

Fundamentals of Computer Graphics

Assignment problems

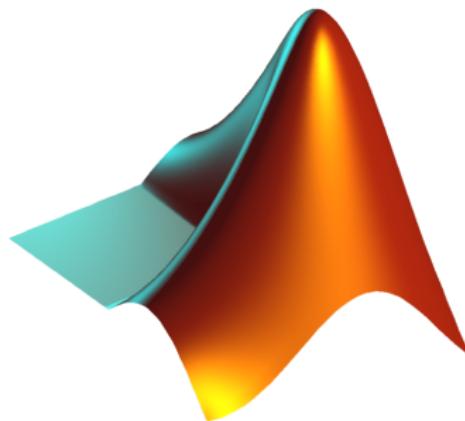
Emanuele Rodolà
rodola@di.uniroma1.it



SAPIENZA
UNIVERSITÀ DI ROMA

Exercises

- Voronoi basis



Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

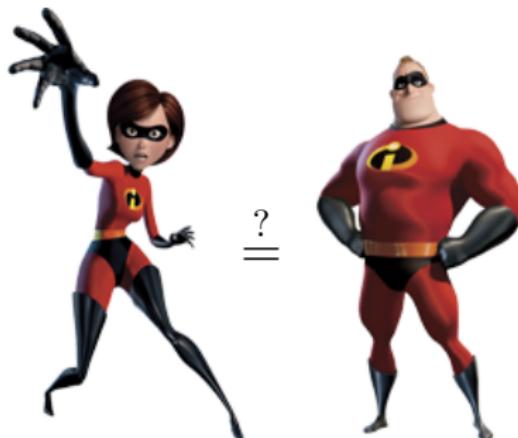
Why should we compute this distance?

Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

Why should we compute this distance? We can compare shapes

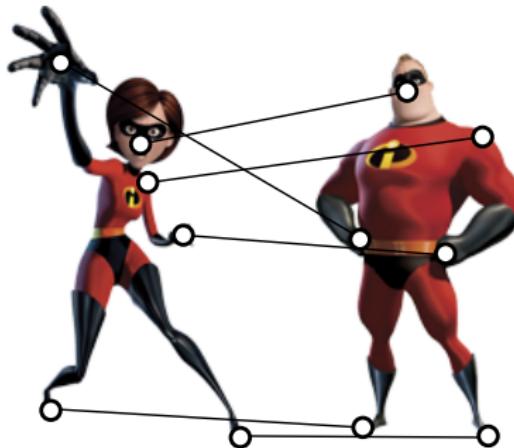


Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

[Why](#) should we compute this distance? We can find [correspondences](#)



Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

How to compute it?

Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

How to compute it?

- It depends on the task (correspondence vs retrieval)

Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

How to compute it?

- It depends on the task (correspondence vs retrieval)
- It is convenient to compute approximations of $d_{\mathcal{GH}}$

Gromov-Hausdorff distance

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \min_{\mathcal{Z}, f, g} d_{\mathcal{H}}^{\mathcal{Z}}(f(\mathcal{X}), g(\mathcal{Y}))$$

where $f : \mathcal{X} \rightarrow \mathcal{Z}$ and $g : \mathcal{Y} \rightarrow \mathcal{Z}$ are isometric embeddings

How to compute it?

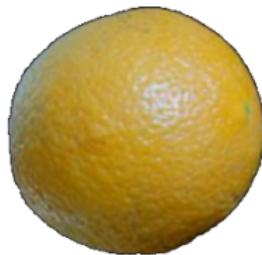
- It depends on the task (correspondence vs retrieval)
- It is convenient to compute approximations of $d_{\mathcal{GH}}$
- Still an open problem! AKA shape matching

Example: Texture transfer

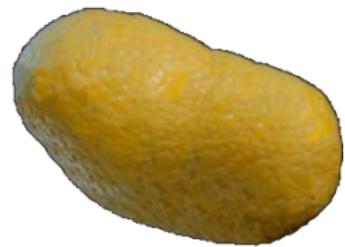
With a correspondence we can **transfer** information (e.g. texture coordinates) across shapes



\mathcal{X}



\mathcal{Y}



texture of \mathcal{Y}
transferred to \mathcal{X}

Example: Texture transfer

With a correspondence we can [transfer](#) information (e.g. texture coordinates) across shapes



Correspondence

A **correspondence** between two sets \mathcal{X} and \mathcal{Y} is a set $\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$ satisfying:

- For every $x \in \mathcal{X}$, there exists at least one $y \in \mathcal{Y}$ such that $(x, y) \in \mathcal{R}$
- For every $y \in \mathcal{Y}$, there exists at least one $x \in \mathcal{X}$ such that $(x, y) \in \mathcal{R}$

Correspondence

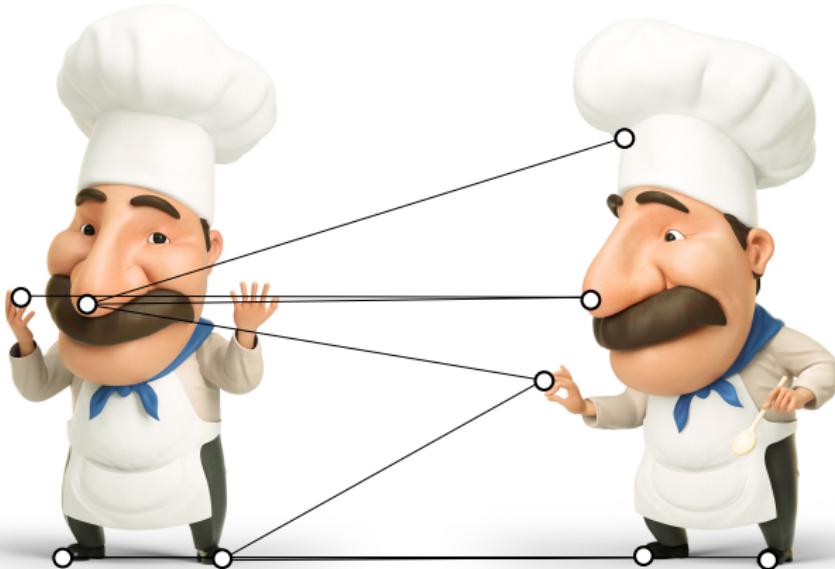
A **correspondence** between two sets \mathcal{X} and \mathcal{Y} is a set $\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$ satisfying:

- For every $x \in \mathcal{X}$, there exists at least one $y \in \mathcal{Y}$ such that $(x, y) \in \mathcal{R}$
- For every $y \in \mathcal{Y}$, there exists at least one $x \in \mathcal{X}$ such that $(x, y) \in \mathcal{R}$

In other words, this is giving **at least one** match for **all** points

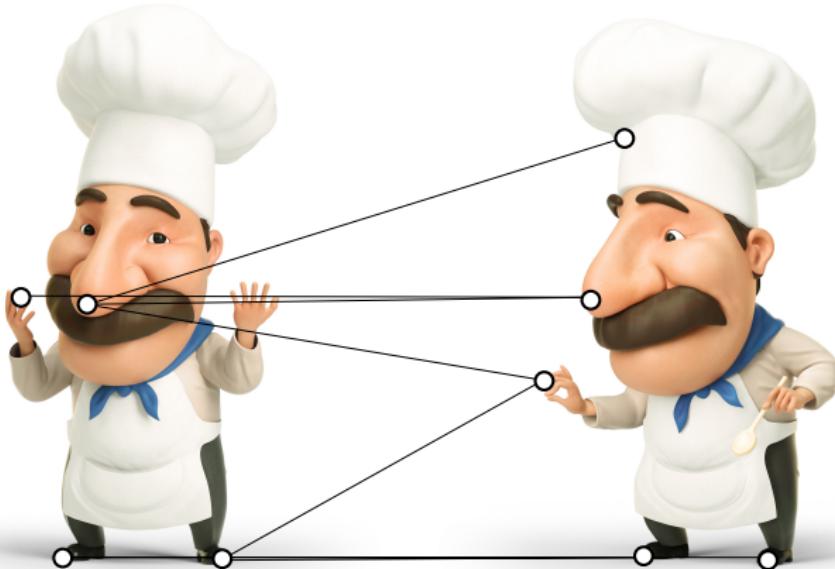
Later on, we will see that **mapping constraints** (e.g. one-to-one correspondence) are often necessary

Example: Correspondence



Correspondence: at least one match for each sample

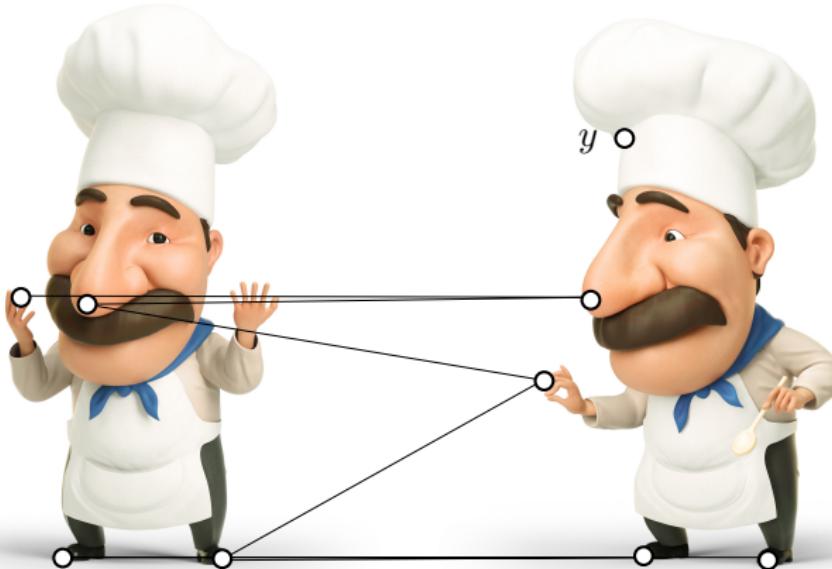
Example: Correspondence



Correspondence: at least one match for each sample

Note: The mathematical definition of correspondence doesn't have anything to do with **distortion** or **semantics**

Example: Correspondence



Not a correspondence: point $y \in \mathcal{Y}$ is not mapped

Note: The mathematical definition of correspondence doesn't have anything to do with **distortion** or **semantics**

Correspondence vs. maps

Any **surjective map** $f : \mathcal{X} \rightarrow \mathcal{Y}$ defines a correspondence:

$$\mathcal{R} = \{(x, f(x)) : x \in \mathcal{X}\}$$

Correspondence vs. maps

Any **surjective map** $f : \mathcal{X} \rightarrow \mathcal{Y}$ defines a correspondence:

$$\mathcal{R} = \{(x, f(x)) : x \in \mathcal{X}\}$$

However, not every correspondence is associated with a map:



$(x, y_1), (x, y_2) \in \mathcal{R}$, but $f(x)$ is unique if f is a map

Correspondence vs. maps

Any **surjective map** $f : \mathcal{X} \rightarrow \mathcal{Y}$ defines a correspondence:

$$\mathcal{R} = \{(x, f(x)) : x \in \mathcal{X}\}$$

However, not every correspondence is associated with a map:



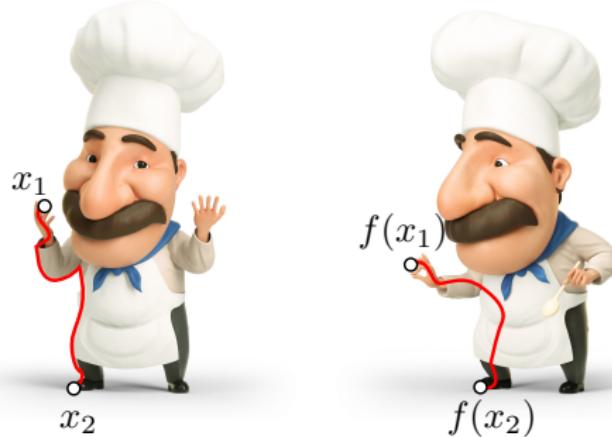
$(x, y_1), (x, y_2) \in \mathcal{R}$, but $f(x)$ is unique if f is a map

Intuitively, a correspondence is a “multi-valued” map where a single point may have more than one image

Metric distortion

For a map $f : \mathcal{X} \rightarrow \mathcal{Y}$ between metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ we define its (absolute) **distortion** as:

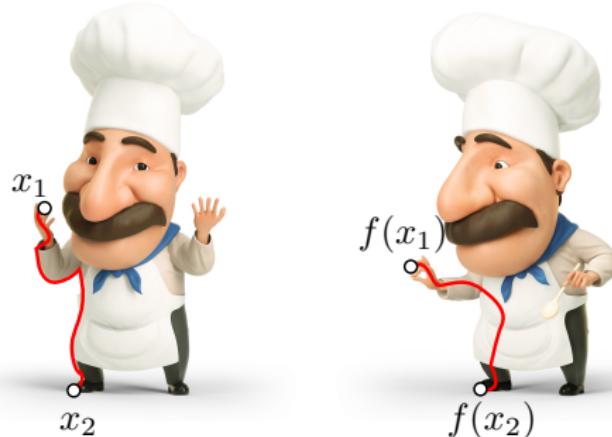
$$\text{dis}(f) = \max_{x_1, x_2 \in \mathcal{X}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(f(x_1), f(x_2))|$$



Metric distortion

For a map $f : \mathcal{X} \rightarrow \mathcal{Y}$ between metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ we define its (absolute) **distortion** as:

$$\text{dis}(f) = \max_{x_1, x_2 \in \mathcal{X}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(f(x_1), f(x_2))|$$



Similarly, for a **correspondence** \mathcal{R} we define:

$$\text{dis}(\mathcal{R}) = \max\{|d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)| : (x_1, y_1), (x_2, y_2) \in \mathcal{R}\}$$

Metric distortion

For a surjective map $f : \mathcal{X} \rightarrow \mathcal{Y}$ and the associated correspondence $\mathcal{R} = \{(x, f(x)) : x \in \mathcal{X}\}$, we have:

$$\text{dis}(f) = \text{dis}(\mathcal{R})$$

Metric distortion

For a surjective map $f : \mathcal{X} \rightarrow \mathcal{Y}$ and the associated correspondence $\mathcal{R} = \{(x, f(x)) : x \in \mathcal{X}\}$, we have:

$$\text{dis}(f) = \text{dis}(\mathcal{R})$$

If f is an **isometry** and \mathcal{R} its induced correspondence, we get:

$$\text{dis}(\mathcal{R}) = 0$$

And viceversa, zero distortion means isometry.

Metric distortion

For a surjective map $f : \mathcal{X} \rightarrow \mathcal{Y}$ and the associated correspondence $\mathcal{R} = \{(x, f(x)) : x \in \mathcal{X}\}$, we have:

$$\text{dis}(f) = \text{dis}(\mathcal{R})$$

If f is an **isometry** and \mathcal{R} its induced correspondence, we get:

$$\text{dis}(\mathcal{R}) = 0$$

And viceversa, zero distortion means isometry.

Finally, if

$$\text{dis}(f) \leq \epsilon$$

we call f an **ϵ -isometry** (which is what we see in the real world)

