Surface Reconstruction

2016 Spring

National Cheng Kung University

Instructors: Min-Chun Hu 胡敏君

Shih-Chin Weng 翁士欽 (西基電腦動畫)

Parts of the slides are from the course

"INF555 Digital Representation and Analysis of Shapes"

by Luca Castelli Aleardi and Maks Ovsjanikov

http://www.enseignement.polytechnique.fr/informatique/INF555/

Parts of the slides are from the 9th ACCV,

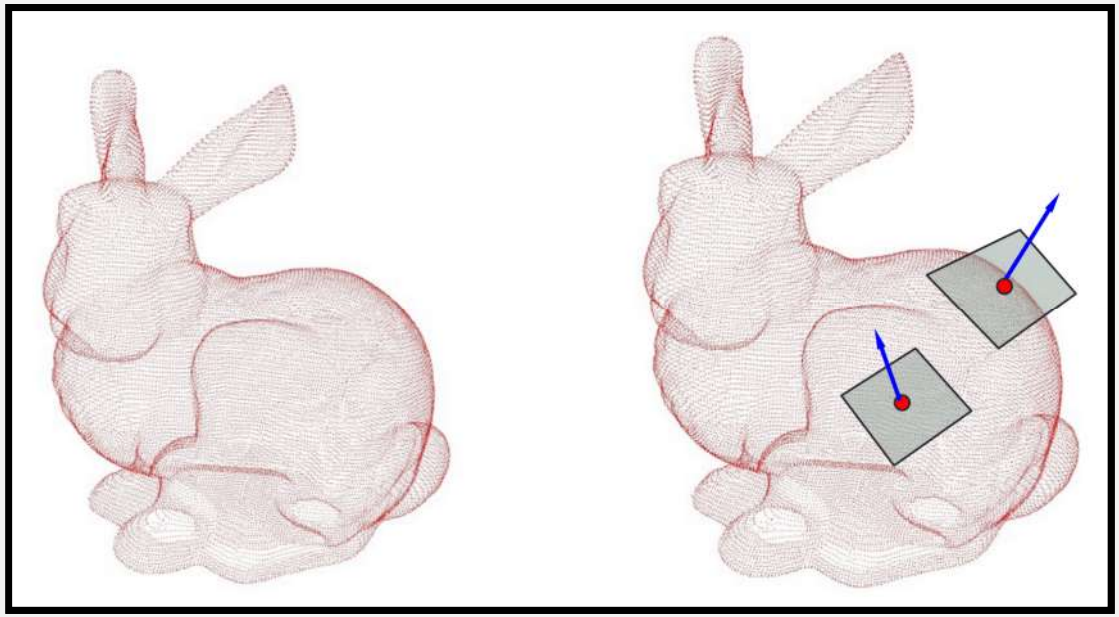
Xi'an China, Sep 25-27 2009

by Nader Salman, INRIA

http://www.cgal.org

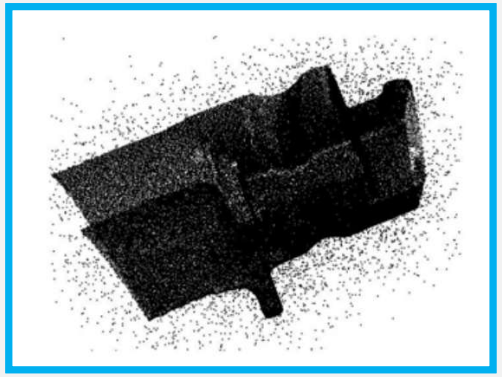
Point Clouds

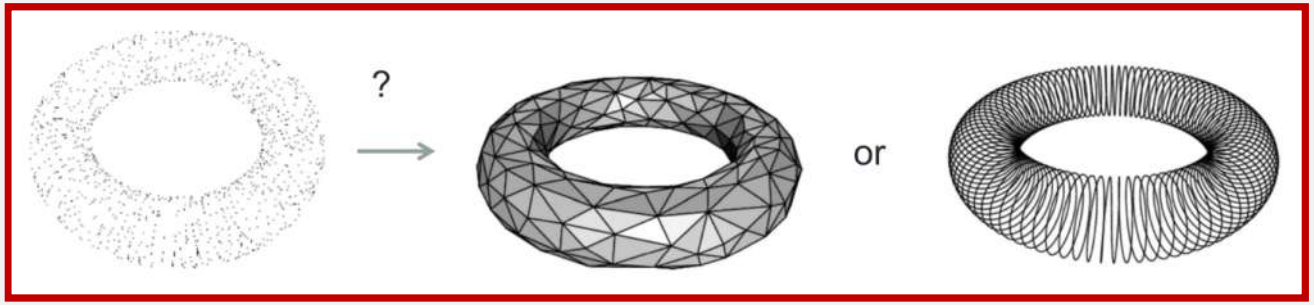
- Simplest representation: only points, no connectivity
- Collection of (x,y,z) coordinates, possibly with normals
- Points with orientation are called **surfels**



Limitations of Point Clouds

- Noise and outliers
- No simplification or subdivision
- No direct smooth rendering
- No topological information
- Weak approximation power





Why Point Cloud?

- That's the only thing that's available
  - Nearly all 3d scanning devices produce point clouds
- Sometimes, easier to handle (esp. in hardware)

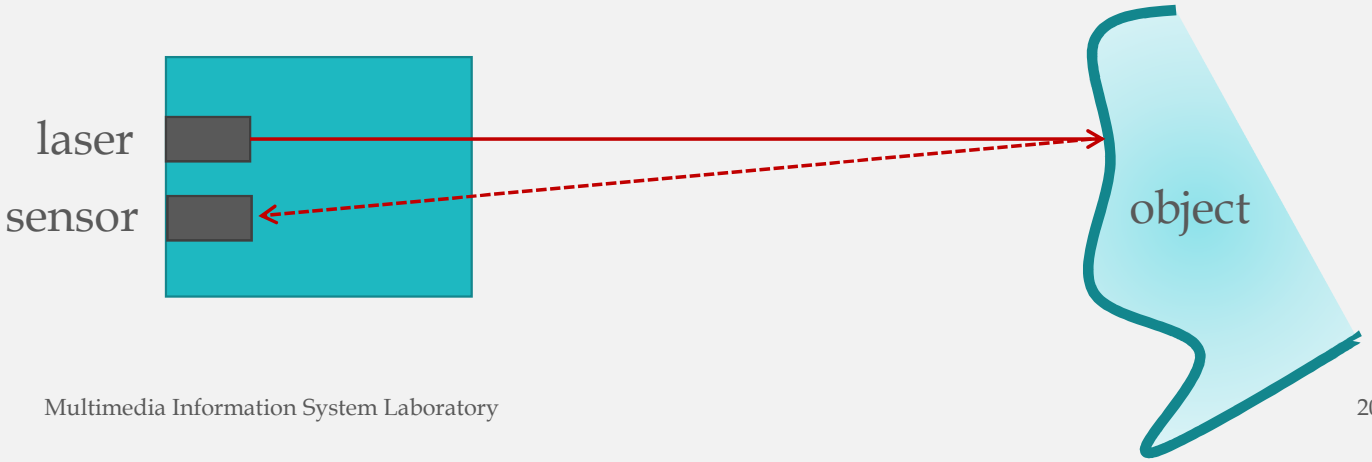
Major Types of 3d Scanners

- Range (emission-based) scanners
  - Time-of-flight laser scanner
  - Phase-based laser scanner
- Triangulation
  - Laser line sweep
  - Structured light
- Stereo / computer vision
  - Passive stereo
  - Active stereo / space time stereo

Time of Flight Scanners

1. Emit a short pulse of laser
2. Capture the reflection
3. Measure the time it took to come back

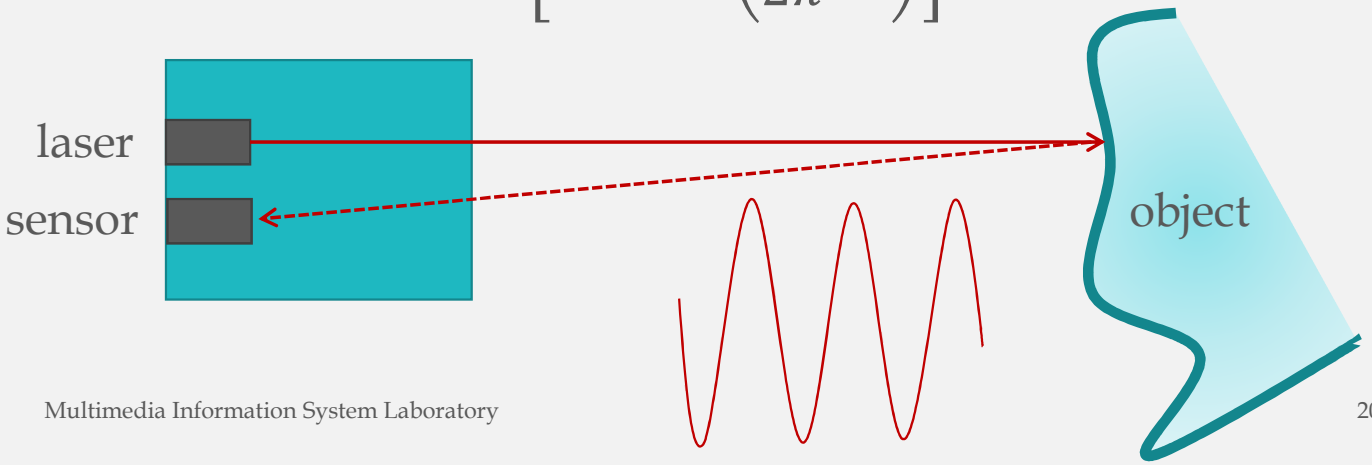
$$D = \frac{cT}{2} \qquad c = speed\ of\ light\ \approx 299792458\ m/s$$



Phase-based Range Scanners

1. Instead of a pulse, emit a continuous **phase-modulated** beam
2. Capture the reflection
3. Measure the **phase-shift** between the output and input signals

$$Output: e(t) = e \cdot \left[1 + \sin\left(\frac{F}{2\pi} \cdot t\right)\right] \qquad Input: s(t) = e \cdot \left[1 + \sin\left(\frac{F}{2\pi} \cdot t - \phi\right)\right]$$

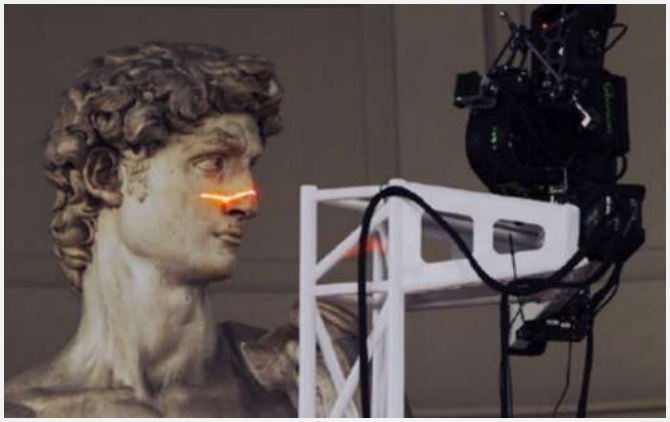


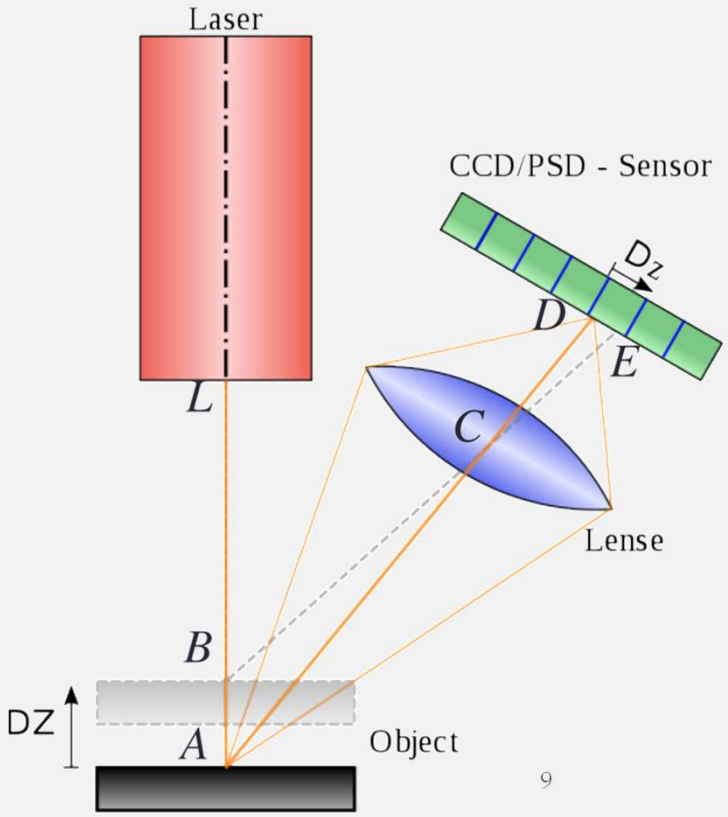
$$D = \frac{cT}{2} = \frac{c}{2f}\frac{\phi}{2\pi}$$

Triangulation-based Approaches

Intuition: the **depth** is related to the **shift** in the camera plane

1. Add a photometric sensor (e.g. camera)
2. Record the position for a reference plane
3. Change in recording position can be used to recover the depth



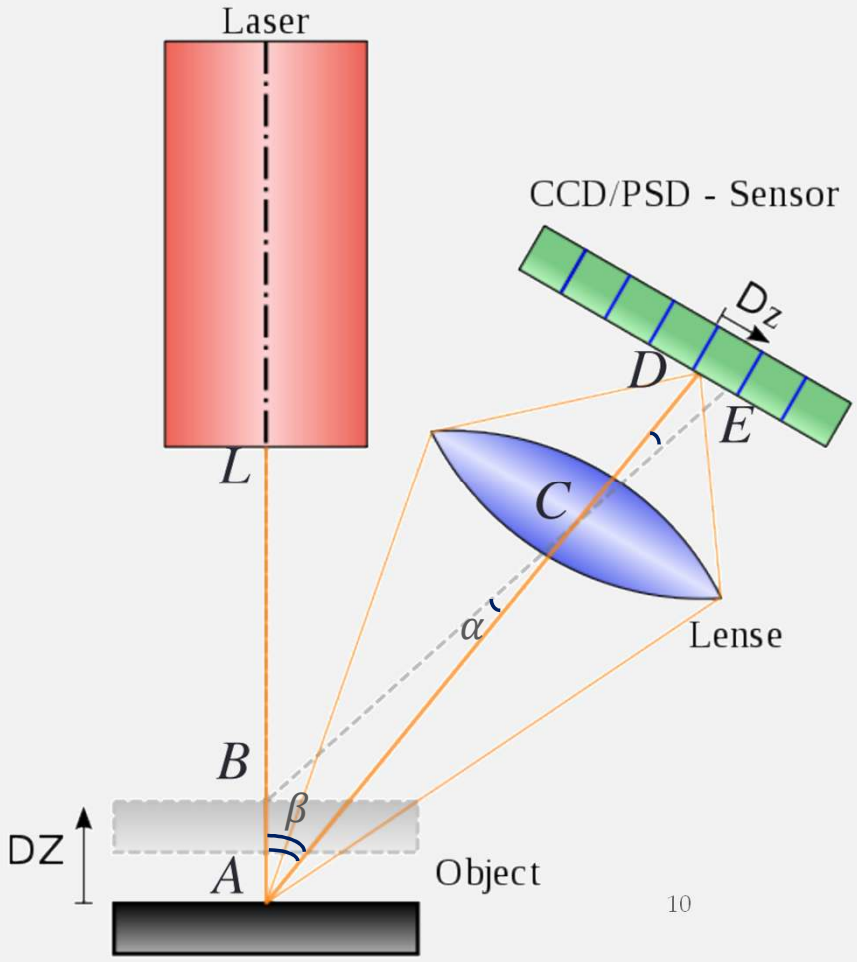


Triangulation-based Approaches (Cont.)

