Hands-on AI based 3D Vision
Summer Semester 25

Lecture 5_0 – Surface Reconstruction

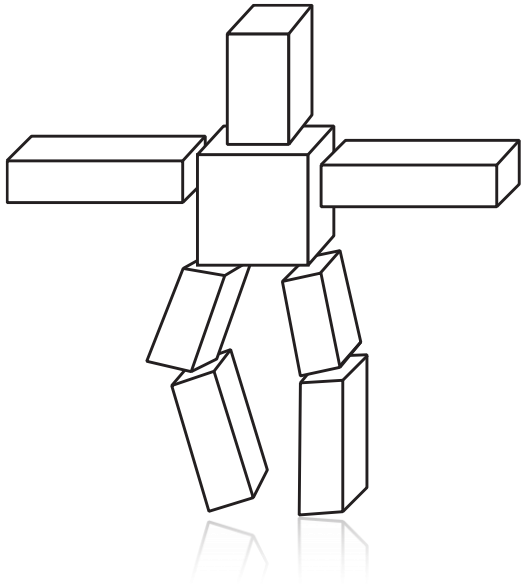
Prof. Dr.-Ing Gerard Pons-Moll

University of Tübingen / MPI-Informatics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Increasing complexity of our models

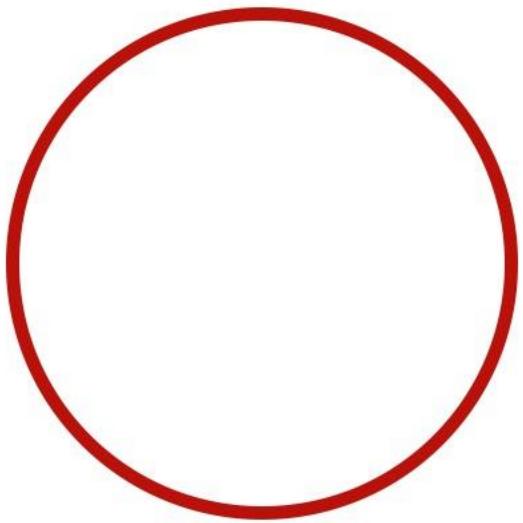
Transformations



Geometry

Materials, lighting, ...

How can we describe geometry?



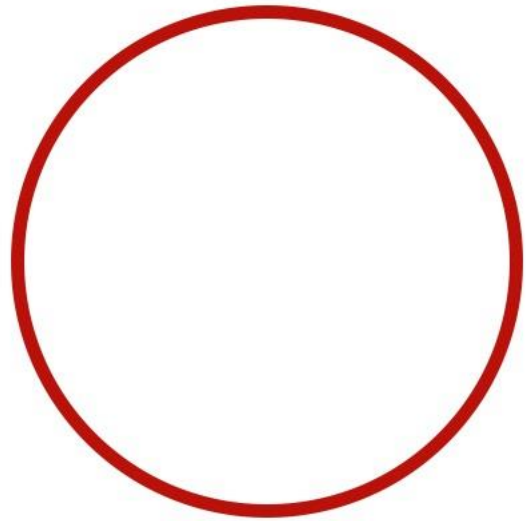
How can we describe geometry?

IMPLICIT

$$x^2+y^2=1$$

LINGUISTIC

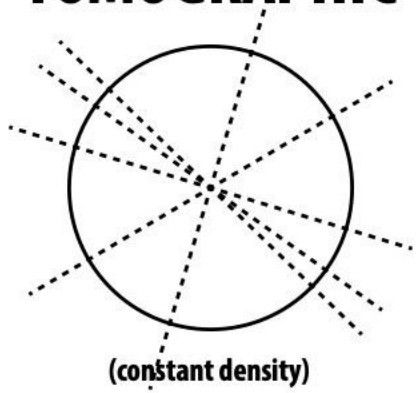
"unit circle"



EXPLICIT

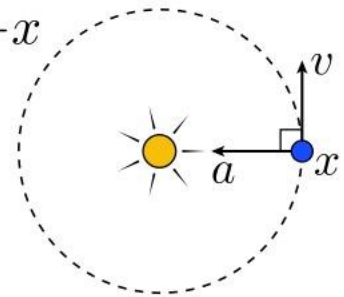
$$(\underbrace{\cos\theta}_{x}, \underbrace{\sin\theta}_{y})$$

TOMOGRAPHIC



DYNAMIC

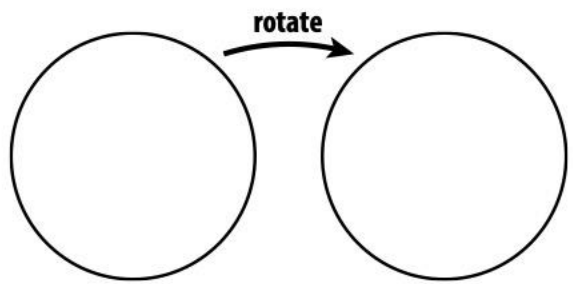
$$\frac{d^2}{dt^2}x = -x$$



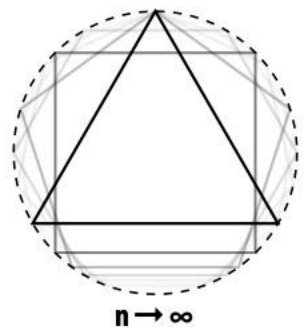
CURVATURE

$$\kappa = 1$$

SYMMETRIC



DISCRETE



Given all these options, what is the best way to encode geometry on a computer?

It's a jungle out there!

No one "best" choice—geometry is hard!

"I hate meshes.
I cannot believe how hard this is.
Geometry is hard."

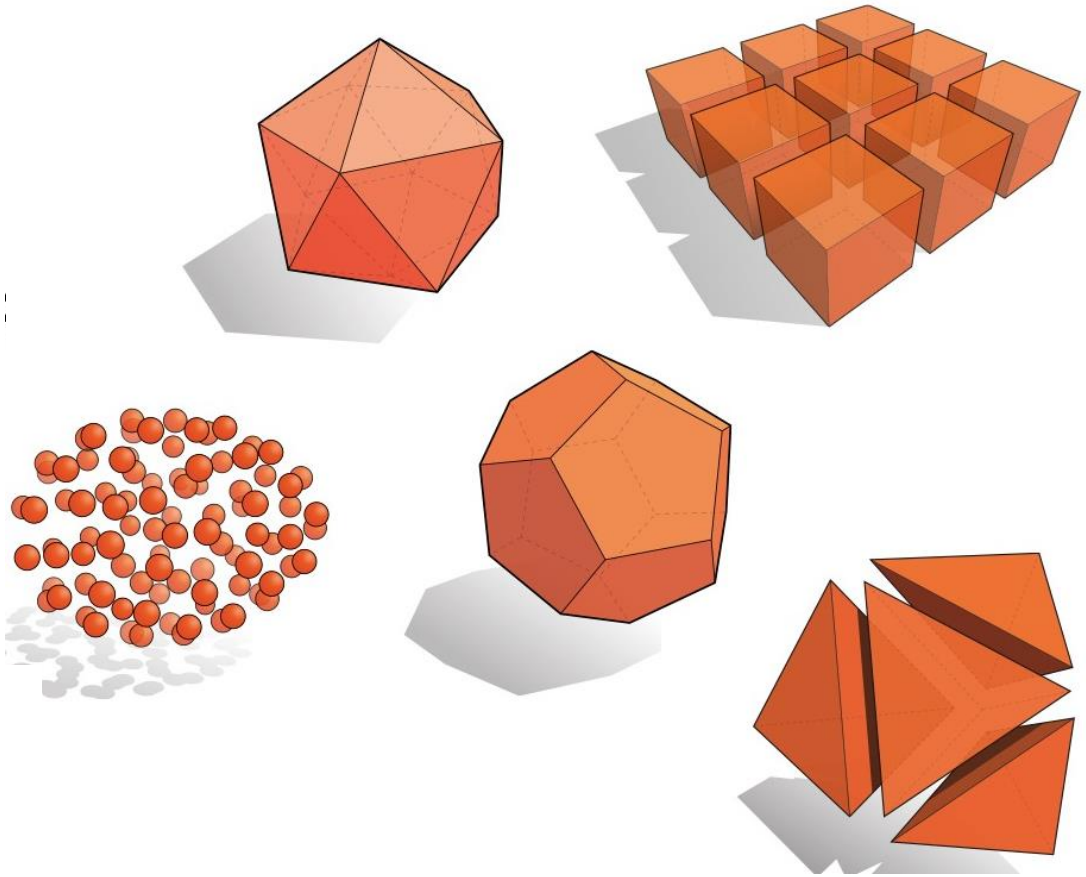
—David Baraff

Senior Research Scientist
Pixar Animation Studios

Slide cribbed from Jeff Erickson.

Many ways to digitally encode geometry

- EXPLICIT
  - point cloud
  - polygon mesh
  - subdivision, NURBS
  - …
- IMPLICIT
  - level set
  - algebraic surface
  - L-systems
  - …
- Each choice best suited to a different task/type of geometry



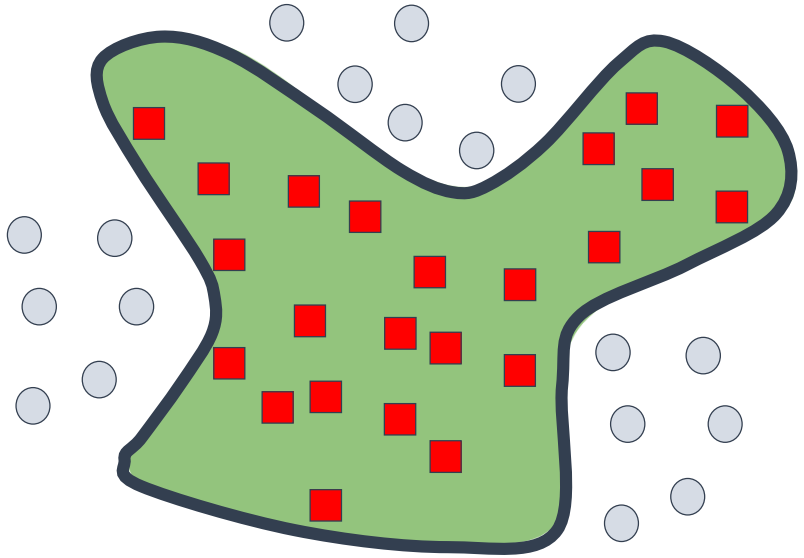
"Implicit" Representations of Geometry

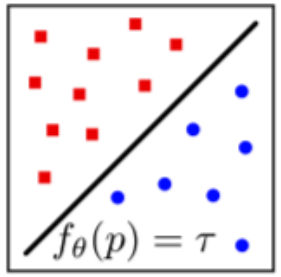
- Points aren't known directly, but satisfy some relationship
- E.g., unit sphere is all points such that $x^2+y^2+z^2=1$
- More generally, f(x,y,z) = 0

Surfaces as an Implicit Function

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3$$

$$f(\mathbf{p}) = \begin{cases} 0, & \text{if} \quad \mathbf{p} \in \text{outside} \\ 1, & \text{if} \quad \mathbf{p} \in \text{inside} \end{cases}$$

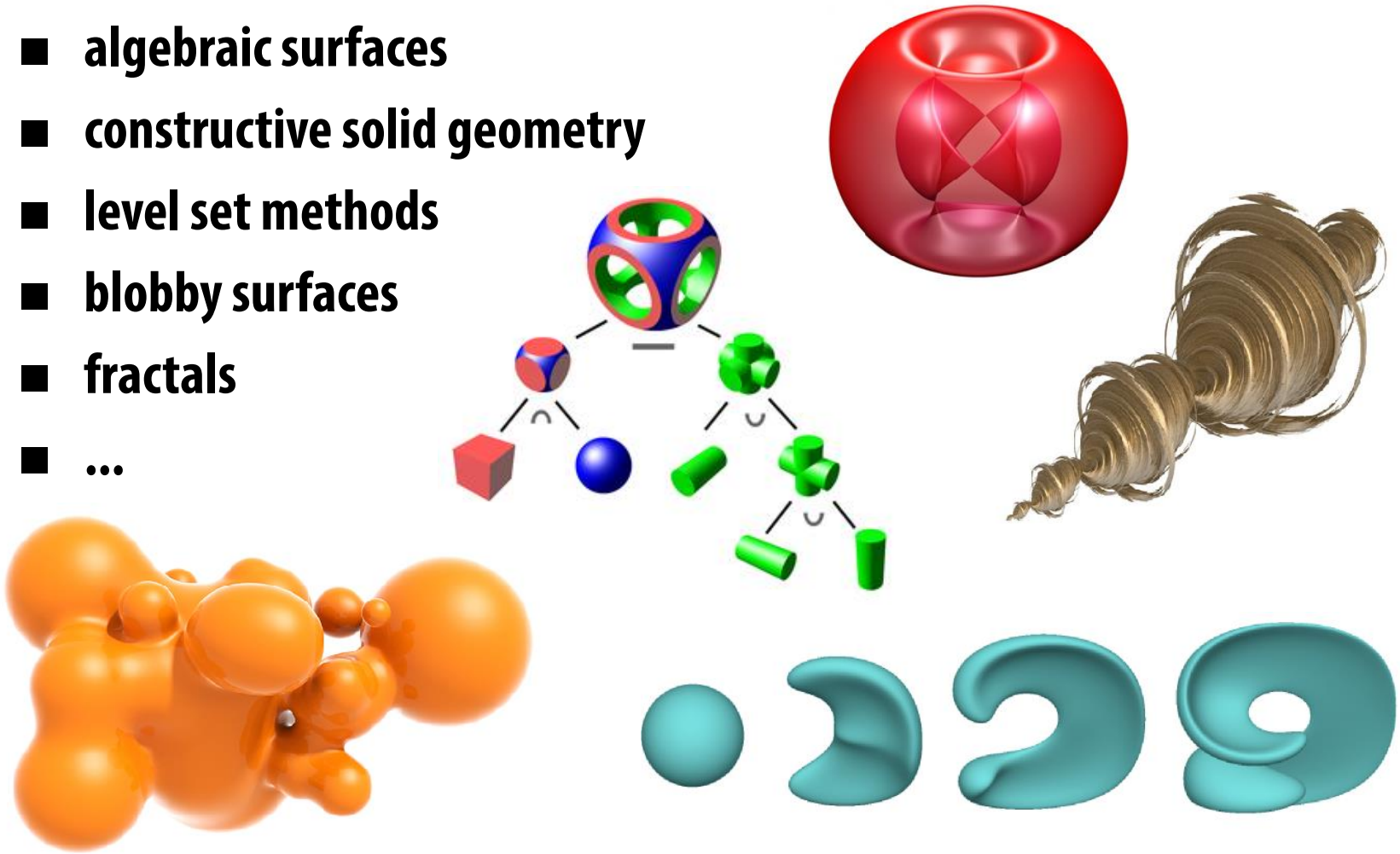




$$\mathcal{S} = \{\mathbf{p},\ f(\mathbf{p}) = \tau\}$$

Many implicit representations in graphics

- **algebraic surfaces**
- **constructive solid geometry**
- **level set methods**
- **blobby surfaces**
- **fractals**
- **…**



(Will see some of these a bit later.)

But first, let's play a game:

I'm thinking of an implicit surface f(x,y,z)=0.

Find *any* point on it.

Let's play another game.

I have a new surface $f(x,y,z) = x^2 + y^2 + z^2 - 1$.

I want to see if a point is *inside* it.

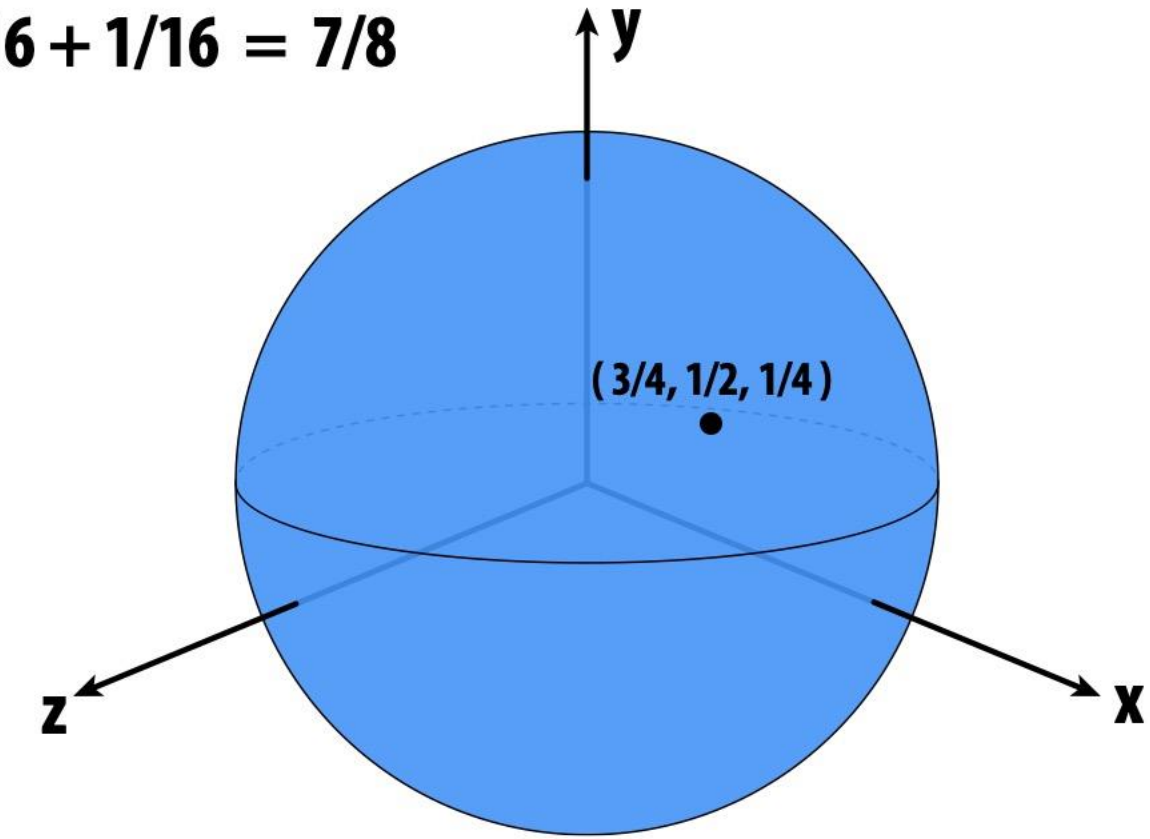
Check if this point is inside the unit sphere

How about the point (3/4, 1/2, 1/4)?

$9/16 + 4/16 + 1/16 = 7/8$

$7/8 < 1$

YES.



Implicit surfaces make other tasks easy (like inside/outside tests).

Algebraic Surfaces (Implicit)

- Surface is zero set of a polynomial in x, y, z
- Examples:

$x^2+y^2+z^2=1$

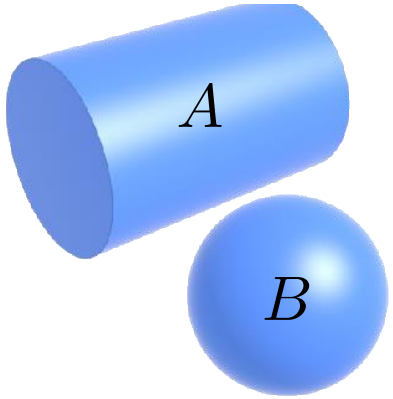
$(R-\sqrt{x^2+y^2})^2+z^2=r^2$

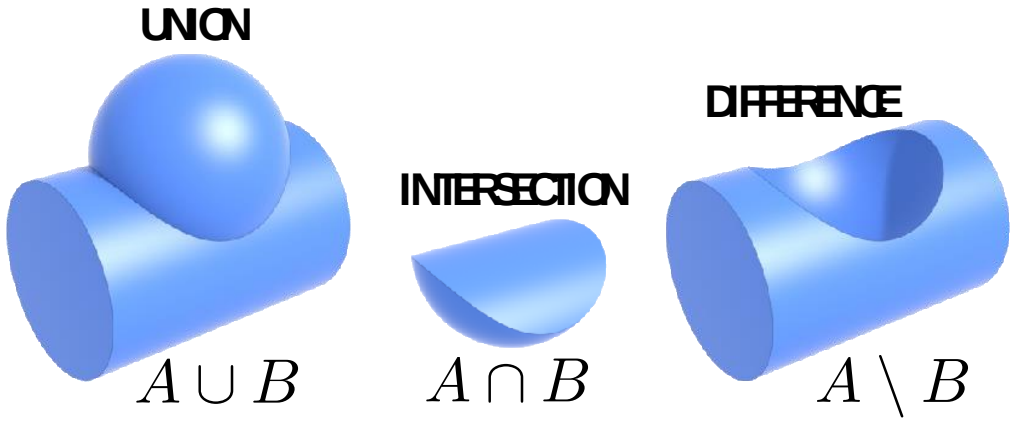
$(x^2+\frac{9y^2}{4}+z^2-1)^3=x^2z^3+\frac{9y^2z^3}{80}$

- What about more complicated shapes?
- Very hard to come up with polynomials!

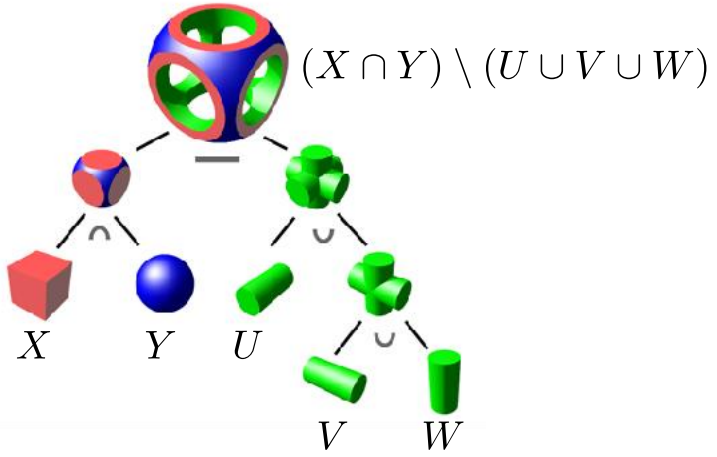
Constructive Solid Geometry (Implicit)

- Build more complicated shapes via Boolean operations
- Basic operations:



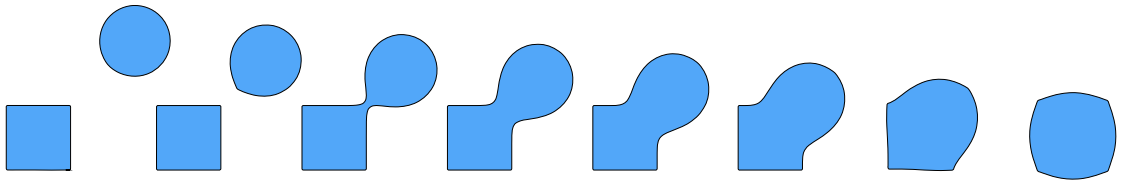


- Then chain together expressions:



Blending Distance Functions (Implicit)

- A distance function gives distance to closest point on object
- Can blend any two distance functions d_1, d_2:

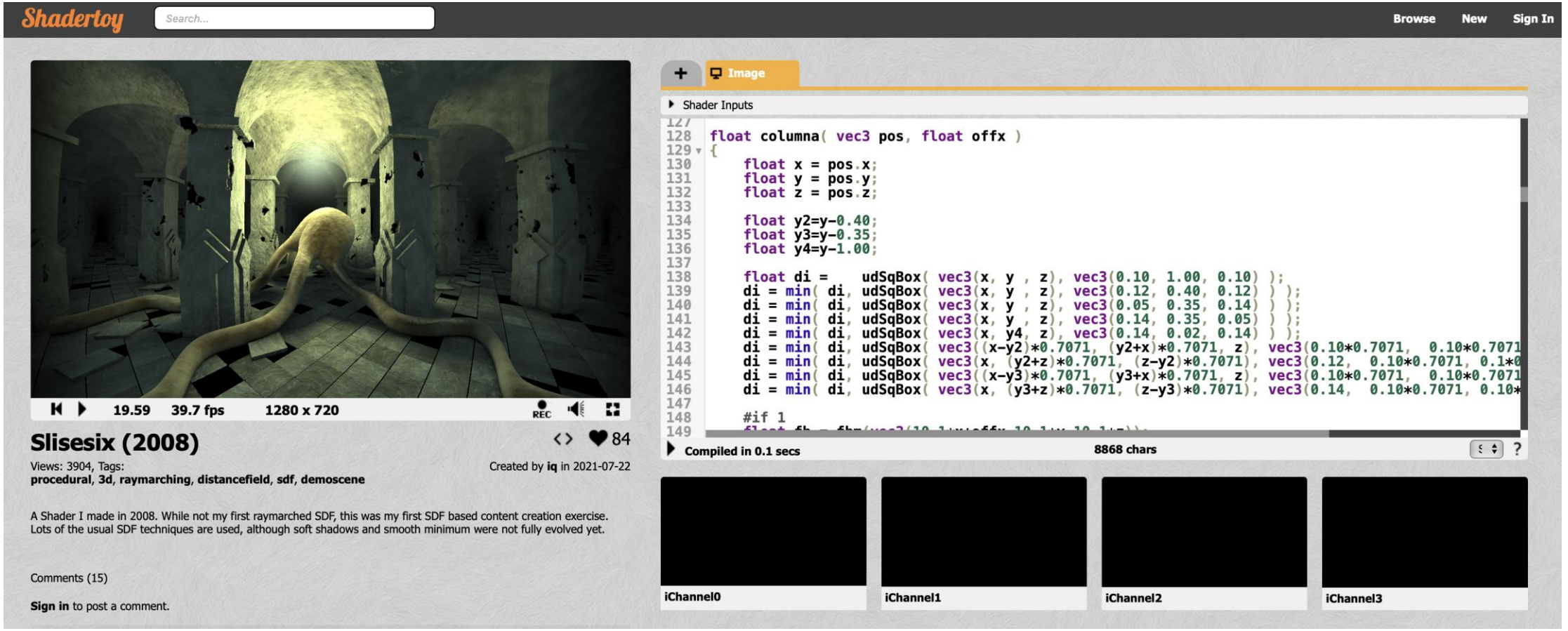


- Similar strategy to points, though many possibilities. E.g.,

$$f(x) := e^{d_1(x)^2} + e^{d_2(x)^2} - \frac{1}{2}$$

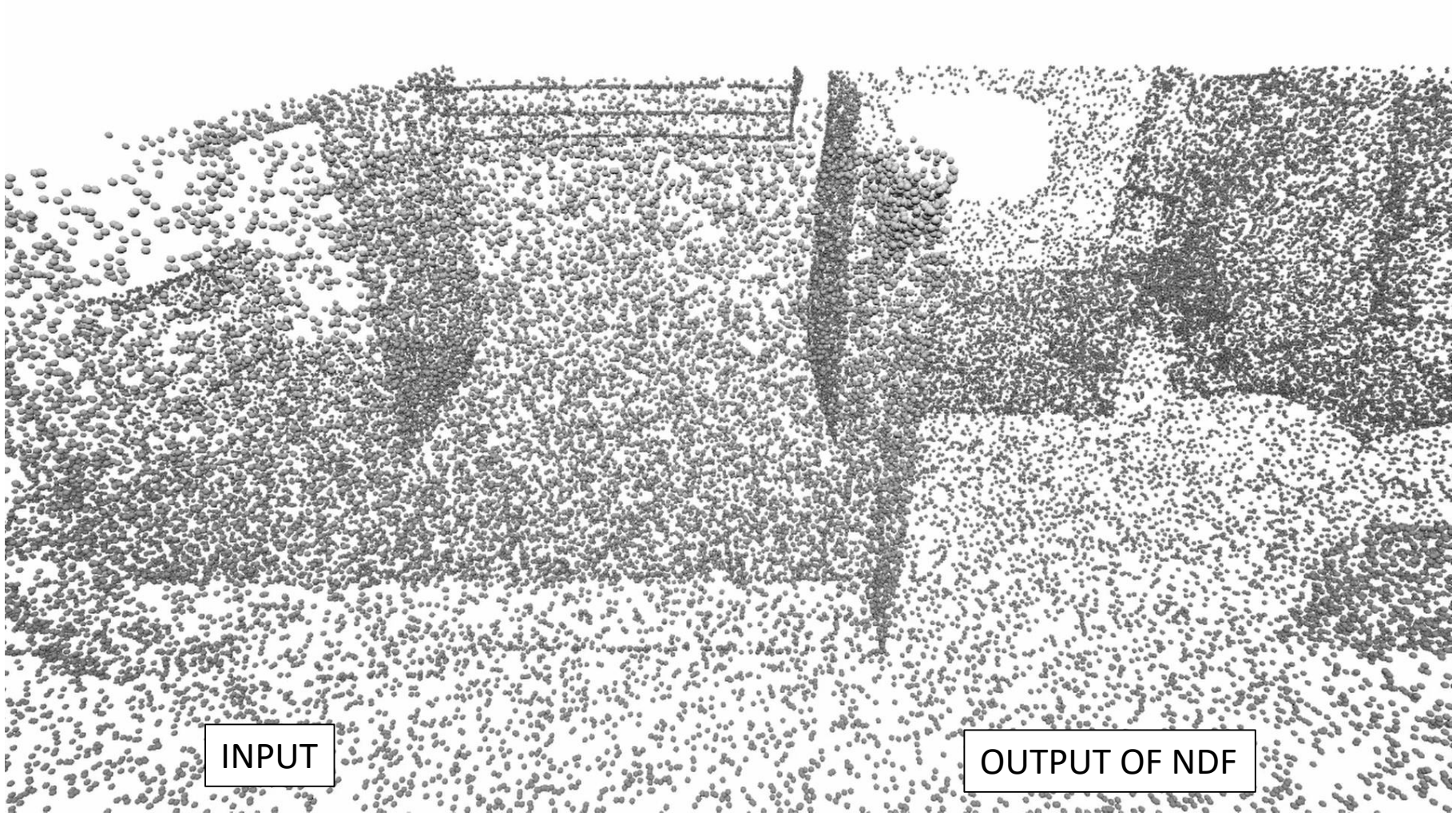
- Appearance depends on how we combine functions
- Q: How do we implement a Boolean union of $d_1(x)$, $d_2(x)$?
- A: Just take the minimum: $f(x) = \min(d_1(x), d_2(x))$

Scene made of pure signed distance functions
Art with math -- really hard!



https://www.shadertoy.com/view/NtlSDs

Nowadays -- Neural Distance Fields (NDF)



Chibane et al. **NDF**, NeurIPS 2020

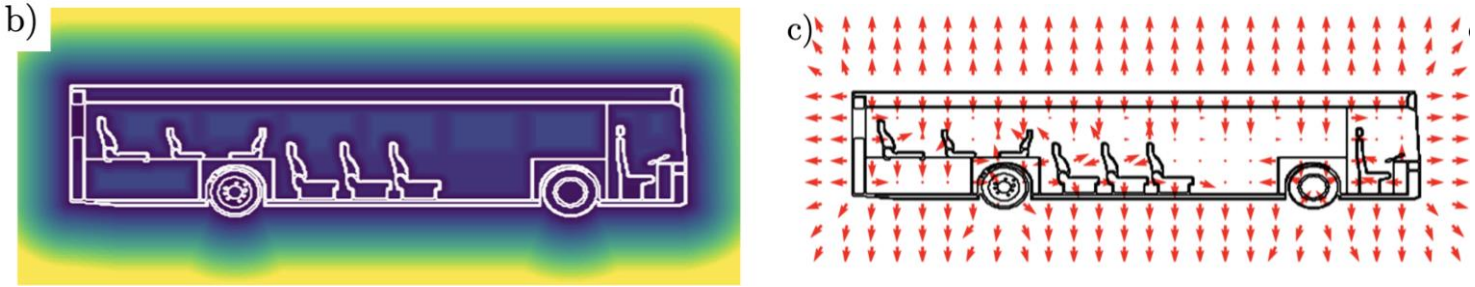
Distance field, field normals and closest points

- **Distance to surface** is given by the distance field itself f(p)

$$f(\mathbf{p}) = \min_{\mathbf{q} \in \mathcal{S}} \|\mathbf{p} - \mathbf{q}\| \qquad \mathcal{S} = \{\mathbf{p} \in \mathbb{R}^3 \,|\, f(\mathbf{p}) = 0)\}$$

- **Surface normal** are the gradients of the function

$$n(\mathbf{p}) = \nabla f(\mathbf{p})$$

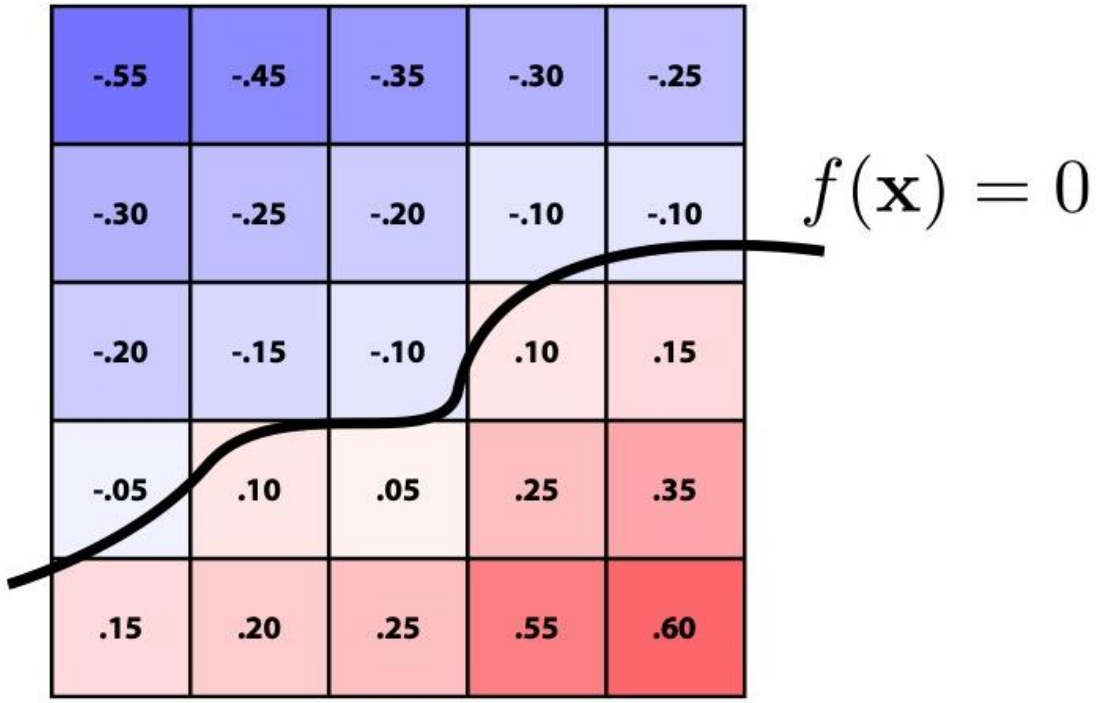


- **Closest points** are found trivially as:

$$\mathbf{q} = \mathbf{p} - f(\mathbf{p}) \nabla_{\mathbf{p}} f(\mathbf{p})$$

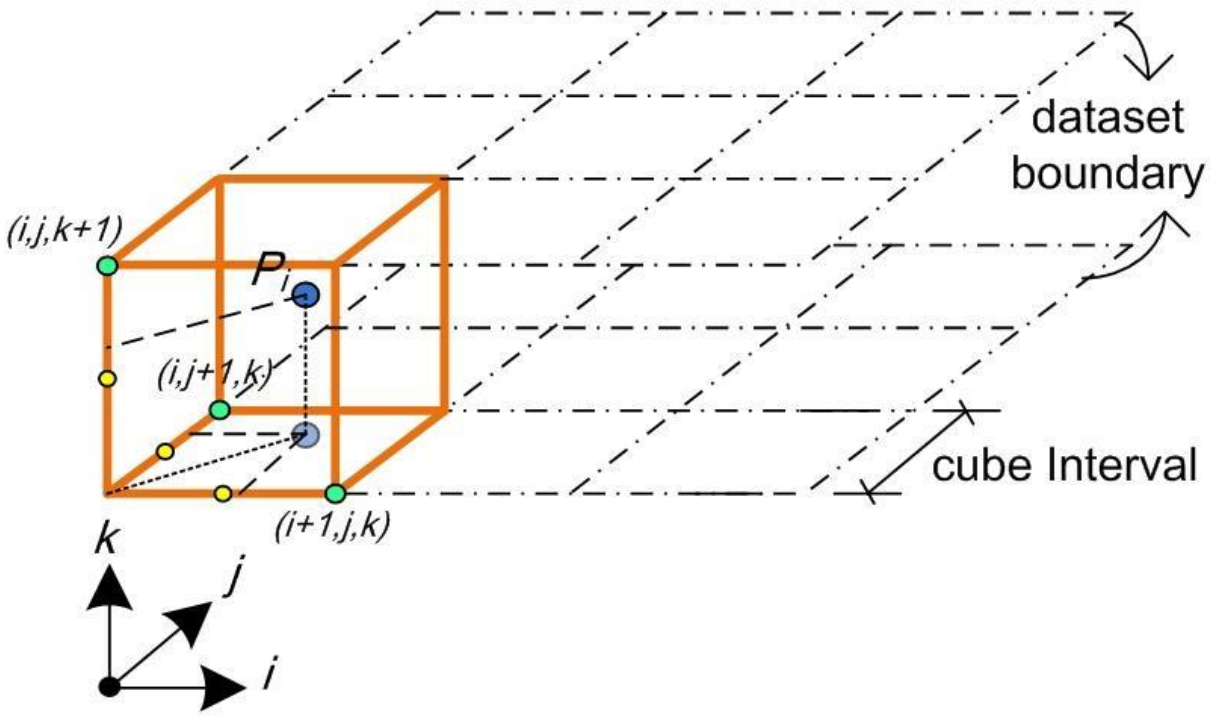
Surface Reconstruction from Implicit Representation

- Implicit surfaces have some nice features (e.g., merging/splitting)
- But, hard to describe complex shapes in closed form
- Alternative: store a grid of values approximating function
- Surface is found where interpolated values equal zero



