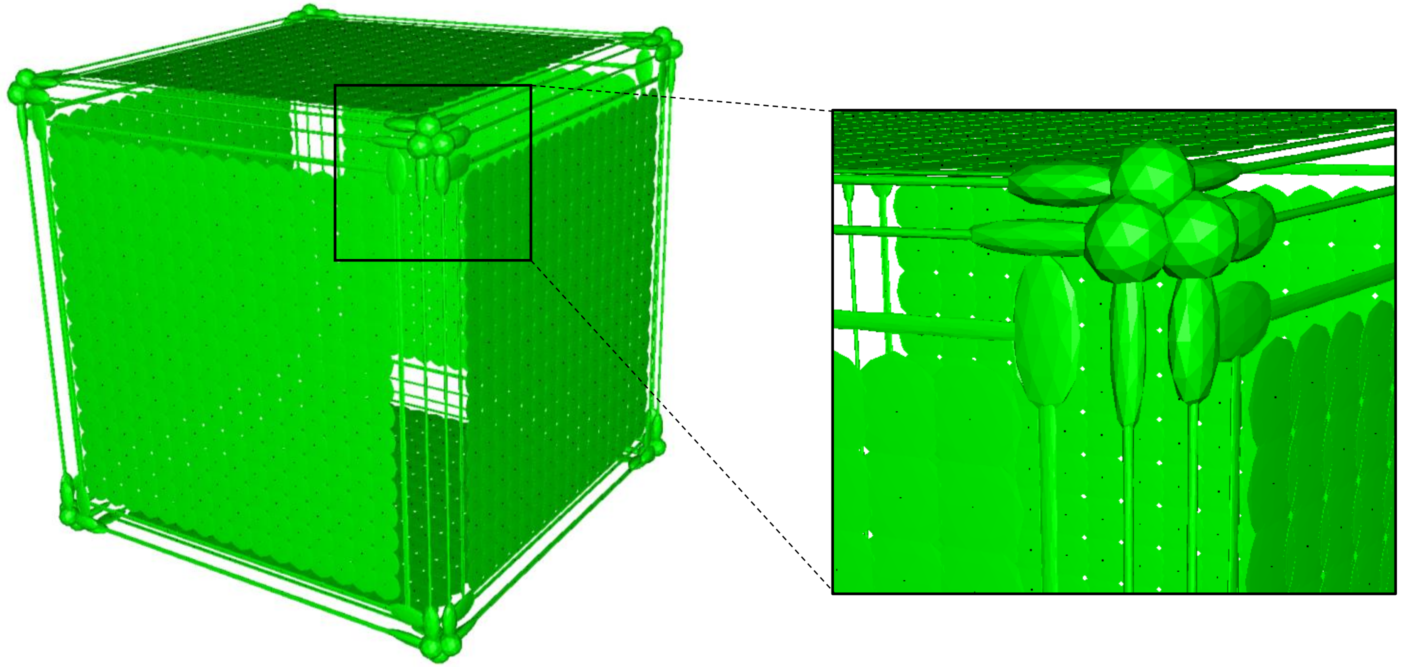
3. Diffused QEM: $Q_{v_i} = \sum_{p_j | (p_i, p_j) \in KNN(\mathcal{P})} a_{p_j} \cdot Q_{p_j}$



QEM Initialization

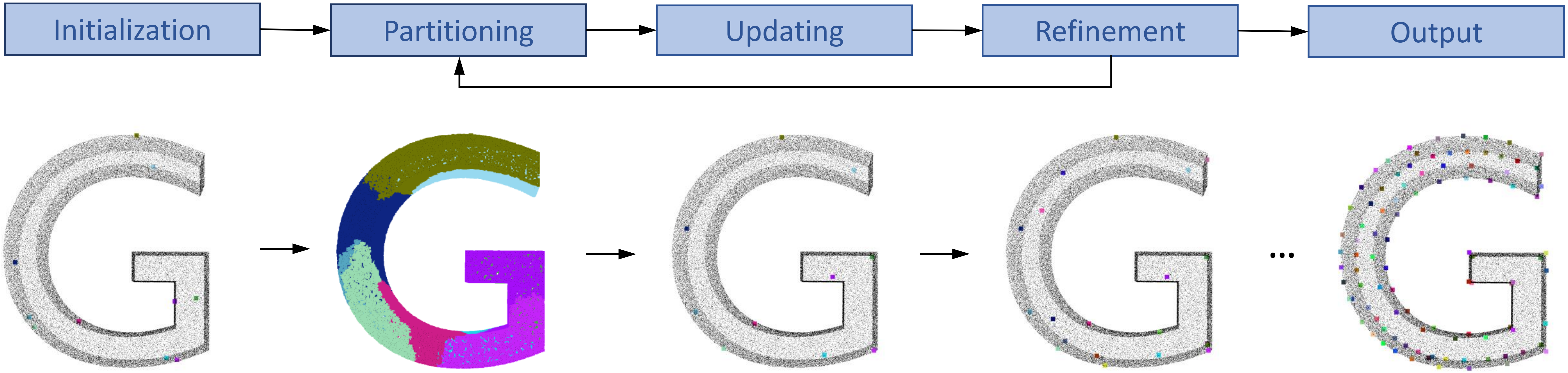


(a) Point cloud

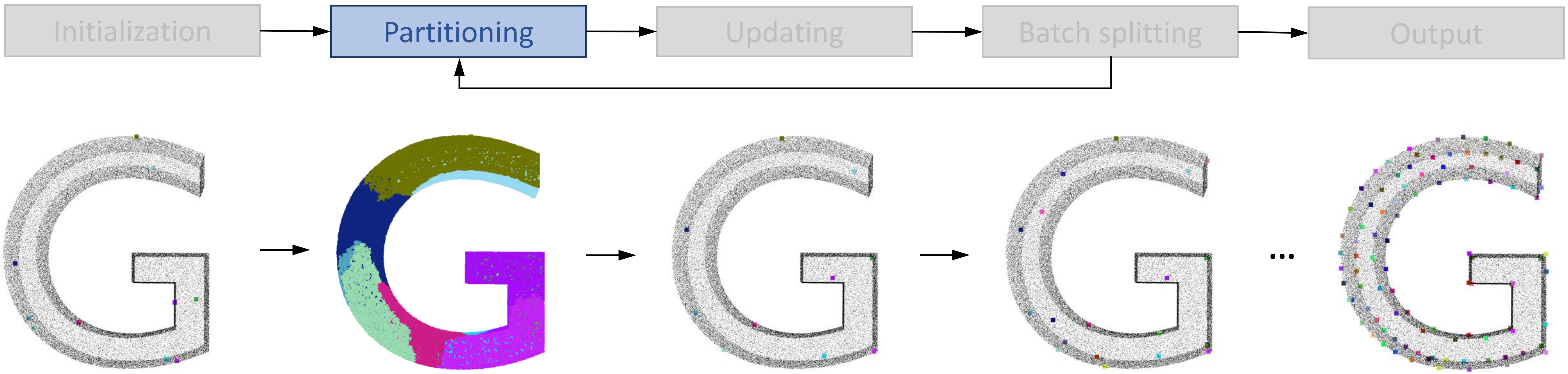


(a) Diffused QEM ellipsoids

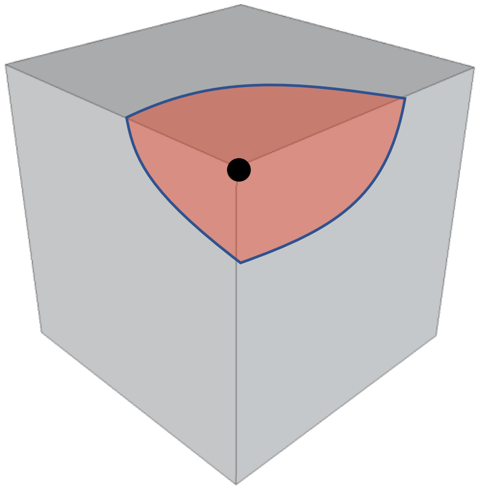
Clustering


Initialization
Partitioning
Updating
Refinement
Output
...

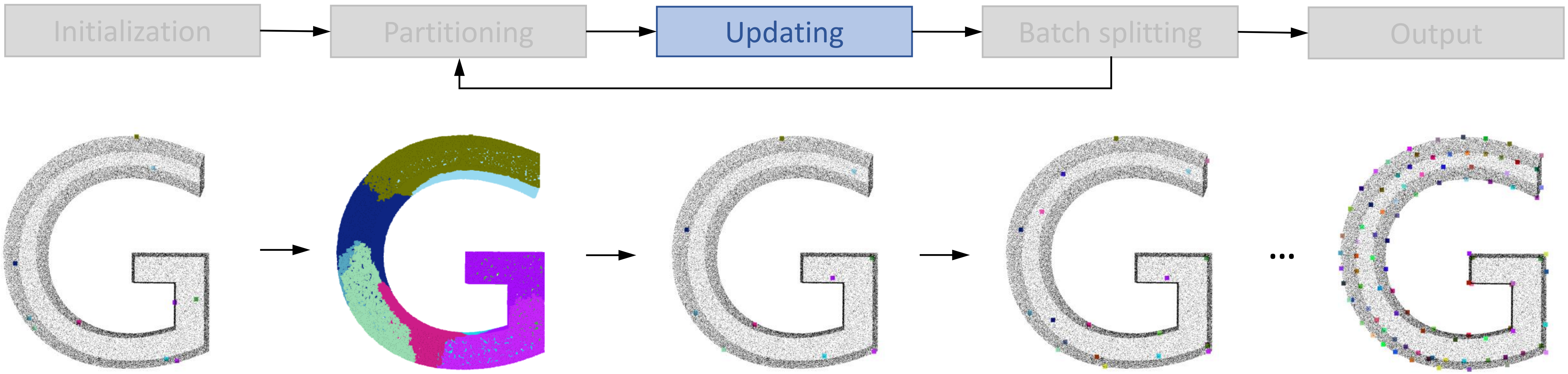
Clustering



$$E(p_i, l_j) = [c_j, 1]^T \cdot Q_{v_i} \cdot [c_j, 1] + \lambda \cdot \|p_i - c_j\|^2$$

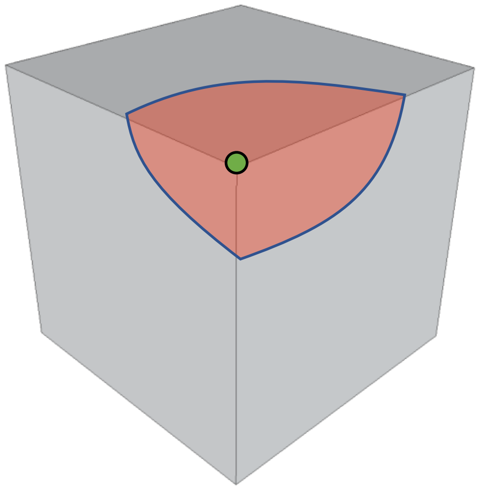


Clustering

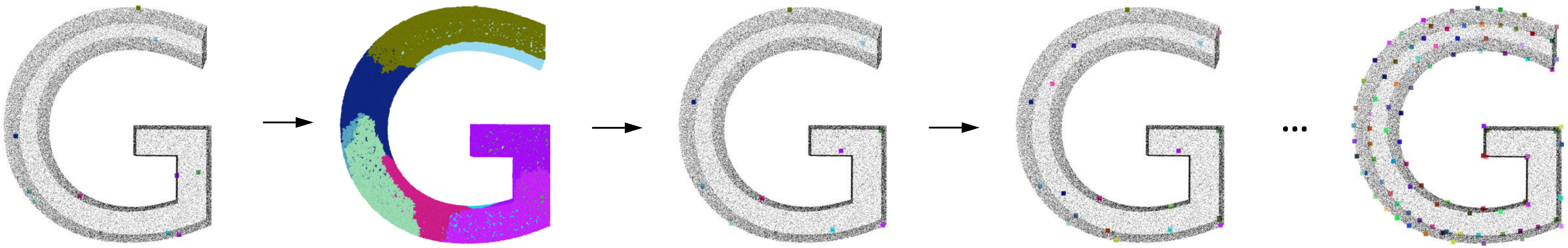


$$E(p_i, l_j) = [c_j, 1]^T \cdot Q_{v_i} \cdot [c_j, 1] + \lambda \cdot \|p_i - c_j\|^2$$

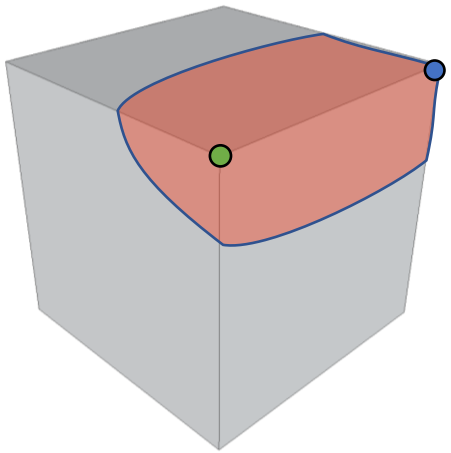
$$c_j^* = argmin_{p \in \mathbb{R}^3} [p, 1]^T \cdot Q_{c_j} \cdot [p, 1]$$



Clustering



$$E(p_i, l_j) = [c_j, 1]^T \cdot Q_{v_i} \cdot [c_j, 1] + \lambda \cdot \|p_i - c_j\|^2$$