- Use Dz, DE and ∠CED, compute ∠ DCE = α
- Use α, β, and AC = AD – CD, compute BC, AB
- The depth LB = LA - AB

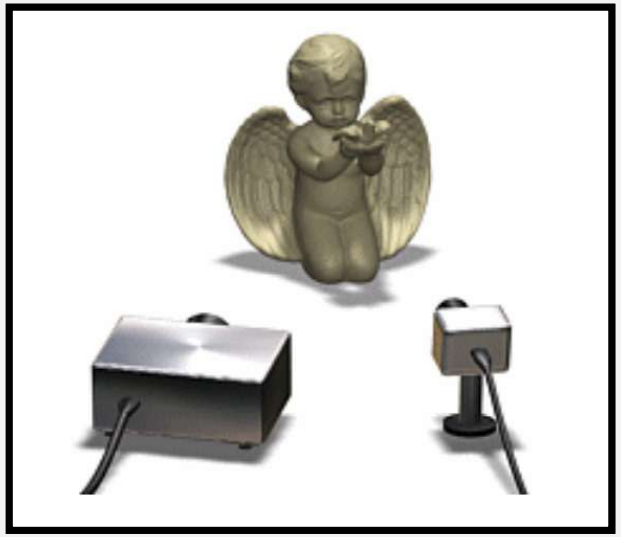
If well-calibrated, can lead to extremely accurate depth measurements.

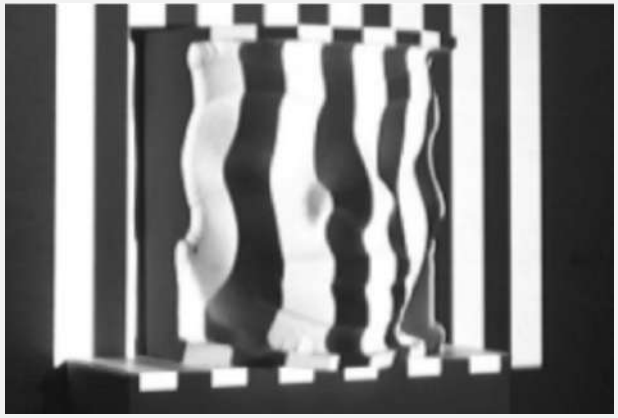
Main problem: slow and expensive !



Structured-Light Scanners

- Same general idea as Triangulation based scanner
- Replace laser with projector (Project stripes instead of sheets)
- Challenge: Need to identify which (input/output) lines correspond





Structured-Light Scanners (Cont.)

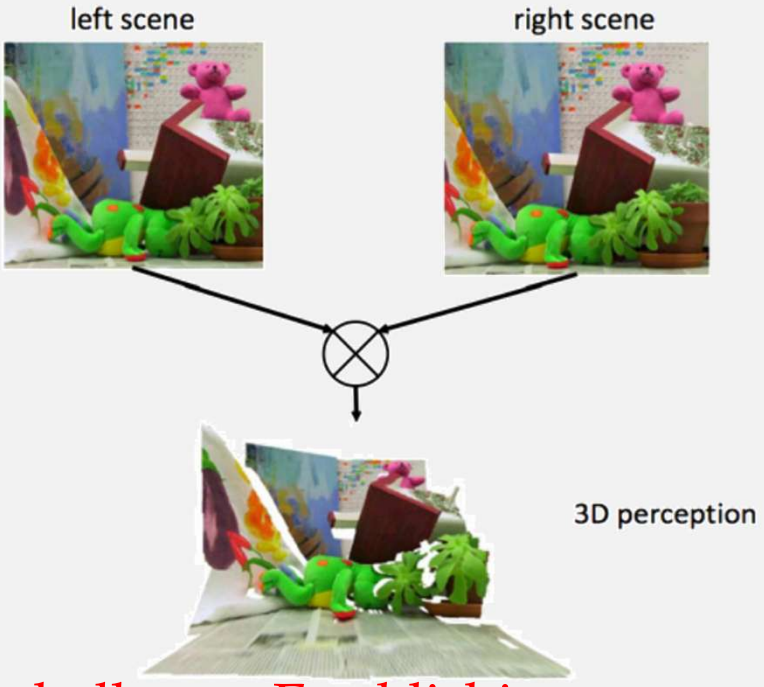
- Project **multiple stripes** to identify the position of a point

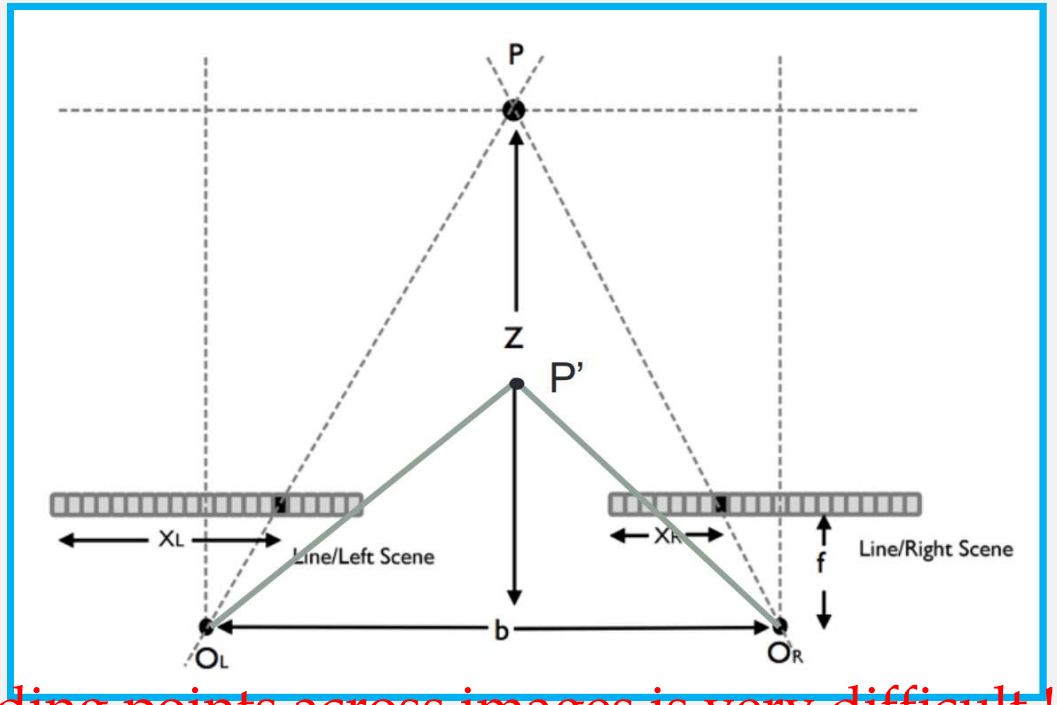
log(N) projections are sufficient to identify N stripes

Advantage: cost and speed
Disadvantage: need controlled conditions & projector calibration

Computer Vision Based Methods (1)

- Depth from stereo:
  - Given 2 images, shift in the x-axis is related to the depth



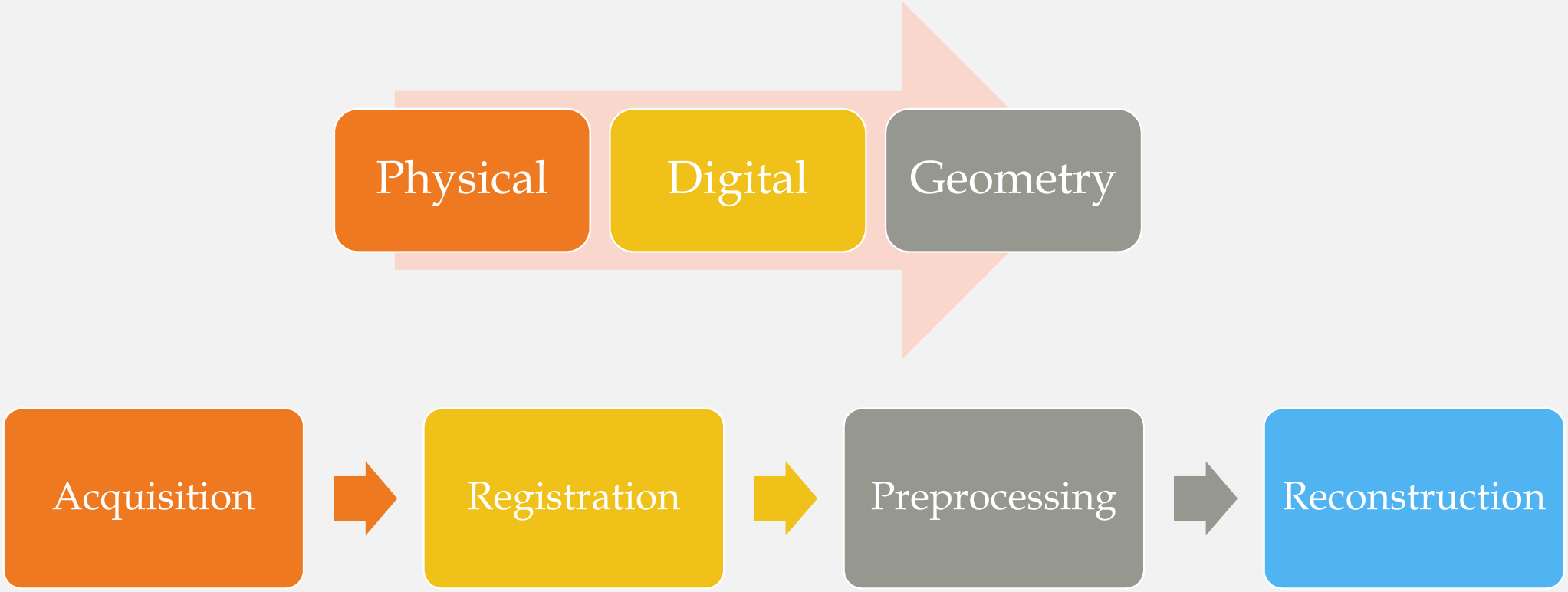


Main challenge: Establishing corresponding points across images is very difficult !

Computer Vision Based Methods (2)

- Depth from blur:
  - Depth can be approximated by detecting how blurry part of the image is (with known focal length)

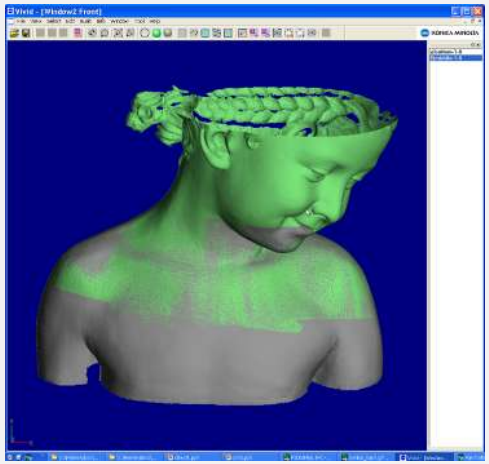
Common Pipeline

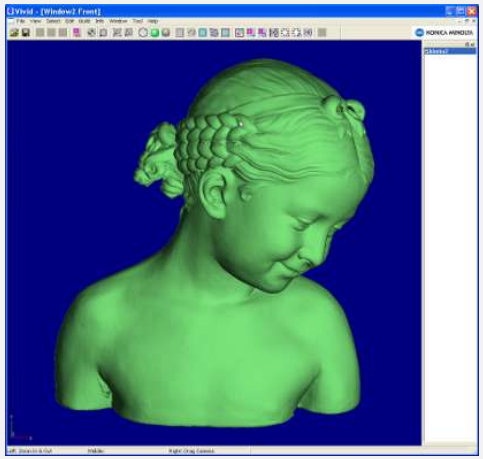


Common Pipeline (Example 1)

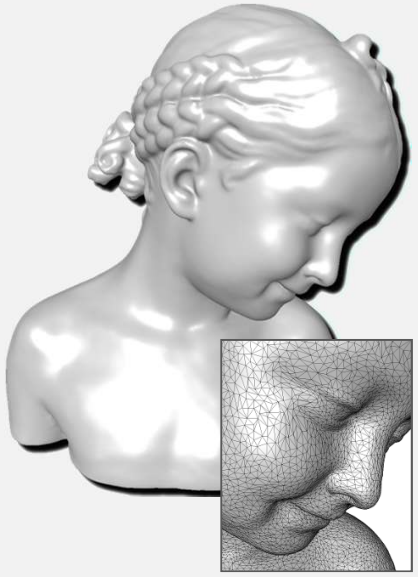
Acquisition → Registration → Preprocessing → Reconstruction

laser range data



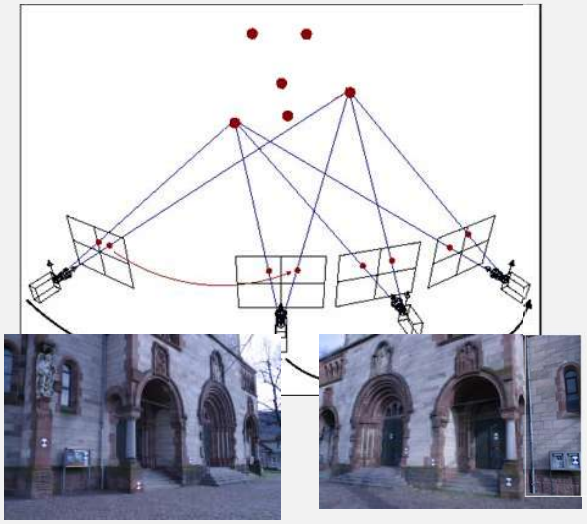


(simplification, denoising, smoothing, …)

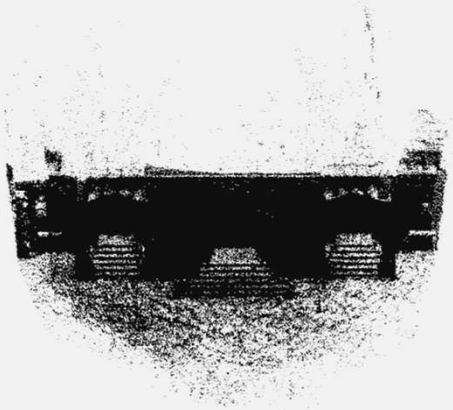


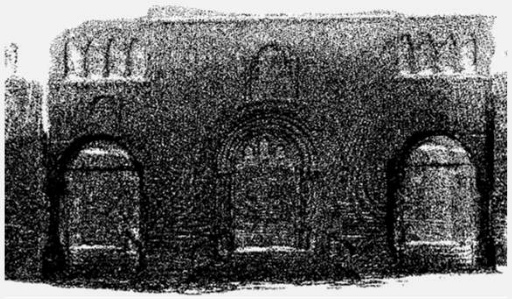
Common Pipeline (Example 2)

Acquisition & Calibration → Point Cloud Generation → Preprocessing → Reconstruction



