Optimal transport with 3D shapes

Jean Feydy
HeKA team, Inria Paris
Inserm, Université Paris-Cité

6th of December, 2023
SMAI-SIGMA day
Laboratoire Jacques-Louis Lions, Jussieu

Who am I?

Background in **mathematics** and **data sciences**:

2012–2016 ENS Paris, mathematics.

2014–2015 M2 mathematics, vision, learning at ENS Cachan.

2016–2019 PhD thesis in **medical imaging** with Alain Trouvé at ENS Cachan.

2019–2021 **Geometric deep learning** with Michael Bronstein at Imperial College.

2021+ **Medical data analysis** in the HeKA INRIA team (Paris).

HeKA : a translational research team for public health

Hôpitaux

Inria Inserm

Universités

My main motivation

Develop **robust and efficient** software that **stimulates other researchers**:

1. Speed up **geometric machine learning** on GPUs:

   $\Longrightarrow$ **pyKeOps** library for distance and kernel matrices, 500k+ downloads.

2. Scale up **pharmacovigilance** to the full French population:

   $\Longrightarrow$ **survivalGPU**, a fast re-implementation of the R survival package.

3. Ease access to modern statistical **shape analysis**:

   $\Longrightarrow$ **GeomLoss**, truly scalable optimal transport in Python.

   $\Longrightarrow$ **scikit-shapes**, to be released soon.

Today's talk – assuming that you would enjoy some applied maths

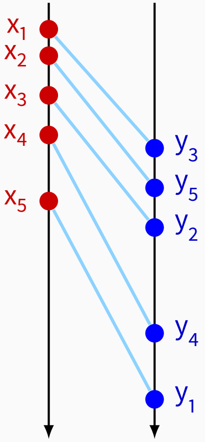
1. The **optimal transport** problem.
2. Efficient discrete **solvers**.
3. **Applications** and **open** problems.

Optimal transport?

Optimal transport (OT) generalizes sorting to spaces of dimension $D > 1$

If $A = (x_1, \dots, x_N)$ and $B = (y_1, \dots, y_N)$ are two clouds of N points in $\mathbb{R}^D$, we define:

$$\mathrm{OT}(A, B) = \min_{\sigma \in \mathcal{S}_N} \frac{1}{2N} \sum_{i=1}^{N} \| x_i - y_{\sigma(i)} \|^2$$



Generalizes **sorting** to metric spaces.

Linear problem on the permutation matrix P:

$$\mathrm{OT}(A, B) = \min_{P \in \mathbb{R}^{N \times N}} \frac{1}{2N} \sum_{i,j=1}^{N} P_{i,j} \cdot \| x_i - y_j \|^2 ,$$

$$\text{s.t.} \quad P_{i,j} \geqslant 0 \qquad \underbrace{\textstyle\sum_j P_{i,j} = 1}_{\text{Each source point...}} \qquad \underbrace{\textstyle\sum_i P_{i,j} = 1}_{\text{is transported onto the target.}} .$$

Practical use

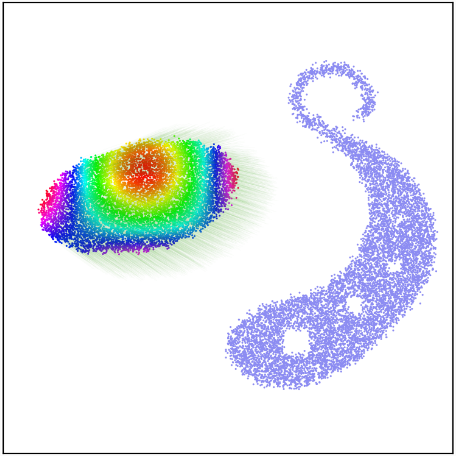
Alternatively, we understand OT as:

- Nearest neighbor **projection** + **incompressibility** constraint.
- Fundamental example of **linear optimization** over the transport plan $\mathrm{P}_{i,j}$.