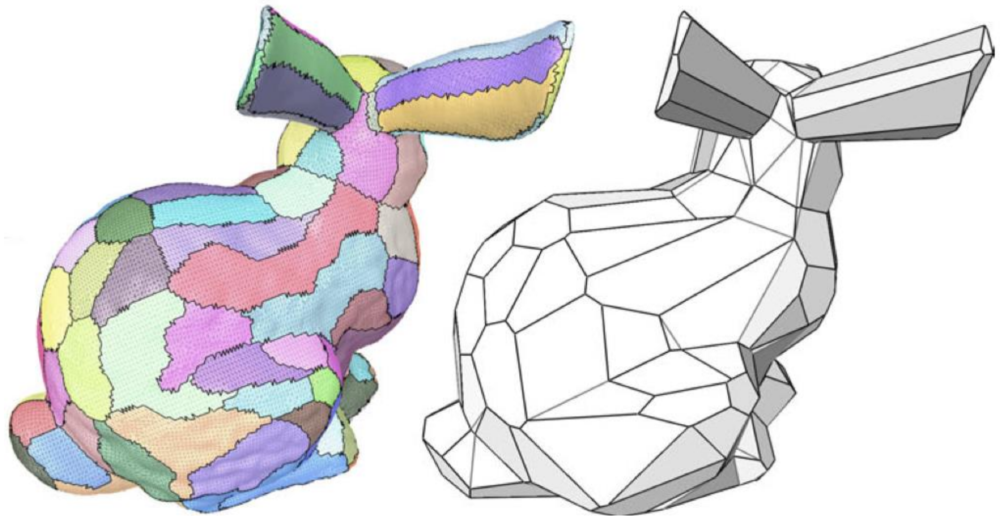
$$c_j^* = argmin_{p \in \mathbb{R}^3} [p, 1]^T \cdot Q_{c_j} \cdot [p, 1]$$

$$p_{max}(l_j) = argmax_{p_i \in l_j} [p_i, 1]^T \cdot Q_{c_j} \cdot [p_i, 1]$$

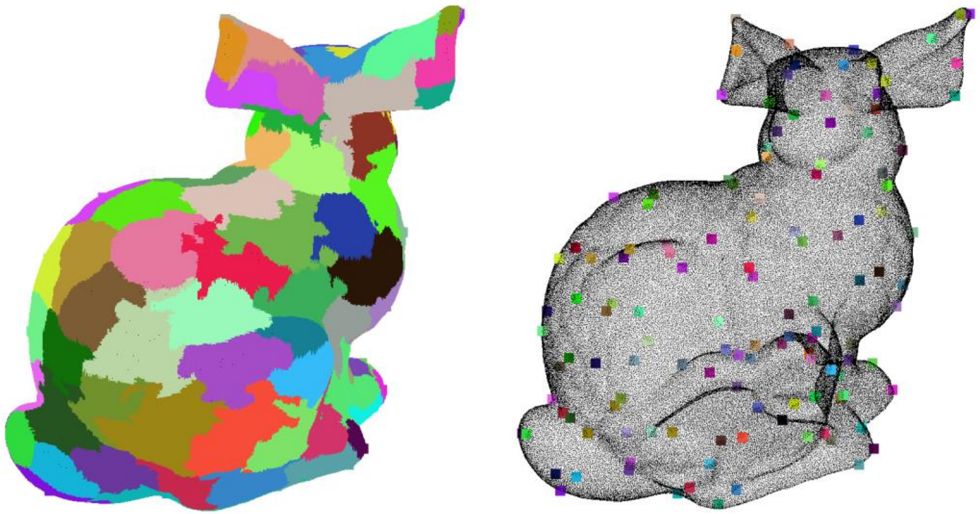
Meshing

Variational shape approximation[1] : “dual“ partitioning → not trivial to extract triangle mesh

Our approach: optimized generators → vertices of the output triangle mesh



VSA[1]



This approach

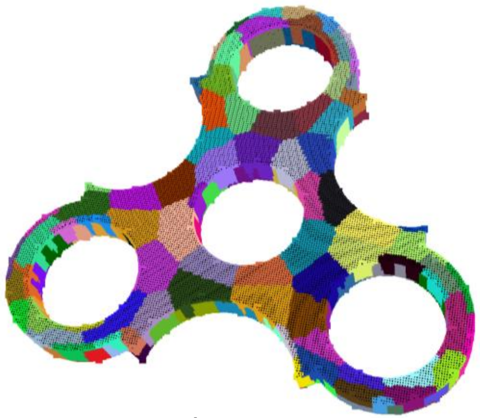
[1] Cohen-Steiner et al. Variational Shape Approximation. ACM SIGGRAPH, 2004.

Meshing

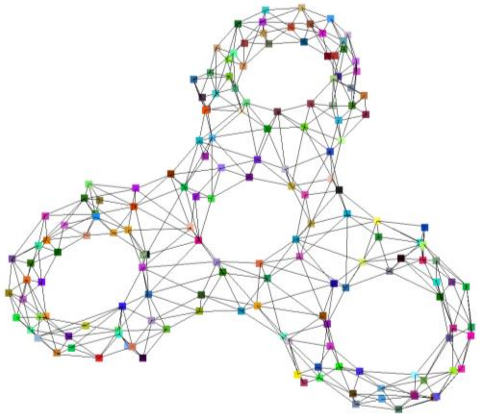
1. Construct edge candidate set: connect adjacent clusters

2. Construct facet candidate set: find 3-cycles

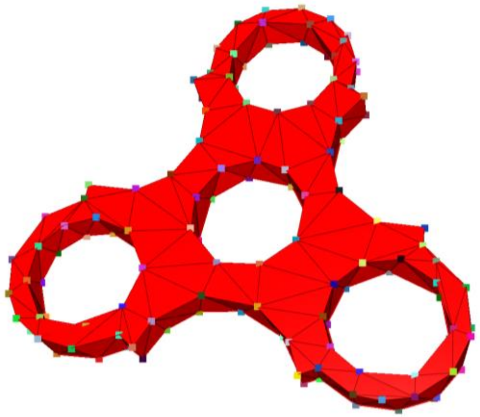
3. Mesh extraction via Binary Integer Programming (BIP) solver [1]



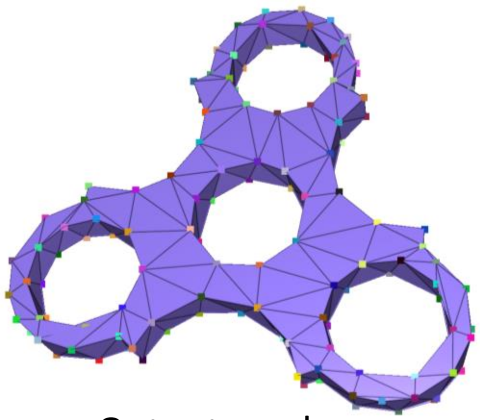
Clusters



Edge graph



Facet candidates



Output mesh

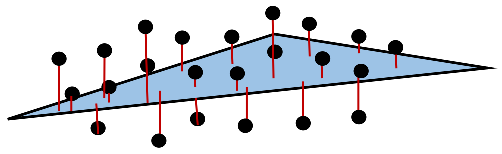
[1] Nan, Liangliang, et al. Polyfit: Polygonal surface reconstruction from point clouds. Proceedings of the IEEE International Conference on Computer Vision. 2017.

Meshing

Fitting term | Coverage term | Manifold constraint

$$E(\mathcal{T}) = \lambda_f \sum_{i=1}^{n} b_{f_i} \cdot F_f(f_i)$$

$$F_f(f_i) = \sum_{p_j | d(p_j, f_i) < \epsilon} \left(1 - \frac{d(p_j, f_i)}{\epsilon}\right)$$

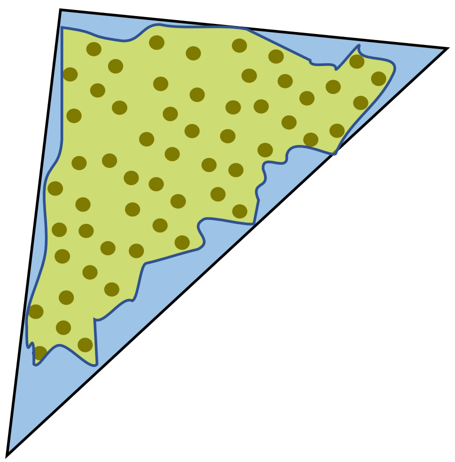


Meshing

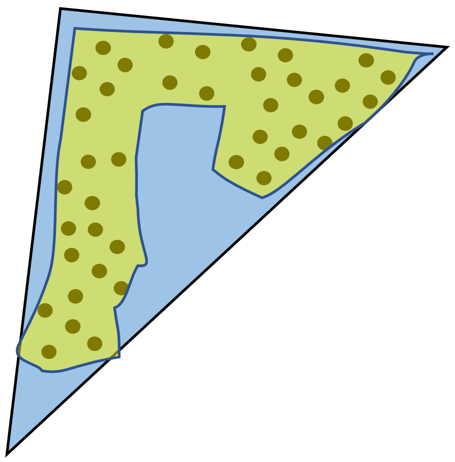
Fitting term | Coverage term | Manifold constraint

$$E(\mathcal{T}) = \lambda_f \sum_{i=1}^{n} b_{f_i} \cdot F_f(f_i) + \lambda_c \sum_{i=1}^{n} b_{f_i} \cdot F_c(f_i)$$

$$F_c(f_i) = min\left(1, \frac{area(\alpha(\{p_j | d(p_j, f_i) < \epsilon\}))}{area(f_i)}\right)$$



$F_c(f_i) = 0.95$



$F_c(f_i) = 0.65$

Meshing

Fitting term

Coverage term

Manifold constraint

$$E(\mathcal{T}) = \lambda_f \sum_{i=1}^{n} b_{f_i} \cdot F_f(f_i) + \lambda_c \sum_{i=1}^{n} b_{f_i} \cdot F_c(f_i)$$

$$s.t. \quad 2b_{e_i} - \sum_{f_j \, around \, e_i} b_{f_j} = 0, \quad \forall e_i$$