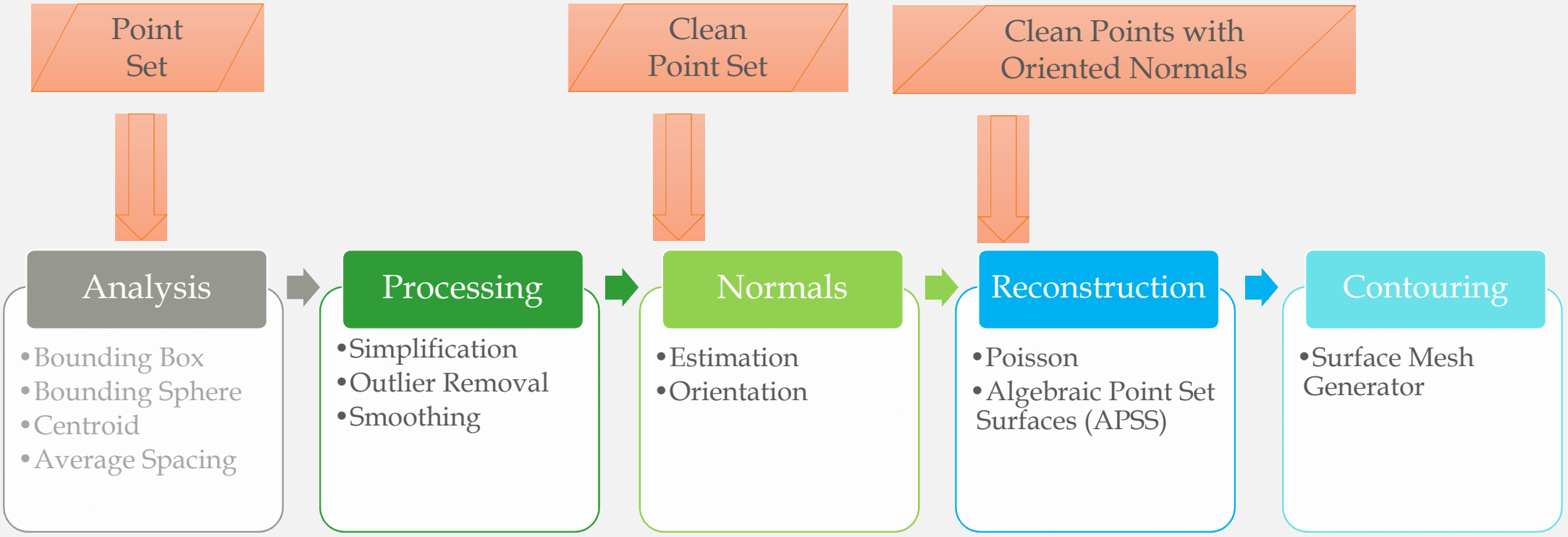
multi-view passive stereo



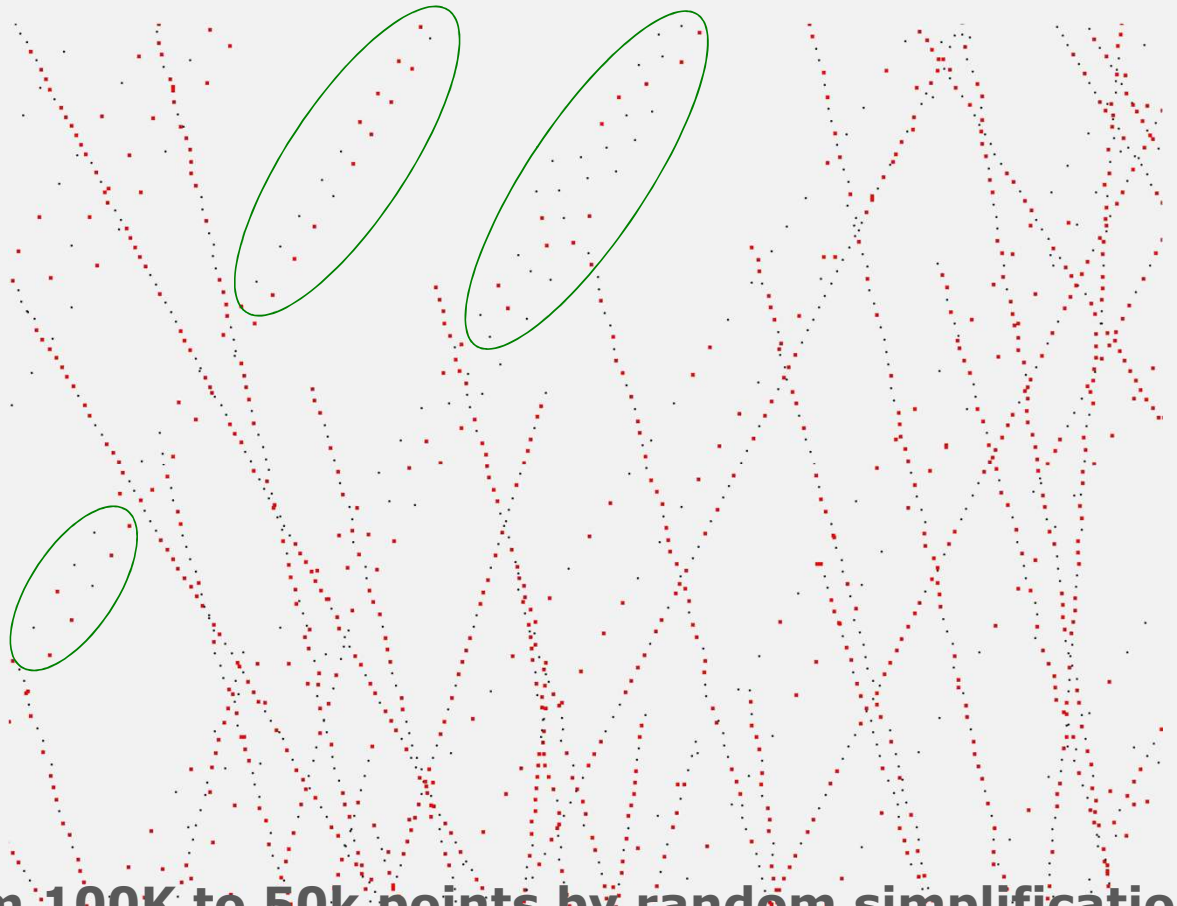


(simplification, denoising, smoothing, …)

Entering the Pipeline

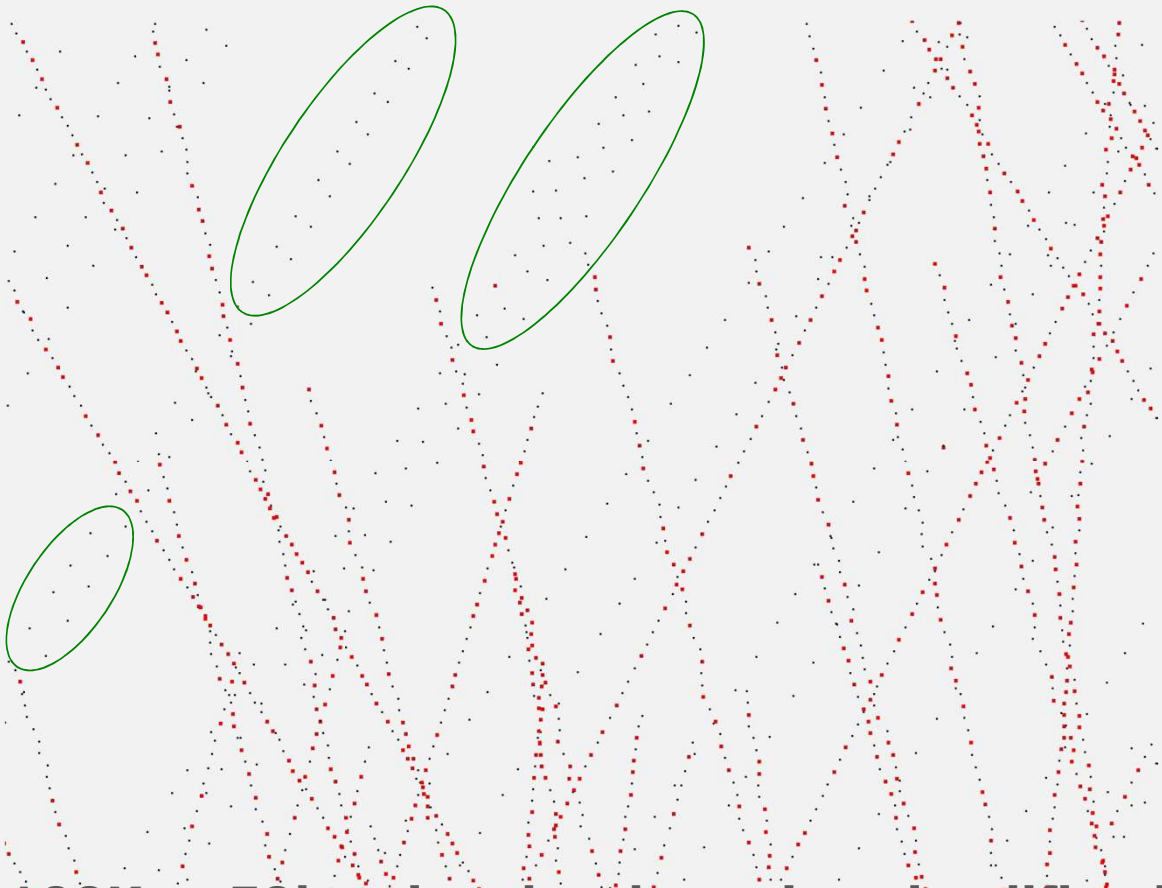


Random Simplification Example



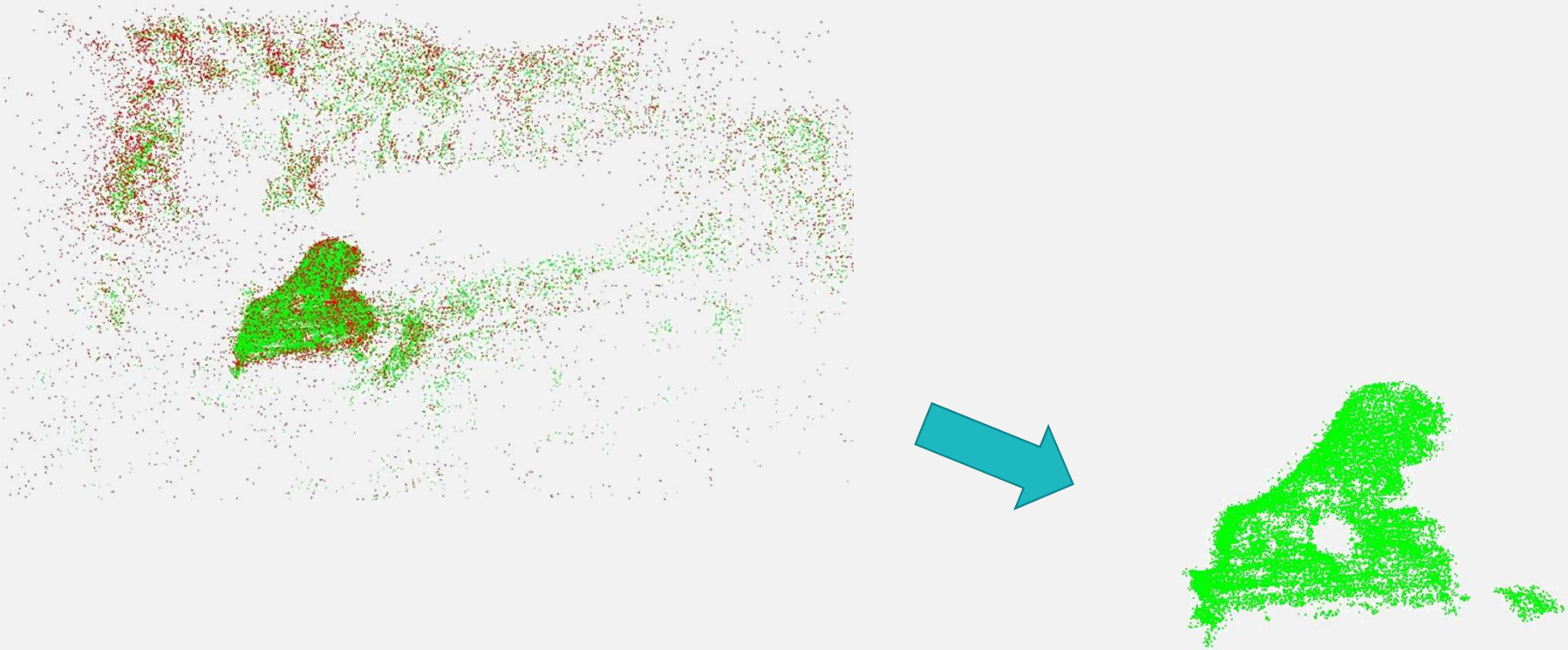
from 100K to 50k points by random simplification

Clustering Simplification Example



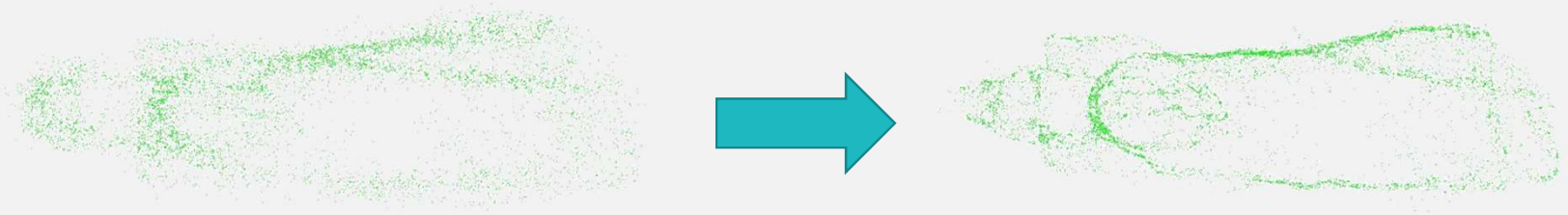
from 100K to 50k points by clustering simplification

Outlier Removal Example



Smoothing Example

- For each point
  - find KNN
  - fit jet (smooth parametric surface)
  - project onto jet



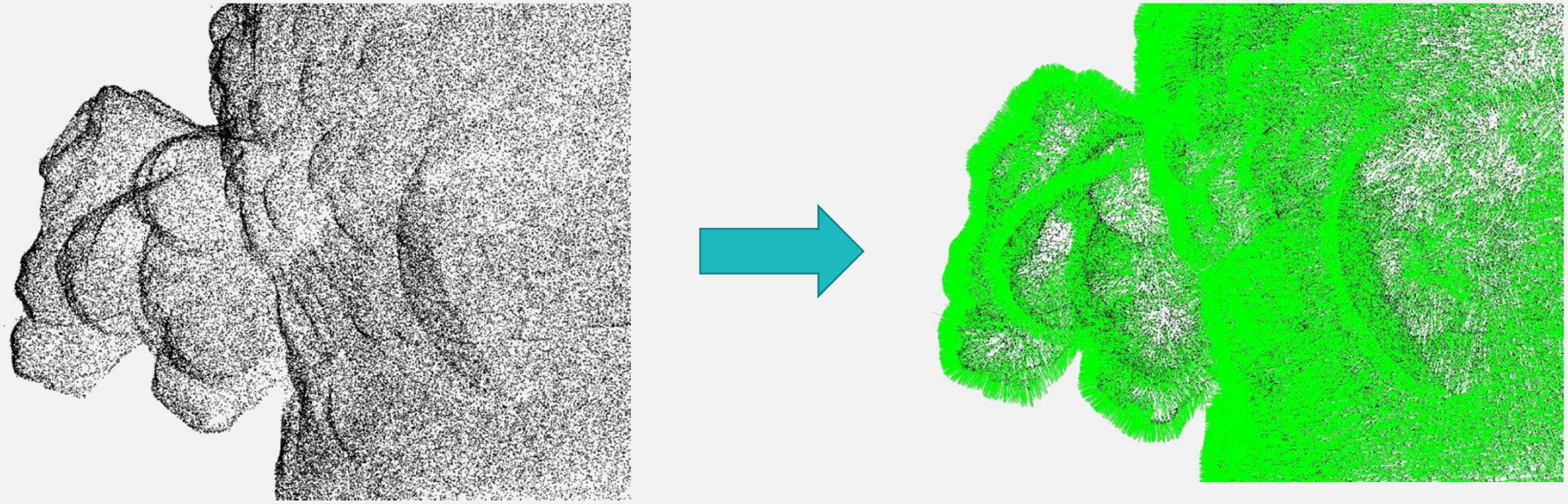
Noisy Point Set **Smooth Point Set**

Normals

- Estimation (no orientation)
  - KNN + PCA (fit a plane)
  - KNN + jet fitting (better for noisy data)