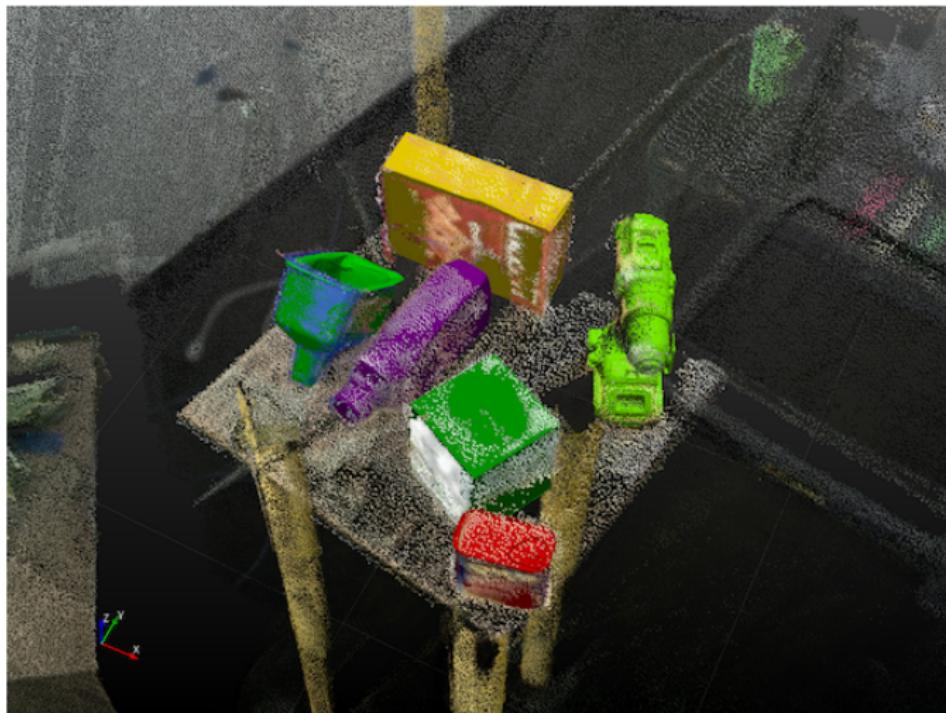
Rigid correspondence

We first consider the simple case where $d_{\mathcal{X}} = d_{\mathcal{Y}} = \|\cdot\|_2$

Rigid correspondence

We first consider the simple case where $d_{\mathcal{X}} = d_{\mathcal{Y}} = \|\cdot\|_2$

If we want zero distortion, we then expect our shapes to be **rigid**



Rigid alignment

We consider the Hausdorff distance:

$$d_{\mathcal{H}}(\mathcal{X}, \mathcal{Y}) = \max\{\max_x \text{dist}_{\mathbb{R}^3}(x, \mathcal{Y}), \max_y \text{dist}_{\mathbb{R}^3}(y, \mathcal{X})\}$$

In particular, its **non-symmetric** version:

$$d_{\mathcal{H}}(\mathcal{X}, \mathcal{Y}) = \max_x \text{dist}_{\mathbb{R}^3}(x, \mathcal{Y})$$

Rigid alignment

We consider the Hausdorff distance:

$$d_{\mathcal{H}}(\mathcal{X}, \mathcal{Y}) = \max\{\max_x \text{dist}_{\mathbb{R}^3}(x, \mathcal{Y}), \max_y \text{dist}_{\mathbb{R}^3}(y, \mathcal{X})\}$$

In particular, its **non-symmetric** version:

$$d_{\mathcal{H}}(\mathcal{X}, \mathcal{Y}) = \max_x \text{dist}_{\mathbb{R}^3}(x, \mathcal{Y})$$

Note that moving \mathcal{X} or \mathcal{Y} in the ambient space changes the value of $d_{\mathcal{H}}$

Rigid alignment

We consider the Hausdorff distance:

$$d_{\mathcal{H}}(\mathcal{X}, \mathcal{Y}) = \max\{\max_x \text{dist}_{\mathbb{R}^3}(x, \mathcal{Y}), \max_y \text{dist}_{\mathbb{R}^3}(y, \mathcal{X})\}$$

In particular, its **non-symmetric** version:

$$d_{\mathcal{H}}(\mathcal{X}, \mathcal{Y}) = \max_x \text{dist}_{\mathbb{R}^3}(x, \mathcal{Y})$$

Note that moving \mathcal{X} or \mathcal{Y} in the ambient space changes the value of $d_{\mathcal{H}}$

Rigid alignment methods minimize $d_{\mathcal{H}}$ over all rigid isometries:

$$\min_{i \in \text{iso}(\mathbb{R}^3)} d_{\mathcal{H}}(i(\mathcal{X}), \mathcal{Y})$$

Shape \mathcal{X} is moved in \mathbb{R}^3 until the Hausdorff distance to \mathcal{Y} is minimum

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} d_{\mathcal{H}}(i(\mathcal{X}), \mathcal{Y})$$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \text{dist}_{\mathbb{R}^3}(i(x), \mathcal{Y})$$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$
 - Output: **nearest neighbors** $\mathcal{N}^{(t)} = \{(i^{(t)}(x), y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$
 - Output: **nearest neighbors** $\mathcal{N}^{(t)} = \{(i^{(t)}(x), y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$
 - Q: is set \mathcal{N} a correspondence?

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$
 - Output: **nearest neighbors** $\mathcal{N}^{(t)} = \{(i^{(t)}(x), y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$
Q: is set \mathcal{N} a correspondence? No
- Compute $i^{(t+1)} = \arg \min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{(x,y) \in \mathcal{N}^{(t)}} \|i(x) - y\|_2$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$
 - Output: **nearest neighbors** $\mathcal{N}^{(t)} = \{(i^{(t)}(x), y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$
Q: is set \mathcal{N} a correspondence? No
- Compute $i^{(t+1)} = \arg \min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{(x,y) \in \mathcal{N}^{(t)}} \|i(x) - y\|_2$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$
 - Output: **nearest neighbors** $\mathcal{N}^{(t)} = \{(i^{(t)}(x), y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$
Q: is set \mathcal{N} a correspondence? No
- Compute $i^{(t+1)} = \arg \min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{(x,y) \in \mathcal{N}^{(t)}} \|i(x) - y\|_2$
 - Output: **rigid transformation** $i^{(t+1)}$

Iterative closest point (ICP)

$$\min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} \|i(x) - y\|_2$$

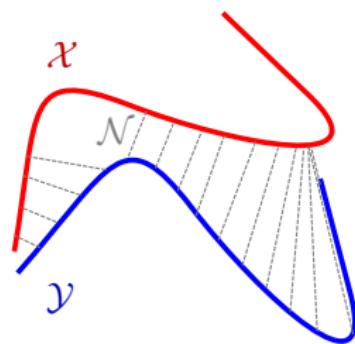
Algorithm:

- Initialize $i^{(t=0)} \in \text{iso}(\mathbb{R}^3)$ with some starting position
- Compute $\arg \min_{y \in \mathcal{Y}} \|i^{(t)}(x) - y\|_2$ for all points $x \in \mathcal{X}$
 - Output: **nearest neighbors** $\mathcal{N}^{(t)} = \{(i^{(t)}(x), y) : x \in \mathcal{X}, y \in \mathcal{Y}\}$
Q: is set \mathcal{N} a correspondence? No
- Compute $i^{(t+1)} = \arg \min_{i \in \text{iso}(\mathbb{R}^3)} \sum_{(x,y) \in \mathcal{N}^{(t)}} \|i(x) - y\|_2$
 - Output: **rigid transformation** $i^{(t+1)}$
- Repeat

Besl and McKay, "A Method for Registration of 3-D Shapes". TPAMI 14(2), 1992

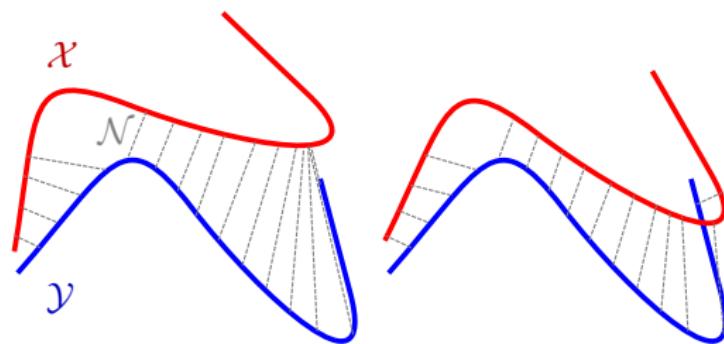
Example: ICP

At each step, we have both a correspondence and a rigid transformation



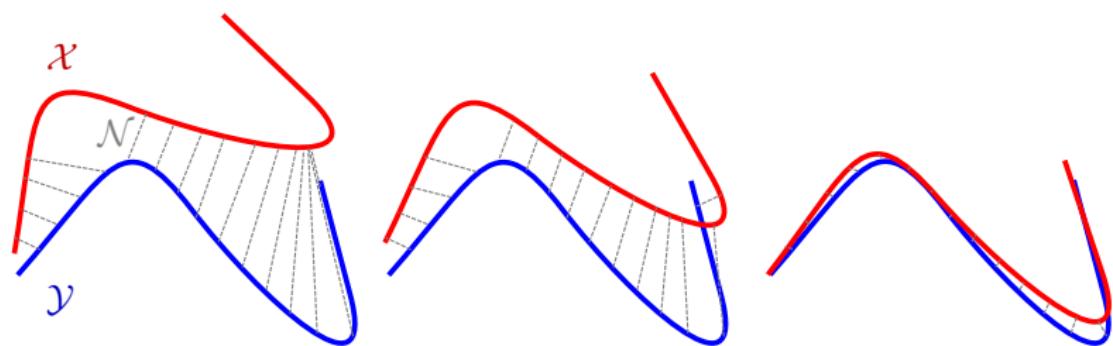
Example: ICP

At each step, we have both a correspondence and a rigid transformation



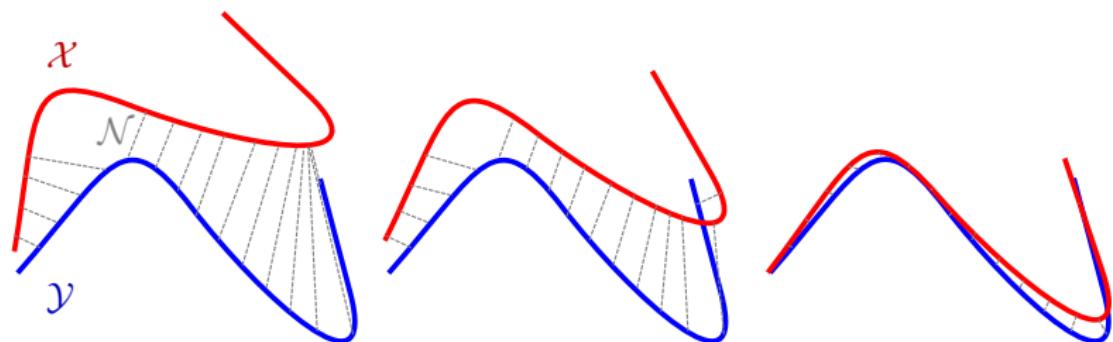
Example: ICP

At each step, we have both a correspondence and a rigid transformation



Example: ICP

At each step, we have both a correspondence and a rigid transformation



- In practice converges to local optima
- Requires a good initialization
- Only works well in the rigid setting

Correspondence and Gromov-Hausdorff

One can prove the following relationship¹:

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) < r \iff |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)| < 2r$$

for all $(x_1, y_1), (x_2, y_2) \in \mathcal{R}$. More formally, the above holds when such a correspondence $\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$ exists.

¹ Burago, Burago, Ivanov, "A course in metric geometry" (Chapter 7). AMS, 2001

Correspondence and Gromov-Hausdorff

One can prove the following relationship¹:

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) < r \iff |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)| < 2r$$

for all $(x_1, y_1), (x_2, y_2) \in \mathcal{R}$. More formally, the above holds when such a correspondence $\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$ exists.

This allows us to speak about $d_{\mathcal{GH}}$ just by using [correspondences](#) \mathcal{R} :

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R}} \text{dis}(\mathcal{R})$$

Still huge!

