UNIVERSITÄT DES SAARLANDES

max planck institut informatik

Body Models I-2

Gerard Pons-Moll and Paul Swoboda

Max Planck Institute for Informatics

December 12, 2018

What is missing

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

PROBLEMS:

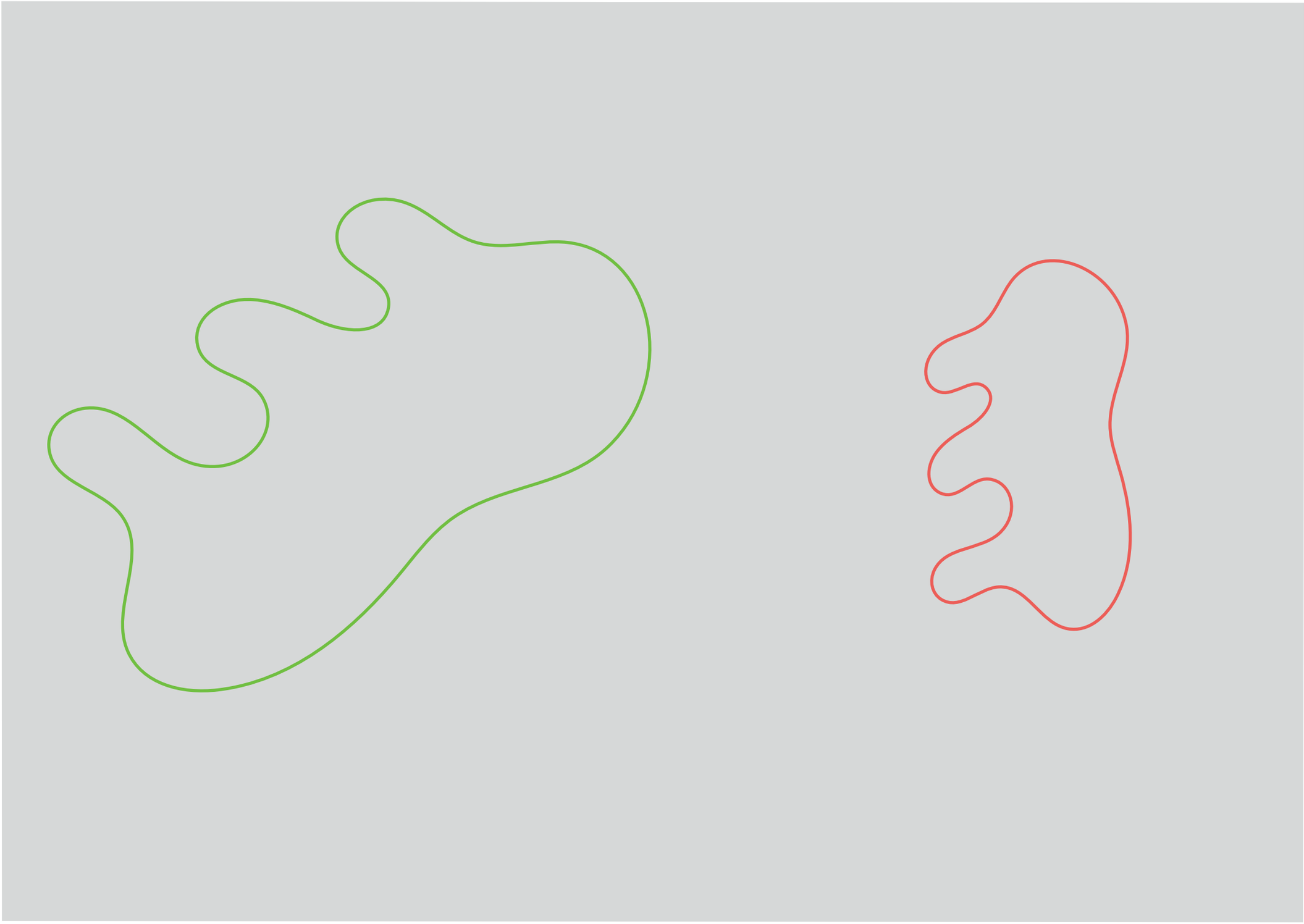
- How do we find the correspondences between shapes ?
- How do we align shapes non-rigidly ?

Today

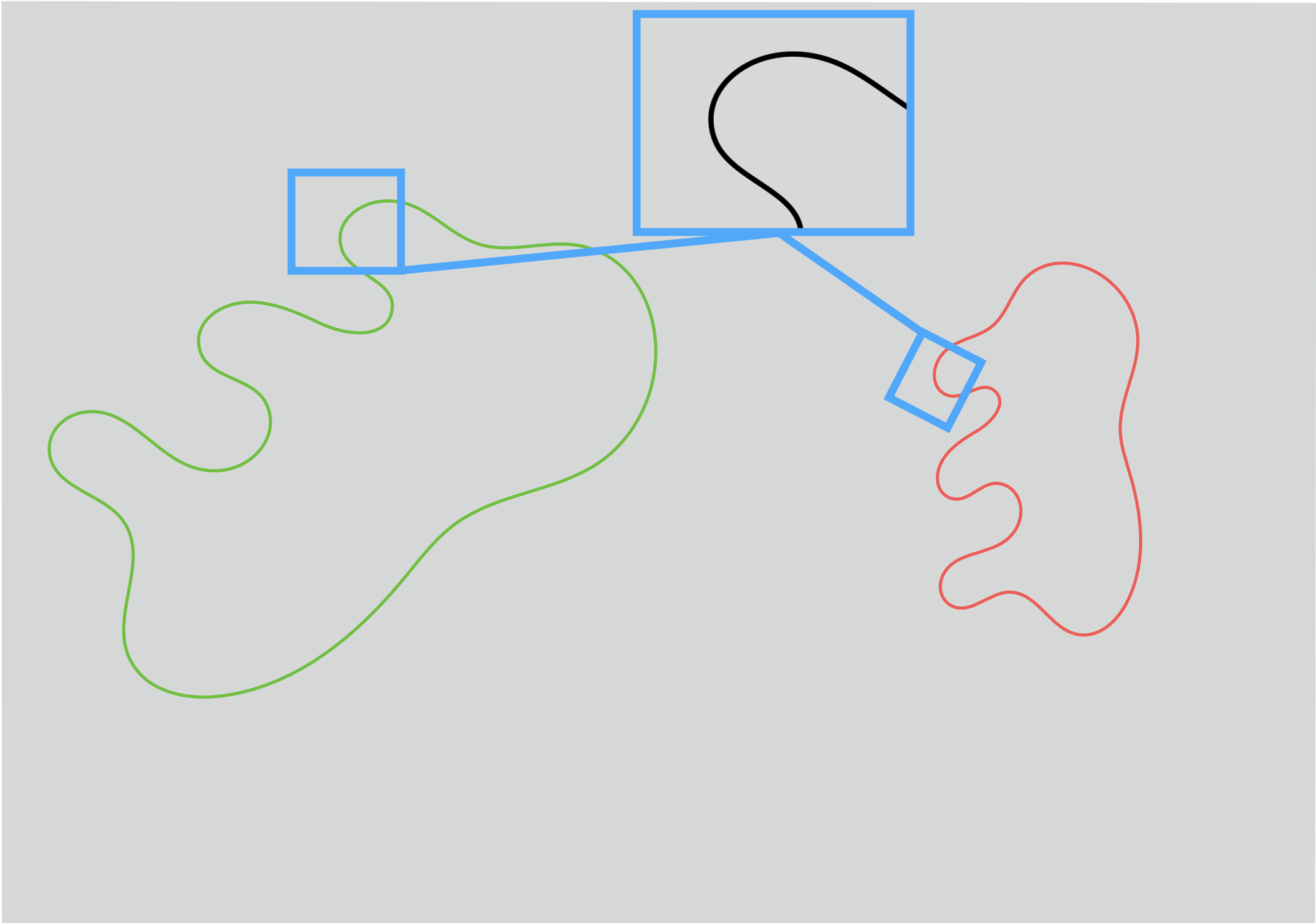
- Optimising alignment and correspondences using *Iterative Closest Point (ICP).*