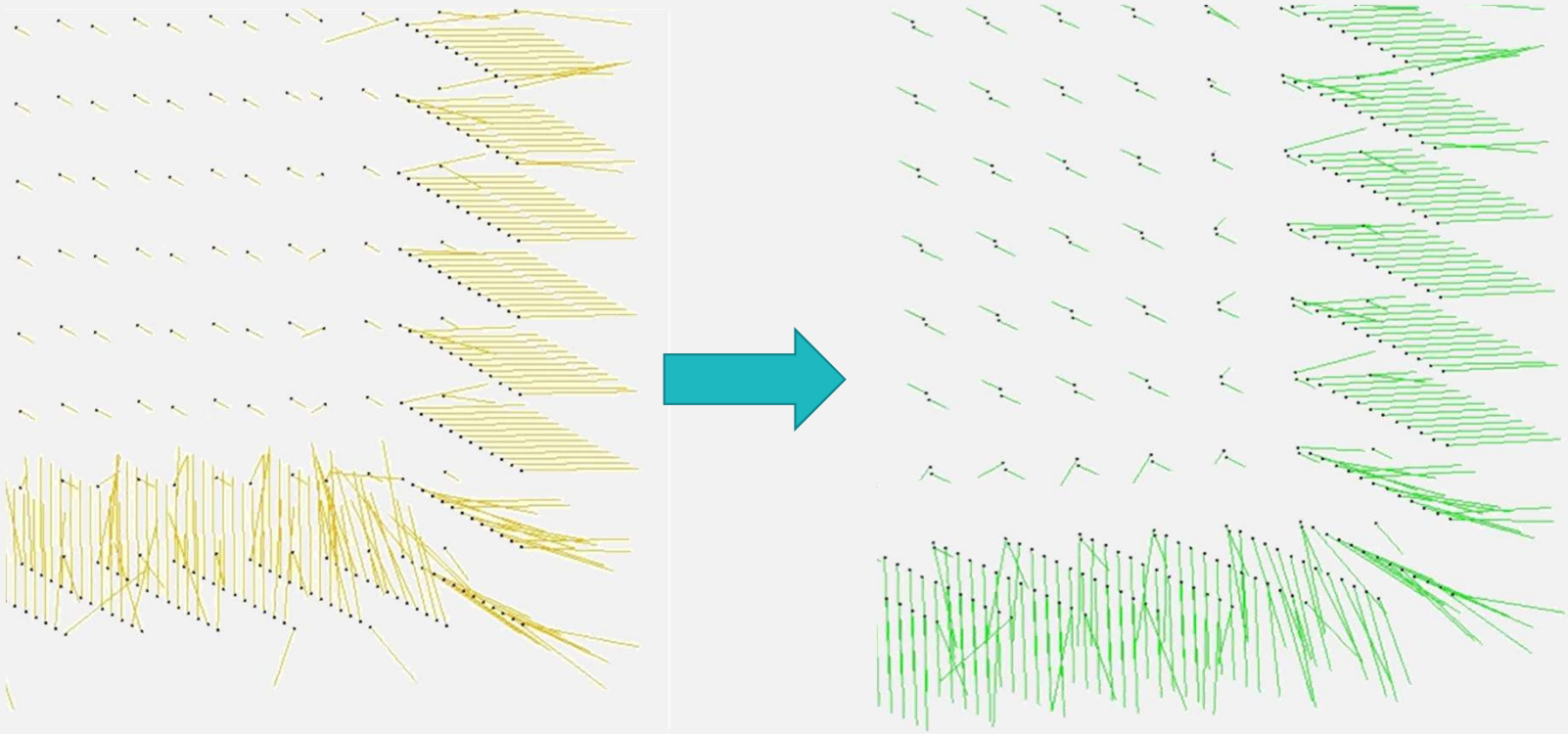
- Orientation
  - KNN + BGL MST (Minimum Spanning Tree) [Hoppe 92]

Normal Estimation Example



Normals estimation through PCA (7 KNN)
Orientation through MST (7 KNN)

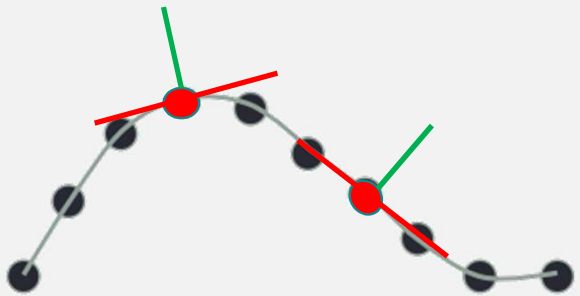
Normal Orientation Example



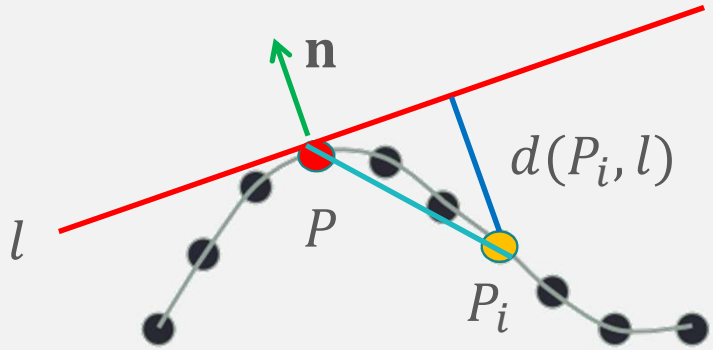
Normal Orientation through MST

Normal Estimation

- Assume we have a clean sampling of the surface, our goal is to find the best approximation of the tangent direction, and thus the normal to the line.



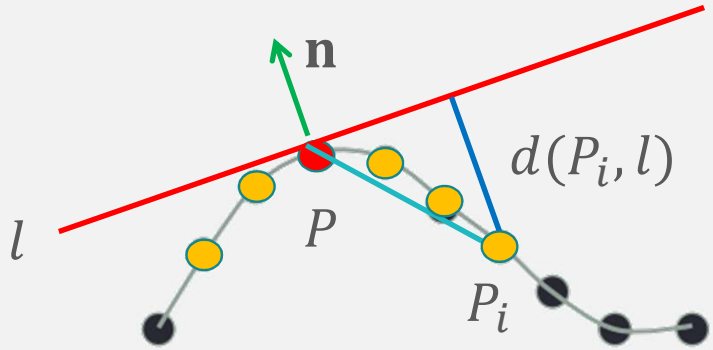
Normal Estimation (Cont.)



- Given line l through P with normal $\mathbf{n}$, for another point P_i :

$$d(P_i, l)^2 = \frac{\left((P_i - P)^T \mathbf{n}\right)^2}{\mathbf{n}^T \mathbf{n}} = \left((P_i - P)^T \mathbf{n}\right)^2 \text{ if } \|\mathbf{n}\| = 1$$

Normal Estimation (Cont.)



- Find the **n** minimizing $\sum_{i=1}^{k} d(P_i, l)^2$ for a set of k points (k nearest neighbors of P)

$$\mathbf{n}_{opt} = \arg \min_{\|n\|=1} \sum_{i=1}^{k} \left((P_i - P)^T \mathbf{n} \right)^2$$

Normal Estimation (Cont.)

- Use Lagrange Multiplier:

$$\frac{\partial}{\partial \mathbf{n}}\left(\sum_{i=1}^{k}\left((P_i - P)^T \mathbf{n}\right)^2\right) - \lambda \frac{\partial}{\partial \mathbf{n}}\left(\mathbf{n}^T \mathbf{n}\right) = 0$$

$$\Longrightarrow \sum_{i=1}^{k} 2(P_i - P)(P_i - P)^T \mathbf{n} = 2\lambda \mathbf{n}$$

$$\Longrightarrow \left(\sum_{i=1}^{k} (P_i - P)(P_i - P)^T\right)\mathbf{n} = \lambda \mathbf{n}$$

$$\Longrightarrow C\mathbf{n} = \lambda \mathbf{n}$$

Normal Estimation (Cont.)

$$C\mathbf{n} = \lambda\mathbf{n} \qquad C = \sum_{i=1}^{k}(P_i - P)(P_i - P)^T$$

➔The normal **n** must be an eigenvector of the matrix ***C***.

$$\mathbf{n}_{opt} = \arg\min_{\|n\|=1} \sum_{i=1}^{k}\left((P_i - P)^T\mathbf{n}\right)^2 = \arg\min_{\|n\|=1} \mathbf{n}^T C\mathbf{n}$$

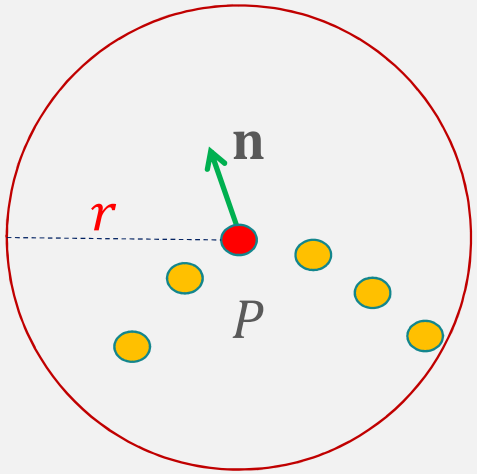
➔ $\mathbf{n}_{opt}$ must be the eigenvector corresponding to the smallest eigenvalue of ***C***.

Normal Estimation (Cont.)

- Can be done by PCA:
  1. Given a point P in the point cloud, find its k nearest neighbors
  2. Compute
$$C = \sum_{i=1}^{k} (P_i - P)(P_i - P)^T$$
  3. **n** should be the eigenvector corresponding to the smallest eigenvalue of C
- Variant:
  - Use
$$C = \sum_{i=1}^{k} (P_i - \bar{P})(P_i - \bar{P})^T, \qquad \bar{P} = \frac{1}{k}\sum_{i=1}^{k} P_i$$

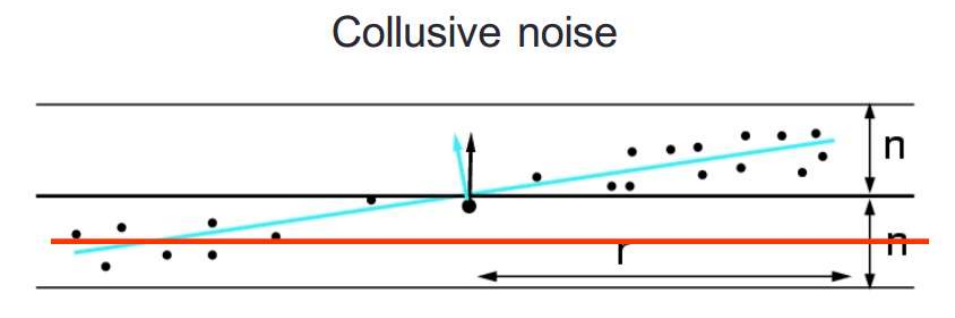
Parameter Selection

- Critical parameter: k
- Because of uneven sampling typically fix a radius r, and use all points inside a ball of radius r.
- How to pick an optimal r?

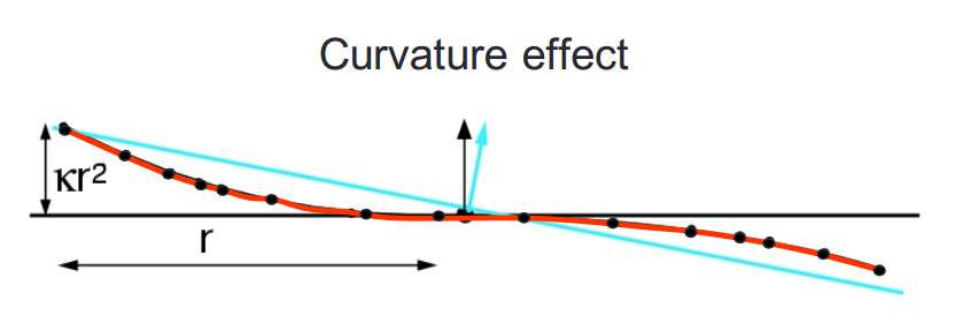


Parameter Selection (Cont.)

- Because of noise in the data, small *r* may lead to underfitting:

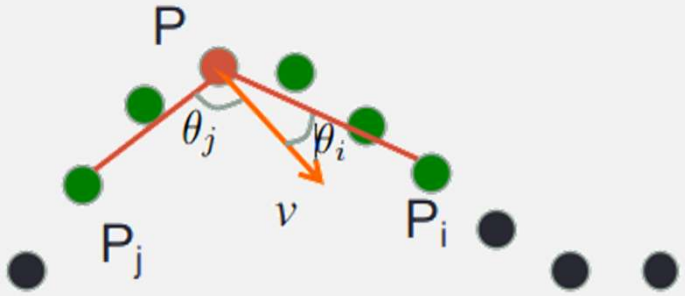


- Due to curvature, large r can lead to estimation bias:



Outlier Removal

- Goal : Remove points that do not lie close to a surface

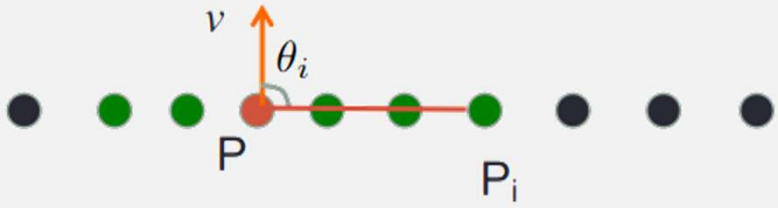


The covariance matrix $C = \sum_{i=1}^{k}(P_i - P)(P_i - P)^T$

For any vector v, the Releigh quotient:

$$\frac{v^T C v}{v^T v} = \sum_{i=1}^{k}\left((P_i - P)^T v\right)^2 = \sum_{i=1}^{k}(\|P_i - P\|\cos\theta_i)^2 \quad if\ \|v\| = 1$$

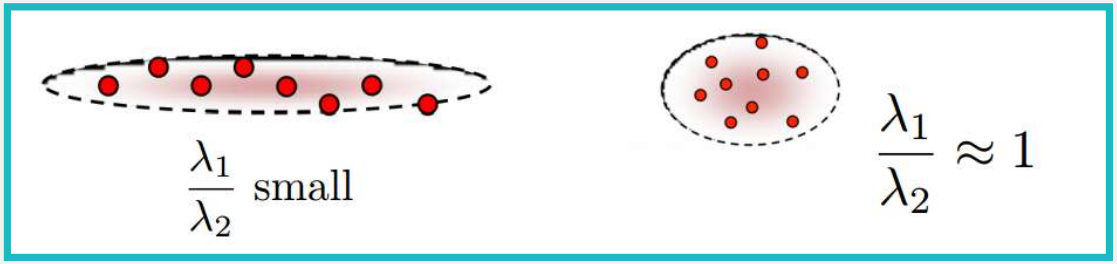
Outlier Removal (Cont.)



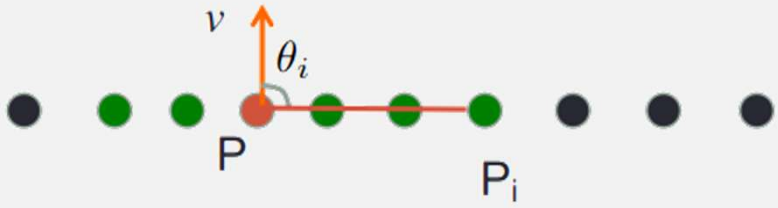
- If all the points are on a line, then $\min_v \frac{v^T C v}{v^T v} = \lambda_{min}(C) = 0$ and $\lambda_{max}(C)$ is large

➔There exists a direction along which the point cloud has no variability.

- If points are scattered randomly, then $\lambda_{max}(C) \approx \lambda_{min}(C)$



Outlier Removal (Cont.)



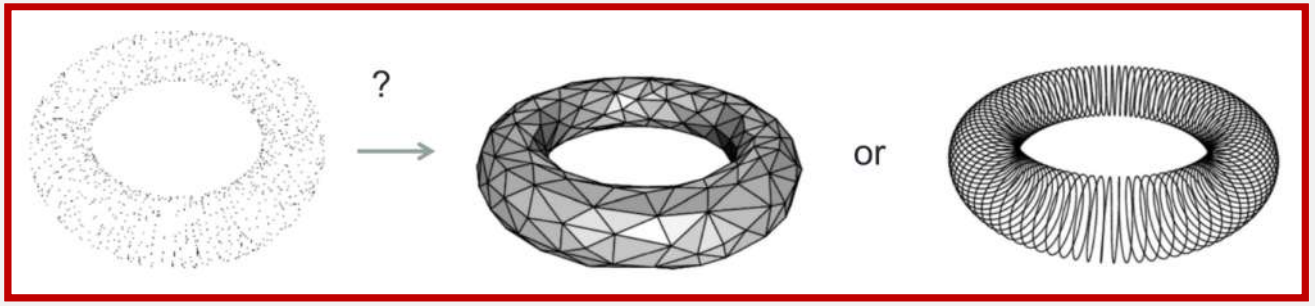
- If all the points are on a line, then $\min_v \frac{v^T C v}{v^T v} = \lambda_{min}(C) = 0$ and $\lambda_{max}(C)$ is large

➔There exists a direction along which the point cloud has no variability.