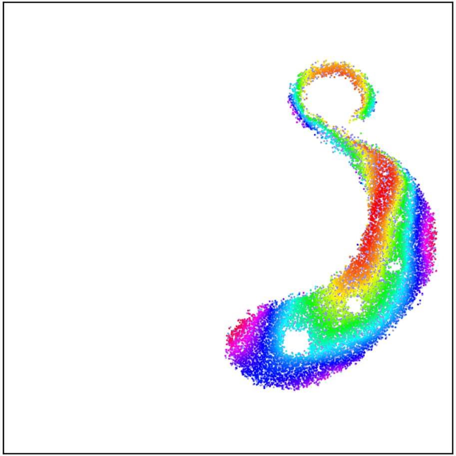
This theory induces two main quantities:

- The transport plan $\mathrm{P}_{i,j} \simeq$ the optimal mapping $x_i \mapsto y_{\sigma(i)}$.
- The "Wasserstein" distance $\sqrt{\mathrm{OT}(\mathrm{A}, \mathrm{B})}$.

The optimal transport plan

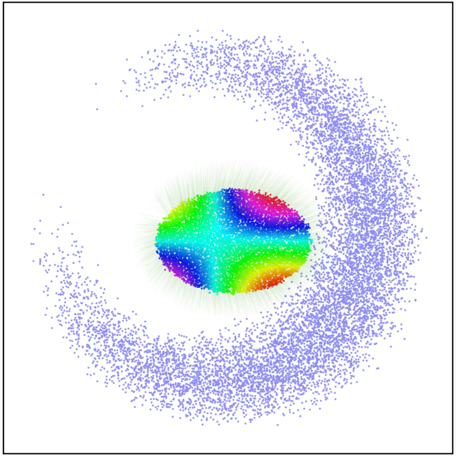


Before

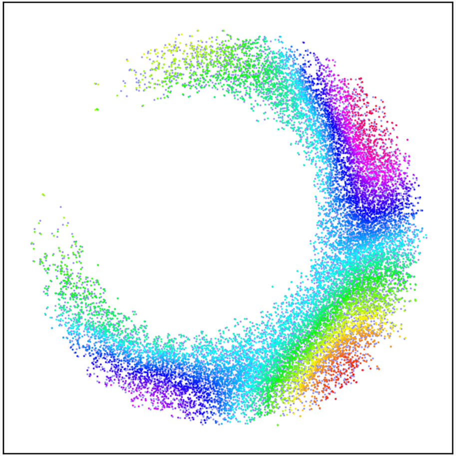


After

The optimal transport plan

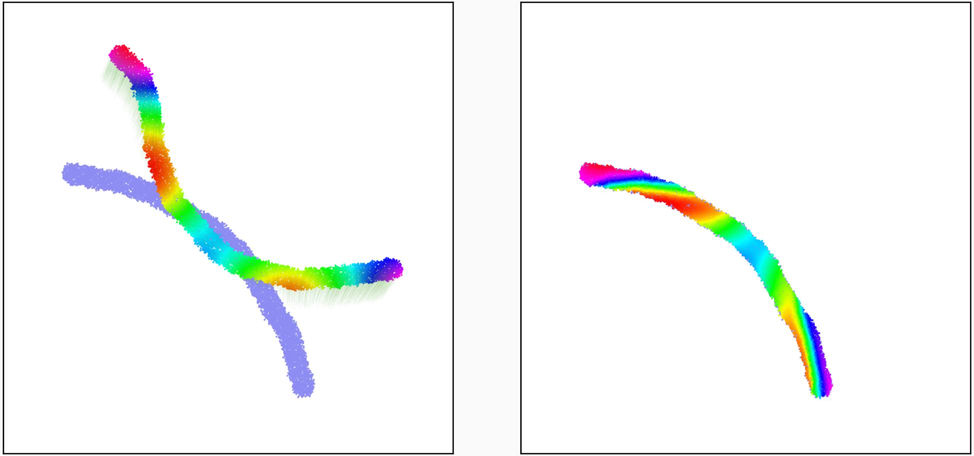


Before



After

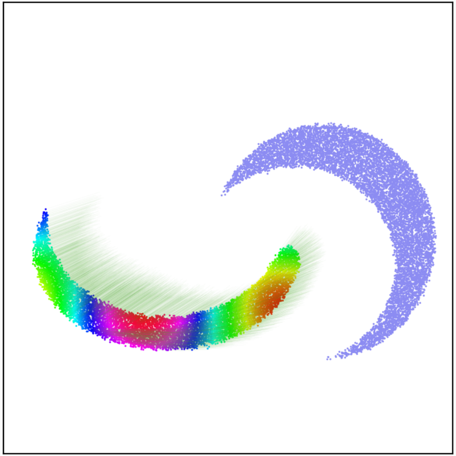
The optimal transport plan



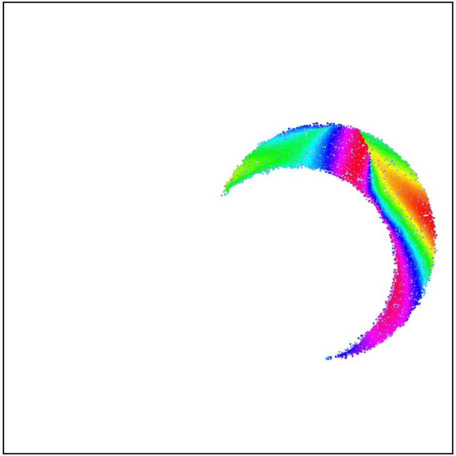
Before

After

The optimal transport plan



Before

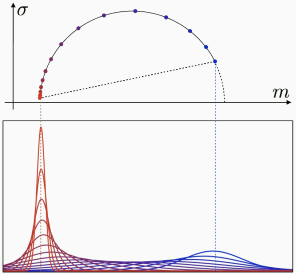


After

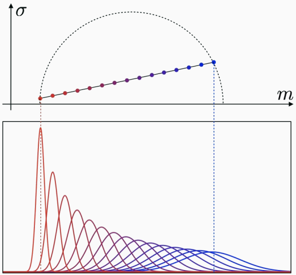
OT induces a geometry-aware distance between probability distributions [PC18]

Gauss map $\mathcal{N} : (m, \sigma) \in \mathbb{R} \times \mathbb{R}_{\geqslant 0} \;\; \mapsto \;\; \mathcal{N}(m, \sigma) \in \mathbb{P}(\mathbb{R})$.