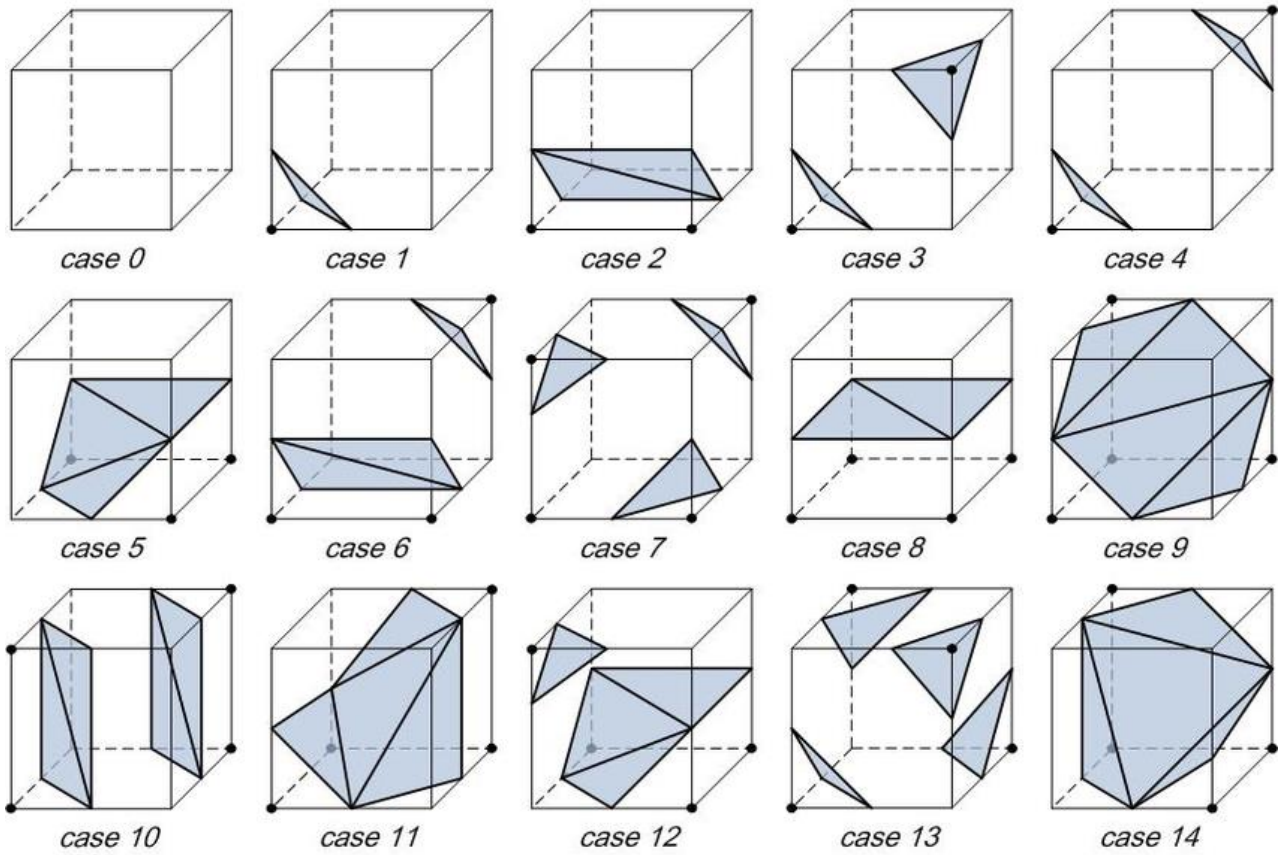
Marching Cubes

1. Read four slices into memory.
2. Scan two slices and create a cube from four neighbors on one slice and four neighbors on the next slice.
3. …
4. …
5. …
6. …
7. …



Marching Cubes

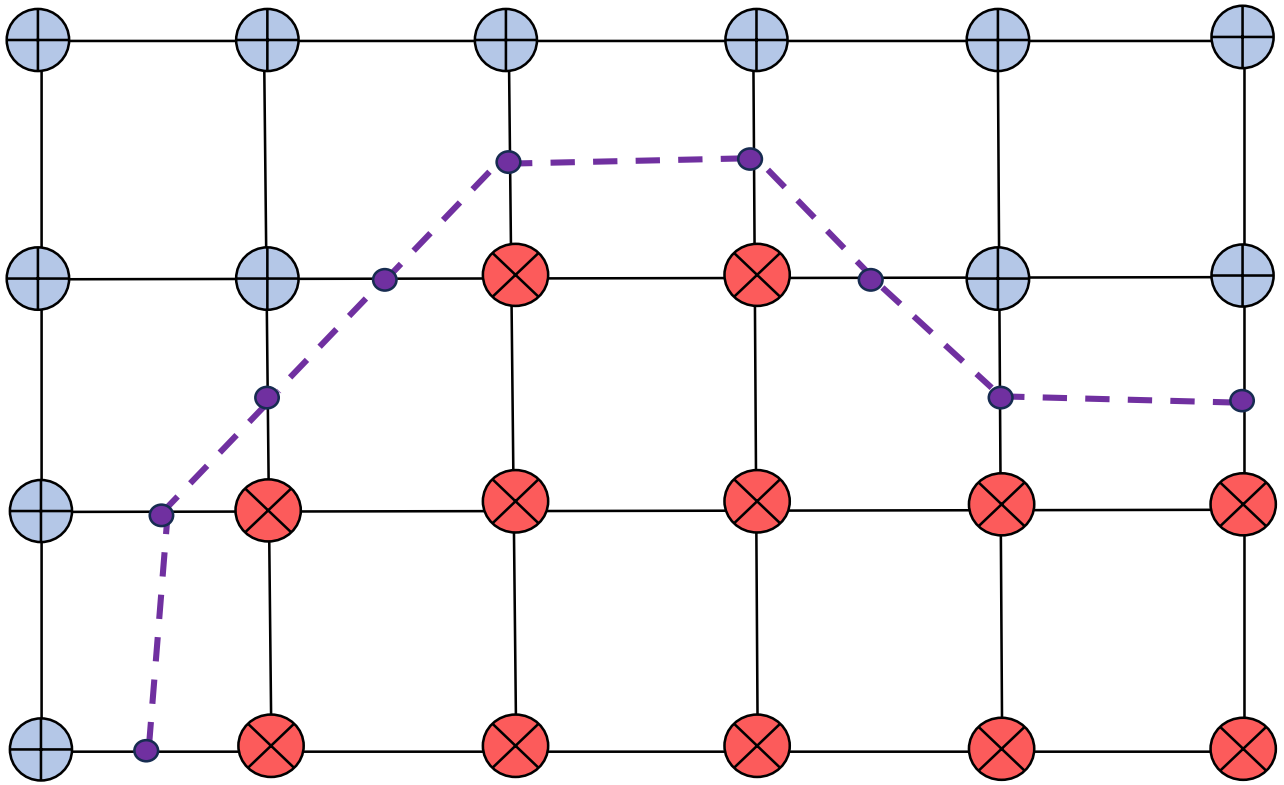
1. …
2. …
3. Calculate an index for the cube by comparing the eight density values at the cube vertices with the surface constant.
4. Using the index, look up the list of edges from a precalculated table.



Marching Cubes

1. …
2. …
3. …
4. …
5. Using the densities at each edge vertex, find the surface-edge intersection via linear interpolation
6. Calculate a unit normal at each cube vertex using central differences. Interpolate the normal to each triangle vertex
7. Output the triangle vertices and vertex normals.

Marching Cubes



Challenges with Marching Cubes

- Generating one polygon for each portion of the contour
- Can only approximate the restriction of the contour
- Fails to capture the sharp detail of the surface
- Difficult to simplify the generated surface mesh
- Becoming challenging to extract surface for higher grid sizes (3D uniform grid induces cubic cost in resolution)

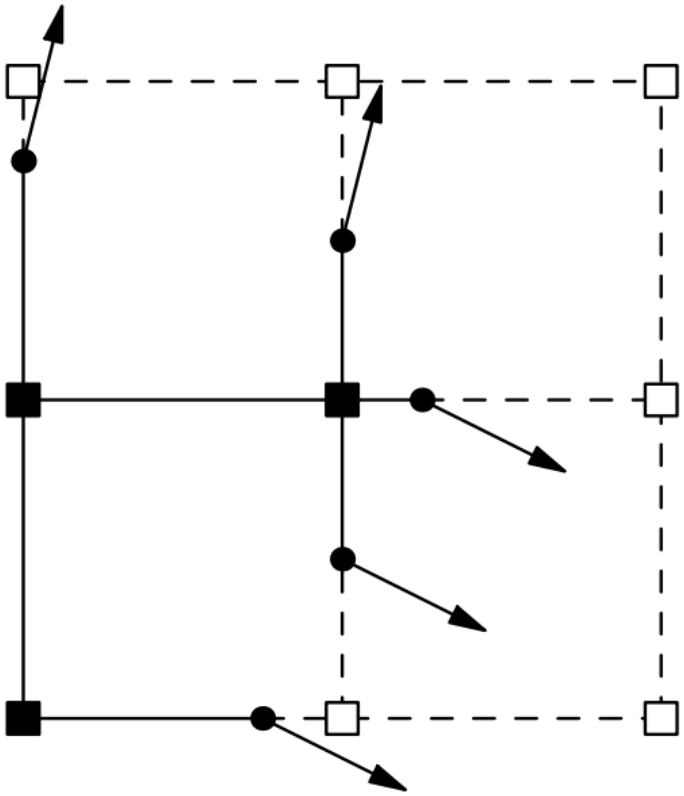
Dual Contouring

- Can capture sharp edges along corners of surface
- Can leverage gradient information along edge intersections to reproduce a wide class of polyhedral shapes as well as curved or sharp edges
- Utilizes octree in place of a 3D uniform grid for better efficiency
- Uses normals to define Quadratic Error Function for each leaf of the octree, inspired by original Extended Marching Cubes (EMC)

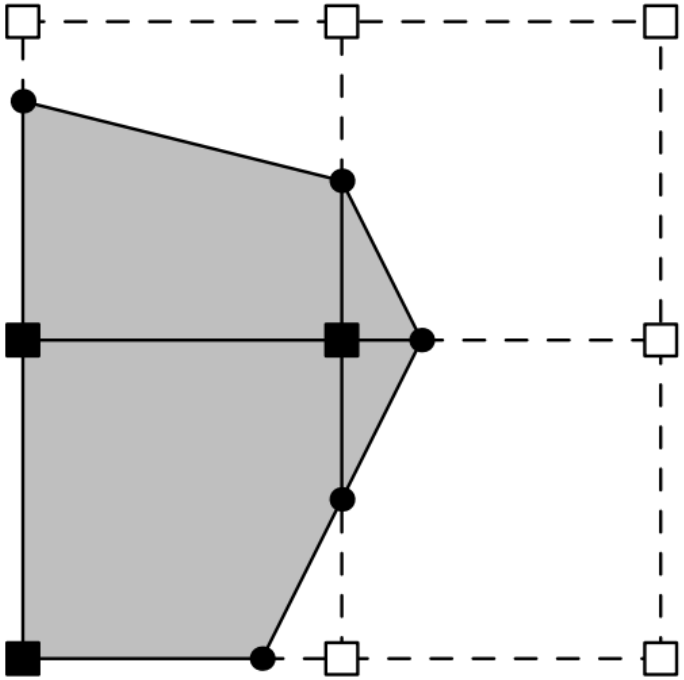
$$E[x] = \sum_i (n_i * (x - p_i))^2$$

where pi and ni correspond to the intersections and normals of the contour with the edges of cube

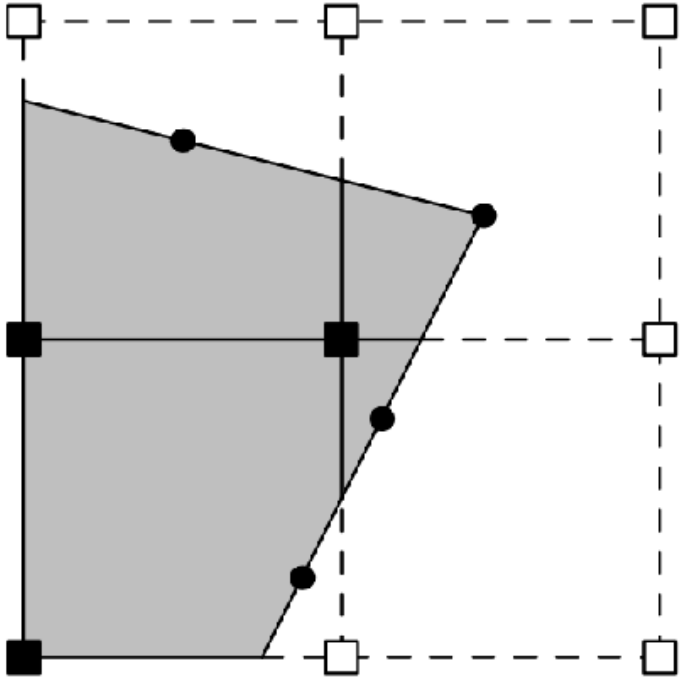
Dual Contouring



Signed Grid with edges tagged by Hermite Data



Marching Cubes Contour



Dual Contour

Challenges with Dual Contouring

- Limited expressivity due to reliance on structured grids
- Cannot faithfully extract very thin surfaces and features (otherwise requiring very fine grid resolution)
- Limited by the amount of surface simplification

Dual Marching Cubes

- Align features of the implicit function with the features of the structured grid; enabling extraction of sharp features
- Can generate adaptive polygonalizations that are crack-free and topologically manifold
- Can reproduce thin features in surface without excessive subdivision of the octree

Dual Marching Cubes

DMC's QEF is calculated by computing a tangent planes to the implicit surface representation f on a grid of points (x_i, y_i, z_i) sampled over c.

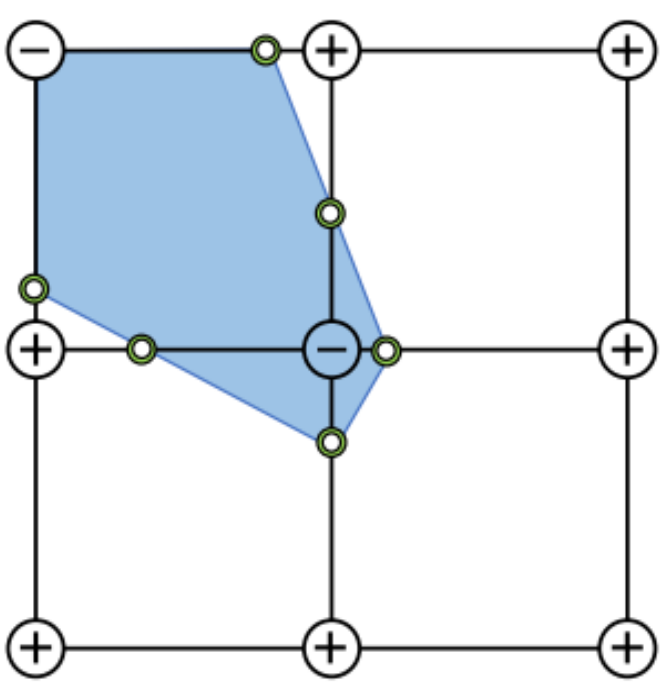
Tangent plane's equation: $T_i(x, y, z) = \nabla f(x_i, y_i, z_i)\ ((x, y, z) - (x_i, y_i, z_i))$

DMC's QEF sums over all sample points yielding:

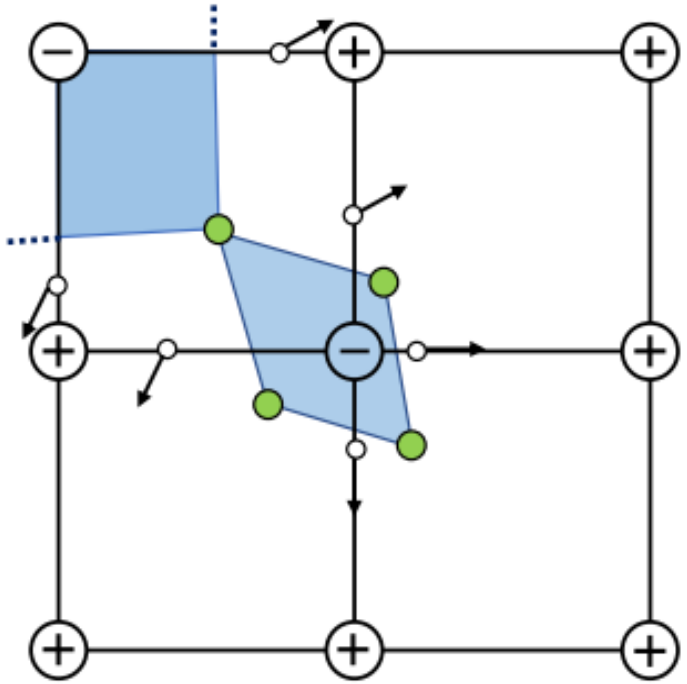
$$E(f(x, y, z), x, y, z) = \sum_i (f(x, y, z) - T_i(x, y, z))^2/(1+|\nabla f(x_i, y_i, z_i)|^2)$$

where the denominator normalizes the contribution of each tangent plane

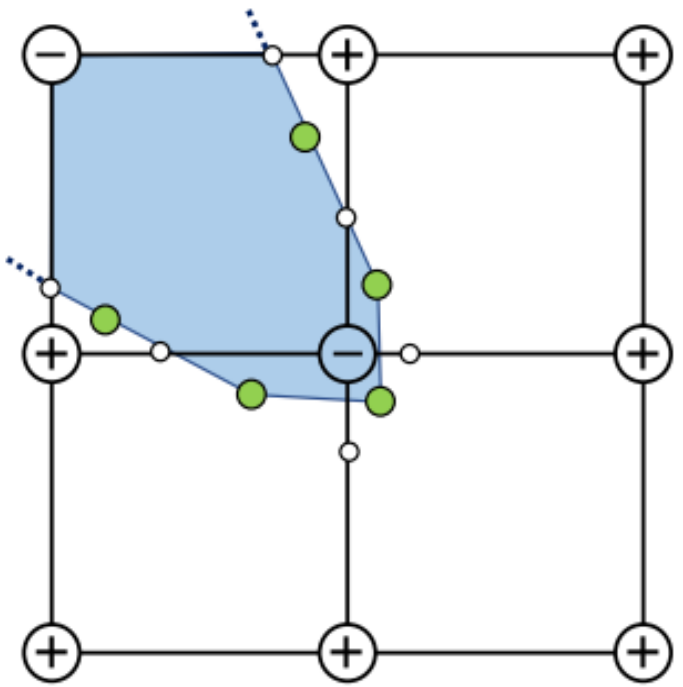
Comparison between Marching Cubes & Dual Contouring & Dual Marching Cubes



MC

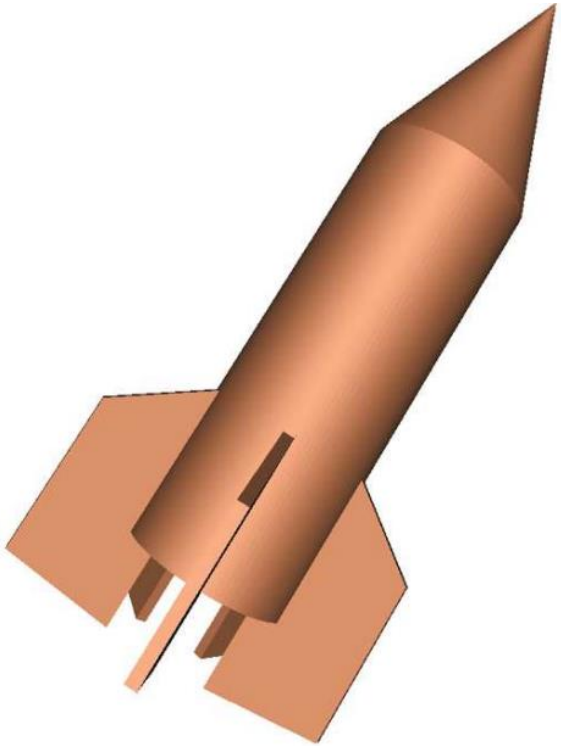


DC

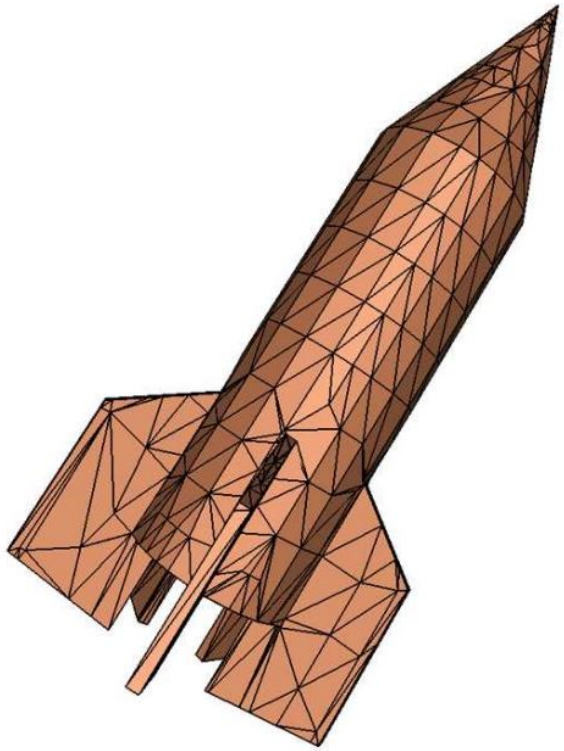


DMC

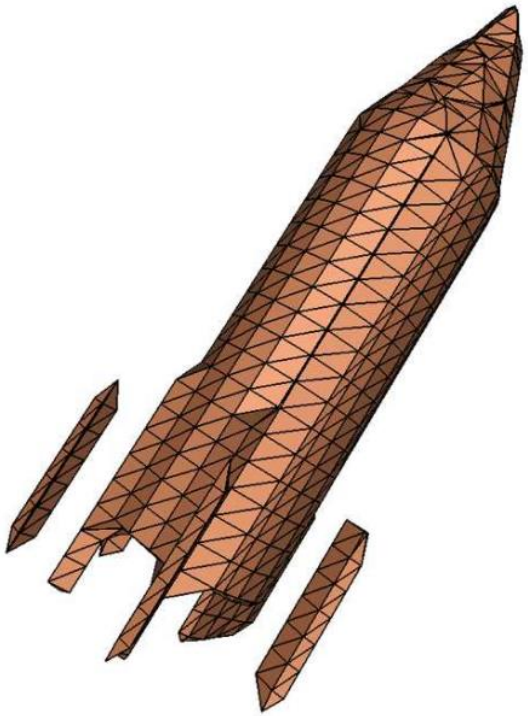
Comparison between Marching Cubes & Dual Contouring & Dual Marching Cubes



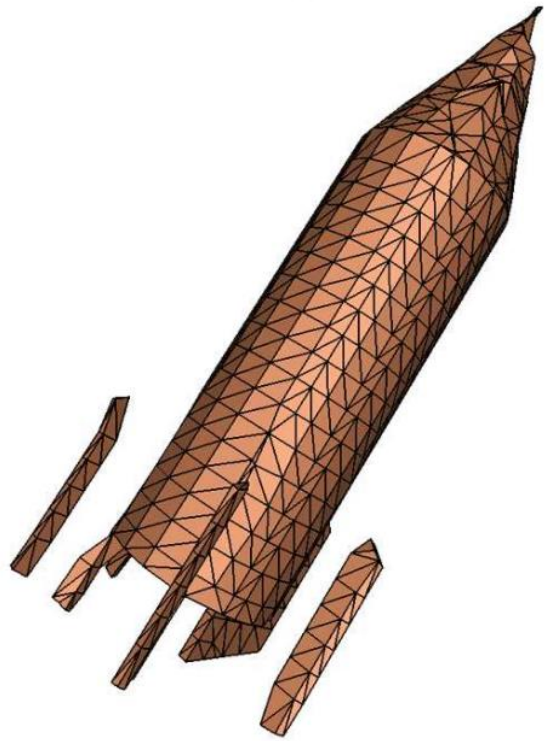
Ground Truth Model



Dual Marching Contour



Marching Contour

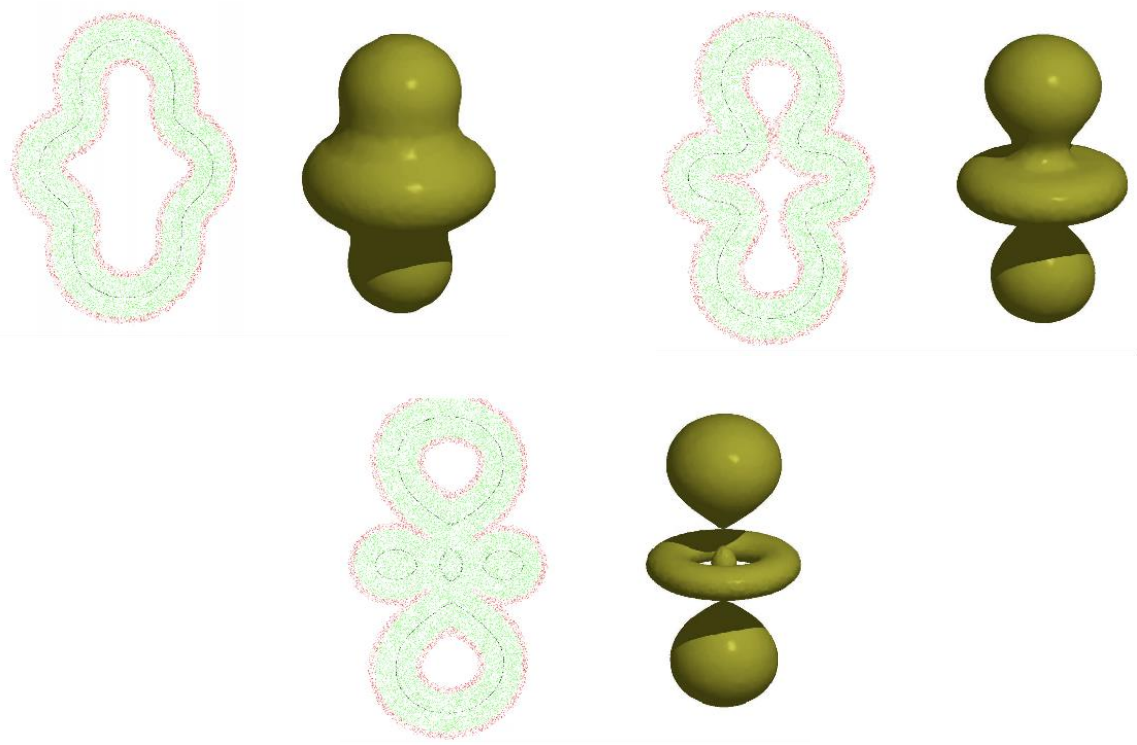


Dual Contour

Challanges faced by Level Sets method

Although Level Set representation provides much more explicit control over shape (like a texture), it

- runs into problems of aliasing! (Unlike closed-form expressions)
- induces $O(n^3)$ storage costs for 3D surface extraction
- can save space by only storing a narrow band around the surface.



Implicit Representations - Pros & Cons

Pros:

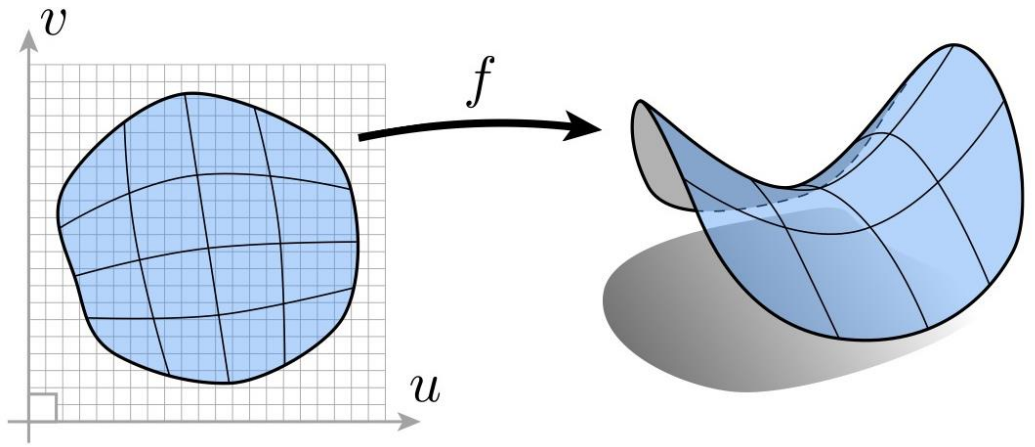
- description can be very compact (e.g., a polynomial)
- easy to determine if a point is in our shape (just plug it in!)
- other queries may also be easy (e.g., distance to surface)
- for simple shapes, exact description/no sampling error
- easy to handle changes in topology (e.g., fluid)

Cons:

- expensive to find all points in the shape (e.g., for drawing)
- very difficult to model complex shapes

"Explicit" Representations of Geometry

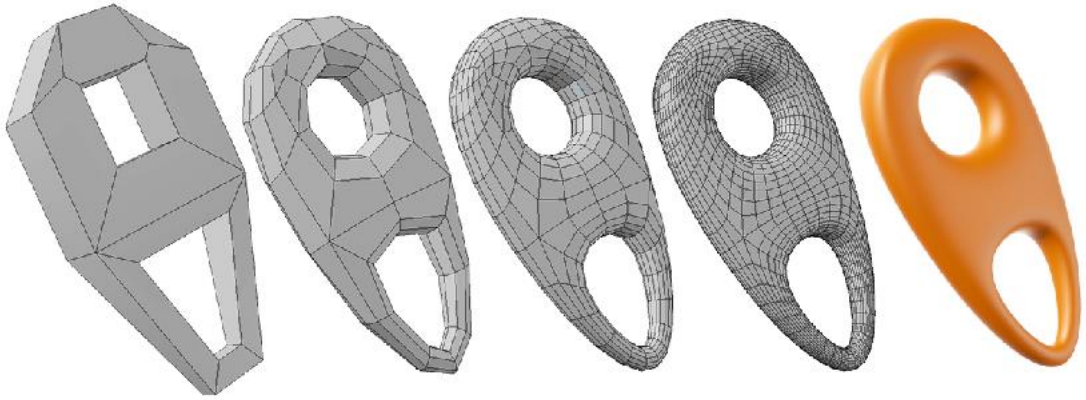
- All points are given directly
- E.g., points on sphere are $(\cos(u)\sin(v), \sin(u)\sin(v), \cos(v)),$
$\text{for } 0 \leq u < 2\pi \text{ and } 0 \leq v \leq \pi$
- More generally: $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \mapsto (x, y, z)$

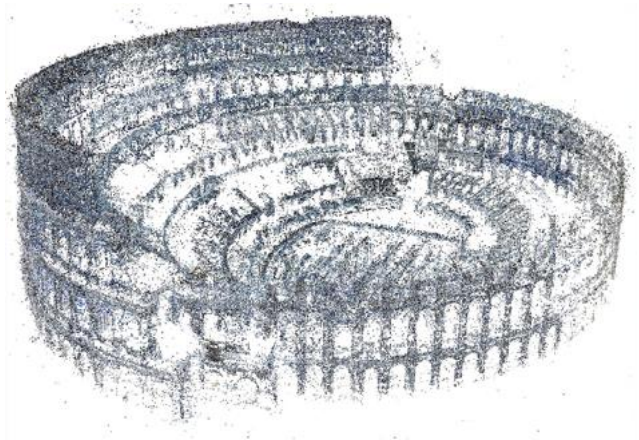


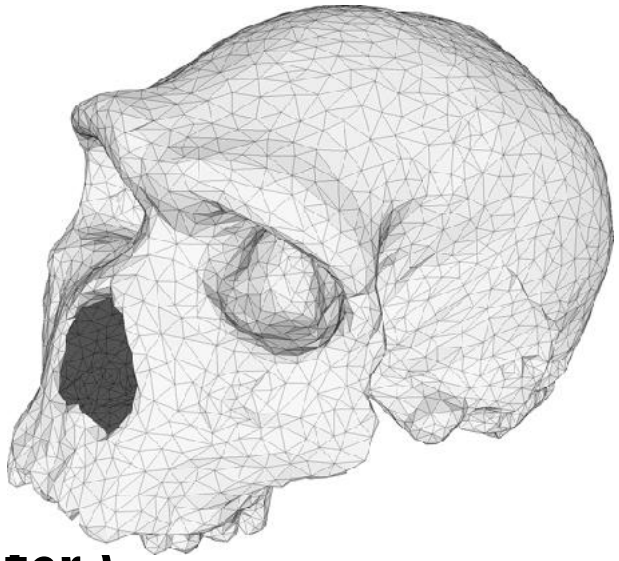
- (Might have a bunch of these maps, e.g., one per triangle!)

Many explicit representations in graphics

- triangle meshes
- polygon meshes
- subdivision surfaces
- NURBS
- point clouds
- …







(Will see some of these a bit later.)

But first, let's play a game:

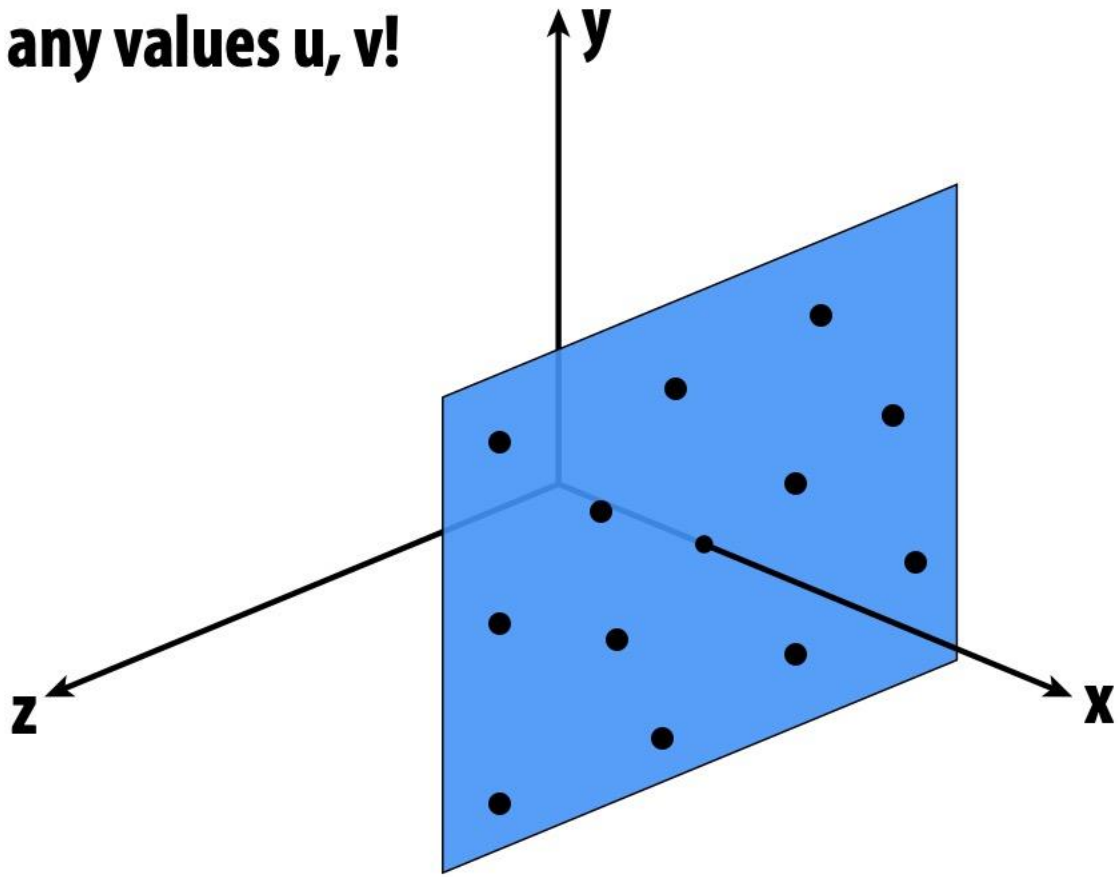
I'll give you an explicit surface.

You give me some points on it.

Sampling an explicit surface

My surface is f(u, v) = (1.23, u, v).

Just plug in any values u, v!



Explicit surfaces make some tasks easy (like sampling).

Let's play another game.

I have a new surface f(u,v).

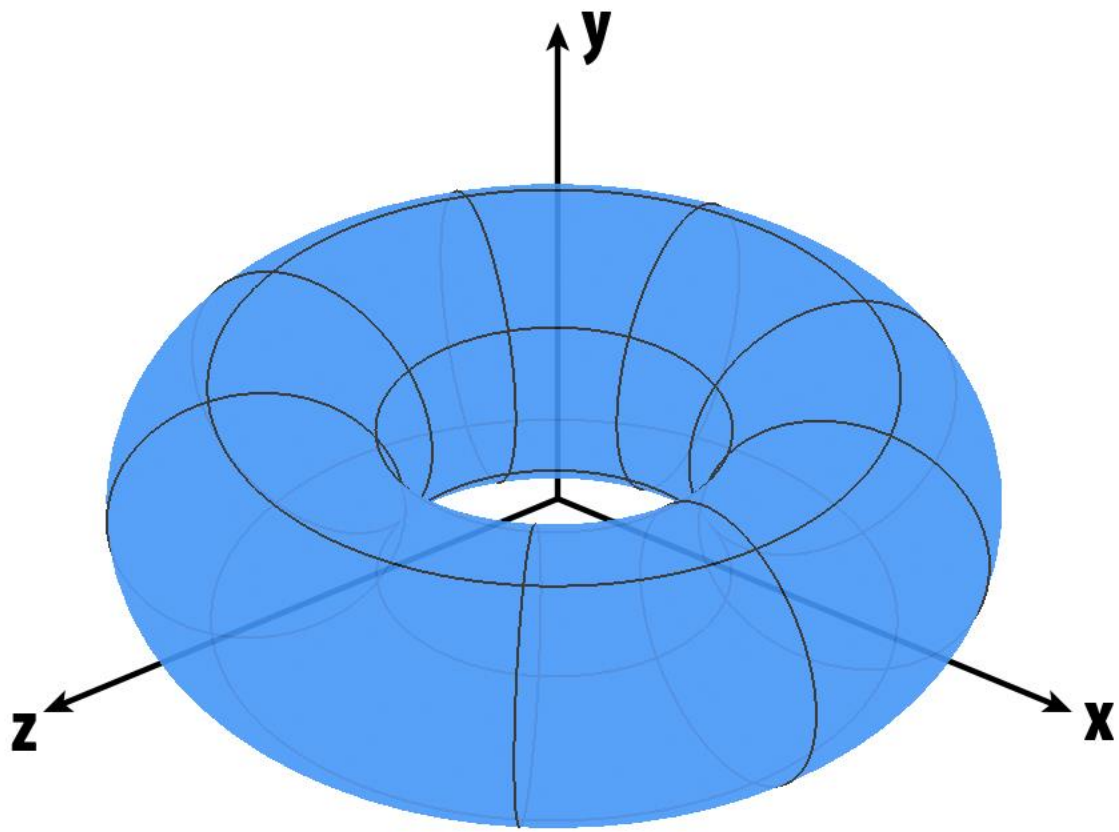
I want to see if a point is *inside* it.

Check if this point is inside the torus

My surface is f(u,v) = ((2+cos u)cos v, (2+cos u)sin v, sin u)

How about the point (1.96, -0.39, 0.9)?

...NO!