- Alignment through *continuous* optimisation.

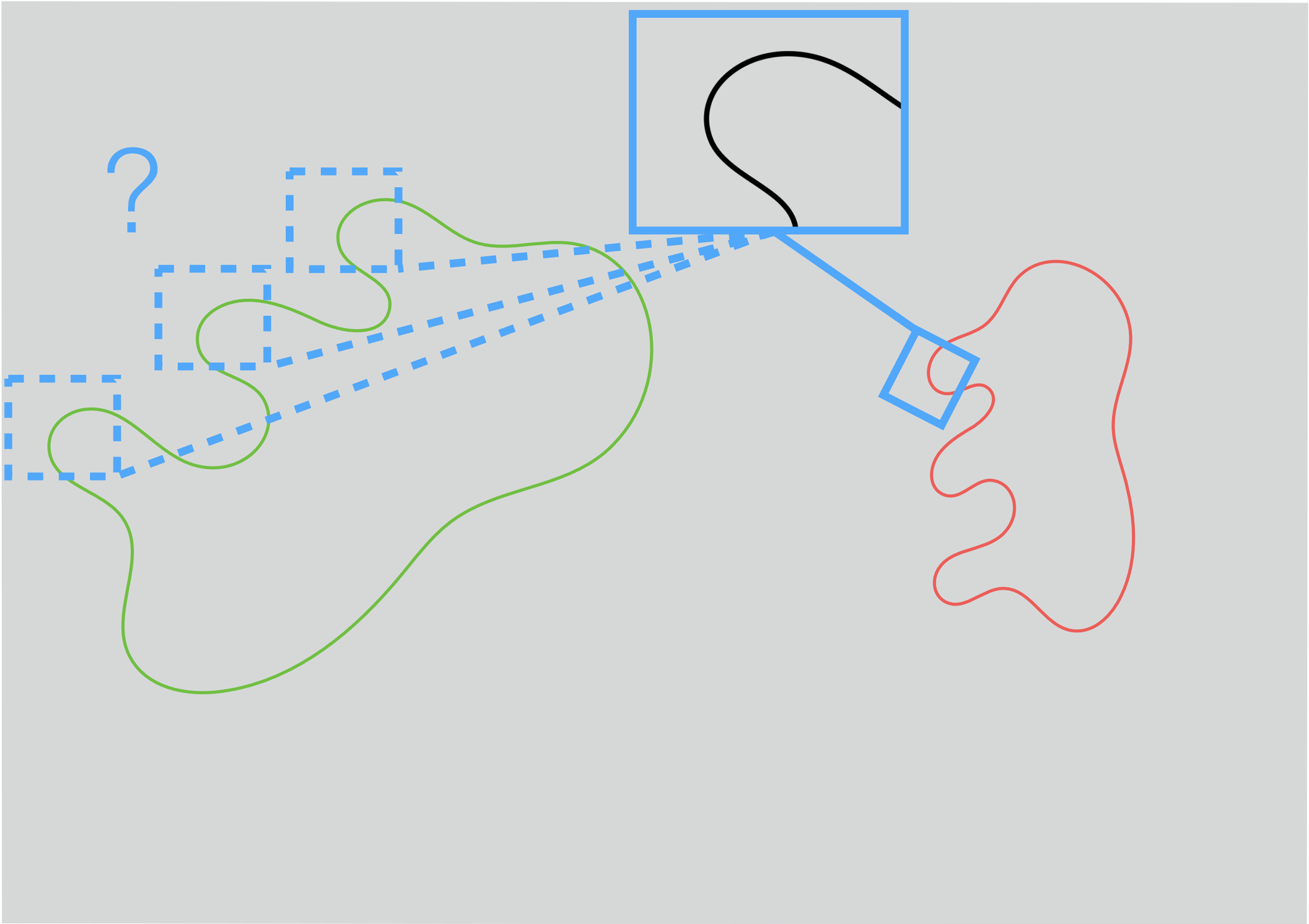
Ideas?



Ideas?



Ideas?



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

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?

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?

iteration

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

original unsorted points

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

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?