If $b_{e_i} = 1$, 2 faces around e_i are selected

If $b_{e_i} = 0$, no face around e_i is selected

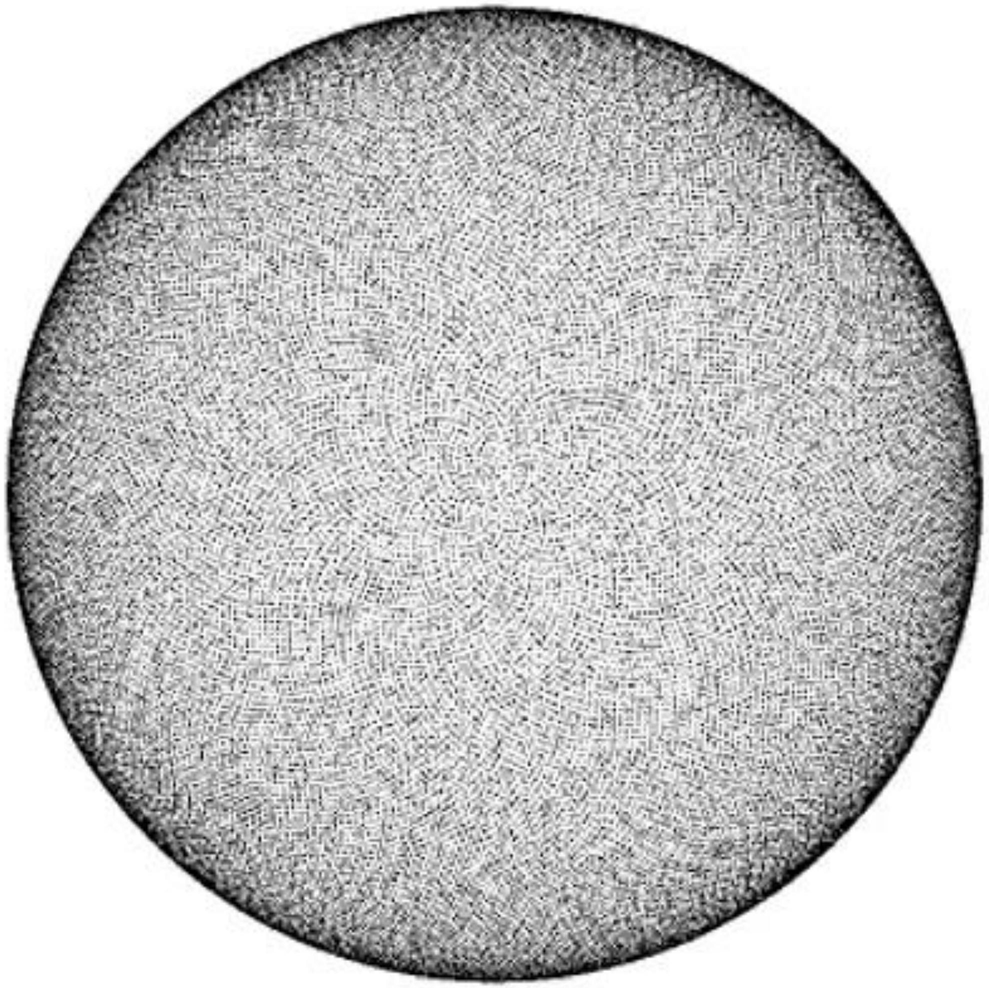
↓

Manifoldness

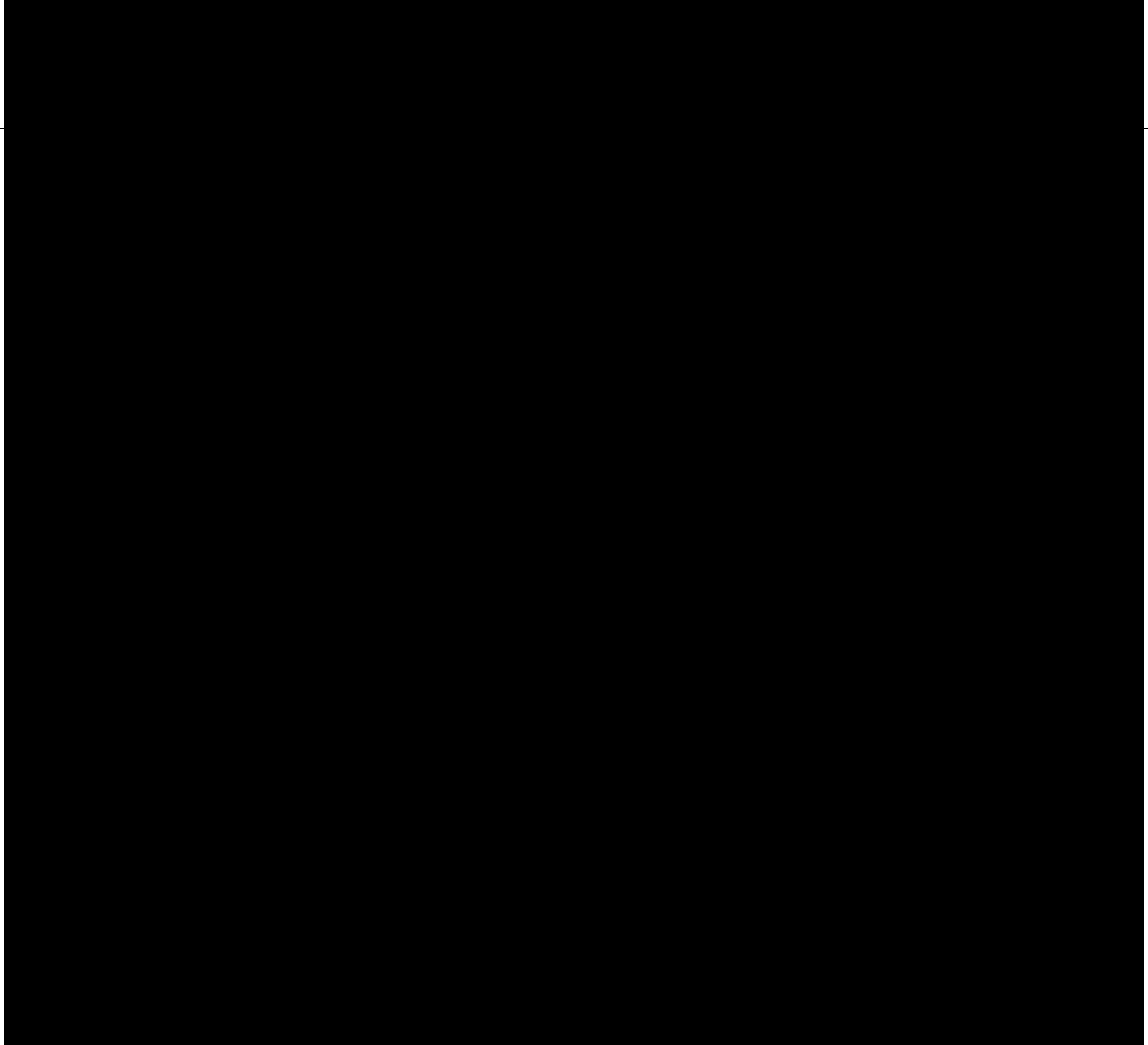
Implementation

- CGAL
- Eigen (quadrics)
- Solver: SCIP (mixed integer programming)

