

# Hands-on AI based 3D Vision Summer Semester 26

Lecture 5\_0 – Surface Reconstruction

Prof. Dr.-Ing Gerard Pons-Moll

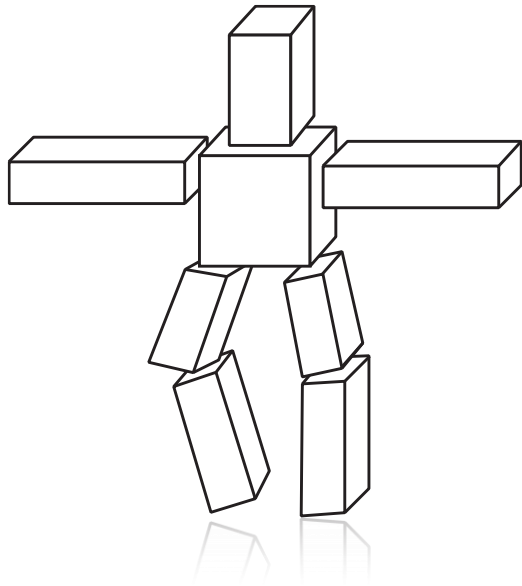
University of Tübingen / MPI-Informatics

EBERHARD KARLS  
UNIVERSITÄT  
TÜBINGEN



# Increasing complexity of our models

**Transformations**



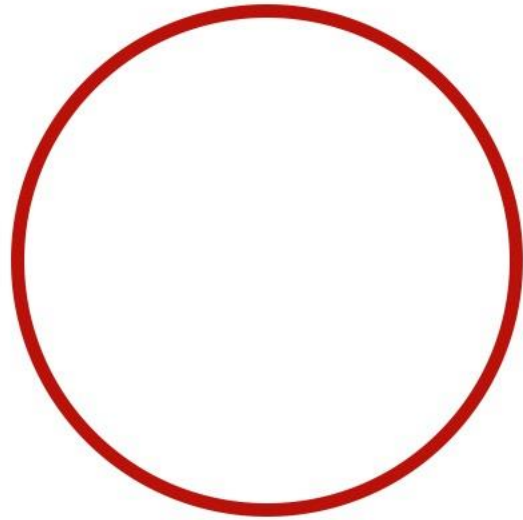
**Geometry**



**Materials, lighting, ...**



# How can we describe geometry?



# How can we describe geometry?

**IMPLICIT**

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

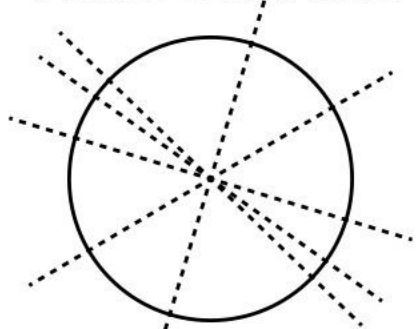
**LINGUISTIC**

“unit circle”

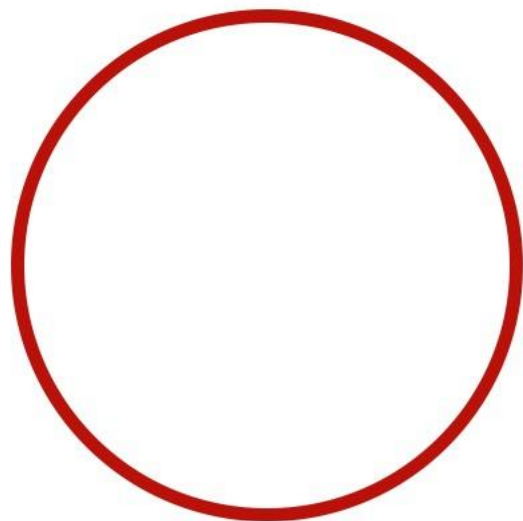
**EXPLICIT**

$$\underbrace{(\cos \theta)}_x, \underbrace{(\sin \theta)}_y$$

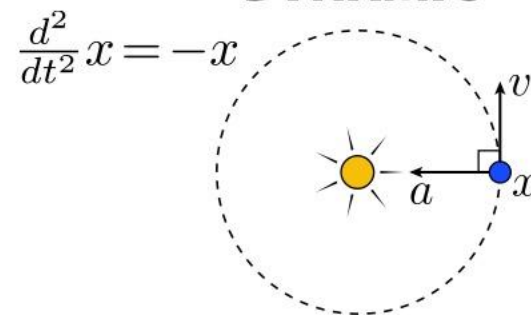
**TOMOGRAPHIC**



(constant density)



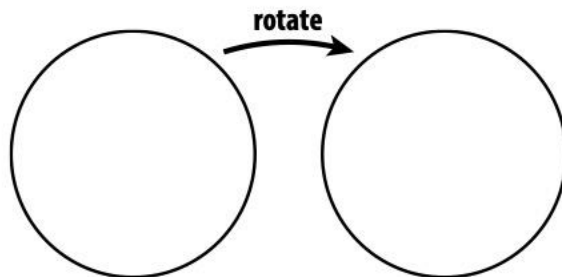
**DYNAMIC**

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


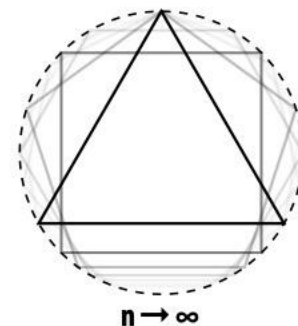
**CURVATURE**

$$\kappa = 1$$

**SYMMETRIC**



**DISCRETE**



$n \rightarrow \infty$

# Examples of geometry



# Examples of geometry



# Examples of geometry



# Examples of geometry

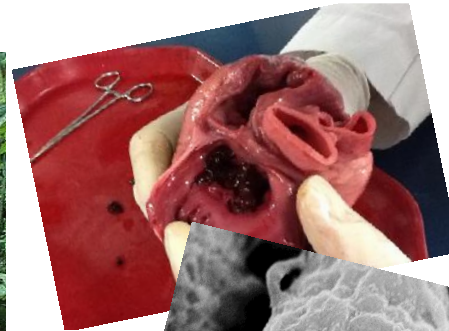


# Examples of geometry



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**

# Many ways to digitally encode geometry

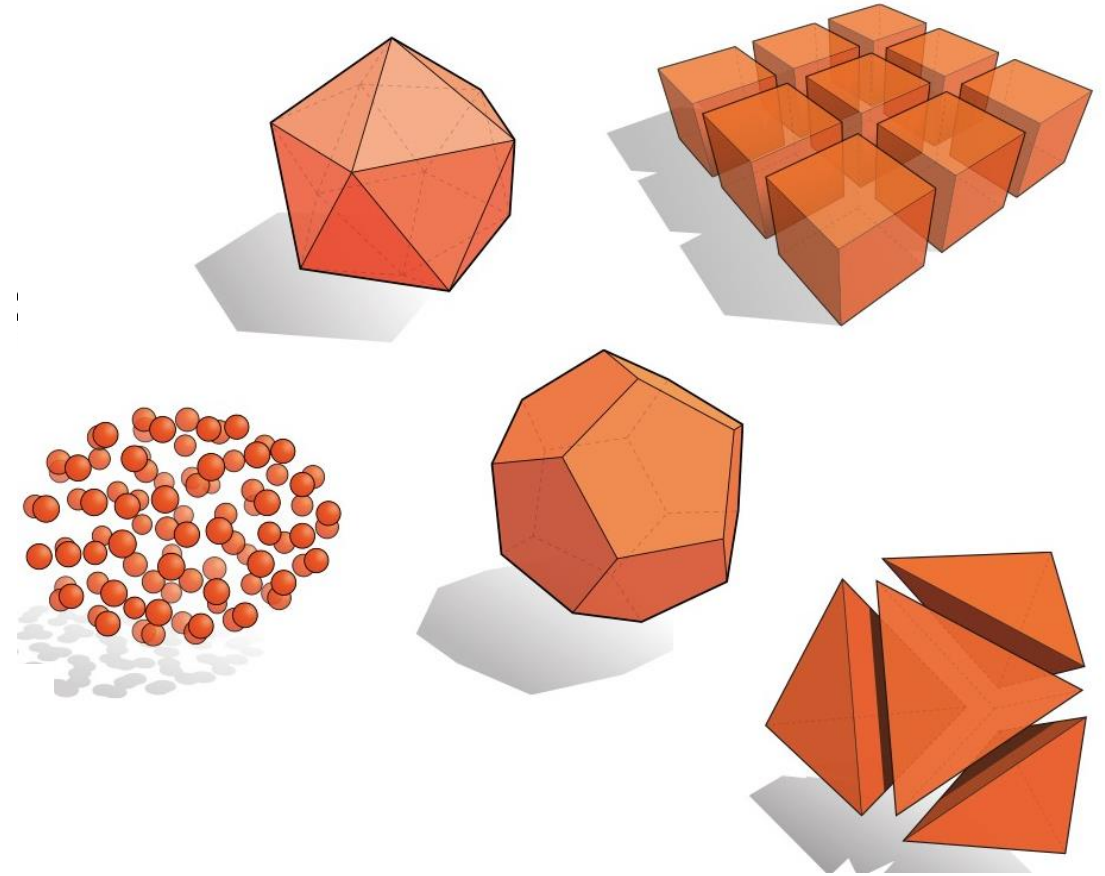
- EXPLICIT

- point cloud
- polygon mesh
- subdivision, NURBS
- ...

- IMPLICIT

- level set
- algebraic surface
- L-systems
- ...

- Each choice best suited to a different task/type of geometry



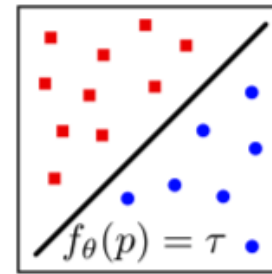
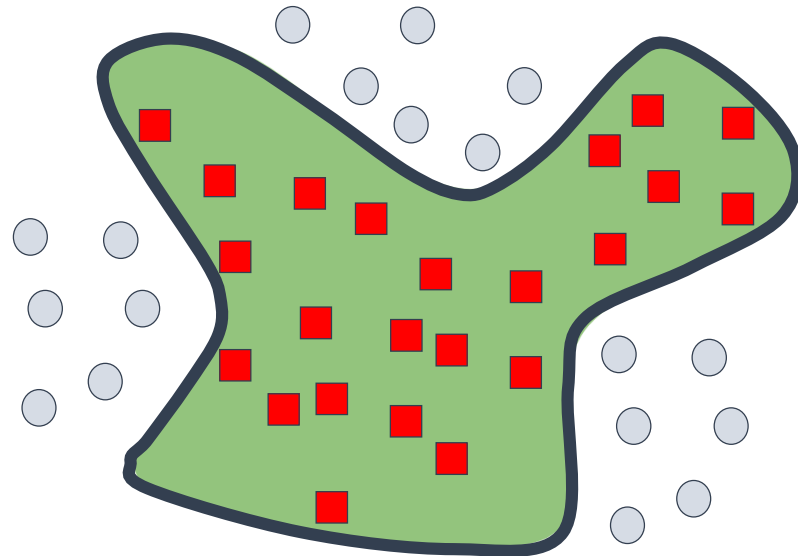
# "Implicit" Representations of Geometry

- 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

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

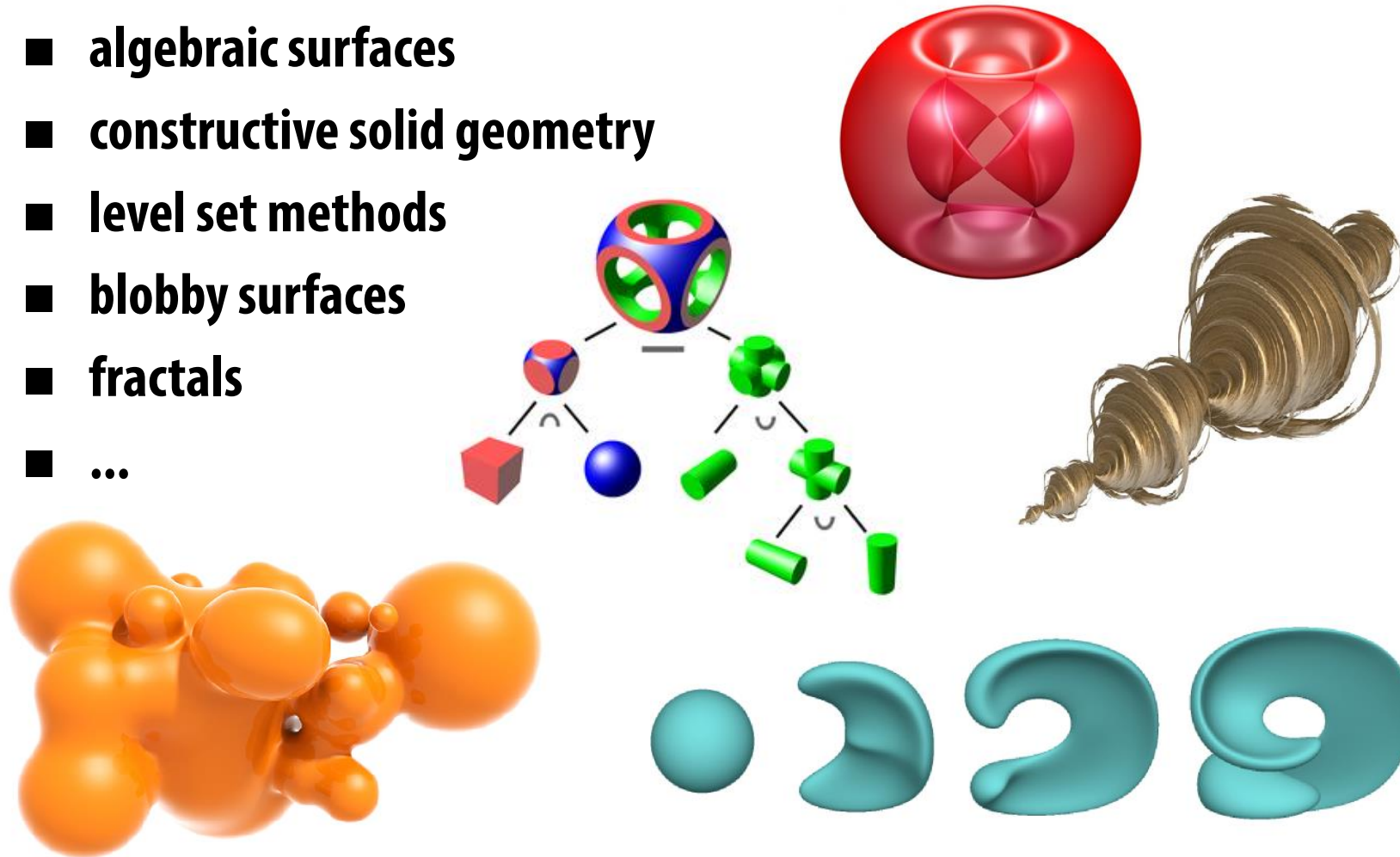
$$f(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in \text{outside } \circ \\ 1, & \text{if } \mathbf{p} \in \text{inside } \blacksquare \end{cases}$$



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

# 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.**

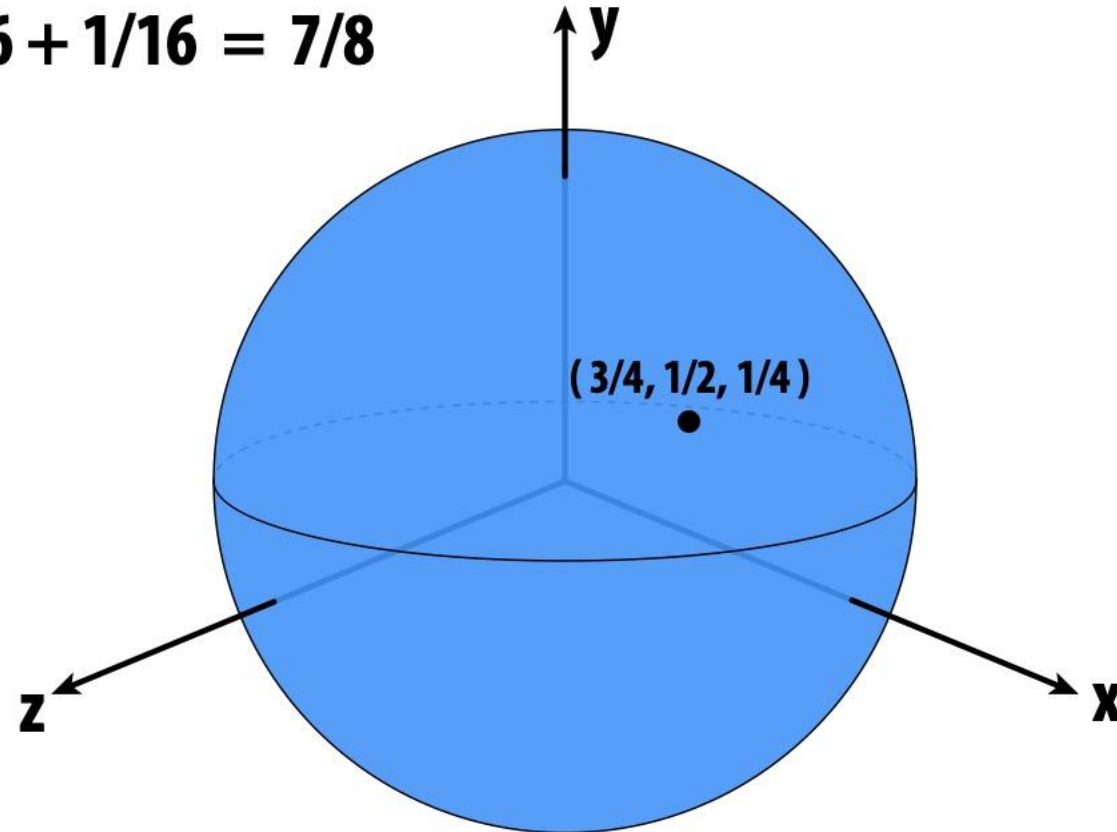
# Check if this point is inside the unit sphere

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:



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



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



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

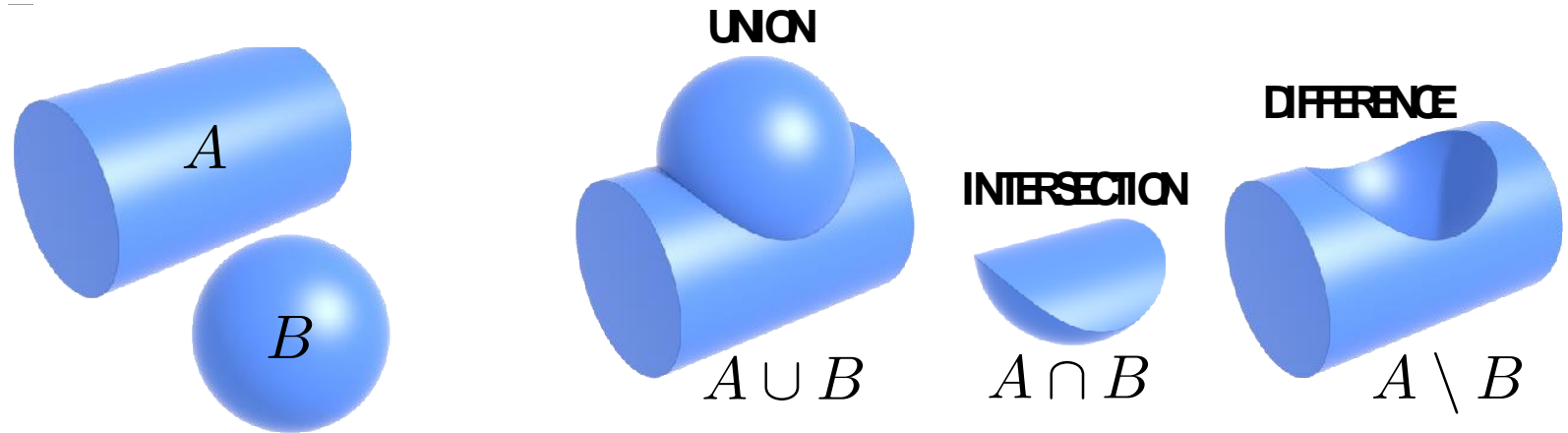
- What about more complicated shapes?
- Very hard to come up with polynomials!



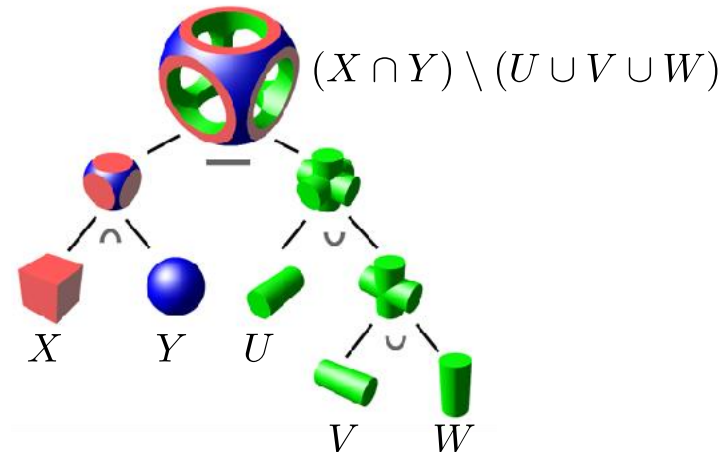
# Constructive Solid Geometry (Implicit)

- Build more complicated shapes via Boolean operations

- Basic operations:

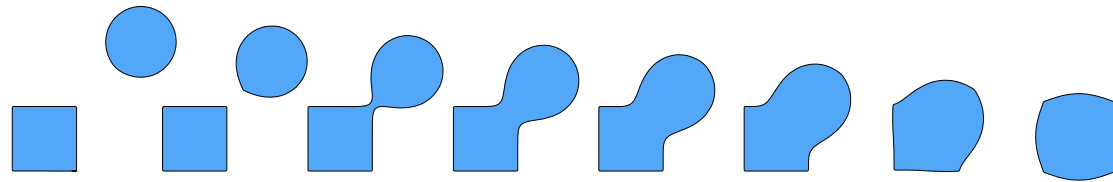


- Then chain together expressions:



# Blending Distance Functions (Implicit)

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



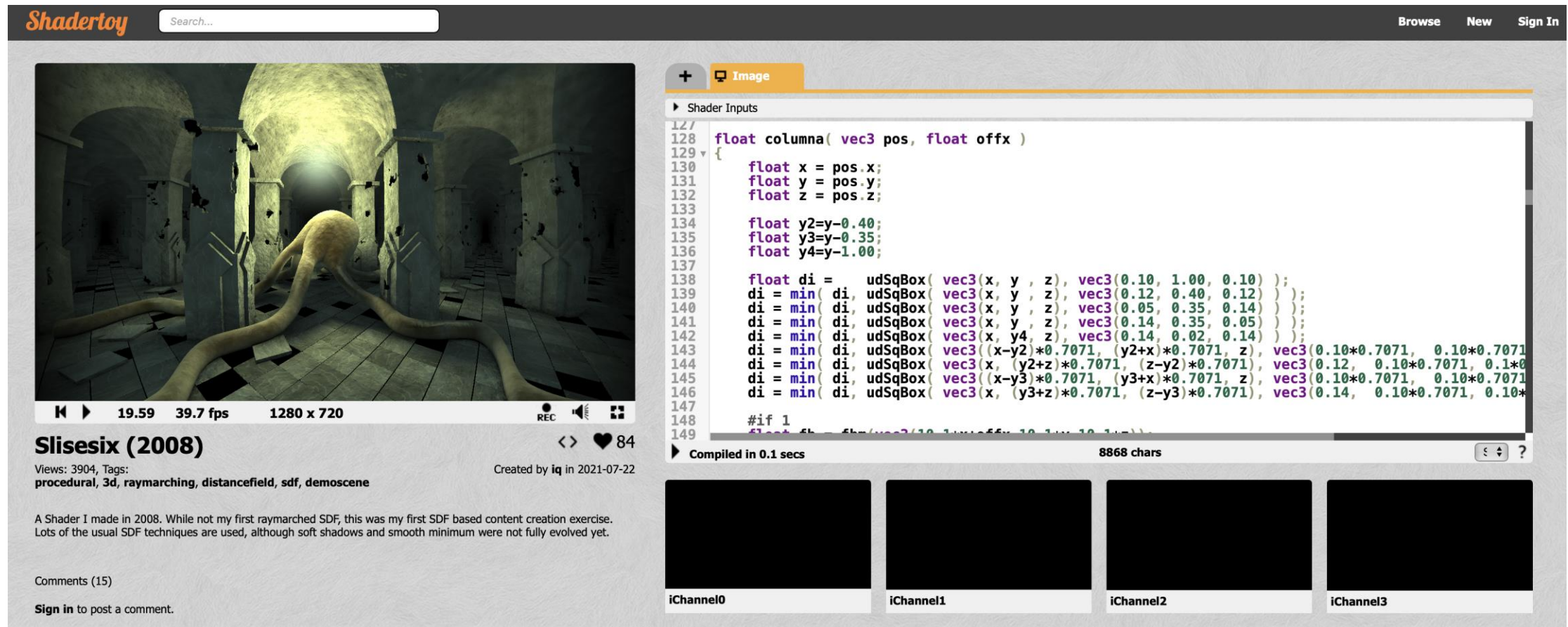
- Similar strategy to points, though many possibilities. E.g.,

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

- 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!



The screenshot shows the Shadertoy website interface. On the left is a video player showing a 3D scene with a large, multi-limbed creature in a dark, arched hallway. Below the video are controls and metadata: "Slisesix (2008)", "Views: 3904", "Tags: procedural, 3d, raymarching, distancefield, sdf, demoscene", and "Created by iq in 2021-07-22". On the right is a code editor showing the shader source code for the scene. The code defines a function `columna` that uses signed distance functions to create the scene's geometry. Below the code editor are four preview windows labeled `iChannel0` through `iChannel3`, which are currently black.