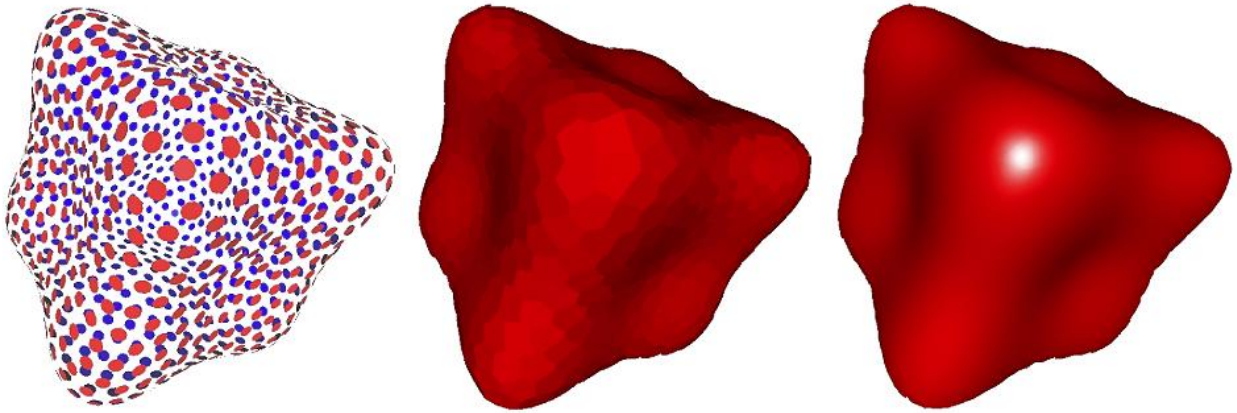


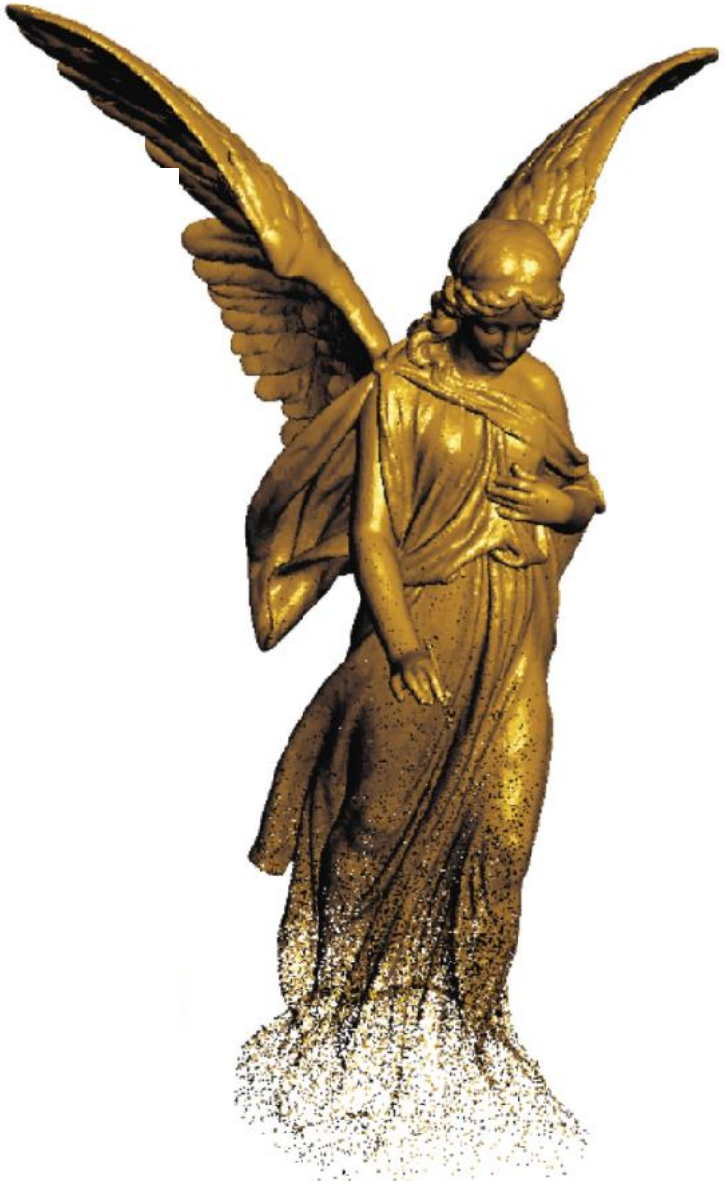
Explicit surfaces make other tasks hard (like inside/outside tests).

CONCLUSION:
Some representations work better than others—depends on the task!

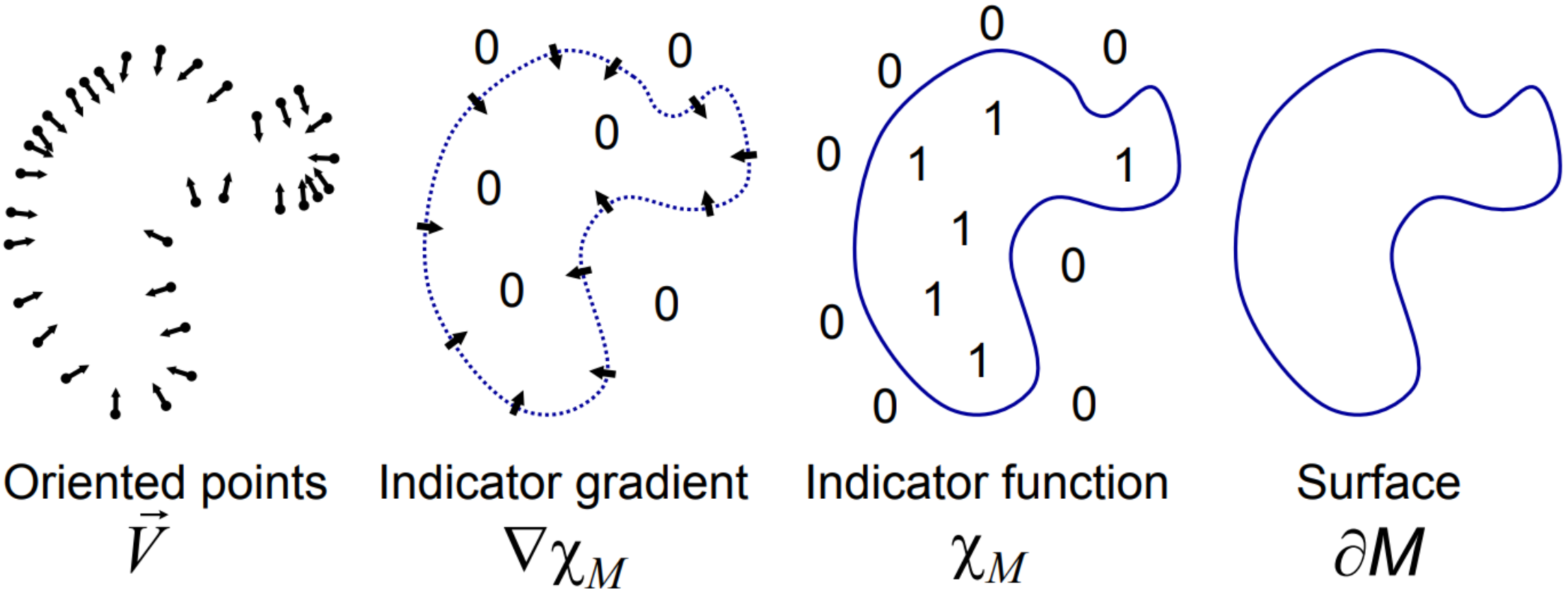
Point Clouds

- Easiest representation: list of points (x,y,z)
- Often augmented with normals
- Easily represent any kind of geometry
- Easy to draw dense cloud (>>1 point/pixel)
- Hard to interpolate under-sampled regions
- Hard to do processing / simulation/...





Poisson Surface Reconstruction



Kazhdan et al. **PSR**, ESGP, June 2006

Poisson Surface Reconstruction

Given an oriented point cloud P with points $p_1, \ldots, p_n$ and corresponding (outward-facing) normal observations $n_1, \ldots, n_n$, we consider the implicit reconstruction task of finding a function $f : \mathrm{R}^d \rightarrow \mathrm{R}$ such that

$$f(p_i) = 0 \, , \, \nabla f(p_i) = n_i \, , \, \forall i \in \{1, \ldots, n\} \, .$$

PSR first utilizes convolutional kernel F to interpolate n_i into a vector field $V : \mathrm{R}^d \rightarrow \mathrm{R}^d$ defined in box B containing P as

$$V(x) = \Sigma_i \, F(x, x_i) \, n_i$$

Kazhdan et al. **PSR**, ESGP, June 2006

Poisson Surface Reconstruction

f : $R^d \rightarrow R$ is then defined as a function whose gradient best matches V

$$f = \text{argmin}_{\,g} \int_{B} \|V(x) - \nabla g(x)\|_2 \, dx$$

Variational problem above is equivalent to the Poisson Equation

$$\Delta f = \nabla \cdot V(x)$$

which is solved by PSR via discretization using the Finite Element Method on an octree and solve using a purpose-built multigrid algorithm

Kazhdan et al. **PSR**, ESGP, June 2006

Poisson Surface Reconstruction

$f : R^d \rightarrow R$ is then defined as a function whose gradient best matches V

$$f = \text{argmin}_g \int_B \|V(x) - \nabla g(x)\|_2 \, dx$$

Variational problem above is equivalent to the Poisson Equation

$$\Delta f = \nabla V(x)$$

which is solved by PSR via discretization using the Finite Element Method on an octree and solve using a purpose-built multigrid algorithm. In Screened PSR (Kazhdan et.al, 2013) authors use

$$f = \text{argmin}_g \int_B \|V(x) - \nabla g(x)\|_2 \, dx + \alpha * \sum_i g(p_i)^2$$

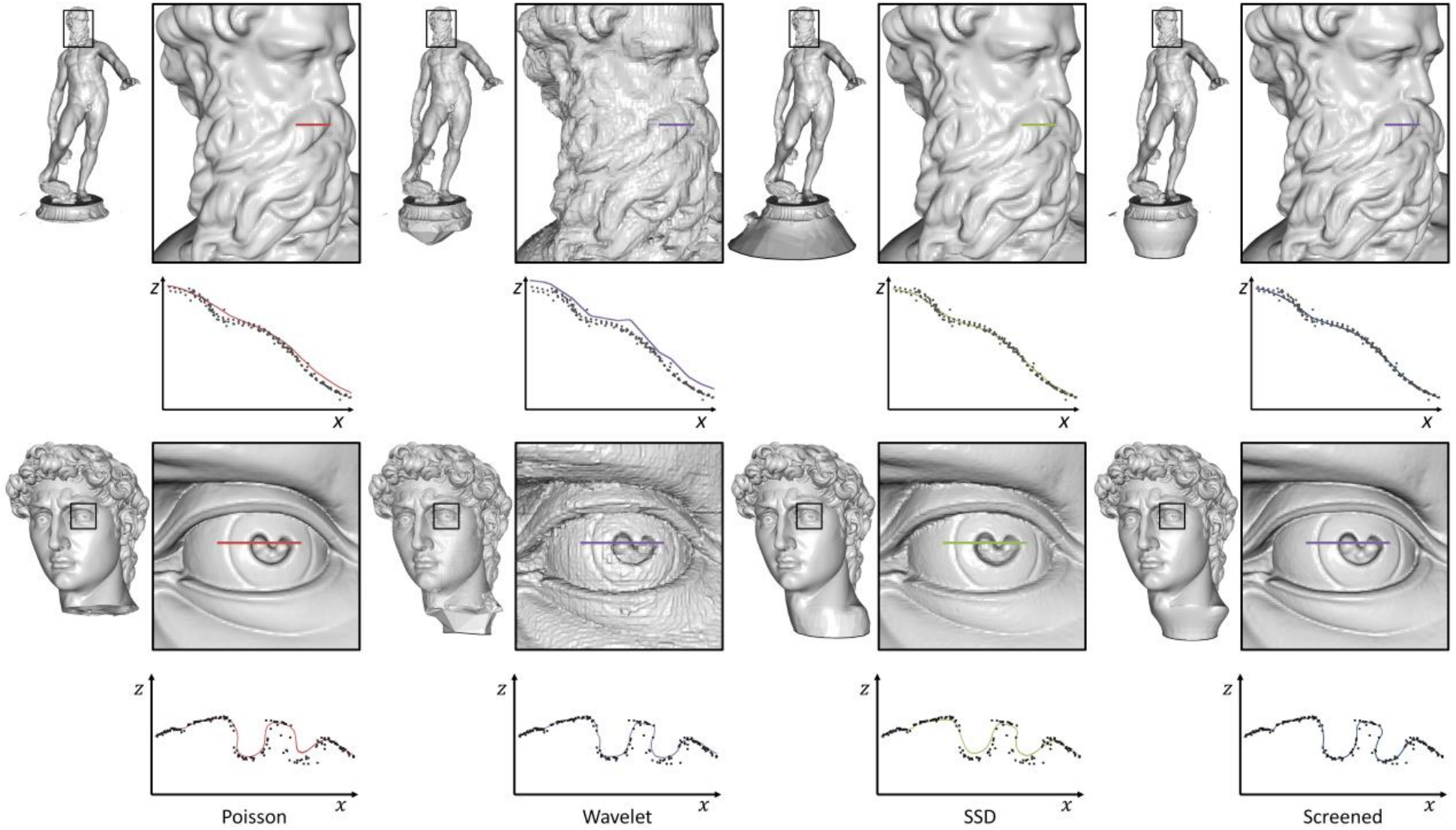
Kazhdan et al. **PSR**, ESGP, June 2006

(Screened) Poisson reconstruction

- Can extract the surface from oriented point clouds
- (Screened PSR) efficiently implements sparsity constraints
- Can better capture the details of the underlying surface
- (Screened PSR) implements efficient octree structure and multigrid algorithm to reduce the time complexity

However, these methods limit themselves to outputting the likeliest surface given the point cloud observations and their assumed prior. Whereas in reality there could be theoretically infinite number of possible surfaces representing same set of Points with normals

(Screened) Poisson reconstruction

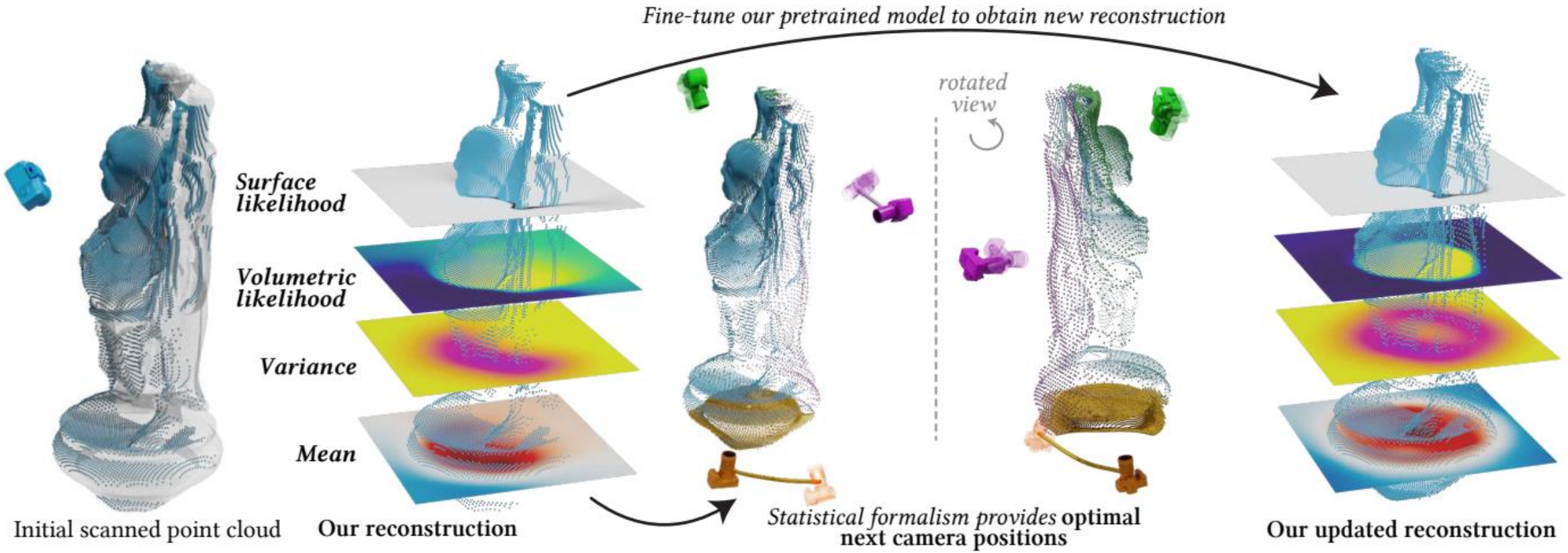


Neural Stochastic Screened Poisson Surface Reconstruction

- Take a step even further and models the posterior multivariate Gaussian distributions representing the likelihood of all possible reconstructions for the given Point Set with normals
- Proposes the parametrization of mean and covariance functions using neural networks, optimizing them through gradient-based methods for a more efficient and flexible approach
- Enabling computation of the statistical quantities over the reconstruction

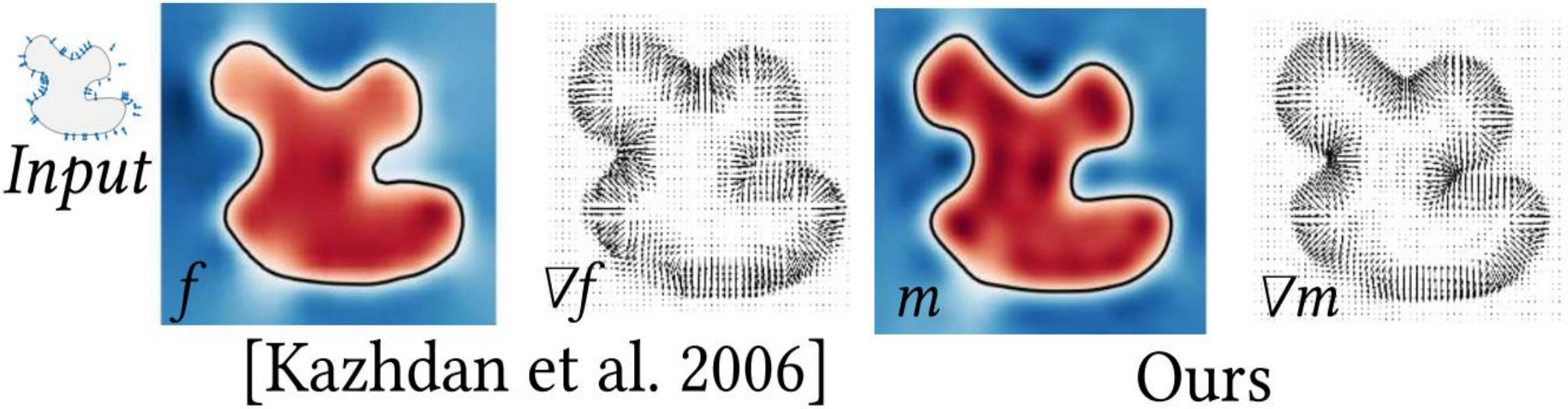
Sellan and Jacobson, **Neural Stochastic Screened PSR**, SIGGRAPH Asia 2023

Neural Stochastic Screened Poisson Surface Reconstruction



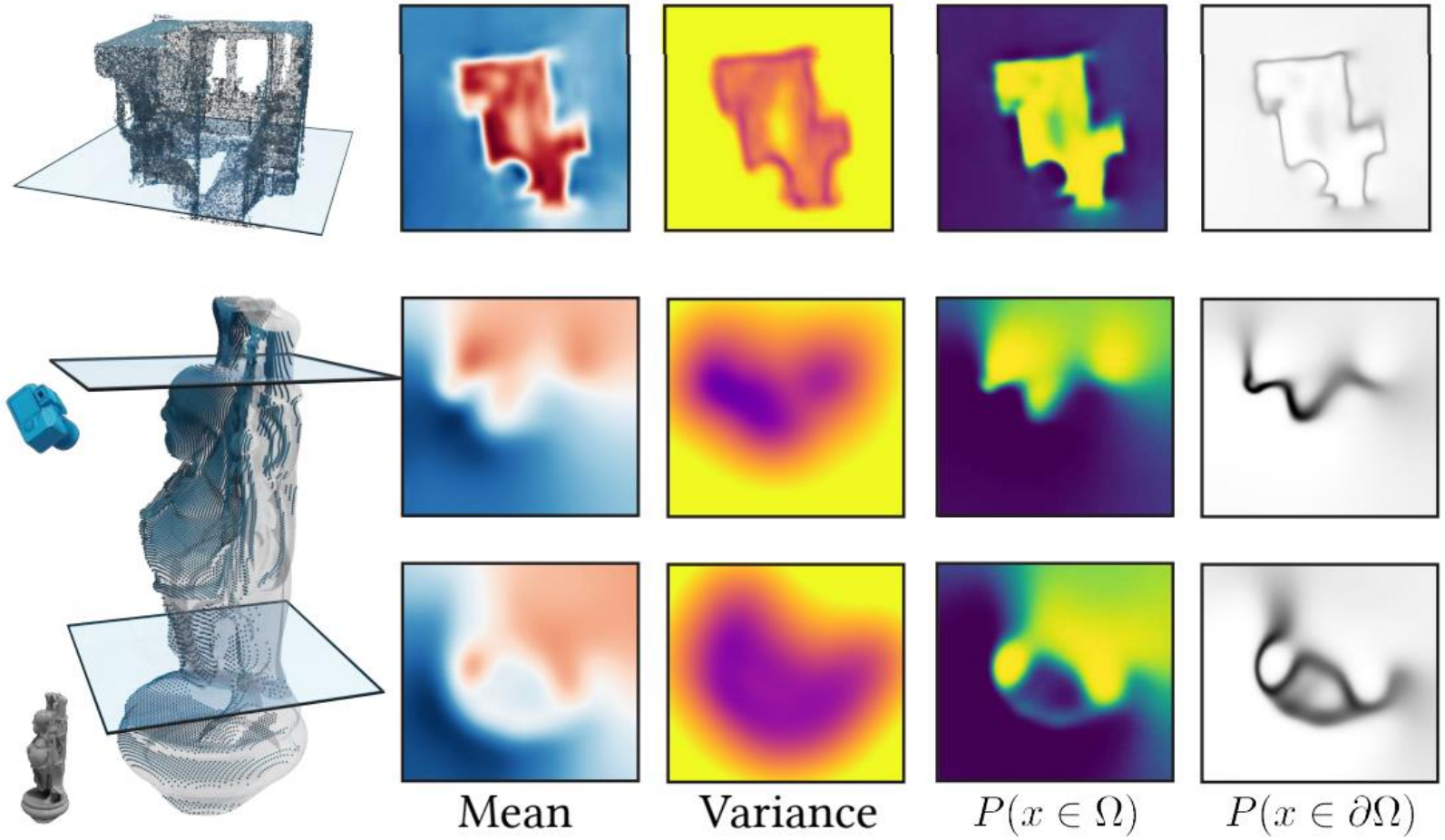
Sellan and Jacobson, **Neural Stochastic Screened PSR**, SIGGRAPH Asia 2023

Neural Stochastic Screened Poisson Surface Reconstruction



Sellan and Jacobson, **Neural Stochastic Screened PSR**, SIGGRAPH Asia 2023

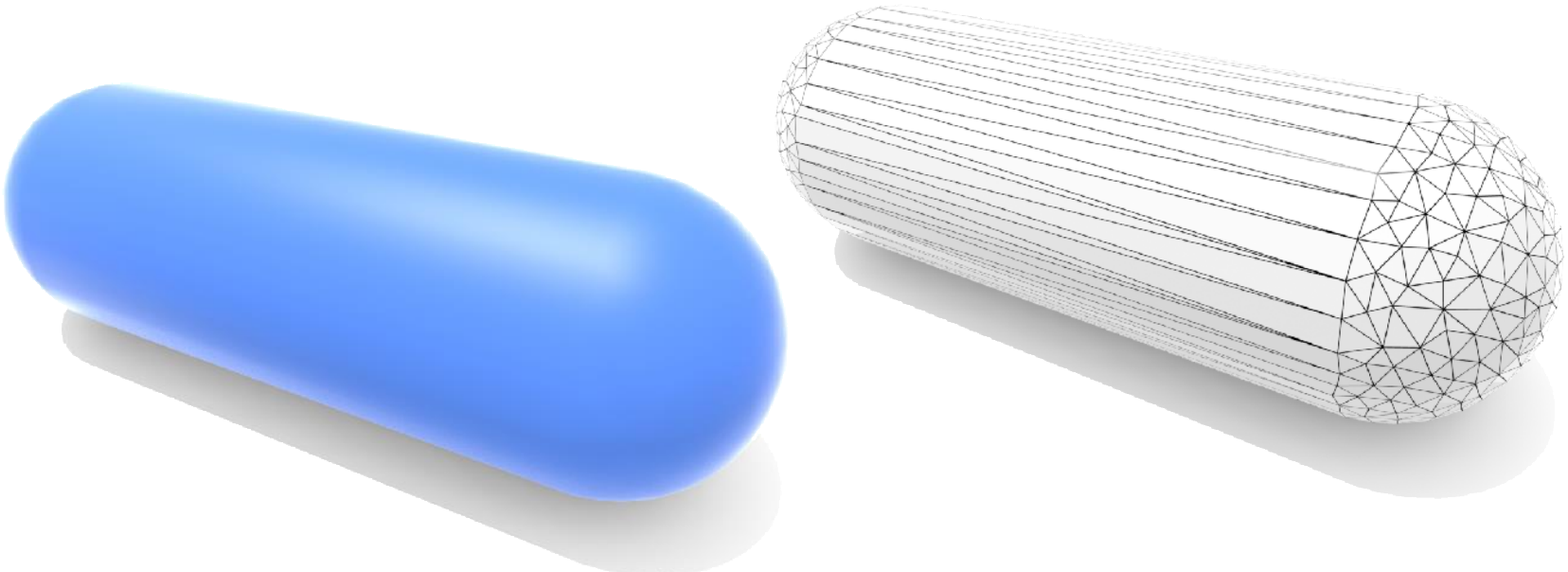
Neural Stochastic Screened PSR



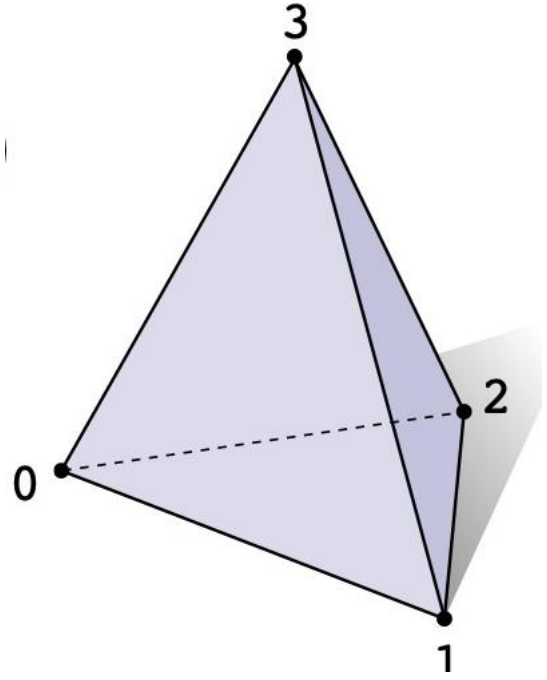
Sellan and Jacobson, **Neural Stochastic Screened PSR**, SIGGRAPH Asia 2023

Polygon Mesh

- Store vertices and polygons (most often triangles or quads)
- Easier to do processing/simulation, adaptive sampling
- More complicated data structures
- Irregular neighbourhoods



Triangle Mesh (Explicit)



- Store vertices as triples of coordinates (x,y,z)
- Store triangles as triples of indices (i,j,k)
- E.g., tetrahedron:

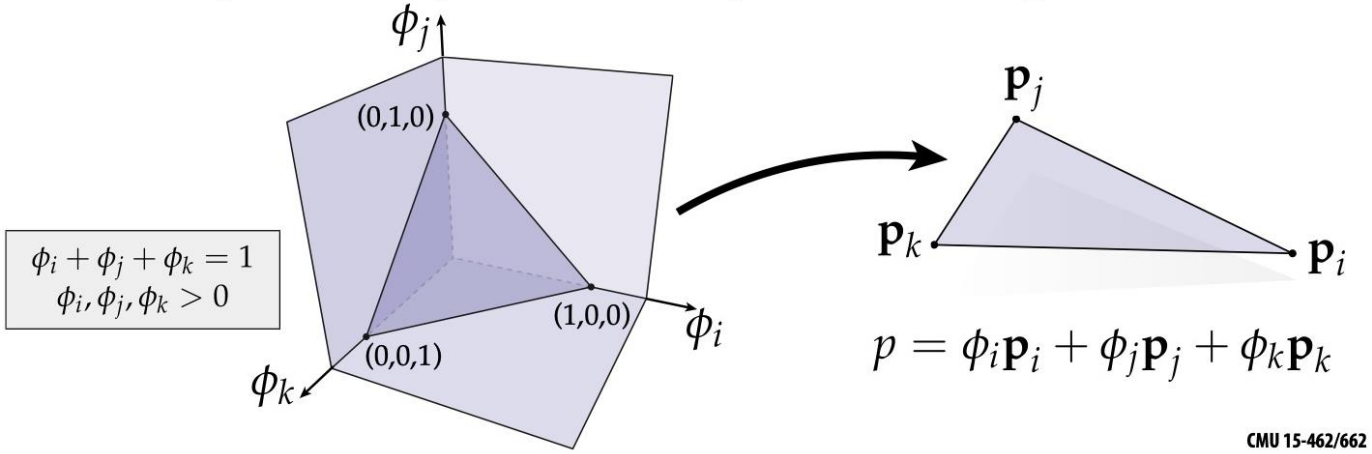
VERTICES

	x	y	z
0:	-1	-1	-1
1:	1	-1	1
2:	1	1	-1
3:	-1	1	1

TRIANGLES

	i	j	k
	0	2	1
	0	3	2
	3	0	1
	3	1	2

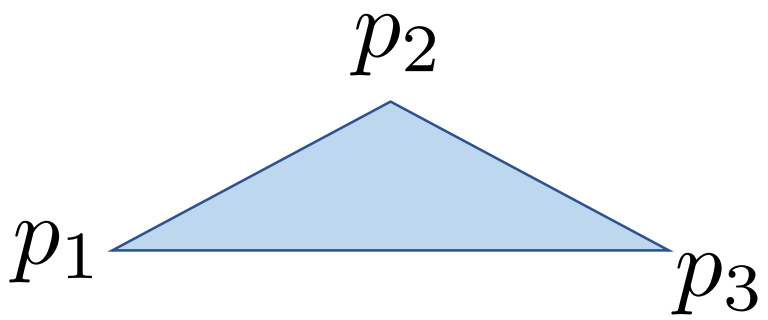
- Use barycentric interpolation to define points inside triangles:



Triangle Mesh – Normals (Explicit)

- Option1: Normals per face (Phong)

$$n = (p_2 - p_1) \times (p_3 - p_1)$$



- Option2: Interpolate incident face normal at each vertex, and interpolate with barycentric interpolation for points inside the triangle

$$n(\phi_1, \phi_2, \phi_3) = \phi_1 n_1 + \phi_2 n_2 + \phi_3 n_3$$

Distance queries and closest points

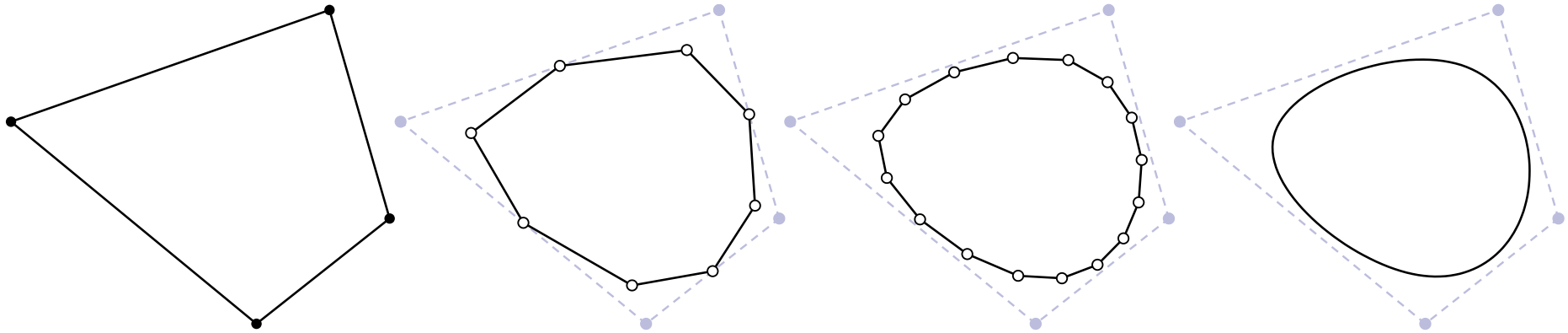
- Smooth parametric surface
  - Evaluation $f(u,v)$
  - Normals $n(u,v) = f_u(u,v) \times f_v(u,v)$
  - Distance and closest point. Find surface point f(u,v) such that:

$$[\mathbf{q} - f(u,v)] \times \mathbf{n}(u,v) = 0$$

- Mesh
  - Distance and closest point:
    - Find closest triangle and then find closest point in triangle, and then finding point in triangle
    - Use Kd-tree, AAB Tree, to find closest triangles

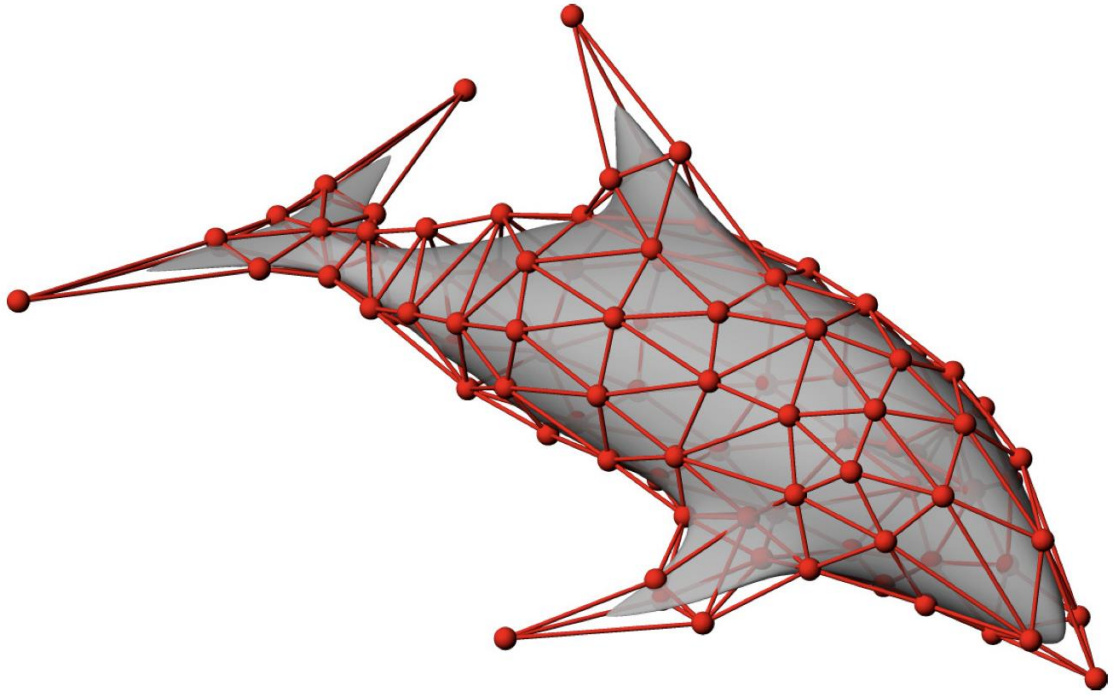
Subdivision

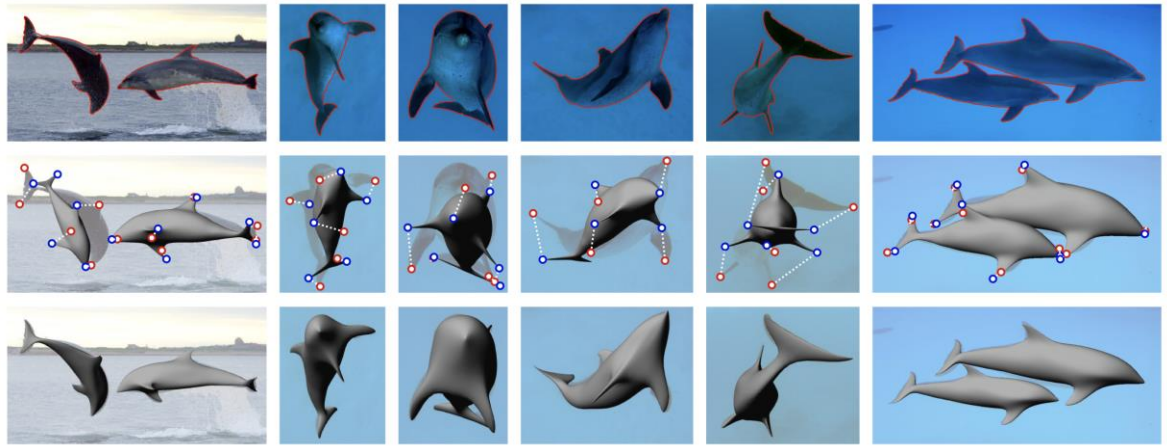
- Alternative starting point for curves/surfaces: subdivision
- Start with "control curve"
- Repeatedly split, take weighted average to get new positions
- For careful choice of averaging rule, approaches nice limit curve
  - Often exact same curve as well-known spline schemes!



Q: Is subdivision an explicit or implicit representation?

Subdivision Surfaces in Computer Vision

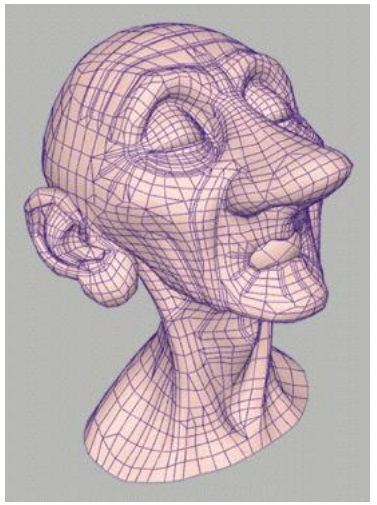


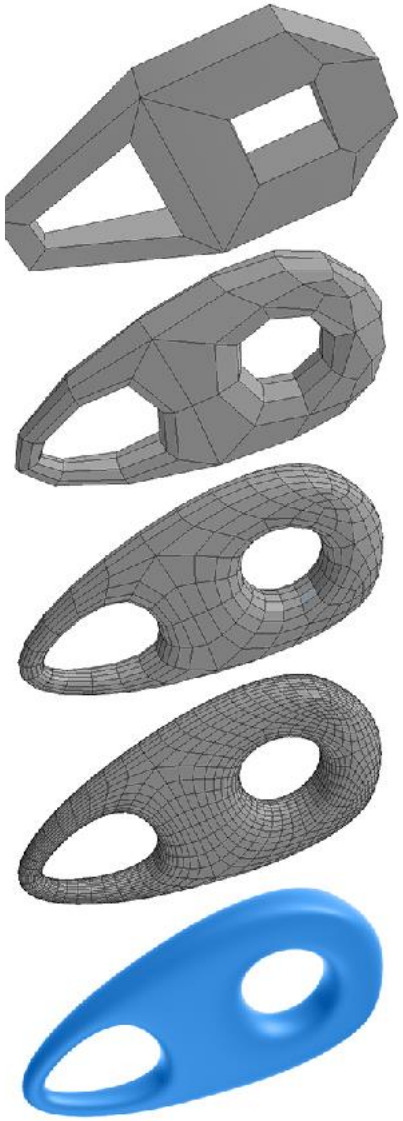


Cashman & Fitzgibbon. What shapes are dolphins?