¹ Burago, Burago, Ivanov, "A course in metric geometry" (Chapter 7). AMS, 2001

Correspondence computation

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}} \text{dis}(\mathcal{R})$$

Correspondence computation

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}} \max_{(x_1, y_1), (x_2, y_2) \in \mathcal{R}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)|$$

Correspondence computation

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}} \max_{(x_1, y_1), (x_2, y_2) \in \mathcal{R}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)|$$

To make it more practical, restrict \mathcal{X}, \mathcal{Y} to farthest point samplings $\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}$

Correspondence computation

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}} \max_{(x_1, y_1), (x_2, y_2) \in \mathcal{R}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)|$$

To make it more practical, restrict \mathcal{X}, \mathcal{Y} to farthest point samplings $\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}$

Also, restrict the feasible set to the set of **one-to-one** correspondences;
this is done by optimizing over **permutations**:

$$d_{\mathcal{P}}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) = \frac{1}{2} \min_{\pi \in \mathcal{P}_n} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

Correspondence computation

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}} \max_{(x_1, y_1), (x_2, y_2) \in \mathcal{R}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)|$$

To make it more practical, restrict \mathcal{X}, \mathcal{Y} to farthest point samplings $\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}$

Also, restrict the feasible set to the set of **one-to-one** correspondences;
this is done by optimizing over **permutations**:

$$d_{\mathcal{P}}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) = \frac{1}{2} \min_{\pi \in \mathcal{P}_n} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

Q: how big is the set of permutations?

Correspondence computation

$$d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \min_{\mathcal{R} \subset \mathcal{X} \times \mathcal{Y}} \max_{(x_1, y_1), (x_2, y_2) \in \mathcal{R}} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_1, y_2)|$$

To make it more practical, restrict \mathcal{X}, \mathcal{Y} to farthest point samplings $\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}$

Also, restrict the feasible set to the set of **one-to-one** correspondences;
this is done by optimizing over **permutations**:

$$d_{\mathcal{P}}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) = \frac{1}{2} \min_{\pi \in \mathcal{P}_n} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_i, x_j) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

Q: how big is the set of permutations? $n!$

In fact, one can prove that $d_{\mathcal{GH}}(\mathcal{X}, \mathcal{Y}) \leq d_{\mathcal{P}}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}})$

Example: Sort by distortion

Assume we know the correspondence π , then we can directly compute:

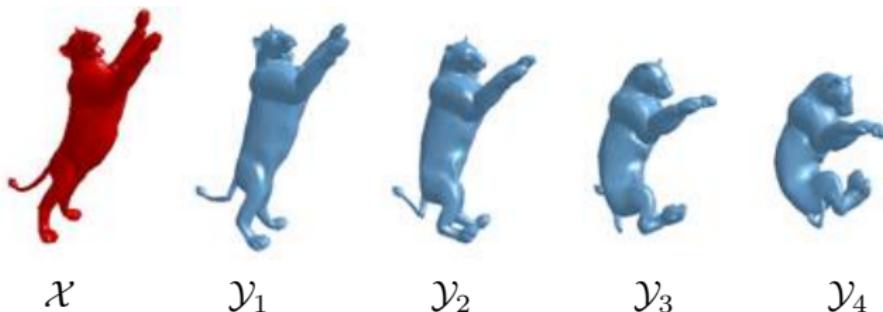
$$d_{\mathcal{P}}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) = \frac{1}{2} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

Example: Sort by distortion

Assume we know the correspondence π , then we can directly compute:

$$d_{\mathcal{P}}(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}) = \frac{1}{2} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

And we can use it to sort shapes by distortion:



where

$$d_{\mathcal{P}}(\mathcal{X}, \mathcal{Y}_1) \leq d_{\mathcal{P}}(\mathcal{X}, \mathcal{Y}_2) \leq d_{\mathcal{P}}(\mathcal{X}, \mathcal{Y}_3) \leq d_{\mathcal{P}}(\mathcal{X}, \mathcal{Y}_4)$$

Correspondence matrix

A common representation is the **correspondence matrix** $\mathbf{T} \in \{0, 1\}^{m \times n}$

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 1 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

where $T_{i,j} = 1$ means that $(x_i, y_j) \in \mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$

Correspondence matrix

A common representation is the **correspondence matrix** $\mathbf{T} \in \{0, 1\}^{m \times n}$

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 1 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

where $T_{i,j} = 1$ means that $(x_i, y_j) \in \mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$

\mathbf{T} encodes in the **standard basis** a linear map between **function spaces**:

$$T : (\mathcal{F}(\mathcal{X}) \rightarrow \mathbb{R}) \rightarrow (\mathcal{F}(\mathcal{Y}) \rightarrow \mathbb{R})$$

Correspondence matrix

A common representation is the **correspondence matrix** $\mathbf{T} \in \{0, 1\}^{m \times n}$

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 1 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

where $T_{i,j} = 1$ means that $(x_i, y_j) \in \mathcal{R} \subset \mathcal{X} \times \mathcal{Y}$

\mathbf{T} encodes in the **standard basis** a linear map between **function spaces**:

$$T : (\mathcal{F}(\mathcal{X}) \rightarrow \mathbb{R}) \rightarrow (\mathcal{F}(\mathcal{Y}) \rightarrow \mathbb{R})$$

In other words, it maps functions to functions

Permutation matrix

A **permutation matrix** $\mathbf{P} \in \{0, 1\}^{n \times n}$ is a special case where each row and each column have **exactly one** value equal to 1:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

Permutation matrix

A **permutation matrix** $\mathbf{P} \in \{0, 1\}^{n \times n}$ is a special case where each row and each column have **exactly one** value equal to 1:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

This can be also written as $\mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^\top\mathbf{1} = \mathbf{1}$

Permutation matrix

A **permutation matrix** $\mathbf{P} \in \{0, 1\}^{n \times n}$ is a special case where each row and each column have **exactly one** value equal to 1:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