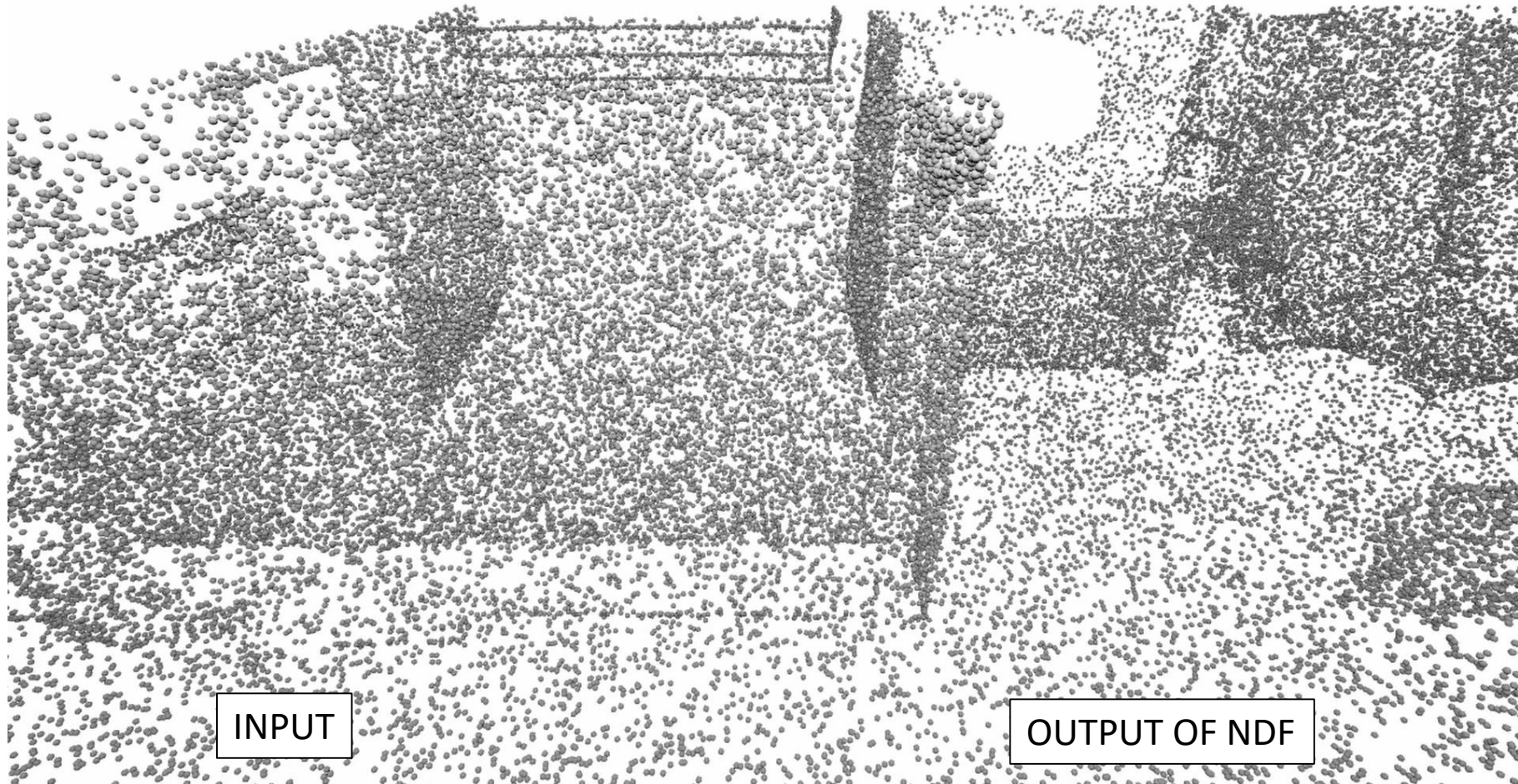
Shadertoy Search... Browse New Sign In

127  
128 float columna( vec3 pos, float offx )  
129 {  
130 float x = pos.x;  
131 float y = pos.y;  
132 float z = pos.z;  
133  
134 float y2=y-0.40;  
135 float y3=y-0.35;  
136 float y4=y-1.00;  
137  
138 float di = udSqBox( vec3(x, y, z), vec3(0.10, 1.00, 0.10) );  
139 di = min( di, udSqBox( vec3(x, y, z), vec3(0.12, 0.40, 0.12) ) );  
140 di = min( di, udSqBox( vec3(x, y, z), vec3(0.05, 0.35, 0.14) ) );  
141 di = min( di, udSqBox( vec3(x, y, z), vec3(0.14, 0.35, 0.05) ) );  
142 di = min( di, udSqBox( vec3(x, y4, z), vec3(0.14, 0.02, 0.14) ) );  
143 di = min( di, udSqBox( vec3(x-y2)\*0.7071, (y2+x)\*0.7071, z), vec3(0.10\*0.7071, 0.10\*0.7071, 0.10\*0.7071) );  
144 di = min( di, udSqBox( vec3(x, (y2+z)\*0.7071, (z-y2)\*0.7071), vec3(0.12, 0.10\*0.7071, 0.10\*0.7071) );  
145 di = min( di, udSqBox( vec3(x-y3)\*0.7071, (y3+x)\*0.7071, z), vec3(0.10\*0.7071, 0.10\*0.7071, 0.10\*0.7071) );  
146 di = min( di, udSqBox( vec3(x, (y3+z)\*0.7071, (z-y3)\*0.7071), vec3(0.14, 0.10\*0.7071, 0.10\*0.7071) );  
147  
148 #if 1  
149 float fb = fb( vec3(10.1-w+offx, 10.1-w, 10.1-w) );  
150  
151 return di\*fb;  
152 }  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

Compiled in 0.1 secs 8868 chars

iChannel0 iChannel1 iChannel2 iChannel3

# Nowadays -- Neural Distance Fields (NDF)



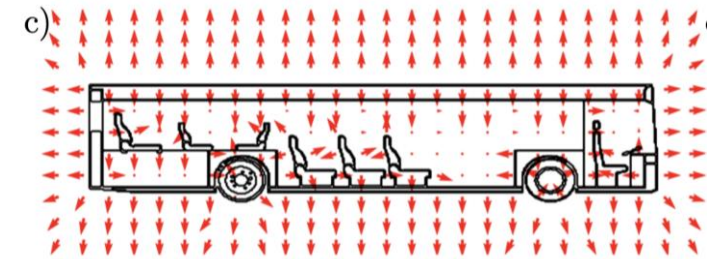
# Distance field, field normals and closest points

- **Distance to surface** is given by the distance field itself  $f(\mathbf{p})$

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

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

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

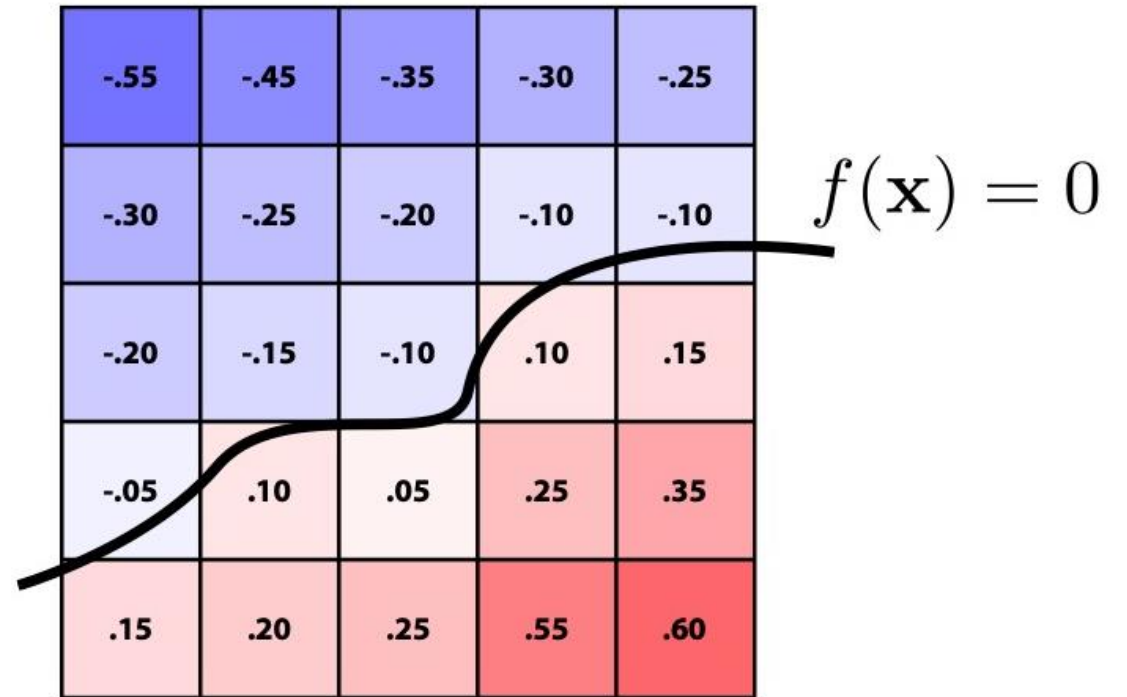


- **Closest points** are found trivially as:

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

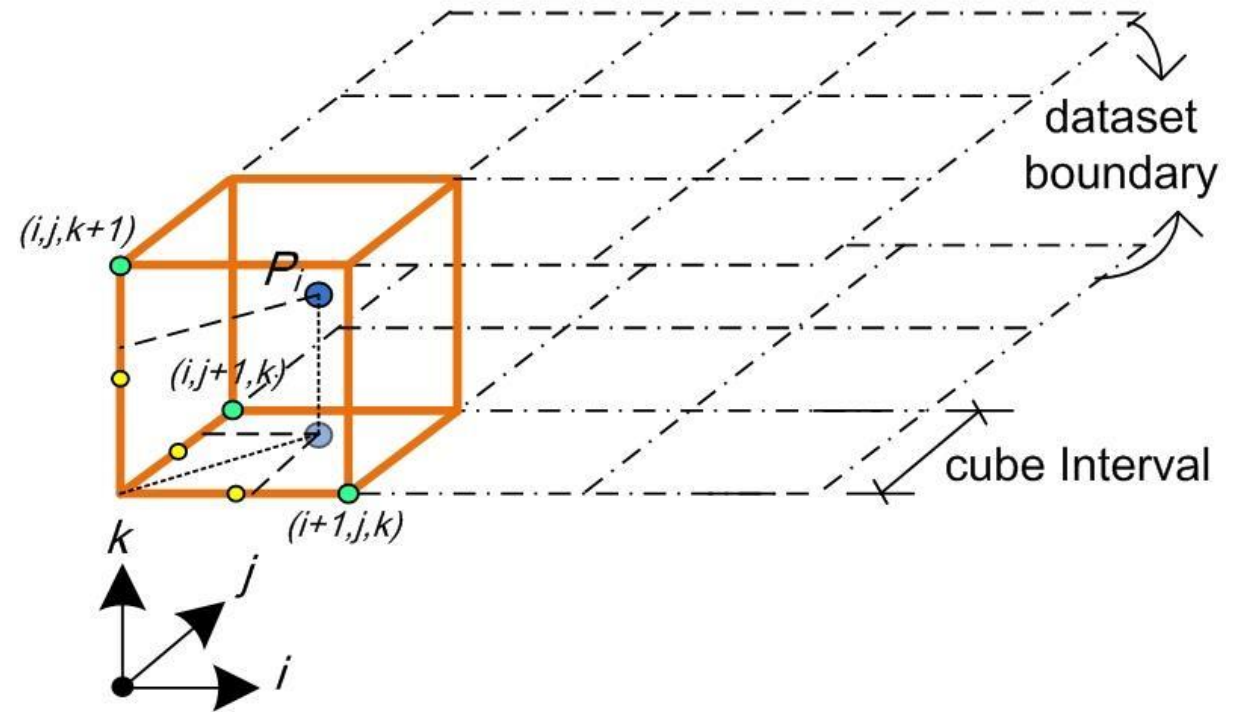
# Surface Reconstruction from Implicit Representation

- 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



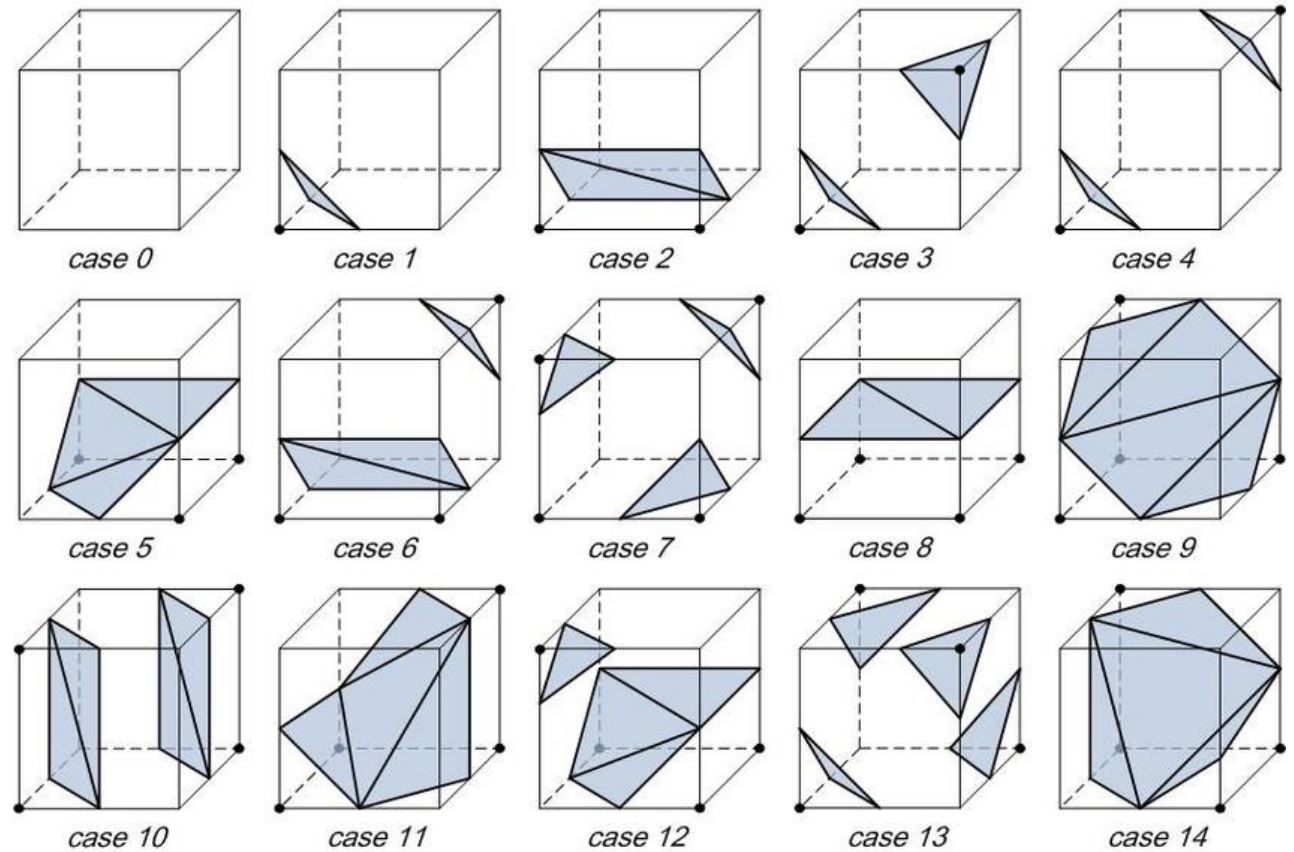
# Marching Cubes

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

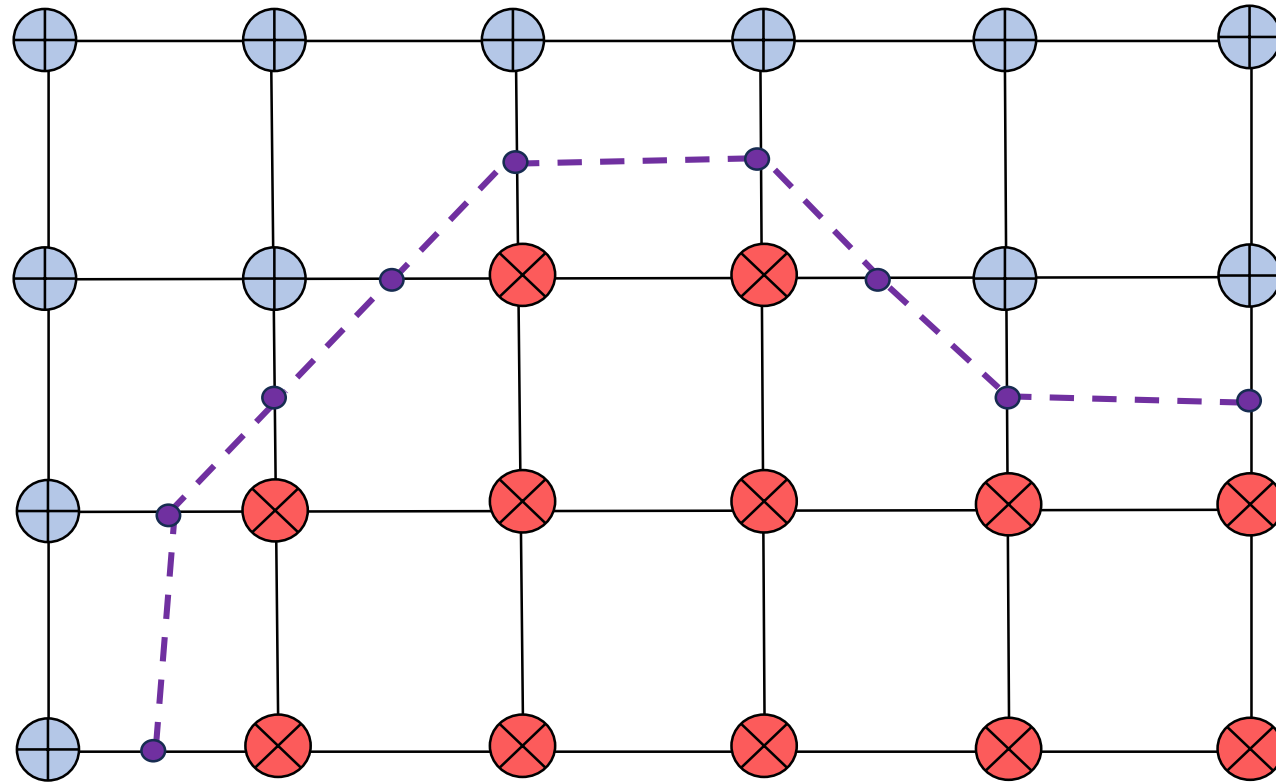
1. ...
2. ...
3. Calculate an index for the cube by comparing the eight density values at the cube vertices with the surface constant.
4. Using the index, look up the list of edges from a precalculated table.



# 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)

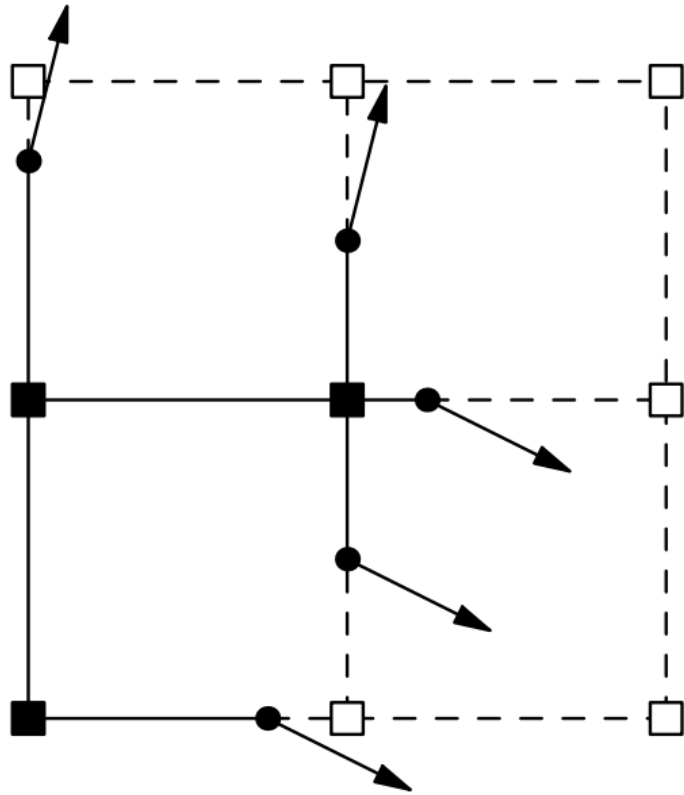
# Dual Contouring

- 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)

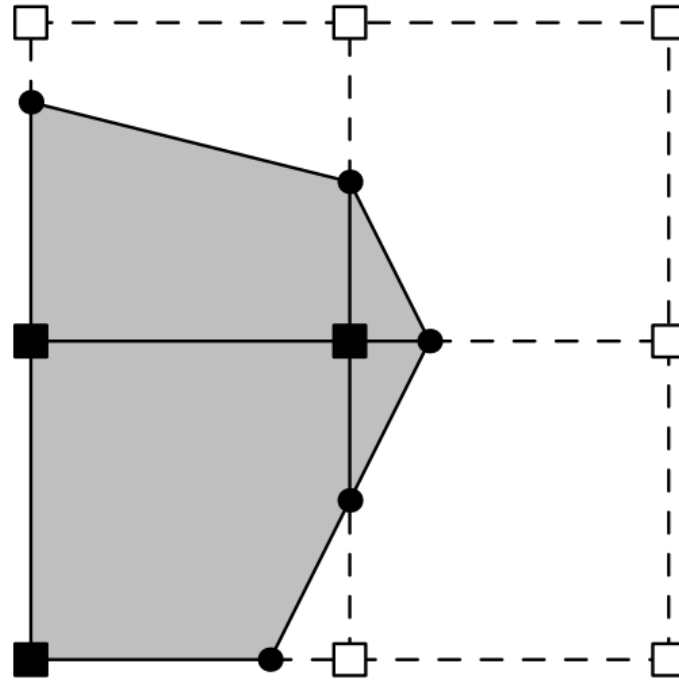
$$E[x] = \sum_i (n_i \cdot (x - p_i))^2$$

where  $p_i$  and  $n_i$  correspond to the intersections and normals of the contour with the edges of cube

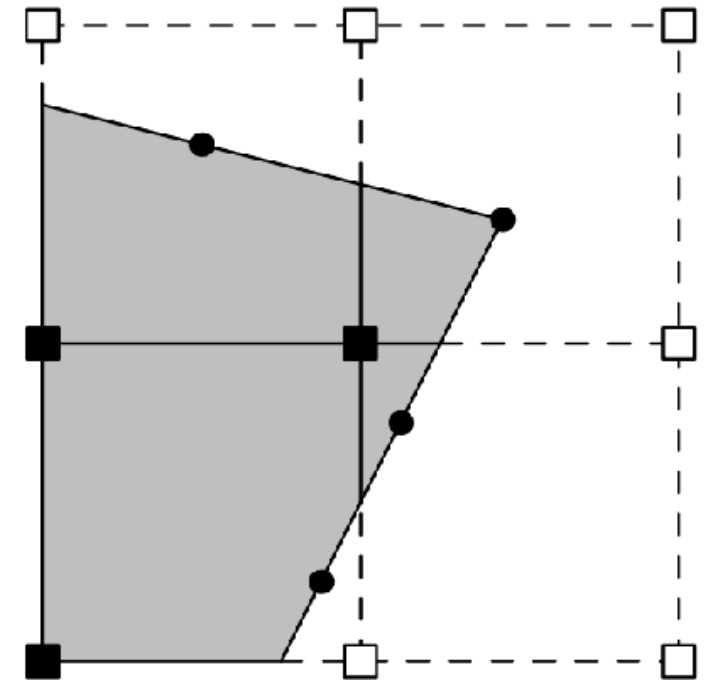
# Dual Contouring



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  $c$ .

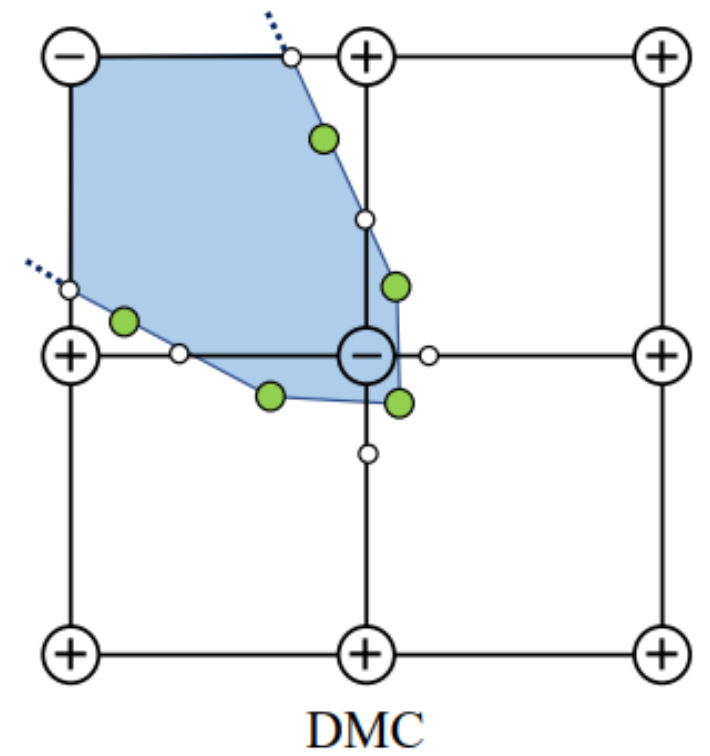
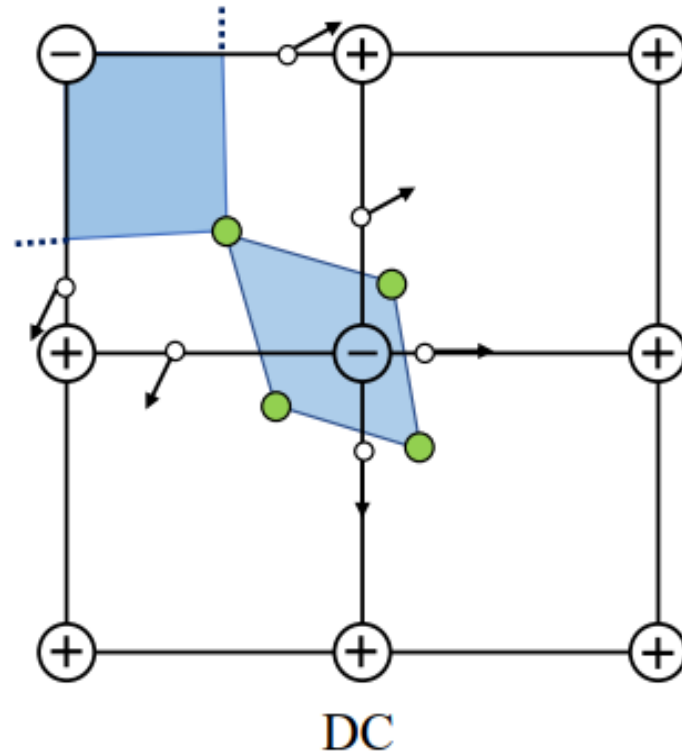
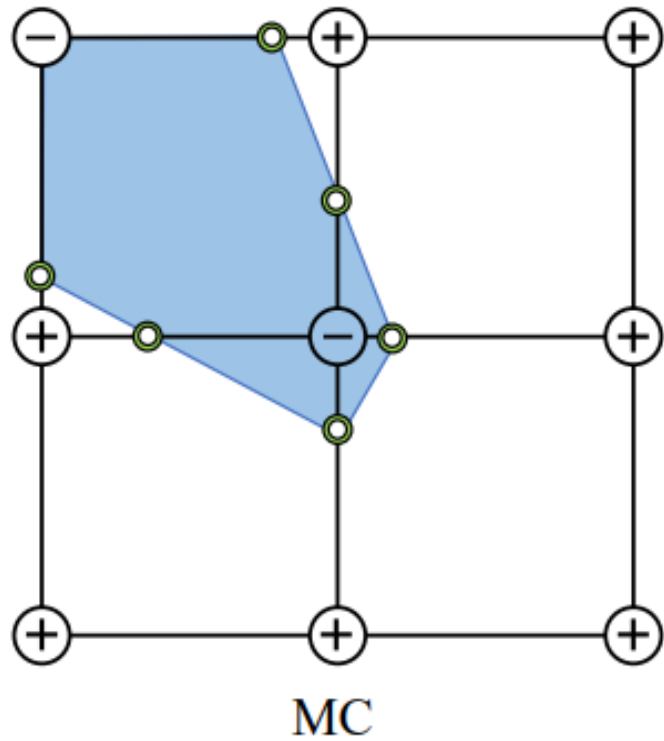
Tangent plane's equation:  $T_i(x, y, z) = \nabla f(x_i, y_i, z_i) \cdot ((x, y, z) - (x_i, y_i, z_i))$

DMC's QEF sums over all sample points yielding:

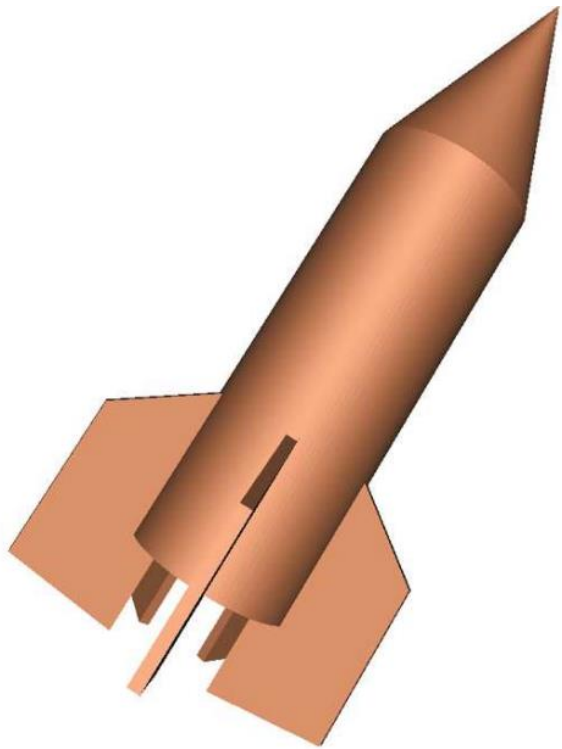
$$E(f(x, y, z), x, y, z) = \sum_i (f(x, y, z) - T_i(x, y, z))^2 / (1 + |\nabla f(x_i, y_i, z_i)|^2)$$

where the denominator normalizes the contribution of each tangent plane

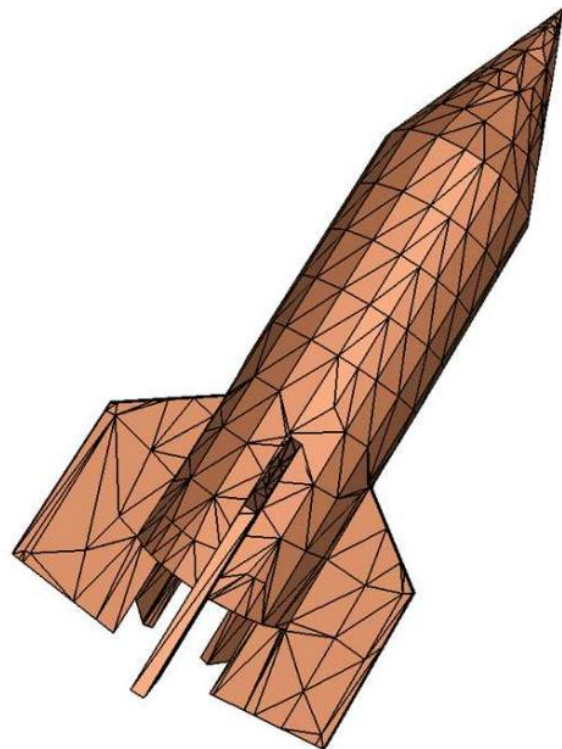
# Comparison between Marching Cubes & Dual Contouring & Dual Marching Cubes



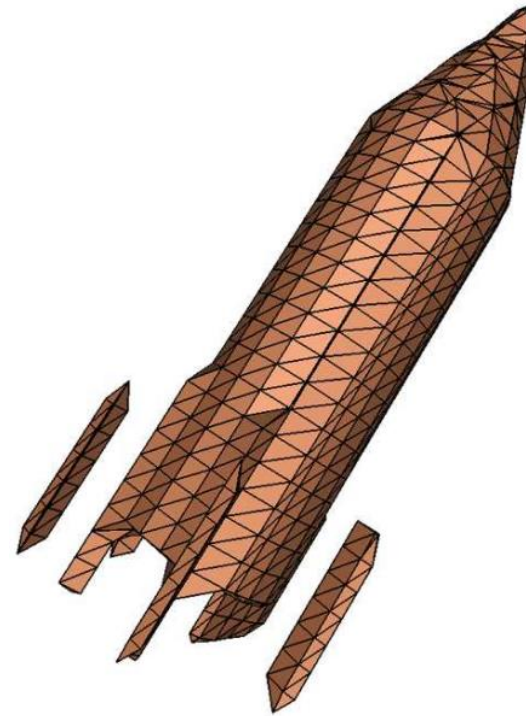
# Comparison between Marching Cubes & Dual Contouring & Dual Marching Cubes



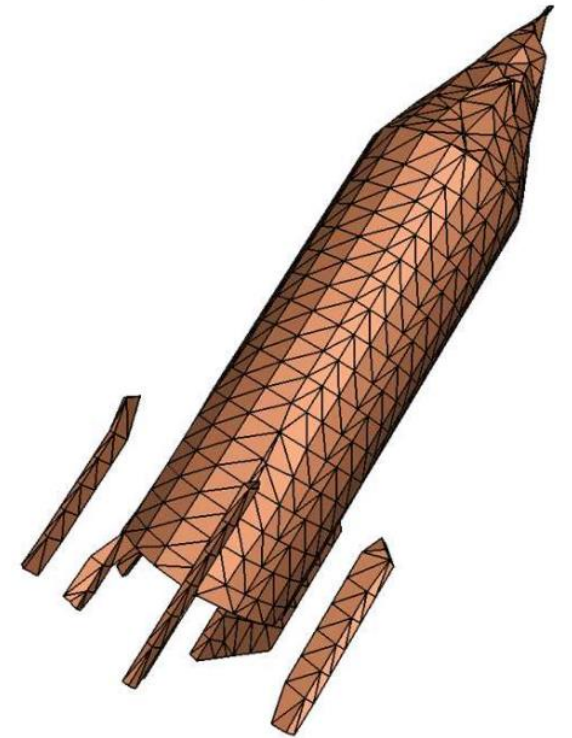
Ground Truth  
Model



Dual Marching  
Contour



Marching  
Contour



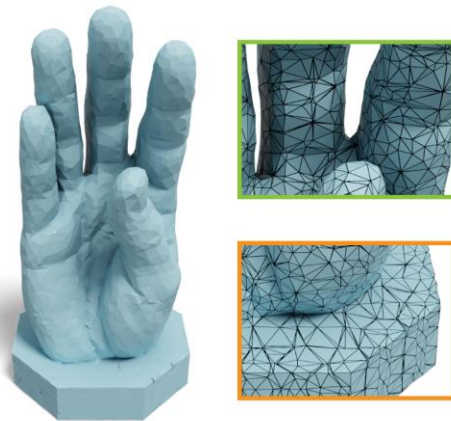
Dual  
Contour

# Flexi-Cubes: Flexible Iso-surface Extraction

Isosurfacing  
w/ gradients



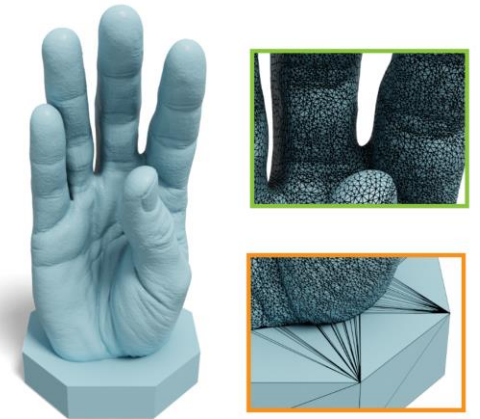
Marching Cubes 15k tris



DMTET 15k tris

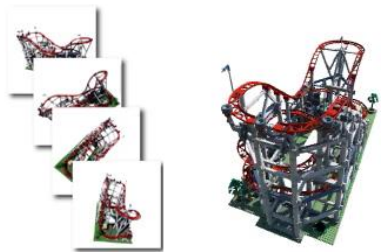


**FLEXICUBES** 13k tris



Reference 91k tris

Applications  
w/ FLEXICUBES



3D reconstruction from images



Generative 3D modeling



Animated 3D reconstruction



Tet-mesh physics simulation

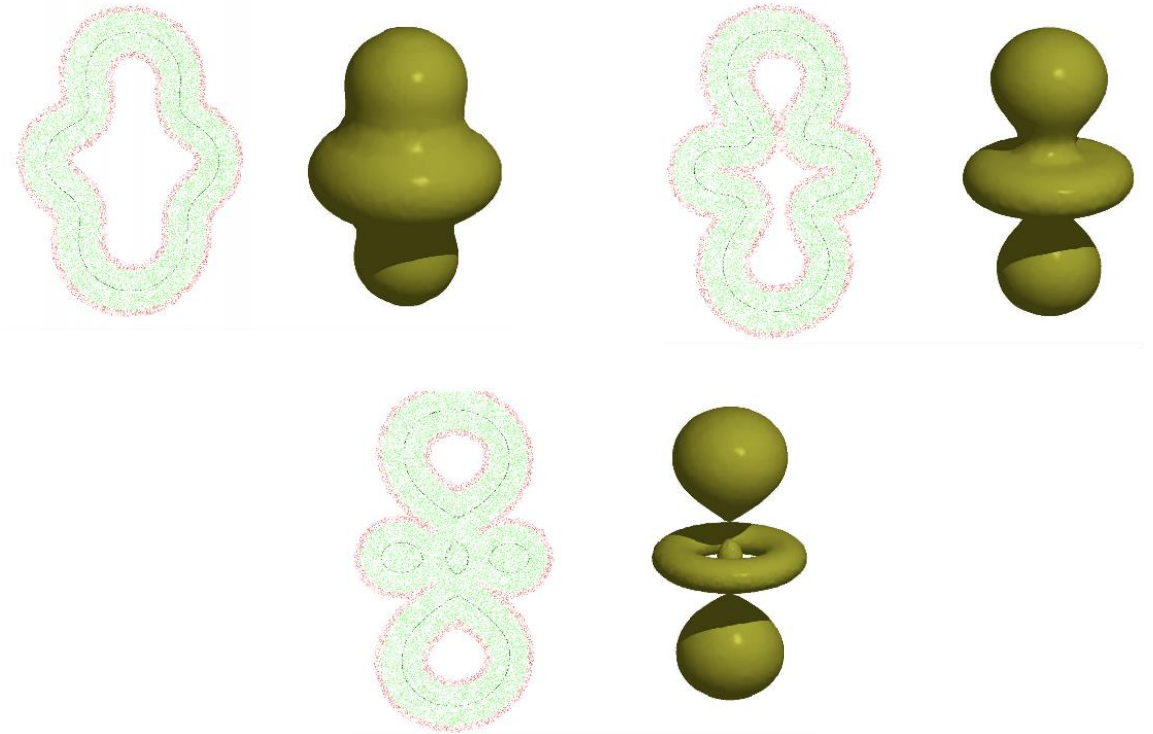


Developability

# Challenges 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:

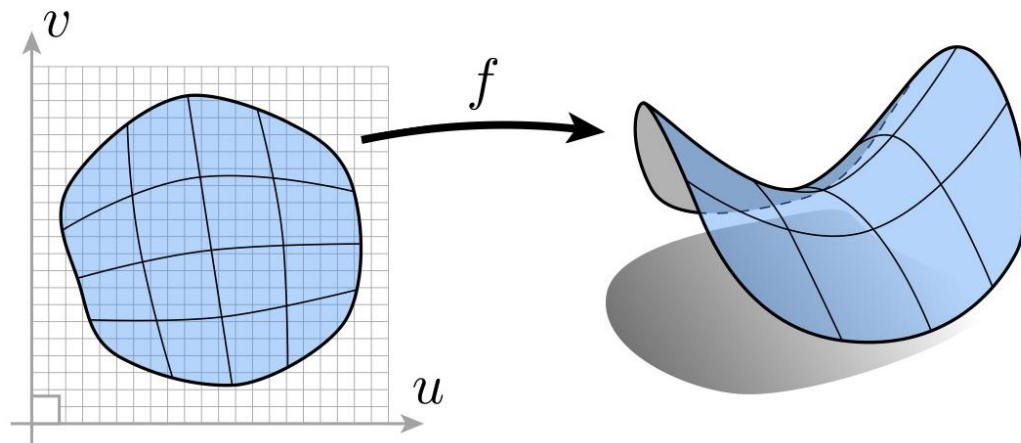
- description can be very compact (e.g., a polynomial)
- easy to determine if a point is in our shape (just plug it in!)
- other queries may also be easy (e.g., distance to surface)
- for simple shapes, exact description/no sampling error
- easy to handle changes in topology (e.g., fluid)

## Cons:

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

# "Explicit" Representations of Geometry

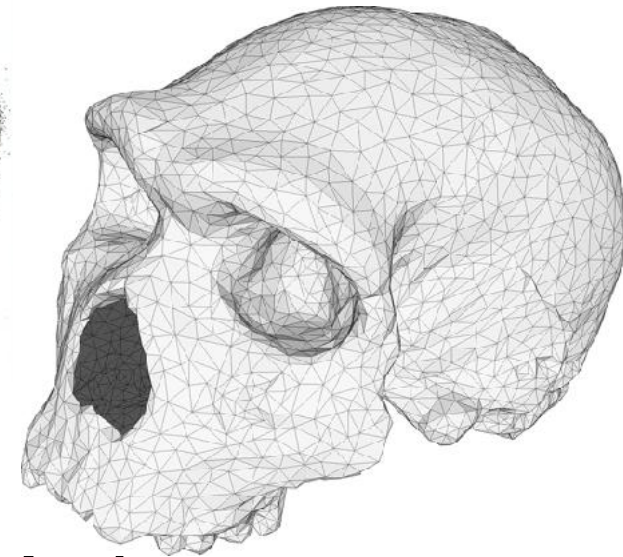
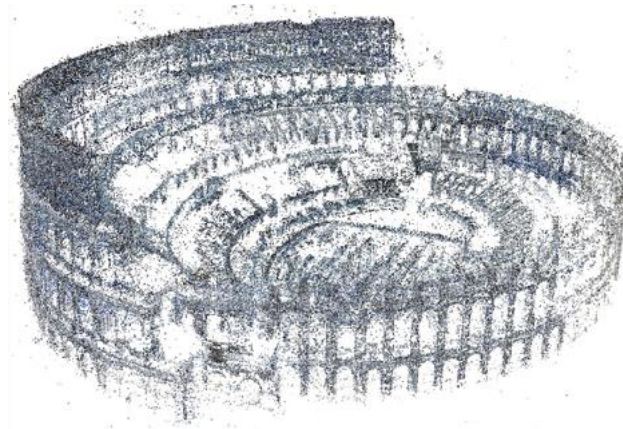
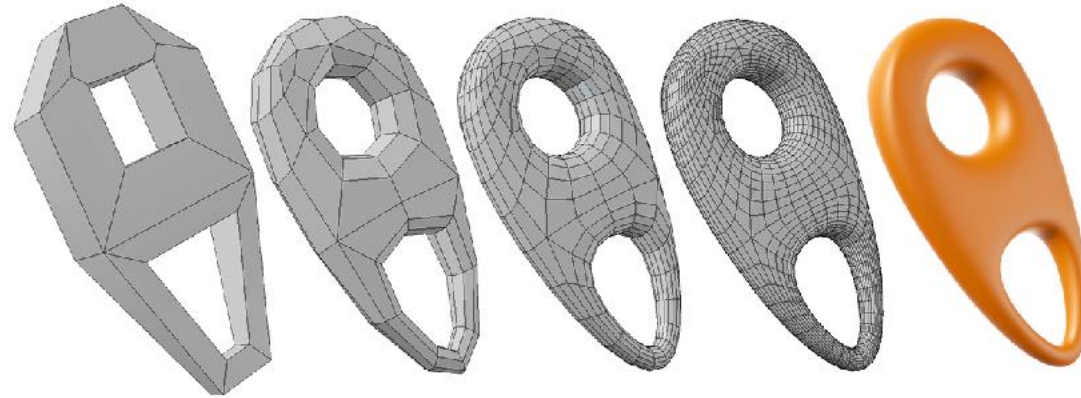
- All points are given directly
- E.g., points on sphere are  $(\cos(u) \sin(v), \sin(u) \sin(v), \cos(v))$ ,  
for  $0 \leq u < 2\pi$  and  $0 \leq v \leq \pi$
- More generally:  $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



- (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:**

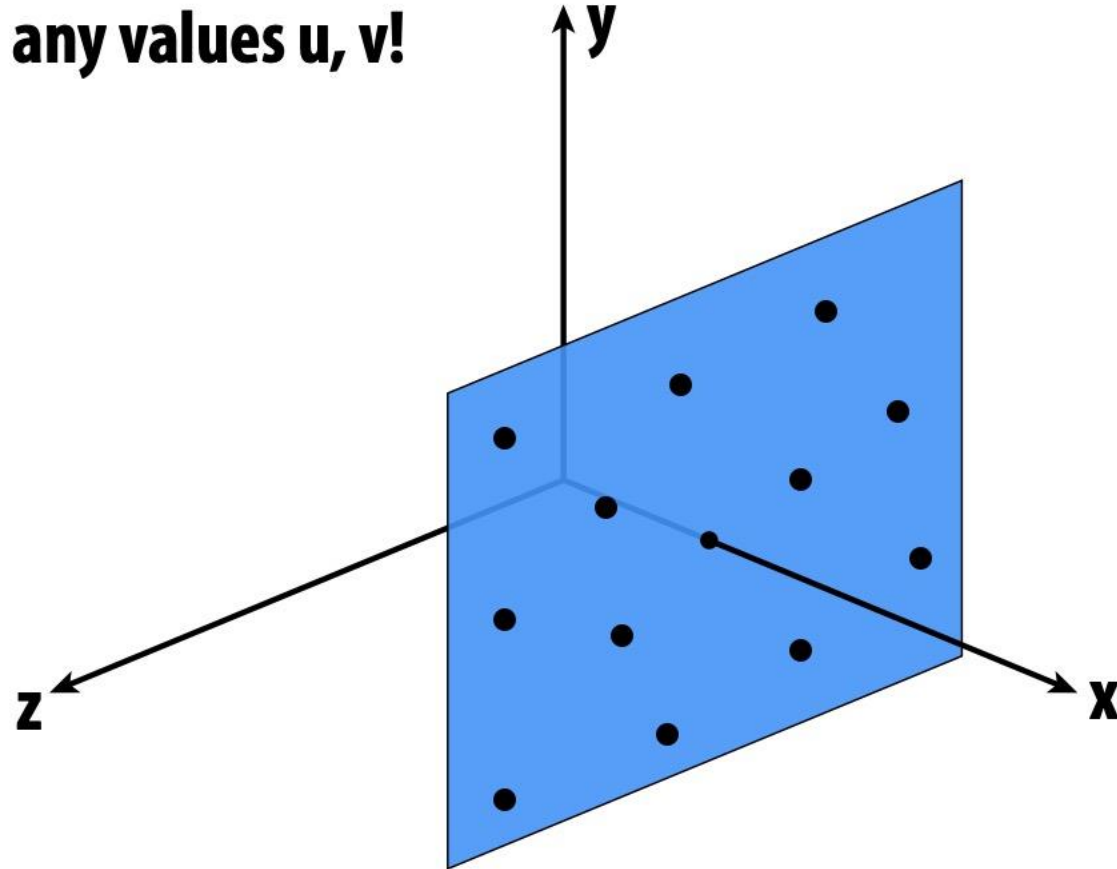
**I'll give you an explicit surface.**

**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  $f(u,v)$ .**

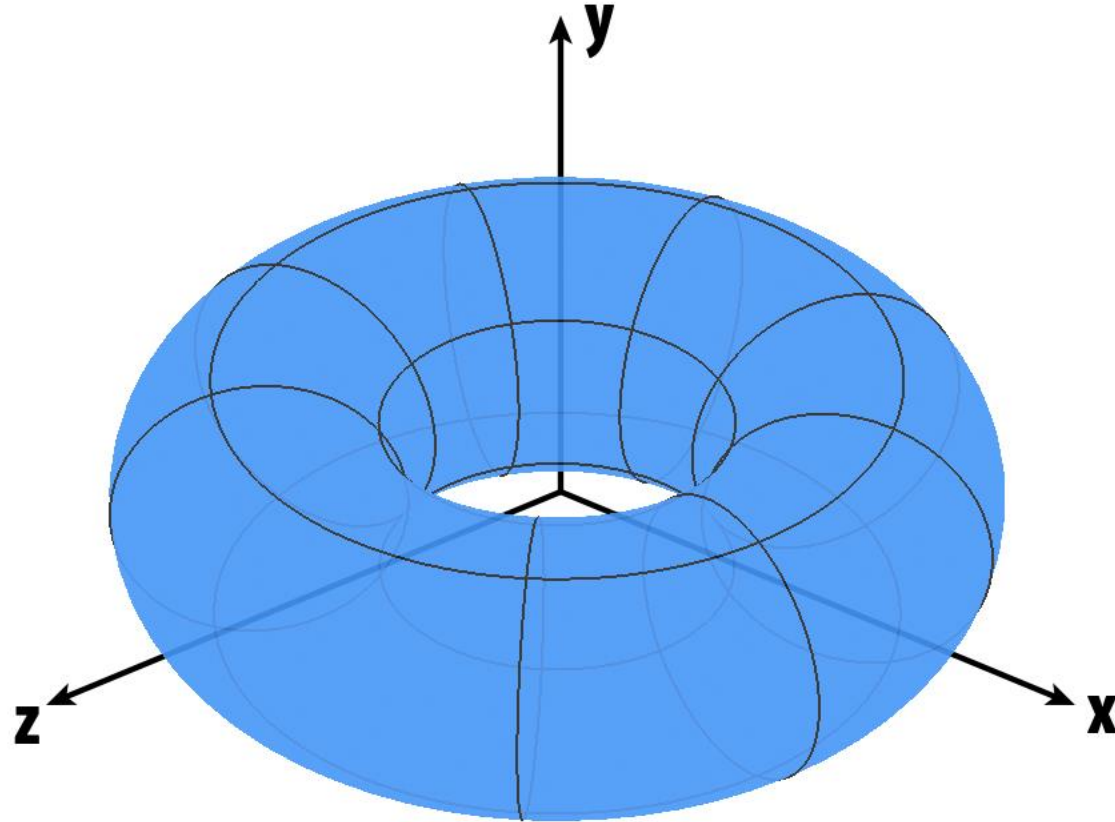
**I want to see if a point is *inside* it.**

# 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!**



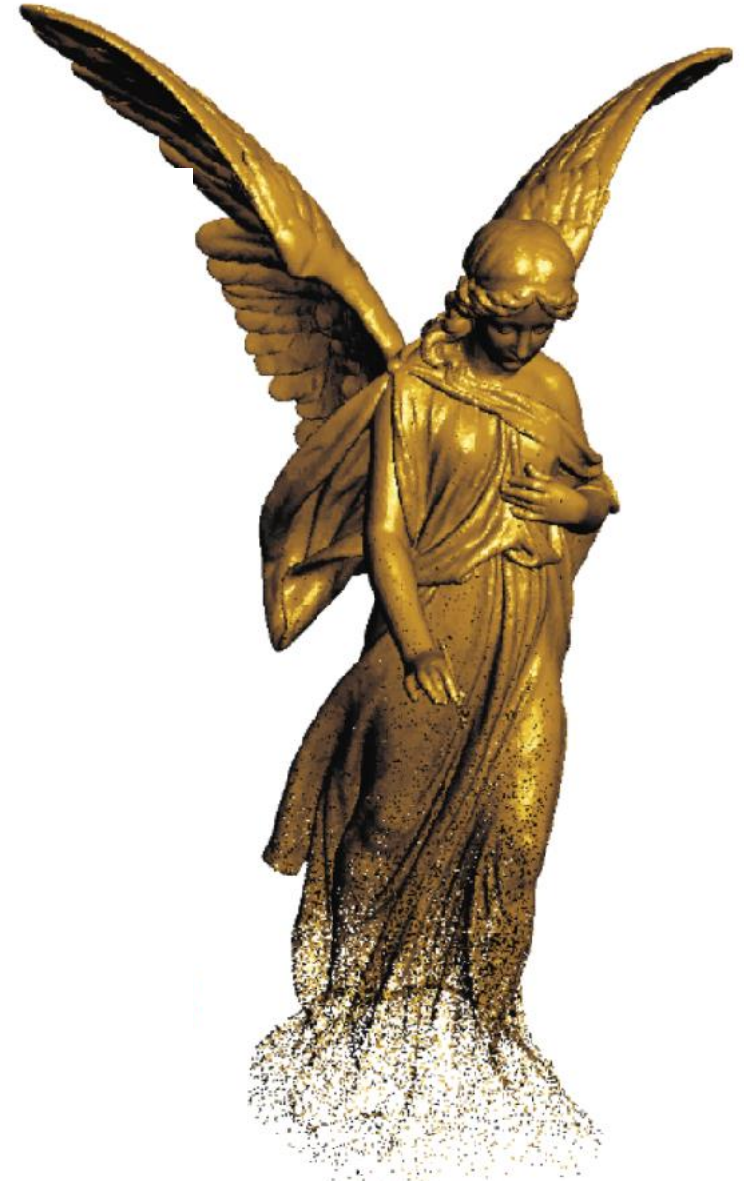
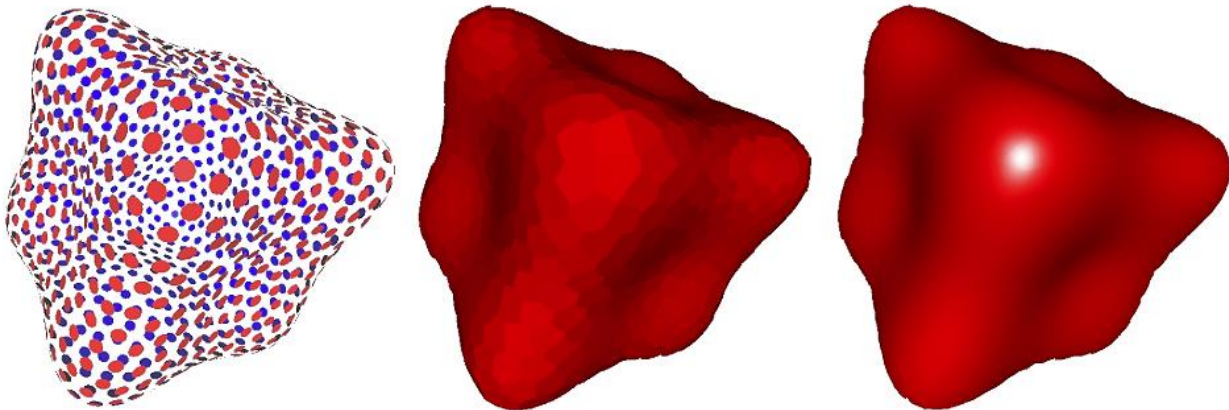
**Explicit surfaces make other tasks hard (like inside/outside tests).**

## **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 ( $\gg 1$  point/pixel)
- Hard to interpolate under-sampled regions
- Hard to do processing / simulation/...