Subdivision Surfaces (Explicit)

- Start with coarse polygon mesh ("control cage")
- Subdivide each element
- Update vertices via local averaging
- Many possible rules:
  - Catmull-Clark (quads)
  - Loop (triangles)
  - …
- Common issues:
  - interpolating or approximating?
  - continuity at vertices?
- Easier than splines for modeling; harder to evaluate pointwise
- Widely used in practice (2019 Academy Awards!)





Subdivision in Pixar (Pixar's "Geri's Game")

see: de Rose et al, "Subdivision Surfaces in Character Animation"

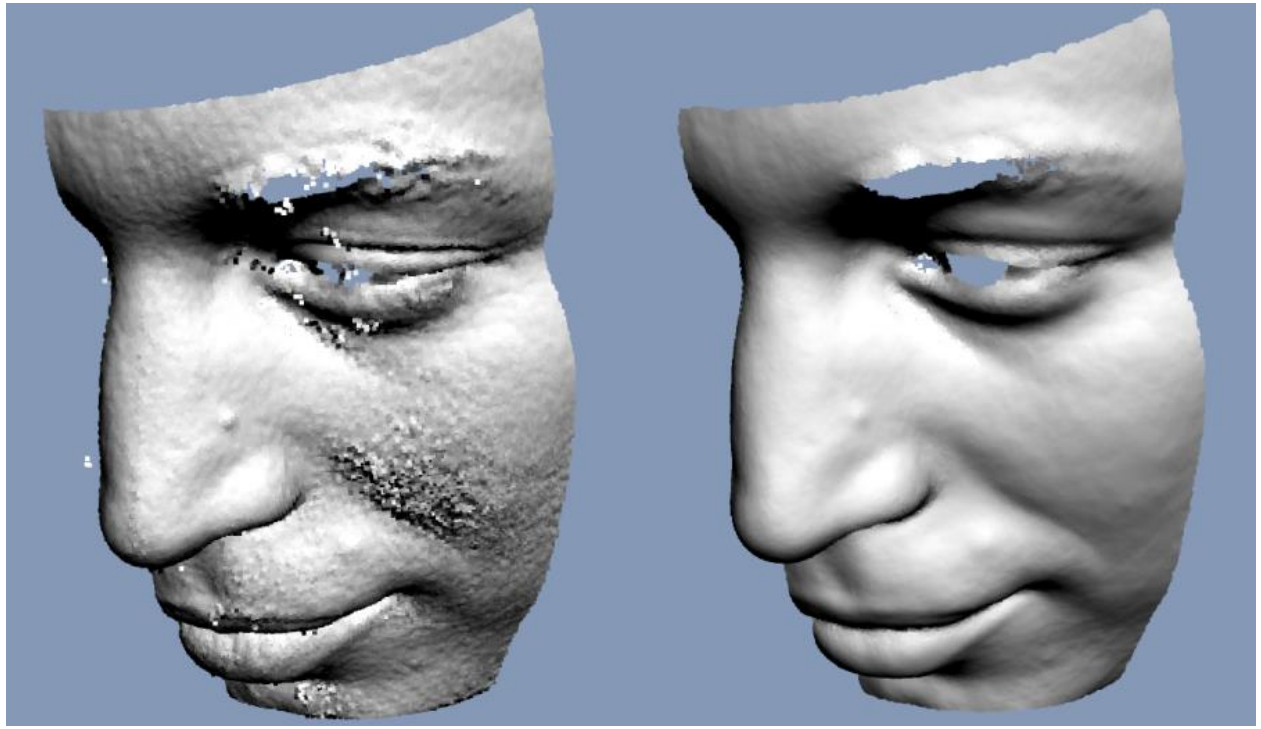
Slide credits and further reading

- Keenan Crane – Computer Graphics Lecture CMU 15-462/662. Lecture 09:Introduction to Geometry.

Point Cloud Denoising

Problems with Poisson Surface Reconstruction:

- cannot handle noise in input point cloud
- uneven distribution of points can make gradient estimation unreliable



Lipman et al. **LOP**, ACM (TOG), July 2007

Locally Optimal Projection Operator

Given an unorganized set of points $P = \{p_j\}$ for $j \in J \subset R^3$, we define a set of projected points $X = \{x_i\}$ for $i \in I \subset R^3$ by a fixed point iteration to minimize

$$\sum_{i \in I} \sum_{j \in J} ||x_i - p_j||_2 * \theta(||\xi^k_{ij}||) +$$

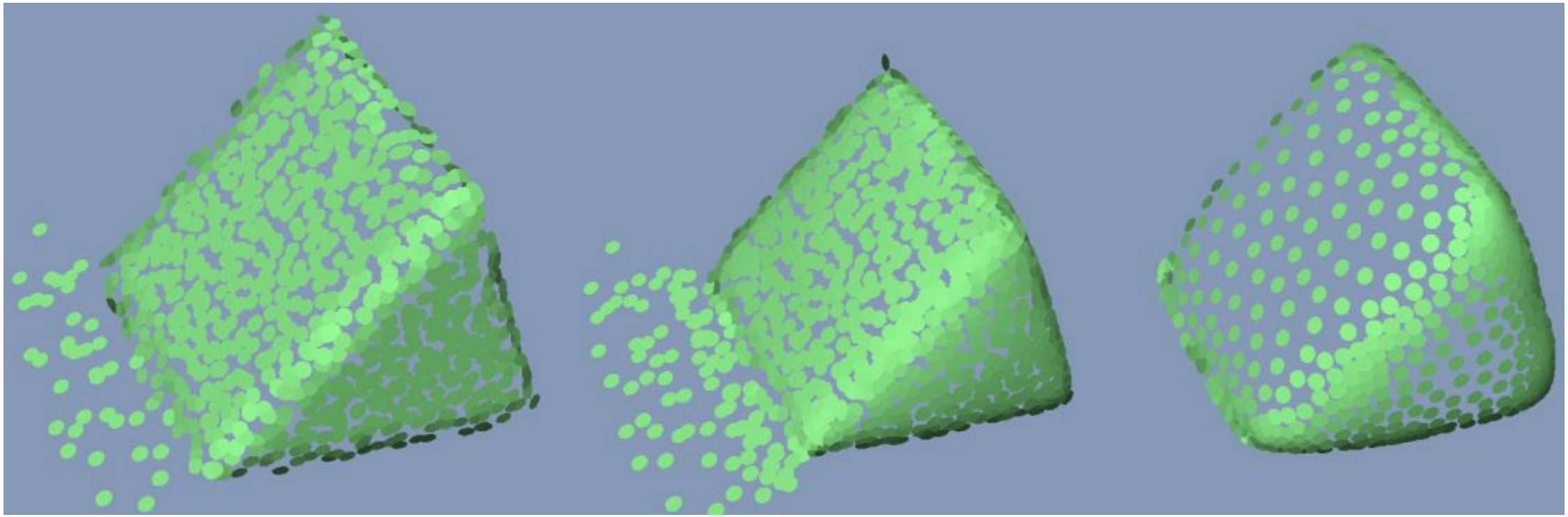
$$+ \lambda_i \sum_{i! \in I \setminus \{i\}} \eta(||x_i - x^k_{i'}||)\theta(||\delta^k_{ii'}||)$$

where $\xi^k_{ij} = x^k_i - p_j$, $\delta^k_{ii'} = x^k_i - x^k_{i'}$, $\theta(r) = e^{-r**2/(h/4)**2}$ and $\eta(r)=(1/3r^3)$.

In practice, $n = |I|$ is often significantly smaller than $m = |J|$.

Lipman et al. **LOP**, ACM (TOG), July 2007

Locally Optimal Projection Operator



Contaminated Point Set

MLS Projections

LOP Projections

Lipman et al. **LOP**, ACM (TOG), July 2007

Locally Optimal Projection Operator

However, LOP has the following disadvantages:

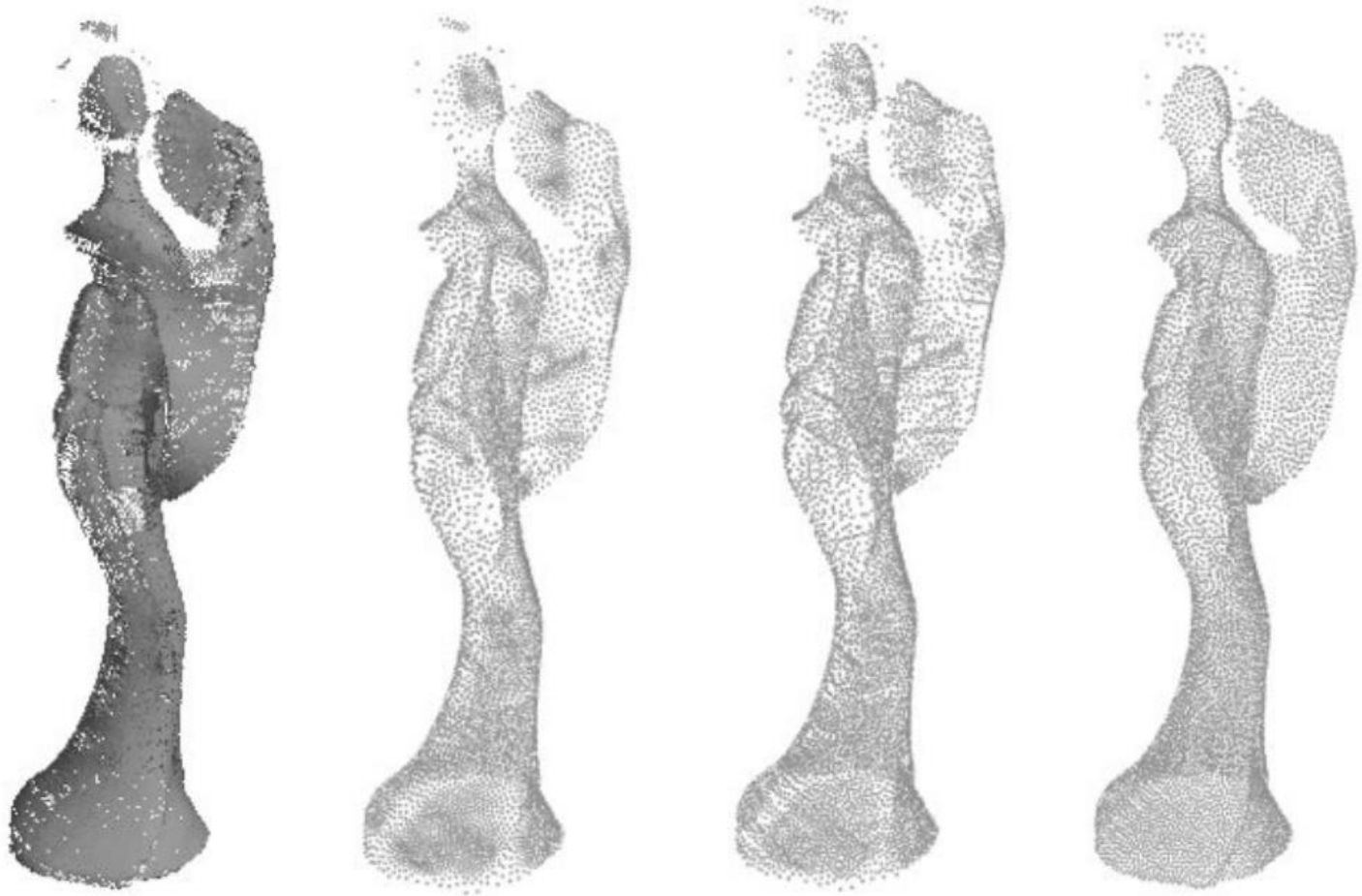
- Repulsion term $\eta(r)$ decreases too drastically
- Imposes strong attraction between point clusters

Weighted LOP improves over original LOP by

- Replacing the original repulsion term $\eta(r) = 1/(3*r^3)$ with $\eta(r) = -r$ which decreases more gently and penalizes more at a large r
- Using weighted local density for each point in a Point Set that enables the attraction of point clusters to be relaxed and the repulsion force from points in dense areas to be strengthened

Lipman et al. **LOP**, ACM (TOG), July 2007

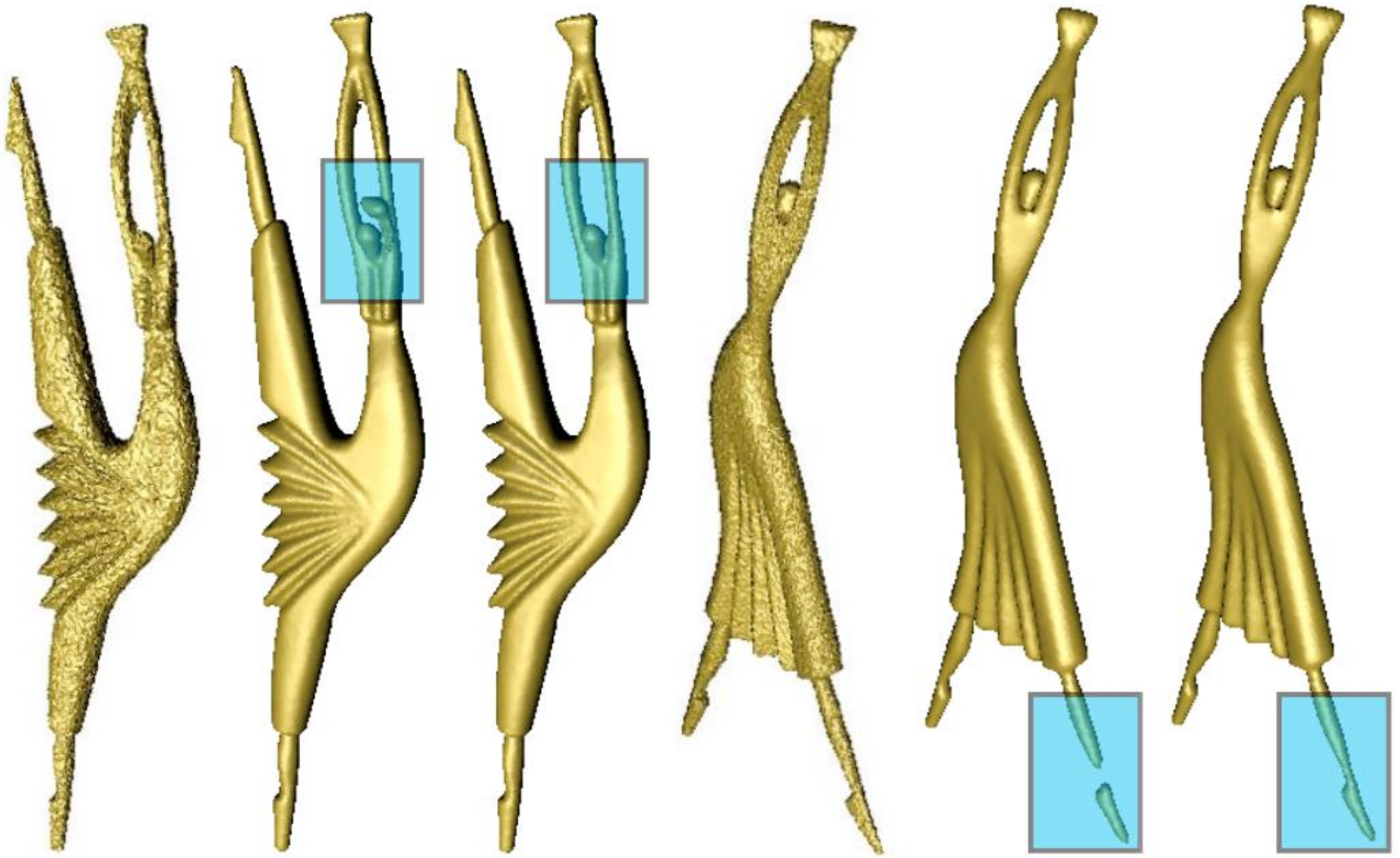
Weighted Locally Optimal Projection Operator



(a) Raw scan. (b) LOP (old η). (c) LOP (new η). (d) WLOP.

Huang et al. **WLOP**, ACM (TOG), December 2009

Weighted Locally Optimal Projection Operator



Huang et al. **WLOP**, ACM (TOG), December 2009

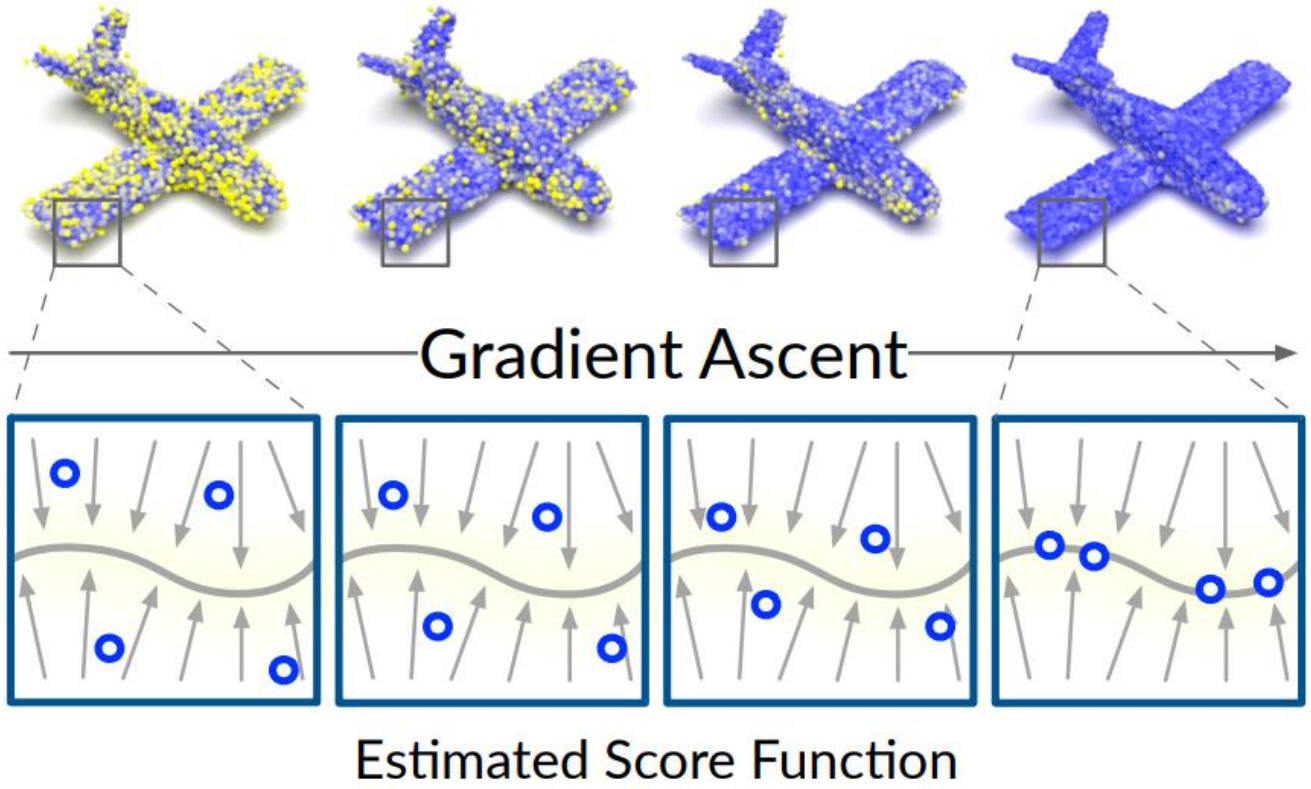
(Weighted) Locally Optimal Projection Operator

- not learning patterns from data
- mainly relied on parameters set heuristically
- applies Local PCA for estimation of point cloud normals
- cannot recover sharp features and handle missing data

Huang et al. **WLOP**, ACM (TOG), December 2009

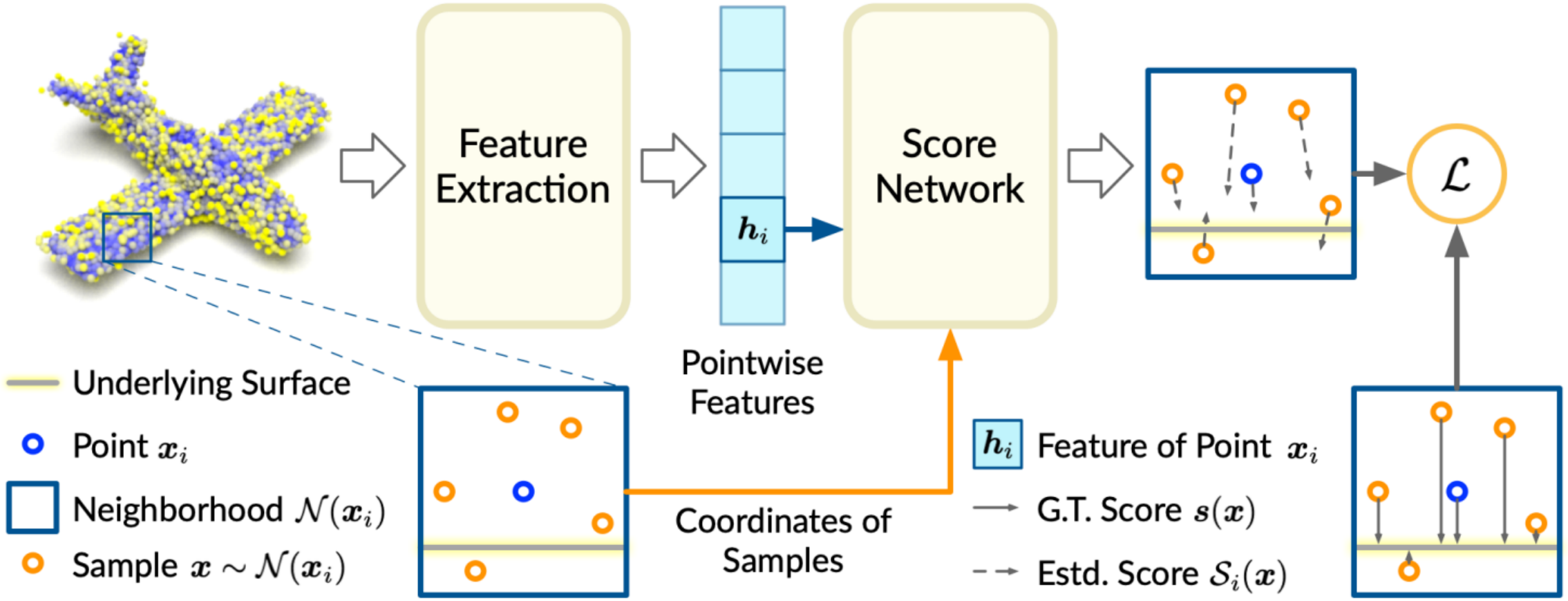
Score-Based Point Cloud Denoising

- Learns to estimate the score of the noisy point cloud
- Iteratively updating each point to increase log-likelihood via gradient descent
- Can learn variety of noisy models



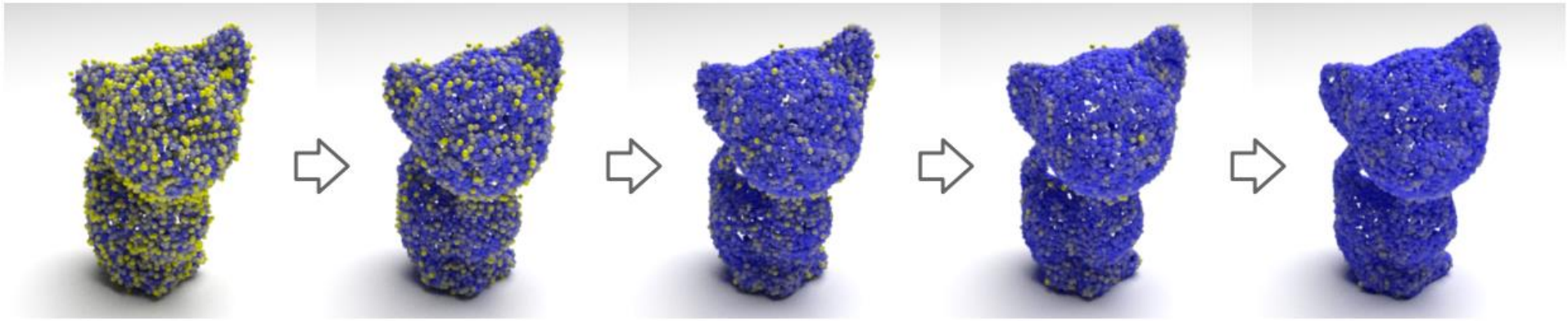
Luo et al. **Score-Based Point Cloud Denoising,** ICCV, July 2021

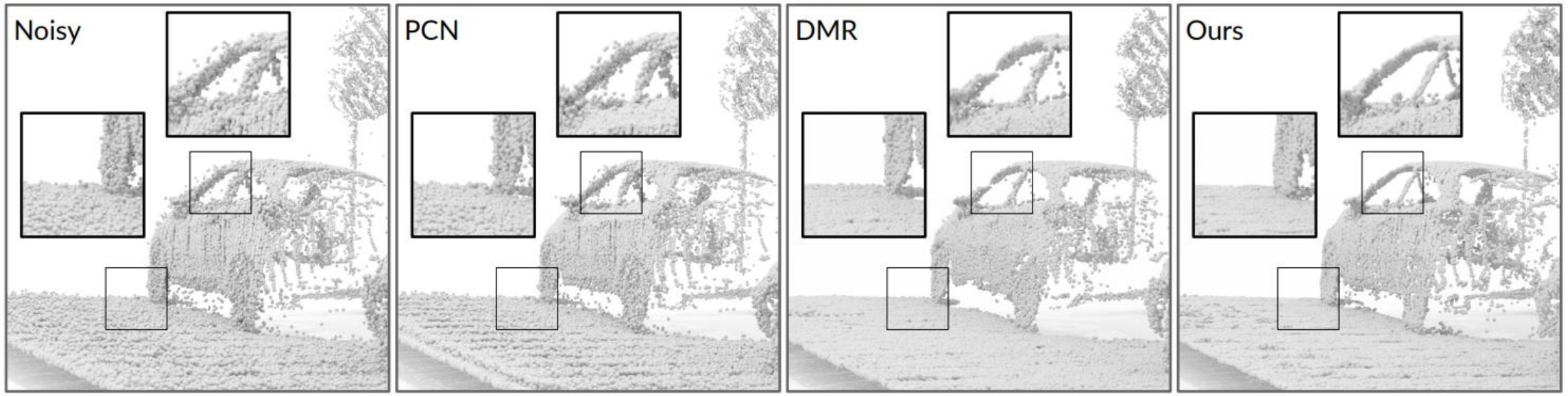
Score-Based Point Cloud Denoising



Luo et al. **Score-Based Point Cloud Denoising,** ICCV, July 2021

Score-Based Point Cloud Denoising





Luo et al. **Score-Based Point Cloud Denoising,** ICCV, July 2021

Score-Based Point Cloud Denoising

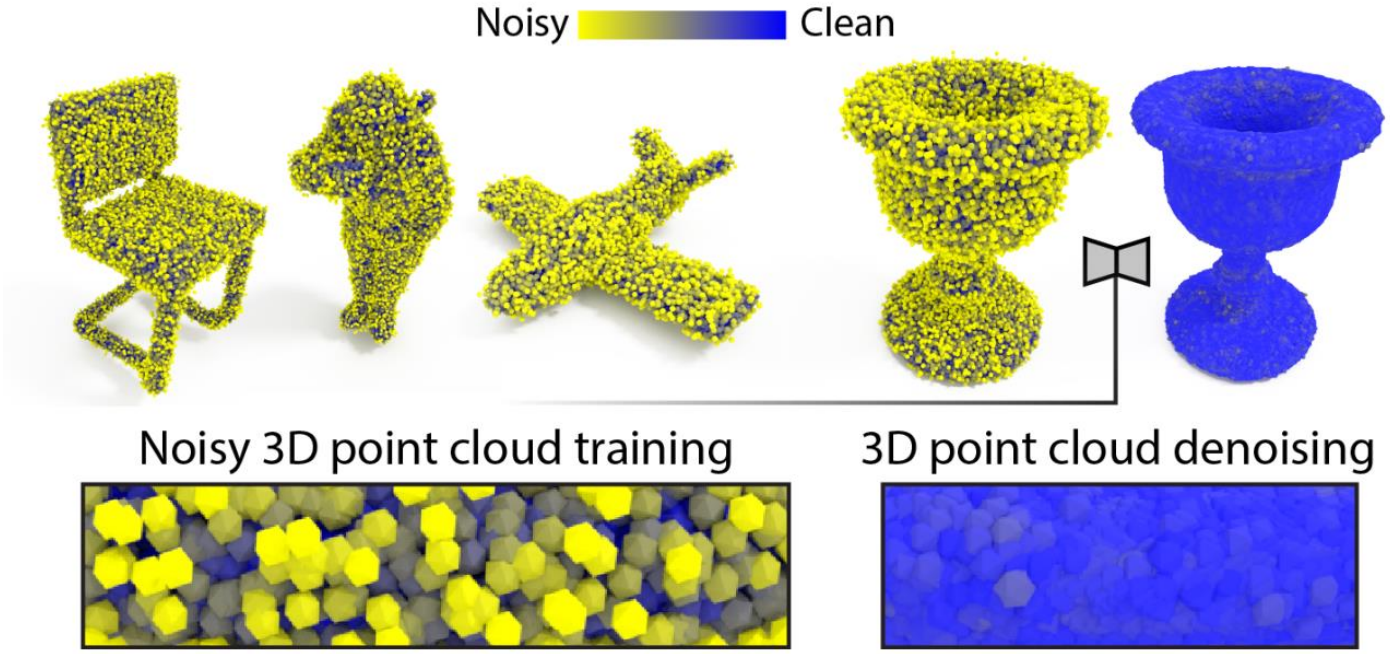
Although performing

- implements synthetic datasets generation by perturbing ground truth point clouds
- thus not scalable with the real world data
- restricts evaluation to pre-determined set of noise models, including Gaussian noise, simulated LiDAR noise, non-isotropic Gaussian noise, uni-directional noise, Laplace noise, uniform noise, and discrete noise

Luo et al. **Score-Based Point Cloud Denoising,** ICCV, July 2021

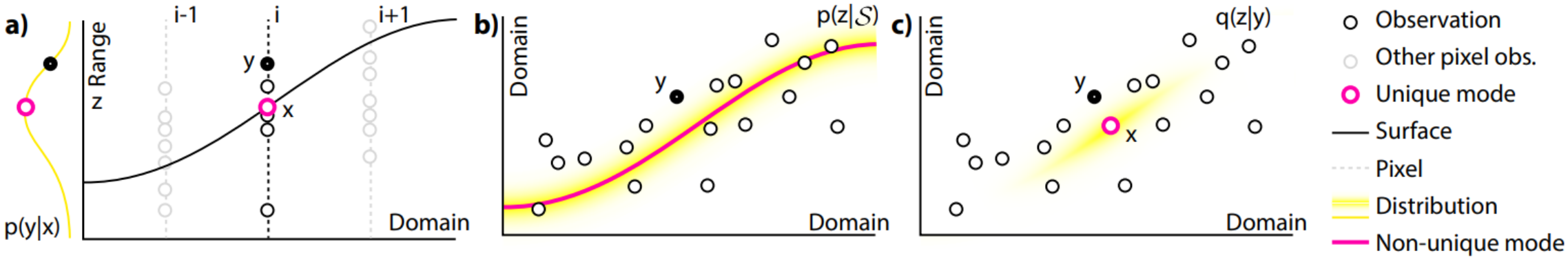
Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning

- Allows learning point cloud cleaning from noisy examples alone
- Enforce convergence to the unique closest out of many possible modes on a manifold of point clouds



Hermosilla et al. **Total Denoising**, ICCV, April 2019

Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning



To enforce convergence onto a unique mode, we enforce regularization over denoising process by imposing prior distribution q(z|y) that captures the probability that a given observation y is a realization of the clean point z

Hermosilla et al. **Total Denoising**, ICCV, April 2019

Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning

Using spatial and appearance proximity:

$$q(z|y) = p(z|S) * k(z - y)$$

$$k(d) = (1 / \sigma \sqrt{(2\pi)}) * \exp\{ - ||Wd||_2^2 /(2*\sigma^2)\}$$

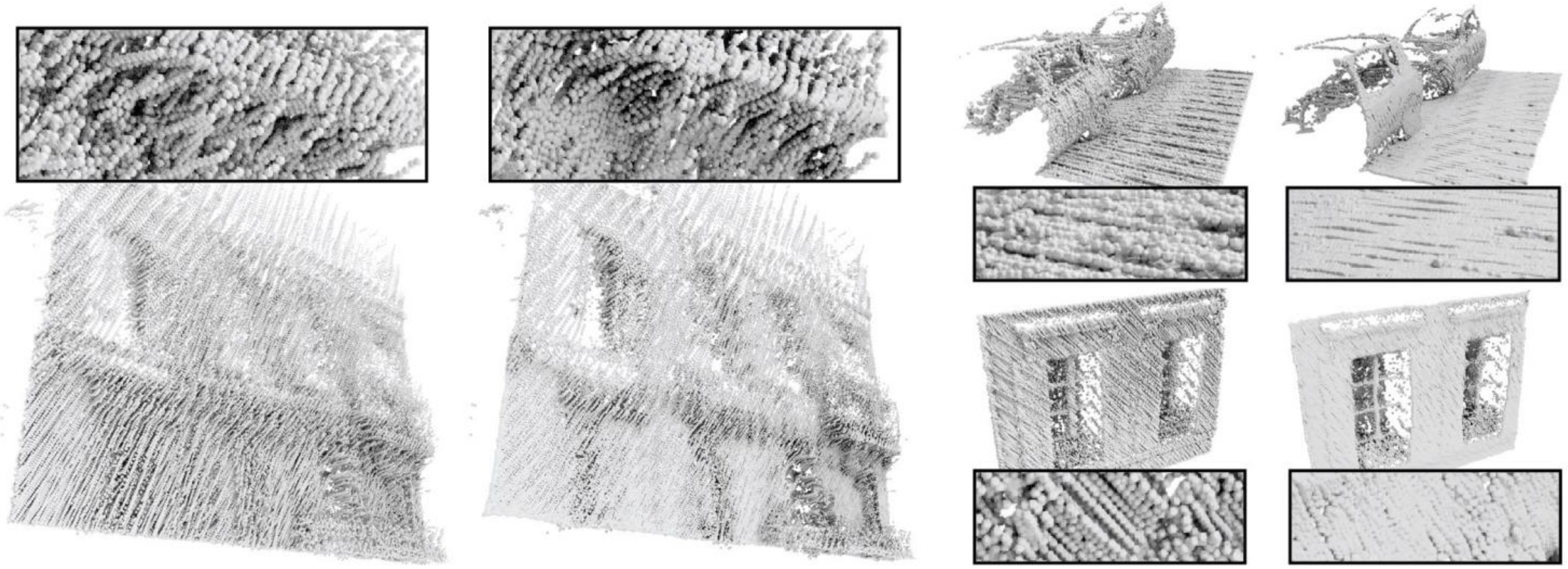
where σ is the bandwidth of k and W = diag(w) is a diagonal weight matrix trading spatial and appearance locality. Optimizing

$$\arg\min_{\Theta} E_{y\sim p(z|S)} E_{q\sim q(z|y)} L(f_{\Theta}(y), q)$$

where L can be chosen to be a standard Chamfer distance and f is implemented using Monte Carlo Convolution

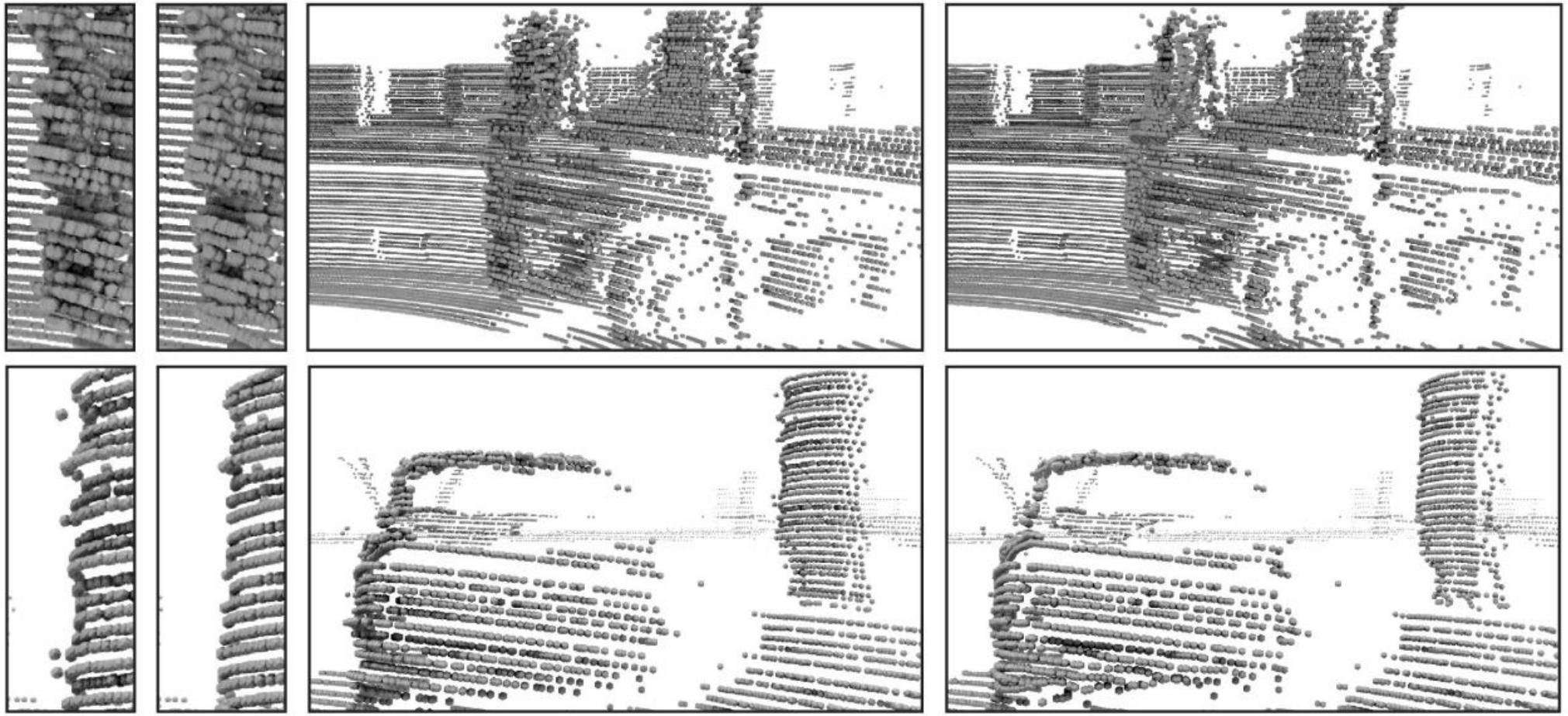
Hermosilla et al. **Total Denoising**, ICCV, April 2019

Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning



Hermosilla et al. **Total Denoising**, ICCV, April 2019

Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning



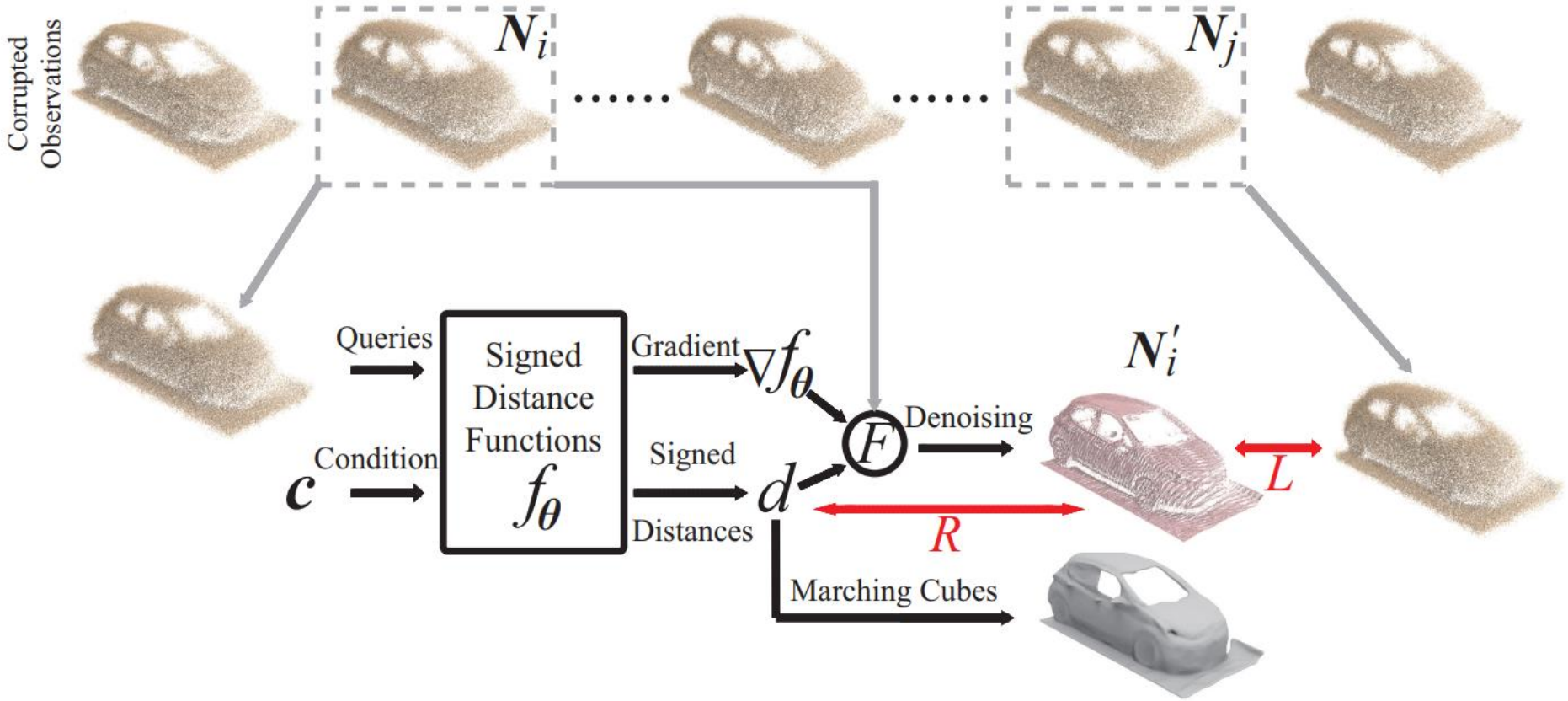
Hermosilla et al. **Total Denoising**, ICCV, April 2019

Fast Learning of Signed Distance Functions from Noisy Point Clouds via Noise to Noise Mapping

- Doesn't require any clean point cloud or ground truth supervision
- Can infer highly accurate distance field from multiple or even a single noisy observation
- Further, introduce a novel schema to improve multi-view reconstruction by estimating SDFs as a prior

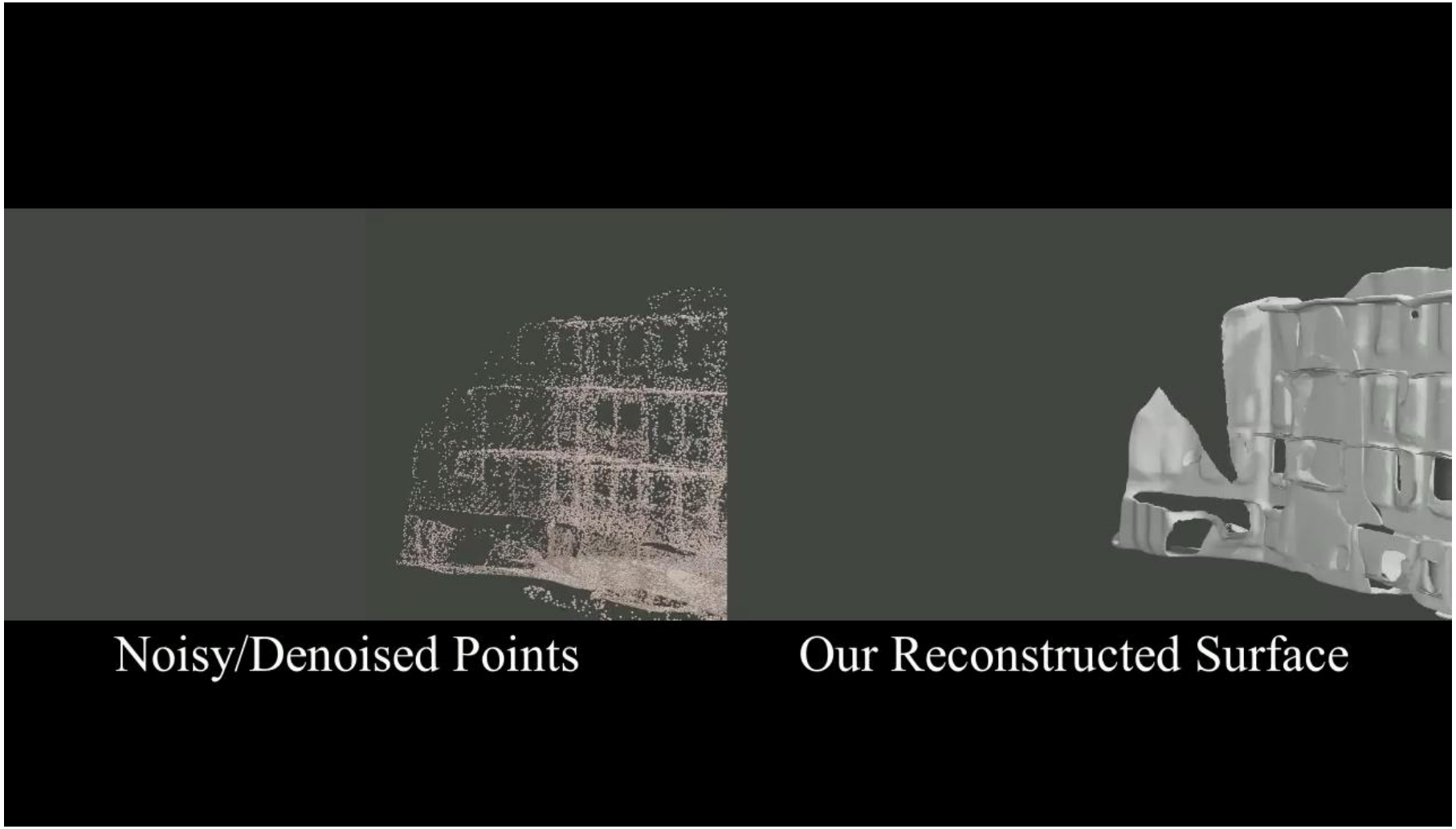
Zhou et al. **Noise2NoiseMapping,** ICML 2023, Oral

Fast Learning of Signed Distance Functions from Noisy Point Clouds via Noise to Noise Mapping



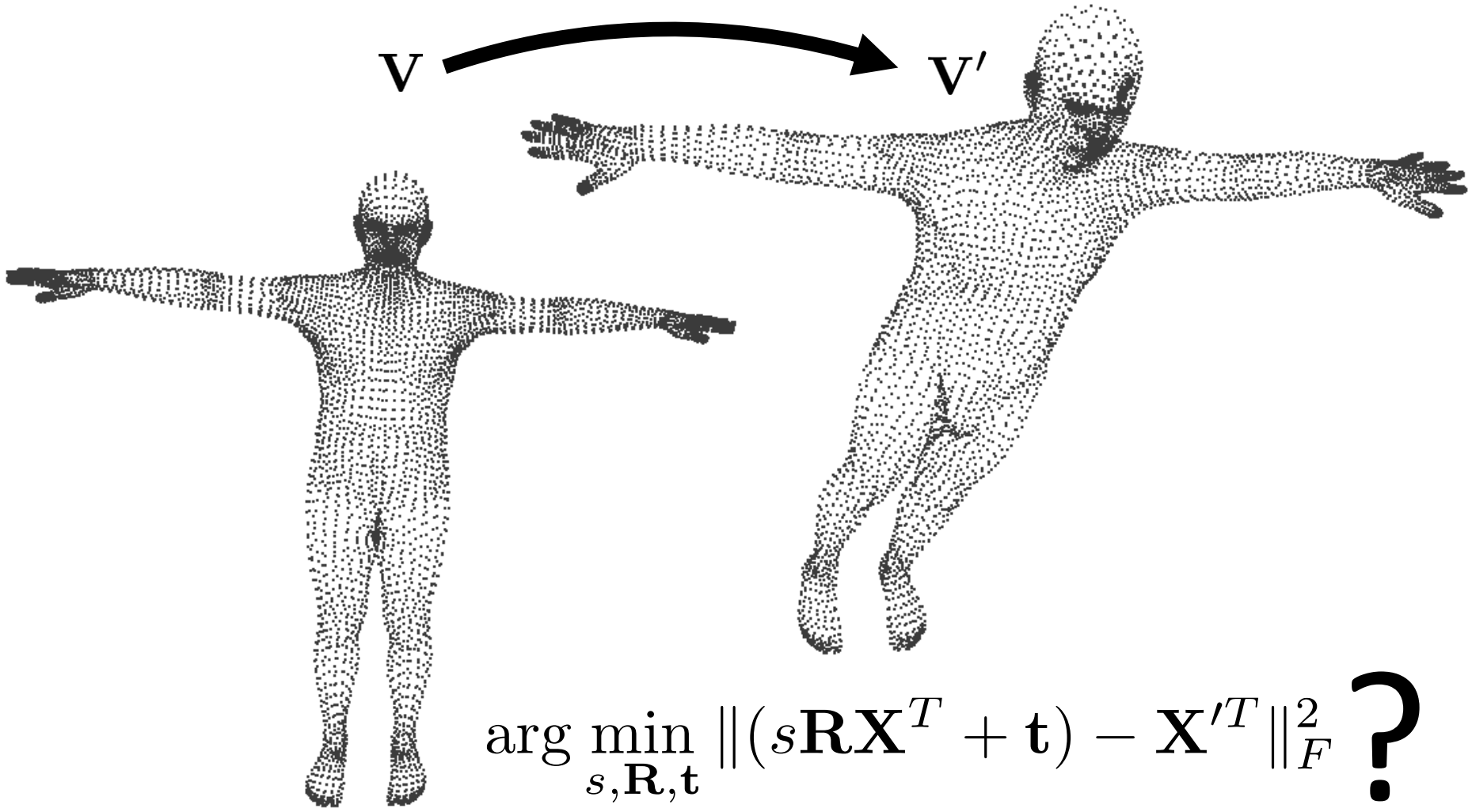
Zhou et al. **Noise2NoiseMapping,** ICML 2023, Oral

Fast Learning of Signed Distance Functions



Zhou et al. **Noise2NoiseMapping,** ICML 2023, Oral

Procrustes alignment problem definition



$$\arg\min_{s,\mathbf{R},\mathbf{t}} \|(s\mathbf{R}\mathbf{X}^T + \mathbf{t}) - \mathbf{X}'^T\|_F^2 \; ?$$

Rigid Alignment

How can we rigidly transform a set of points ?

- Translate it
- Rotate it
- Today we'll consider also
- Scale it

$$\mathbf{X}' = \mathbf{X} + \mathbf{t}, \mathbf{t} \in \mathbb{R}^3$$

$$\mathbf{X}'^T = \mathbf{R} \cdot \mathbf{X}^T, \mathbf{R} \in \mathbf{SO}(3)$$

$$\mathbf{X}' = s\mathbf{X}, s \in \mathbb{R}$$

Optimisation Problem

$$s, \mathbf{R}, \mathbf{t} = \arg\min_{s,\mathbf{R},\mathbf{t}} \sum_i \| s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i \|^2$$

Quick recap of SVD

in general, applied to a real matrix:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}}$$

$$\mathbf{A} \equiv (M \times N) \text{ real}$$

$$\mathbf{U} \equiv (M \times M) \text{ orthogonal, unit norm}$$

$$\mathbf{V} \equiv (N \times N) \text{ orthogonal, unit norm}$$

$$\Sigma \equiv (M \times N) \text{ diagonal}$$

warning: this is not the vertex matrix!

* See also clarification slide 1 at the end of slide deck

Procrustes alignment steps

$$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]^T \qquad \mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_n]^T$$

Matrices of points $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{(N \times 3)}$

$\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^{(3 \times 1)}$

$$\bar{\mathbf{Y}}^T \mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^T$$
$$\mathbf{R} = \mathbf{U}\mathbf{V}^T$$

Optimal **rotation** obtained by computing SVD on the point cross-covariance

$$\mathbf{t} = s\mathbf{R}\bar{\mathbf{x}} - \bar{\mathbf{y}}$$

Translation is the centroid difference

$$\mathbf{s} = \frac{\mathrm{tr}(\Sigma)}{\|\bar{\mathbf{X}}\|_F^2}$$

Scale is a quotient of eigenvalue sums

Proof

$$s, \mathbf{R}, \mathbf{t} = \arg\min_{s,\mathbf{R},\mathbf{t}} \sum_i \| s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i \|^2$$

Procrustes derivation: translation

$$\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{(N \times 3)}$$

$$\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^{(3 \times 1)}$$

$$s, \mathbf{R}, \mathbf{t} = \arg\min_{s,\mathbf{R},\mathbf{t}} E$$

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2$$

Minimize the L2 distance between transformed source points and target points

$$= \sum_i (s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i)^\mathsf{T}(s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i)$$

$$= \sum_i s^2\mathbf{x}_i^\mathsf{T}\mathbf{x}_i + \mathbf{t}^\mathsf{T}\mathbf{t} + \mathbf{y}_i^\mathsf{T}\mathbf{y}_i + 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{t} - 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{y}_i - 2\mathbf{t}^\mathsf{T}\mathbf{y}_i$$

Procrustes derivation: translation

We remove the elements that do not depend on the translation and solve for t.

$$\mathbf{t} = \arg\min_{\mathbf{t}} E = \arg\min_{\mathbf{t}} \sum_i \cancel{s^2 \mathbf{x}_i^\mathsf{T}\mathbf{x}_i} + \mathbf{t}^\mathsf{T}\mathbf{t} + \cancel{\mathbf{y}_i^\mathsf{T}\mathbf{y}_i} + 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{t} - \cancel{2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{y}_i} - 2\mathbf{t}^\mathsf{T}\mathbf{y}_i$$

$$= \arg\min_{\mathbf{t}} \sum_i \mathbf{t}^\mathsf{T}\mathbf{t} + 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{t} - 2\mathbf{t}^\mathsf{T}\mathbf{y}_i$$

$$\bar{\mathbf{x}} \equiv \frac{\sum_i \mathbf{x}_i}{N}, \bar{\mathbf{y}} \equiv \frac{\sum_i \mathbf{y}_i}{N}$$

Compute the centroid of the point clouds

$$\mathbf{t} = \arg\min_{\mathbf{t}} E = \arg\min_{\mathbf{t}} (\mathbf{t}^\mathsf{T}(2s\mathbf{R}\bar{\mathbf{x}} + \mathbf{t} - 2\bar{\mathbf{y}})) = \bar{\mathbf{y}} - s\mathbf{R}\bar{\mathbf{x}}$$

So given s and R, we can compute the translation t

Procrustes derivation: Rotation

Subtract the centroid from the points to obtain a simpler expression for E

$$\bar{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \bar{\mathbf{y}}_i \equiv \mathbf{y}_i - \bar{\mathbf{y}}$$

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 = \sum_i \|s\mathbf{R}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2$$

subject to $\mathbf{R} \in SO(3)$

Optimal rotation does not depend on scale

$$\mathbf{R} = \arg\min_{\mathbf{R}} ||\mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T||_F^2$$

Procrustes derivation: Rotation

Subtract the centroid from the points to obtain a simpler expression for E

$$\bar{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \bar{\mathbf{y}}_i \equiv \mathbf{y}_i - \bar{\mathbf{y}}$$

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 = \sum_i \|s\mathbf{R}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2$$

Optimal rotation does not depend on scale

$$\begin{aligned}
\mathbf{R} &= \arg\min_{\mathbf{R}} ||\mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T\|_F^2 \\
&= \arg\min_{\mathbf{R}} \langle \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T, \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\mathbf{R}\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle \mathbf{R}, \bar{\mathbf{Y}}^T\bar{\mathbf{X}} \rangle_F
\end{aligned}$$

Rotation does not change norm!

Inner product should be maximum!

Procrustes derivation: Rotation

Subtract the centroid from the points to obtain a simpler expression for E

$$\bar{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \bar{\mathbf{y}}_i \equiv \mathbf{y}_i - \bar{\mathbf{y}}$$

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 = \sum_i \|s\mathbf{R}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2$$

$$\begin{aligned}
\mathbf{R} &= \arg\min_{\mathbf{R}} ||\mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T||_F^2 \\
&= \arg\min_{\mathbf{R}} \langle \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T, \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\mathbf{R}\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle \mathbf{R}, \bar{\mathbf{Y}}^T\bar{\mathbf{X}} \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle \mathbf{R}, U\Sigma V^T \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle U^T\mathbf{R}V, \Sigma \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle S, \Sigma \rangle_F \quad \text{where } S = U^T\mathbf{R}V
\end{aligned}$$

Optimal rotation does not depend on scale

Rotation does not change norm!

Inner product should be maximum!

Cross-covariance matrix

SVD decomposition

Procrustes derivation: Rotation

$$= \arg\max_{\mathbf{R}} \langle S, \Sigma \rangle_F \quad \text{where } S = U^T \mathbf{R} V$$

What kind of matrix is S ?

What kind of matrix is $\sum$?

Procrustes derivation: Rotation

$$= \arg\max_{\mathbf{R}} \langle S, \Sigma \rangle_F \quad \text{where } S = U^T \mathbf{R} V$$

What kind of matrix is S ? Orthogonal

What kind of matrix is $\sum$? Diagonal

Hence the quantity above is maximised when S equals the identity, hence:

$$I = U^T \mathbf{R} V$$

$$\mathbf{R} = UV^T$$

SVD of cross-covariance of pointsclouds

$$\bar{\mathbf{Y}}^T \mathbf{X} = U \Sigma V^T$$

Procrustes derivation: scale

Optimize **scale** given the rotation

$$s = \arg\min_s E = \arg\min_s \Big(\sum_i s^2 \bar{\mathbf{x}}_i^{\mathsf{T}} \bar{\mathbf{x}}_i + \cancel{\bar{\mathbf{y}}_i^{\mathsf{T}} \bar{\mathbf{y}}_i} - 2s \bar{\mathbf{x}}_i^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \bar{\mathbf{y}}_i\Big)$$

$$= \arg\min_s \Big(s^2 \sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \bar{\mathbf{x}}_i) - 2s \sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \bar{\mathbf{y}}_i)\Big)$$

$$= \arg\min_s (s^2 a - 2sb) = \frac{b}{a} = \frac{\sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \bar{\mathbf{y}}_i))}{\sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \bar{\mathbf{x}}_i)}$$

$$= \frac{\mathrm{tr}(\bar{\mathbf{X}} \mathbf{R}^T \bar{\mathbf{Y}}^T)}{\|\bar{\mathbf{X}}\|_F^2} = \frac{\mathrm{tr}(\mathbf{R}^T \bar{\mathbf{Y}}^T \bar{\mathbf{X}})}{\|\bar{\mathbf{X}}\|_F^2} = \frac{\mathrm{tr}(\mathbf{V}\mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T)}{\|\bar{\mathbf{X}}\|_F^2} = \frac{\mathrm{tr}(\Sigma)}{\|\bar{\mathbf{X}}\|_F^2}$$

We used:

1) Trace invariance with shifts
2) $\bar{\mathbf{Y}}^T \bar{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{R} = \mathbf{U}\mathbf{V}^T$
3) Trace equals sum of eigenvals for square matrices

Slide credits and further reading

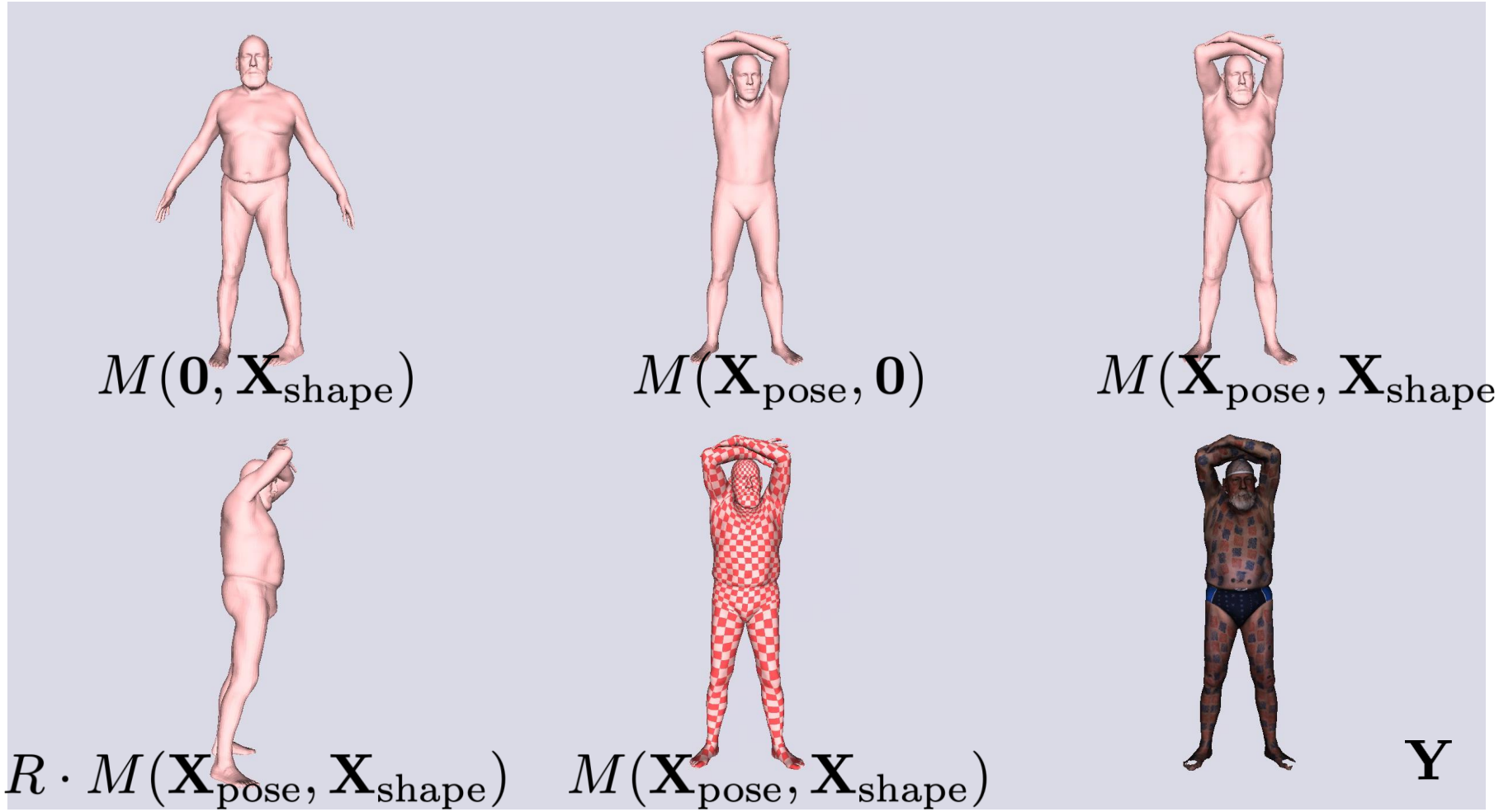
Hands-On AI Based 3D Vision Winter 24/25

Lecture 5_2 – Procrustes Alignment

Prof. Dr.-Ing. Gerard Pons-Moll
University of Tübingen / MPI-Informatics

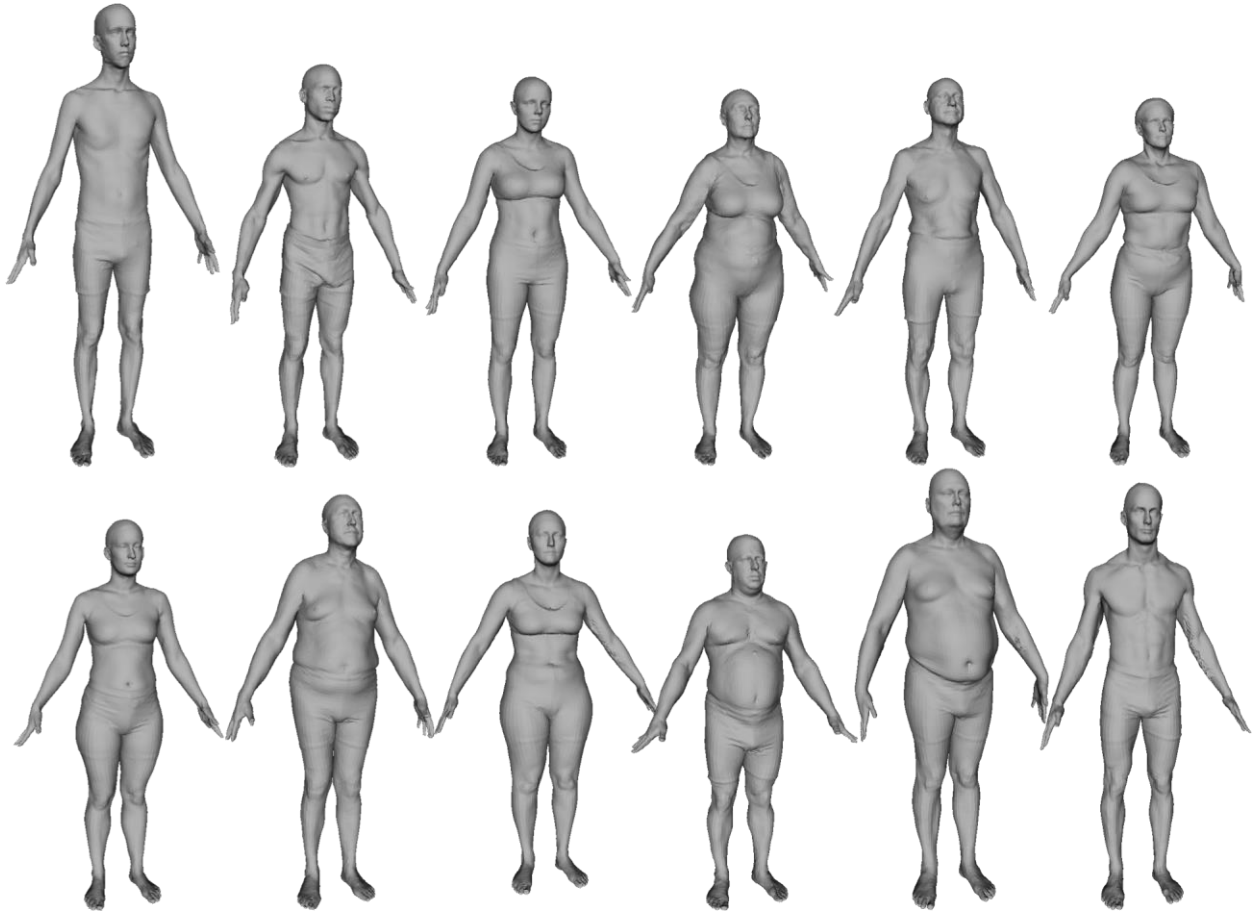
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Goal: learn a model of pose and shape



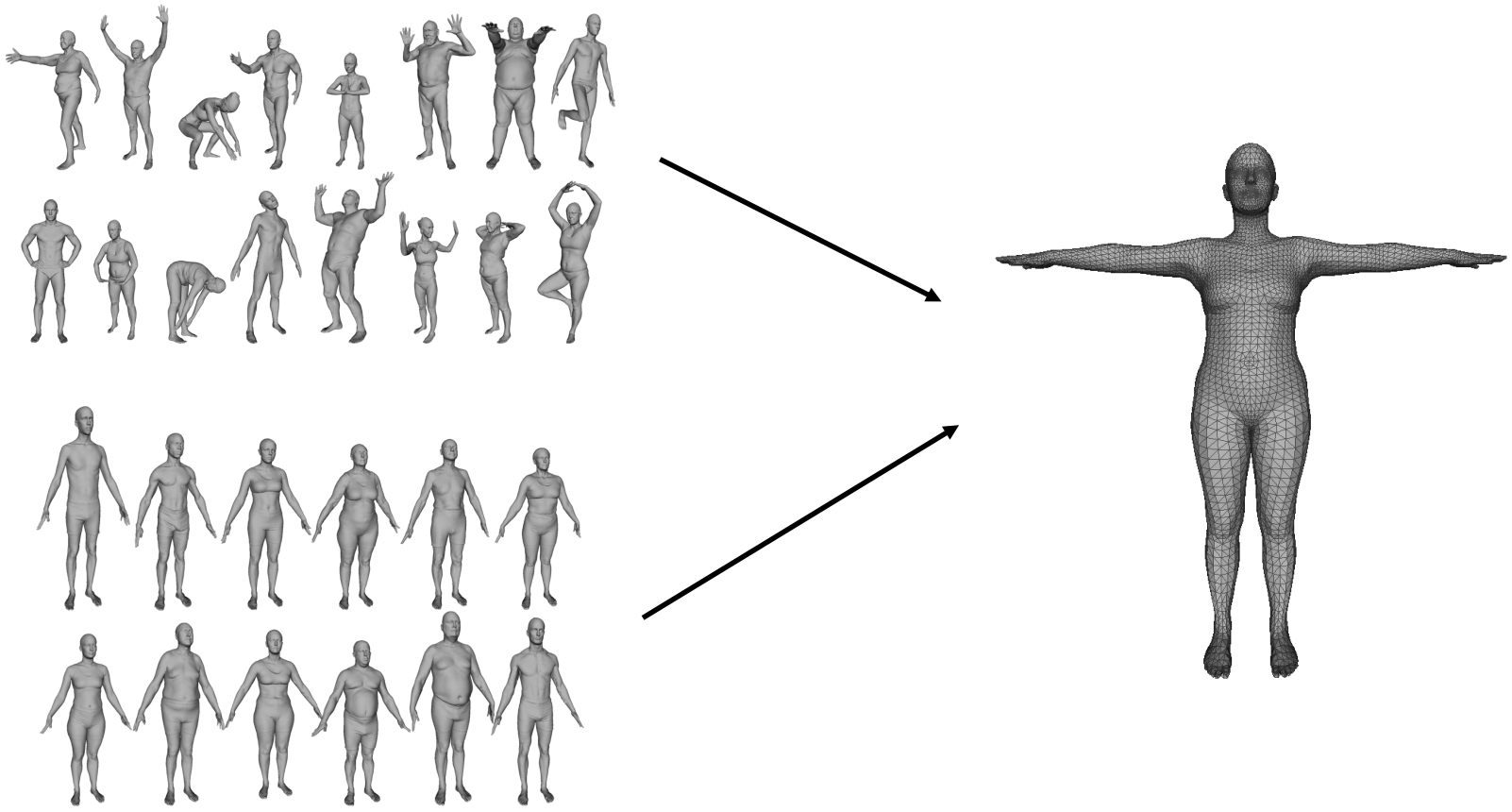
$$\mathbf{X} = \{\mathbf{X}_{\text{pose}}, \mathbf{X}_{\text{shape}}\}$$

Scan a Lot of People

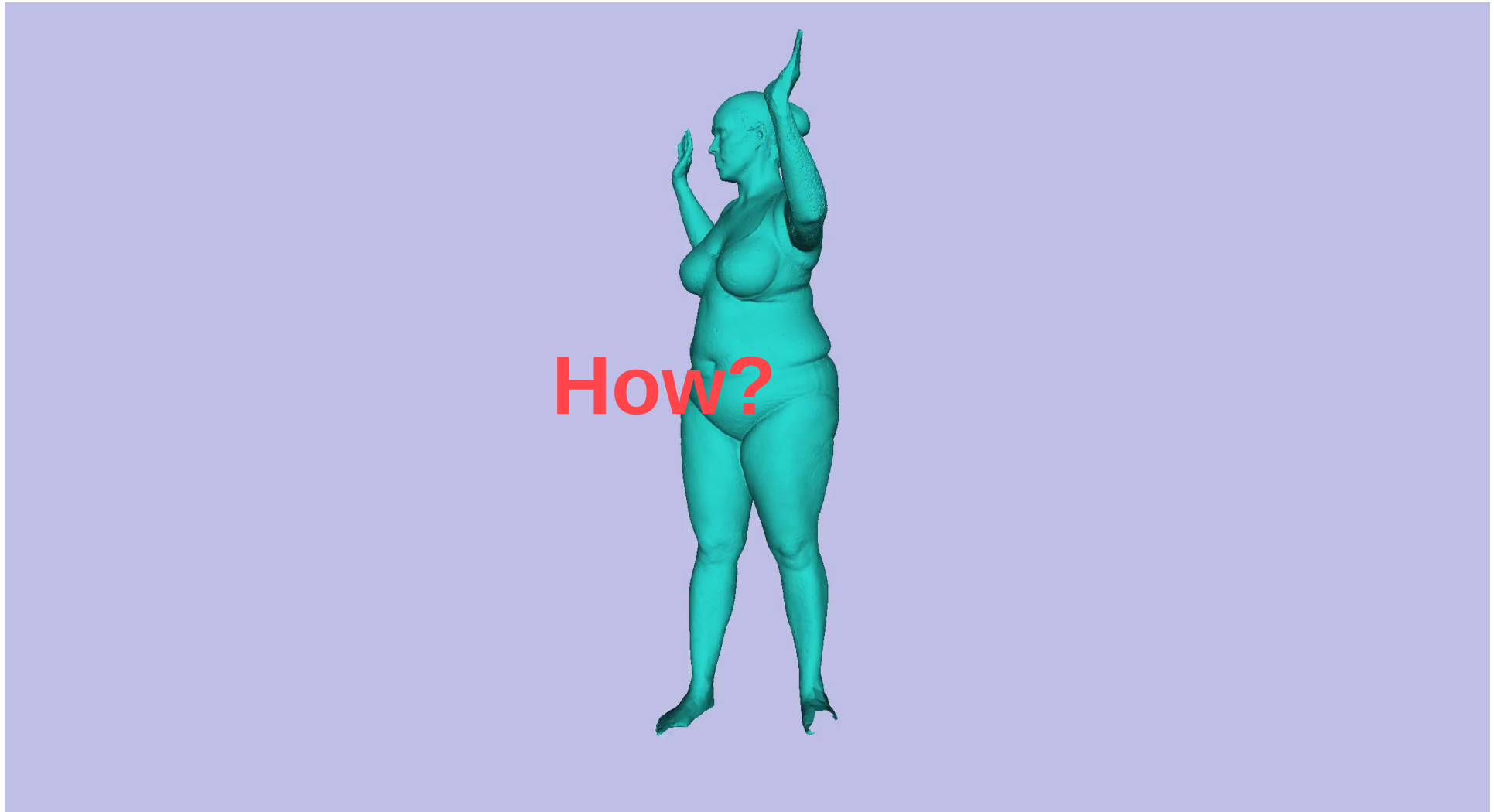


In lots of poses

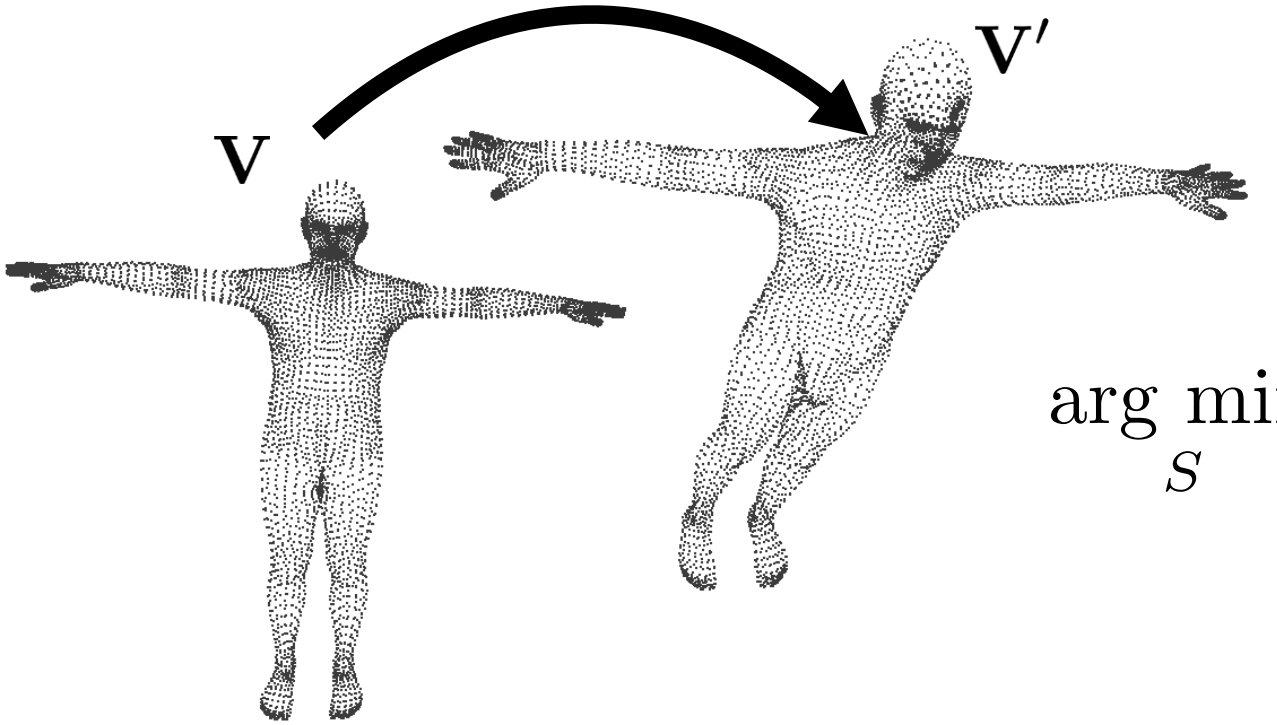
To learn a model we need correspondence



Non-rigid Articulated Registration



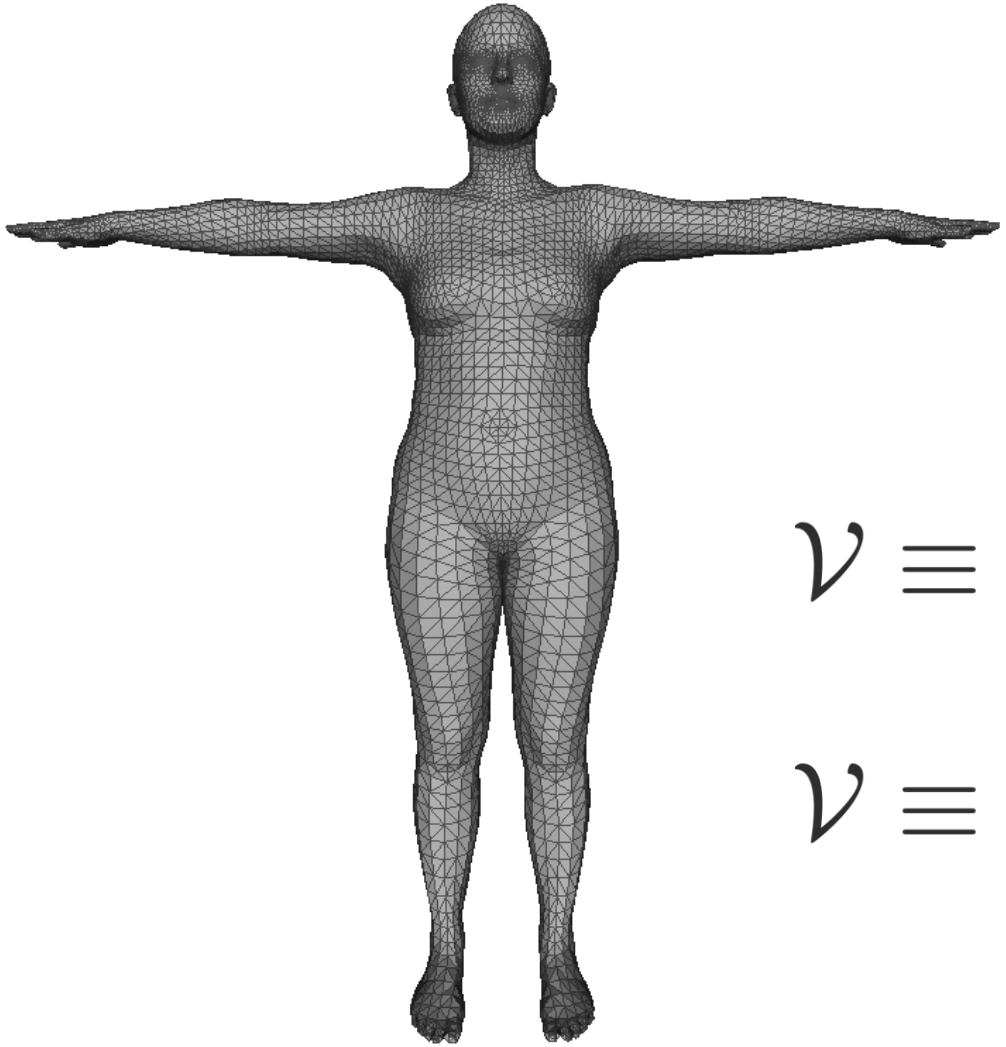
Today: Rigid Alignment (Procrustes)



$$\arg\min_{S} ||(s\mathbf{R} \cdot \mathbf{X}^T + \mathbf{t}) - \mathbf{X}'^T||$$

?