This can be also written as $\mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^\top\mathbf{1} = \mathbf{1}$

Permutations encode **one-to-one** correspondences, meaning that they map **indicator functions** to indicator functions

Permutation matrix

A **permutation matrix** $\mathbf{P} \in \{0, 1\}^{n \times n}$ is a special case where each row and each column have **exactly one** value equal to 1:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

This can be also written as $\mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^\top\mathbf{1} = \mathbf{1}$

Permutations encode **one-to-one** correspondences, meaning that they map **indicator functions** to indicator functions

In turn, one-to-one correspondences are associated to **bijective maps**
 $f : \mathcal{X} \rightarrow \mathcal{Y}$

Correspondence problem

$$\pi^* = \arg \min_{\pi \in \mathcal{P}_n} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

is commonly called **minimum distortion correspondence** problem

Correspondence problem

$$\pi^* = \arg \min_{\pi \in \mathcal{P}_n} \max_{1 \leq i, j \leq n} |d_{\mathcal{X}}(x_1, x_2) - d_{\mathcal{Y}}(y_{\pi i}, y_{\pi j})|$$

is commonly called **minimum distortion correspondence** problem

The metric distortion terms can be encoded into a matrix $\mathbf{C} \in \mathbb{R}^{n^2 \times n^2}$:

$$C_{(i\ell)(jm)} = |d_{\mathcal{X}}(x_i, x_j) - d_{\mathcal{Y}}(y_\ell, y_m)|$$

where $(i\ell)$ and (jm) identify the matches $(x_i, y_\ell), (x_j, y_m) \in \mathcal{R}$

$$\mathbf{C} = \begin{pmatrix} (x_1, y_1) & (x_3, y_5) & \cdots & (x_i, y_\ell) & \cdots \\ & & & \vdots & \\ & & & \vdots & \\ & \cdots & \cdots & \cdots & |d_{\mathcal{X}}(x_i, x_j) - d_{\mathcal{Y}}(y_\ell, y_m)| & \cdots \\ & & & & \vdots & \\ \end{pmatrix} \begin{pmatrix} (x_1, y_1) \\ (x_3, y_5) \\ \vdots \\ (x_j, y_m) \\ \vdots \end{pmatrix}$$

L_p Gromov-Hausdorff

$$\frac{1}{2} \min_{\mathbf{P} \in \mathcal{P}} \max_{i,j,\ell,m} C_{(i\ell)(jm)} P_{ij} P_{\ell m}$$

L_p Gromov-Hausdorff

$$\frac{1}{2} \min_{\mathbf{P} \in \mathcal{P}} \sum_{i,j,\ell,m} C_{(i\ell)(jm)} P_{ij} P_{\ell m}$$

L_p Gromov-Hausdorff

$$\frac{1}{2} \min_{\mathbf{P} \in \mathcal{P}} \sum_{i,j,\ell,m} C_{(i\ell)(jm)} P_{ij} P_{\ell m}$$

Or, more in general, we can use the L_p distortion

$$C_{(i\ell)(jm)}^{(p)} = |d_{\mathcal{X}}(x_i, x_j) - d_{\mathcal{Y}}(y_\ell, y_m)|^p$$

yielding the minimization problem

$$\frac{1}{2} \min_{\mathbf{P} \in \mathcal{P}} \sum_{i,j,\ell,m} C_{(i\ell)(jm)}^{(p)} P_{ij} P_{\ell m}$$

Quadratic assignment problem

Rewriting in matrix notation, we get:

$$\begin{aligned} \min_{\mathbf{P} \in \{0,1\}^{n \times n}} & \text{vec}(\mathbf{P})^\top \mathbf{C} \text{vec}(\mathbf{P}) \\ \text{s.t. } & \mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^\top \mathbf{1} = \mathbf{1} \end{aligned}$$

also known as quadratic assignment problem (QAP)

Quadratic assignment problem

Rewriting in matrix notation, we get:

$$\begin{aligned} & \min_{\mathbf{P} \in \{0,1\}^{n \times n}} \text{vec}(\mathbf{P})^\top \mathbf{C} \text{vec}(\mathbf{P}) \\ \text{s.t. } & \mathbf{P}\mathbf{1} = \mathbf{1}, \mathbf{P}^\top \mathbf{1} = \mathbf{1} \end{aligned}$$

also known as **quadratic assignment problem** (QAP)

- The problem is **quadratic** in \mathbf{P}
- It is **NP-hard**
- Still an open problem (a survey might be a final project)

Continuous relaxation

Replace $\{0, 1\}$ with continuous values in $[0, 1]$

$$\begin{aligned} & \min_{\mathbf{S} \in [0,1]^{n \times n}} \text{vec}(\mathbf{S})^\top \mathbf{C} \text{vec}(\mathbf{S}) \\ \text{s.t. } & \mathbf{S}\mathbf{1} = \mathbf{1}, \mathbf{S}^\top \mathbf{1} = \mathbf{1} \end{aligned}$$

Continuous relaxation

Replace $\{0, 1\}$ with continuous values in $[0, 1]$

$$\begin{aligned} & \min_{\mathbf{S} \in [0,1]^{n \times n}} \text{vec}(\mathbf{S})^\top \mathbf{C} \text{vec}(\mathbf{S}) \\ \text{s.t. } & \mathbf{S}\mathbf{1} = \mathbf{1}, \mathbf{S}^\top \mathbf{1} = \mathbf{1} \end{aligned}$$

We are replacing permutations by **doubly-stochastic** matrices:

$$\mathbf{S} = \begin{pmatrix} 0.2 & 0.3 & \cdots & 0.5 \\ 0.1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0.7 & 0 & \cdots & 0.1 \end{pmatrix}$$