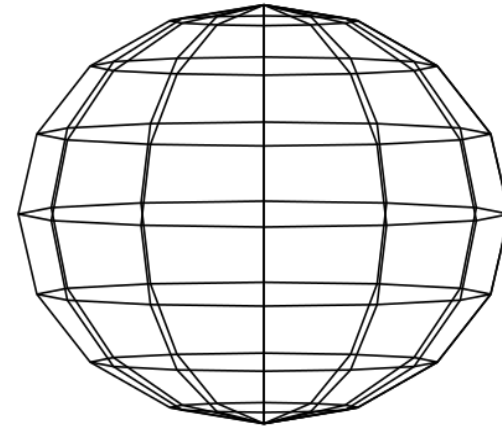
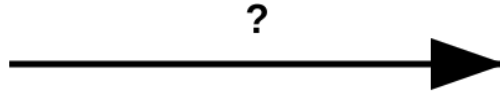


# From Points to Surfaces

*Why do we need surface reconstruction?*



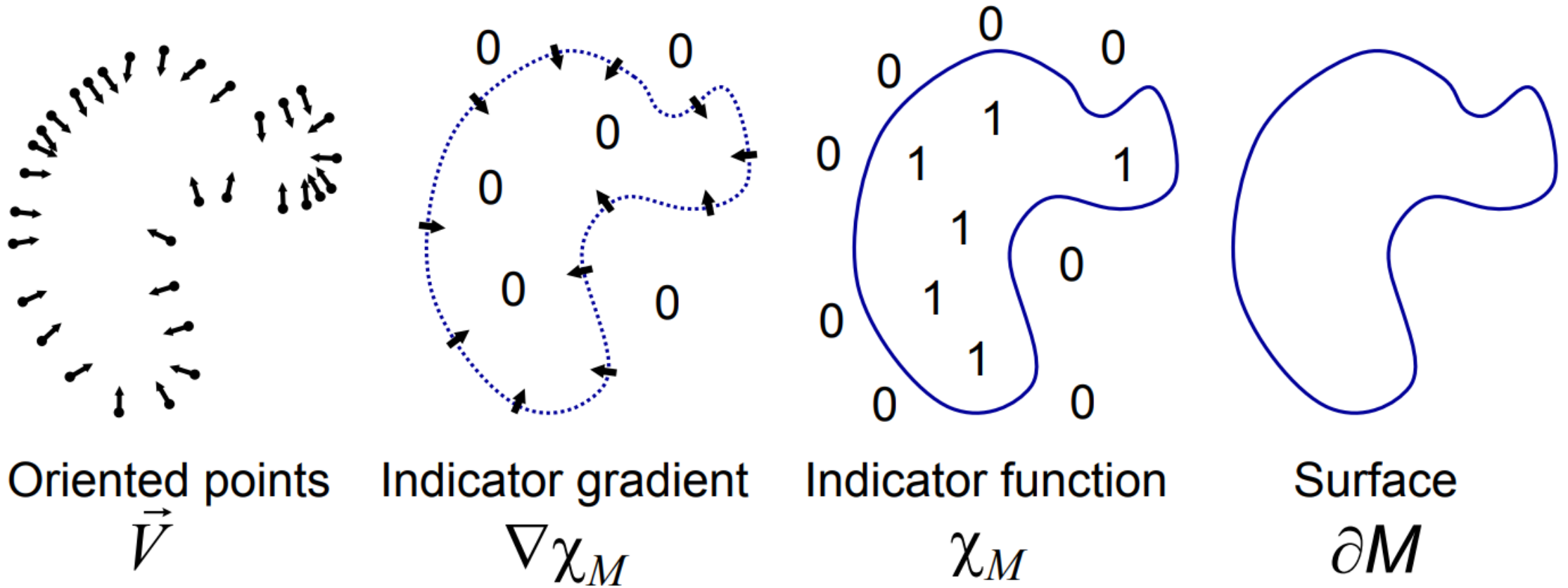
Oriented Point Cloud (P, N)



Watertight Surface Mesh

- Point clouds are discrete, unordered — no connectivity or topology
- Downstream tasks (rendering, simulation, 3D printing) require a continuous surface
- Goal: a method that is global, robust to noise, and produces watertight meshes
- **Key idea: recast surface reconstruction as solving a Poisson equation**

# Poisson Surface Reconstruction



# Poisson Surface Reconstruction

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

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

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

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

# Poisson Surface Reconstruction

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

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

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

# Poisson Surface Reconstruction

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

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

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. In Screened PSR (Kazhdan et.al, 2013) authors use

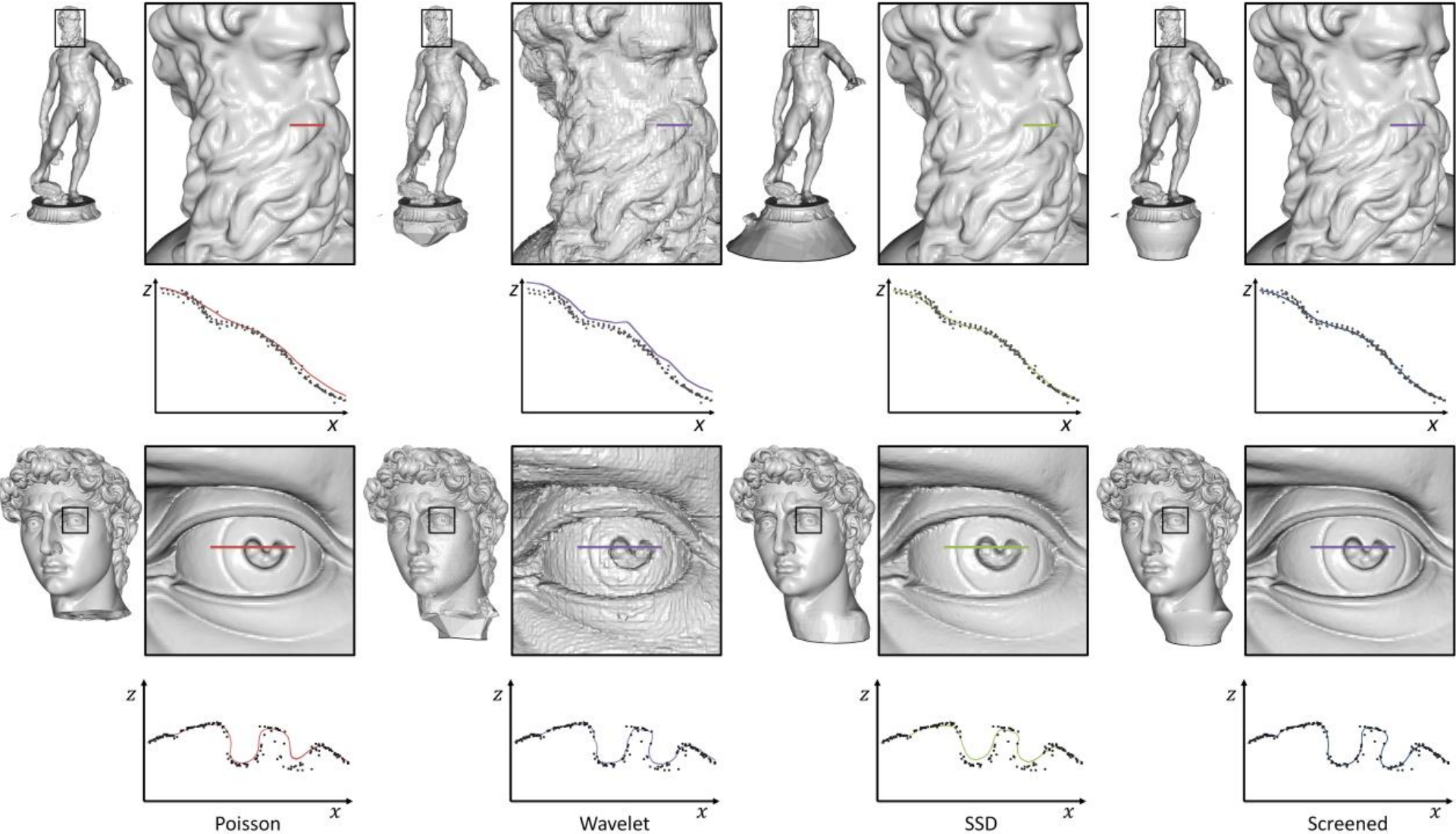
$$f = \operatorname{argmin}_g \int_B \|V(x) - \nabla g(x)\|_2 dx + \alpha * \sum_i g(p_i)^2$$

# (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

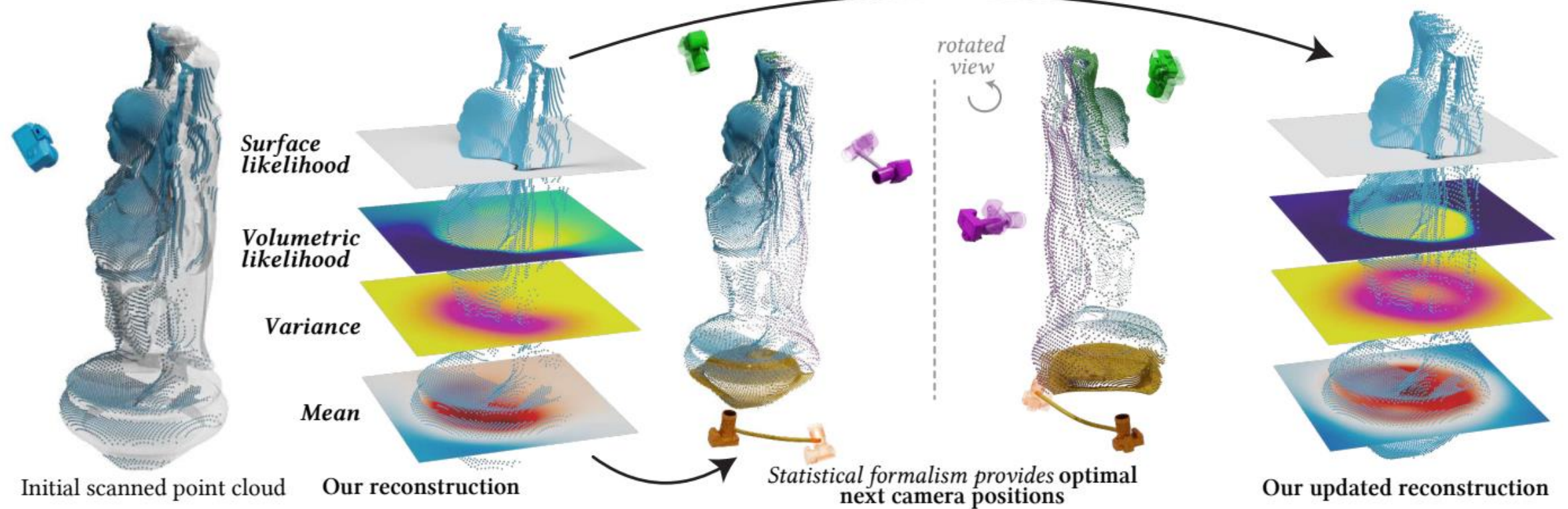


# Neural Stochastic Screened Poisson Surface Reconstruction

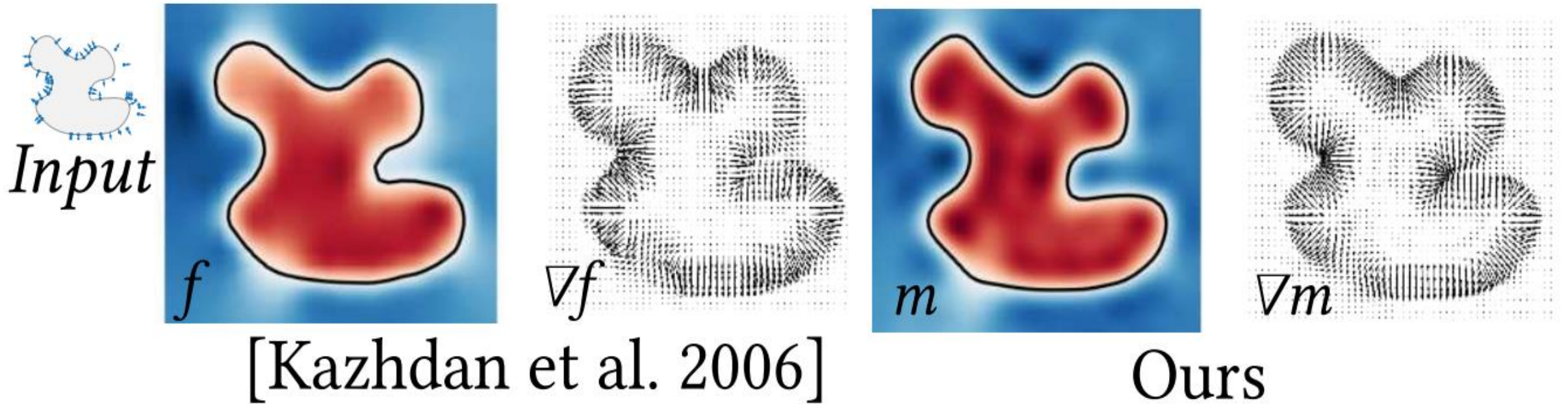
- Take a step even further and models the posterior multivariate Gaussian distributions representing the likelihood of all possible reconstructions for the given Point Set with normals
- Proposes the parametrization of mean and covariance functions using neural networks, optimizing them through gradient-based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

# Neural Stochastic Screened Poisson Surface Reconstruction

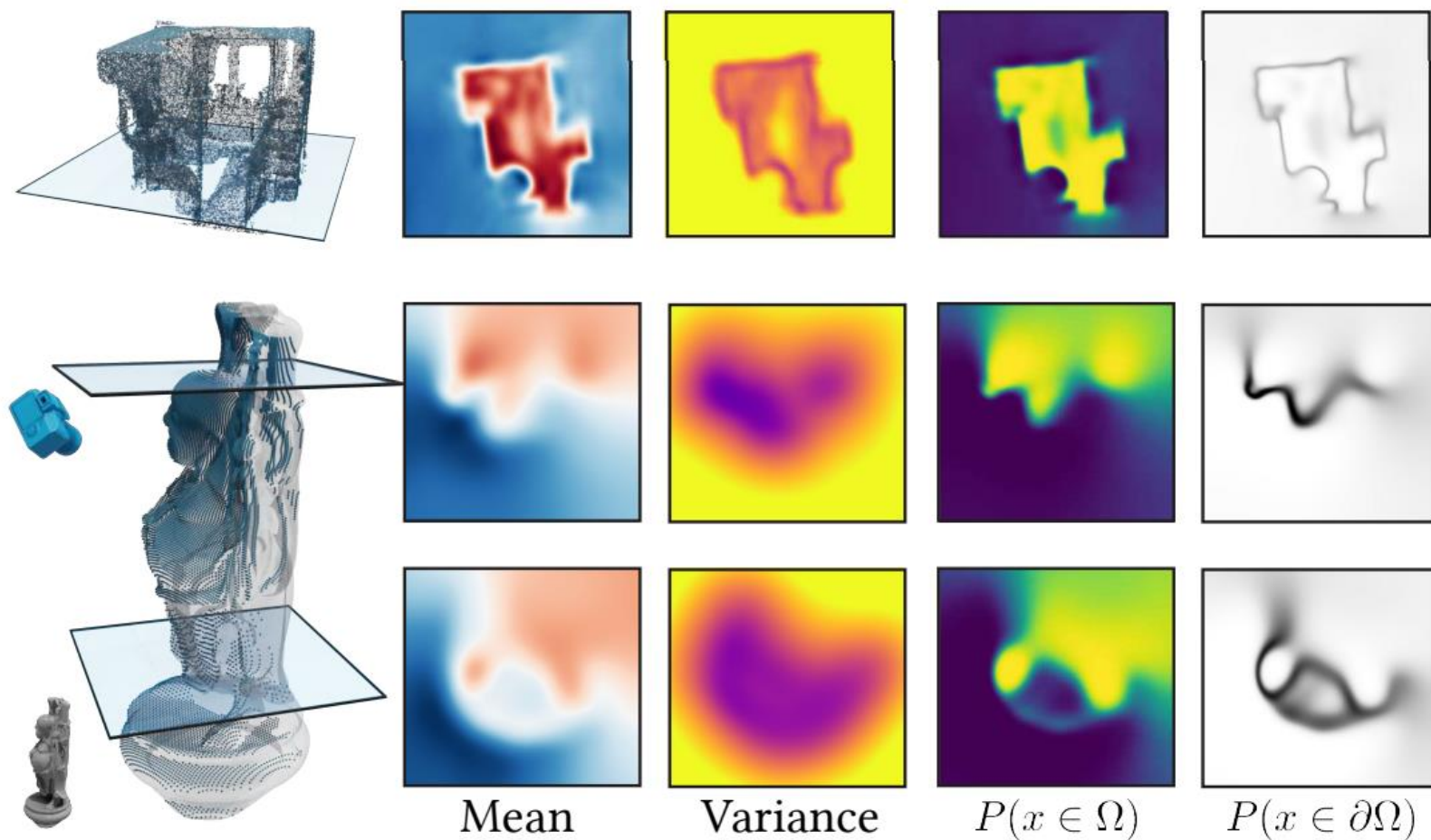
*Fine-tune our pretrained model to obtain new reconstruction*



# Neural Stochastic Screened Poisson Surface Reconstruction

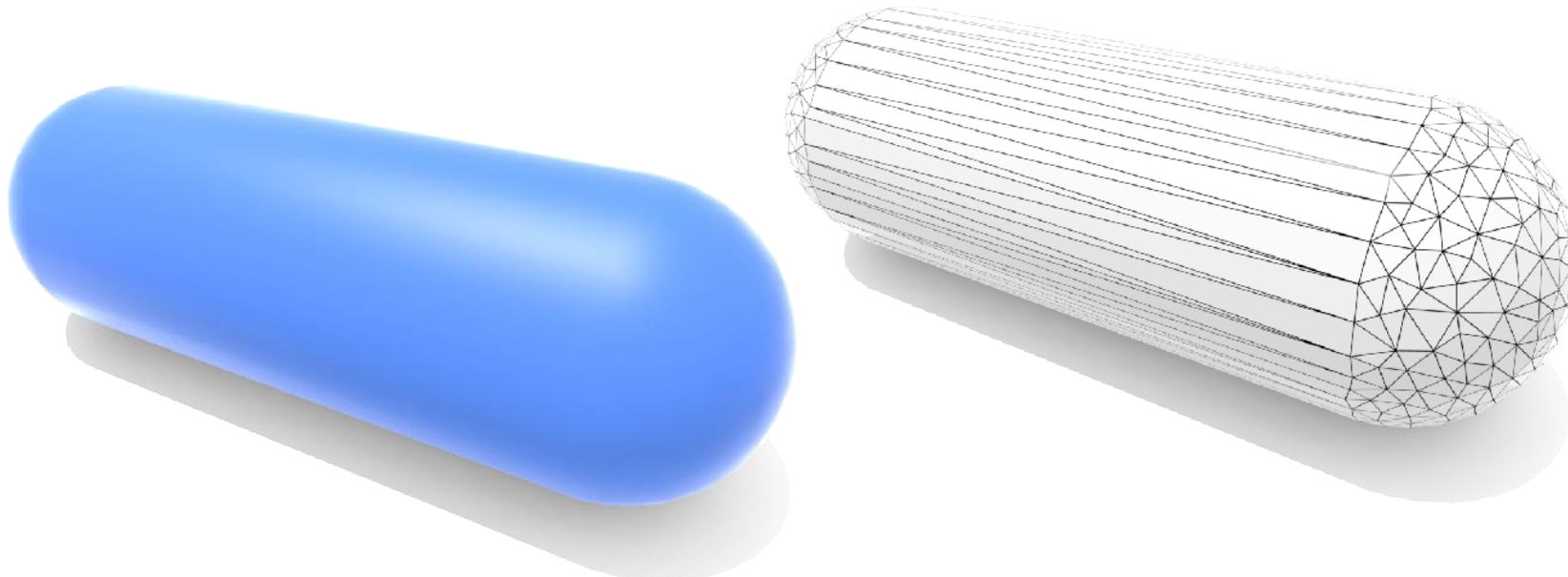


# Neural Stochastic Screened PSR



# Polygon Mesh

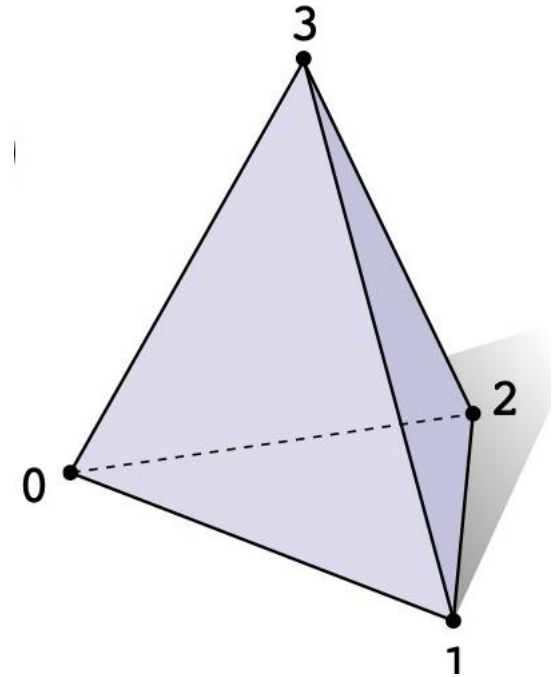
- Store vertices and polygons (most often triangles or quads)
- Easier to do processing/simulation, adaptive sampling
- More complicated data structures
- Irregular neighbourhoods



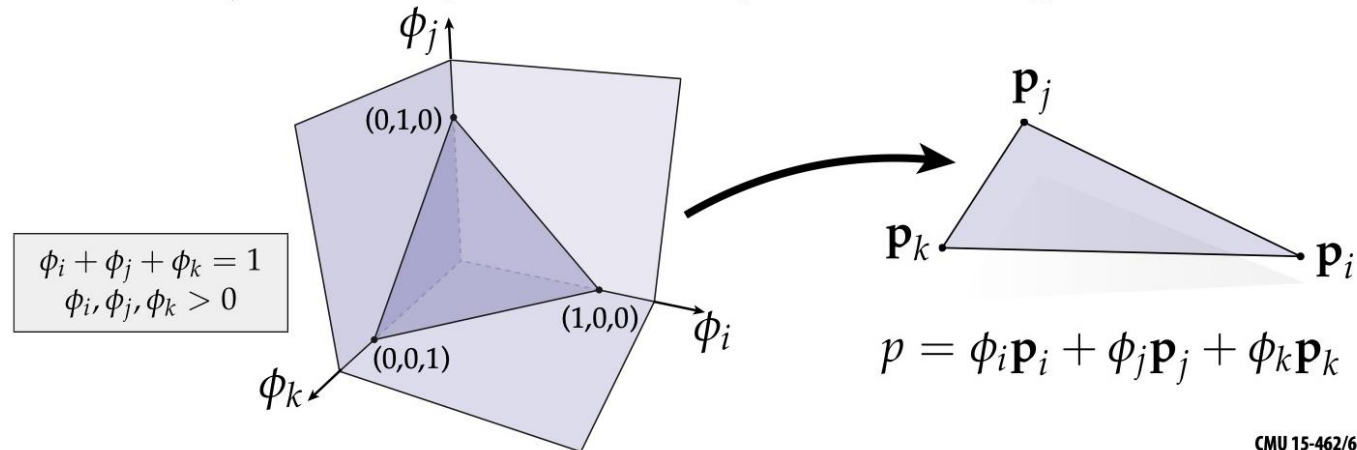
# Triangle Mesh (Explicit)

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

	VERTICES			TRIANGLES		
	x	y	z	i	j	k
0:	-1	-1	-1	0	2	1
1:	1	-1	1	0	3	2
2:	1	1	-1	3	0	1
3:	-1	1	1	3	1	2



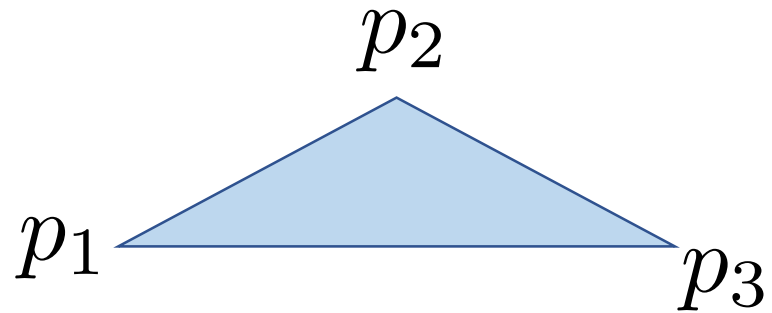
- Use barycentric interpolation to define points inside triangles:



# 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  $\mathbf{n}(u, v) = f_u(u, v) \times f_v(u, v)$

- Distance and closest point. Find surface point  $f(u, v)$  such that:

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

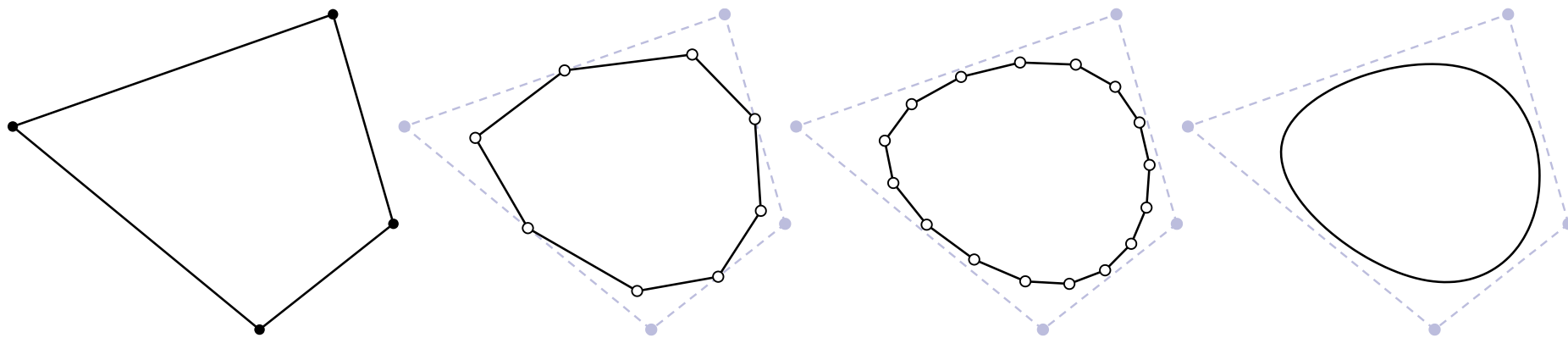
- Mesh

- 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

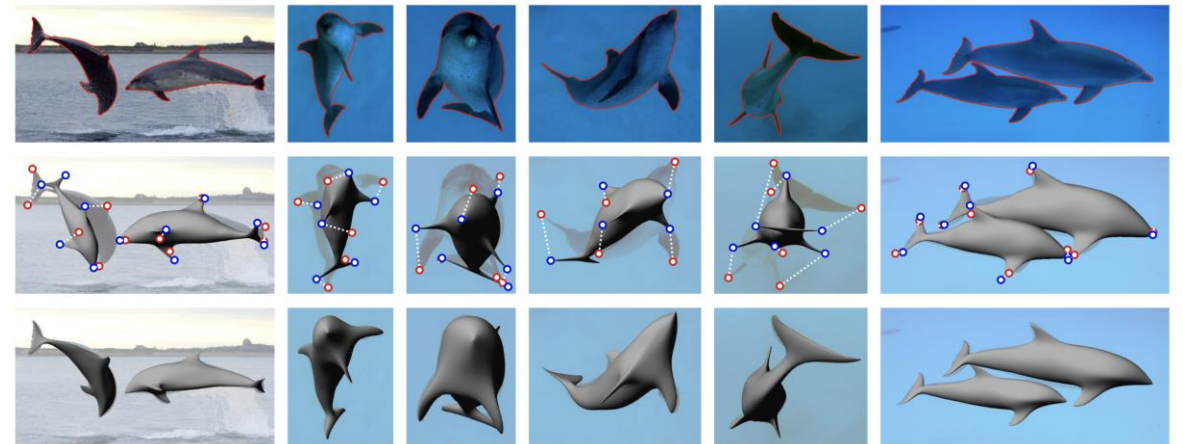
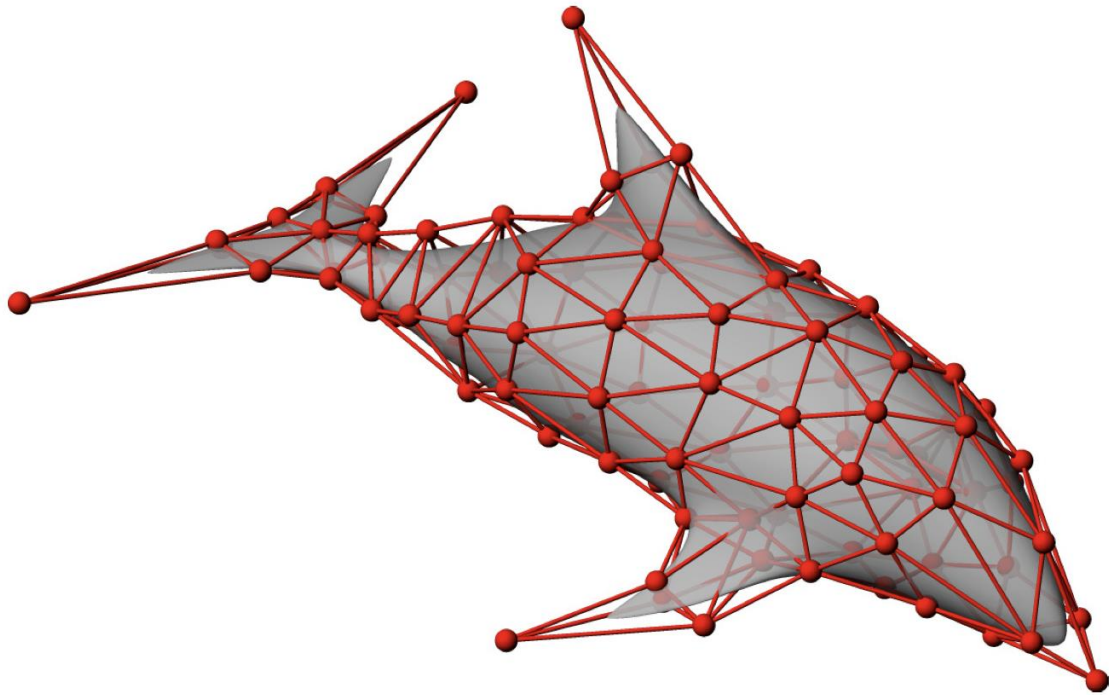
# Subdivision

- 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!