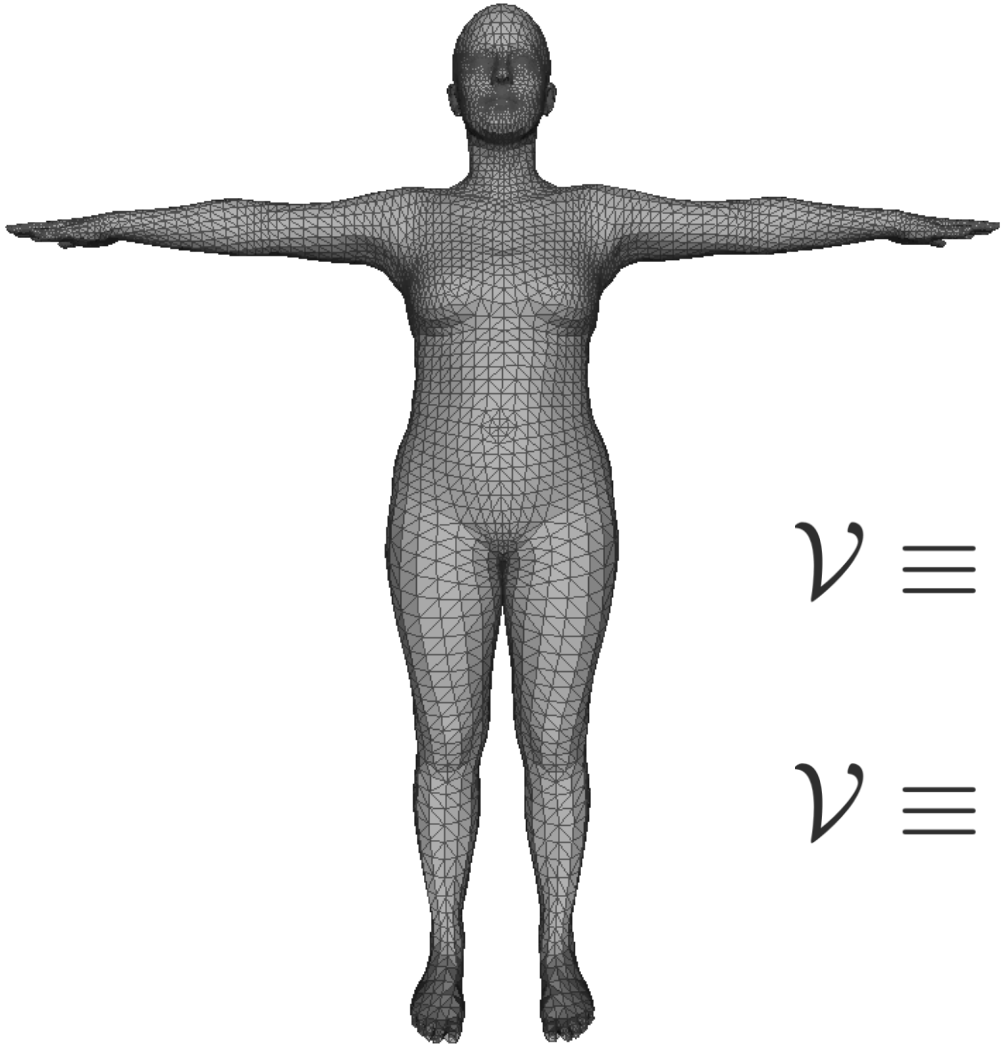
Surface representation: Mesh



$$\mathcal{V} \equiv \begin{cases} \mathbf{F} \in \mathbb{N}^{M \times 3} \\ \mathbf{X} \in \mathbb{R}^{N \times 3} \end{cases}$$

$$\mathcal{V} \equiv \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

Surface representation: Mesh



$$\mathcal{V} \equiv \begin{cases} \mathbf{F} \in \mathbb{N}^{M \times 3} \\ \mathbf{X} \in \mathbb{R}^{N \times 3} \end{cases}$$

$$\mathcal{V} \equiv \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

Surface representation: Mesh

How can we rigidly transform a set of points ?

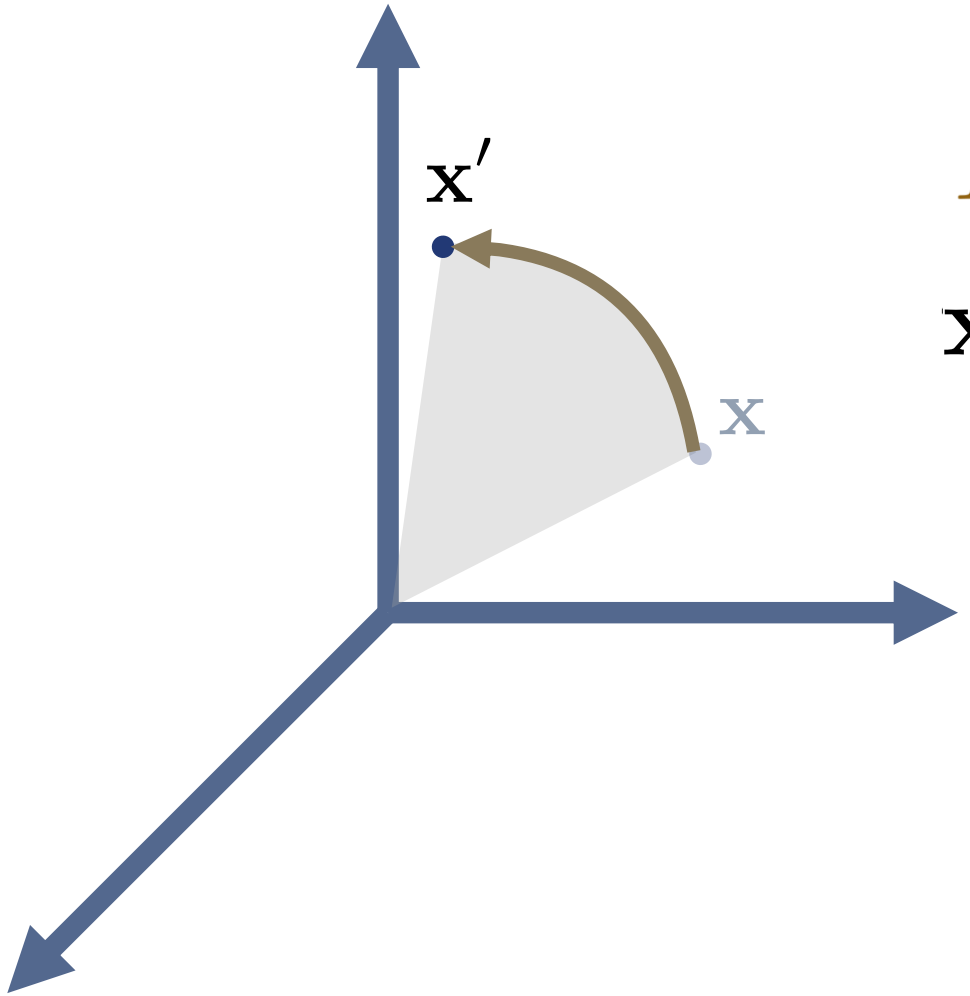
- Translate it
- Rotate it
- Today we'll consider also
- Scale it

$$\mathbf{X}' = \mathbf{X} + \mathbf{t}, \mathbf{t} \in \mathbb{R}^3$$

$$\mathbf{X}'^T = \mathbf{R} \cdot \mathbf{X}^T, \mathbf{R} \in \mathbf{SO}(3)$$

$$\mathbf{X}' = s\mathbf{X}, s \in \mathbb{R}$$

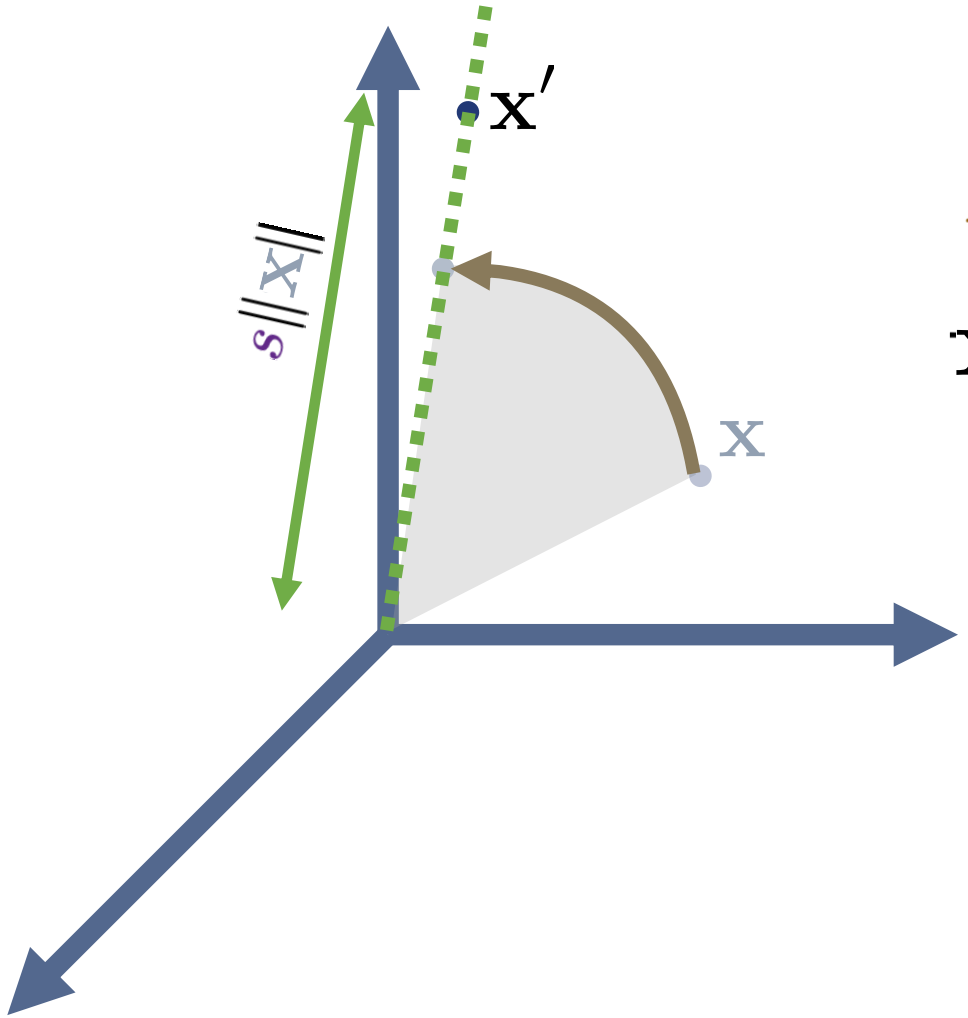
Rigid transformation + scale



$$R : \mathbb{R}^3 \to \mathbb{R}^3$$

$$\mathbf{x}' = R\mathbf{x}$$

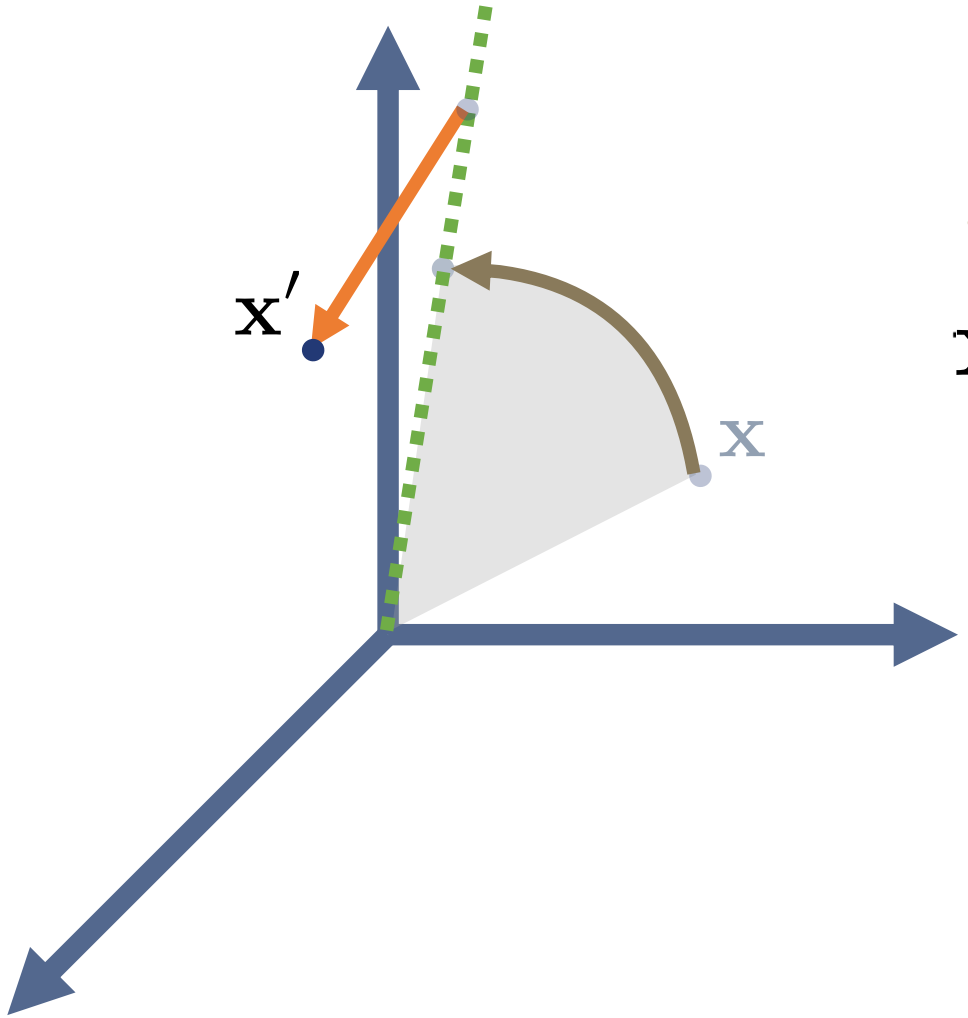
Rigid transformation + scale



$$R : \mathbb{R}^3 \to \mathbb{R}^3$$

$$\mathbf{x}' = sR\mathbf{x}$$

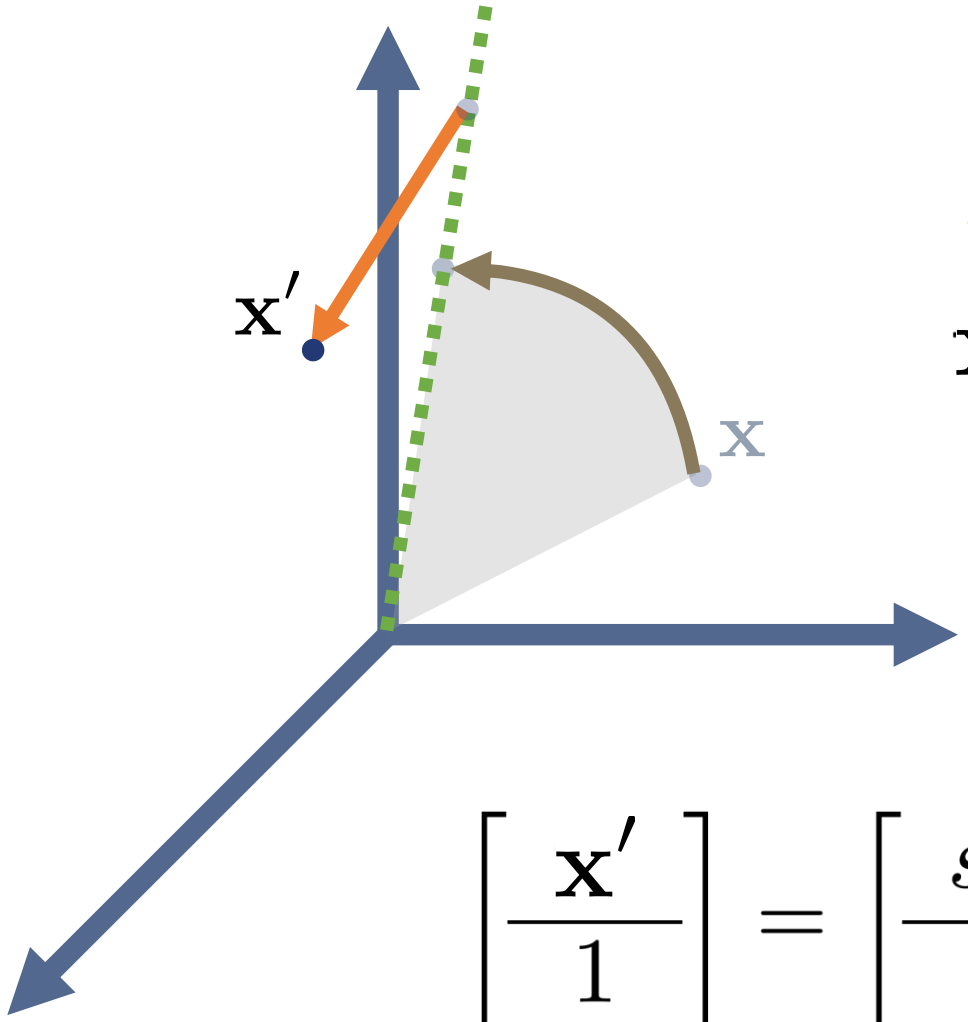
Rigid transformation + scale



$$R : \mathbb{R}^3 \to \mathbb{R}^3$$

$$\mathbf{x}' = sR\mathbf{x} + \mathbf{t}$$

Rigid transformation + scale



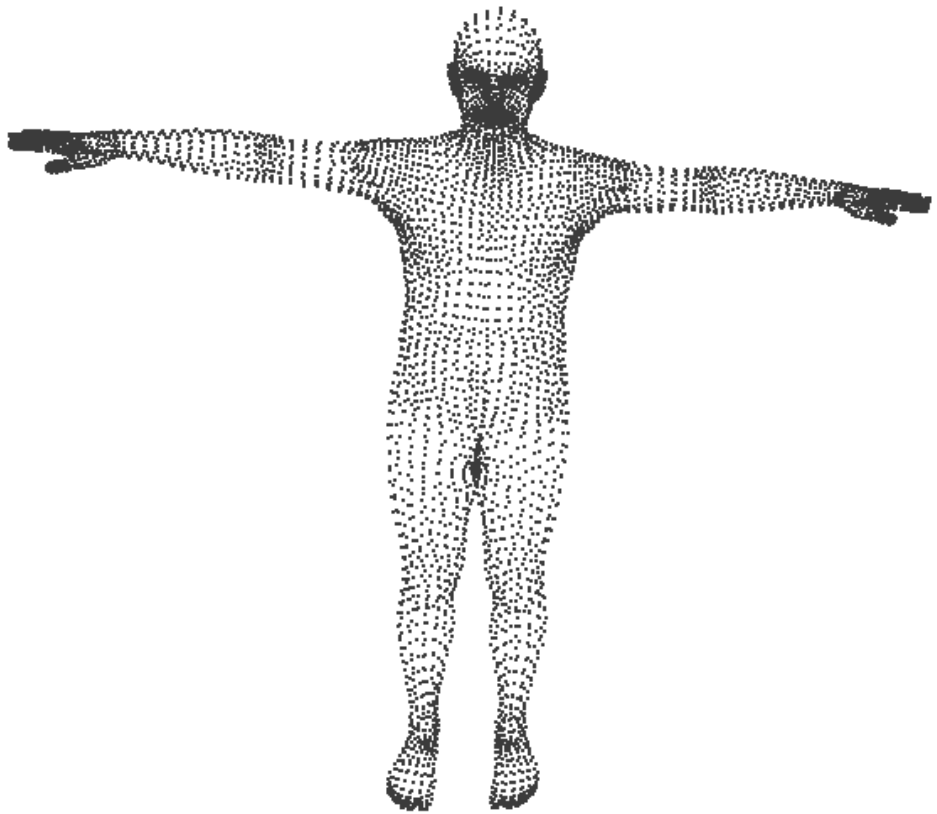
$$R : \mathbb{R}^3 \to \mathbb{R}^3$$

$$\mathbf{x}' = sR\mathbf{x} + \mathbf{t}$$

$$\begin{bmatrix} \mathbf{x}' \\ \hline 1 \end{bmatrix} = \left[\begin{array}{c|c} s\mathbf{R} & \mathbf{t} \\ \hline 0 & 1 \end{array}\right] \begin{bmatrix} \mathbf{x} \\ \hline 1 \end{bmatrix}$$

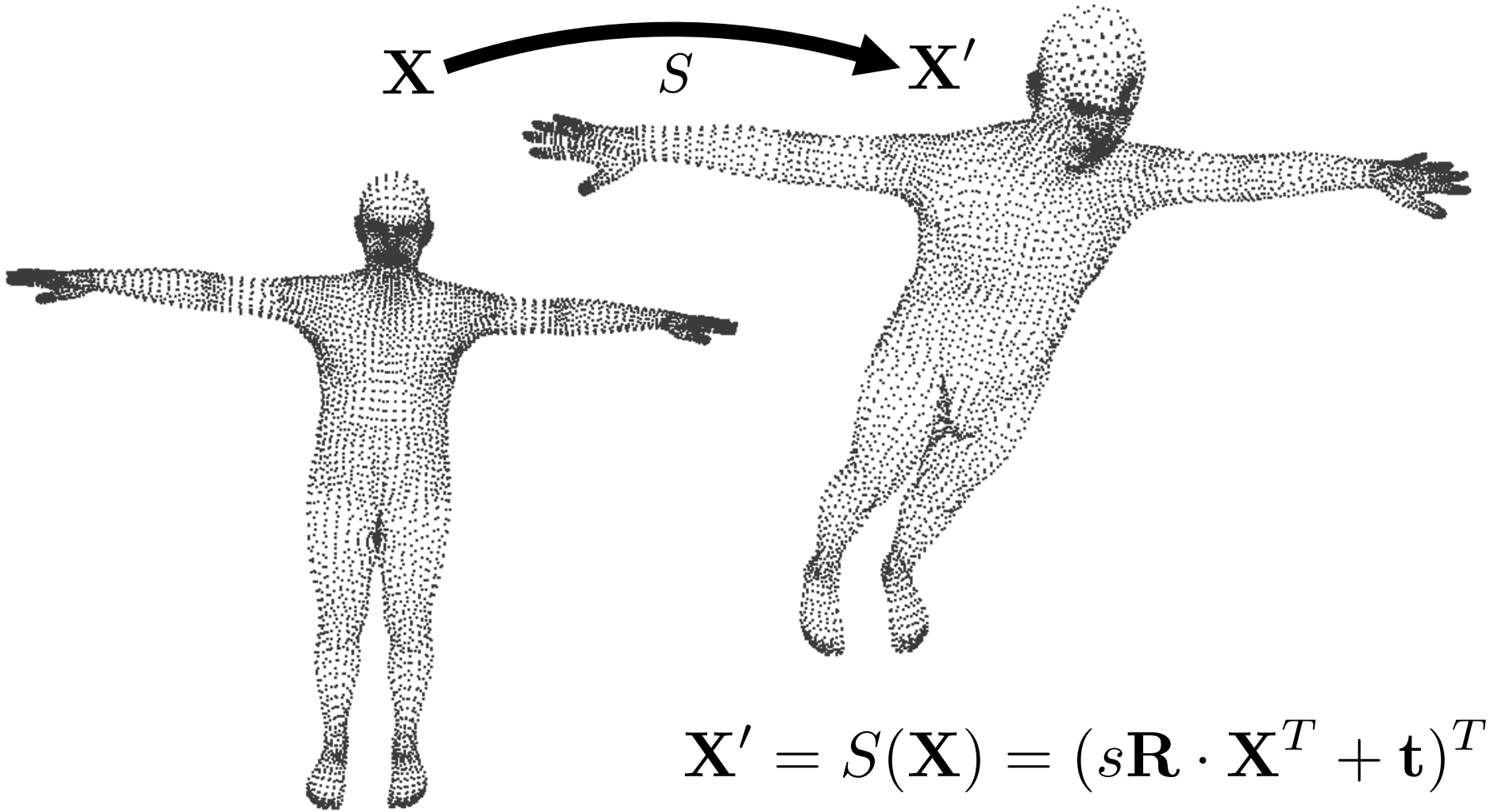
Procrustes alignment problem definition

$$\mathbf{X} \xrightarrow{\quad S \quad} \mathbf{X}'$$



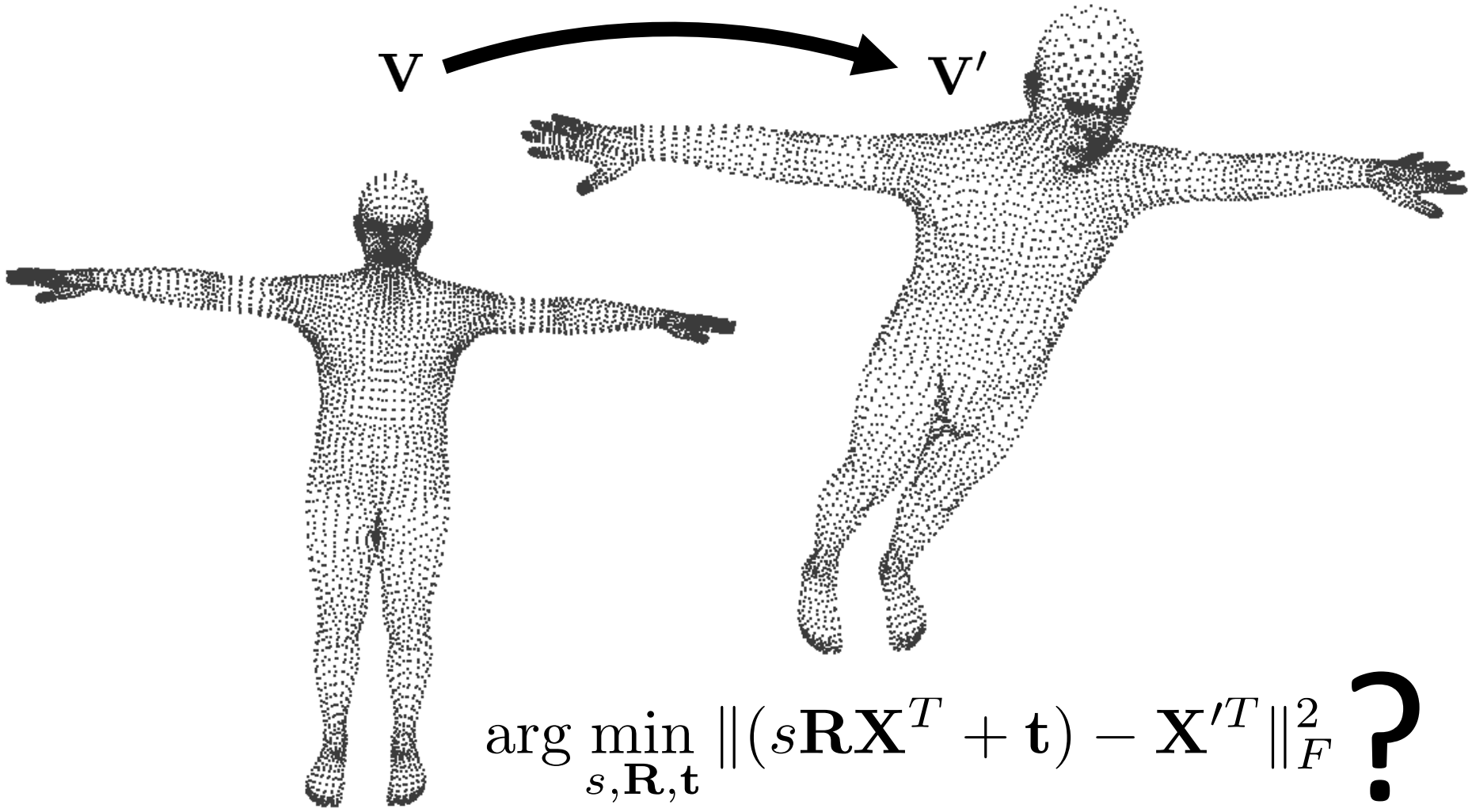
?

Procrustes alignment problem definition



$$\mathbf{X}' = S(\mathbf{X}) = (s\mathbf{R} \cdot \mathbf{X}^T + \mathbf{t})^T$$

Procrustes alignment problem definition



$$\arg\min_{s,\mathbf{R},\mathbf{t}} \|(s\mathbf{R}\mathbf{X}^T + \mathbf{t}) - \mathbf{X}'^T\|_F^2 \; ?$$

Why Procrustes?

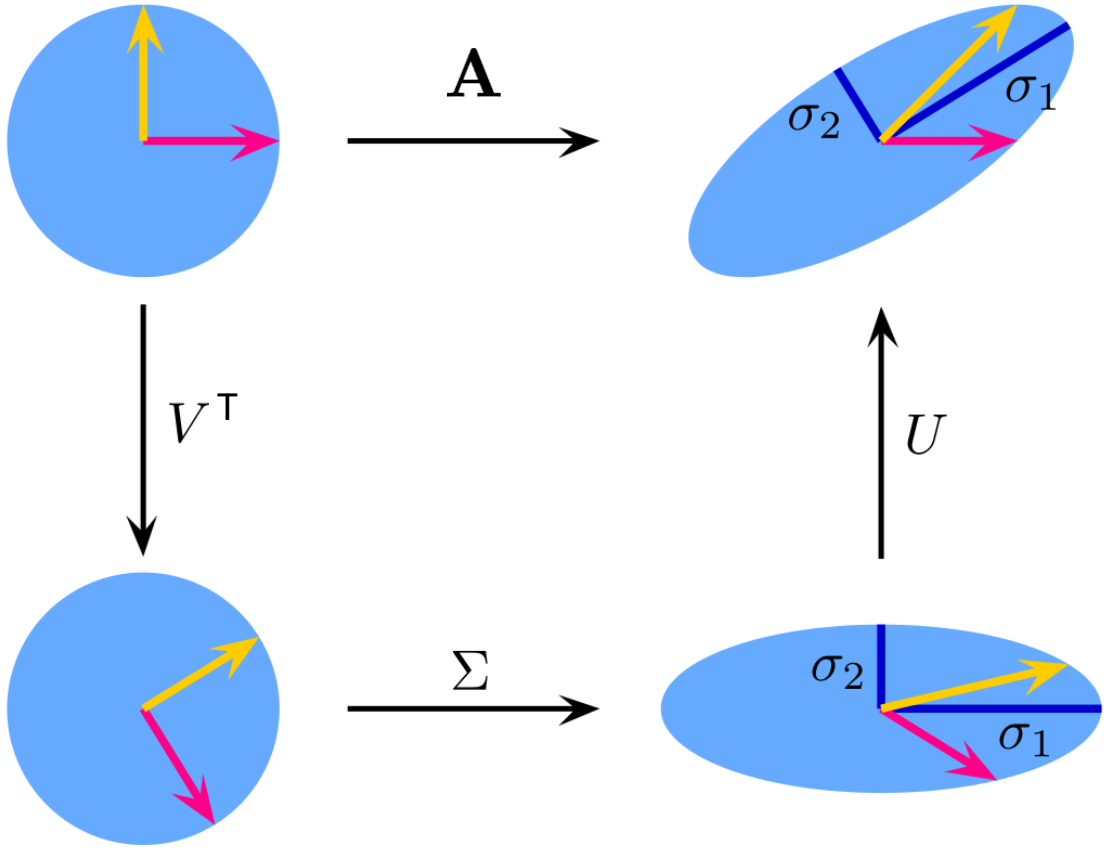
The name *Procrustes* (Greek: Προκρούστης) refers to a bandit from Greek mythology who made his victims fit his bed either by stretching their limbs or cutting them off.

Prcrustes means “he who stretches”

Optimisation Problem

$$s, \mathbf{R}, \mathbf{t} = \arg\min_{s,\mathbf{R},\mathbf{t}} \sum_i \| s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i \|^2$$

Quick recap of SVD



$$A = U\Sigma V^T$$

Quick recap of SVD

in general, applied to a real matrix:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}}$$

$$\mathbf{A} \equiv (M \times N) \text{ real}$$

$$\mathbf{U} \equiv (M \times M) \text{ orthogonal, unit norm}$$

$$\mathbf{V} \equiv (N \times N) \text{ orthogonal, unit norm}$$

$$\Sigma \equiv (M \times N) \text{ diagonal}$$

warning: this is not the vertex matrix!

* See also clarification slide 1 at the end of slide deck

Procrustes alignment steps

$$\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_n]^T \qquad \mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_n]^T$$

Matrices of points $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{(N \times 3)}$

$\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^{(3 \times 1)}$

$$\bar{\mathbf{Y}}^T \mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^T$$
$$\mathbf{R} = \mathbf{U}\mathbf{V}^T$$

Optimal **rotation** obtained by computing SVD on the point cross-covariance

$$\mathbf{t} = s\mathbf{R}\bar{\mathbf{x}} - \bar{\mathbf{y}}$$

Translation is the centroid difference

$$\mathbf{s} = \frac{\mathrm{tr}(\Sigma)}{\|\bar{\mathbf{X}}\|_F^2}$$

Scale is a quotient of eigenvalue sums

Proof

$$s, \mathbf{R}, \mathbf{t} = \arg\min_{s,\mathbf{R},\mathbf{t}} \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2$$

Procrustes derivation: translation

$$\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{(N \times 3)}$$

$$\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^{(3 \times 1)}$$

$$s, \mathbf{R}, \mathbf{t} = \arg\min_{s,\mathbf{R},\mathbf{t}} E$$

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2$$

Minimize the L2 distance between transformed source points and target points

$$= \sum_i (s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i)^\mathsf{T}(s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i)$$

$$= \sum_i s^2\mathbf{x}_i^\mathsf{T}\mathbf{x}_i + \mathbf{t}^\mathsf{T}\mathbf{t} + \mathbf{y}_i^\mathsf{T}\mathbf{y}_i + 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{t} - 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{y}_i - 2\mathbf{t}^\mathsf{T}\mathbf{y}_i$$

Procrustes derivation: translation

We remove the elements that do not depend on the translation and solve for t.

$$\mathbf{t} = \arg\min_{\mathbf{t}} E = \arg\min_{\mathbf{t}} \sum_i \cancel{s^2 \mathbf{x}_i^\mathsf{T}\mathbf{x}_i} + \mathbf{t}^\mathsf{T}\mathbf{t} + \cancel{\mathbf{y}_i^\mathsf{T}\mathbf{y}_i} + 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{t} - \cancel{2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{y}_i} - 2\mathbf{t}^\mathsf{T}\mathbf{y}_i$$

$$= \arg\min_{\mathbf{t}} \sum_i \mathbf{t}^\mathsf{T}\mathbf{t} + 2s\mathbf{x}_i^\mathsf{T}\mathbf{R}^\mathsf{T}\mathbf{t} - 2\mathbf{t}^\mathsf{T}\mathbf{y}_i$$

$$\bar{\mathbf{x}} \equiv \frac{\sum_i \mathbf{x}_i}{N}, \bar{\mathbf{y}} \equiv \frac{\sum_i \mathbf{y}_i}{N}$$

Compute the centroid of the point clouds

$$\mathbf{t} = \arg\min_{\mathbf{t}} E = \arg\min_{\mathbf{t}} (\mathbf{t}^\mathsf{T}(2s\mathbf{R}\bar{\mathbf{x}} + \mathbf{t} - 2\bar{\mathbf{y}})) = \bar{\mathbf{y}} - s\mathbf{R}\bar{\mathbf{x}}$$

So given s and R, we can compute the translation t

Procrustes derivation: Rotation

Subtract the centroid from the points to obtain a simpler expression for E

$$\bar{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \bar{\mathbf{y}}_i \equiv \mathbf{y}_i - \bar{\mathbf{y}}$$

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 = \sum_i \|s\mathbf{R}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2$$

subject to $\mathbf{R} \in SO(3)$

Optimal rotation does not depend on scale

$$\mathbf{R} = \arg\min_{\mathbf{R}} ||\mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T||_F^2$$

Procrustes derivation: Rotation

Subtract the centroid from the points to obtain a simpler expression for E

$$\bar{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \bar{\mathbf{y}}_i \equiv \mathbf{y}_i - \bar{\mathbf{y}}$$

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 = \sum_i \|s\mathbf{R}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2$$

Optimal rotation does not depend on scale

$$\begin{aligned}
\mathbf{R} &= \arg\min_{\mathbf{R}} ||\mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T\|_F^2 \\
&= \arg\min_{\mathbf{R}} \langle \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T, \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\mathbf{R}\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle \mathbf{R}, \bar{\mathbf{Y}}^T\bar{\mathbf{X}} \rangle_F
\end{aligned}$$

Rotation does not change norm!