Sphere

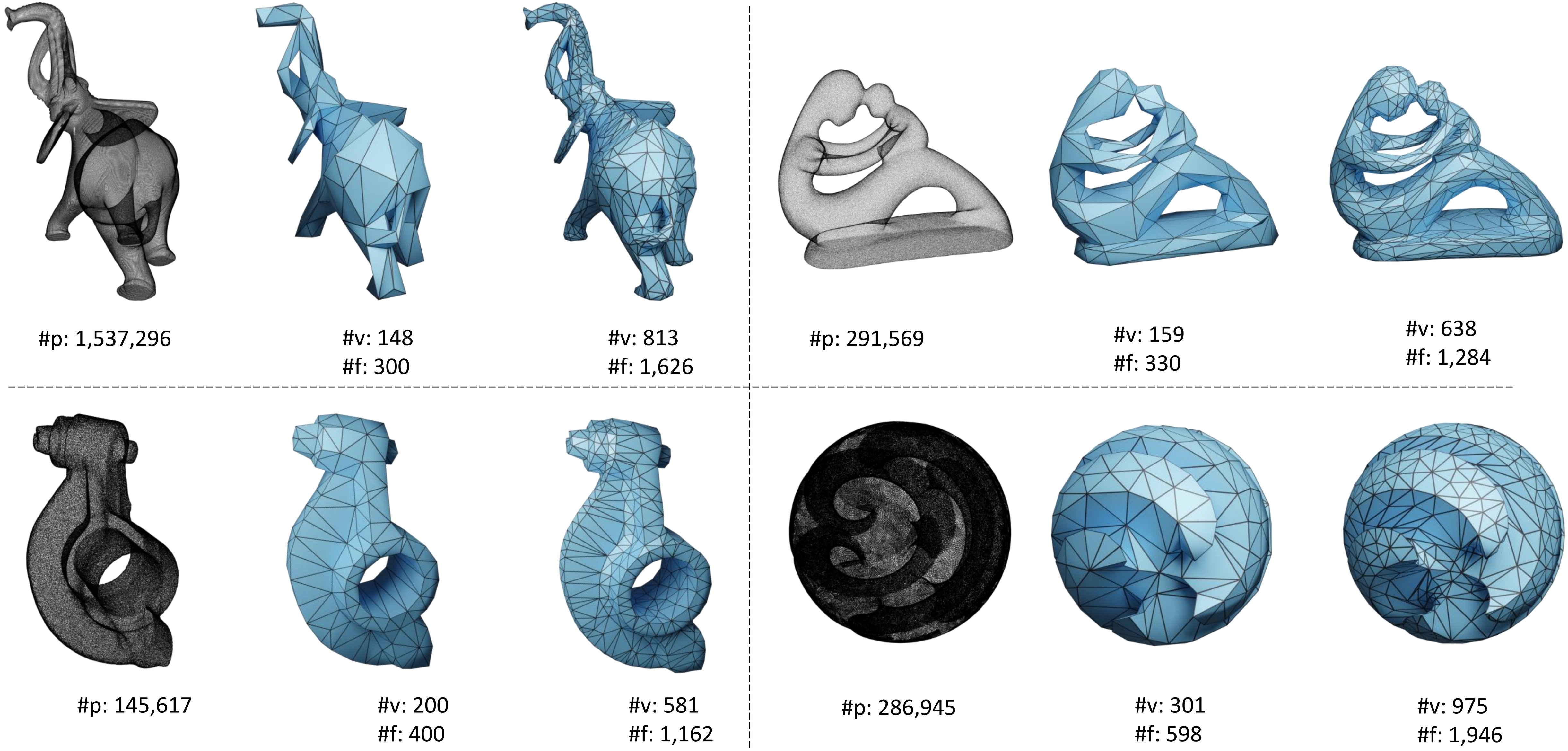


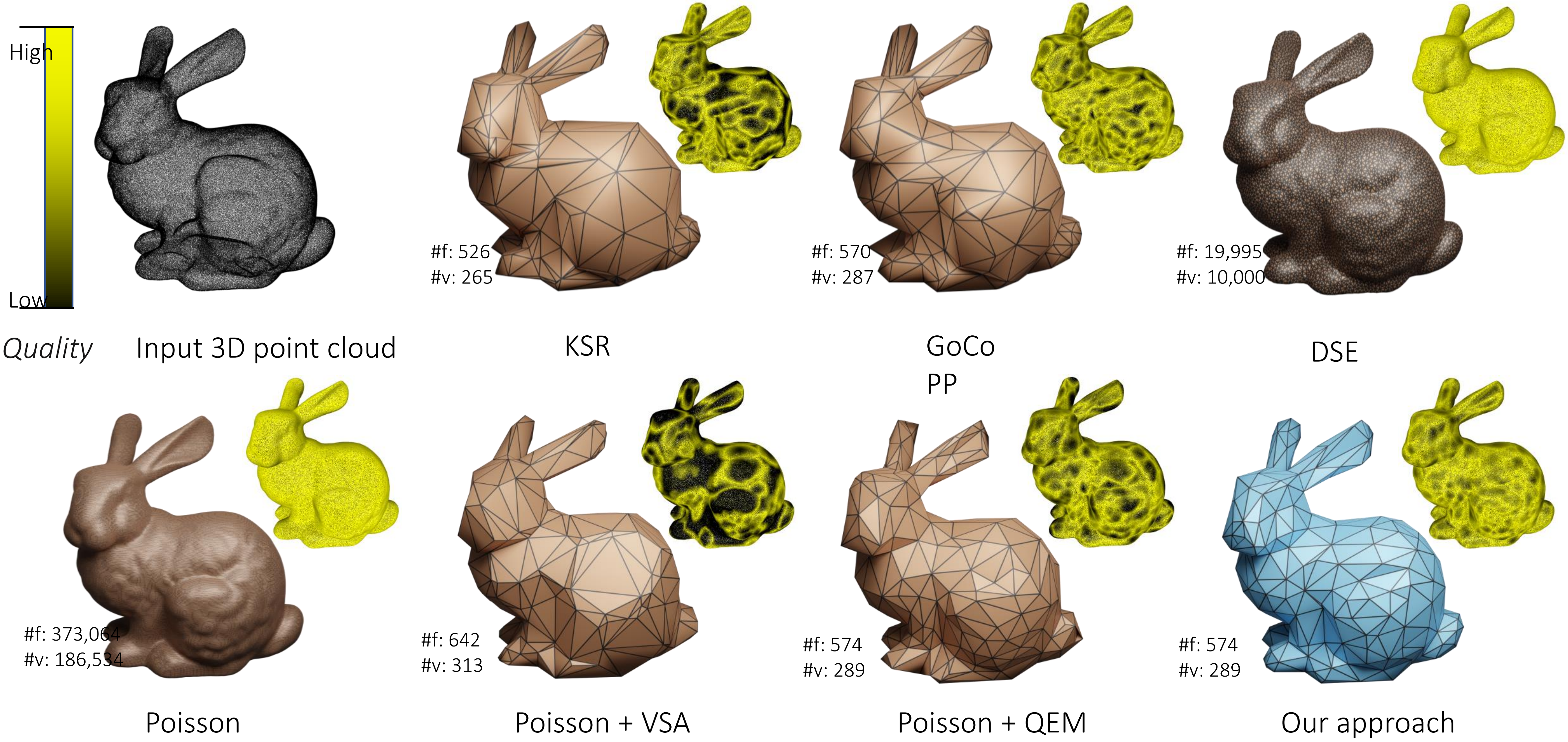
Hand



Blade

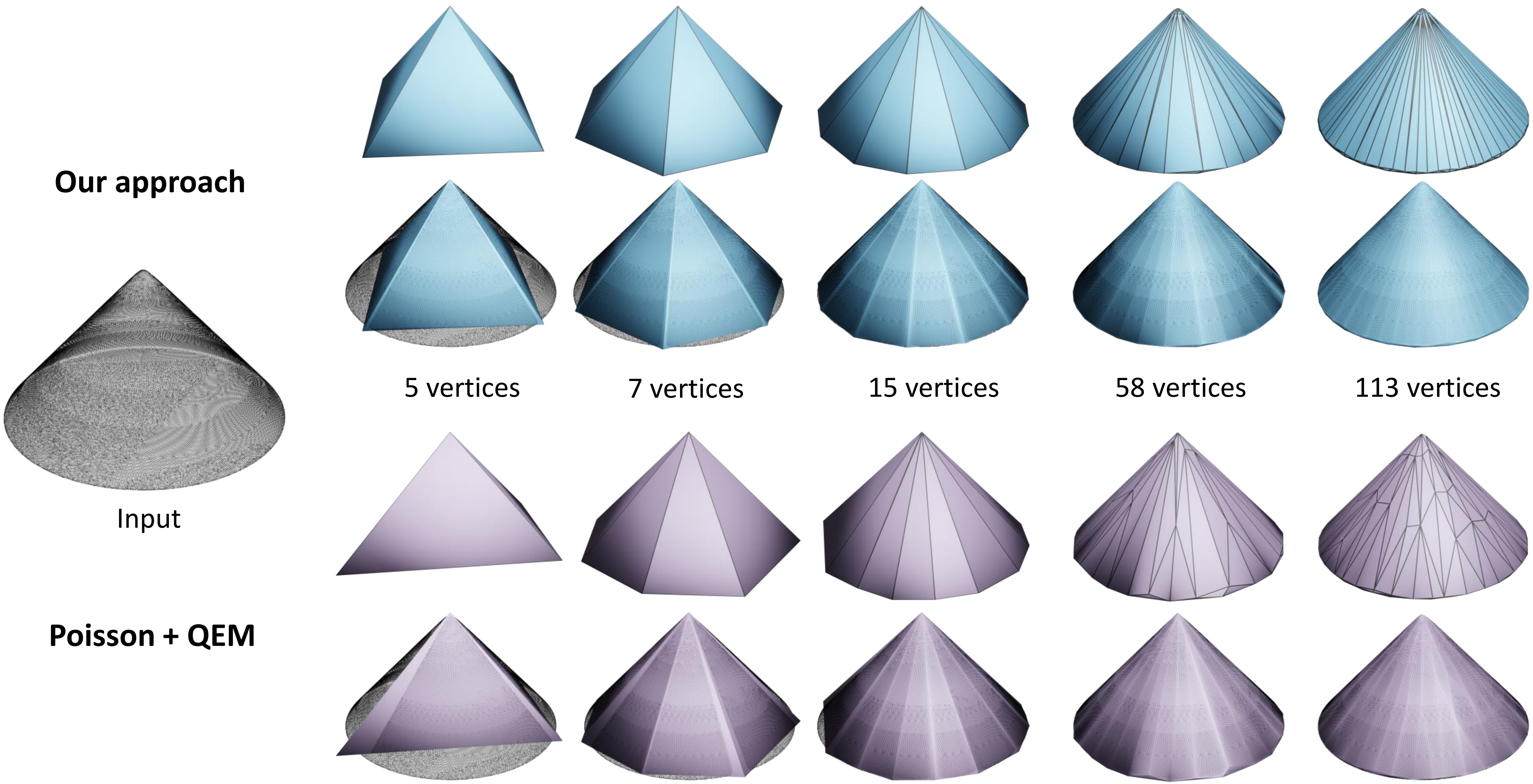
Results


#p: 1,537,296
#v: 148
#f: 300
#v: 813
#f: 1,626
#p: 291,569
#v: 159
#f: 330
#v: 638
#f: 1,284
#p: 145,617
#v: 200
#f: 400
#v: 581
#f: 1,162
#p: 286,945
#v: 301
#f: 598
#v: 975
#f: 1,946

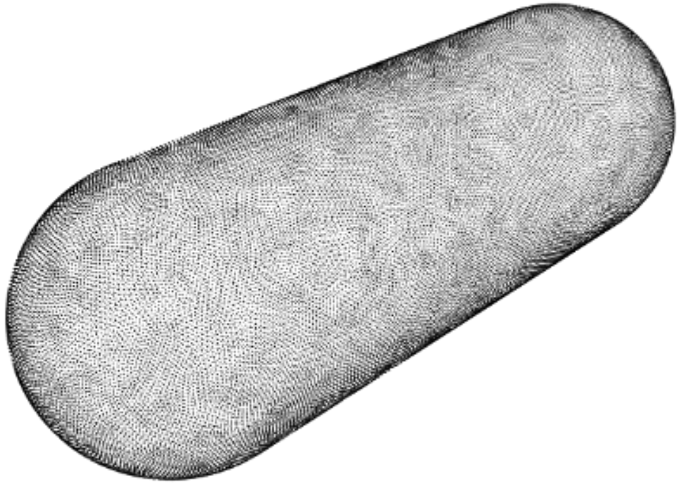


[1] Bauchet, Jean-Philippe, et al. Kinetic shape reconstruction. ACM TOG 39(5), 2020.
[2] Yu, Mulin, et al. Finding Good Configurations of Planar Primitives in Unorganized Point Clouds. IEEE CVPR, 2022.
[3] Rakotosaona, Marie-Julie, et al. Learning delaunay surface elements for mesh reconstruction. IEEE CVPR, 2021.
[4] Kazhdan, Michael, et al. Screened poisson surface reconstruction. ACM Transactions on Graphics 32, no. 3, 2013.
[5] Cohen-Steiner, David, et al. Variational shape approximation. ACM SIGGRAPH, 2004.
[6] Garland, Michael, and et al. Surface simplification using quadric error metrics. ACM SIGGRAPH. 1997.