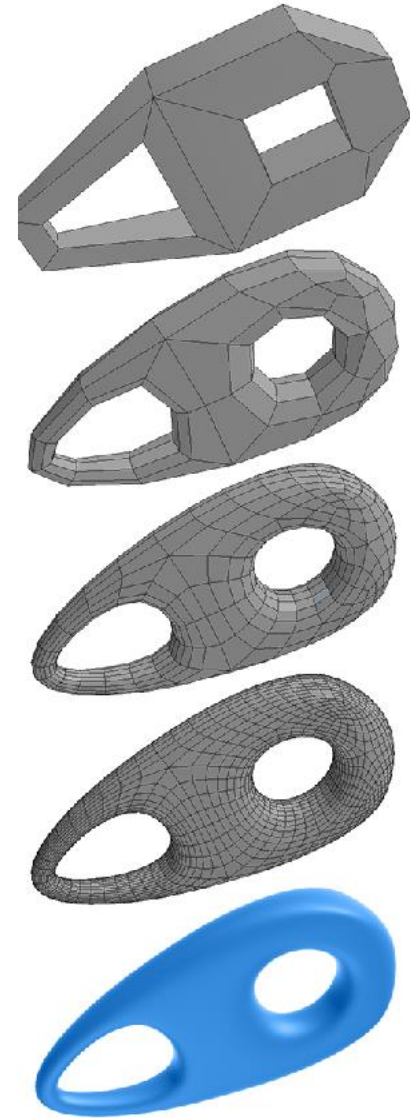
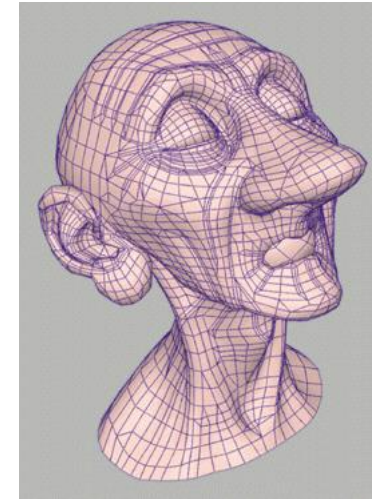
**Q: Is subdivision an explicit or implicit representation?**

# Subdivision Surfaces in Computer Vision



# 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")



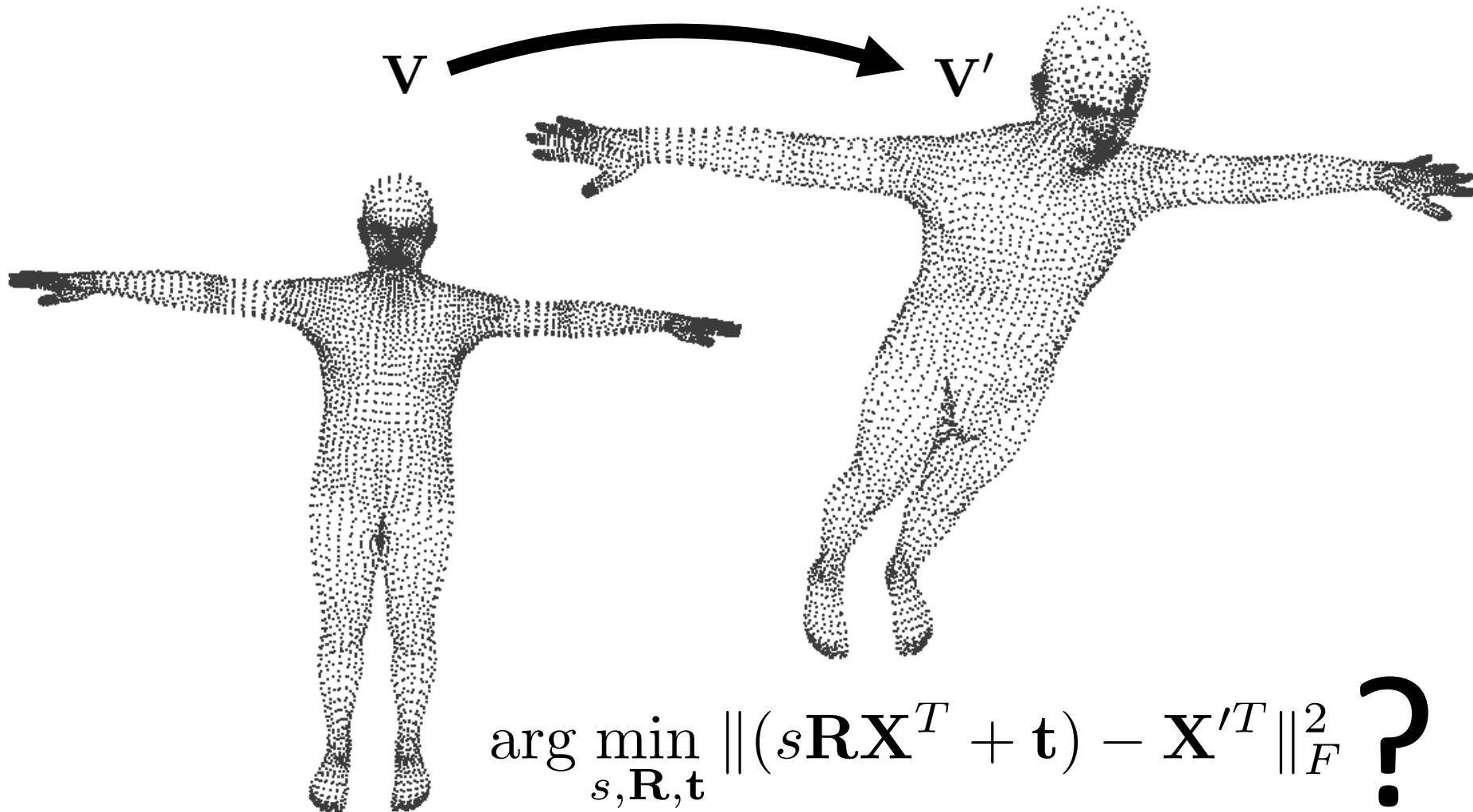
see: de Rose et al, "Subdivision Surfaces in Character Animation"

# Slide credits and further reading

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

# Aligning Pointclouds to a Common Frame

# Procrustes alignment problem definition



# 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:

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

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

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

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

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

warning: this is not the vertex matrix!

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

# Procrustes alignment steps

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

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

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

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

**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

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

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

# Procrustes derivation: translation

We remove the elements that do not depend on the translation and solve for  $\mathbf{t}$ .

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

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

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

Compute the centroid of the point clouds

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

So given  $s$  and  $R$ , we can compute the translation  $\mathbf{t}$

# 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

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

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

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

$$= \arg \min_{\mathbf{R}} \langle \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T, \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T \rangle_F$$

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

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

$$= \arg \max_{\mathbf{R}} \langle \mathbf{R}, \bar{\mathbf{Y}}^T \bar{\mathbf{X}} \rangle_F$$

Optimal rotation does not depend on scale

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  $\Sigma$  ?

# 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  $\Sigma$  ?

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 pointsclds

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

# Procrustes derivation: scale

Optimize **scale** given the rotation

$$s = \arg \min_s E = \arg \min_s \left( \sum_i s^2 \bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i + \bar{\mathbf{y}}_i^\top \bar{\mathbf{y}}_i - 2s \bar{\mathbf{x}}_i^\top \mathbf{R}^\top \bar{\mathbf{y}}_i \right)$$

$$= \arg \min_s \left( s^2 \sum_i (\bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i) - 2s \sum_i (\bar{\mathbf{x}}_i^\top \mathbf{R}^\top \bar{\mathbf{y}}_i) \right)$$

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

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

We used:

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

# Slide credits and further reading

# Hands-On AI Based 3D Vision

## Summer Semester 26

Lecture 5\_2 – Procrustes Alignment

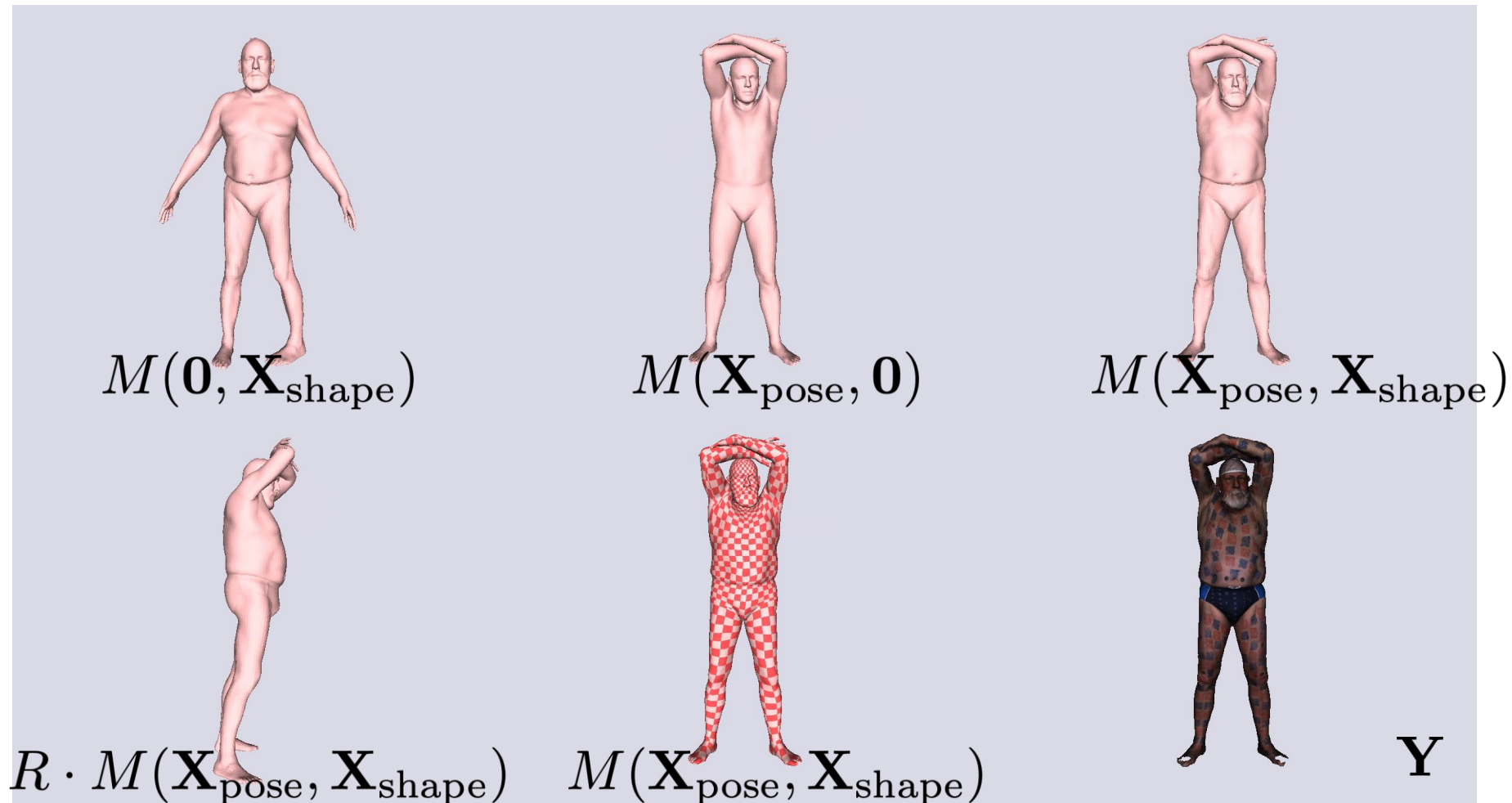
Prof. Dr.-Ing. Gerard Pons-Moll

University of Tübingen / MPI-Informatics

EBERHARD KARLS  
UNIVERSITÄT  
TÜBINGEN

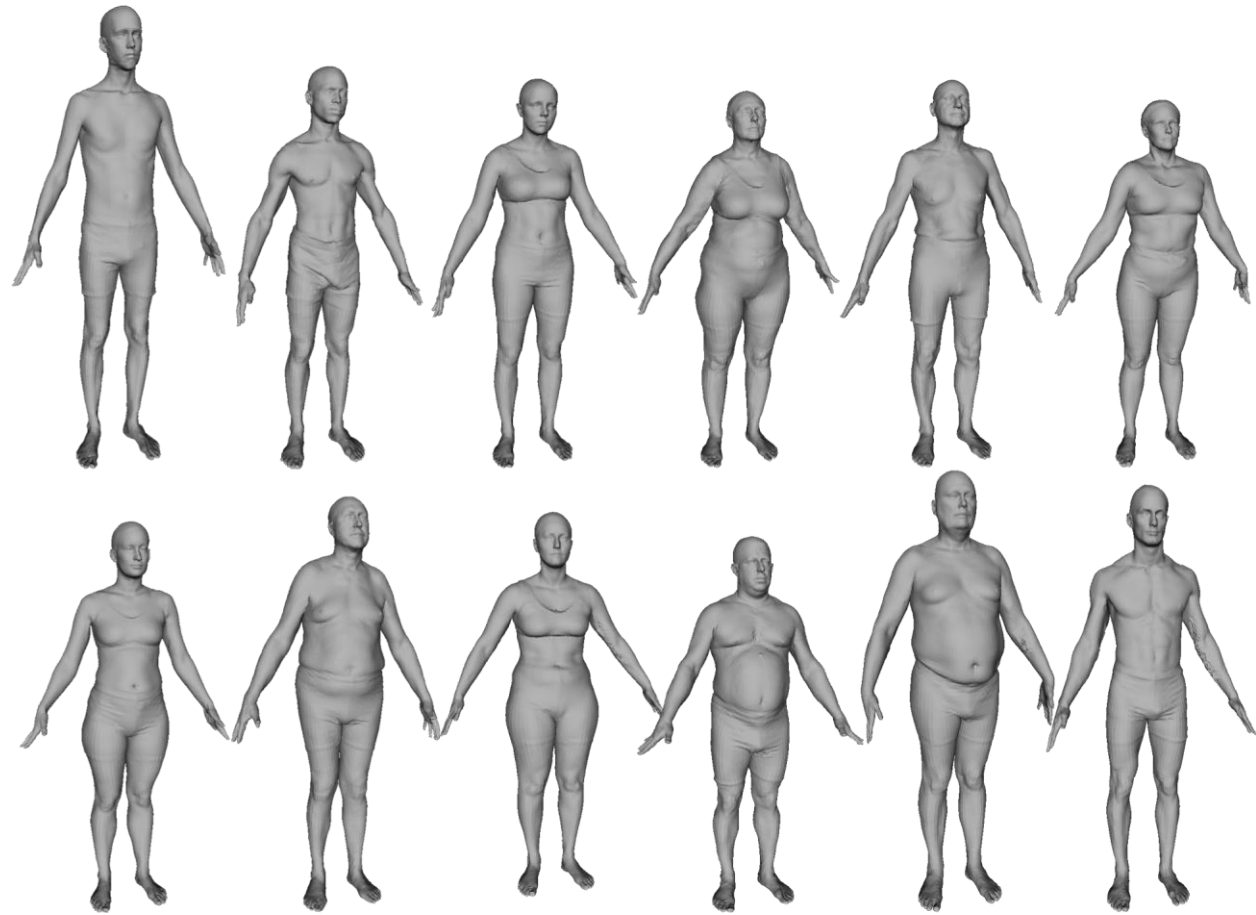


# Goal: learn a model of pose and shape



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

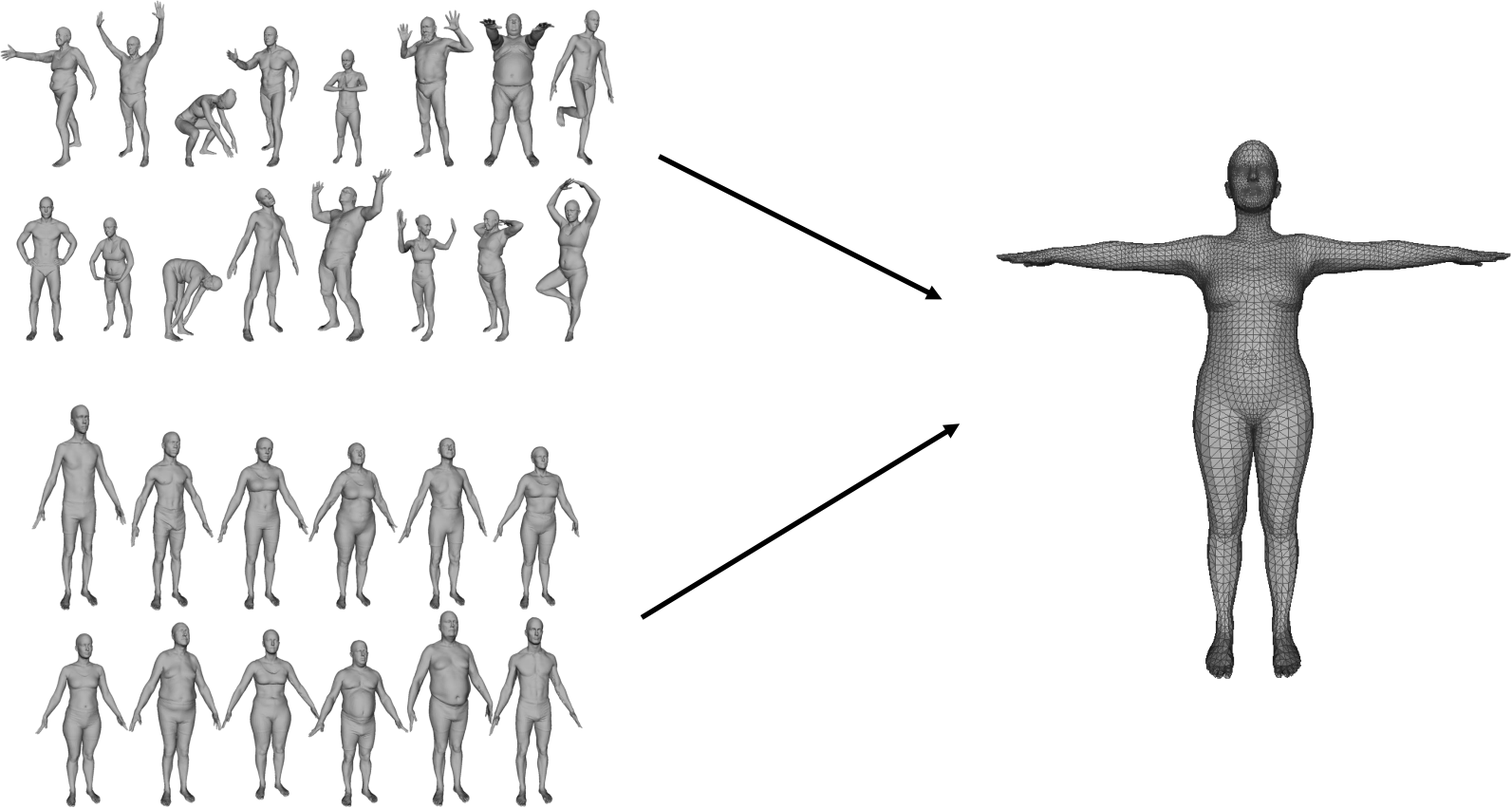
# Scan a Lot of People



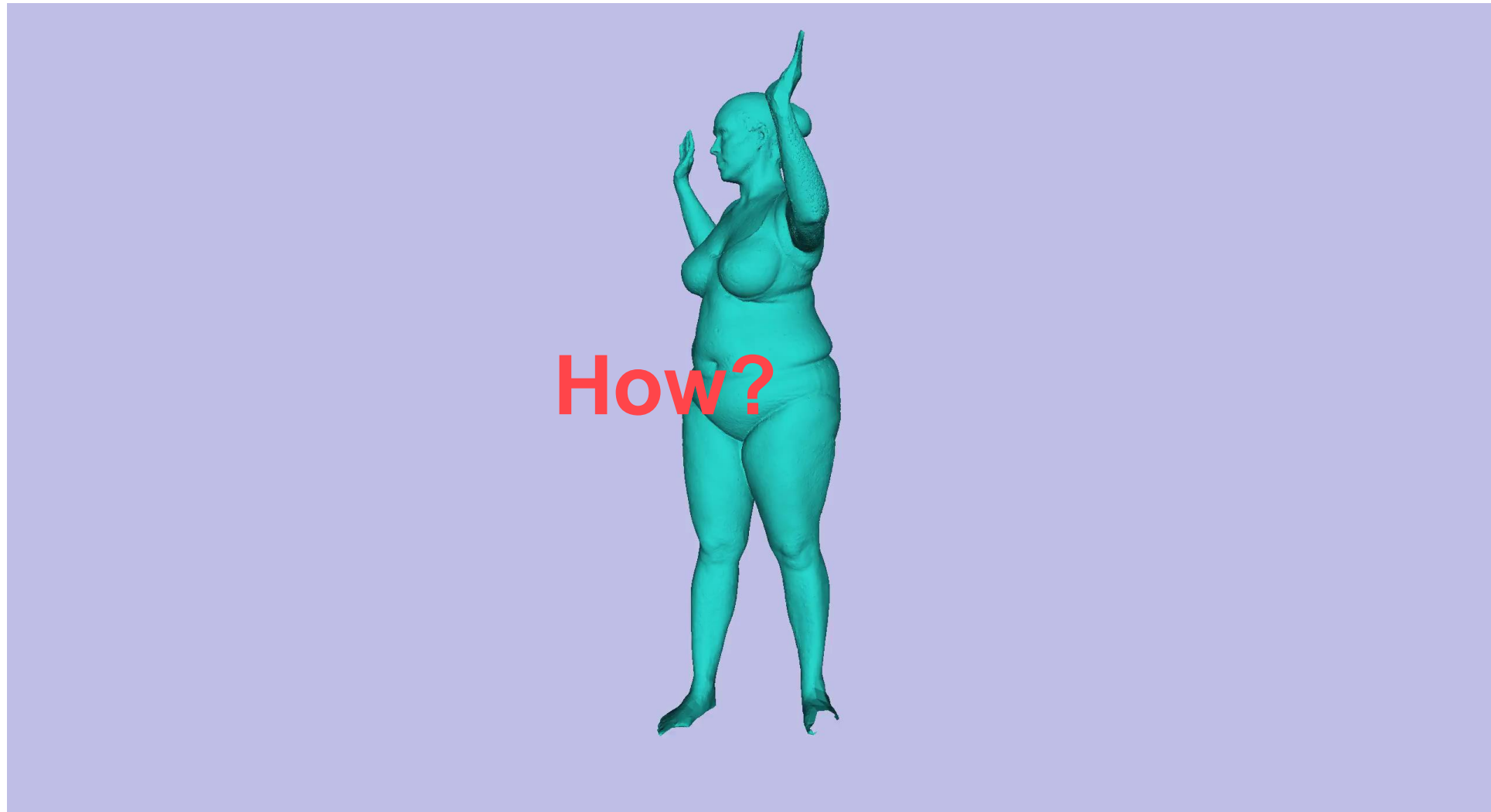
# In lots of poses



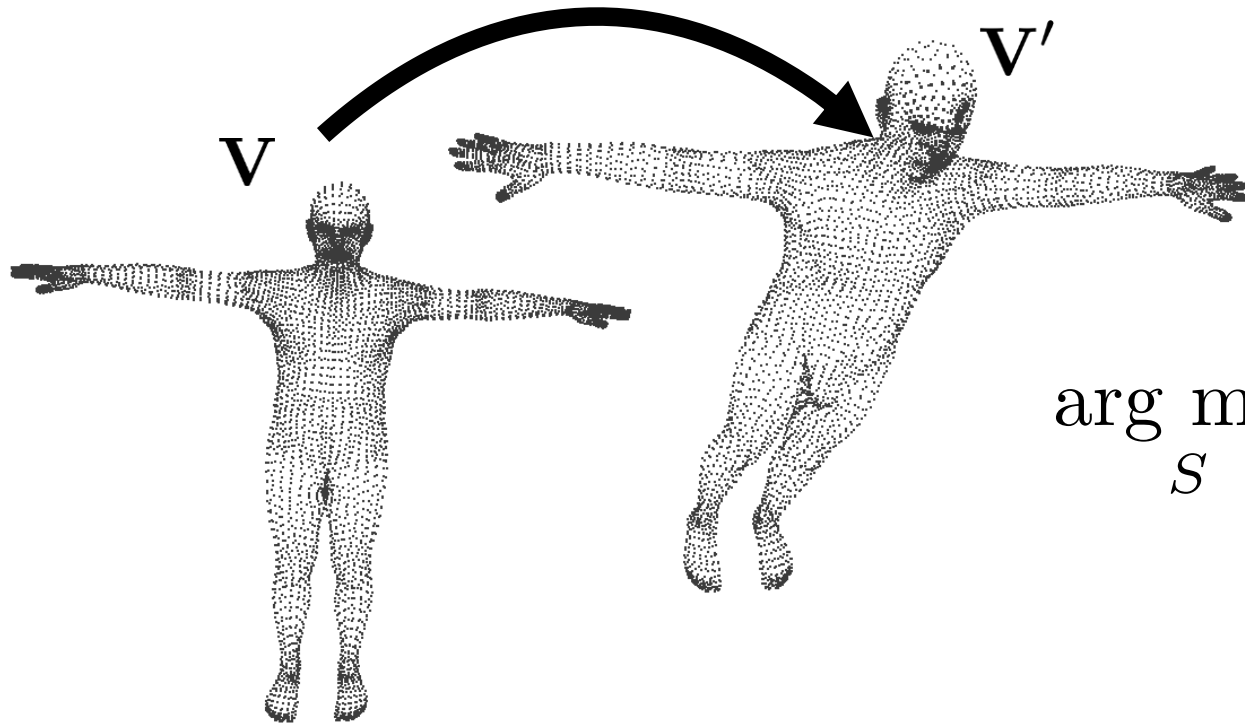
# To learn a model we need correspondence



# Non-rigid Articulated Registration



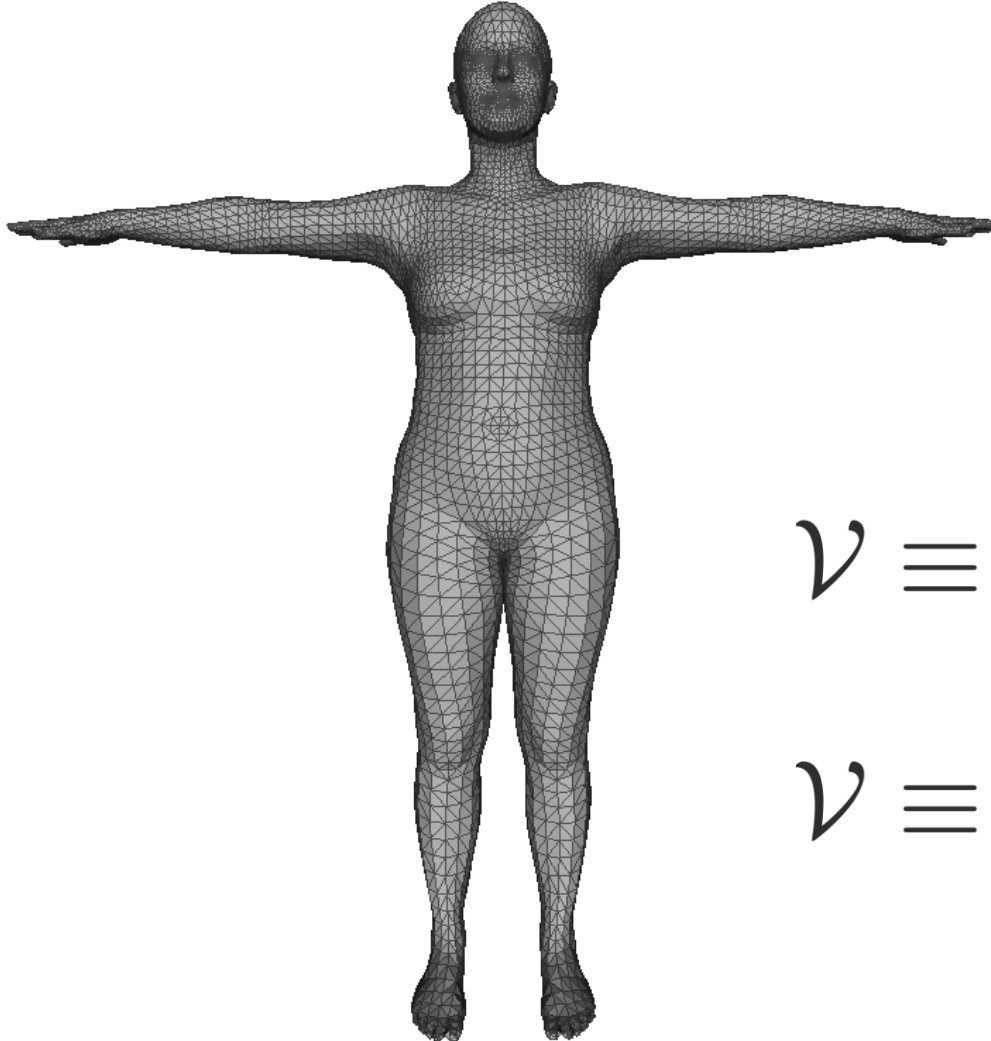
# Today: Rigid Alignment (Procrustes)



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

?