iteration

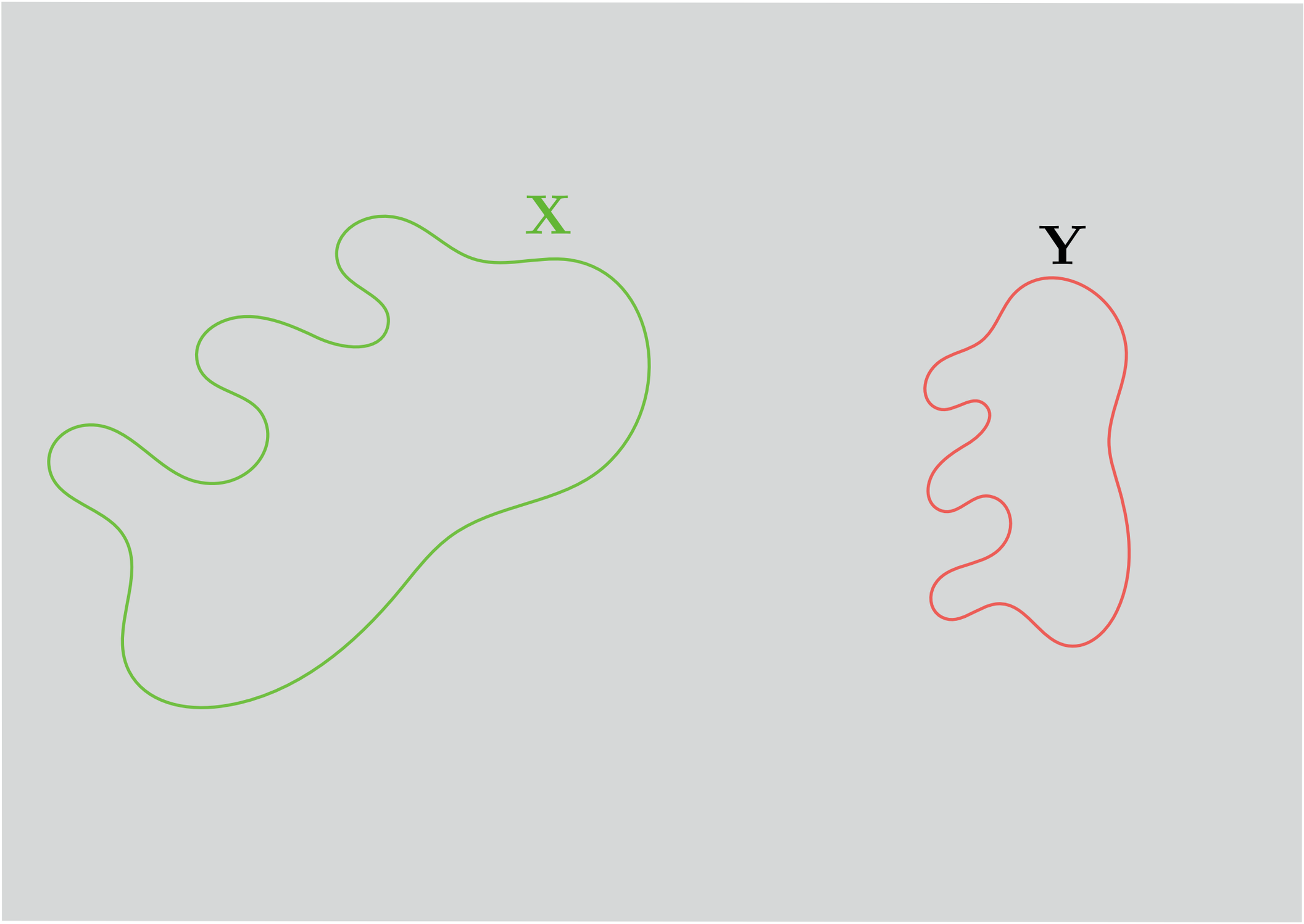
$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$

Given current best transformation, which are the closest correspondences?

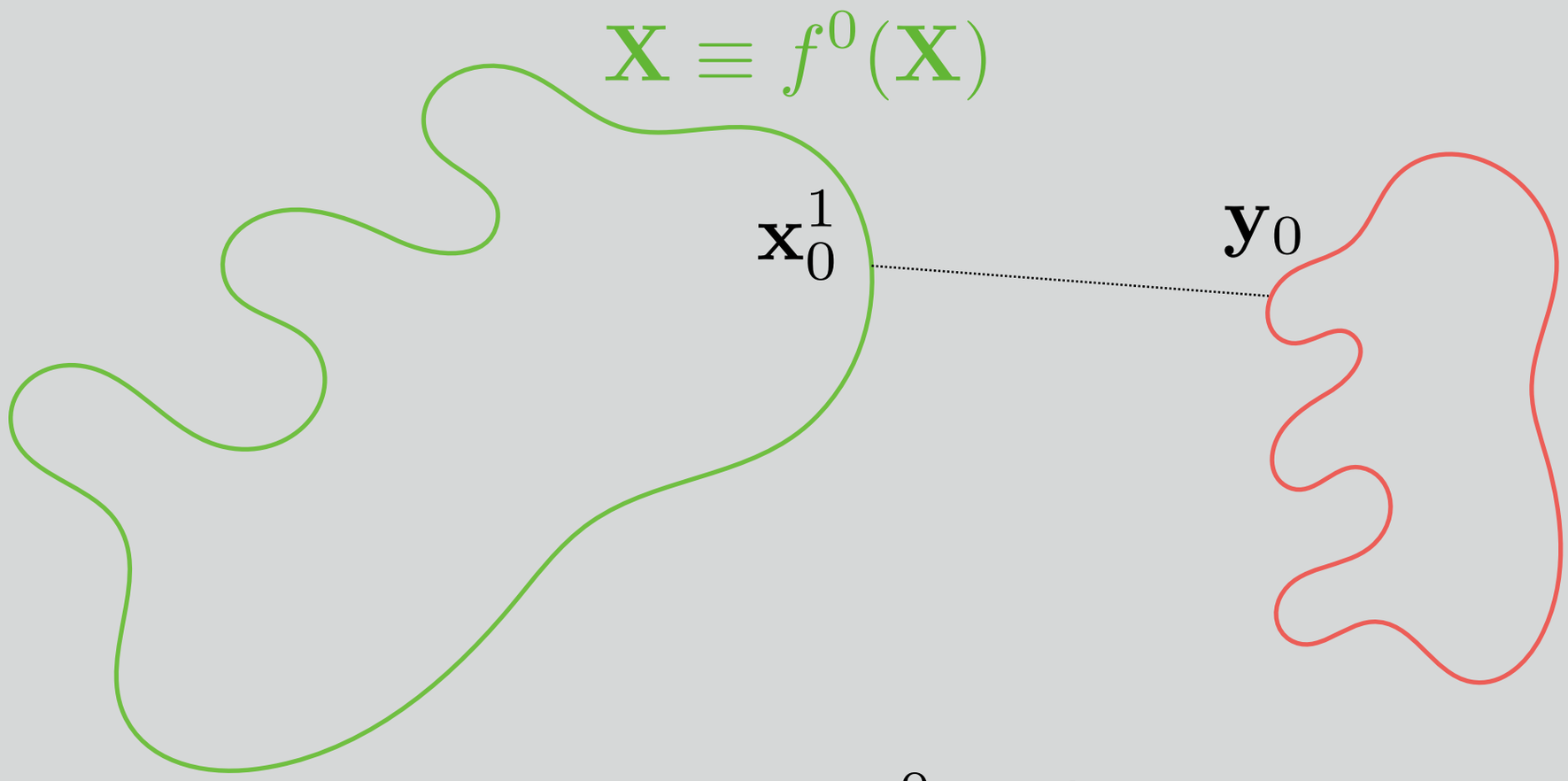
$$f^{j+1} = \arg\min_f \sum_i \|f(\mathbf{x}_i^{j+1}) - \mathbf{y}_i\|^2$$

Given current best correspondences, which is the best transformation?