Inner product should be maximum!

Procrustes derivation: Rotation

Subtract the centroid from the points to obtain a simpler expression for E

$$\bar{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \bar{\mathbf{y}}_i \equiv \mathbf{y}_i - \bar{\mathbf{y}}$$

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 = \sum_i \|s\mathbf{R}\bar{\mathbf{x}}_i - \bar{\mathbf{y}}_i\|^2$$

$$\begin{aligned}
\mathbf{R} &= \arg\min_{\mathbf{R}} ||\mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T||_F^2 \\
&= \arg\min_{\mathbf{R}} \langle \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T, \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\mathbf{R}\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\min_{\mathbf{R}} \|\bar{\mathbf{X}}^T\|_F^2 + \|\bar{\mathbf{Y}}^T\|_F^2 - 2\langle \mathbf{R}\bar{\mathbf{X}}^T, \bar{\mathbf{Y}}^T \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle \mathbf{R}, \bar{\mathbf{Y}}^T\bar{\mathbf{X}} \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle \mathbf{R}, U\Sigma V^T \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle U^T\mathbf{R}V, \Sigma \rangle_F \\
&= \arg\max_{\mathbf{R}} \langle S, \Sigma \rangle_F \quad \text{where } S = U^T\mathbf{R}V
\end{aligned}$$

Optimal rotation does not depend on scale

Rotation does not change norm!

Inner product should be maximum!

Cross-covariance matrix

SVD decomposition

Procrustes derivation: Rotation

$$= \arg\max_{\mathbf{R}} \langle S, \Sigma \rangle_F \quad \text{where } S = U^T \mathbf{R} V$$

What kind of matrix is S ?

What kind of matrix is $\sum$?

Procrustes derivation: Rotation

$$= \arg\max_{\mathbf{R}} \langle S, \Sigma \rangle_F \quad \text{where } S = U^T \mathbf{R} V$$

What kind of matrix is S ? Orthogonal

What kind of matrix is $\sum$? Diagonal

Hence the quantity above is maximised when S equals the identity, hence:

$$I = U^T \mathbf{R} V$$

$$\mathbf{R} = UV^T$$

SVD of cross-covariance of pointsclouds

$$\bar{\mathbf{Y}}^T \mathbf{X} = U \Sigma V^T$$

Procrustes derivation: scale

Optimize **scale** given the rotation

$$s = \arg\min_s E = \arg\min_s \Big(\sum_i s^2 \bar{\mathbf{x}}_i^{\mathsf{T}} \bar{\mathbf{x}}_i + \cancel{\bar{\mathbf{y}}_i^{\mathsf{T}} \bar{\mathbf{y}}_i} - 2s \bar{\mathbf{x}}_i^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \bar{\mathbf{y}}_i\Big)$$

$$= \arg\min_s \Big(s^2 \sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \bar{\mathbf{x}}_i) - 2s \sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \bar{\mathbf{y}}_i)\Big)$$

$$= \arg\min_s (s^2 a - 2sb) = \frac{b}{a} = \frac{\sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \bar{\mathbf{y}}_i))}{\sum_i (\bar{\mathbf{x}}_i^{\mathsf{T}} \bar{\mathbf{x}}_i)}$$

$$= \frac{\mathrm{tr}(\bar{\mathbf{X}} \mathbf{R}^T \bar{\mathbf{Y}}^T)}{\|\bar{\mathbf{X}}\|_F^2} = \frac{\mathrm{tr}(\mathbf{R}^T \bar{\mathbf{Y}}^T \bar{\mathbf{X}})}{\|\bar{\mathbf{X}}\|_F^2} = \frac{\mathrm{tr}(\mathbf{V}\mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T)}{\|\bar{\mathbf{X}}\|_F^2} = \frac{\mathrm{tr}(\Sigma)}{\|\bar{\mathbf{X}}\|_F^2}$$

We used:

1) Trace invariance with shifts
2) $\bar{\mathbf{Y}}^T \bar{\mathbf{X}} = \mathbf{U}\Sigma\mathbf{V}^T$, $\mathbf{R} = \mathbf{U}\mathbf{V}^T$
3) Trace equals sum of eigenvals for square matrices

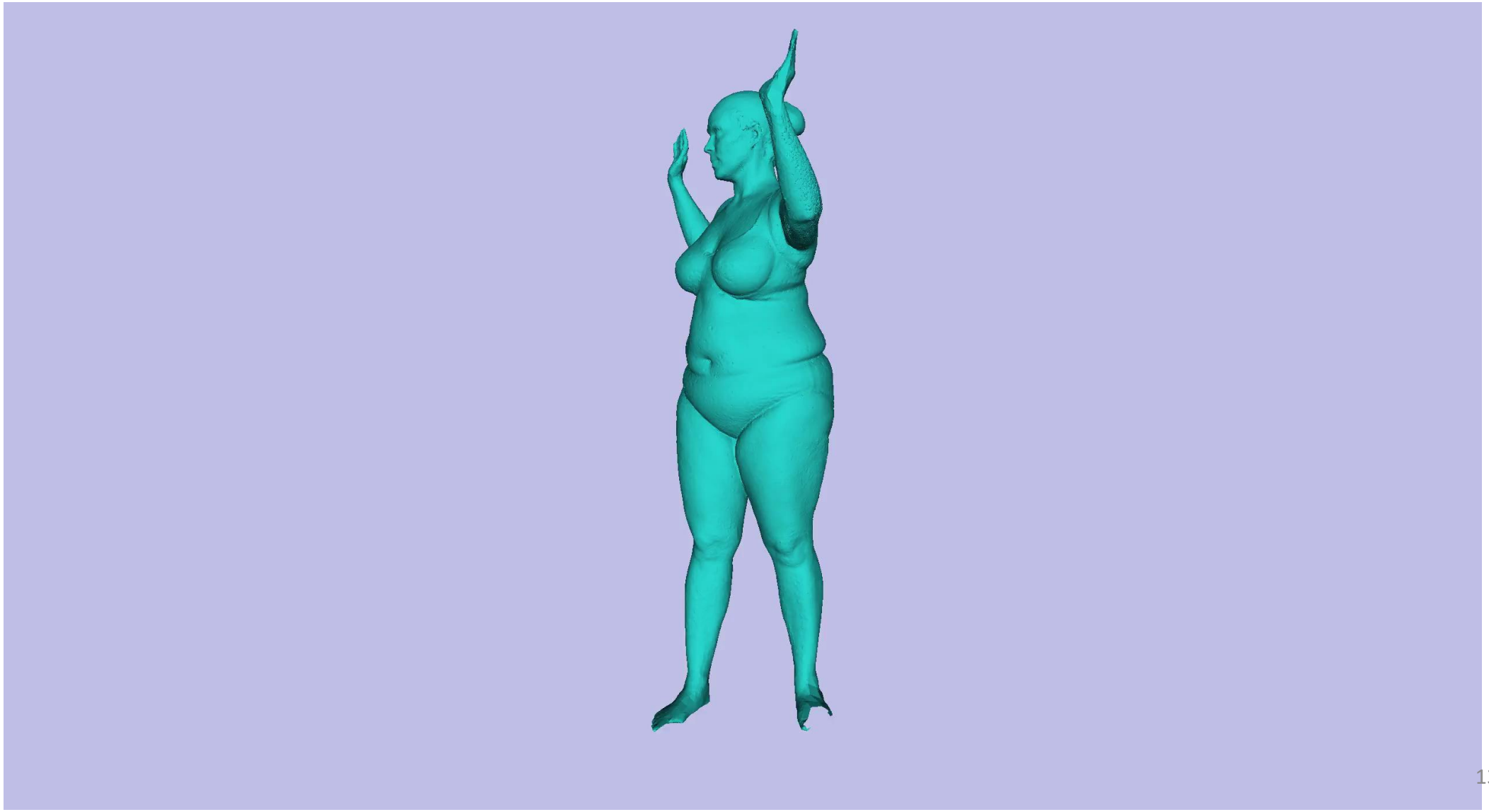
Virtual Humans – Winter 24/25

Lecture 4_1 – ICP: Iterative Closest Points

Prof. Dr.-Ing. Gerard Pons-Moll
University of Tübingen / MPI-Informatics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Non-rigid Articulated Registration



What is missing?

Given correspondences, we can find the optimal rigid alignment with Procrustes.

PROBLEMS:

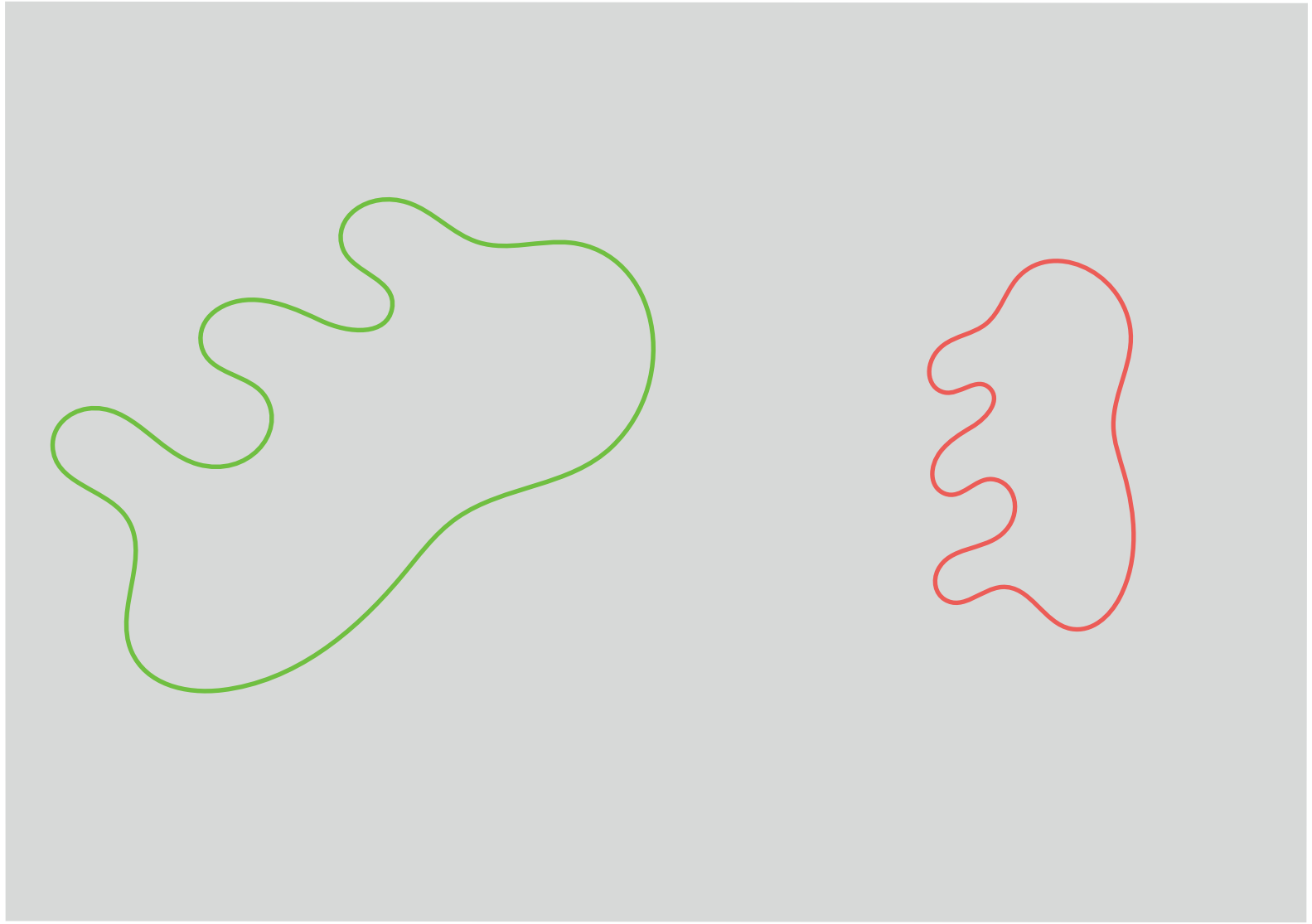
- How do we find the **correspondences** between shapes ?
- How do we **align** shapes **non-rigidly** ?

ICP and alignment based on optimisation

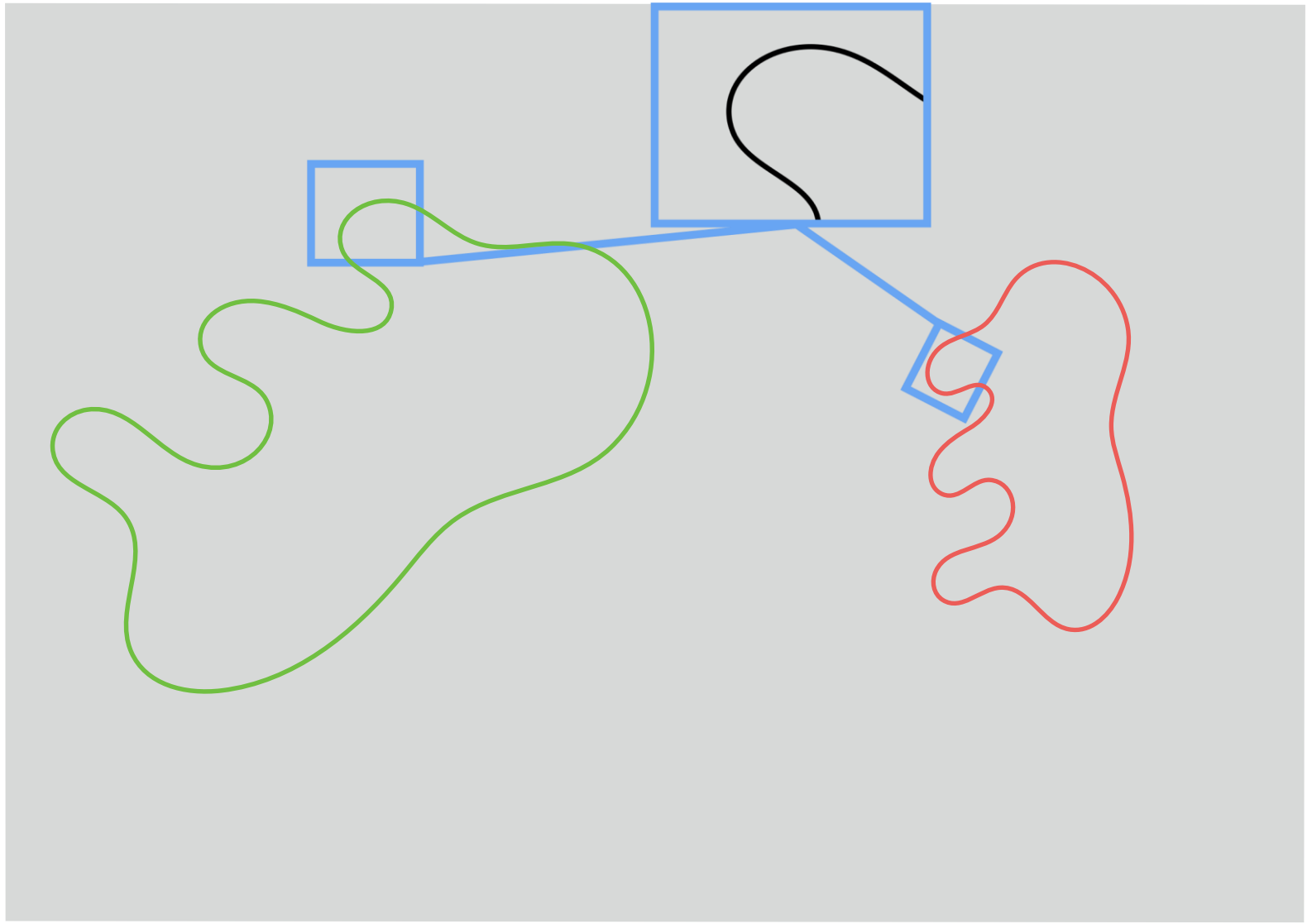
- Optimising alignment and correspondences using *Iterative Closest Point (ICP).*

- Alignment through *continuous* optimisation.

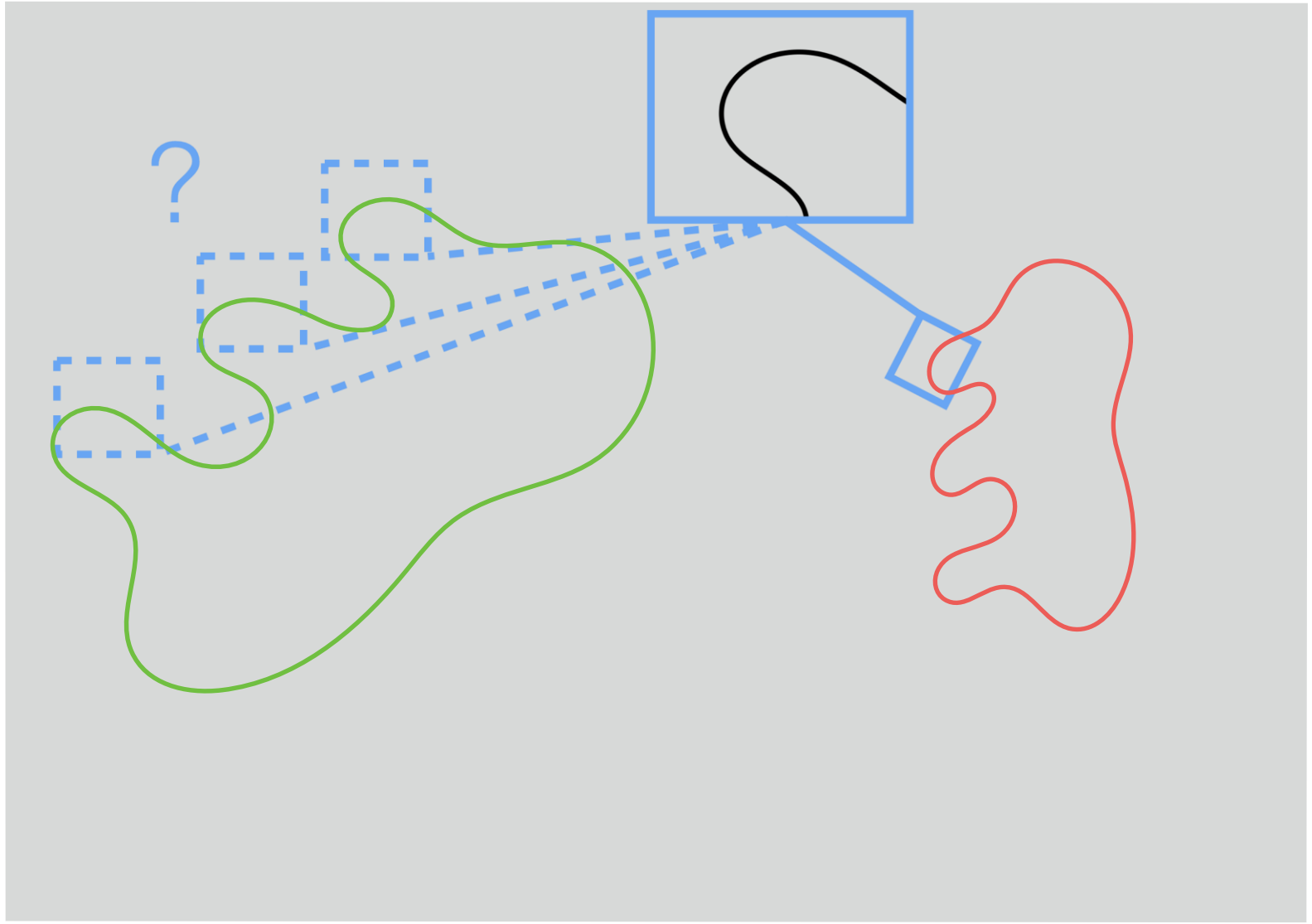
How do we find correspondences?



How do we find correspondences?



How do we find correspondences?



How do we find **correspondences**?

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 \equiv \sum_i \|f(\mathbf{x}_i) - \mathbf{y}_i\|^2$$

compact notation: f contains translation, rotation and isotropic scale

$\mathbf{x}_i$ Closest point to target shape point $\mathbf{y}_i$

The optimisation is over:

- the transform f
- the correspondences $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_i^N$

$$E(\mathcal{C}, f) = \sum_i \min_{\mathbf{x} \in \mathbf{X}} \|f(\mathbf{x}) - \mathbf{y}_i\|^2$$

How do we find correspondences?

The idea was to minimise the sum of distances between the one set of points and the other set, transformed

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 \equiv \sum_i \|f(\mathbf{x}_i) - \mathbf{y}_i\|^2$$

compact notation: f contains translation, rotation and isotropic scale

Ideas

The idea was to minimise the sum of distances between the one set of points and the other set, transformed

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 \equiv \sum_i \|f(\mathbf{x}_i) - \mathbf{y}_i\|^2$$

compact notation: f contains translation, rotation and isotropic scale

What if we estimate the correspondences?

Solution: Iteratively find correspondences

The idea was to minimise the sum of distances between the one set of points and the other set, transformed

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 \equiv \sum_i \|f(\mathbf{x}_i) - \mathbf{y}_i\|^2$$

compact notation: f contains translation, rotation and isotropic scale

What if we estimate the correspondences?

iteration

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

original unsorted points

$$f^{j+1} = \arg\min_f \sum_i \|f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i\|^2$$

Alternate between finding correspondences and finding the optimal transformation

The idea was to minimise the sum of distances between the one set of points and the other set, transformed

$$E \equiv \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2 \equiv \sum_i \|f(\mathbf{x}_i) - \mathbf{y}_i\|^2$$

compact notation: f contains translation, rotation and isotropic scale

What if we estimate the correspondences?

iteration

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

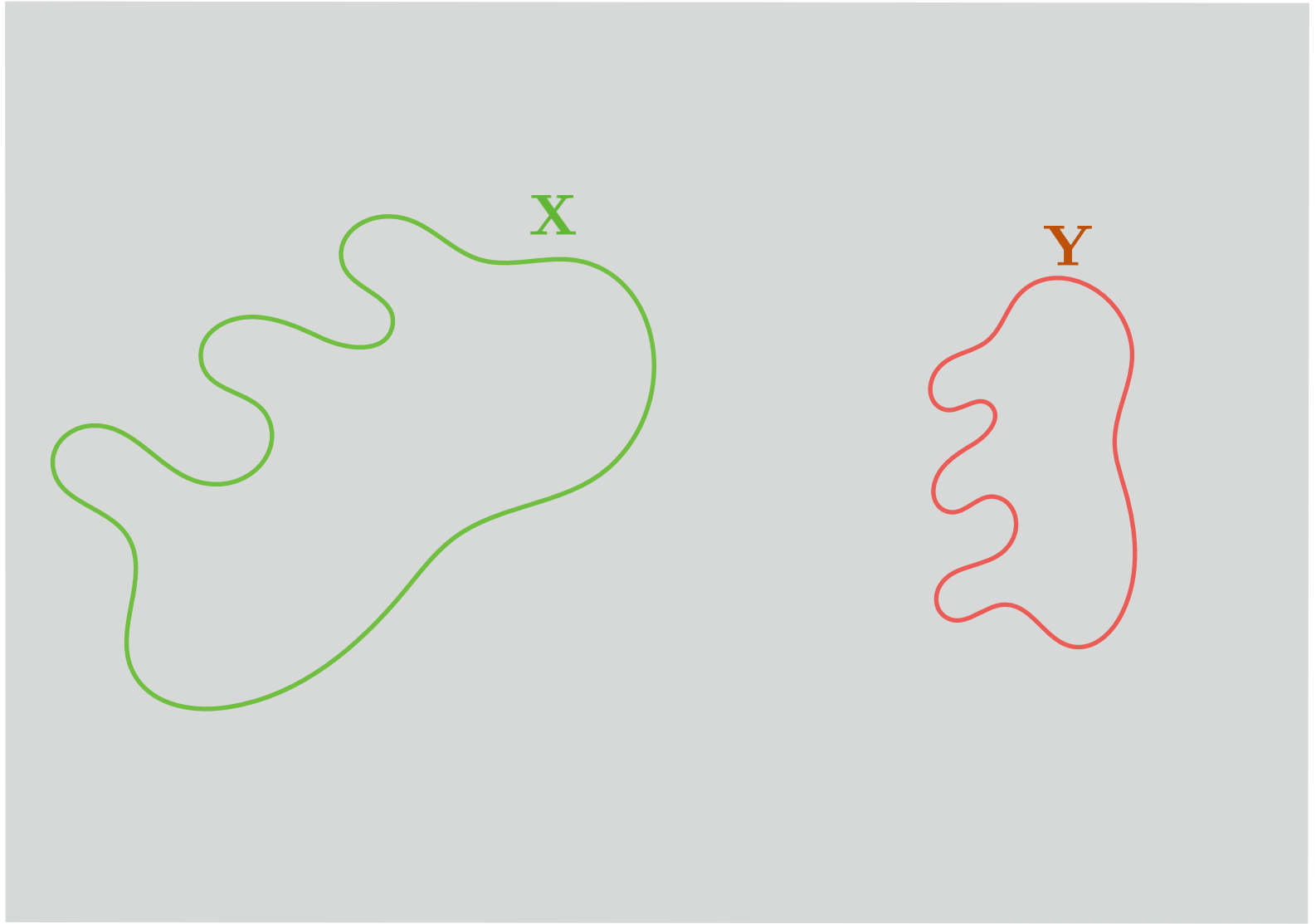
original unsorted points

$$f^{j+1} = \arg\min_f \sum_i \|f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i\|^2$$

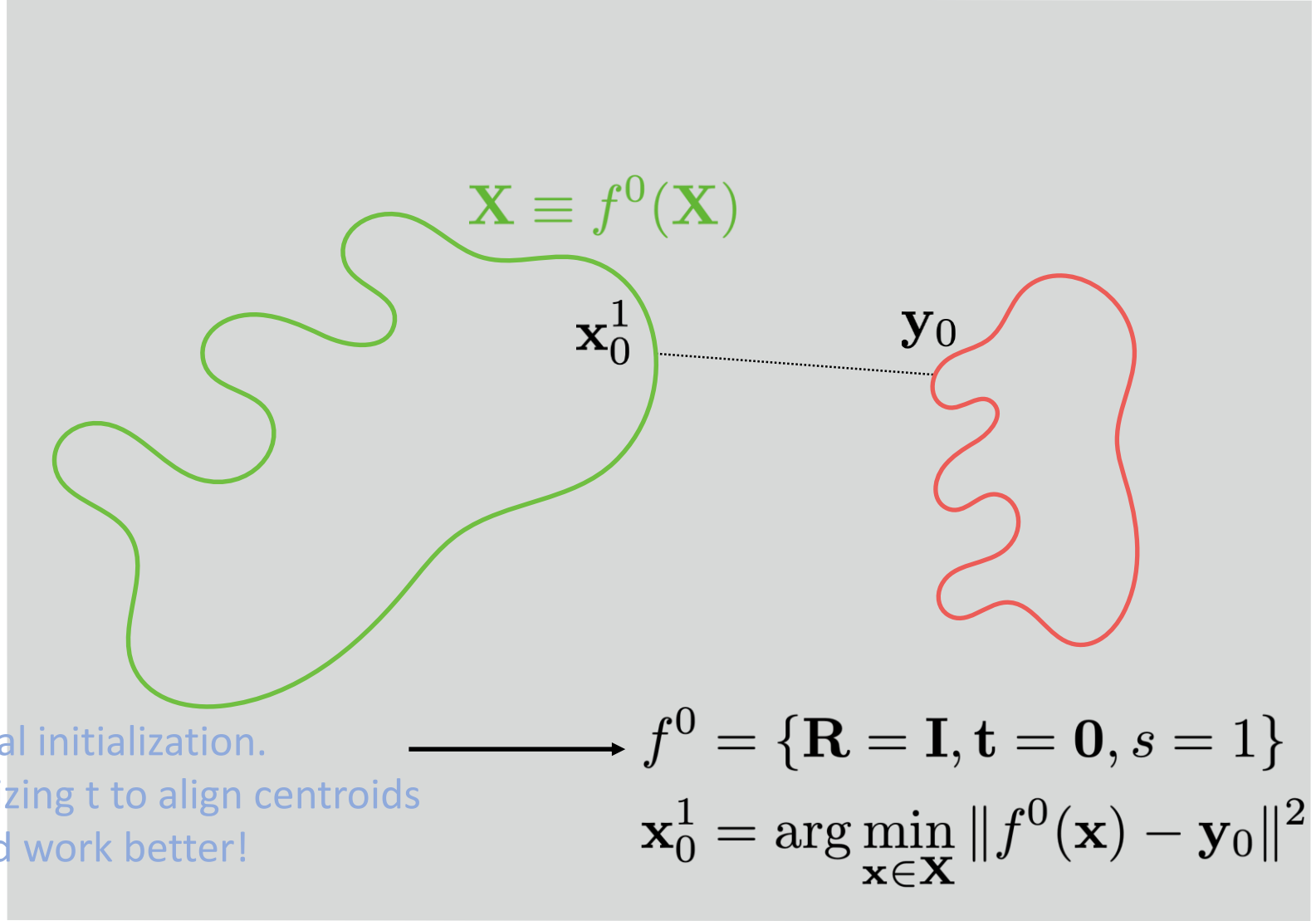
Given current best transformation, which are the closest correspondences?

Given current best correspondences, which is the best transformation?

Make up reasonable correspondences

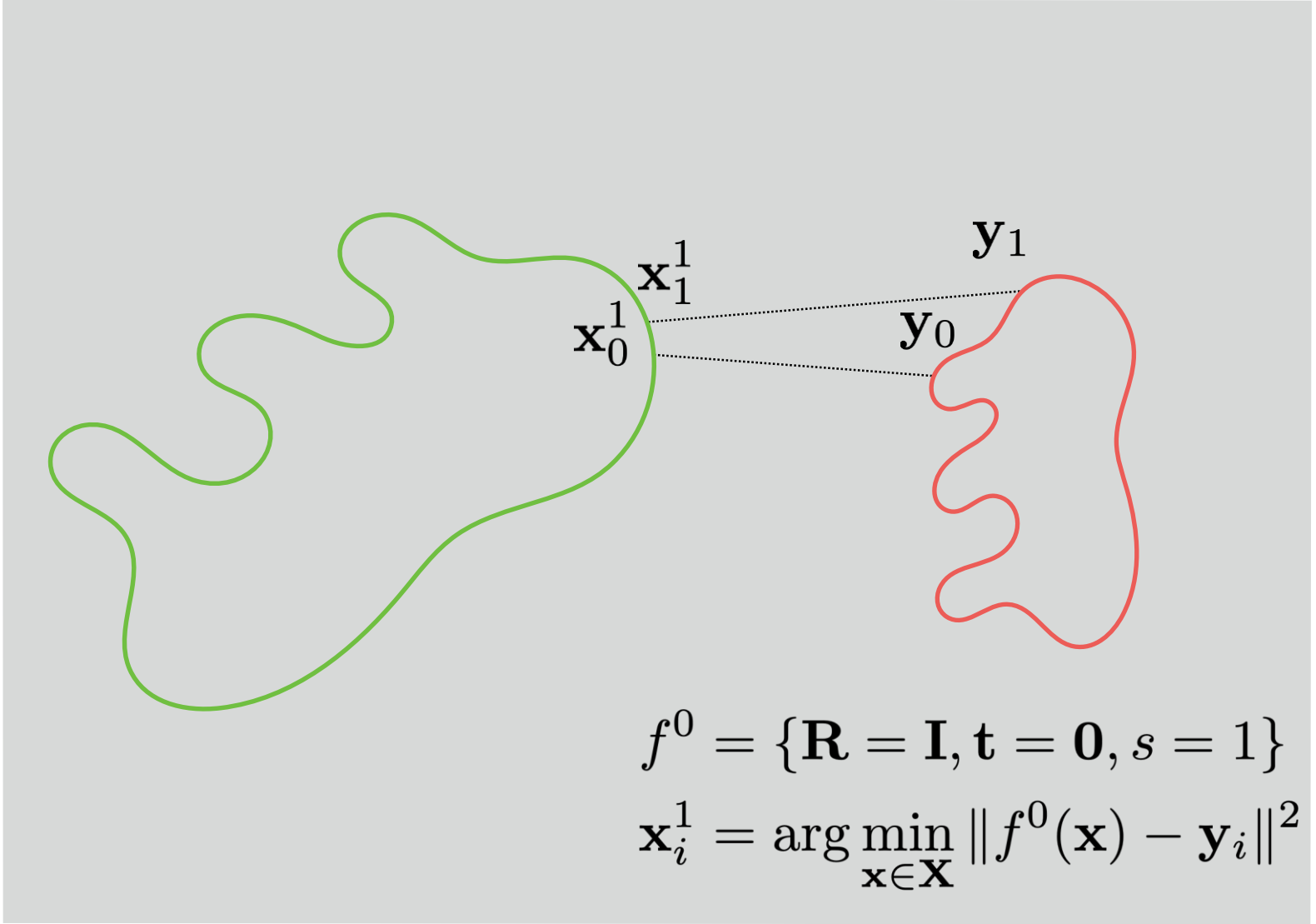


Make up reasonable correspondences

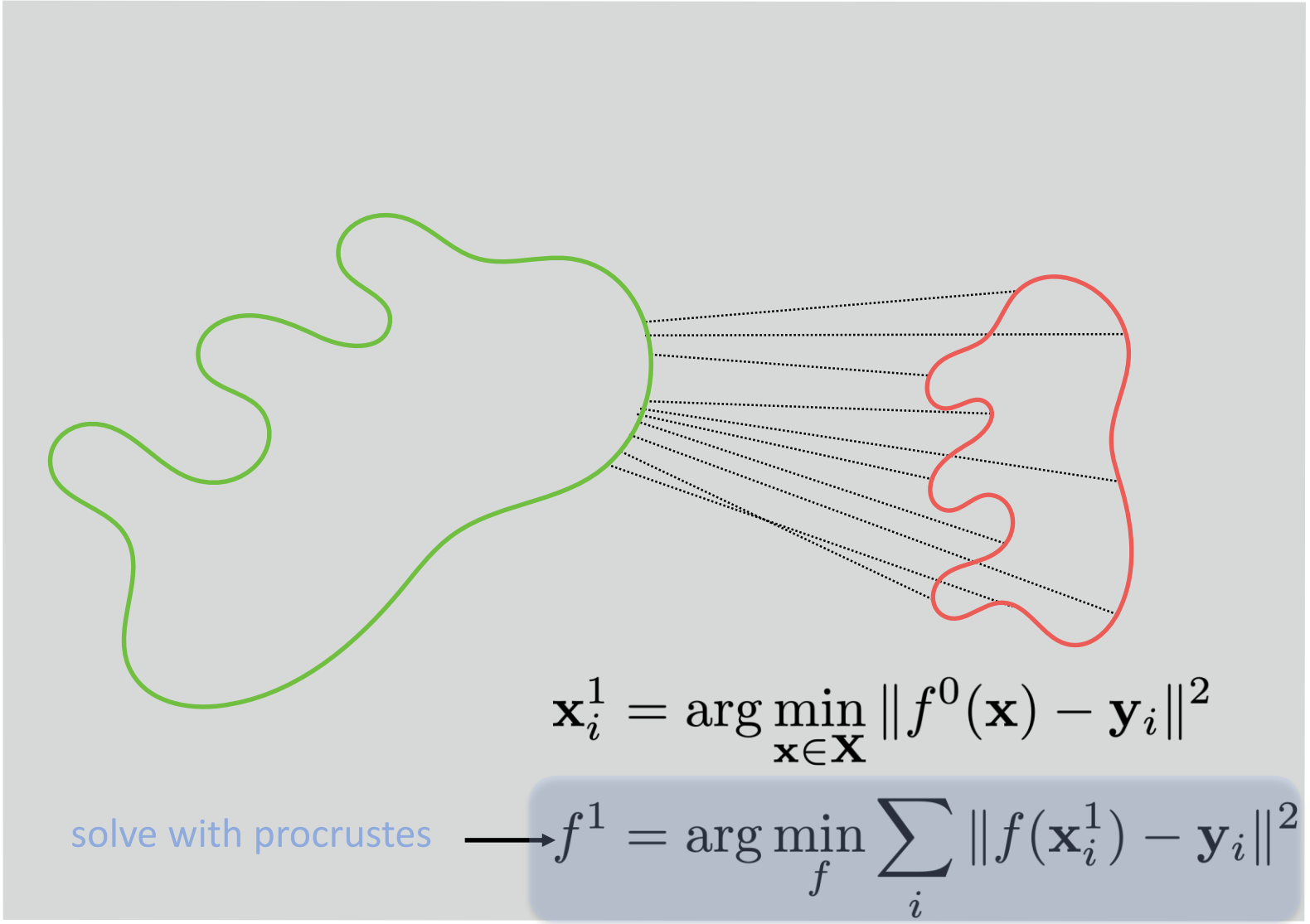


Neutral initialization.
Initializing t to align centroids should work better!