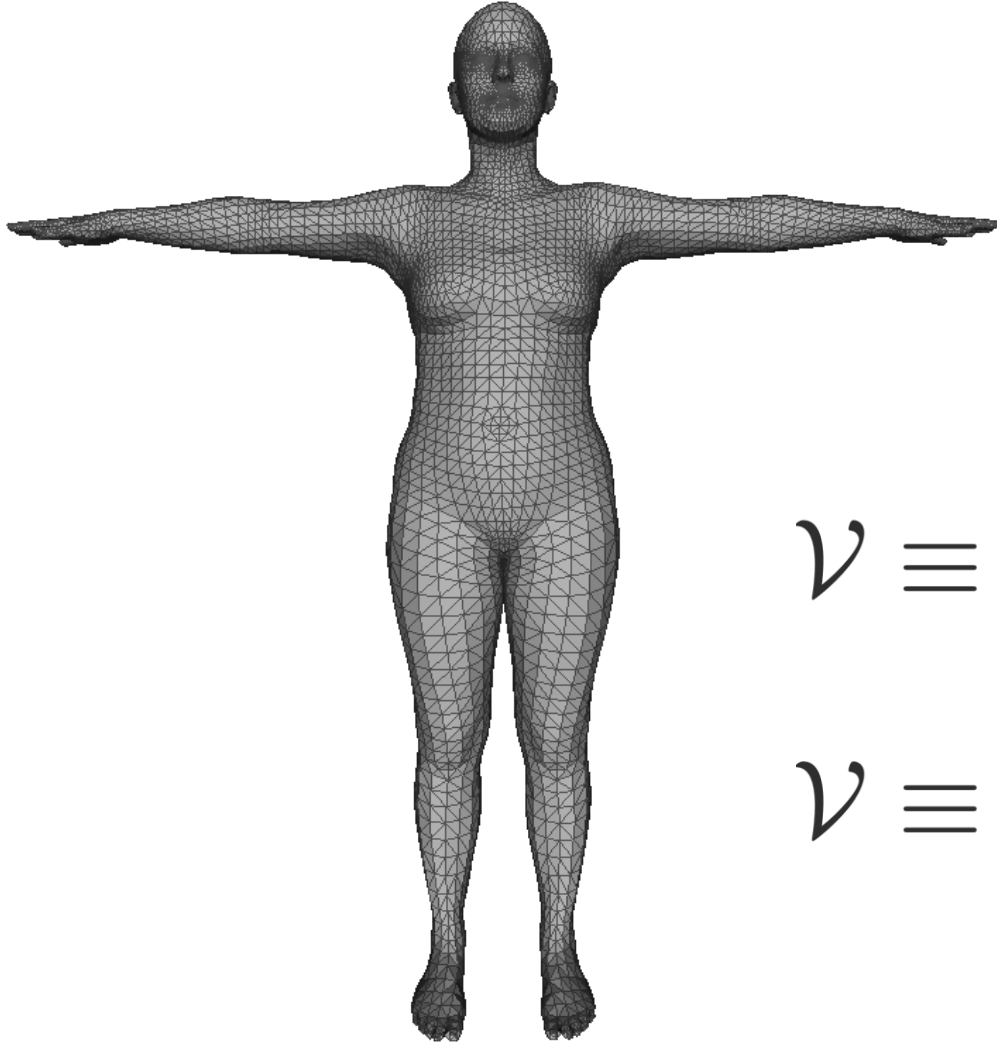
# Surface representation: Mesh



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

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

# 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 ?

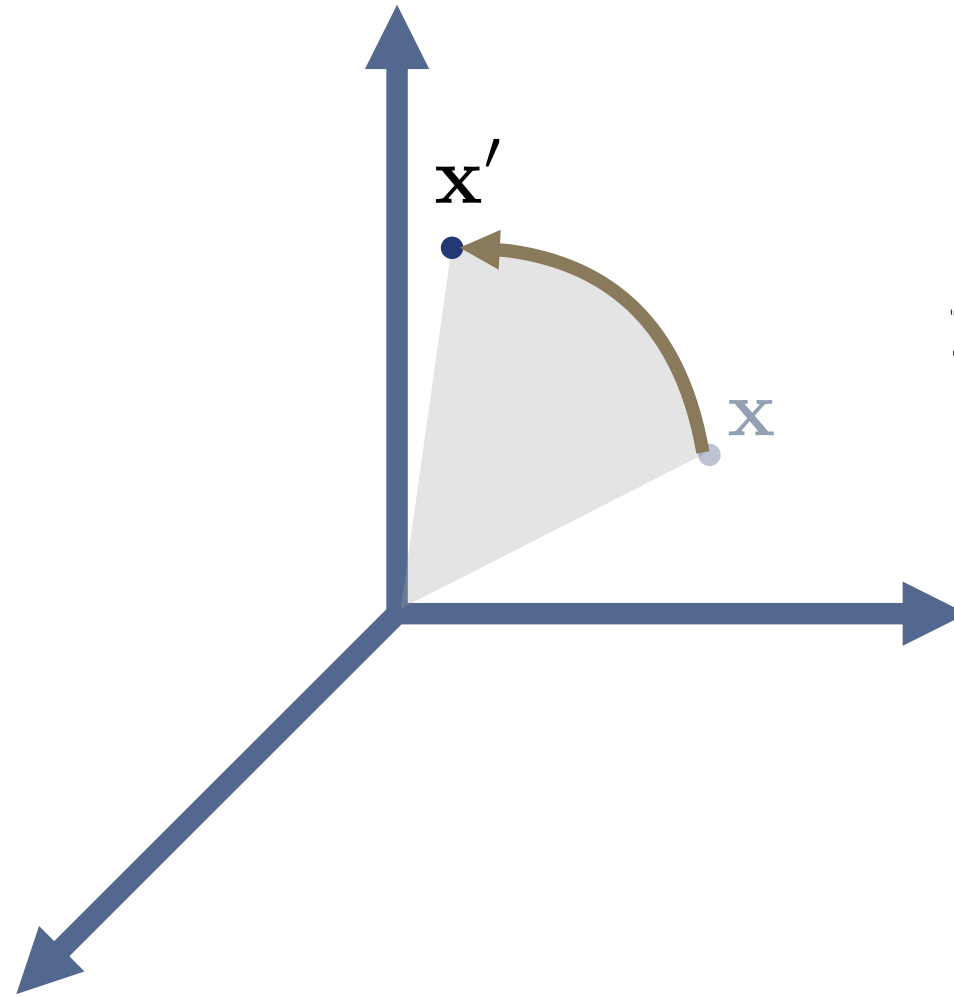
- Translate it
- Rotate it
- Today we'll consider also
- Scale it

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

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

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

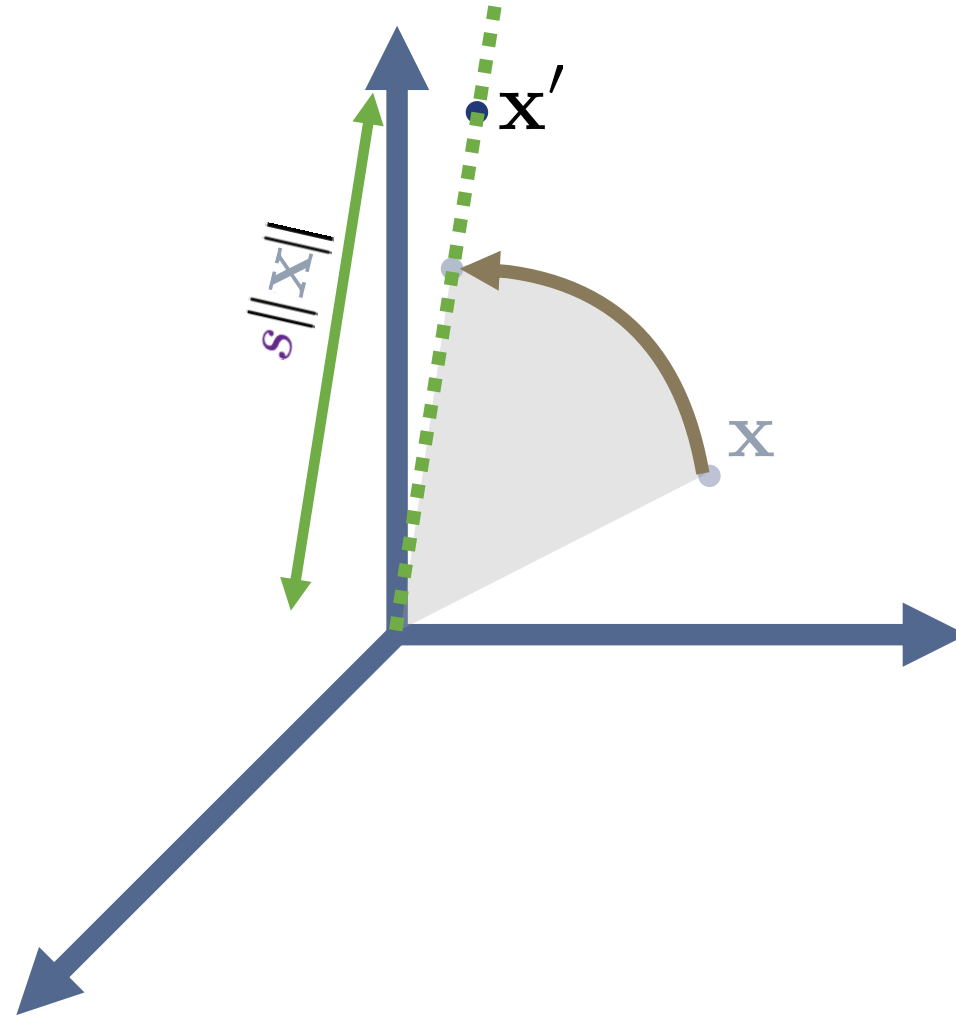
# Rigid transformation + scale



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

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

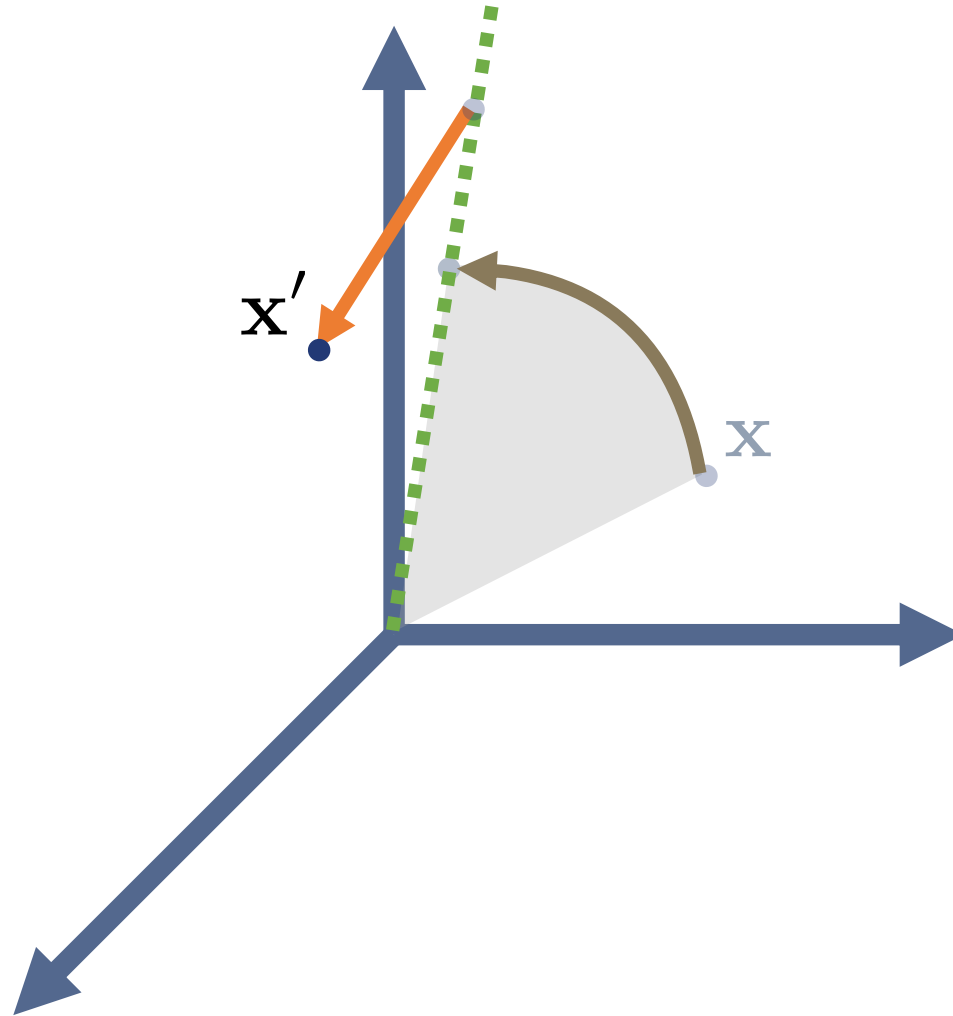
# Rigid transformation + scale



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

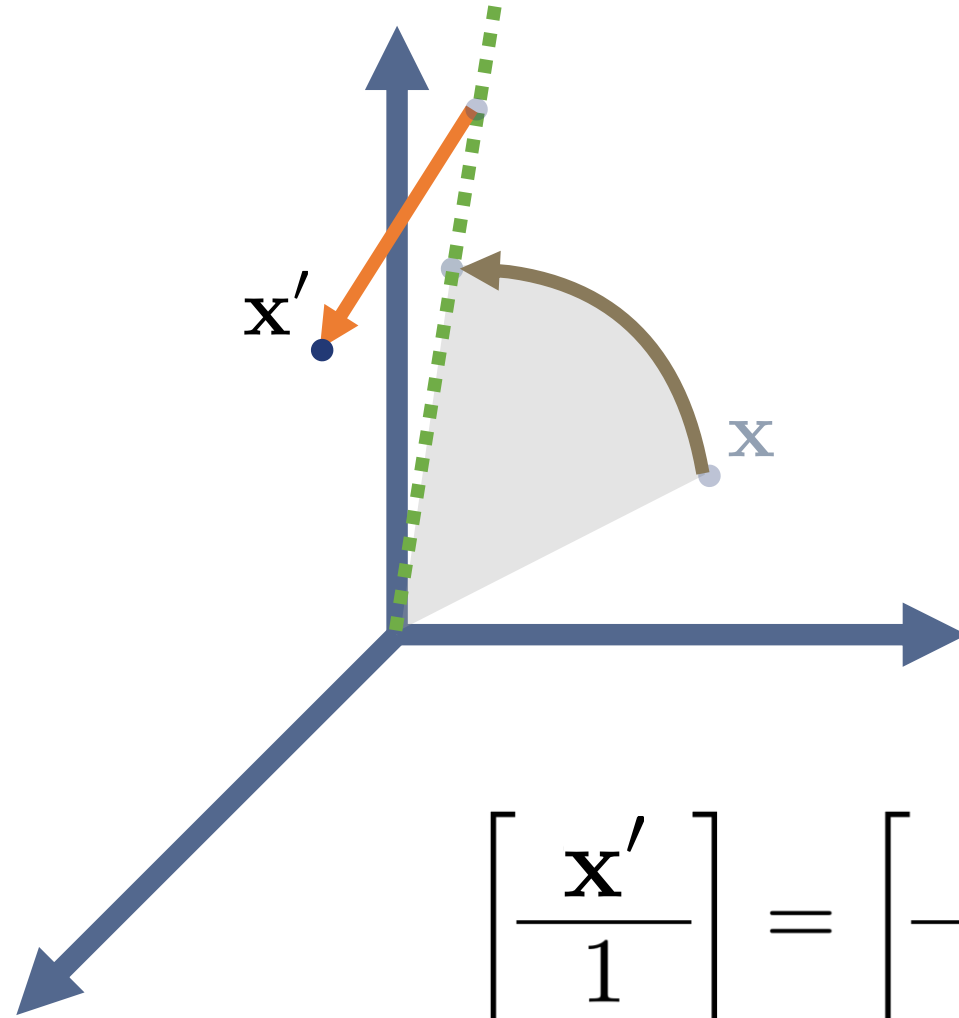
$$x' = sRx$$

# Rigid transformation + scale



$$R : \mathbb{R}^3 \rightarrow \mathbb{R}^3$$
$$\mathbf{x}' = sR\mathbf{x} + \mathbf{t}$$

# Rigid transformation + scale



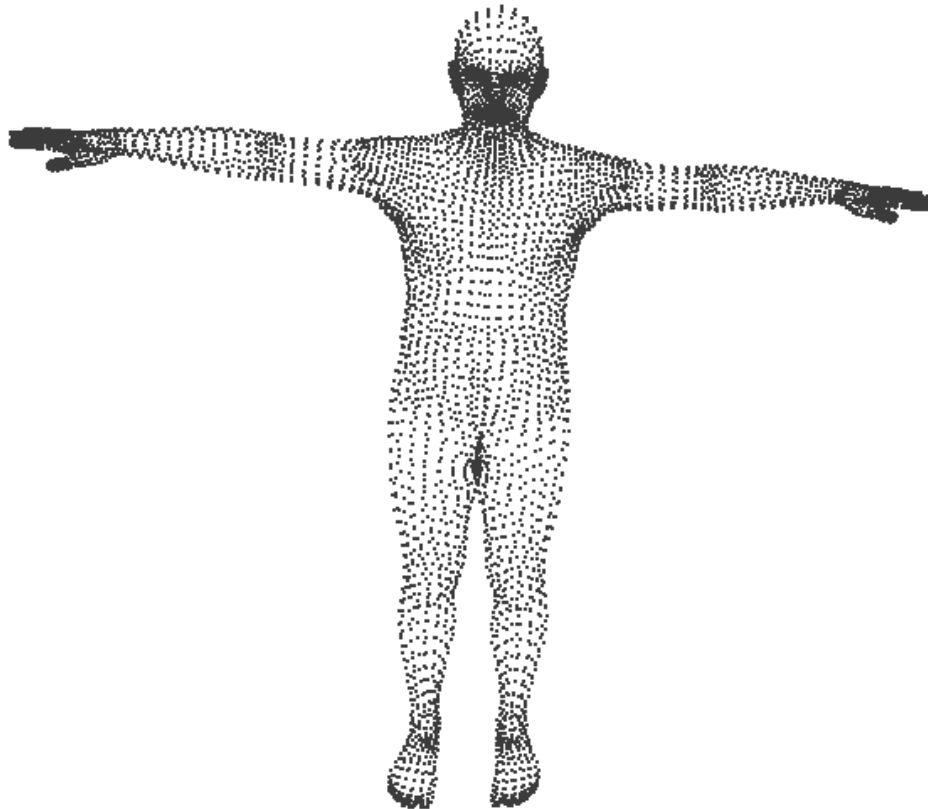
$$R : \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

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

$$\begin{bmatrix} \mathbf{x}' \\ 1 \end{bmatrix} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

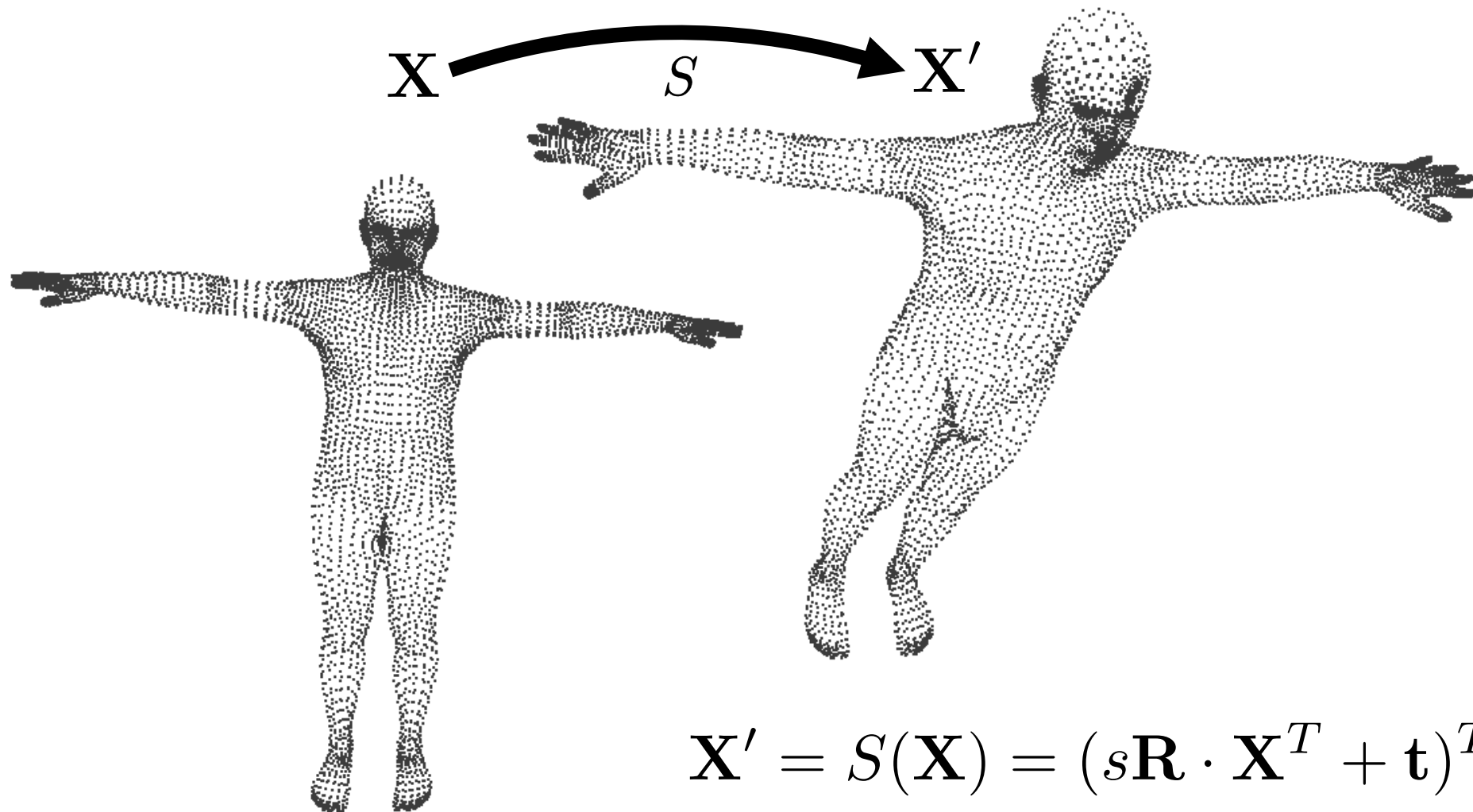
# Procrustes alignment problem definition

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



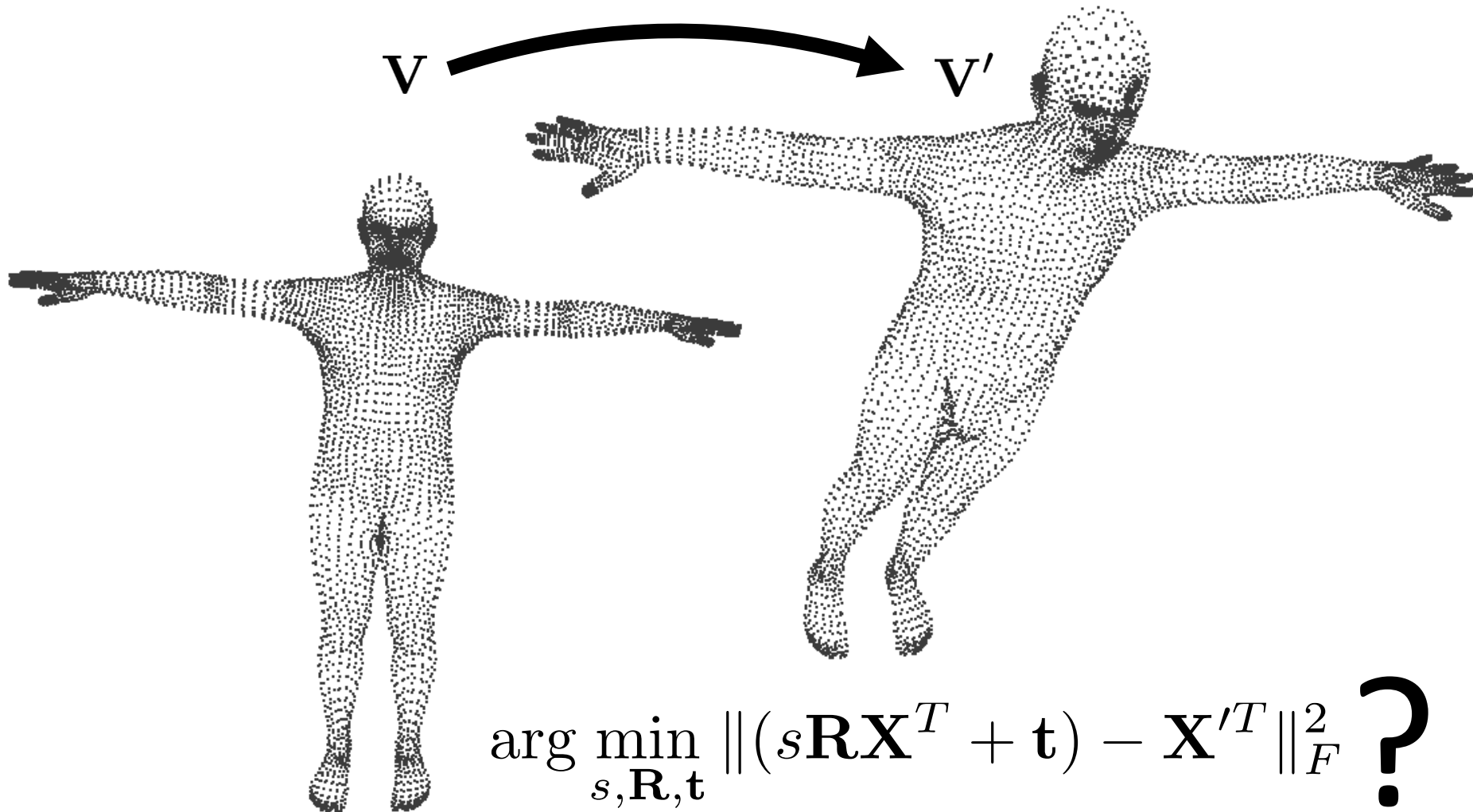
?