If the space of **probability distributions** $\mathbb{P}(\mathbb{R})$ is endowed with a given metric, what is the "pull-back" geometry on the space of **parameters** (m, σ)?



Fisher-Rao ($\simeq$ relative entropy) on $\mathcal{N}(m, \sigma)$
$\rightarrow$ Hyperbolic **Poincaré** metric on (m, σ).



OT on $\mathcal{N}(m, \sigma)$
$\rightarrow$ Flat **Euclidean** metric on (m, σ).

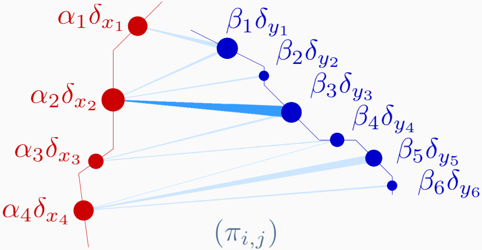
How should we solve the OT problem?

Duality: central planning with NM variables $\simeq$ outsourcing with N + M variables

$$\mathrm{OT}(\mathrm{A},\mathrm{B}) = \min_{\pi} \langle \pi , \mathrm{C} \rangle, \text{ with } \mathrm{C}(x_i, y_j) = \tfrac{1}{p}\|x_i - y_j\|^p \quad \longrightarrow \quad \text{Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \quad \pi\mathbf{1} = \mathrm{A}, \quad \pi^\top\mathbf{1} = \mathrm{B}$$

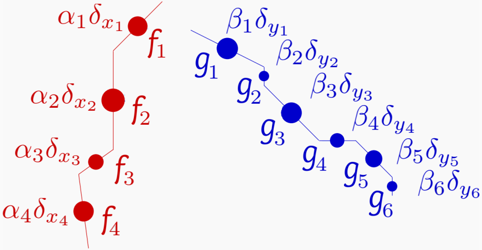
$\sum_{i,j} \pi_{i,j}\, \mathrm{C}(x_i, y_j)$