Make up reasonable correspondences



Make up reasonable correspondences

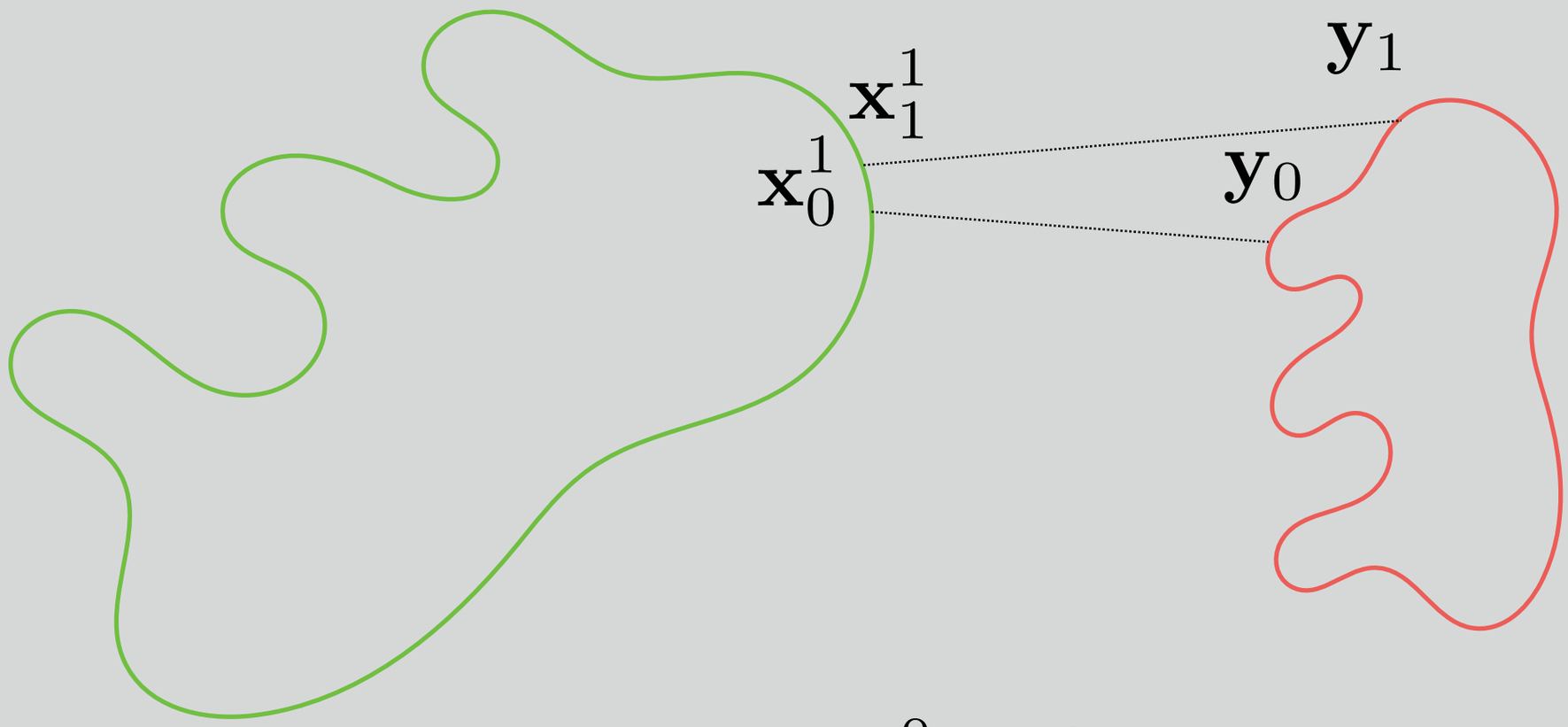


Neutral initialisation.
Initialising t to align centroids should work better!