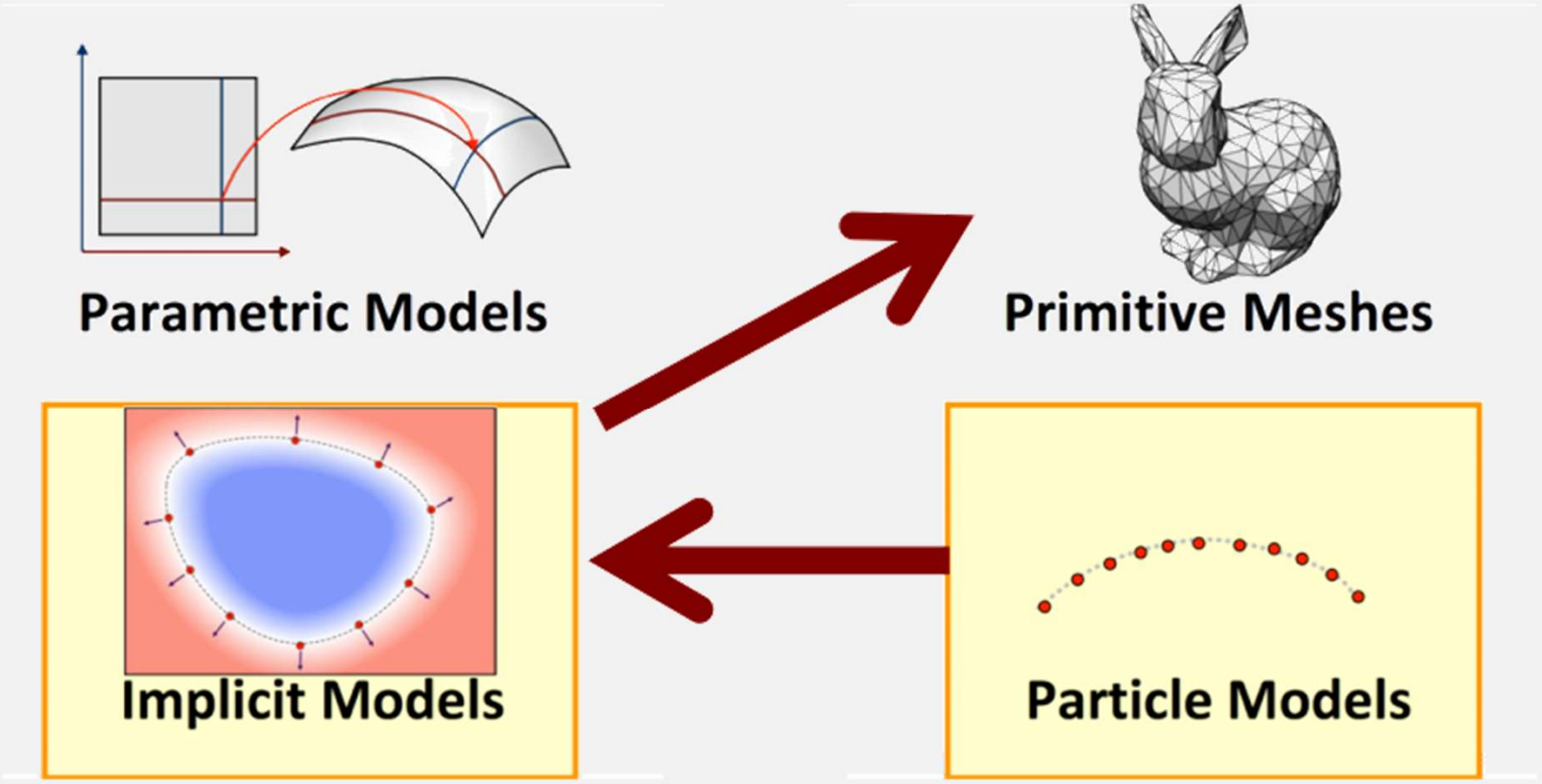
- If points are scattered randomly, then $\lambda_{max}(C) \approx \lambda_{min}(C)$
- Thus, can remove points where $\frac{\lambda_1}{\lambda_2} > \epsilon$ for some threshold
- In 3D we expect two zero eigenvalues, so use $\frac{\lambda_2}{\lambda_3} > \epsilon$ for for some threshold

Surface Reconstruction

- Goal: Construct a polygonal (e.g. triangle mesh) representation of the point cloud
- Main Problem: Data is **unstructured**
  - E.g. the points are not ordered
- An inherently **ill-posed** (a.k.a. difficult) problem

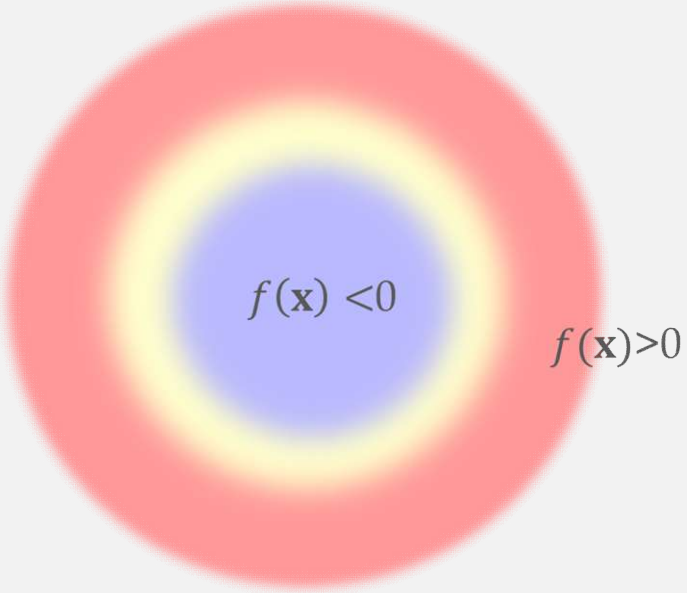


Reconstruction Through Implicit Models



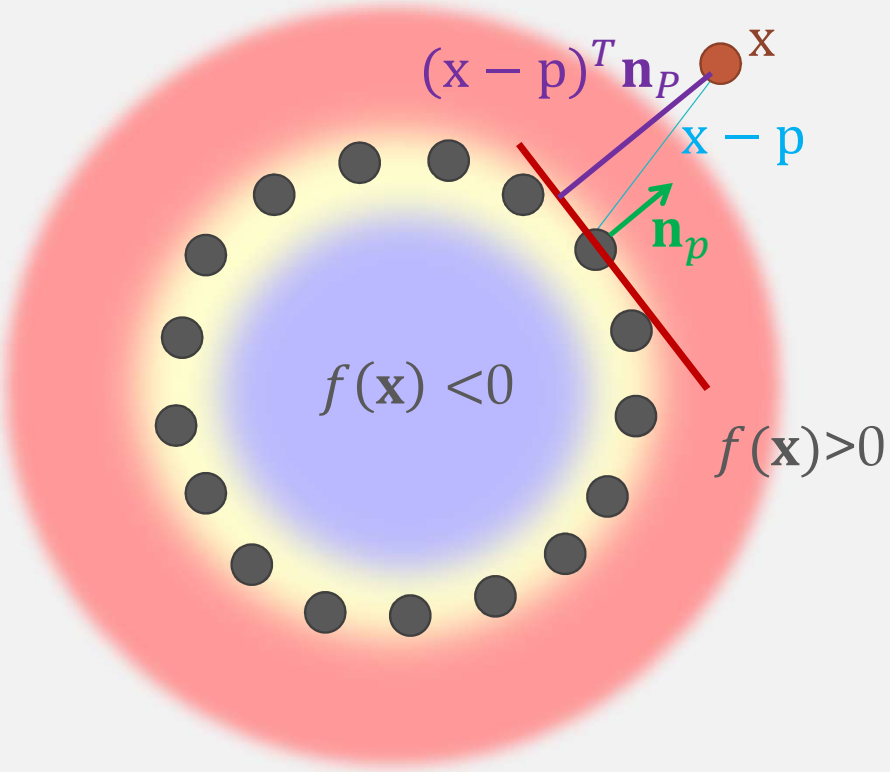
Implicit Surfaces

- Given a function f(**x**), the surface is defined as $\{\mathbf{x}, \text{s.t.} f(\mathbf{x}) = 0\}$



$$f(x, y) = x^2 + y^2 - r^2$$

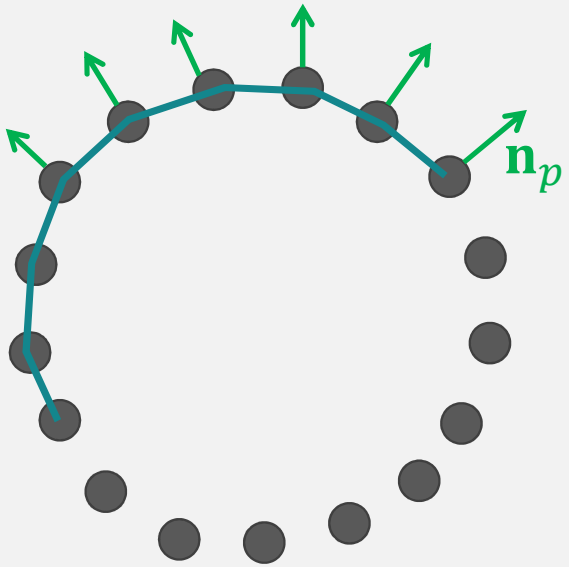
Implicit Surfaces (Cont.)



Set $f(\mathbf{x}) = (\mathrm{x}-\mathrm{p})^T\mathbf{n}_P$

= Signed distance to the tangent plane

Implicit Surfaces (Cont.)



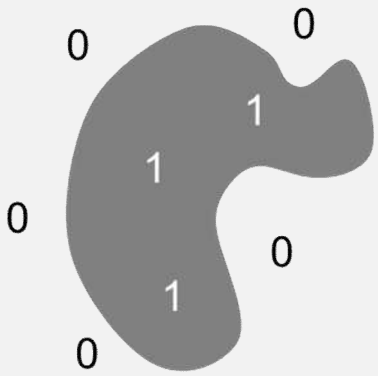
Set $f(\mathbf{x}) = (\mathrm{x} - \mathrm{p})^T \mathbf{n}_P$

- Need consistently oriented normals , but PCA only gives normals **up to orientation**.
- In general, difficult problem, but can try to locally connect points and fix orientations.

Poisson Surface Reconstruction

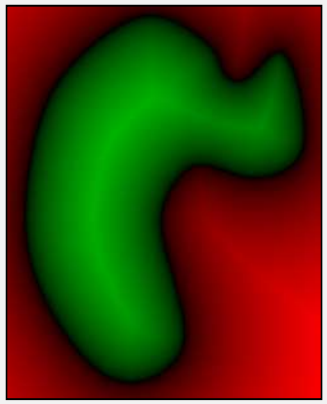
- Poisson surface reconstruction [Kazhdan-Bolitho-Hoppe, SGP 2006]
  - Reconstruct the surface of the model by solving for the **indicator function** of the shape.
  - Isosurface extracted by CGAL surface mesh generator

$$\chi_M(p) = \begin{cases} 1 & \text{if } p \in M \\ 0 & \text{if } p \notin M \end{cases}$$



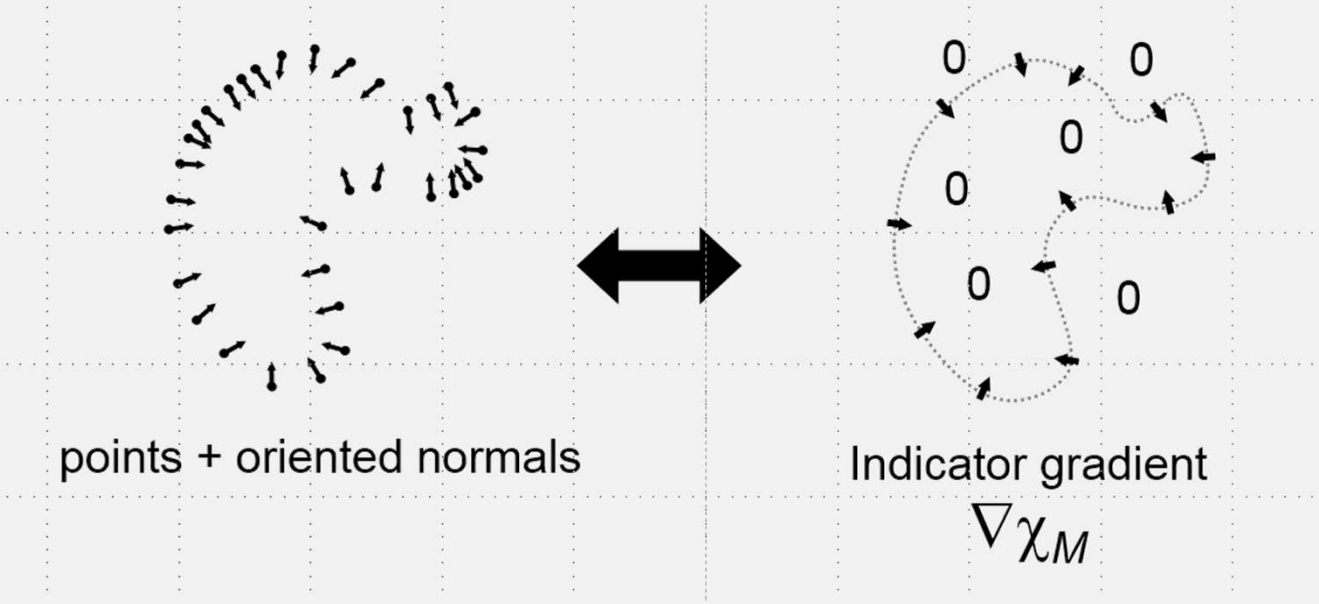
Indicator function

χ_M



Poisson Surface Reconstruction (Cont.)

- There is a relationship between the normal field and gradient of indicator function



Poisson Surface Reconstruction (Cont.)

- Represent the points by a vector field $\vec{V}$
- Find the function χ whose gradient best approximates $\vec{V}$:

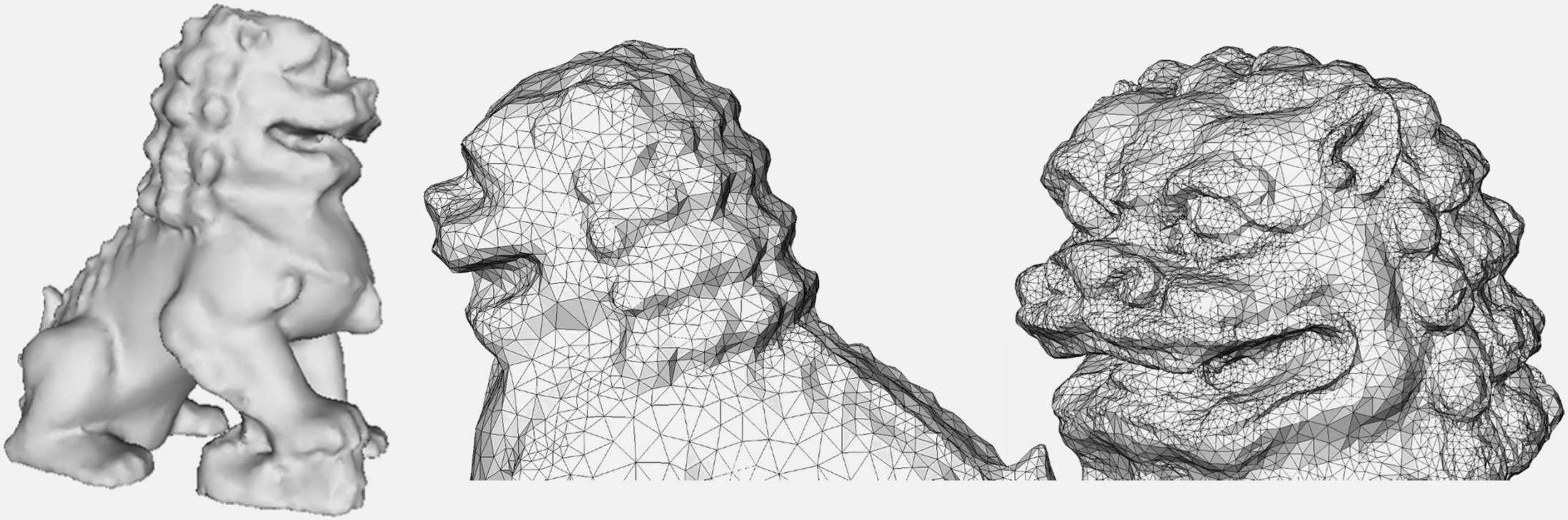
$$\min_{\chi} \left\| \nabla\chi - \vec{V} \right\|$$