The underlying map is often called a **soft map**

Continuous relaxation

Replace $\{0, 1\}$ with continuous values in $[0, 1]$

$$\begin{aligned} & \min_{\mathbf{S} \in [0,1]^{n \times n}} \text{vec}(\mathbf{S})^\top \mathbf{C} \text{vec}(\mathbf{S}) \\ \text{s.t. } & \mathbf{S}\mathbf{1} = \mathbf{1}, \mathbf{S}^\top \mathbf{1} = \mathbf{1} \end{aligned}$$

We are replacing permutations by **doubly-stochastic** matrices:

$$\mathbf{S} = \begin{pmatrix} 0.2 & 0.3 & \cdots & 0.5 \\ 0.1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 0.7 & 0 & \cdots & 0.1 \end{pmatrix}$$

The underlying map is often called a **soft map**

Doubly-stochastic matrices are **convex combinations** of permutations:

$$\mathbf{S} = t\mathbf{P}_1 + (1 - t)\mathbf{P}_2, \quad t \in [0, 1]$$

Example: Soft maps

Each row / column of a soft map can be interpreted as a probability distribution on the shape:



Exercise: Sort by distortion

Load the shapes `tr_reg_000`, `tr_reg_001`, `tr_reg_002`, and `tr_reg_003` (download from course website).

- The **ground-truth** correspondence $T^{\mathcal{X}, \mathcal{Y}}$ between each pair of shapes $(\mathcal{X}, \mathcal{Y})$ is the **identity map**

Exercise: Sort by distortion

Load the shapes `tr_reg_000`, `tr_reg_001`, `tr_reg_002`, and `tr_reg_003` (download from course website).

- The **ground-truth** correspondence $T^{\mathcal{X}, \mathcal{Y}}$ between each pair of shapes $(\mathcal{X}, \mathcal{Y})$ is the **identity map**
- Compute for each shape a **farthest point sampling** (FPS) of 100 points starting from the same seed

Exercise: Sort by distortion

Load the shapes `tr_reg_000`, `tr_reg_001`, `tr_reg_002`, and `tr_reg_003` (download from course website).

- The **ground-truth** correspondence $T^{\mathcal{X}, \mathcal{Y}}$ between each pair of shapes $(\mathcal{X}, \mathcal{Y})$ is the **identity map**
- Compute for each shape a **farthest point sampling** (FPS) of 100 points starting from the same seed
- Compute the **metric distortion** of the maps, restricted to the fps, from shape `tr_reg_000` to all the others (three maps in total)

Exercise: Sort by distortion

Load the shapes `tr_reg_000`, `tr_reg_001`, `tr_reg_002`, and `tr_reg_003` (download from course website).

- The **ground-truth** correspondence $T^{\mathcal{X}, \mathcal{Y}}$ between each pair of shapes $(\mathcal{X}, \mathcal{Y})$ is the **identity map**
- Compute for each shape a **farthest point sampling** (FPS) of 100 points starting from the same seed
- Compute the **metric distortion** of the maps, restricted to the fps, from shape `tr_reg_000` to all the others (three maps in total)
- Sort the shapes by increasing distortion

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$
- Implement the **ICP algorithm** to rigidly align the two sets of samples:

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$
- Implement the **ICP algorithm** to rigidly align the two sets of samples:
 - Initialize with the 3×3 identity map in the standard basis

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$
- Implement the **ICP algorithm** to rigidly align the two sets of samples:
 - Initialize with the 3×3 identity map in the standard basis
 - For the nearest neighbors, you can use `knnsearch()` in Matlab

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$
- Implement the **ICP algorithm** to rigidly align the two sets of samples:
 - Initialize with the 3×3 identity map in the standard basis
 - For the nearest neighbors, you can use `knnsearch()` in Matlab
 - Encode them in two $n \times 3$ matrices \mathbf{X} and \mathbf{Y} where the rows correspond

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$
- Implement the **ICP algorithm** to rigidly align the two sets of samples:
 - Initialize with the 3×3 identity map in the standard basis
 - For the nearest neighbors, you can use `knnsearch()` in Matlab
 - Encode them in two $n \times 3$ matrices \mathbf{X} and \mathbf{Y} where the rows correspond
 - Compute the isometry T as $[U, \tilde{V}] = \text{svd}(\mathbf{X}' * \mathbf{Y})$; $T = U * V'$

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Exercise: ICP

Load the shapes bun000 and bun045 (download from course website).

- Compute a **FPS** of $n =$ a few hundred points for each scan
- Center each set of samples at $(0, 0, 0)$
- Implement the **ICP algorithm** to rigidly align the two sets of samples:
 - Initialize with the 3×3 identity map in the standard basis
 - For the nearest neighbors, you can use `knnsearch()` in Matlab
 - Encode them in two $n \times 3$ matrices \mathbf{X} and \mathbf{Y} where the rows correspond
 - Compute the isometry T as $[U, \tilde{V}] = \text{svd}(\mathbf{X}' * \mathbf{Y})$; $T = U * V'$;
- Print the alignment error across the iterations

Visualize the alignment with the entire shapes (not just the farthest point samplings) **without** running ICP on the entire shapes

Suggested reading

- An extensive treatment of Gromov-Hausdorff distances and correspondences is in Chapter 7 of:
Burago, Burago, Ivanov, “A course in metric geometry”. AMS, 2001
- A very accessible survey of ICP and its variants:
Rusinkiewicz and Levoy, “Efficient variants of the ICP algorithm” .
Proc. 3DIM, 2011