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

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

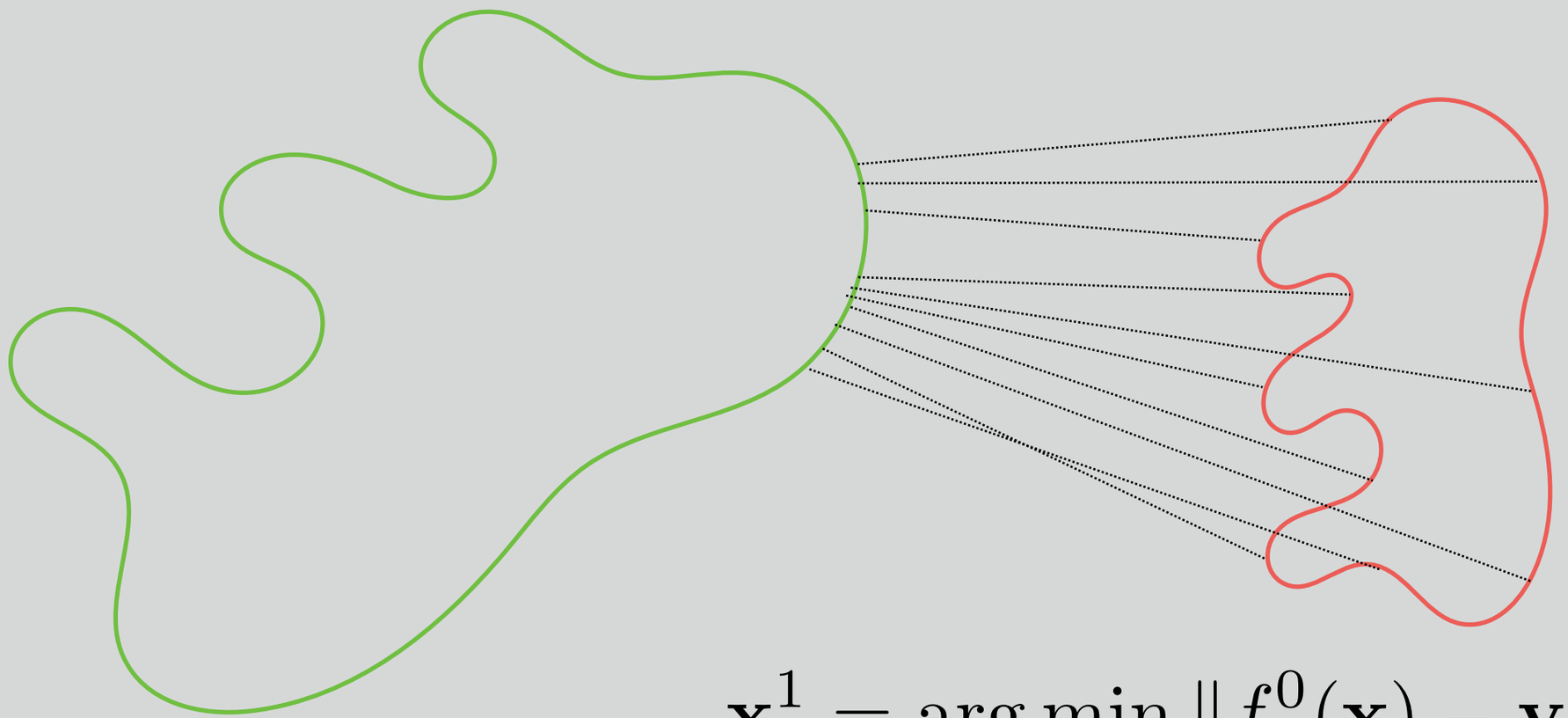
Make up reasonable correspondences



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

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

Solve for the best transformation

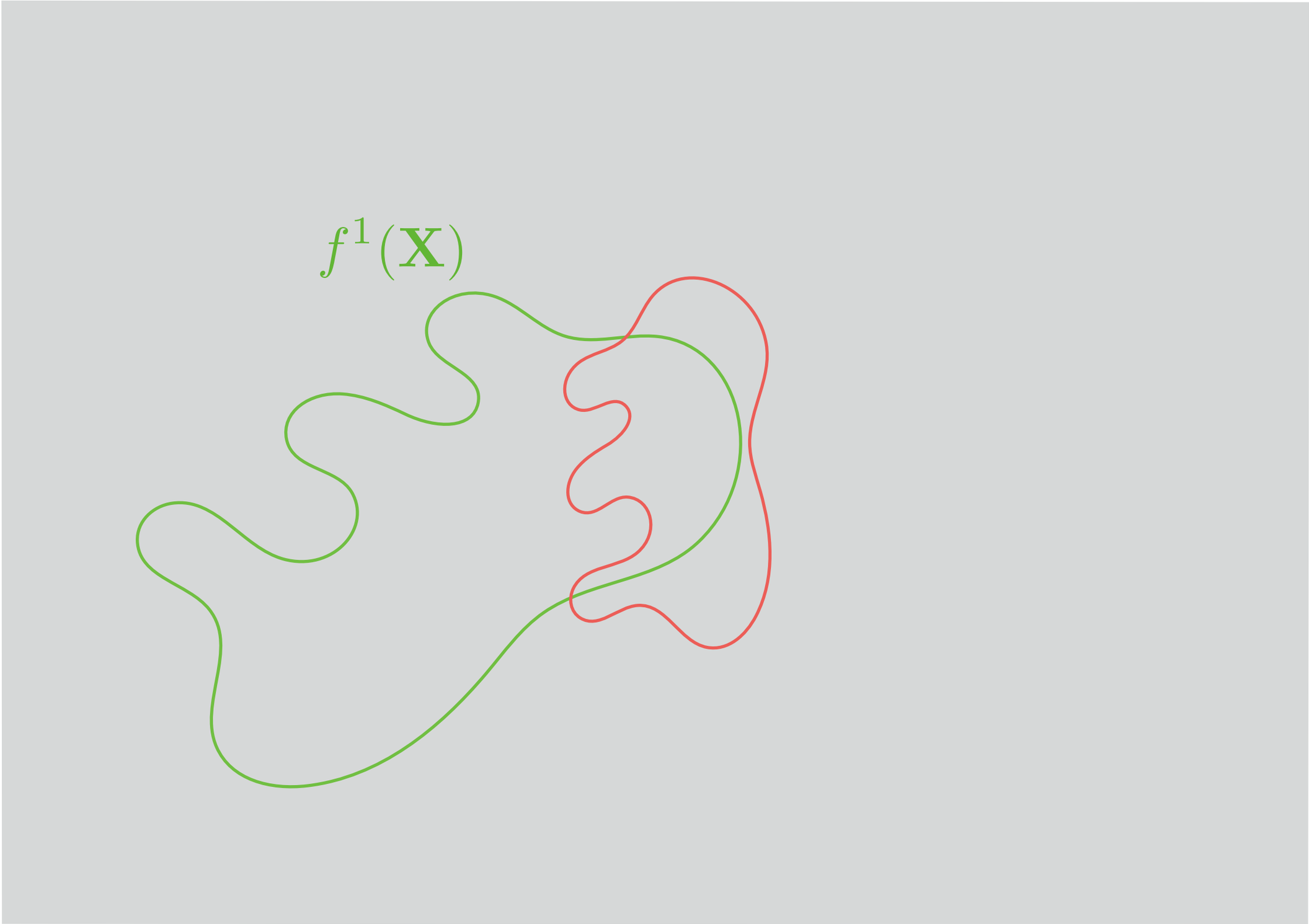


$$\mathbf{x}_i^1 = \arg\min_{\mathbf{x}\in\mathbf{X}} \|f^0(\mathbf{x}) - \mathbf{y}_i\|^2$$