# Procrustes alignment problem definition



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

# Procrustes alignment problem definition



# Why Procrustes?



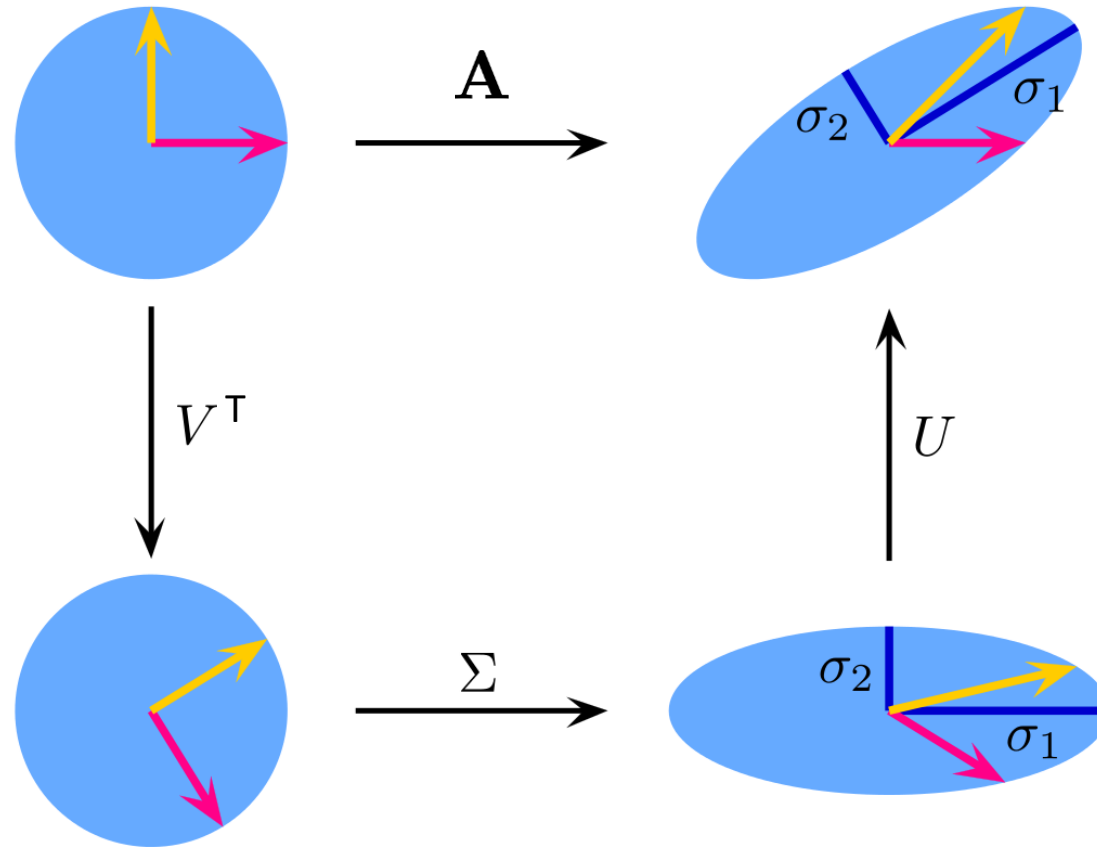
The name [Procrustes](#) ([Greek](#): Προκρούστης) refers to a bandit from Greek mythology who made his victims fit his bed either by stretching their limbs or cutting them off.

Procrustes 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:

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

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

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

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

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

warning: this is not the vertex matrix!

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

# Procrustes alignment steps

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

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

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

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

**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

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

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

# Procrustes derivation: translation

We remove the elements that do not depend on the translation and solve for  $\mathbf{t}$ .

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

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

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

Compute the centroid of the point clouds

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

So given  $s$  and  $R$ , we can compute the translation  $\mathbf{t}$

# 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

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

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

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

$$= \arg \min_{\mathbf{R}} \langle \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T, \mathbf{R}\bar{\mathbf{X}}^T - \bar{\mathbf{Y}}^T \rangle_F$$

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

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

$$= \arg \max_{\mathbf{R}} \langle \mathbf{R}, \bar{\mathbf{Y}}^T \bar{\mathbf{X}} \rangle_F$$

Optimal rotation does not depend on scale

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  $\Sigma$  ?

# 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  $\Sigma$  ?

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 pointsclds

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

# Procrustes derivation: scale

Optimize **scale** given the rotation

$$s = \arg \min_s E = \arg \min_s \left( \sum_i s^2 \bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i + \bar{\mathbf{y}}_i^\top \bar{\mathbf{y}}_i - 2s \bar{\mathbf{x}}_i^\top \mathbf{R}^\top \bar{\mathbf{y}}_i \right)$$

$$= \arg \min_s \left( s^2 \sum_i (\bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_i) - 2s \sum_i (\bar{\mathbf{x}}_i^\top \mathbf{R}^\top \bar{\mathbf{y}}_i) \right)$$

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

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

We used:

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

# Virtual Humans – Winter 24/25

Lecture 4\_1 – ICP: Iterative Closest Points

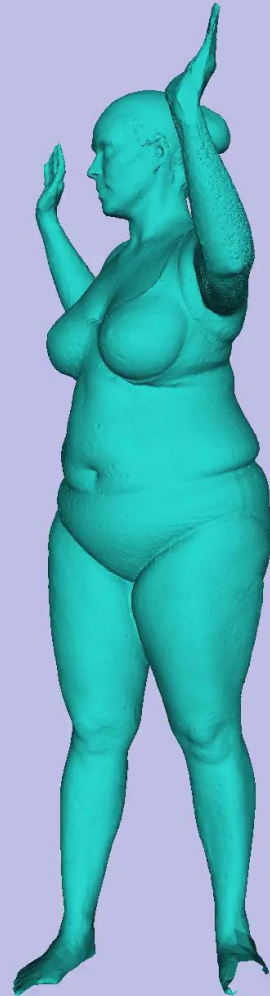
Prof. Dr.-Ing. Gerard Pons-Moll

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.

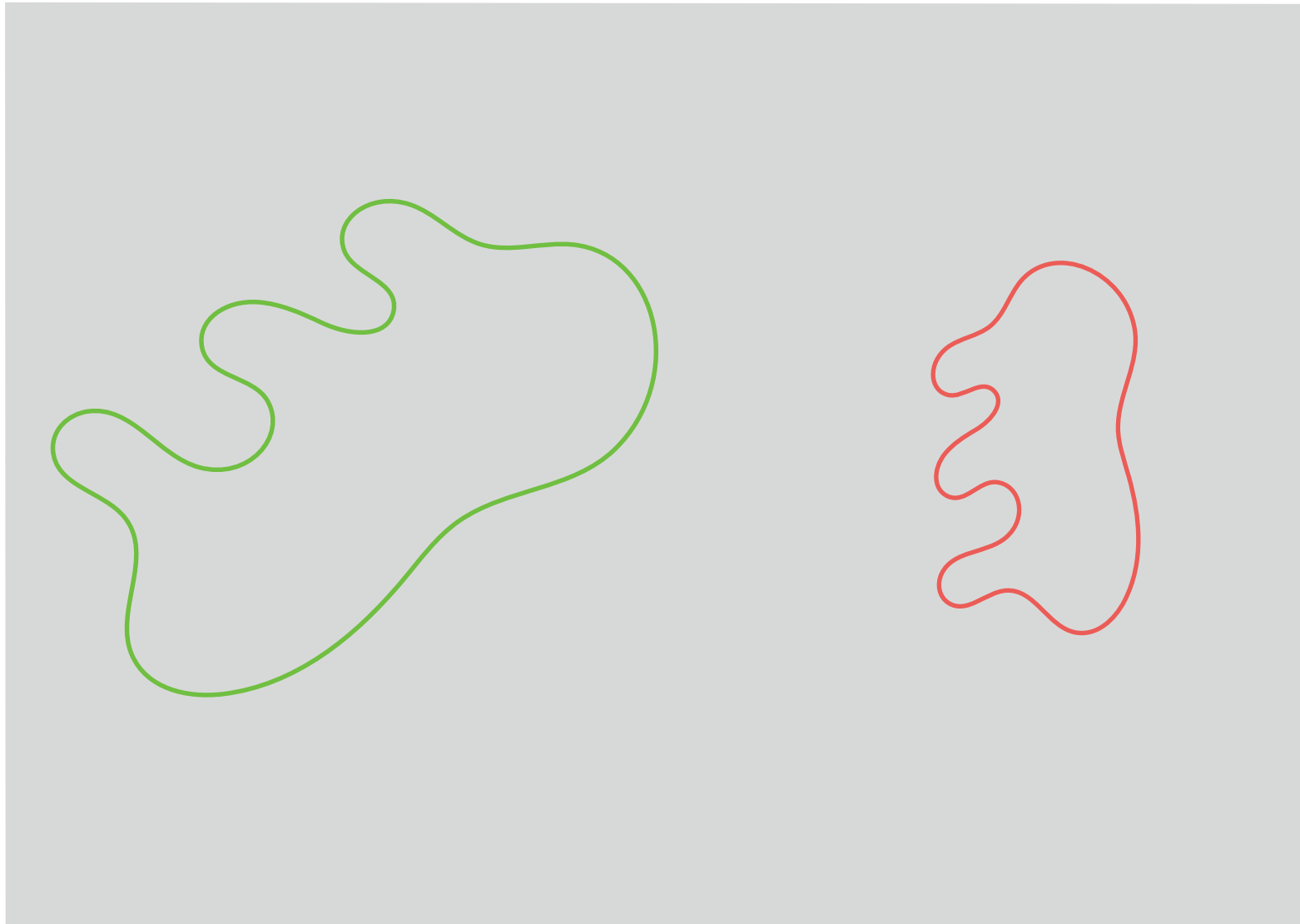
## PROBLEMS:

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

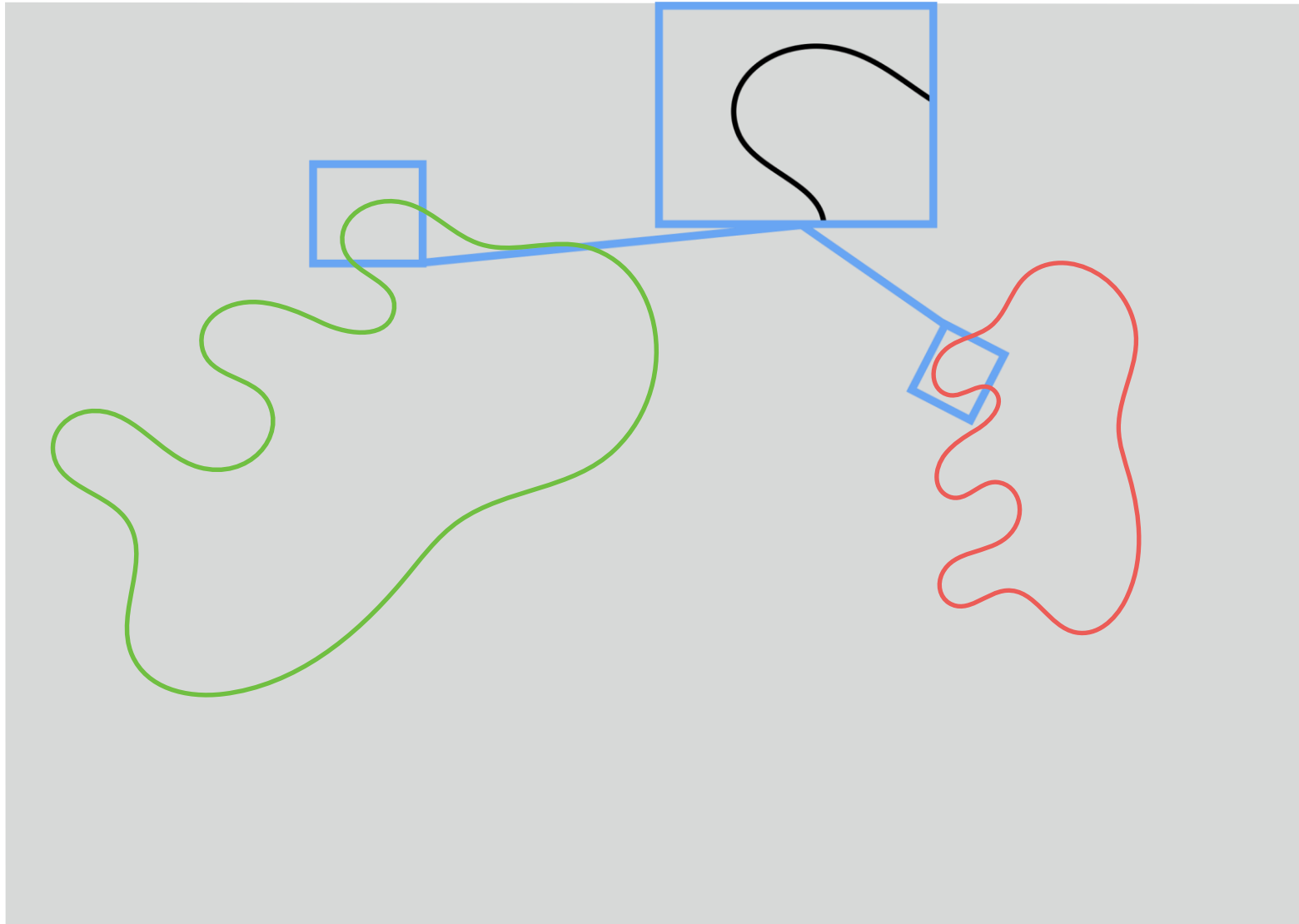
# ICP and alignment based on optimisation

- Optimising alignment and correspondences using *Iterative Closest Point (ICP)*.
- Alignment through *continuous* optimisation.

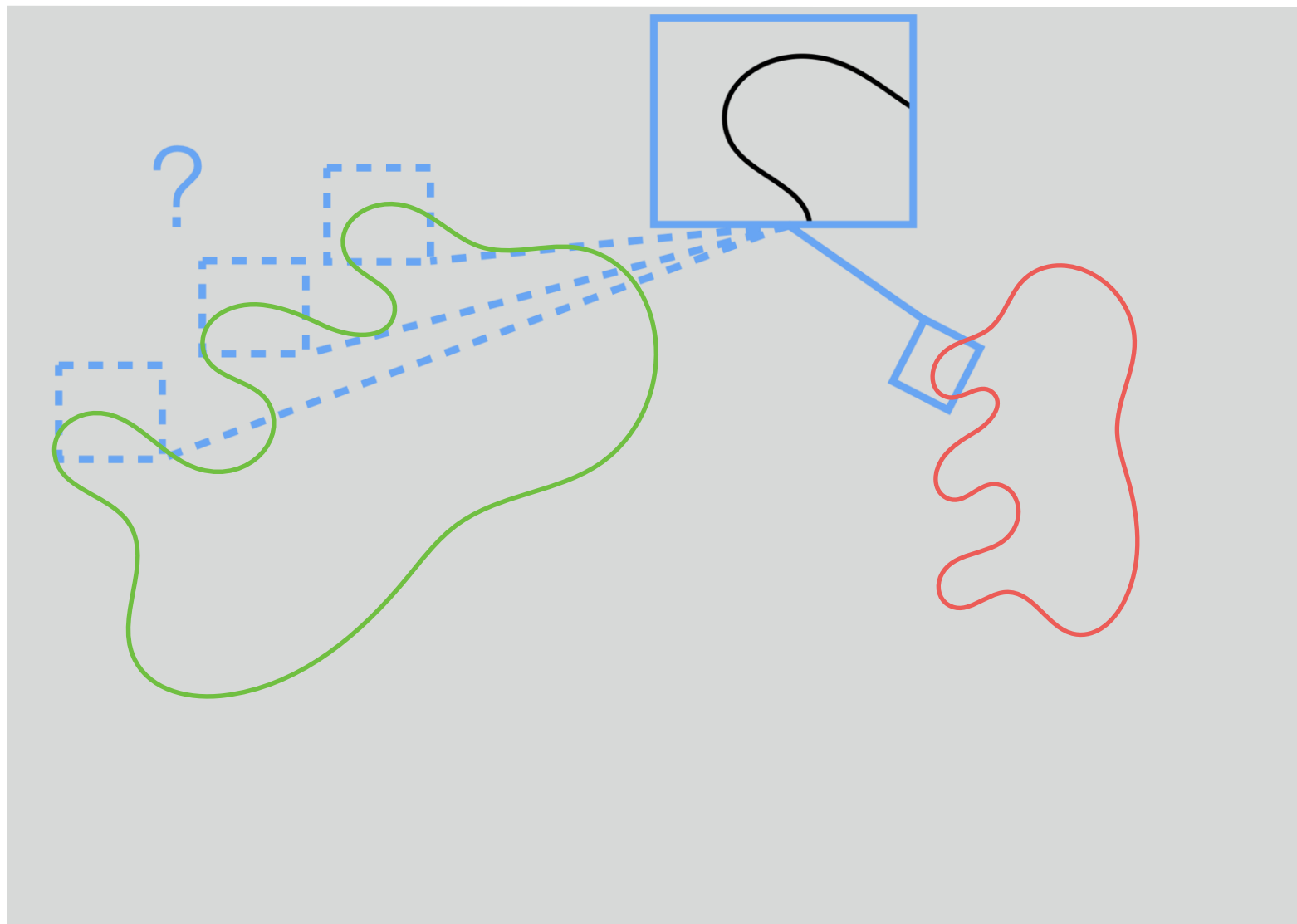
# How do we find correspondences?



# 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?

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

# 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

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

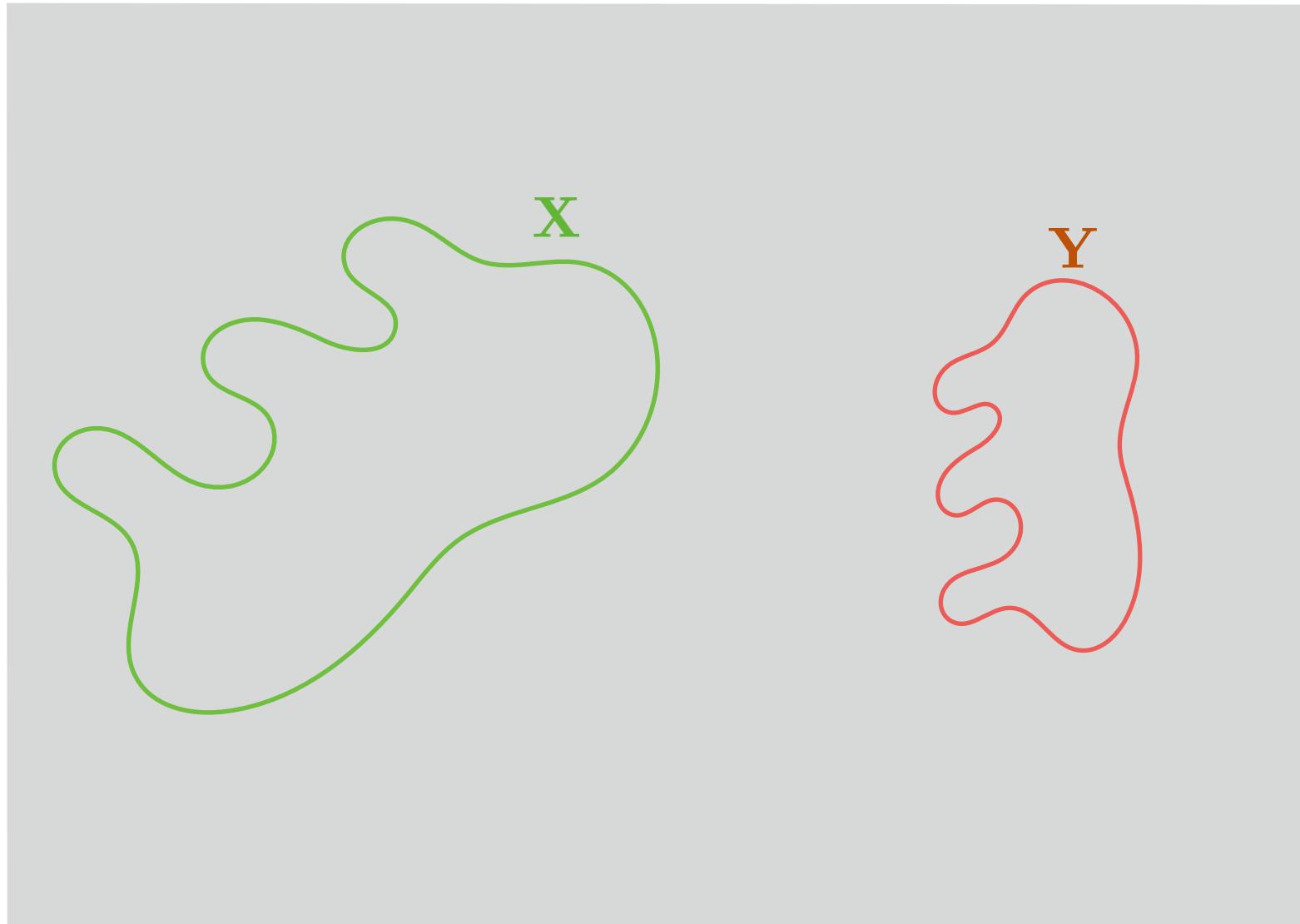
original unsorted points

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

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

$\mathbf{X} \equiv f^0(\mathbf{X})$

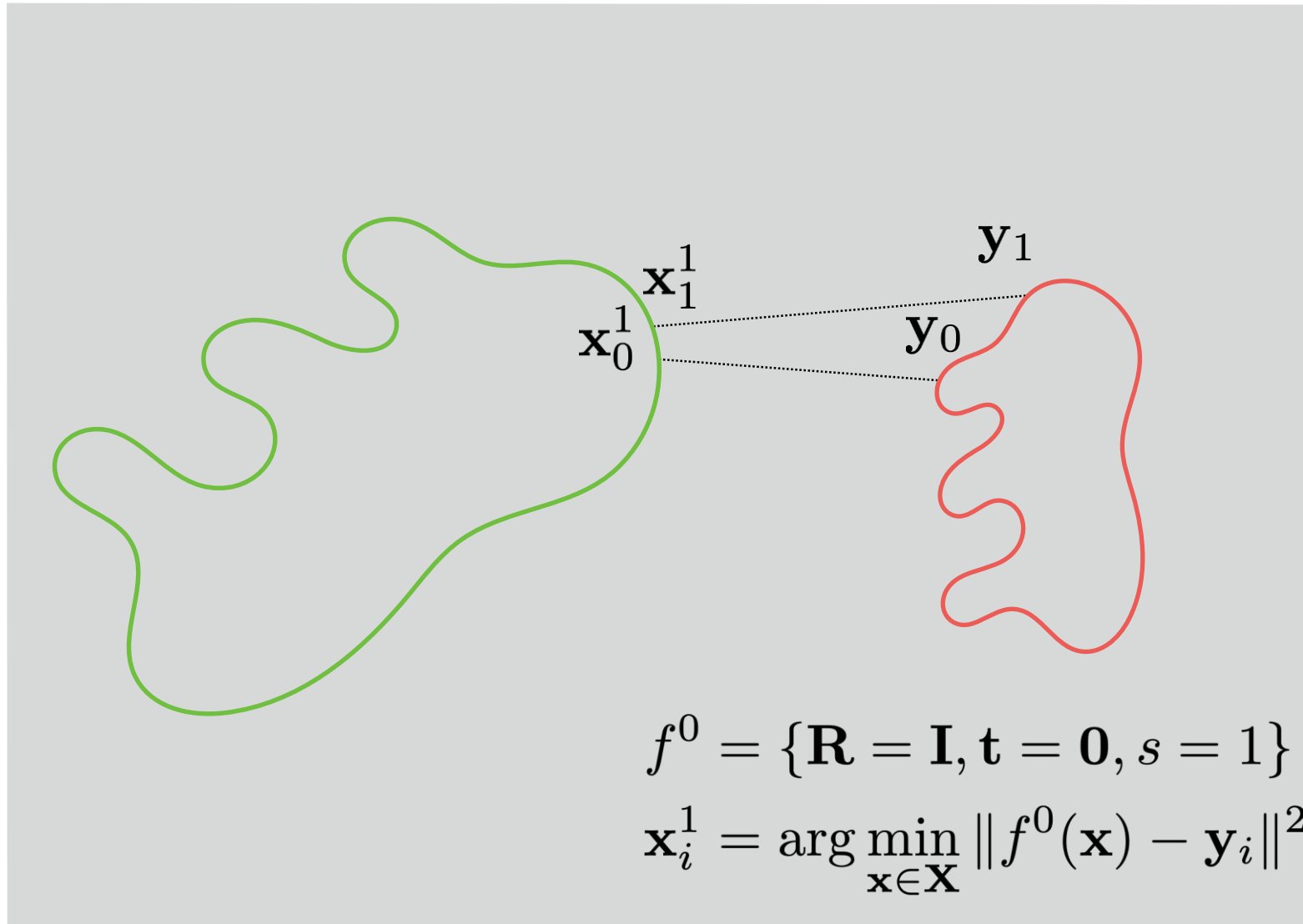
$\mathbf{x}_0^1$   $\mathbf{y}_0$

Neutral initialization.  
Initializing  $\mathbf{t}$  to align centroids  
should work better!