Make up reasonable correspondences



Solve for the best transformation

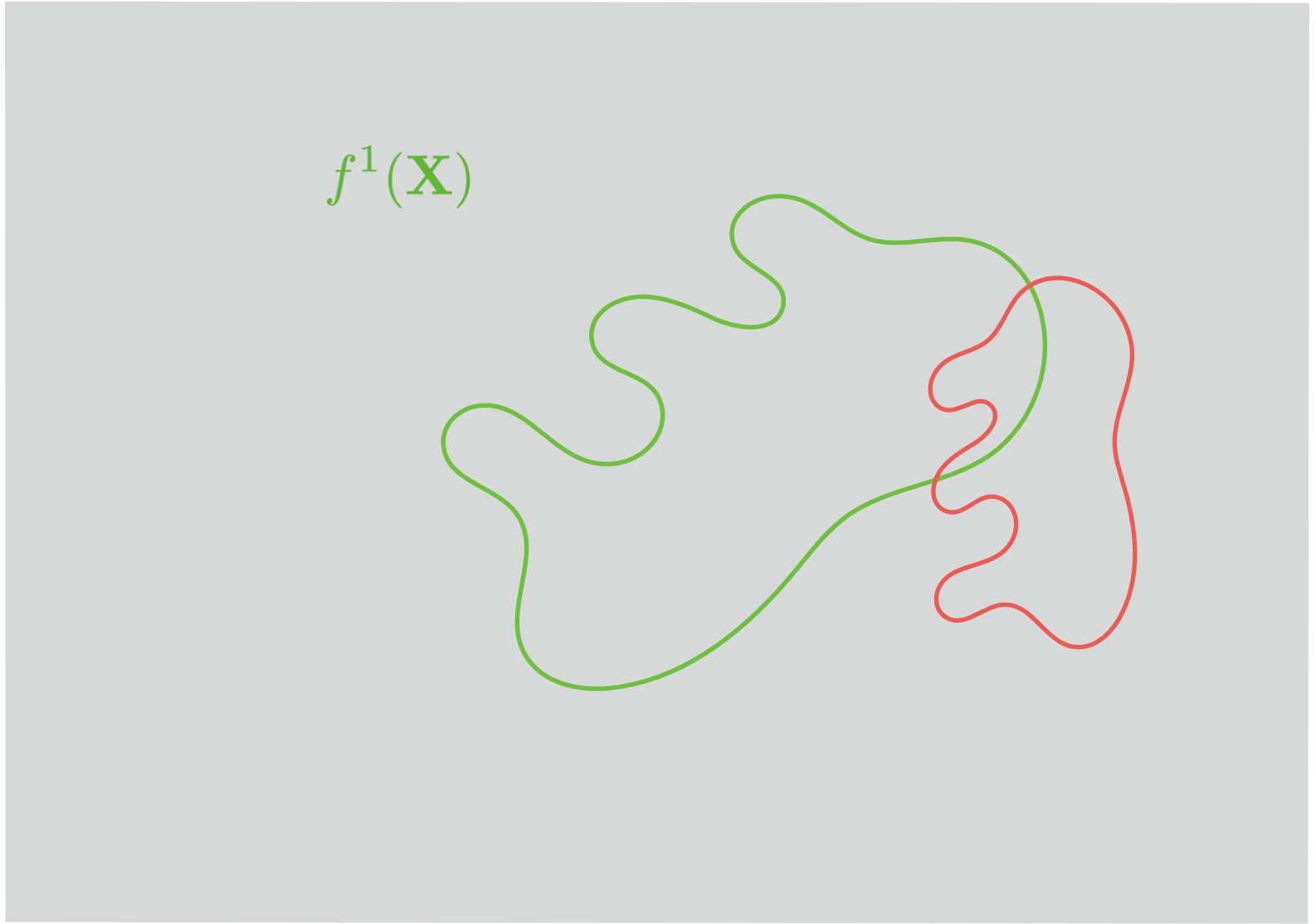


$$\mathbf{x}_i^1 = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^0(\mathbf{x}) - \mathbf{y}_i\|^2$$

solve with procrustes

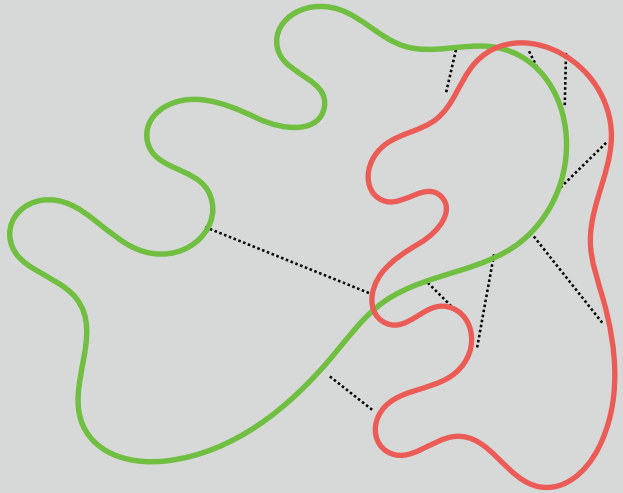
$$f^1 = \arg\min_f \sum_i \|f(\mathbf{x}_i^1) - \mathbf{y}_i\|^2$$

Apply it …



and iterate!

$f^1(\mathbf{X})$

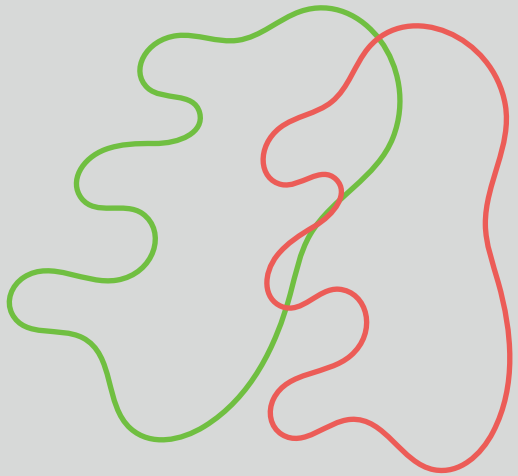


$$f^1 = \arg\min_f \sum_i \|f(\mathbf{x}_i^1) - \mathbf{y}_i\|^2$$

$$\mathbf{x}_i^2 = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^1(\mathbf{x}) - \mathbf{y}_i\|^2$$

and iterate!

$f^j(\mathbf{X})$



$$f^j = \arg\min_f \sum_i \|f(\mathbf{x}_i^j) - \mathbf{y}_i\|^2$$

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

and iterate!

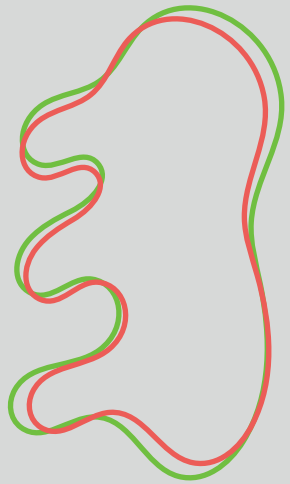
$f^j(\mathbf{X})$

$$f^j = \arg\min_f \sum_i \|f(\mathbf{x}_i^j) - \mathbf{y}_i\|^2$$

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

and iterate!

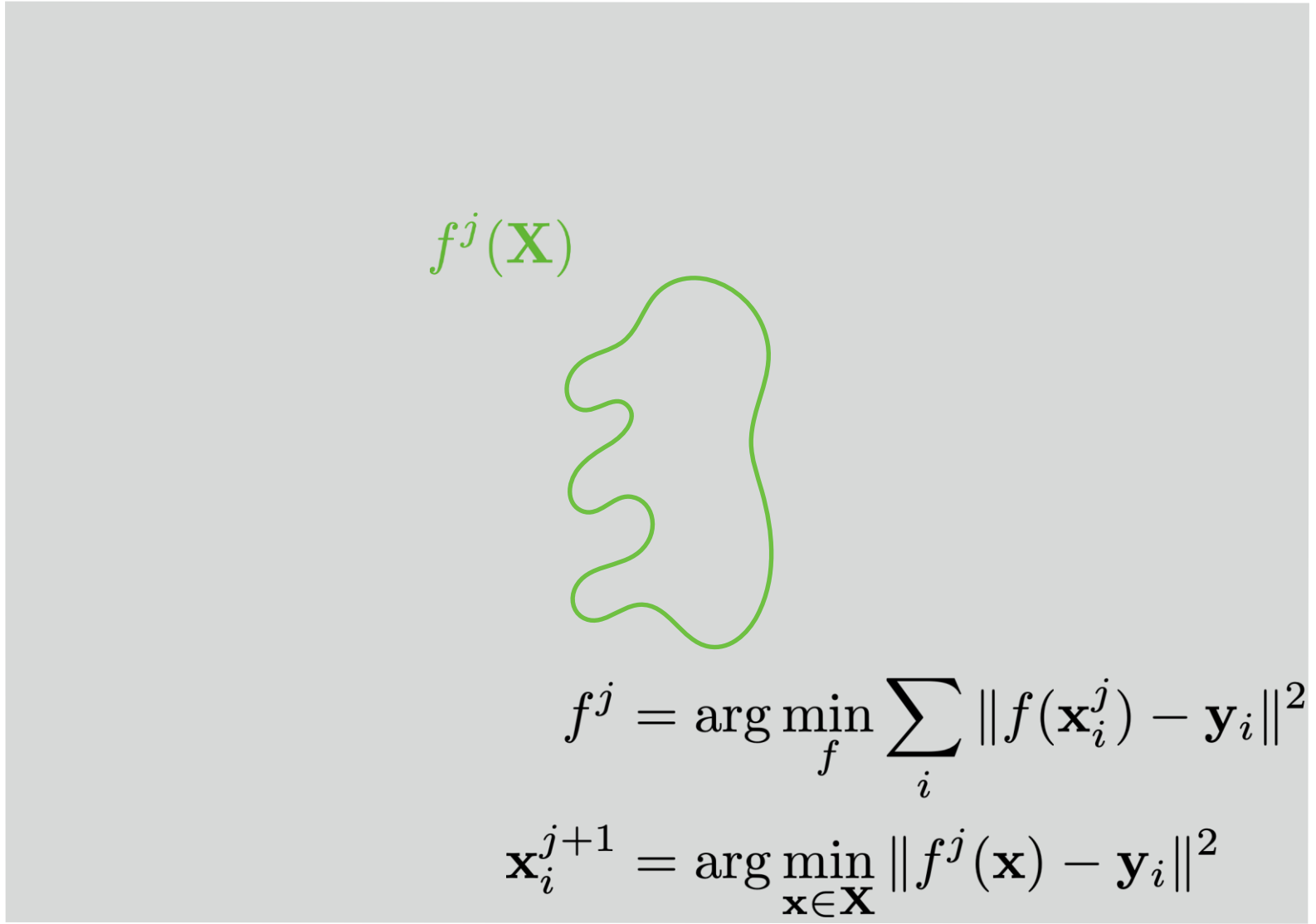
$f^j(\mathbf{X})$



$$f^j = \arg\min_f \sum_i \|f(\mathbf{x}_i^j) - \mathbf{y}_i\|^2$$

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

and iterate!


$f^j(\mathbf{X})$
$f^j = \arg\min_f \sum_i \|f(\mathbf{x}_i^j) - \mathbf{y}_i\|^2$
$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$

Iterative Closest Point (ICP)

1. Initialize

typically better than 0

$$f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \frac{\sum \mathbf{y}_i}{N} - \frac{\sum \mathbf{x}_i}{N}, s = 1\}$$

Iterative Closest Point (ICP)

1. Initialize $f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \frac{\sum \mathbf{y}_i}{N} - \frac{\sum \mathbf{x}_i}{N}, s = 1\}$

2. Compute correspondences according to current best transform

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x} \in \mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

Iterative Closest Point (ICP)

1. Initialize

$$f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \frac{\sum \mathbf{y}_i}{N} - \frac{\sum \mathbf{x}_i}{N}, s = 1\}$$

2. Compute correspondences according to current best transform

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x} \in \mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

3. Compute optimal transformation ($\mathbf{s}, \mathbf{R}, \mathbf{t}$)with Procrustes

$$f^{j+1} = \arg\min_f \sum_i \|f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i\|^2$$

Iterative Closest Point (ICP)

1. Initialize $f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \frac{\sum \mathbf{y}_i}{N} - \frac{\sum \mathbf{x}_i}{N}, s = 1\}$

2. Compute correspondences according to current best transform
$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x} \in \mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

3. Compute optimal transformation $(\mathbf{s}, \mathbf{R}, \mathbf{t})$ with Procrustes
$$f^{j+1} = \arg\min_f \sum_i \|f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i\|^2$$

4. Terminate if converged (error below a threshold), otherwise iterate

Iterative Closest Point (ICP)

1. Initialize $f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \frac{\sum \mathbf{y}_i}{N} - \frac{\sum \mathbf{x}_i}{N}, s = 1\}$
2. Compute correspondences according to current best transform
$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x} \in \mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$
3. Compute optimal transformation $(\mathbf{s}, \mathbf{R}, \mathbf{t})$ with Procrustes
$$f^{j+1} = \arg\min_f \sum_i \|f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i\|^2$$
4. Terminate if converged (error below a threshold), otherwise iterate
5. Converges to local minima

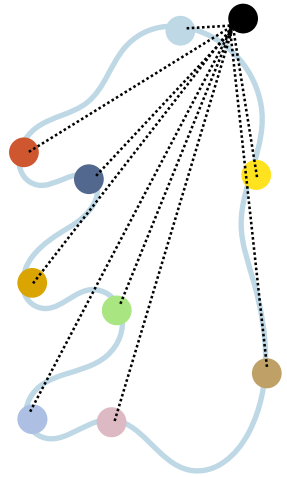
Is ICP the best we can do?

Iteration j:

- compute closest points
- compute optimal transformation with Procrustes
- apply transformation
- terminate if converged, otherwise iterate

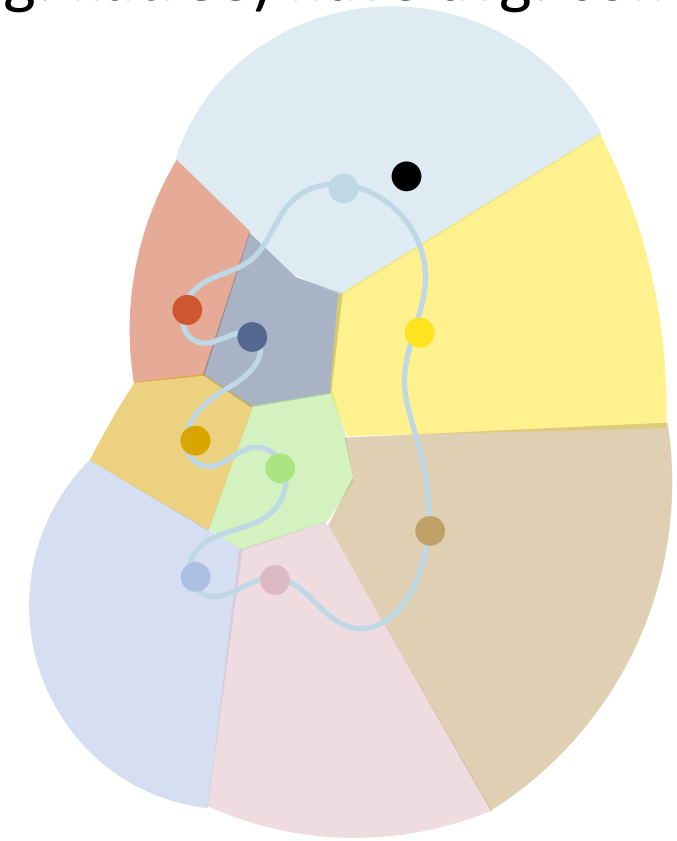
Closest points

- Brute force is $O(n^2)$
- For every source point find a neighbor point on the source shape



Closest points

- Tree based methods (e.g. kdtree) have avg. complexity log(n)

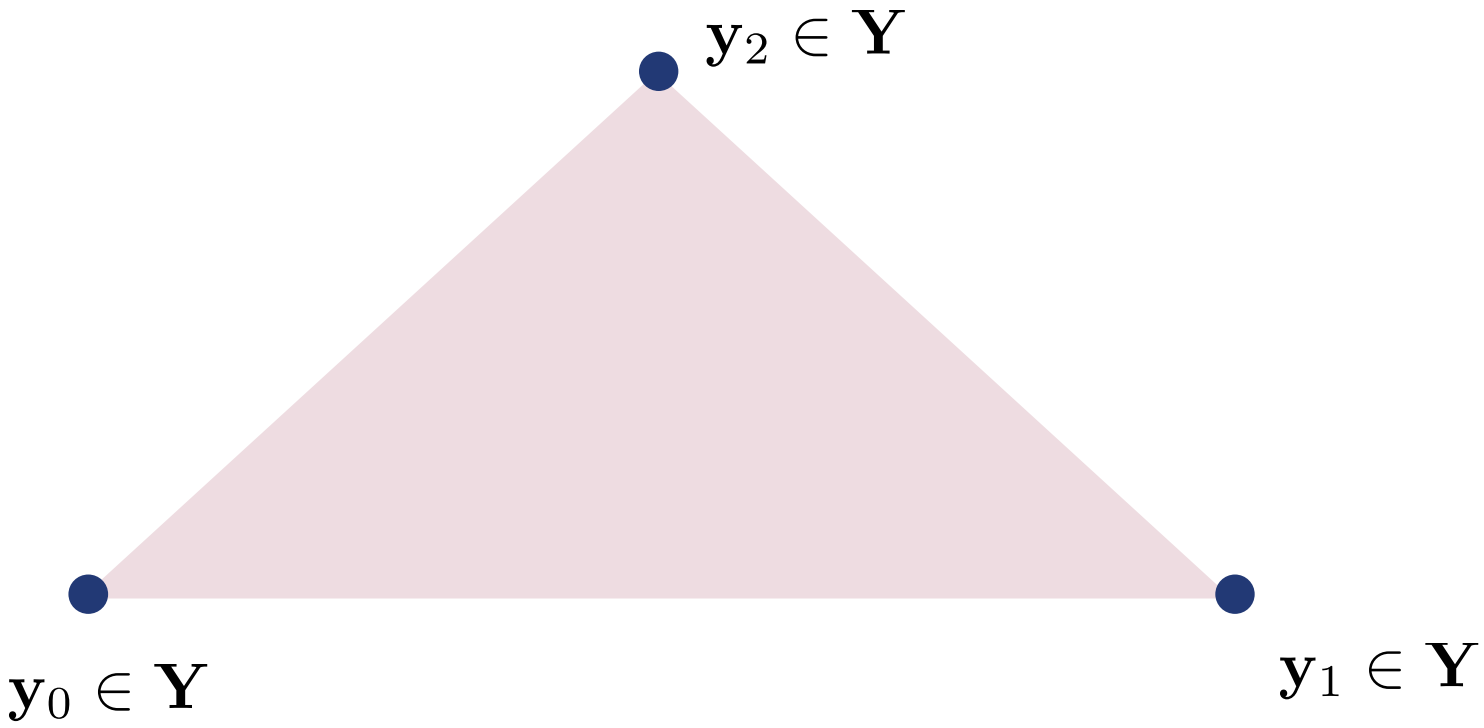


- Random point sampling also reduces the running time

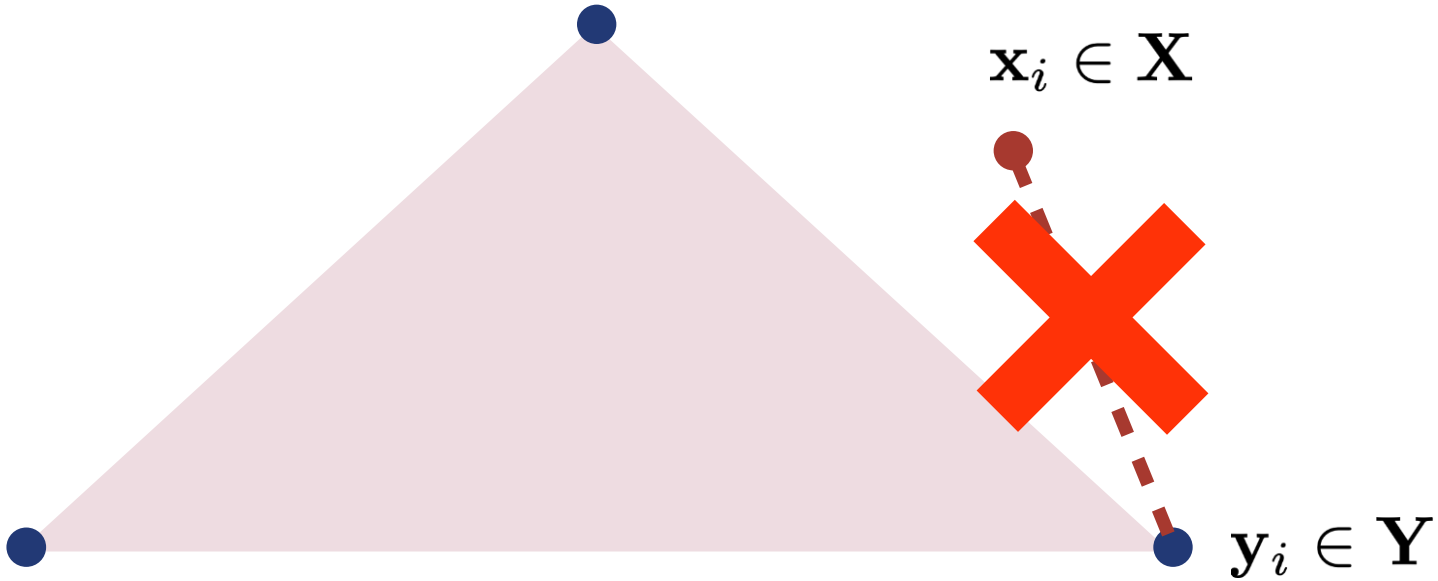
ICP: Tips to avoid local minima

- Always find correspondences from target to source!
  Proper **data term**
- Outliers —> Robust cost functions
- Use additional information (e.g. normals)
- Compute transformation based on greedy subsets of points: RANSAC

A much better objective: Point-to-surface distance

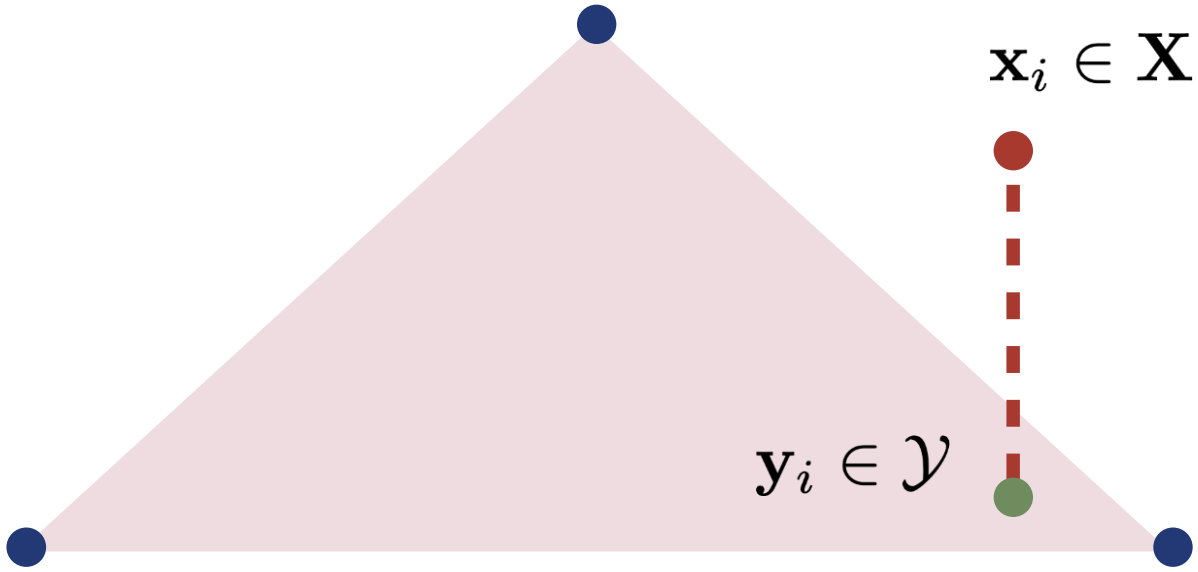


Closest points: avoid local minima



Point-to-point distance

Closest points: avoid local minima



Point-to-surface distance

Is ICP the best we can do?

Iteration j:

- compute closest points
- compute optimal transformation with Procrustes
- apply transformation
- terminate if converged, otherwise iterate

Best transformation?

- Procrustes gives us the optimal **rigid** transformation and scale given correspondences

- What if the deformation model is **not rigid** ?

- Can we generalise ICP to non-rigid deformation ?

Iterative Closest Point (ICP)

Iteration j:

- compute closest points ➔ Which direction to move?
- compute optimal transformation with Procrustes
- apply transformation
- terminate if converged, otherwise iterate

Iterative Closest Point (ICP)

Iteration j:

- compute closest points ➔ Which direction to move?
- compute optimal transformation with Procrustes ➔
  Compute a transform that reduces the error
- apply transformation
- terminate if converged, otherwise iterate

Gradient-based ICP

Iteration j:

- compute closest points ➔ Which direction to move?

- ~~compute optimal transformation with Procrustes~~ ➔

  Compute descent step by linearising the energy

  Jacobian of distance-based energy

- apply transformation

- terminate if converged, otherwise iterate

Gradient-based ICP

$$\arg\min_f E(f) = \arg\min_f \sum_i \| f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i \|^2$$

- If f is a rigid transformation we can solve this minimisation using Procrustes

- If f is a general non-linear function ?
- Gradient descent: $f^{k+1} = f^k - \lambda \nabla_f E(f)$

- For least squares, is there a better optimisation method ?
  yes: Gauss-Newton based methods.

Gradient-based ICP

1. Energy:

$$E \equiv \sum_i \| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \|^2$$

2. Consider the correspondences fixed in each iteration j+1

$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x} \in \mathbf{X}} \| f^j(\mathbf{x}) - \mathbf{y}_i \|^2$$

3. Compute gradient of the energy around current estimation

$$g^{j+1} = \nabla E(f^j)$$

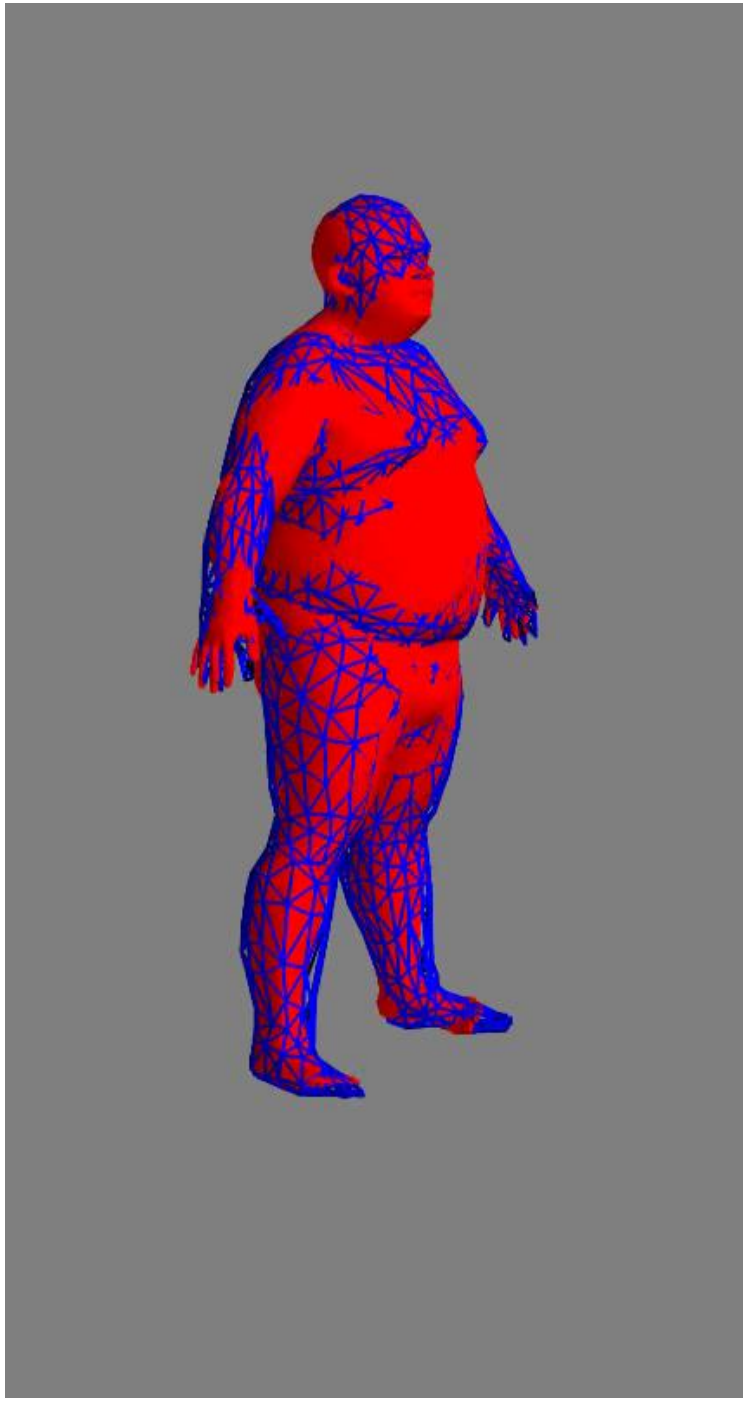
4. Apply step (gradient descent, dogleg, LM, BFGS…)

$$f^{j+1} = k_{step}(g^{0\ldots j+1}, f^{0\ldots j})$$

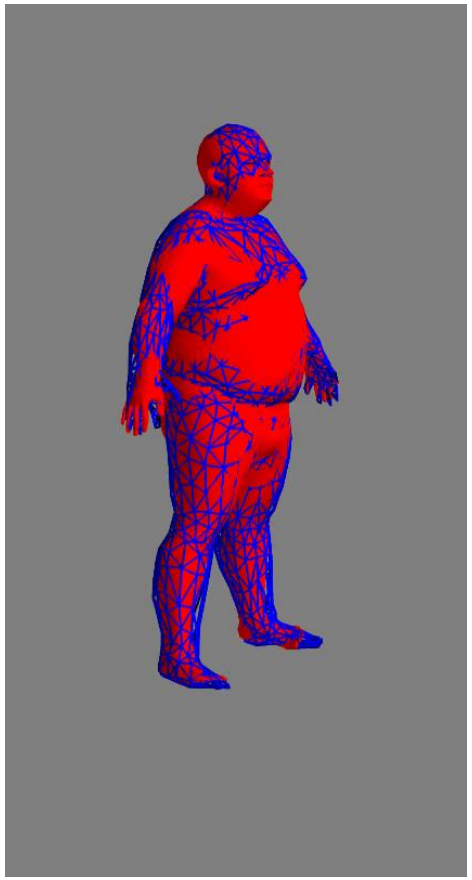
(for example $f^{j+1} = f^j - \alpha g^{j+1}$)

5. terminate if converged, otherwise iterate (go to step 2)

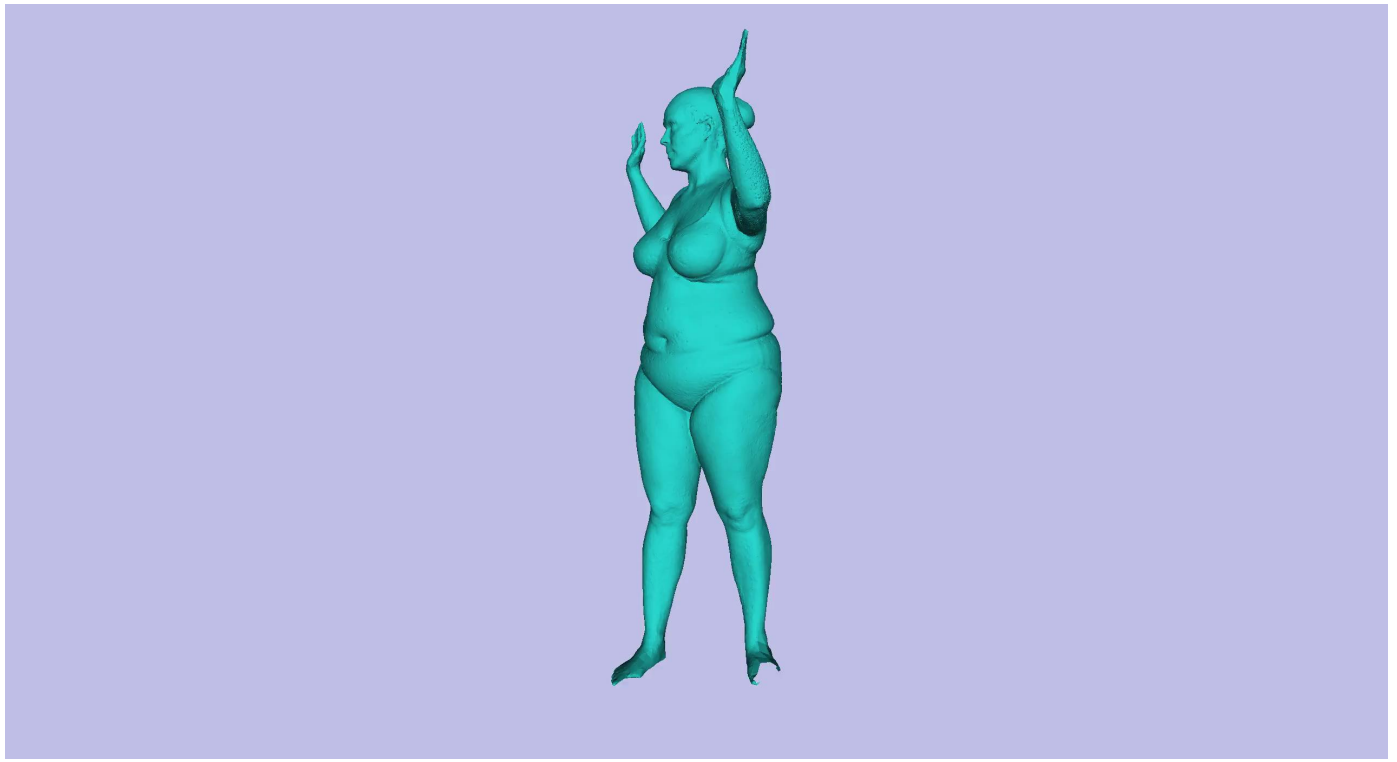
Gradient-based ICP



Why is convergence on the left less smooth?



Point to point objective



Point to surface objective

Gradient-based ICP

- Energy:
- Consider the correspondences fixed in each iteration j+1
- Compute gradient of the energy around current estimation
- Apply step (gradient descent, dogleg, LM, BFGS…)
- terminate if converged, otherwise iterate $f^{j+1} = k_{step}(g^{0\ldots j+1}, f^{0\ldots j})$

Gradient-based ICP

- Gradient: derivative of the sum of squared distances with respect to transformation f parameters

$$E \equiv \sum_i \| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \|^2$$

$$g^{j+1} = \nabla E(f^j)$$

Gradient-based ICP

- Gradient: derivative of the sum of squared distances with respect to transformation f parameters

$$E \equiv \sum_i \| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \|^2$$

$$g^{j+1} = \nabla E(f^j)$$

- Each derivative is easy
  - Who wants to writes it down?

Gradient-based ICP

- Gradient: derivative of the sum of squared distances with respect to transformation f parameters

$$E \equiv \sum_i \| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \|^2$$

$$g^{j+1} = \nabla E(f^j)$$

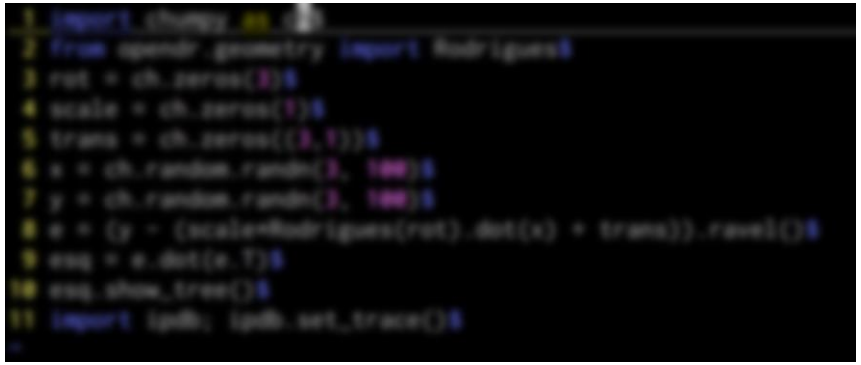
- Each derivative is easy
  - Who wants to writes it down?

- **Chain rule and automatic differentiation!**

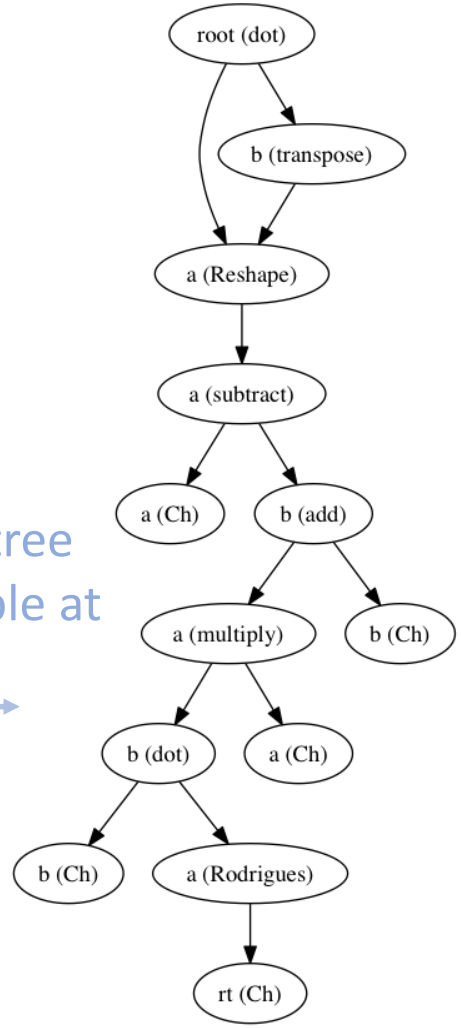
Automatic differentiation

$$E = \sum_i \|s\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^2$$

write as if it was numpy code



results in expression tree with jacobians available at each step



Gradient-based ICP

- Energy:
- Consider the correspondences fixed in each iteration j+1
- Compute gradient of the energy around current estimation
- Apply step (gradient descent, dogleg, LM, BFGS…)
- terminate if converged, otherwise iterate $f^{j+1} = k_{step}(g^{0\ldots j+1}, f^{0\ldots j})$

Why Gradient-based ICP?

- Formulation is much more generic: the energy can incorporate other terms, more parameters, etc
- A lot of available software for solving this least squares problem (cvx, ceres, …)
- **However,** the resulting energy is non-convex for general deformation models. Optimisation can get trapped in local minima.

Take-home message

- **Procrustes** is **optimal** for **rigid alignment problems** with *known* correspondences. For other problems:

- We can compute **correspondences** and solve for the best **transformation** iteratively with Iterative Closest Point (**ICP**)

Slide credits

- Javier Romero