- Applying the **divergence operator**, we can transform this into a **Poisson** problem:

$$\nabla \cdot (\nabla\chi) = \nabla \cdot \vec{V} \quad \Leftrightarrow \quad \Delta\chi = \nabla \cdot \vec{V}$$

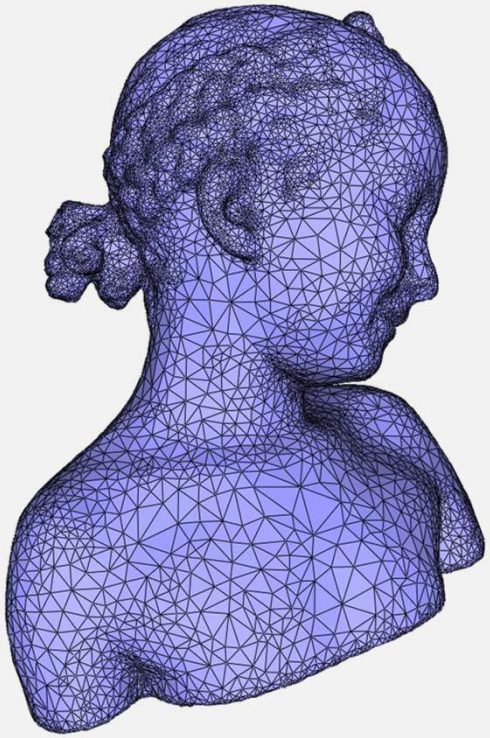
- We solve for the Poisson equation onto the vertices of a (refined) 3D Delaunay triangulation [TAUCS linear solver]

Poisson Surface Reconstruction Example



Poisson Surface Reconstruction Example

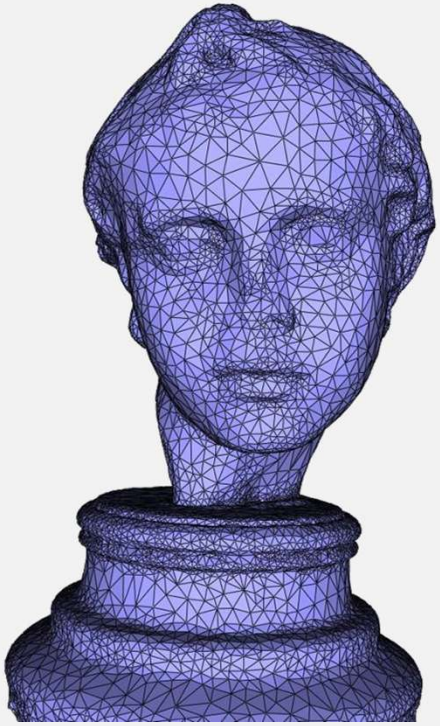
120K points sampled on child statue (Minolta laser scanner)



Reconstructed surface

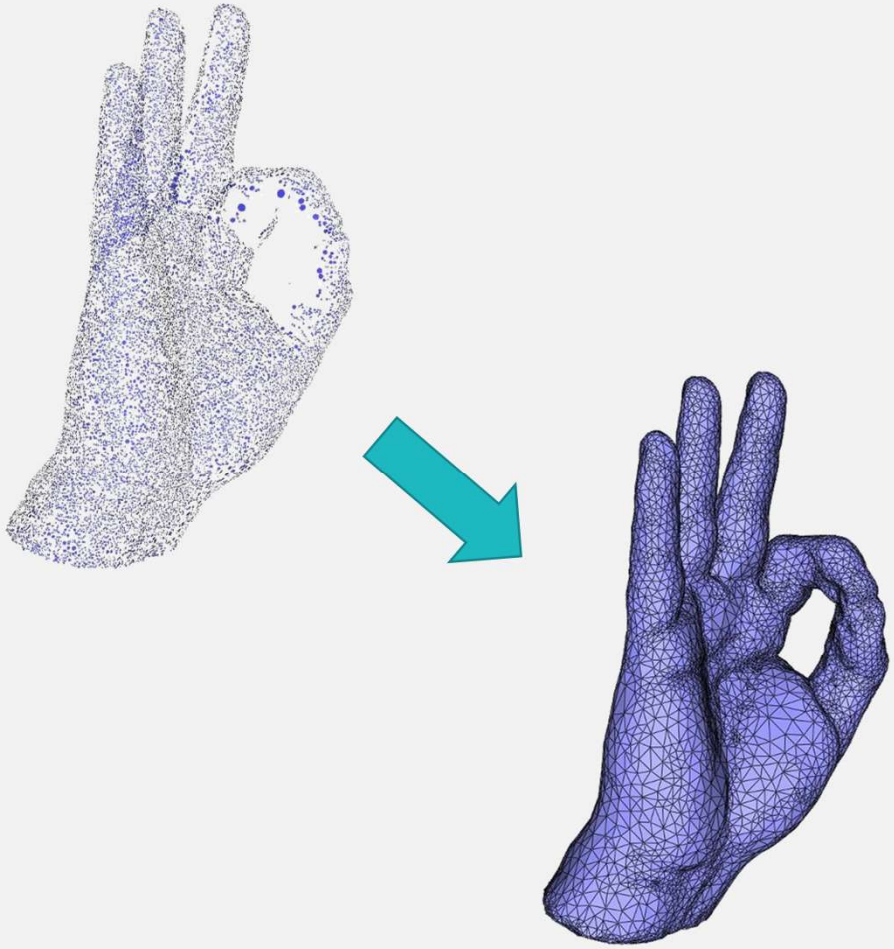
Poisson Surface Reconstruction Example

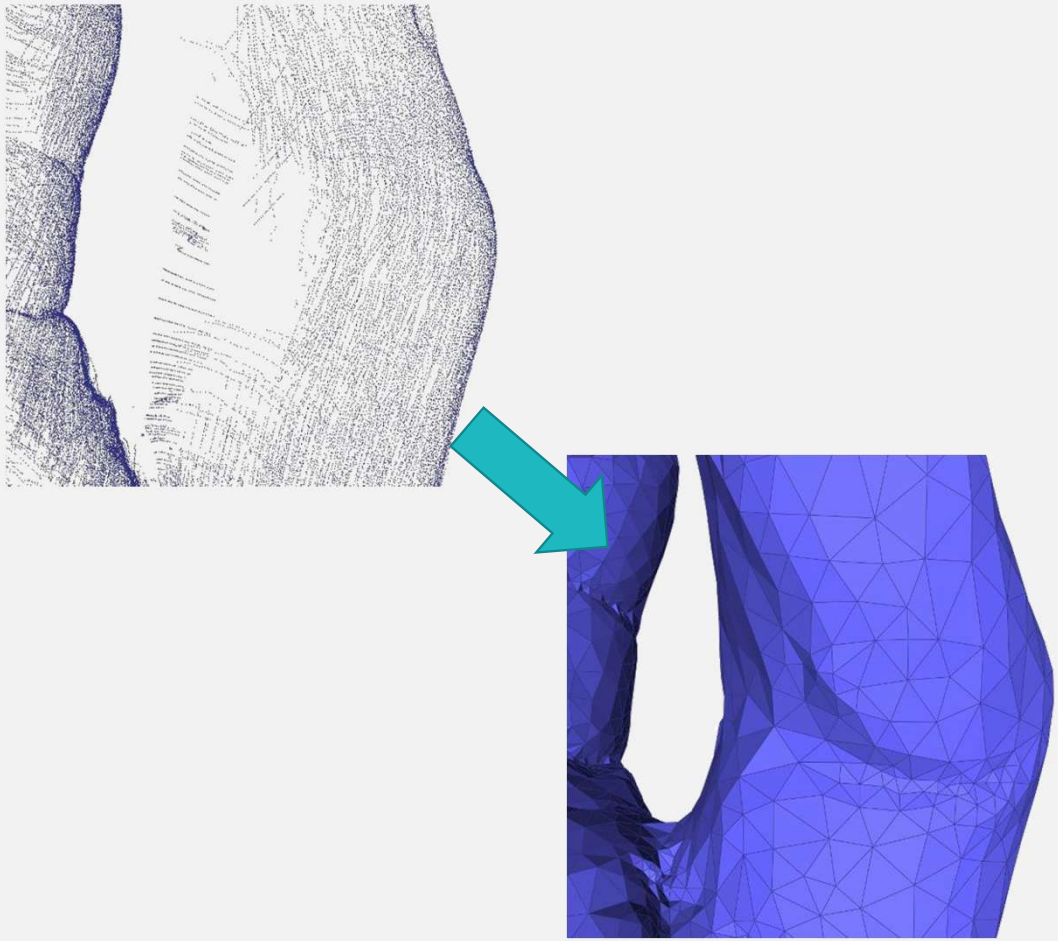
120K points sampled on a statue (Minolta laser scanner)