Variational VS Greedy

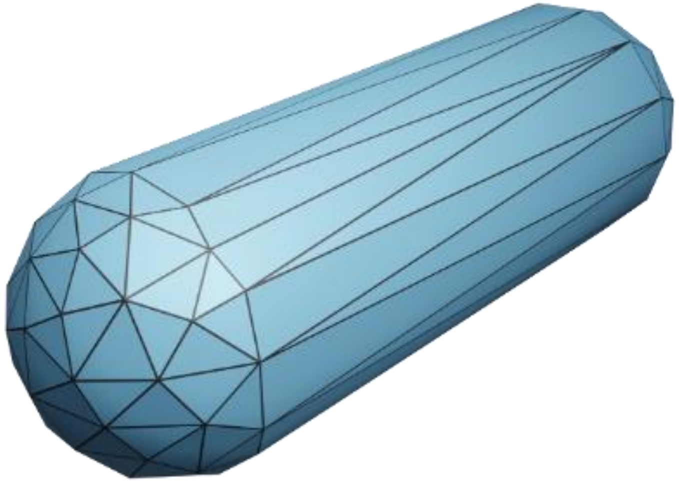


Capsule



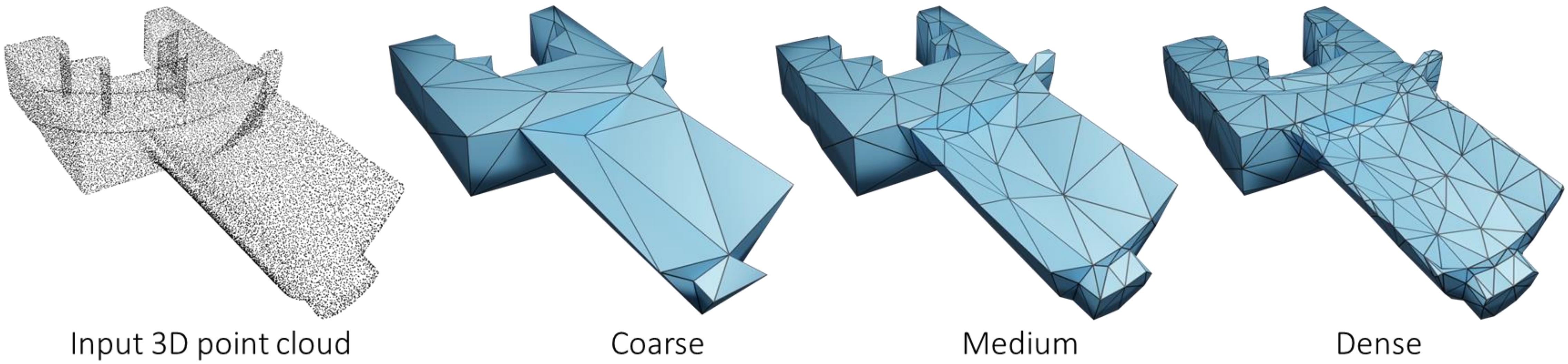
Input 3D point cloud

Poisson + QEM

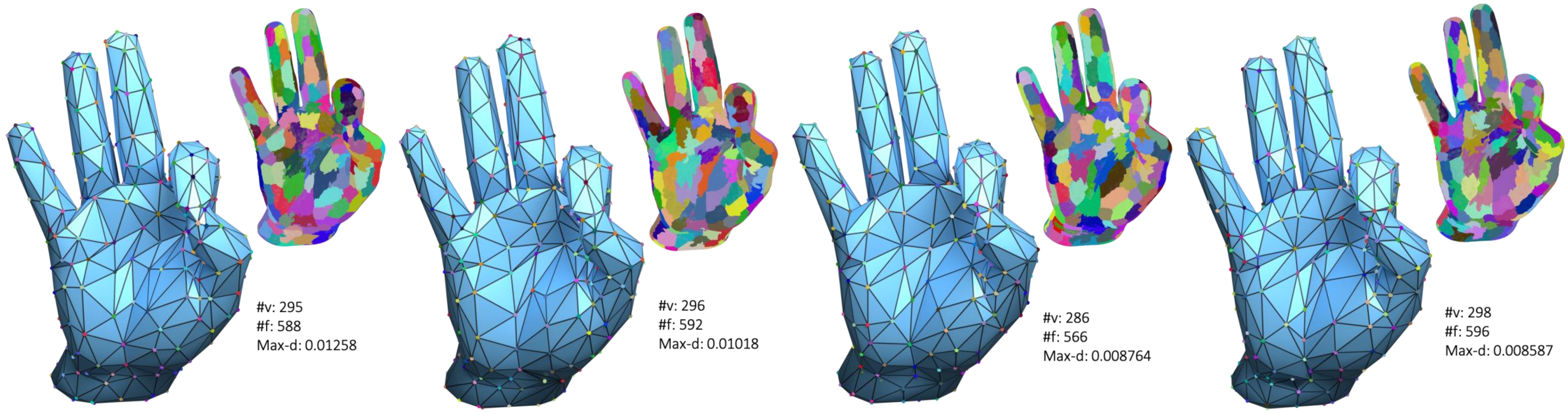


Our approach

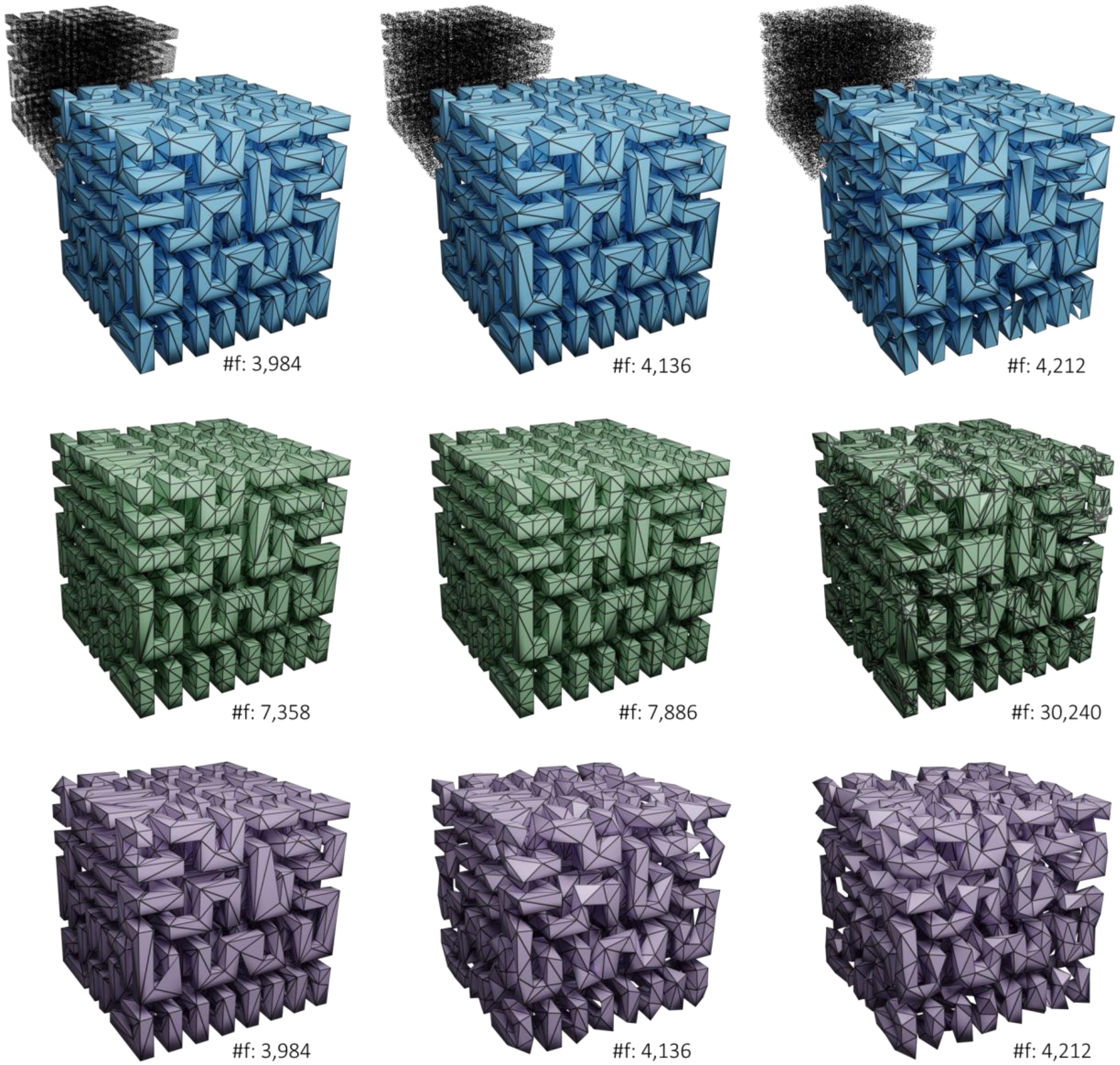
Blade



Initialization

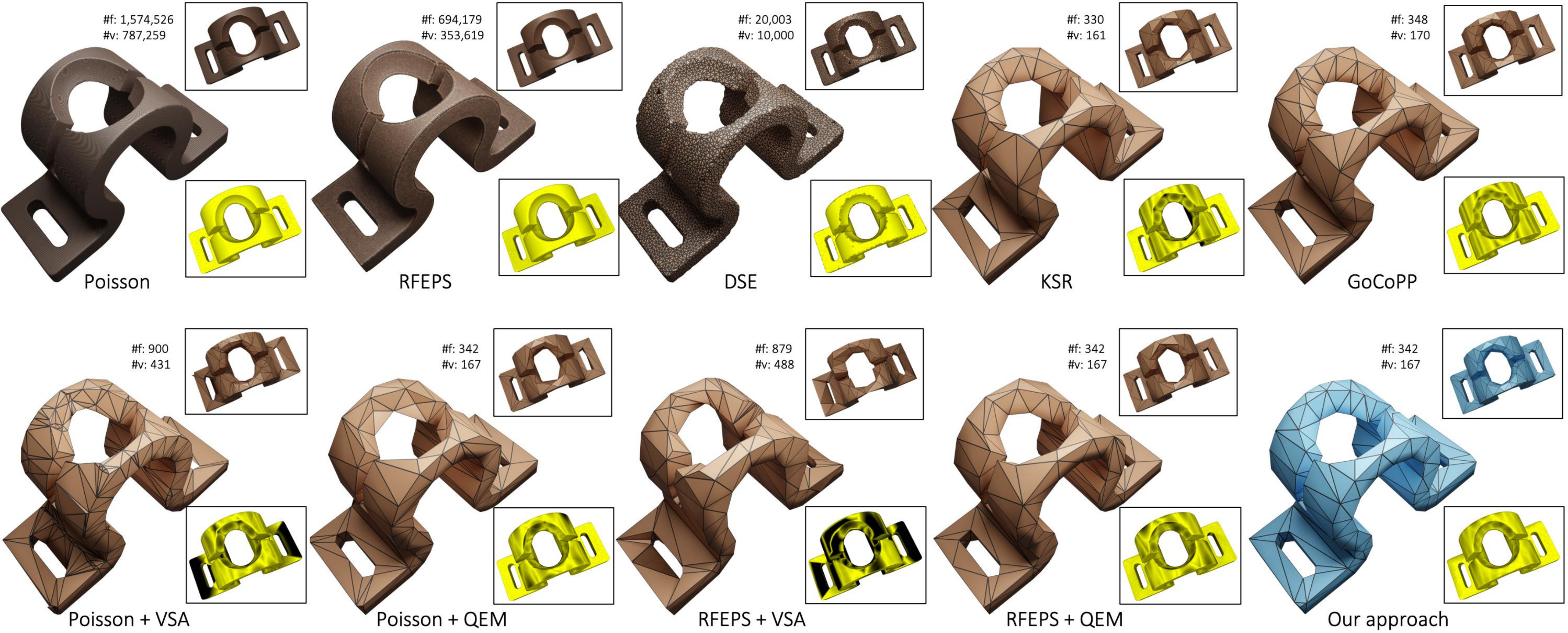


Robustness to Noise



No noise 0.5% noise 1% noise

Result


#f: 1,574,526
#v: 787,259
Poisson
#f: 694,179
#v: 353,619
RFEPS
#f: 20,003
#v: 10,000
DSE
#f: 330
#v: 161
KSR
#f: 348
#v: 170
GoCoPP
#f: 900
#v: 431
Poisson + VSA
#f: 342
#v: 167
Poisson + QEM
#f: 879
#v: 488
RFEPS + VSA
#f: 342
#v: 167
RFEPS + QEM
#f: 342
#v: 167
Our approach

Discussion

- Initialization
  - Learning to regress quadrics ?
  - Sharp features ?
- Scheduling batches of refinement
  - Parameter-free?
  - Tolerance error?
- Fine-to-coarse?
  - How to interleave cluster merging and relaxation?
- Surfaces with boundaries
- 2-Manifold: hard vs soft constraint
- Valid mesh as output? (no self-intersections)

PoNQ: a Neural QEM-based Mesh Representation

Nissim Maruani, Maks Ovsjanikov, P.A., Mathieu Desbrun

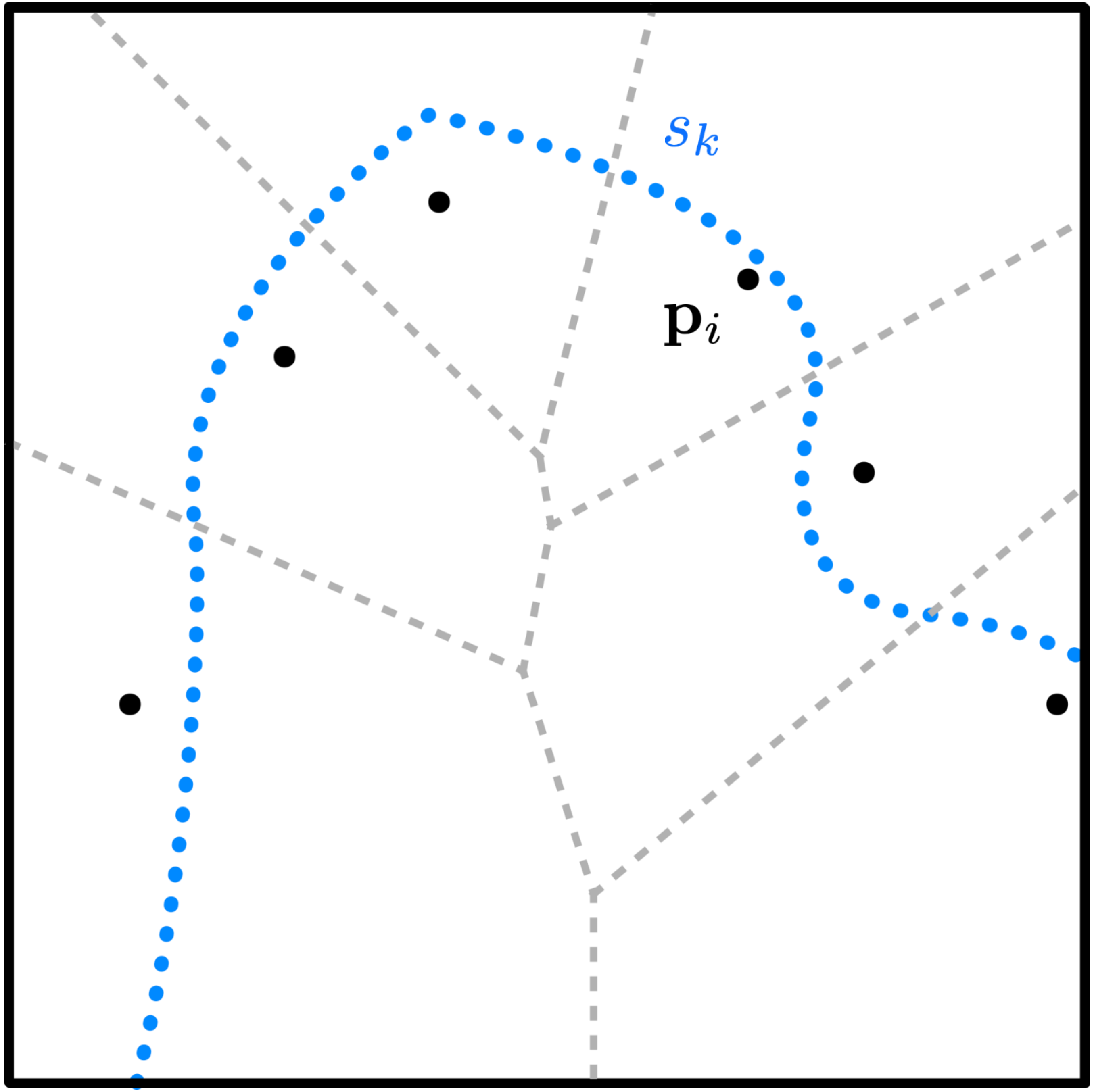
CVPR
JUNE 17-21, 2024
SEATTLE, WA

Method

$s_k \in \mathbb{R}^3, n(s_k) \in \mathbb{R}^3$

$\mathbf{p}_i \in \mathbb{R}^3$

(Qualitative illustration)



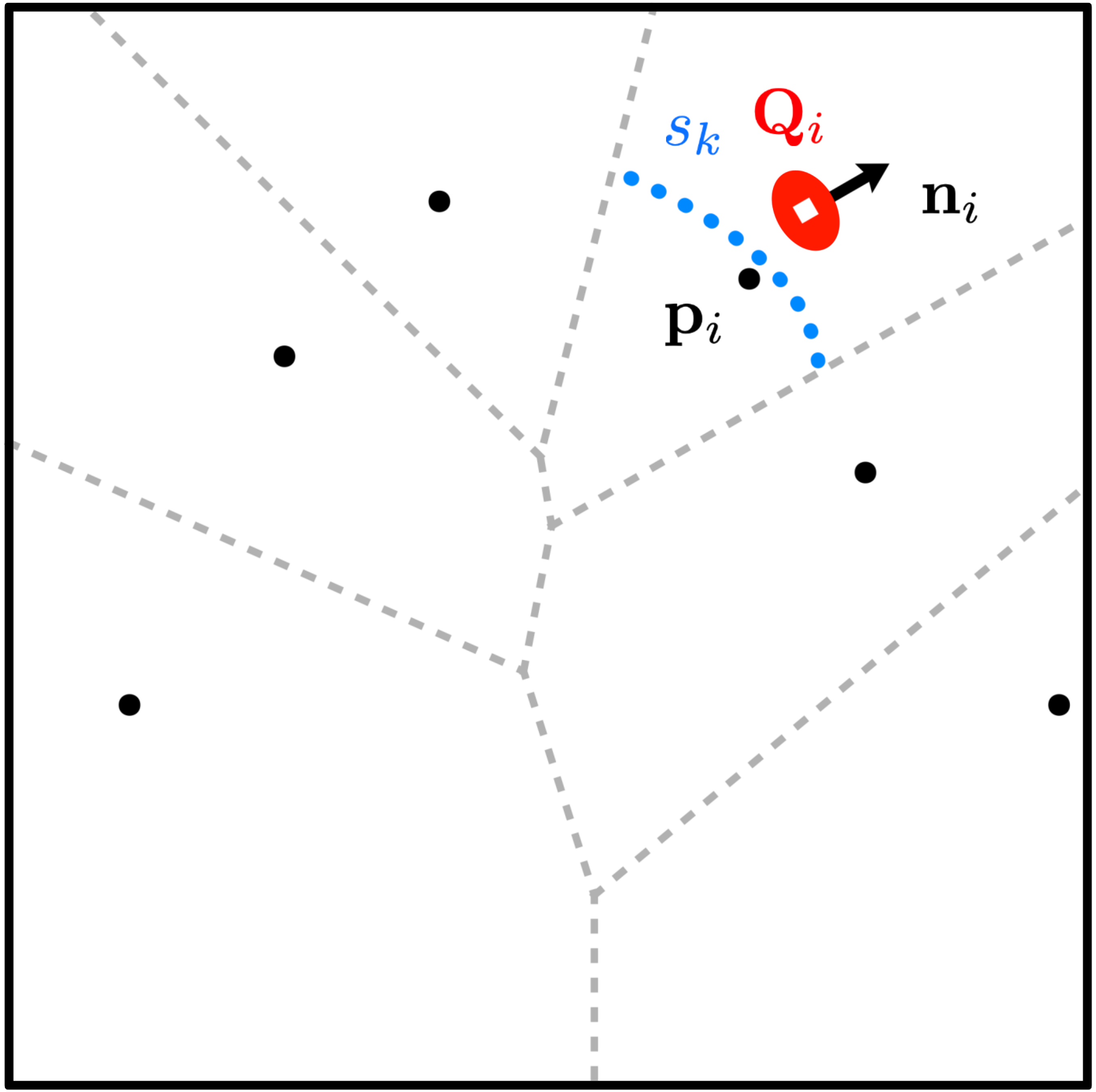
Method

$s_k \in \mathbb{R}^3, n(s_k) \in \mathbb{R}^3$

$\mathbf{p}_i \in \mathbb{R}^3$

$\mathbf{Q}_i \in \mathbb{R}^{4\times4}$

$\mathbf{n}_i \in \mathbb{R}^3$



(Qualitative illustration)