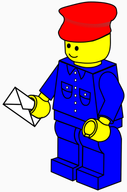


Duality: central planning with NM variables $\simeq$ outsourcing with N + M variables

$$\mathrm{OT}(\mathsf{A},\mathsf{B}) \;=\; \min_{\pi}\ \langle \pi\,,\, \mathsf{C}\rangle,\ \ \text{with } \mathsf{C}(x_i,y_j) \;=\; \tfrac{1}{p}\|x_i - y_j\|^p \qquad \longrightarrow\ \text{Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \quad \pi\mathbf{1} = \mathsf{A}, \quad \pi^\top\mathbf{1} = \mathsf{B}$$





$\sum_{i,j} \pi_{i,j}\, \mathsf{C}(x_i,y_j)$

$\sum_i \mathsf{A}_i f_i + \sum_j \mathsf{B}_j g_j \ \longrightarrow\ $ FedEx

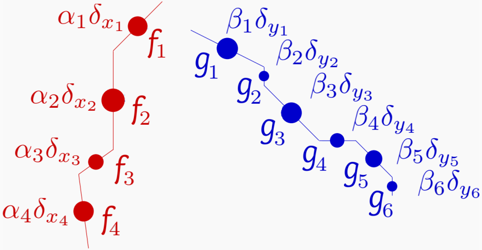
$$\max_{f\,,\,g}\ \langle \mathsf{A}\,,\, f\rangle + \langle \mathsf{B}\,,\, g\rangle$$

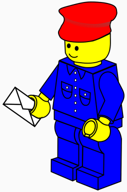
$$\text{s.t.}\quad f(x_i) + g(y_j) \leqslant \mathsf{C}(x_i,y_j),$$

Duality: central planning with NM variables $\simeq$ outsourcing with N + M variables

$$\mathrm{OT}(\mathsf{A},\mathsf{B}) = \min_{\pi} \langle \pi , \mathsf{C} \rangle, \text{ with } \mathsf{C}(x_i, y_j) = \tfrac{1}{p}\|x_i - y_j\|^p \quad \longrightarrow \text{ Assignment}$$

$$\text{s.t. } \pi \geqslant 0, \quad \pi\mathbf{1} = \mathsf{A}, \quad \pi^\top\mathbf{1} = \mathsf{B}$$





$\sum_{i,j} \pi_{i,j}\,\mathsf{C}(x_i, y_j)$

$\sum_i \mathsf{A}_i f_i + \sum_j \mathsf{B}_j g_j$

$$= \max_{f,g} \quad \langle \mathsf{A}, f\rangle + \langle \mathsf{B}, g\rangle \qquad \longrightarrow \text{ FedEx}$$

$$\text{s.t.} \quad f(x_i) + g(y_j) \leqslant \mathsf{C}(x_i, y_j),$$

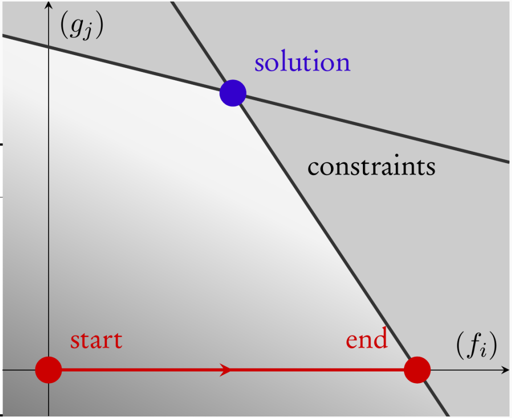
Being too greedy... doesn't work!

$$\mathrm{OT}(\alpha, \beta) = \max_{\substack{(f_i) \in \mathbb{R}^{\mathrm{N}} \\ (g_j) \in \mathbb{R}^{\mathrm{M}}}} \sum_{i=1}^{\mathrm{N}} \alpha_i f_i + \sum_{j=1}^{\mathrm{M}} \beta_j g_j$$

$$\text{s.t. } \forall\, i, j,\ f_i + g_j \leqslant \mathbf{C}(x_i, y_j)$$

Algorithm 3.1: Naive greedy algorithm

1: $f_i,\ g_j \leftarrow \mathbf{0}_{\mathbb{R}^{\mathrm{N}}},\ \mathbf{0}_{\mathbb{R}^{\mathrm{M}}}$
2: **repeat**
3: $\quad f_i \leftarrow \min_{j=1}^{\mathrm{M}} \left[\, \mathbf{C}(x_i, y_j) - g_j \,\right]$
4: $\quad g_j \leftarrow \min_{i=1}^{\mathrm{N}} \left[\, \mathbf{C}(x_i, y_j) - f_i \,\right]$
5: **until** convergence.
6: **return** $f_i,\ g_j$