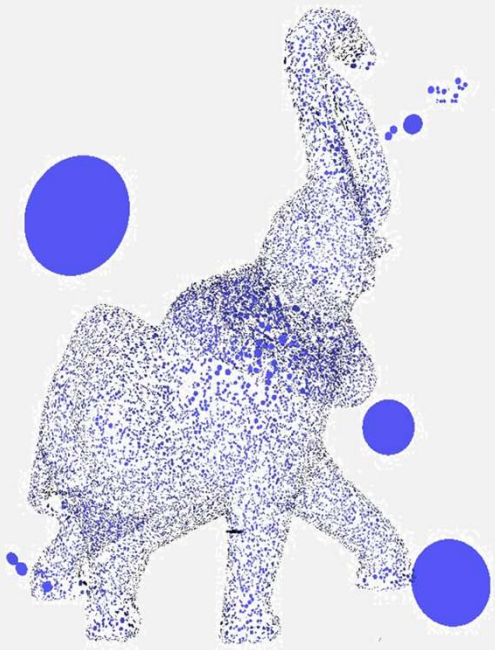
Reconstructed surface

Poisson Surface Reconstruction Example

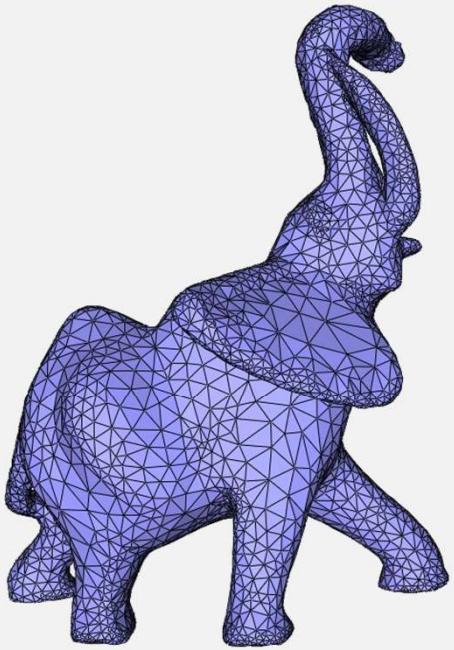




Poisson Surface Reconstruction Example

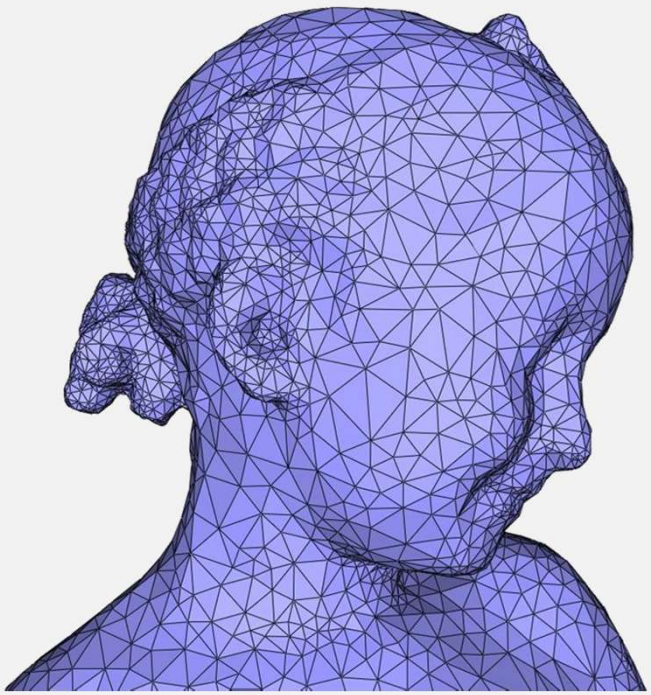


Points with Outliers

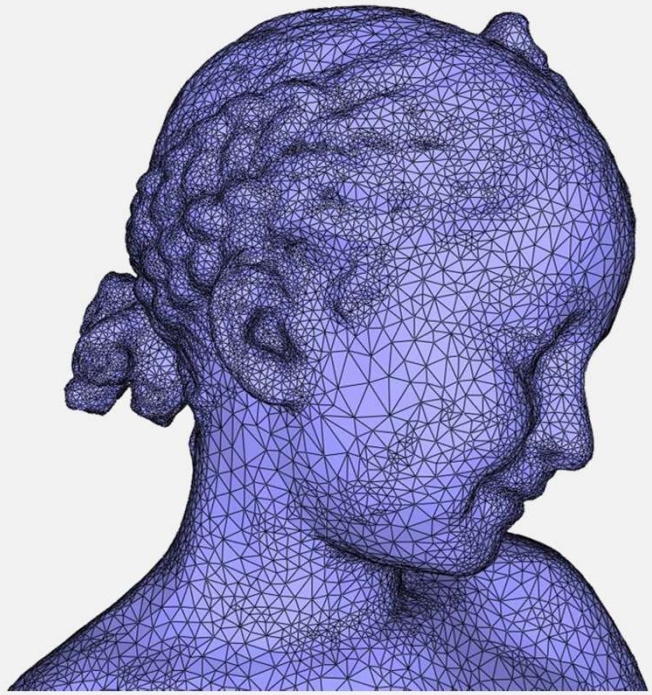


Reconstructed surface

Poisson Surface Reconstruction Example

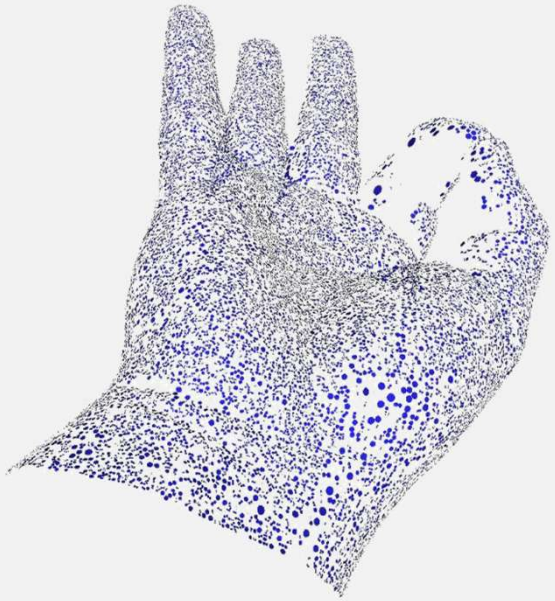


Bimba 120K reconstructed with distance = 0.25*average spacing

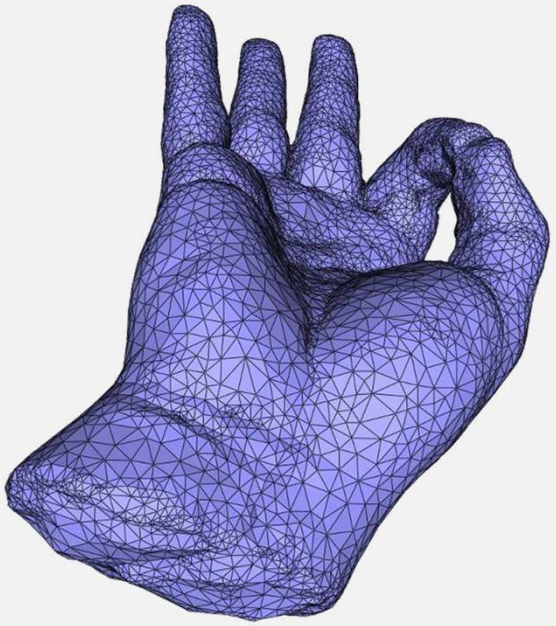


Bimba 120K reconstructed with distance = 0.15*average spacing

Poisson Surface Reconstruction Example

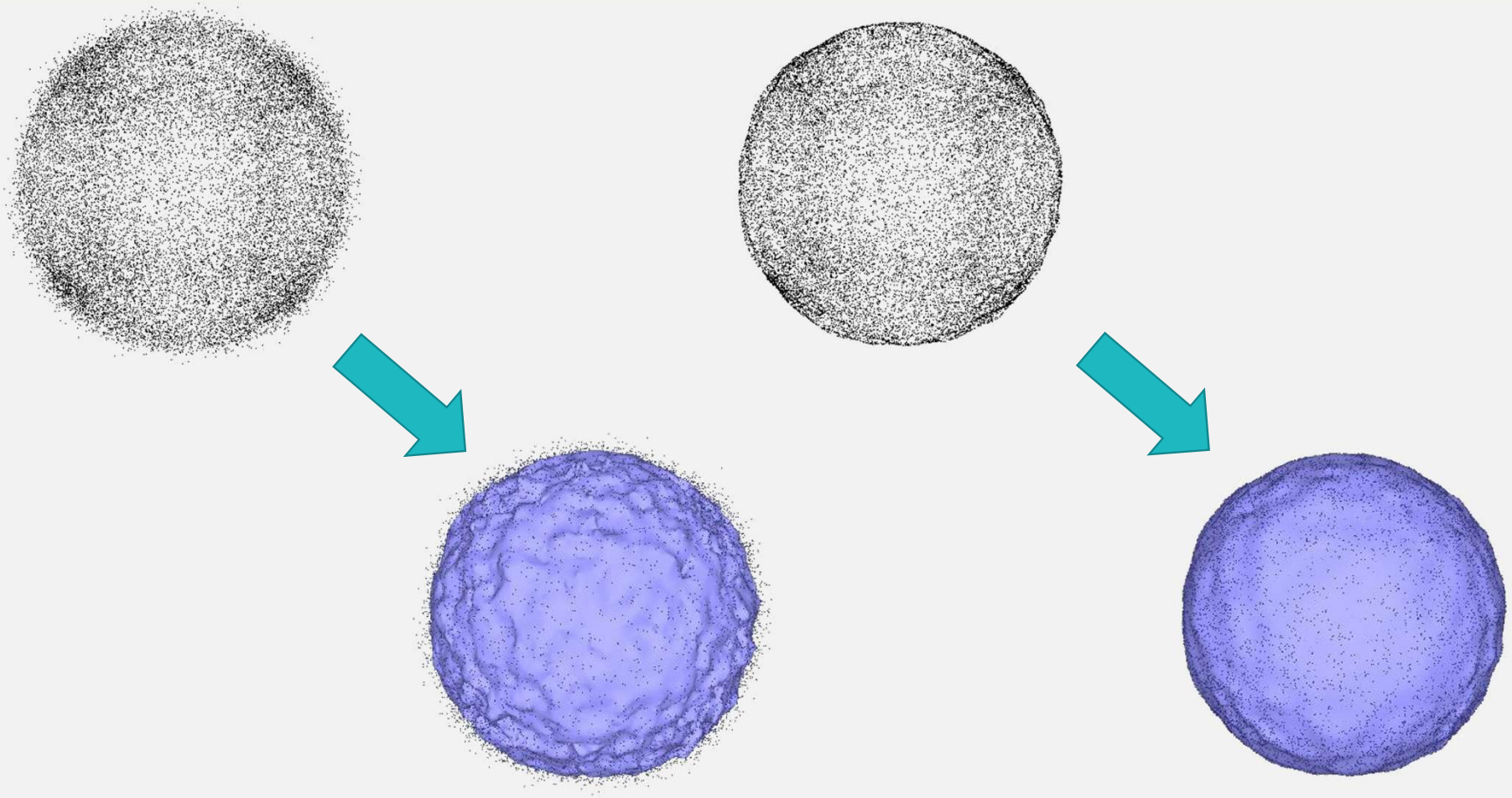


65K points sampled on a hand with no data at the wrist base (Kreon laser scanner)

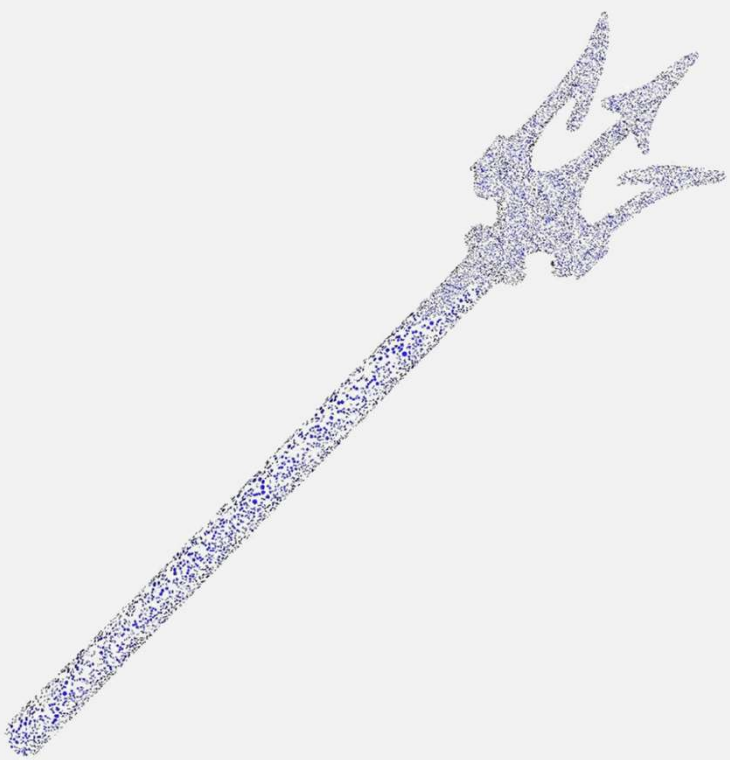


Reconstructed surface

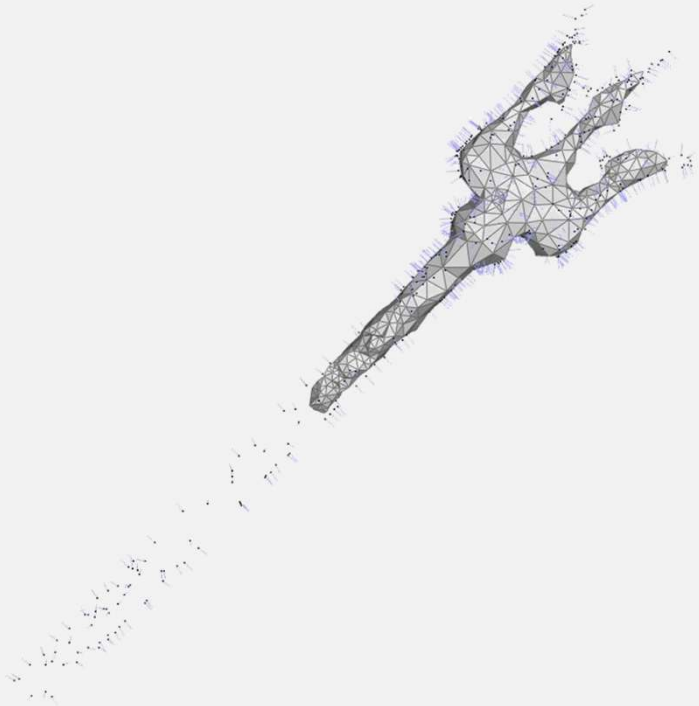
Poisson Surface Reconstruction Example



Poisson Surface Reconstruction Example

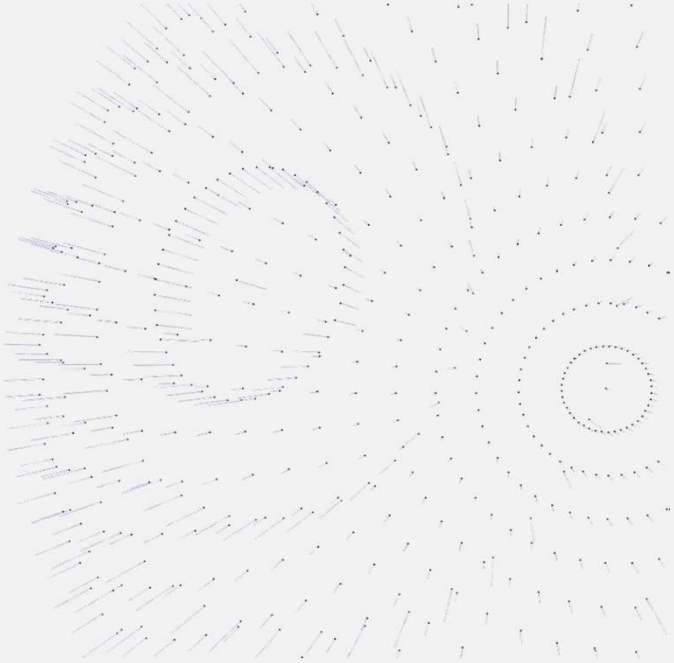


50K points sampled on Neptune trident

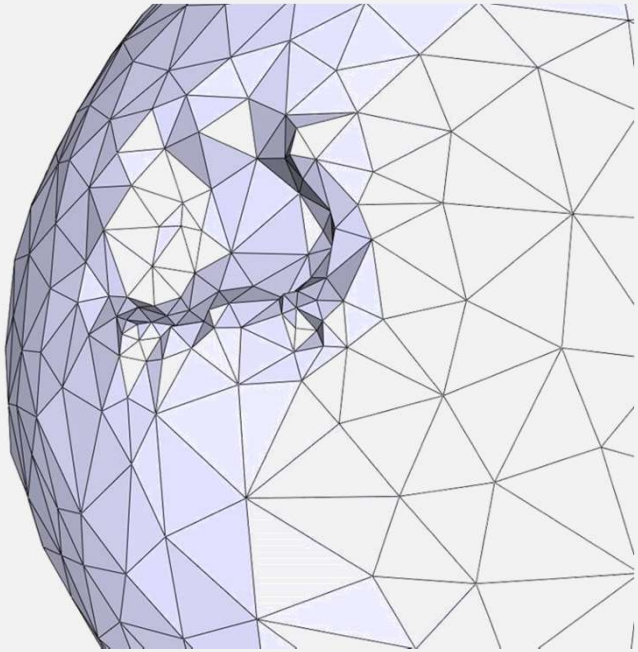


Point set simplified to 1K then reconstructed

Poisson Surface Reconstruction Example



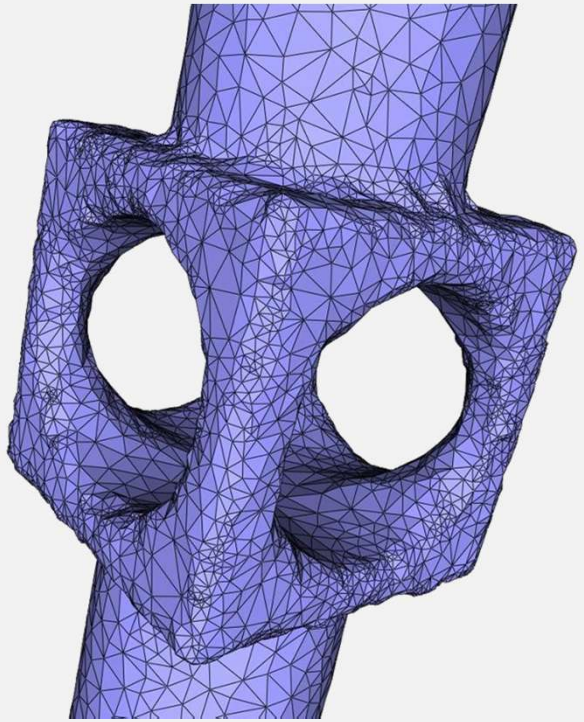
Points sampled on a sphere with flipped normals



Reconstructed surface

Poisson Surface Reconstruction Example

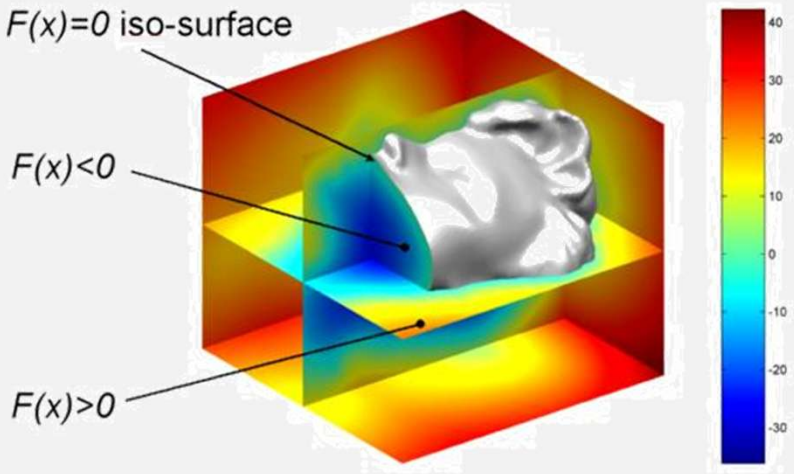
4K points sampled on a mechanical piece with sharp edges



Reconstructed surface

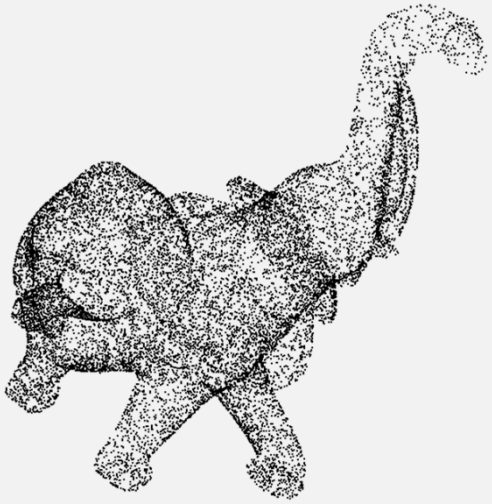
APSS Surface Reconstruction

- Algebraic point set surfaces (APSS) [Guennebaud, SIGGRAPH 2007]
  - Evaluates an implicit function on the fly (~signed distance function)
  - Isosurface extracted by CGAL surface mesh generator
  - Based on **Moving Least Squares fitting** on algebraic spheres

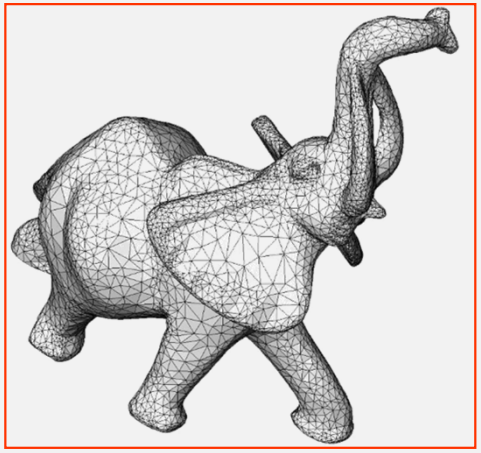


APSS Surface Reconstruction (Cont.)