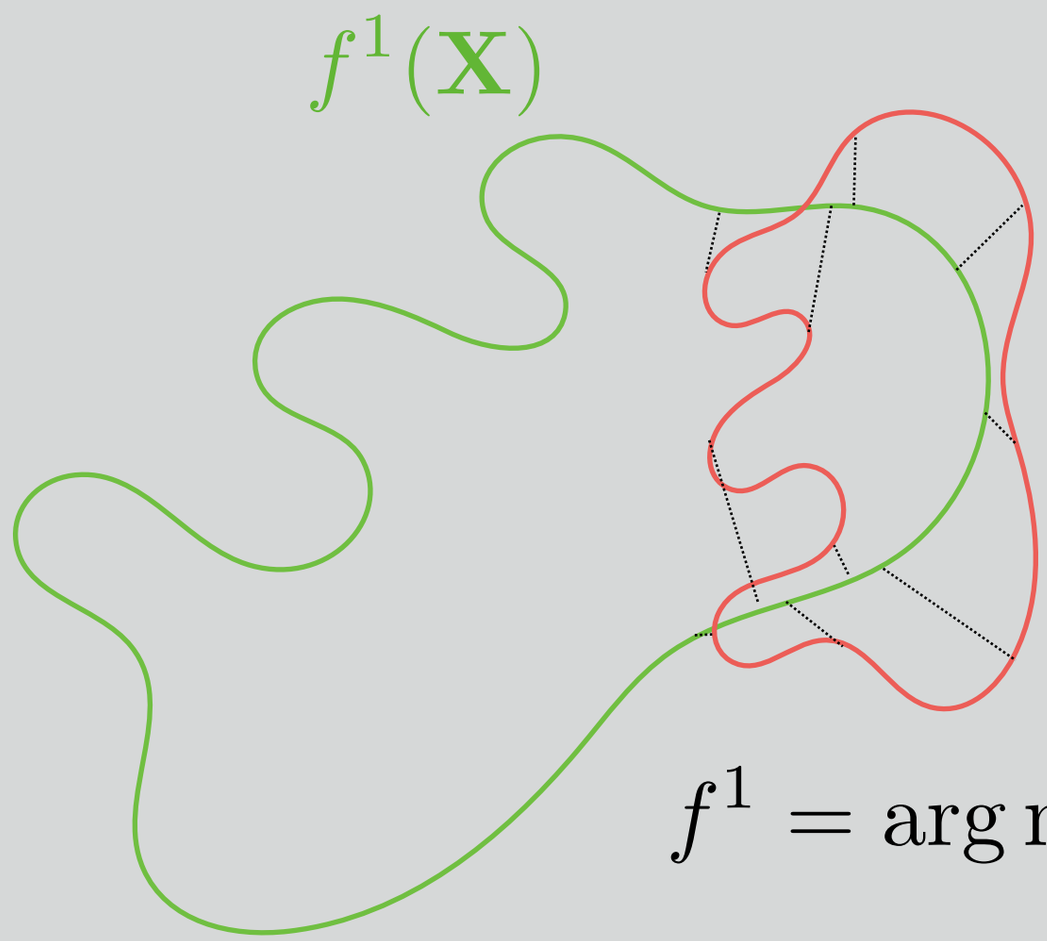
solve with procrustes

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

Apply it …



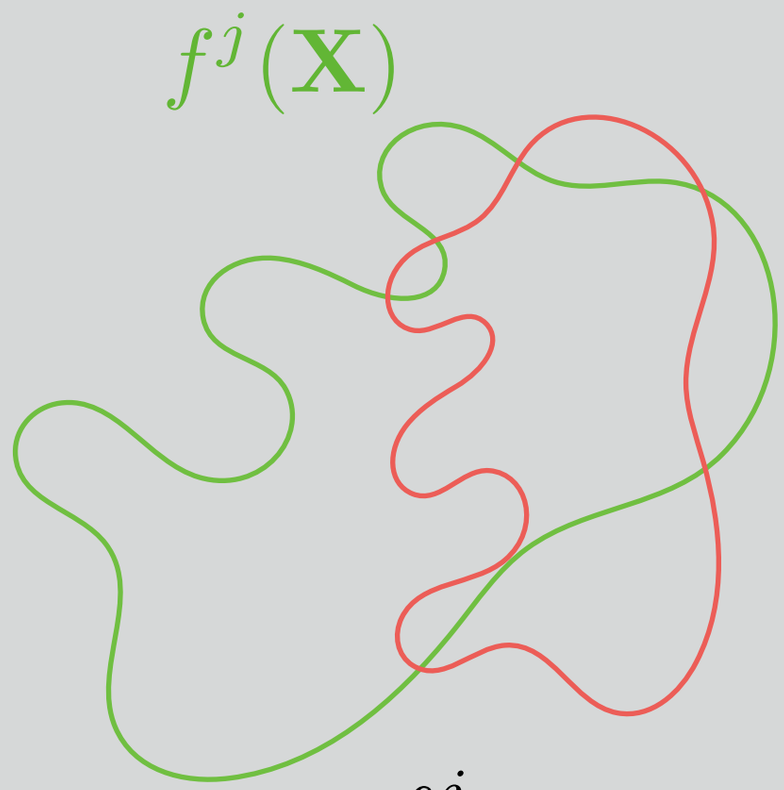
and iterate!



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

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

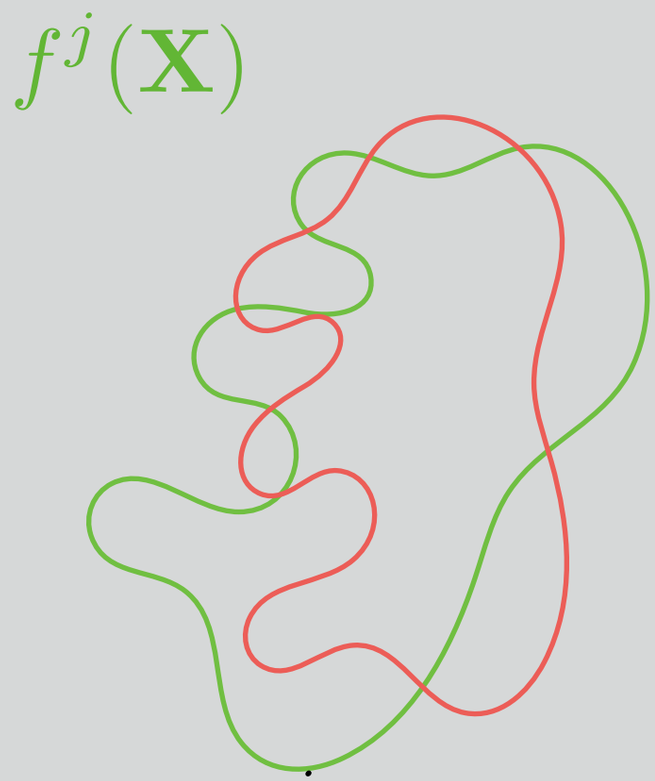
and iterate!



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

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

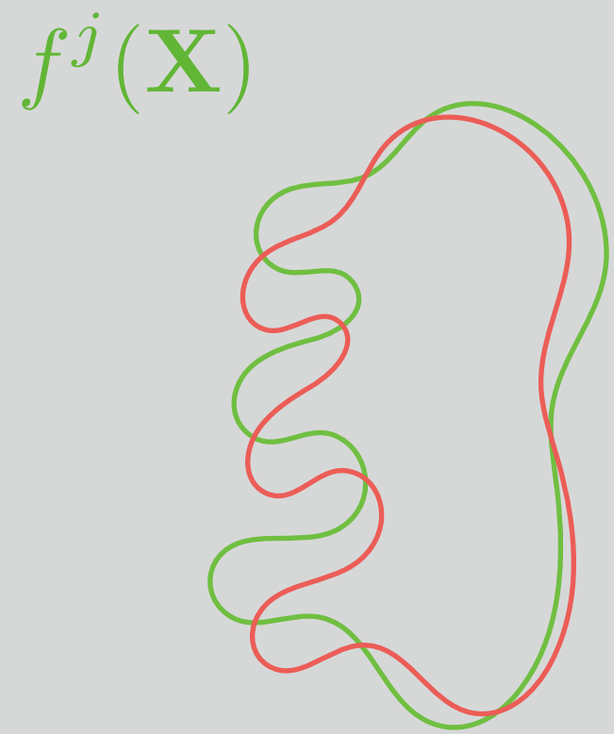
and iterate!