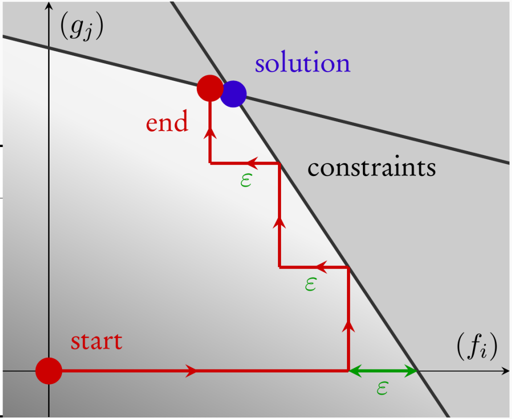
The auction algorithm: take it easy with a slackness $\varepsilon > 0$

$$\mathrm{OT}(\alpha,\beta) = \max_{\substack{(f_i)\in\mathbb{R}^{\mathrm{N}}\\(g_j)\in\mathbb{R}^{\mathrm{M}}}} \sum_{i=1}^{\mathrm{N}} \alpha_i f_i + \sum_{j=1}^{\mathrm{M}} \beta_j g_j$$

$$\text{s.t. } \forall\, i,j,\; f_i + g_j \leqslant \mathbf{C}(x_i,y_j)$$

Algorithm 3.2: Pseudo-auction algorithm

1: $f_i,\ g_j \leftarrow \mathbf{0}_{\mathbb{R}^{\mathrm{N}}},\ \mathbf{0}_{\mathbb{R}^{\mathrm{M}}}$
2: **repeat**
3: $\quad f_i \leftarrow \min_{j=1}^{\mathrm{M}} \left[\, \mathbf{C}(x_i,y_j) - g_j \,\right] - \varepsilon$
4: $\quad g_j \leftarrow \min_{i=1}^{\mathrm{N}} \left[\, \mathbf{C}(x_i,y_j) - f_i \,\right]$
5: **until** $\forall\, i,\ \exists\, j,\ f_i + g_j \geqslant \mathbf{C}(x_i,y_j) - \varepsilon$.
6: **return** $f_i,\ g_j$



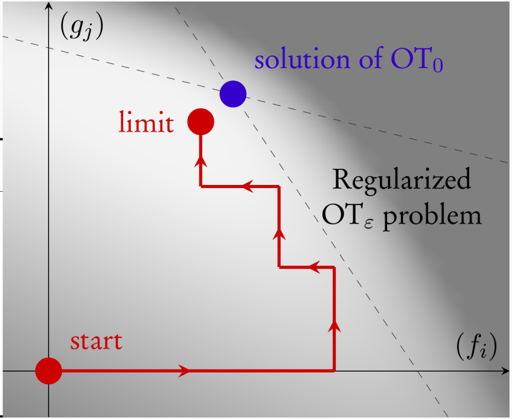
The Sinkhorn algorithm: use a softmin, get a well-defined optimum

$$\mathrm{OT}(\alpha,\beta) = \max_{\substack{(f_i)\in\mathbb{R}^N \\ (g_j)\in\mathbb{R}^M}} \sum_{i=1}^{N} \alpha_i f_i + \sum_{j=1}^{M} \beta_j g_j$$

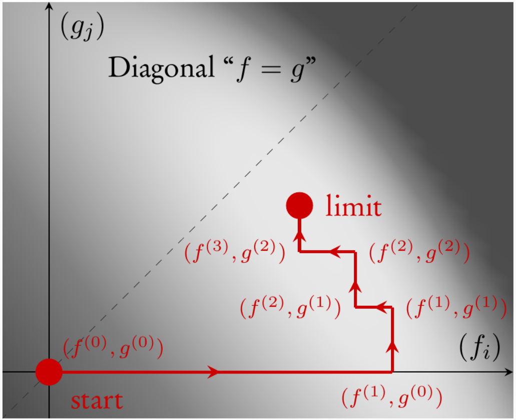
$$\text{s.t. } \forall\, i,j,\ f_i + g_j \leqslant \mathbf{C}(x_i,y_j)$$