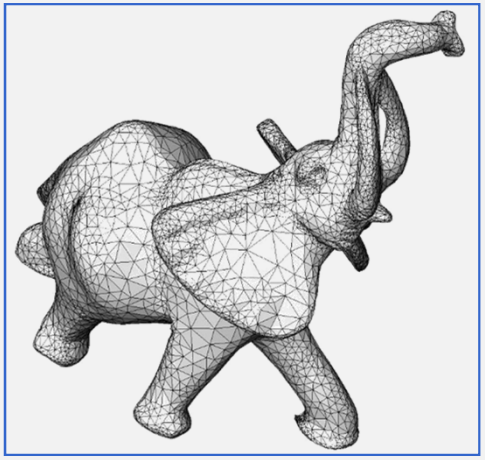
Reconstruction Example



17K points sampled on an elephant (Minolta laser scanner)



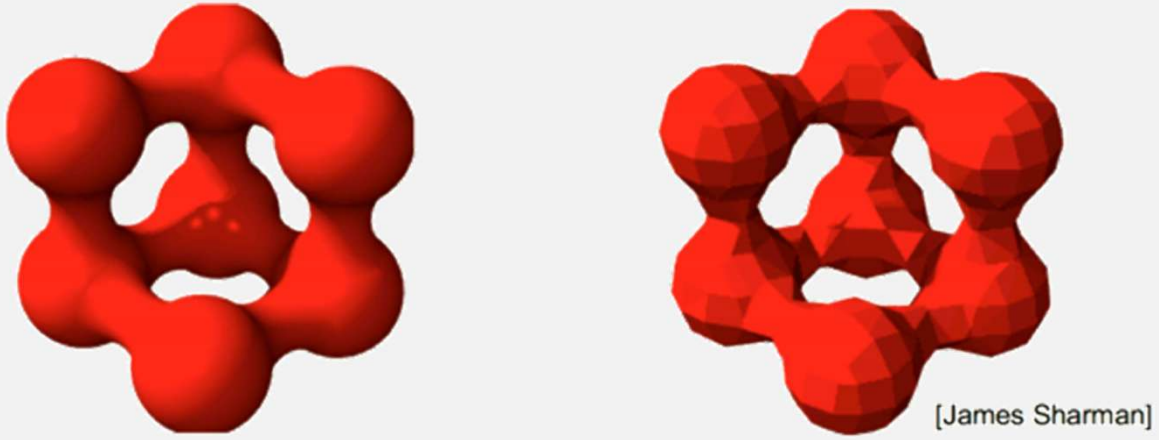
reconstructed surface using Poisson



reconstructed surface using APSS (in progress)

Marching Cubes

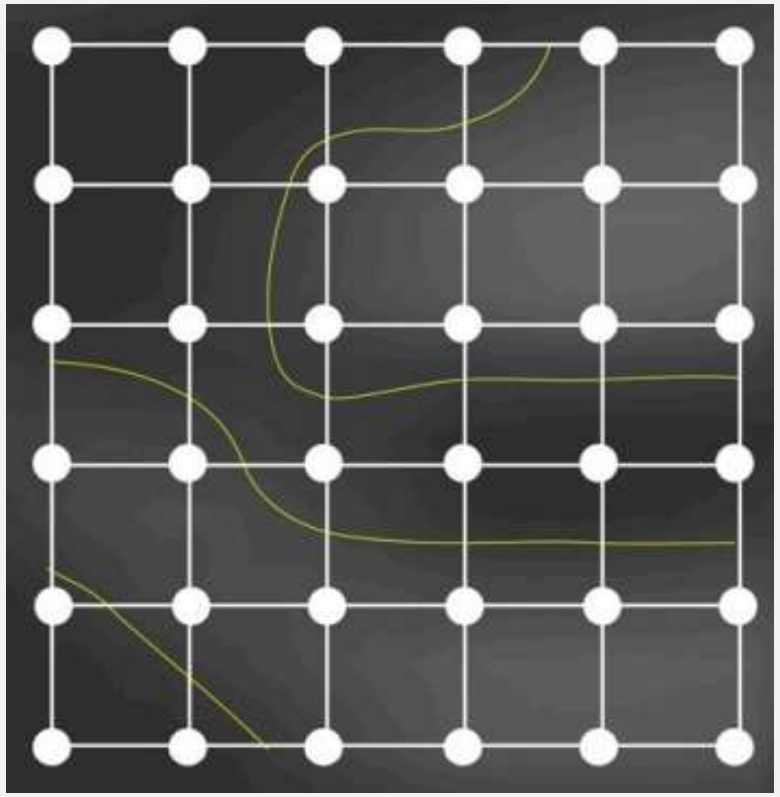
- Goal: Given an implicit representation $\{\mathbf{x}, \text{s. t. } f(\mathbf{x}) = 0\}$,
  create a triangle mesh that approximates the surface.



Marching Squares (2D)

- Given a function $f(\mathbf{x})$
  - $f(\mathbf{x}) < 0$: inside
  - $f(\mathbf{x}) > 0$: outside

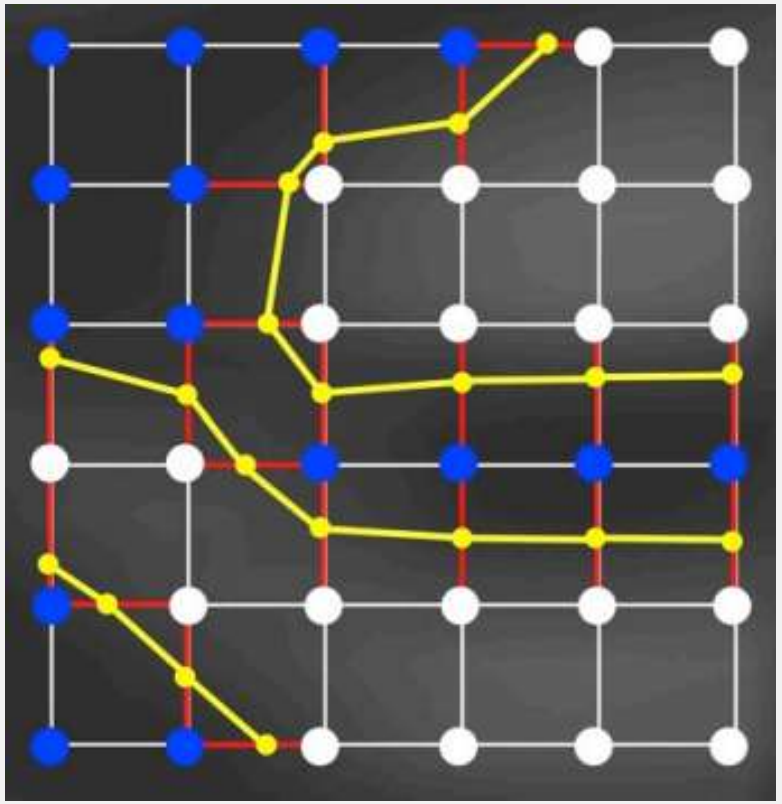
1. Discreyize space
2. Evaluate $f(\mathbf{x})$ on a grid



Marching Squares (2D)

- Given a function $f(\mathbf{x})$
  - $f(\mathbf{x}) < 0$: inside
  - $f(\mathbf{x}) > 0$: outside

1. Discreyize space
2. Evaluate $f(\mathbf{x})$ on a grid
3. Classify grid points (+-)
4. Classify grid edges
5. Compute intersections
6. Connect intersections

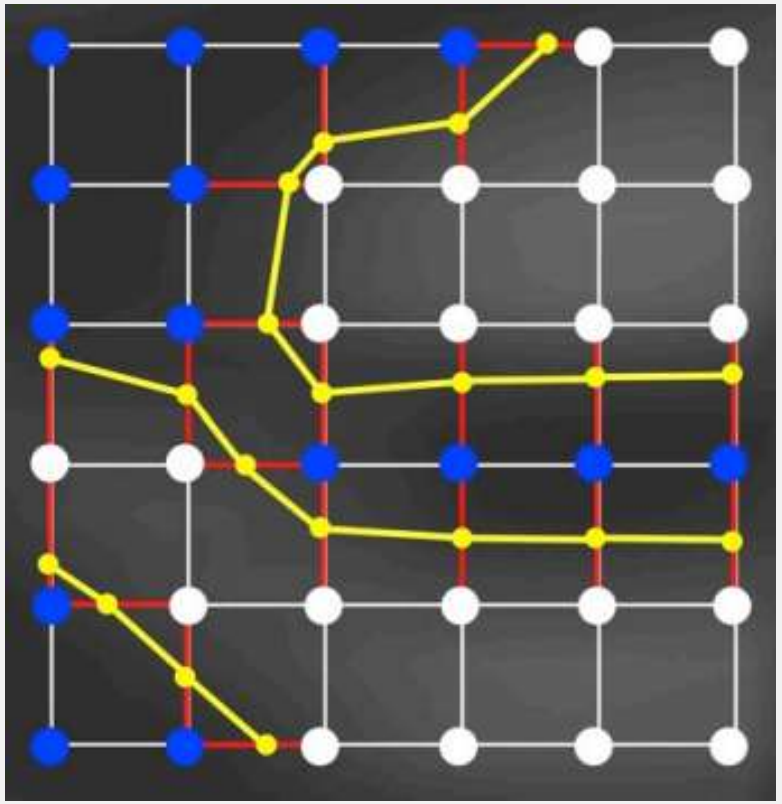


Marching Squares (2D)

- Compute intersections :
  - Edges with a sign switch contain intersections
  - $f(\mathbf{x}_1) < 0\,, f(\mathbf{x}_2) > 0$

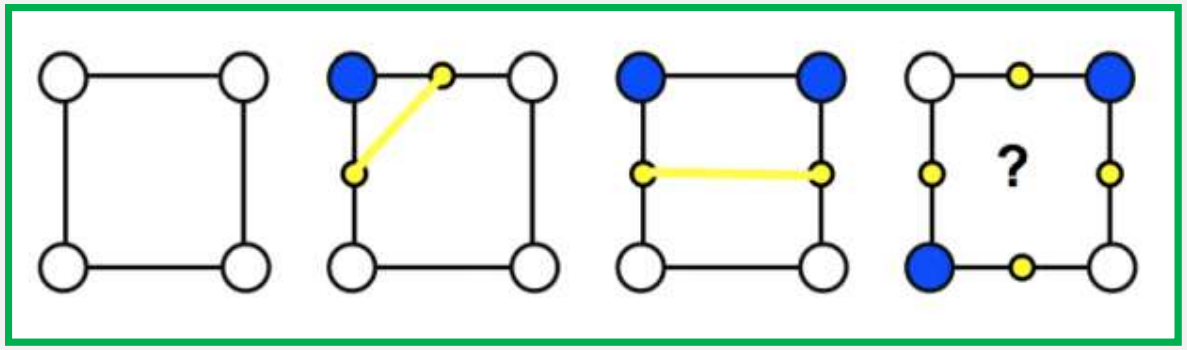
  ➔$f(\mathbf{x}_1 + \boldsymbol{t}(\mathbf{x}_2 - \mathbf{x}_1)) = 0$ for some $0 \leq \boldsymbol{t} \leq \mathbf{1}$

- Connecting intersections
  - Grand principle: treat each cell separately!
  - Enumerate all possible inside/outside combinations
  - Group those leading to the same intersections
  - Group equivalent after rotation
  - Connect intersections



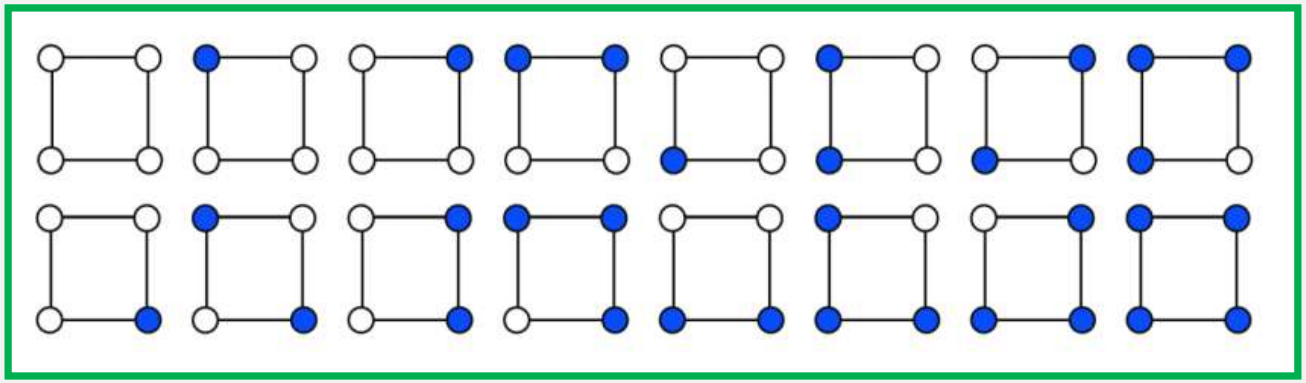
Marching Squares (2D)

- Connecting intersections
  - Grand principle: treat each cell separately!
  - Enumerate all possible inside/outside combinations
  - Group those leading to the same intersections
  - Group equivalent after rotation
  - Connect intersections



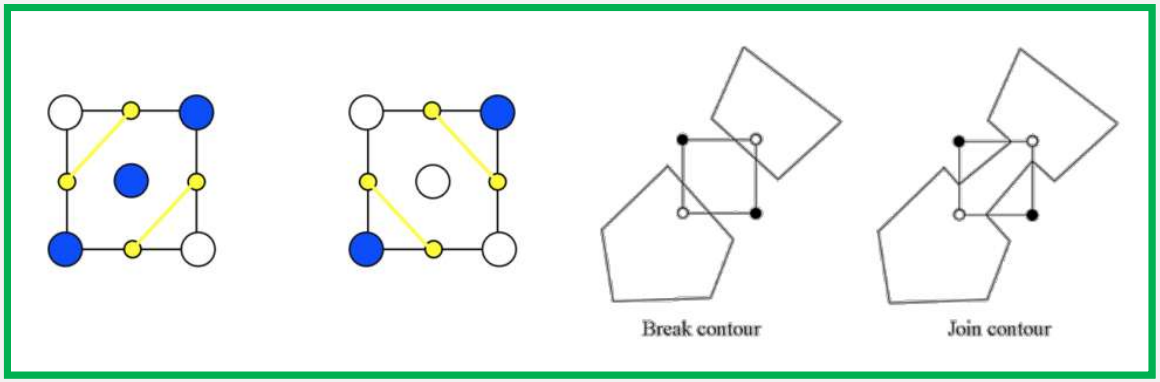
Marching Squares (2D)

- Connecting intersections
  - Grand principle: treat each cell separately!
  - Enumerate all possible inside/outside combinations
  - Group those leading to the same intersections
  - Group equivalent after rotation
  - Connect intersections



Marching Squares (2D)

- Connecting intersections
  - Ambiguous Cases:



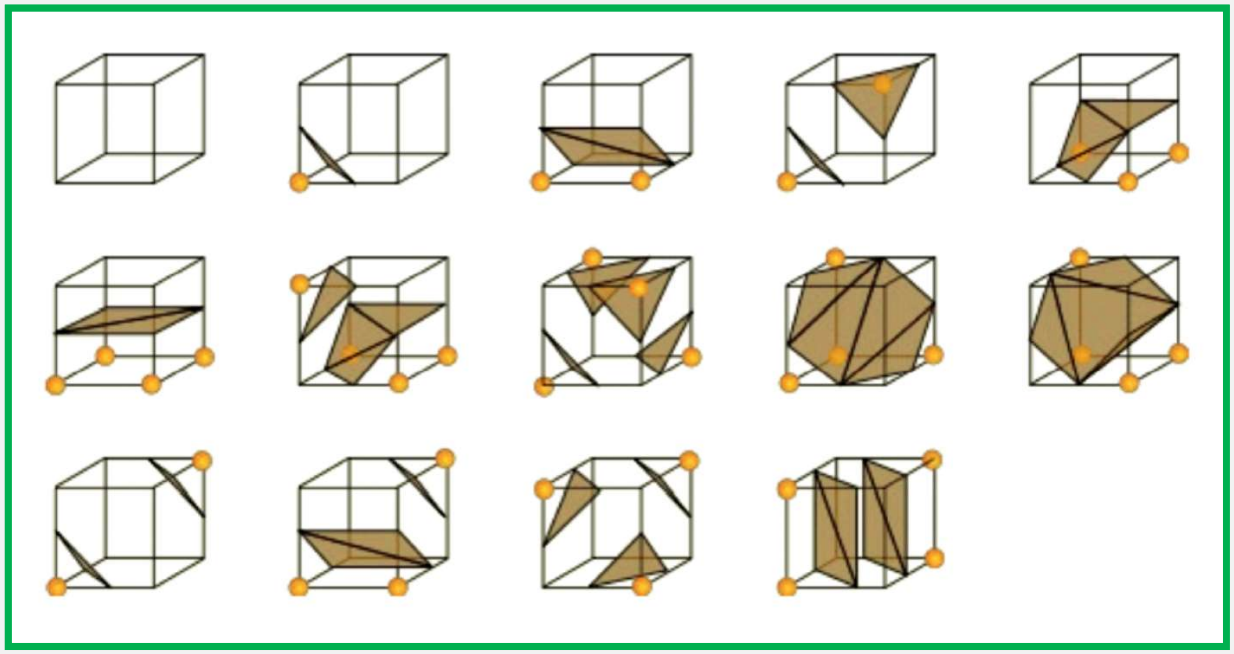
  - Two options:

    1) Resolve ambiguity by subsampling inside the cell.

    2) If subsampling is impossible, pick one of the two possibilities.

Marching Cubes (3D)

- Same basic machinery applies to 3D
  - cells become cubes (voxels), lines become triangles
- 256 different cases, 14 after symmetries



Marching Cubes (3D)

- 6 ambiguous cases :