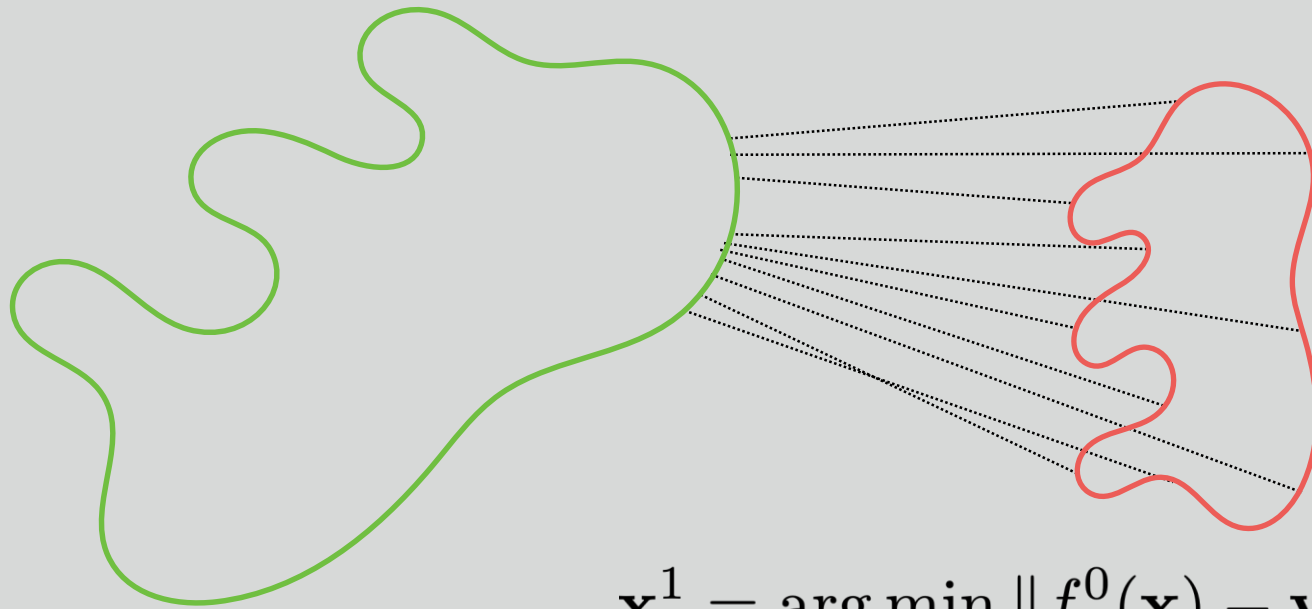
$\longrightarrow f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \mathbf{0}, s = 1\}$

$\mathbf{x}_0^1 = \arg \min_{\mathbf{x} \in \mathbf{X}} \|f^0(\mathbf{x}) - \mathbf{y}_0\|^2$

# Make up reasonable correspondences



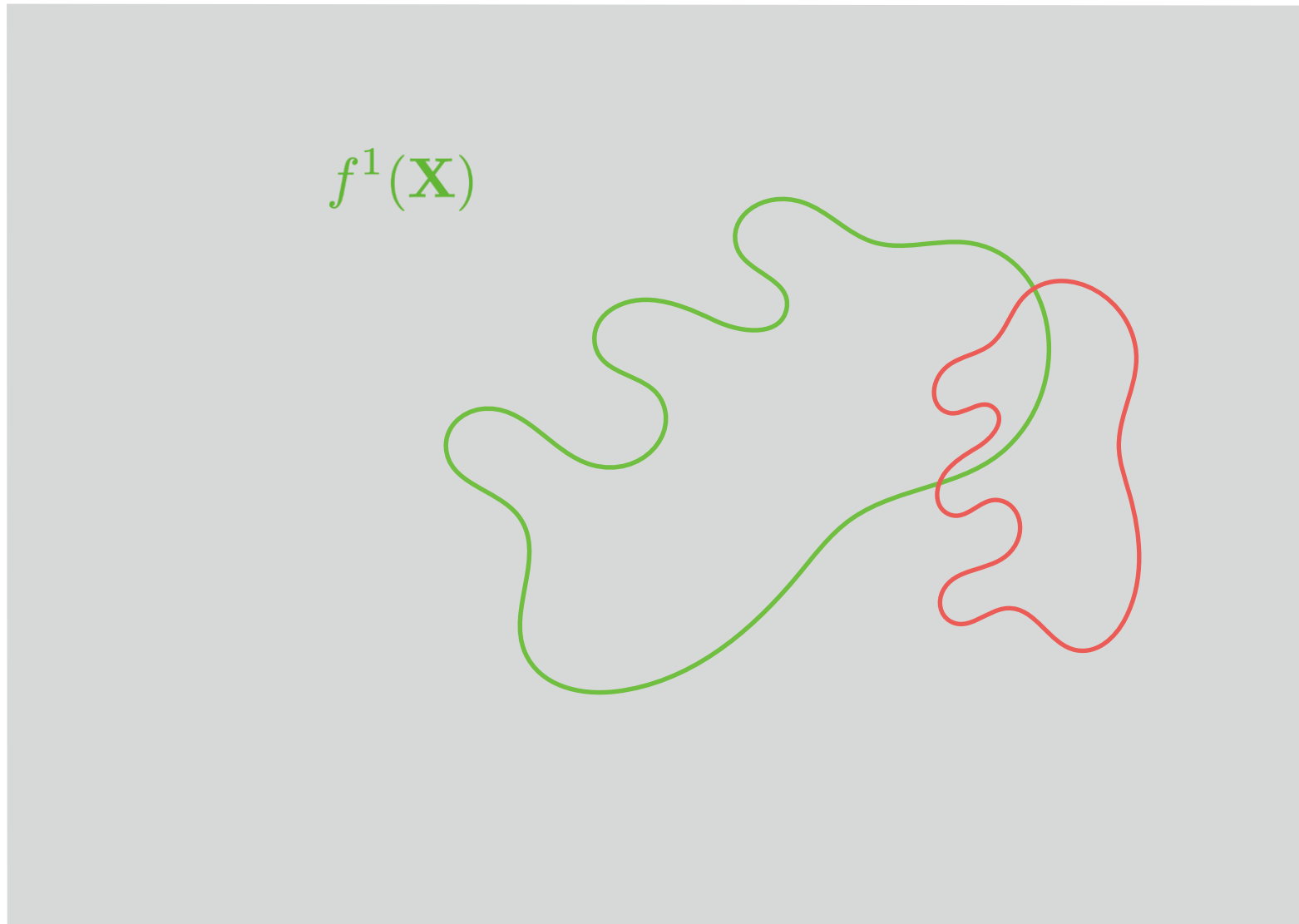
# Solve for the best transformation



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

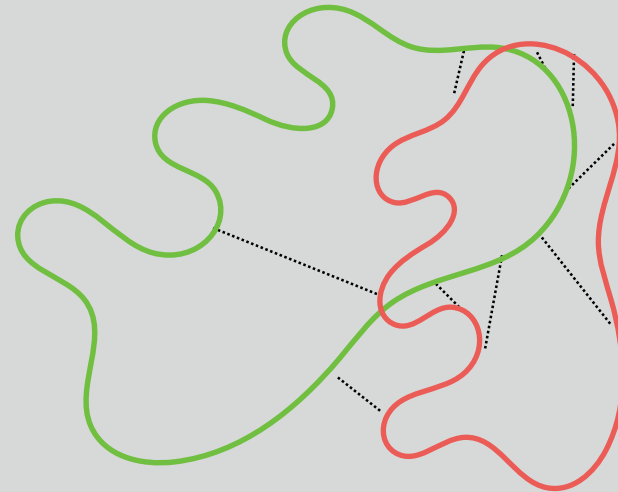
solve with procrustes  $\longrightarrow f^1 = \arg \min_f \sum_i \|f(\mathbf{x}_i^1) - \mathbf{y}_i\|^2$

Apply it ...



and iterate!

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

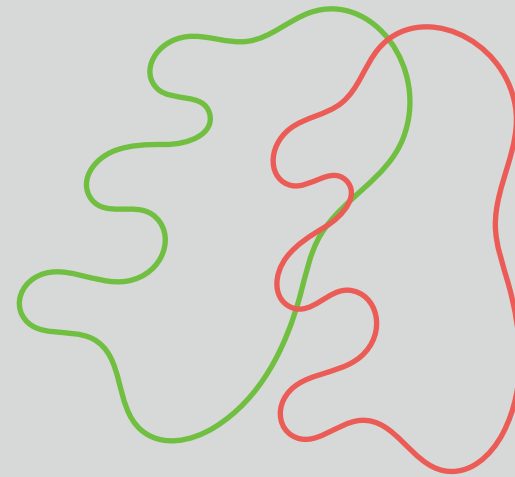


$$f^1 = \arg \min_f \sum_i \|f(\mathbf{x}_i^1) - \mathbf{y}_i\|^2$$

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

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

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

typically better than 0



# 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 (  $s, \mathbf{R}, \mathbf{t}$  ) 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  $(s, \mathbf{R}, \mathbf{t})$  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  $(s, \mathbf{R}, \mathbf{t})$  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

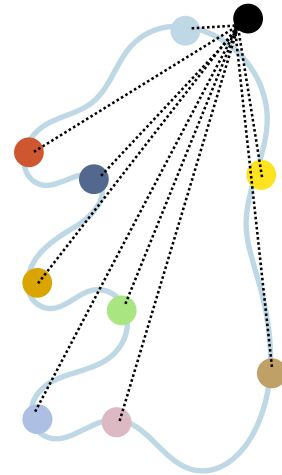
# Is ICP the best we can do?

Iteration j:

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

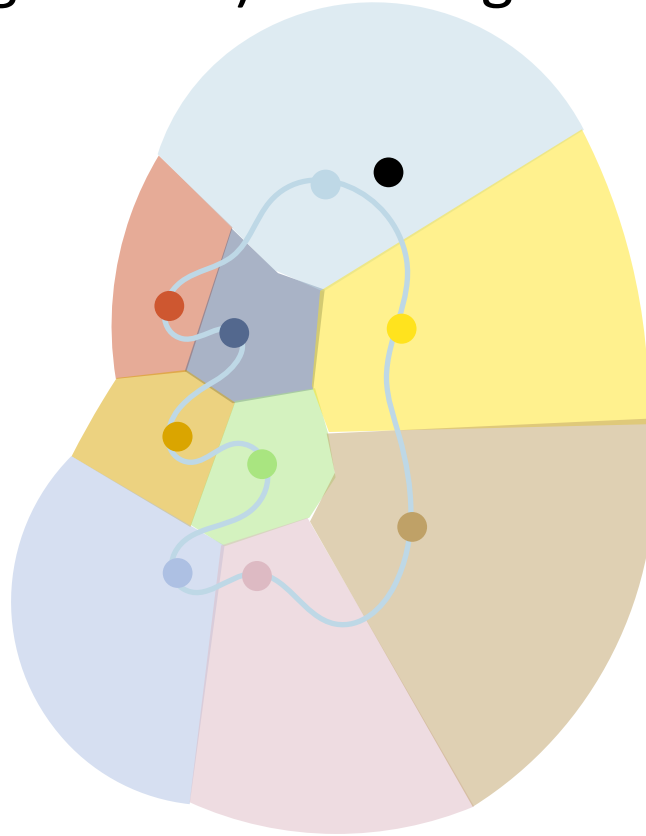
# Closest points

- 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  $\log(n)$

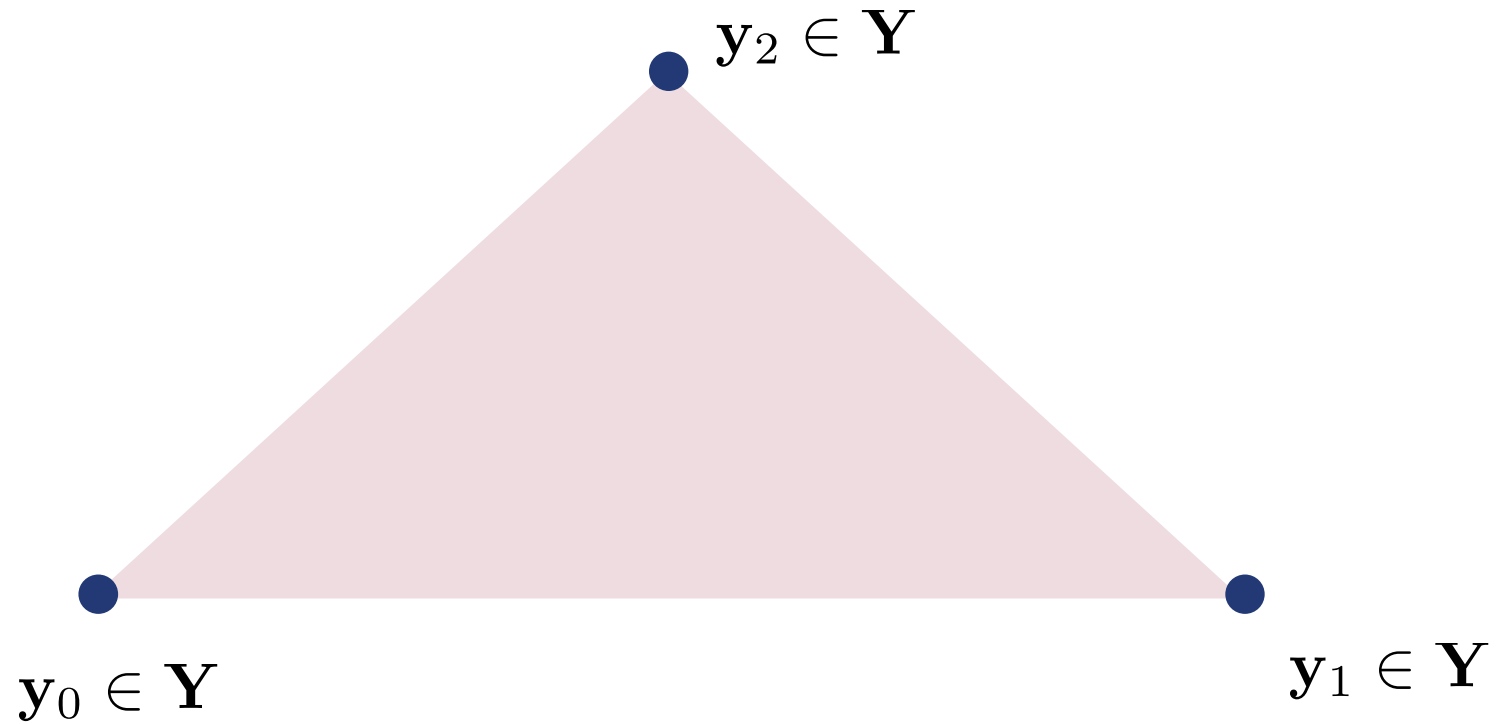


- Random point sampling also reduces the running time

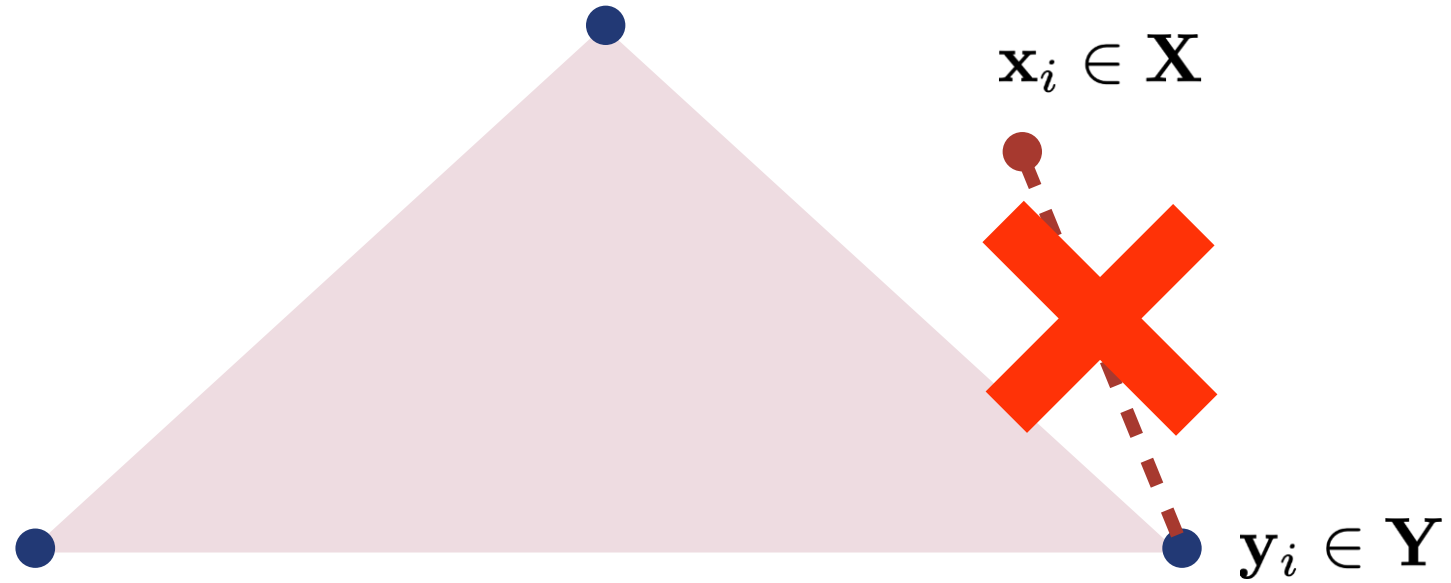
# ICP: Tips to avoid local minima

- 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

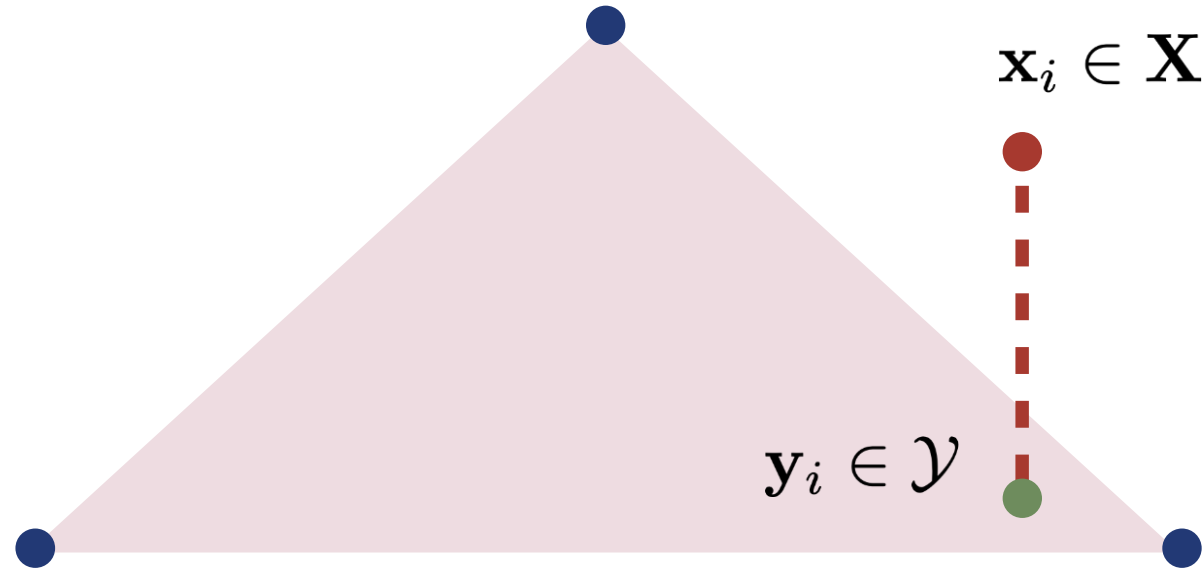


# Closest points: avoid local minima



**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:

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

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

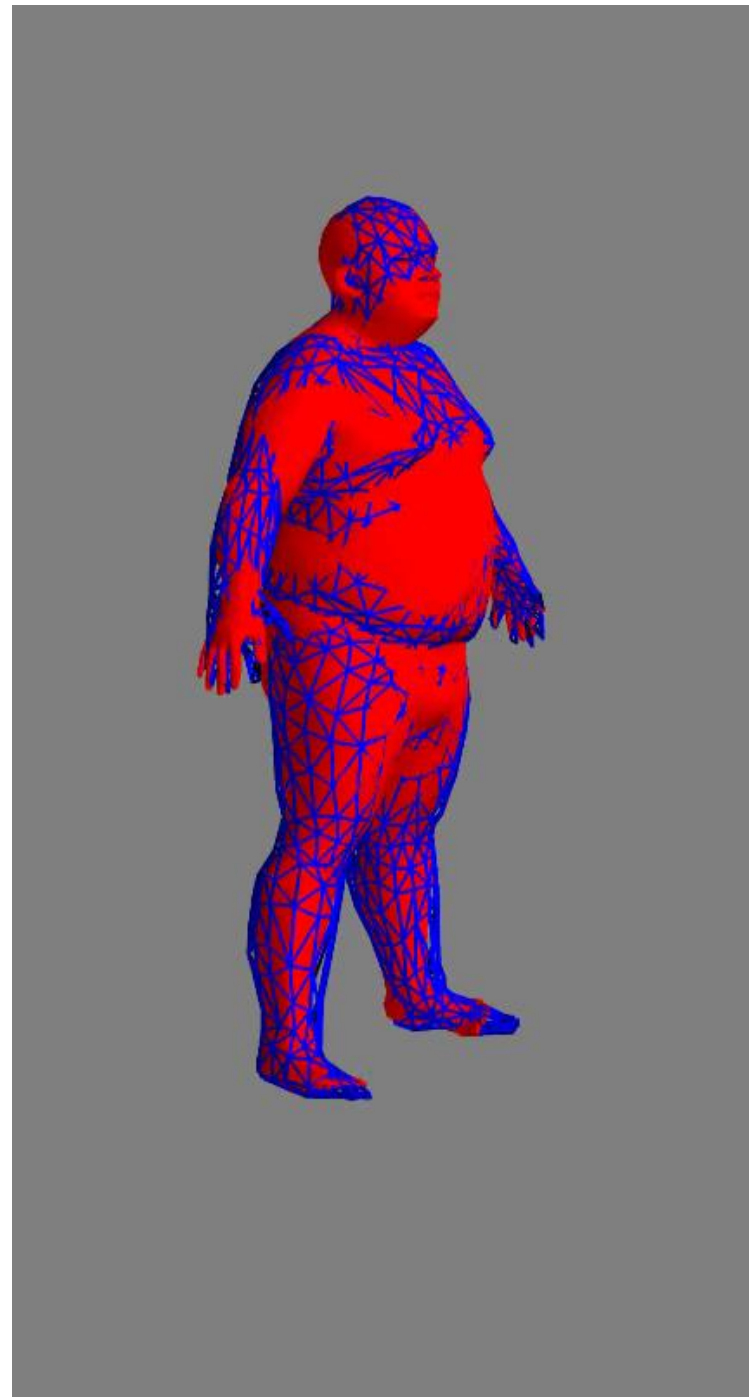
$$g^{j+1} = \nabla E(f^j)$$

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

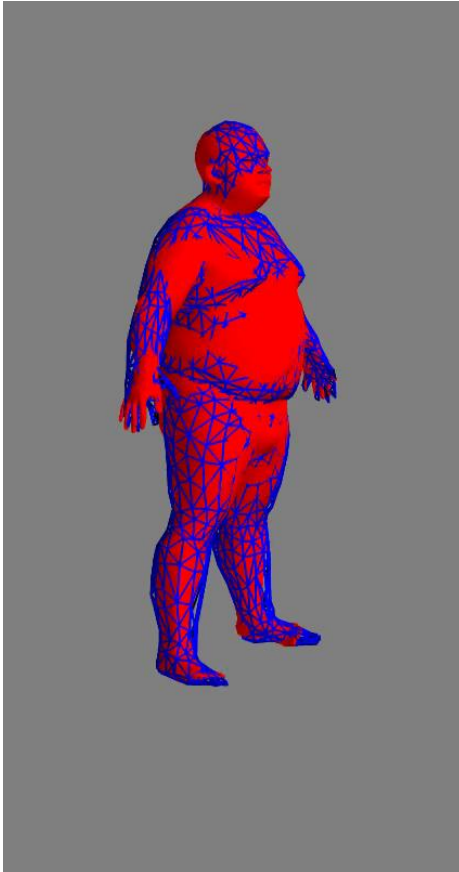
$$f^{j+1} = k_{step}(g^{0\dots j+1}, f^{0\dots j}) \leftarrow \text{(for example } f^{j+1} = f^j - \alpha g^{j+1} \text{)}$$

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

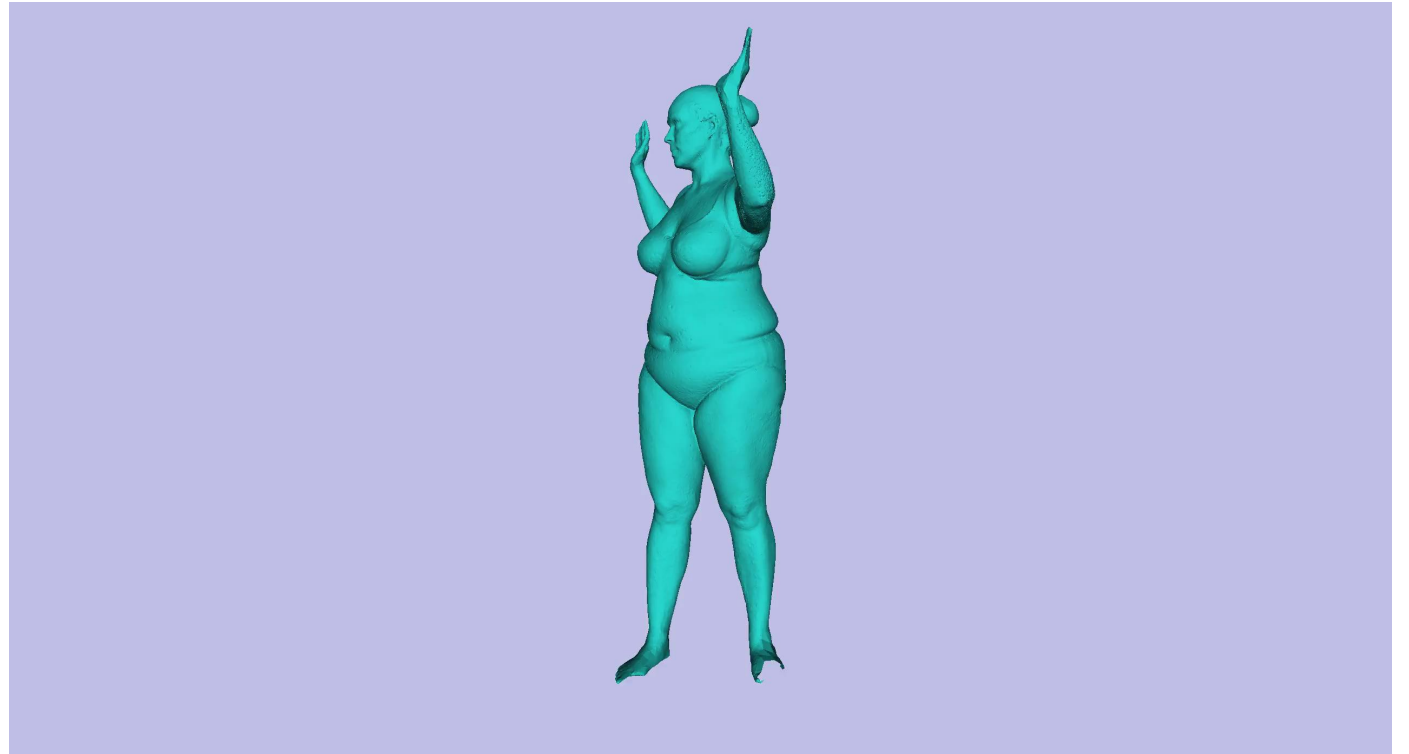
# Gradient-based ICP



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

# Gradient-based ICP

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

$$E \equiv \sum_i \left\| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \right\|^2$$
$$g^{j+1} = \nabla E(f^j)$$

# Gradient-based ICP

- Gradient: derivative of the sum of squared distances with respect to transformation  $f$  parameters
- Each derivative is easy
  - Who wants to writes it down?

$$E \equiv \sum_i \left\| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \right\|^2$$
$$g^{j+1} = \nabla E(f^j)$$

# Gradient-based ICP

- Gradient: derivative of the sum of squared distances with respect to transformation  $f$  parameters
- Each derivative is easy
  - Who wants to writes it down?
- **Chain rule and automatic differentiation!**

$$E \equiv \sum_i \left\| \min_{\mathbf{x}} f(\mathbf{x}) - \mathbf{y}_i \right\|^2$$
$$g^{j+1} = \nabla E(f^j)$$

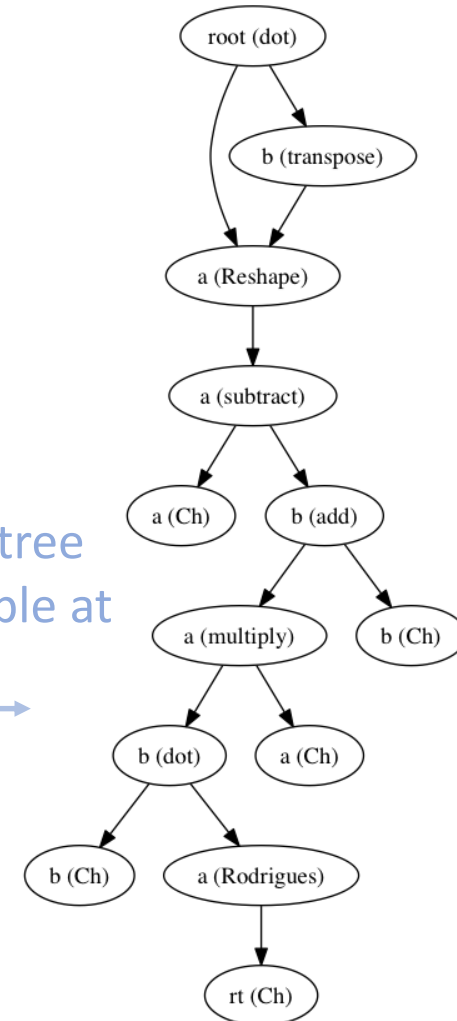
# Automatic differentiation

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

write as if it was numpy code

```
1 import numpy as np
2 from numpy.linalg import norm
3 rot = np.zeros((3,3))
4 scale = np.zeros(3)
5 trans = np.zeros((3,1))
6 x = np.random.randn(3, 100)
7 y = np.random.randn(3, 100)
8 s = (y - (scale*rodriguez(rot).dot(x) + trans)).norm()
9 rot = s.dot(x.T)
10 rot = rot.dot(x)
11 import sys; sys.exit(0)
```

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

# 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