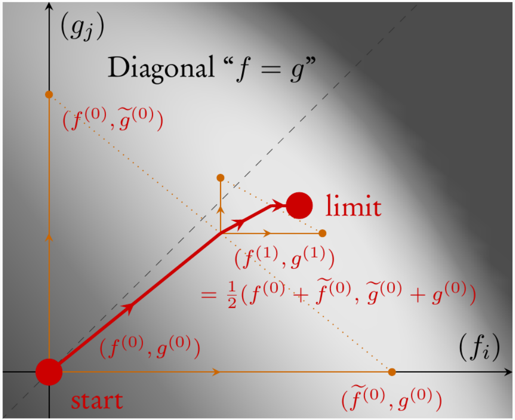
Algorithm 3.3: Sinkhorn or "soft-auction" algorithm

1: $f_i,\ g_j \leftarrow \mathbf{0}_{\mathbb{R}^N},\ \mathbf{0}_{\mathbb{R}^M}$
2: **repeat**
3: $\quad f_i \leftarrow -\varepsilon \log \sum_{j=1}^{M} \beta_j \exp \frac{1}{\varepsilon}[\, g_j - \mathbf{C}(x_i,y_j) \,]$
4: $\quad g_j \leftarrow -\varepsilon \log \sum_{i=1}^{N} \alpha_i \exp \frac{1}{\varepsilon}[\, f_i - \mathbf{C}(x_i,y_j) \,]$
5: **until** convergence up to a set tolerance.
6: **return** $f_i,\ g_j$



The symmetric Sinkhorn algorithm: stay close to the diagonal if A $\simeq$ B





Remark 1: a streamlined algorithm

One key operation – the soft, **weighted distance transform**:

$$\forall i \in [1, \mathsf{N}], \quad f(x_i) \leftarrow \min_{y \sim \beta} \left[\mathsf{C}(x_i, y) - g(y) \right] = -\varepsilon \log \sum_{j=1}^{\mathsf{M}} \beta_j \exp \tfrac{1}{\varepsilon} \left[g_j - \mathsf{C}(x_i, y_j) \right].$$

Similar to the chamfer distance transform, convolution with a Gaussian kernel…

Fast implementations with **pyKeOps**:

- If $\mathsf{C}(x_i, y_j)$ is a closed formula: **bruteforce** scales to N, M $\simeq$ 100k in 10ms on a GPU.
- If A and B have a low-dimensional support:
  use a clustering and **truncation** strategy to get a x10 speed-up.
- If A and B are supported on a 2D or 3D grid and $\mathsf{C}(x_i, y_j) = \frac{1}{2}\|x_i - y_j\|^2$:
  use a **separable** distance transform to get a second x10 speed-up.
  (N.B.: FFTs run into numerical accuracy issues.)

Remark 2: annealing works!

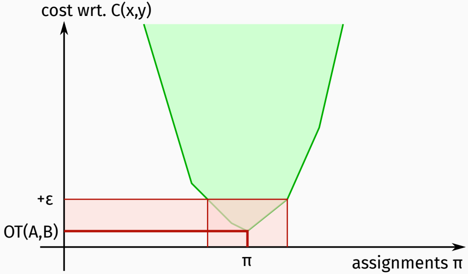
The **Auction/Sinkhorn** algorithms:

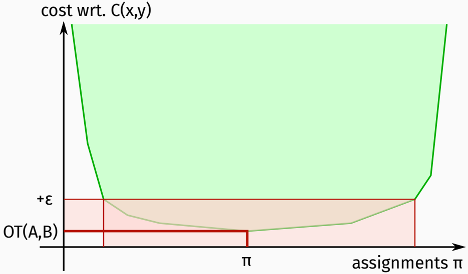
- Improve the dual cost by at least ε at each (early) step.
- Reach an ε-optimal solution with $(\max \mathrm{C})\ /\ \varepsilon$ steps.