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

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

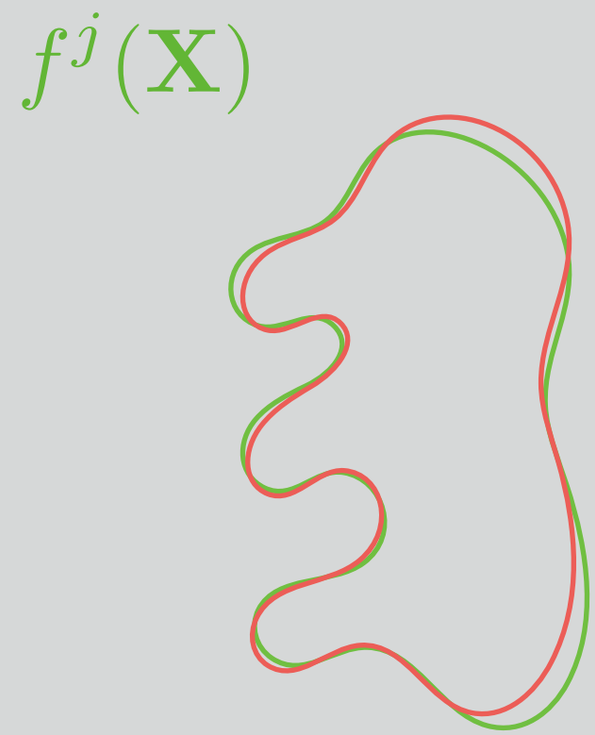
and iterate!



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

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

and iterate!



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

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

and iterate!

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

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

Iterative Closest Point (ICP)

typically better than 0

1. initialise

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

Iterative Closest Point (ICP)

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

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

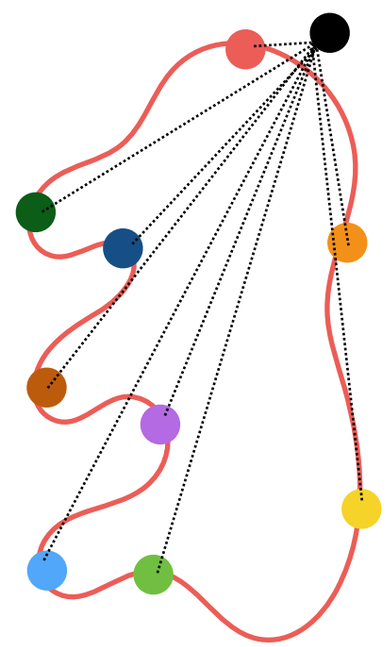
1. initialise $$f^0 = \{\mathbf{R} = \mathbf{I}, \mathbf{t} = \frac{\sum \mathbf{y}_i}{N} - \frac{\sum \mathbf{x}_i}{N}, s = 1\}$$
2. compute correspondences according to current best transform
$$\mathbf{x}_i^{j+1} = \arg\min_{\mathbf{x} \in \mathbf{X}} \|f^j(\mathbf{x}) - \mathbf{y}_i\|^2$$
3. compute 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 (go to step 2)
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

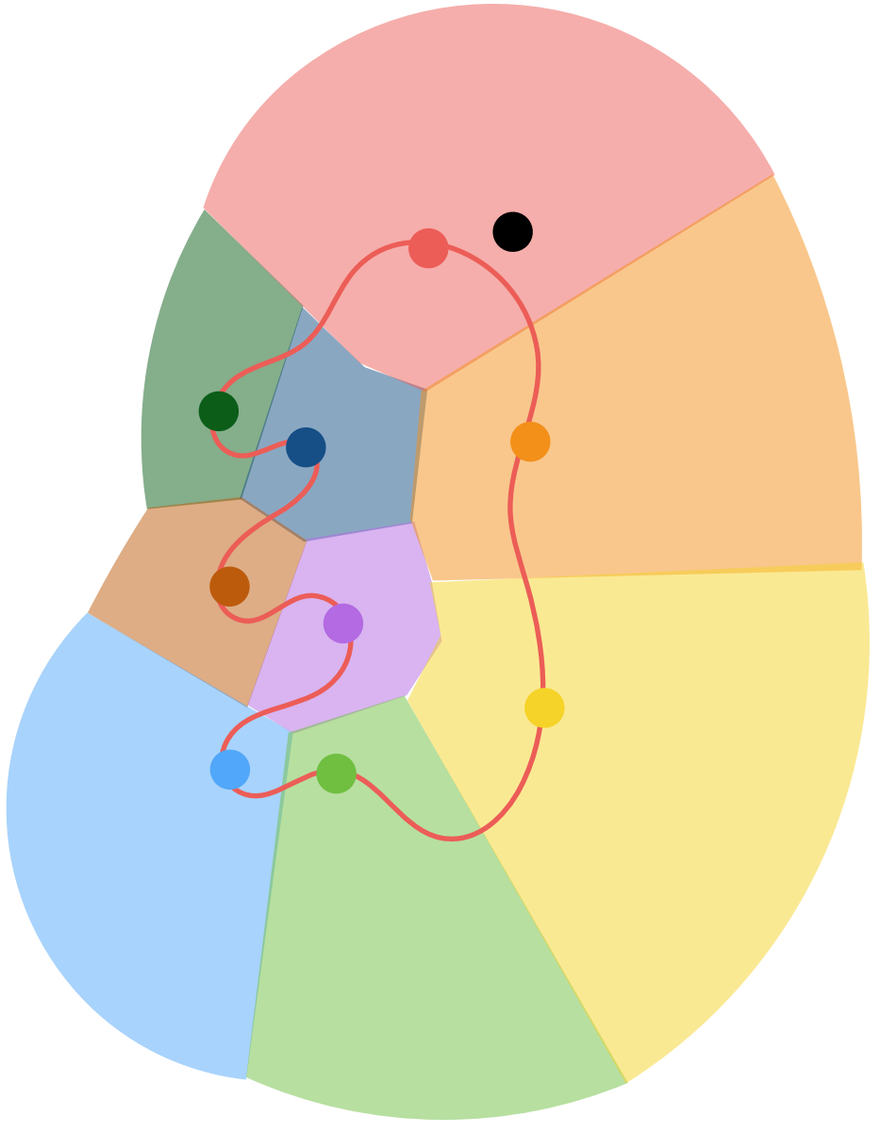
Closest points

- Brute force is n^2



Closest points

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



- Random point sampling also reduces the running time

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 → In which direction should I move?
- compute optimal transformation with Procrustes
- apply transformation
- terminate if converged, otherwise iterate

Iterative Closest Point (ICP)

- iteration j
- compute closest points → In which direction should I 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 ➡ Jacobian of distance-based energy
- ~~compute optimal transformation with Procrustes~~
  ➡ compute descent step by linearising the energy
- apply transformation
- terminate if converged, otherwise iterate