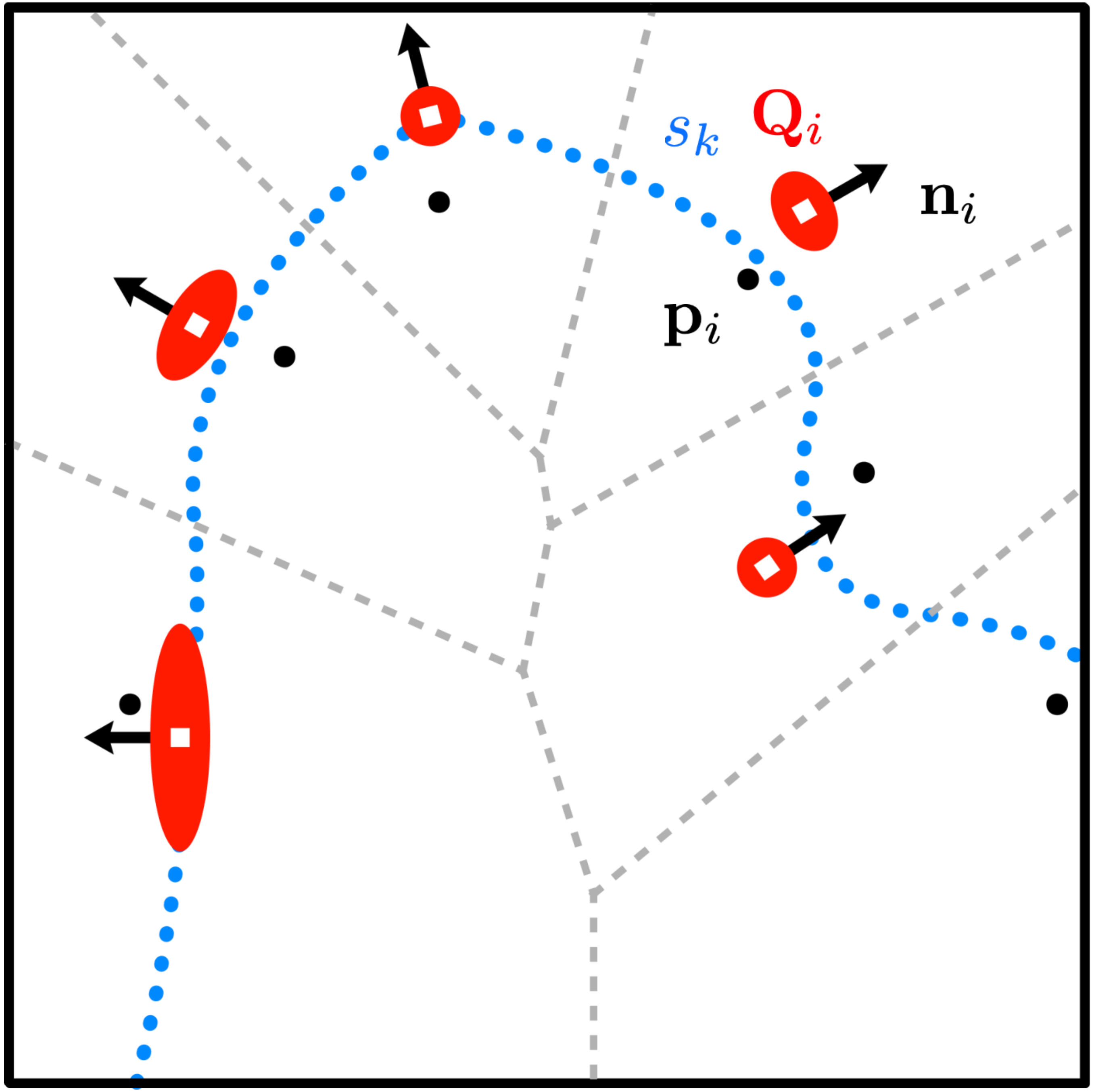
Method

$s_k \in \mathbb{R}^3, n(s_k) \in \mathbb{R}^3$

$\mathbf{p}_i \in \mathbb{R}^3$

$\mathbf{Q}_i \in \mathbb{R}^{4\times4}$

$\mathbf{n}_i \in \mathbb{R}^3$



(Qualitative illustration)

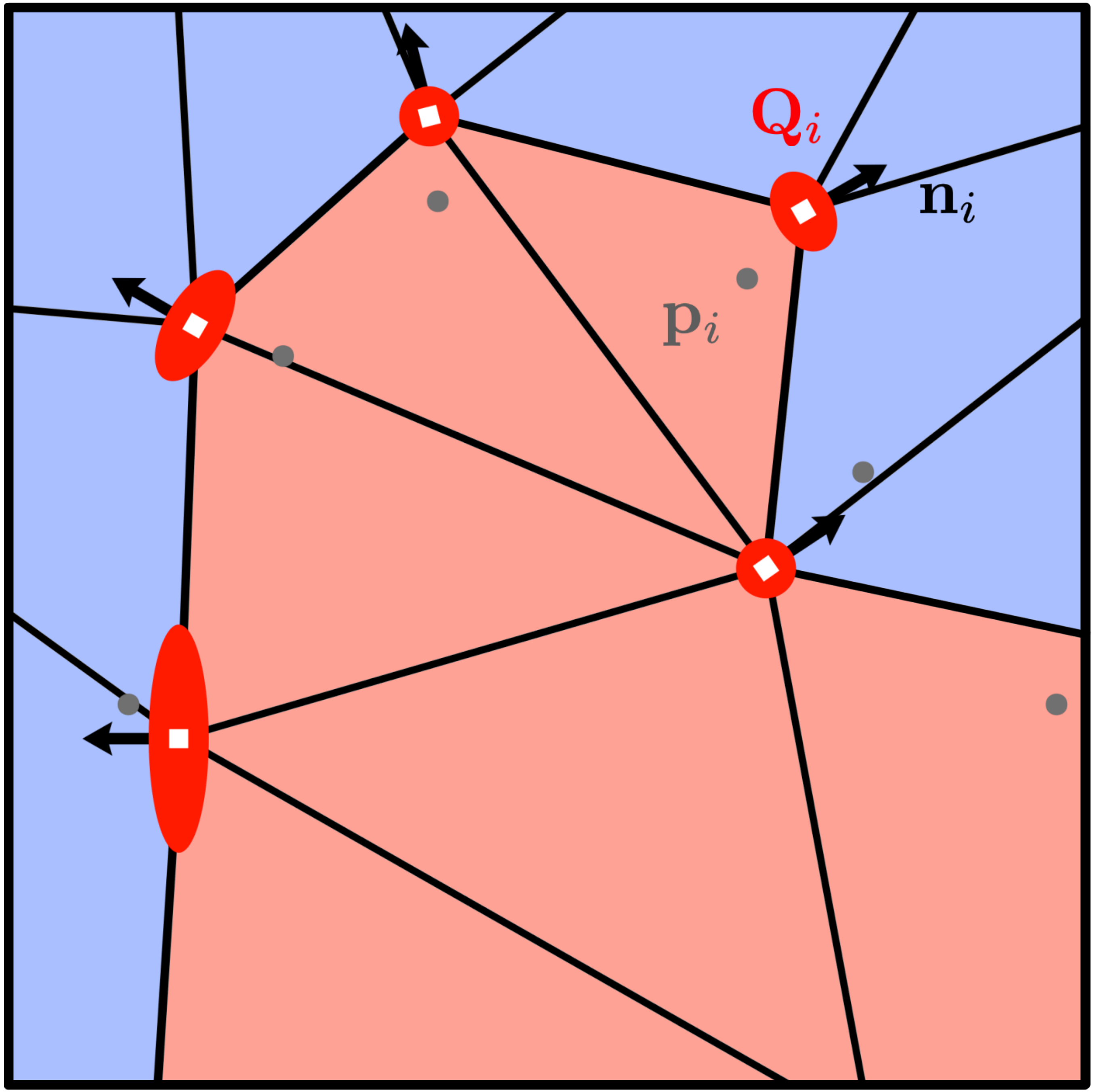
Method

$s_k \in \mathbb{R}^3, n(s_k) \in \mathbb{R}^3$

$\mathbf{p}_i \in \mathbb{R}^3$

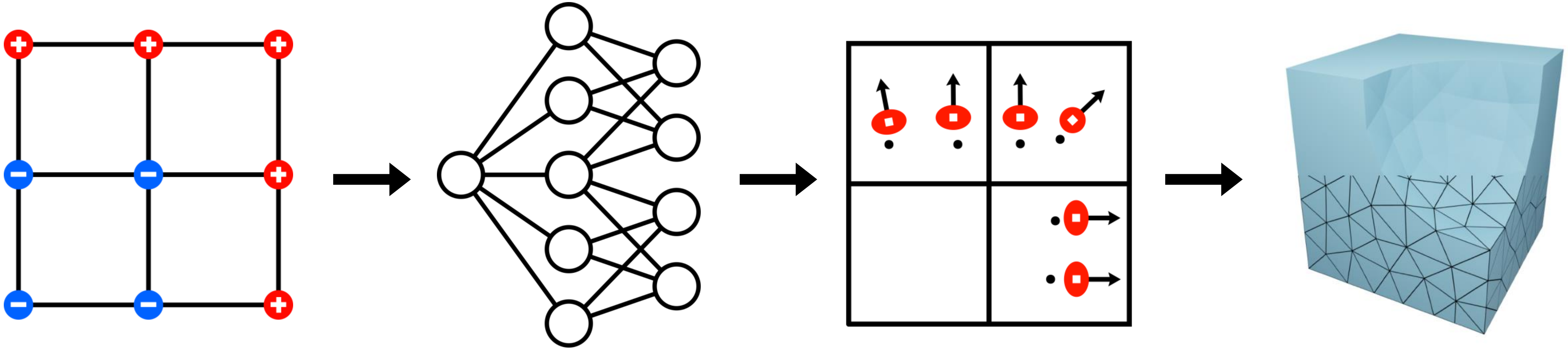
$\mathbf{Q}_i \in \mathbb{R}^{4\times4}$

$\mathbf{n}_i \in \mathbb{R}^3$



(Qualitative illustration)

Learning Pipeline



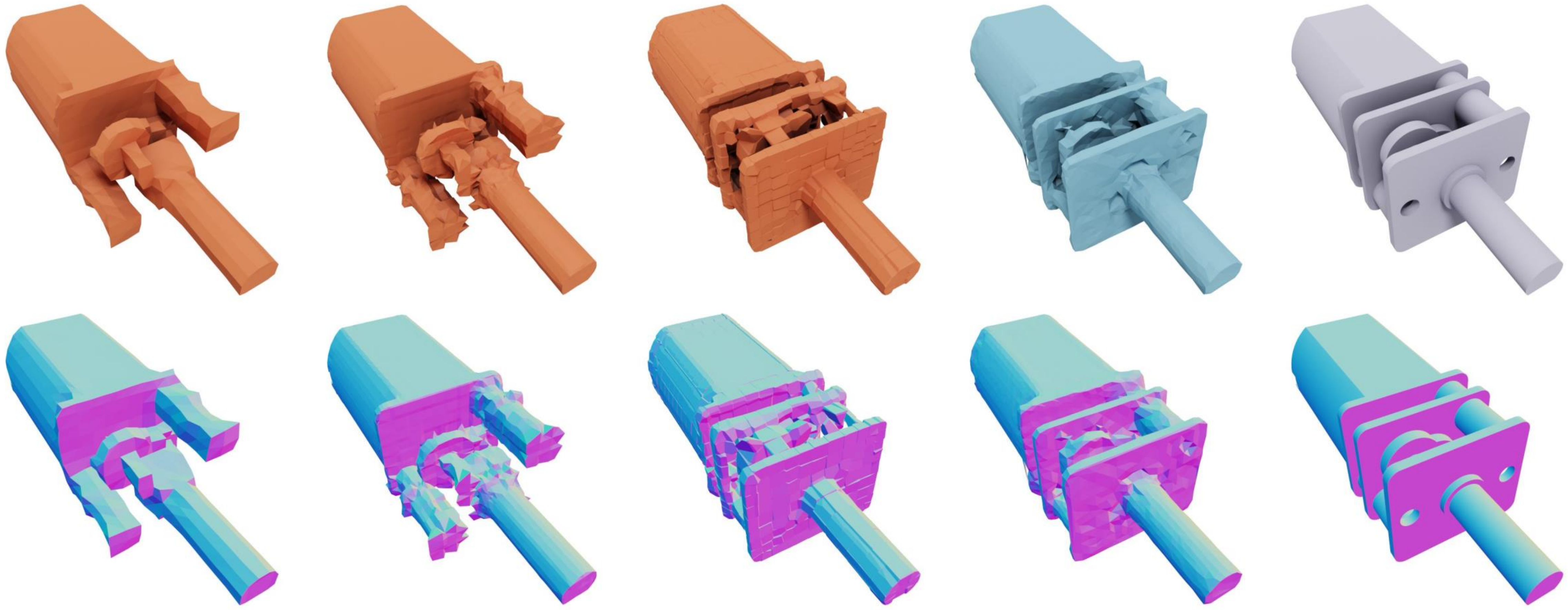
Regular Grid Sampling of SDF

3D CNN

Points, **N**ormals, **Q**uadrics

PoNQ Mesh

Learning-based Results



NDC [4] NMC [5] VoroMesh [6] PoNQ Gr. Truth

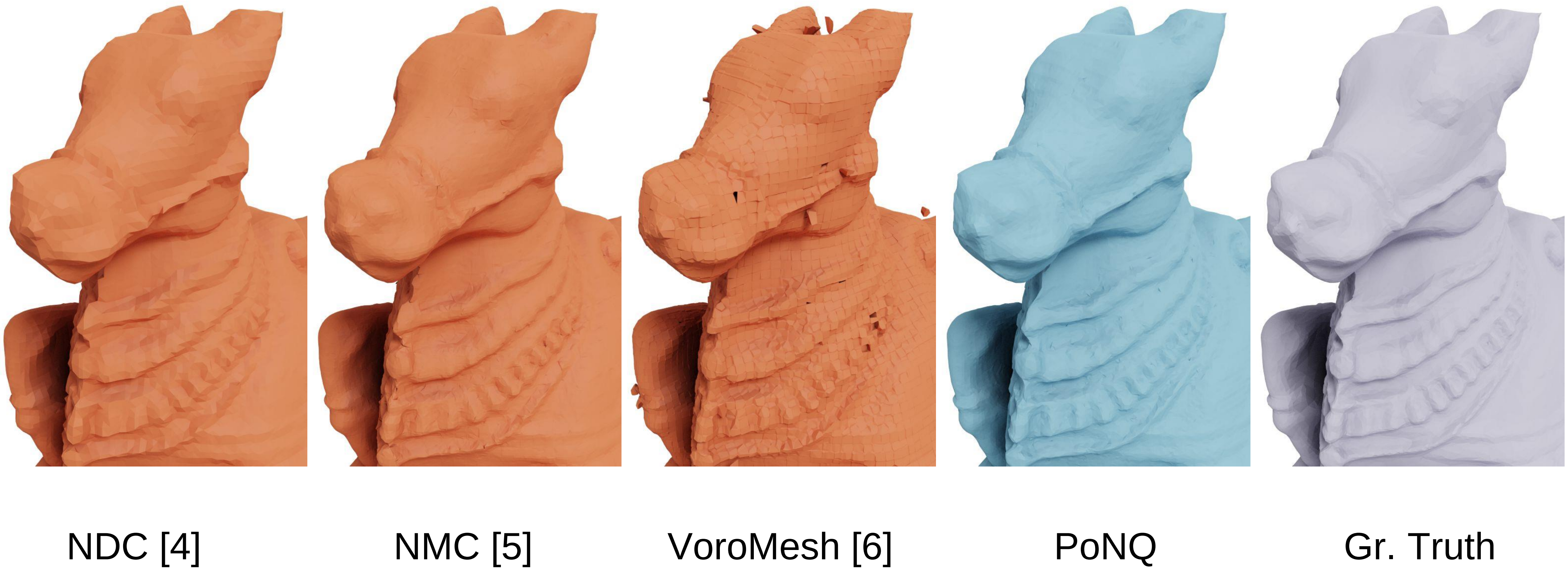
[4] Chen, et al. (2022) Neural Dual Contouring. [5] Chen, et al. (2021) Neural Marching Cubes.
[6] Maruani, et al. (2023) VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams.