Gradient-based ICP

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

- 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

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

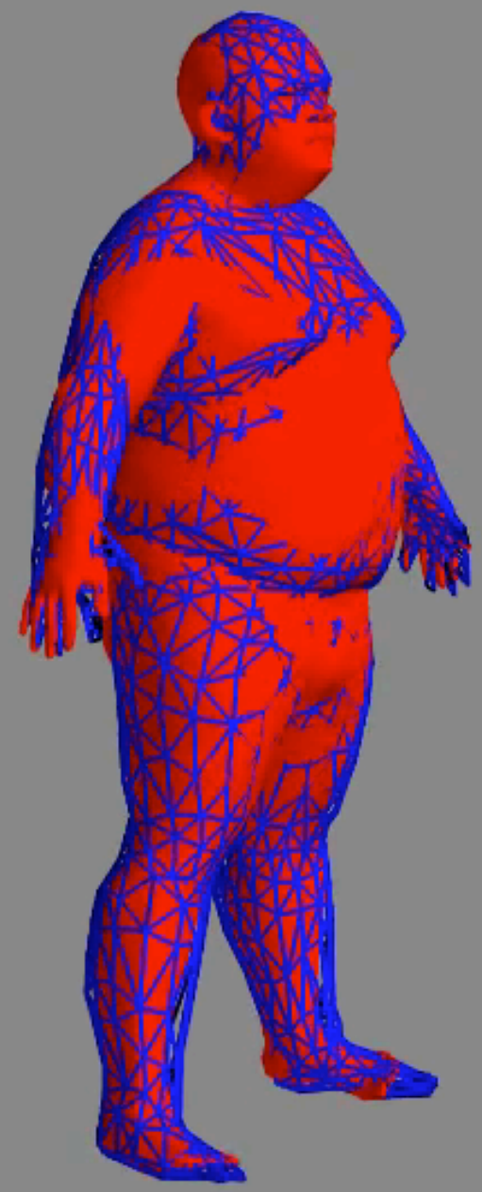
4. Apply step (gradient descent, dogleg, LM, BFGS…)

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

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

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

Try it!



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

Gradient-based ICP

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

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

- gradient: derivative of the sum of squared distances between target points and scale, rotated and translated source points, with respect to the the scale, rotation and translation

Gradient-based ICP

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

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

- gradient: derivative of the sum of squared distances between target points and scale, rotated and translated source points, with respect to the the scale, rotation and translation
- Each derivative is easy
  - Who takes the chalk and writes it down?

Gradient-based ICP

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

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

- gradient: derivative of the sum of squared distances between target points and scale, rotated and translated source points, with respect to the the scale, rotation and translation
- Each derivative is easy
  - Who takes the chalk and writes it down?
- Chain rule and automatic differentiation!

Chumpy

- https://pypi.python.org/pypi/chumpy
- Automatic differentiation compatible with numpy
- Jacobian: matrix encoding partial derivative of outputs (rows) with respect to inputs (columns)

$$\mathbf{J} = \frac{d\mathbf{b}}{d\mathbf{c}} = \begin{bmatrix} \frac{\delta b_1}{\delta c_1} & \cdots & \frac{\delta b_1}{\delta c_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta b_m}{\delta c_1} & \cdots & \frac{\delta b_m}{\delta c_n} \end{bmatrix}$$

- The Jacobians of each operation are encoded for you
- The composed Jacobian is computed with the chain rule

$$\mathbf{J}_{\mathbf{a} \circ \mathbf{b}}(\mathbf{c}) = \mathbf{J}_{\mathbf{a}}(\mathbf{b}(\mathbf{c}))\mathbf{J}_{\mathbf{b}}(\mathbf{c})$$

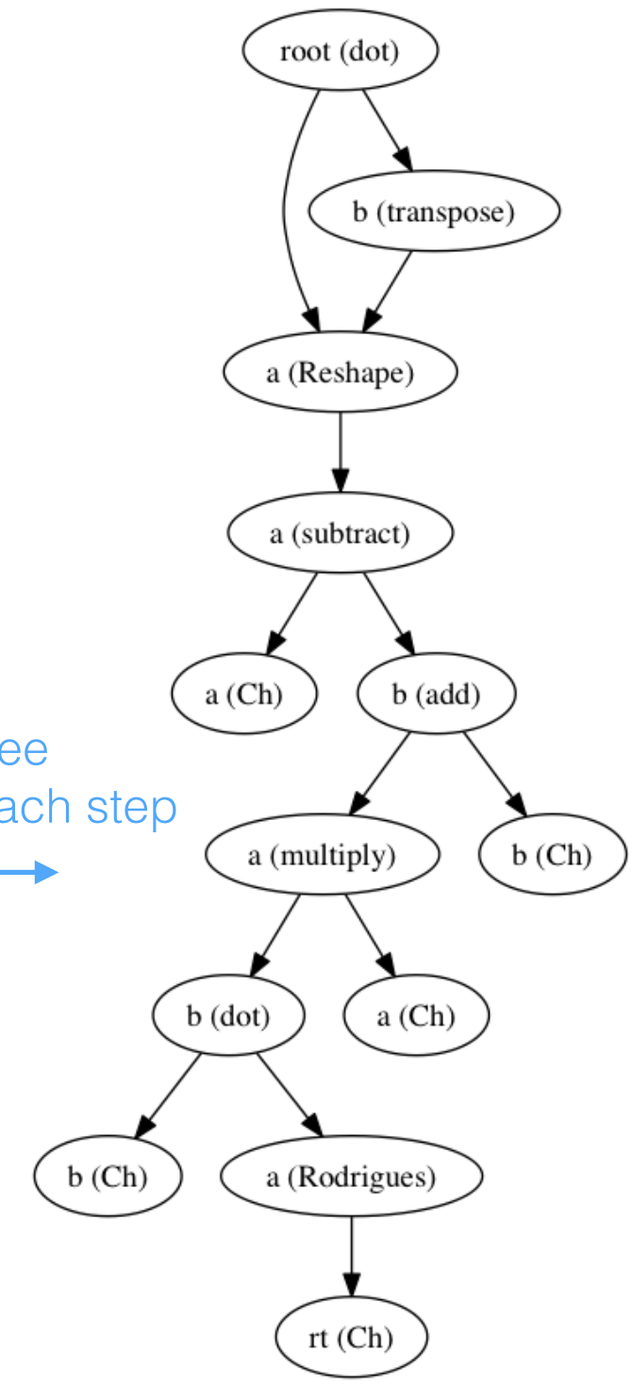
Chumpy

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

write as if it was numpy code

```
import chumpy as ch
from opendr.geometry import Rodrigues
rot = ch.zeros(3)
scale = ch.zeros(1)
trans = ch.zeros((3,1))
x = ch.random.randn(3, 100)
y = ch.random.randn(3, 100)
e = (y - (scale*Rodrigues(rot).dot(x) + trans)).ravel()
ssq = e.dot(e.T)
ssq.show_tree()
import ipdb; ipdb.set_trace()
```

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…)

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

- terminate if converged, otherwise iterate

Gradient-based ICP

- However, lots of standard ways are available in scientific libraries like scipy
  - And chumpy integrates well with it
  - Minimisation in a single line:

```
ch.minimize(fun=energy, x0=[scale, rot, trans], method='dogleg')
```

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 given optimal correspondences and for rigid alignment problems. For other problems:
- We can compute correspondences and solve for the best transformation iteratively with Iterative Closest Point (ICP)