Learning-based Results

NDC [4] NMC [5] VoroMesh [6] PoNQ Gr. Truth

[4] Chen, et al. (2022) Neural Dual Contouring. [5] Chen, et al. (2021) Neural Marching Cubes.
[6] Maruani, et al. (2023) VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams.

Learning-based Results

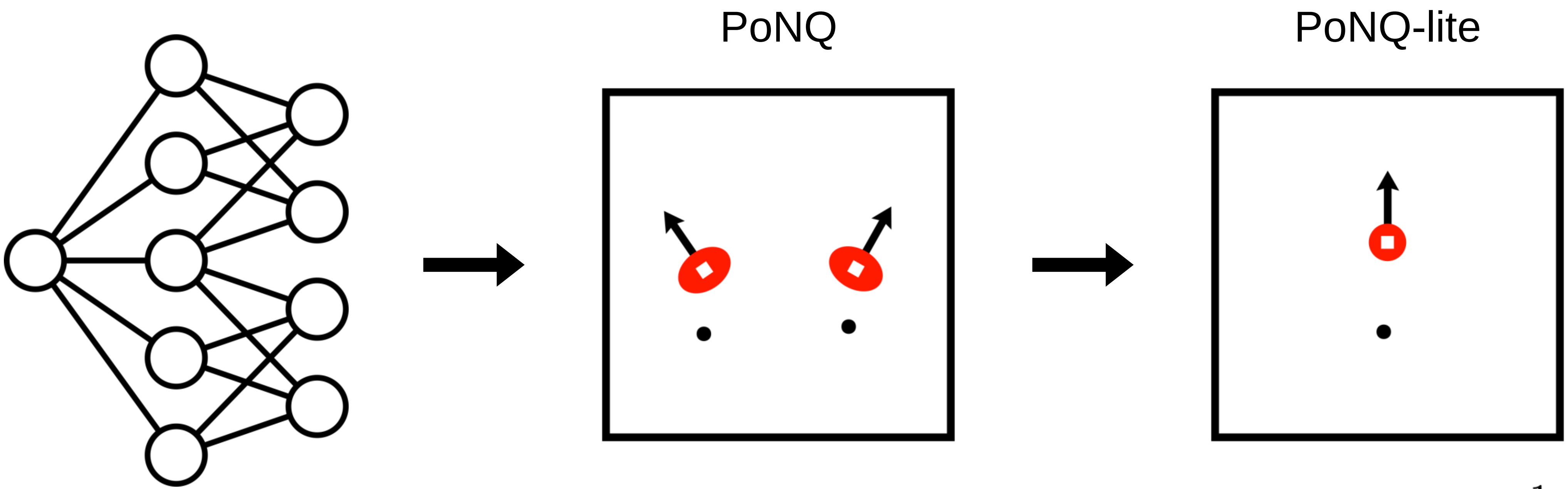


[4] Chen, et al. (2022) Neural Dual Contouring. [5] Chen, et al. (2021) Neural Marching Cubes.
[6] Maruani, et al. (2023) VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams.

High Resolutions

Learning-based PoNQ, 512^3

Extension: PoNQ-lite

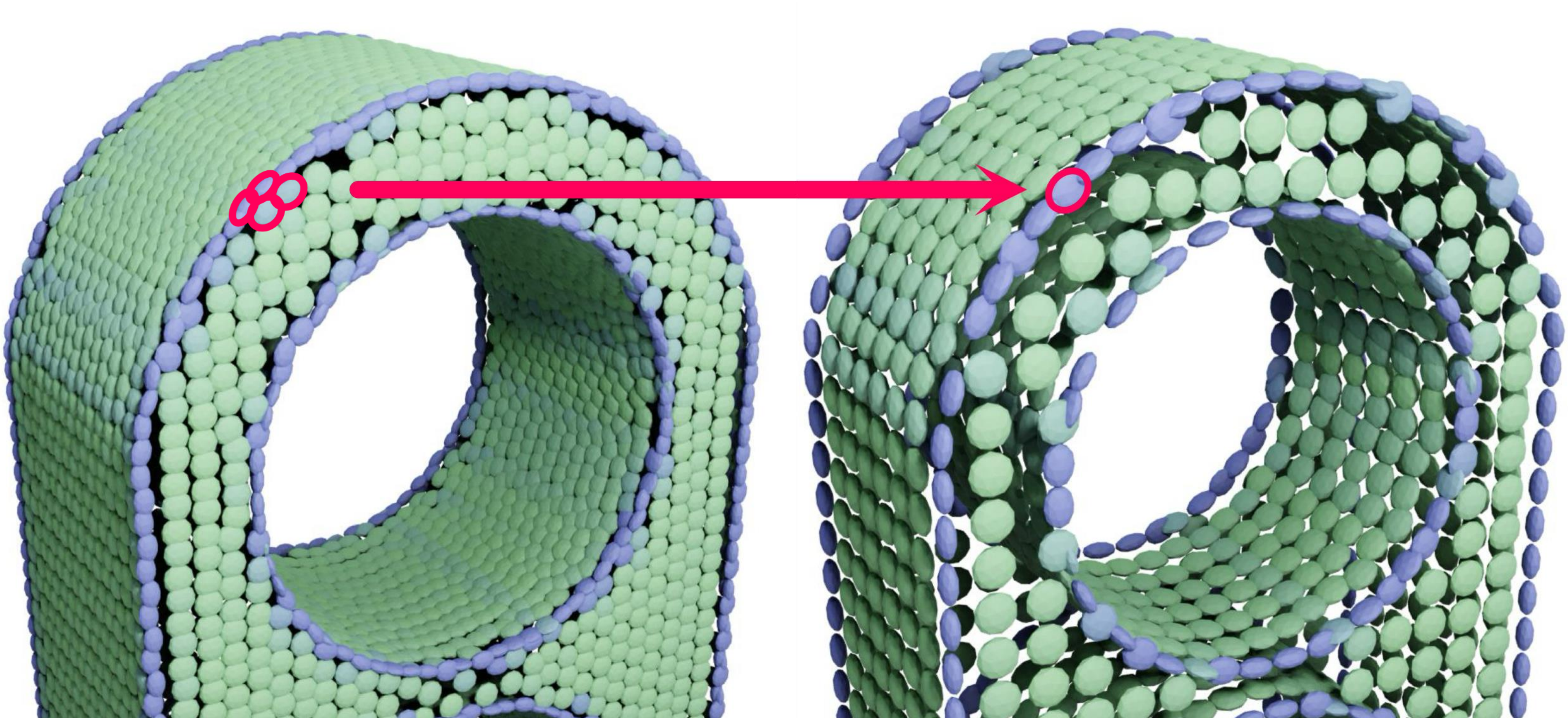


Multiple points $\{\mathbf{p}_i\}_{i=1..P}$ — Single point $\mathbf{p} = \frac{1}{P}\sum \mathbf{p}_i$

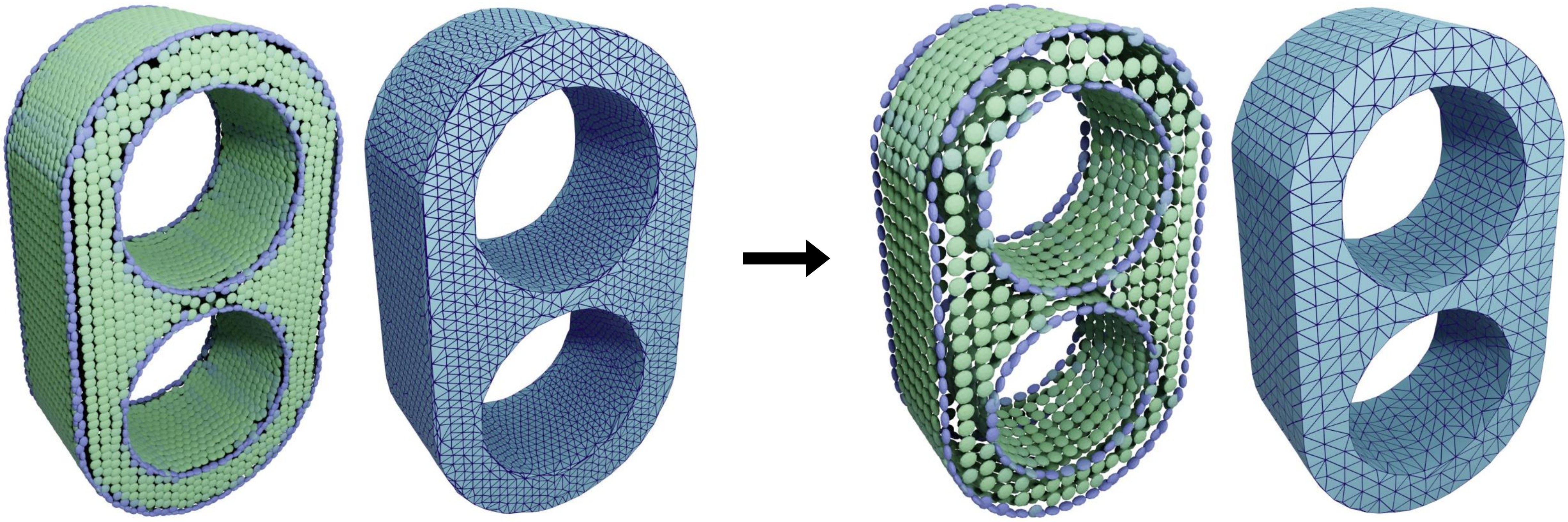
Multiple normals $\{\mathbf{n}_i\}_{i=1..P}$ — Single normal $\mathbf{n} = \frac{1}{P}\sum \mathbf{n}_i$

Multiple quadrics $\{\mathbf{Q}_i\}_{i=1..P}$ — Single quadric $\mathbf{Q} = \sum \mathbf{Q}_i$

Extension: PoNQ-lite



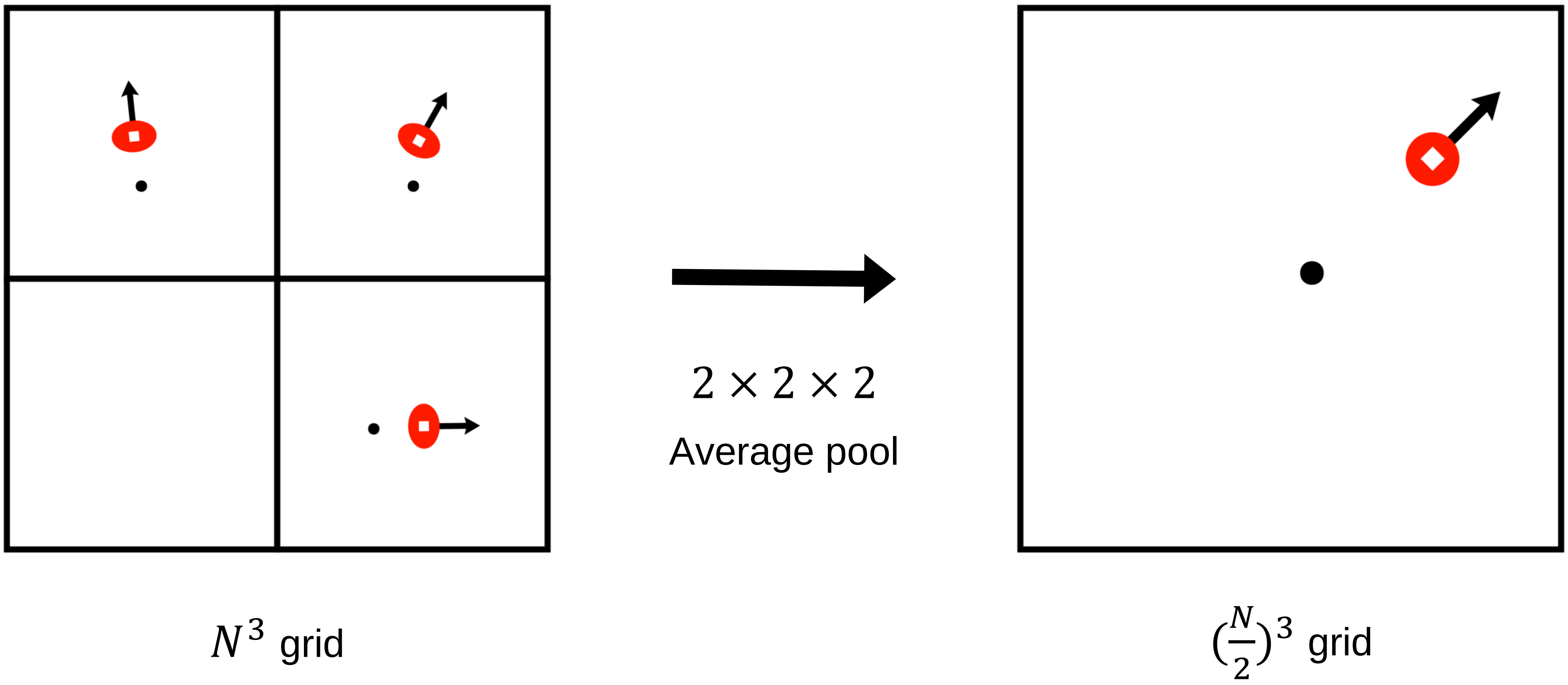
Extension: PoNQ-lite



PoNQ

PoNQ-lite

Extension: PoNQ-lite



Extension: PoNQ-lite

GPU-based decimation:

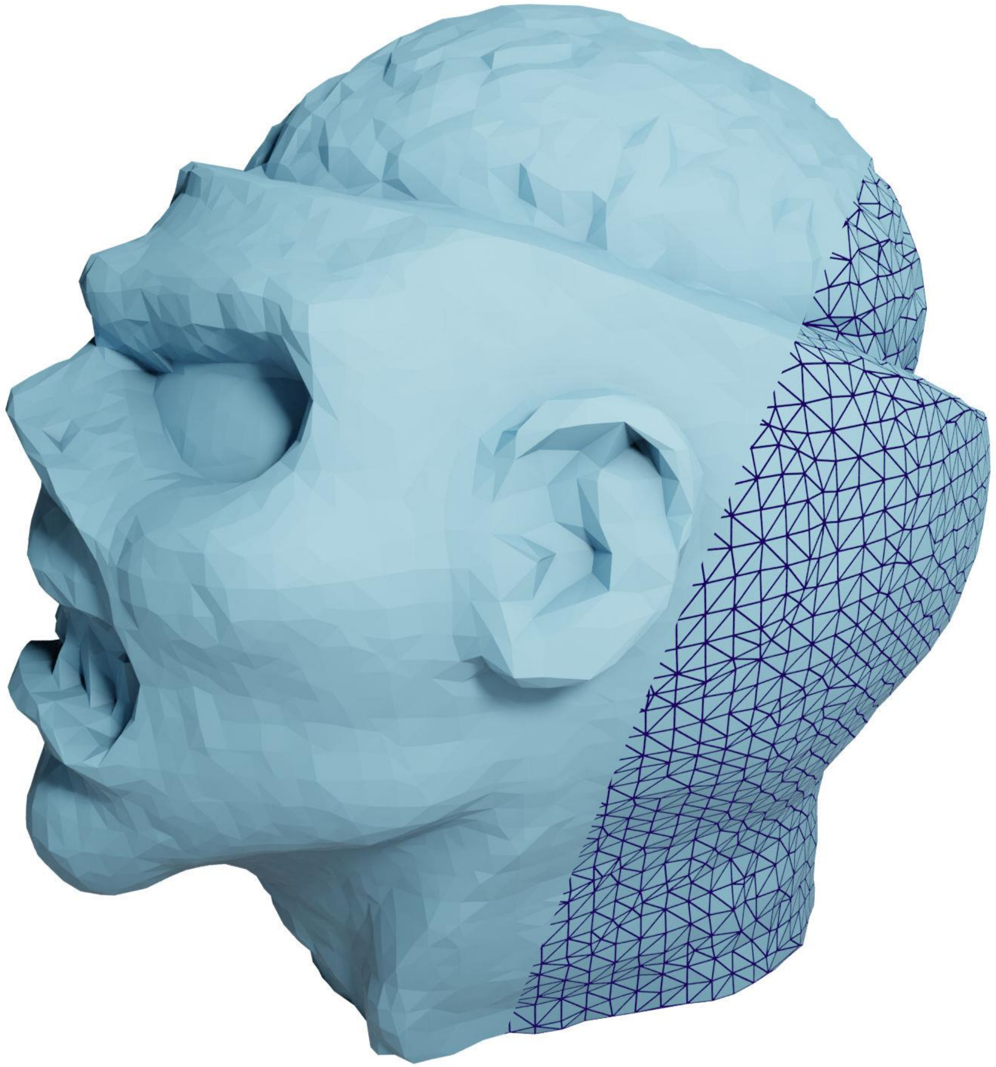
128^3 grid of predicted points,

normals and quadrics

Extension: PoNQ-lite

GPU-based decimation:

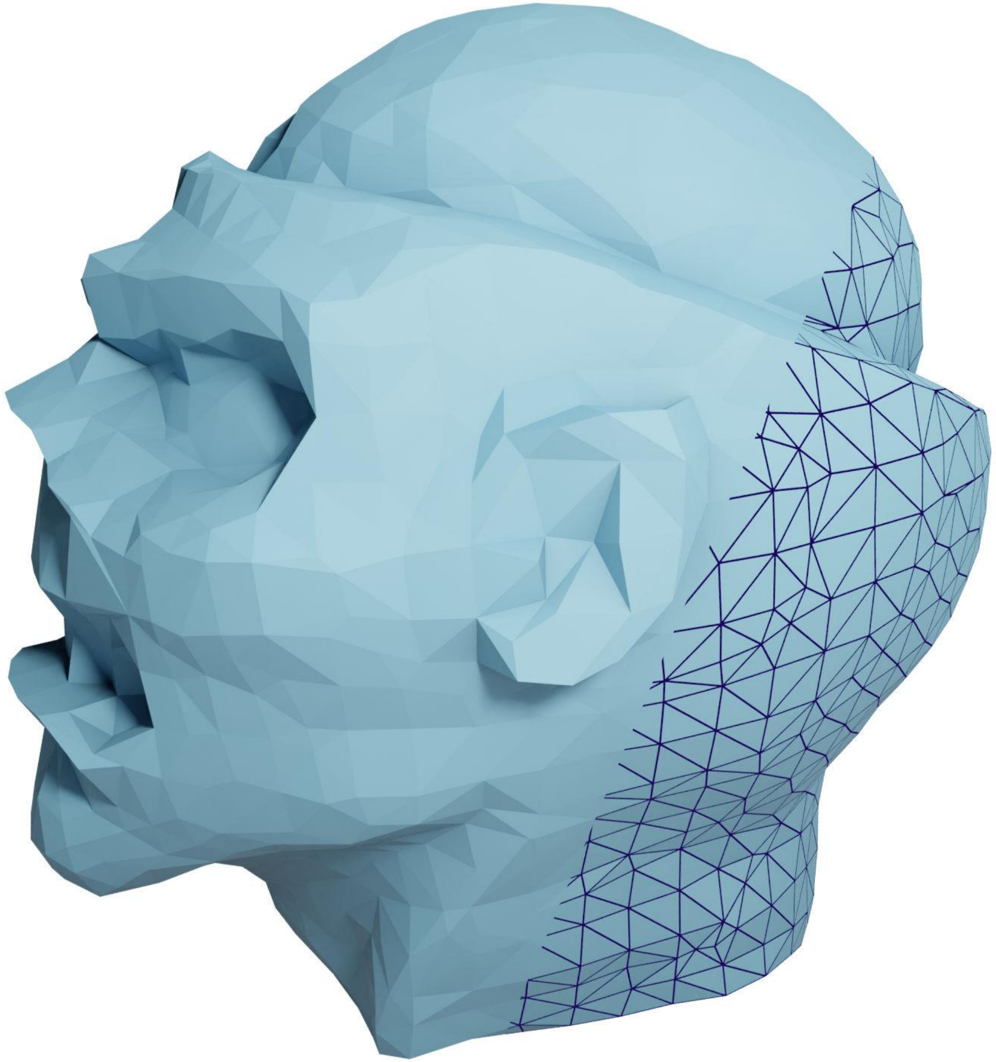
64^3 grid of predicted points, normals and quadrics



Extension: PoNQ-lite

GPU-based decimation:

32^3 grid of predicted points, normals and quadrics



Extension: PoNQ-lite

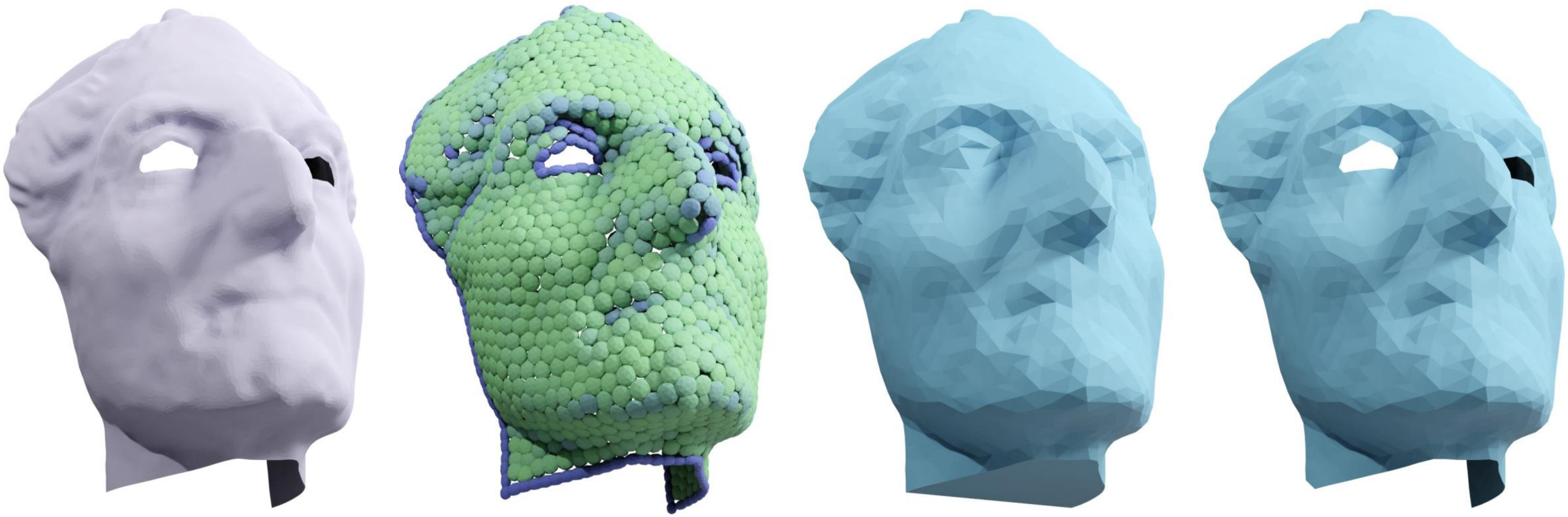
GPU-based decimation:

16^3 grid of predicted points,

normals and quadrics

Extension: Open Surfaces

Extension: Open Surfaces



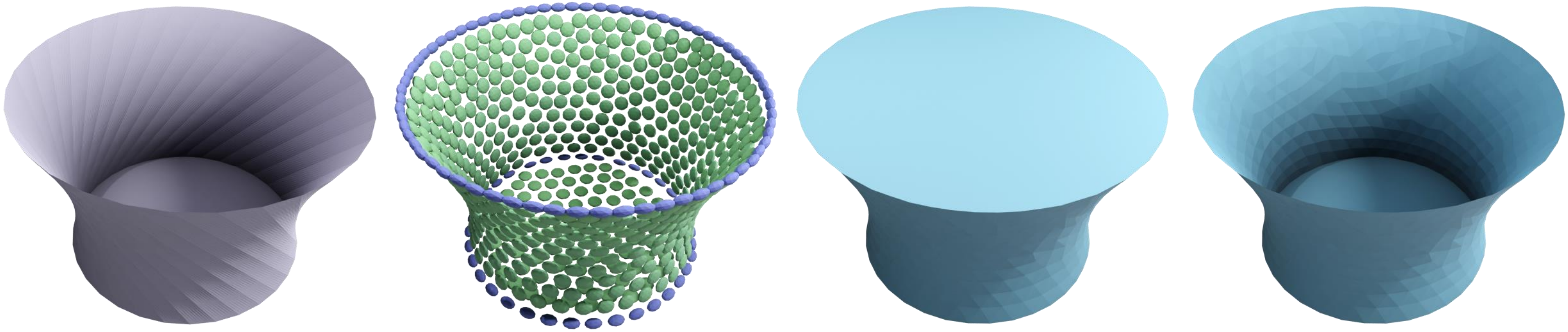
Gr. Truth

PoNQ

Closed Mesh

Open Mesh

Extension: Open Surfaces



Gr. Truth | PoNQ | Closed Mesh | Open Mesh

Comparison to NeuRBF [1]

PoNQ, 50k parameters
#V 0.05M

PoNQ, 500k parameters
#V 0.5M

NeuRBF [1], 50k parameters
#V 1.4M

[1] Z. Chen, et al. NeuRBF: A Neural Fields Representation with Adaptive Radial Basis Functions. In *CVPR*, 2023