Simple heuristic: run the optimization with **decreasing values** of ε.

ε-scaling
$=$ **simulated annealing**
$=$ **multiscale** strategy
$=$ **divide and conquer**

Remark 3: the curse of dimensionality





In **low dimension**:

- $\|x - y\|$ takes large and small values.
- The OT objective is **peaky** wrt. π.
- ε-optimal solutions are **useful**.
- OT(discrete samples) $\simeq$ OT(underlying distributions)

In **high dimension**:

- $\|x - y\|$ gets closer to a constant.
- The OT objective is **flat** wrt. π.
- ε-optimal solutions are **random**.
- OT(discrete samples) $\neq$ OT(underlying distributions)

To recap 80+ years of work...

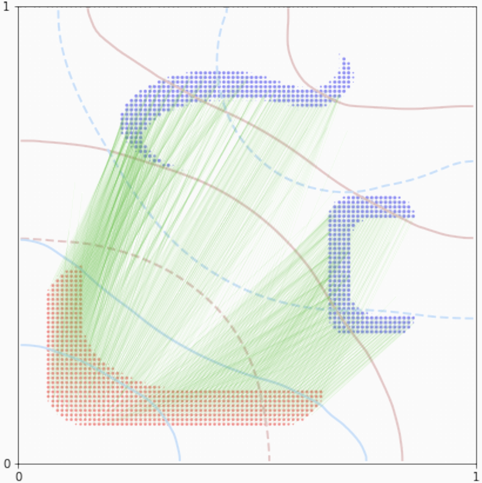
Key dates for discrete optimal transport with N points:

- [Kan42]: **Dual** problem of Kantorovitch.
- [Kuh55]: **Hungarian** methods in $O(\text{N}^3)$.
- [Ber79]: **Auction** algorithm in $O(\text{N}^2)$.
- [KY94]: **SoftAssign** = Sinkhorn + simulated annealing, in $O(\text{N}^2)$.
- [GRL$^+$98, CR00]: **Robust Point Matching** = Sinkhorn as a loss.
- [Cut13]: Start of the **GPU era.**
- [Mér11, Lév15, Sch19]: **multi-scale** solvers in $O(\text{N} \log \text{N})$.

- **Solution,** today: **Multiscale Sinkhorn algorithm, on the GPU**.

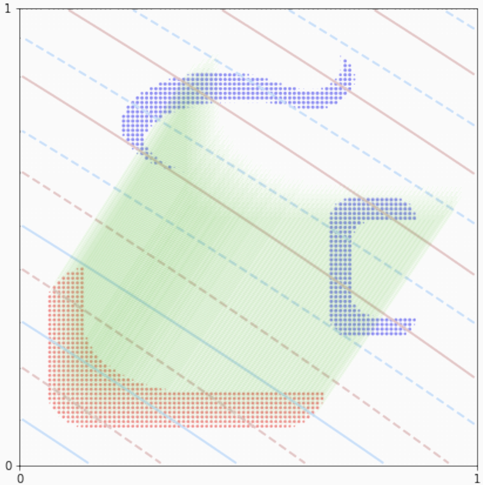
$\Longrightarrow$ Generalized **QuickSort** algorithm.

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathbf{OT}(\alpha, \beta)$



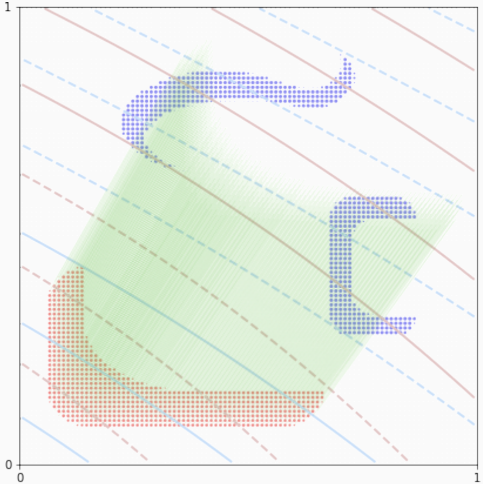
OT plan in 2D.

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha,\beta)$



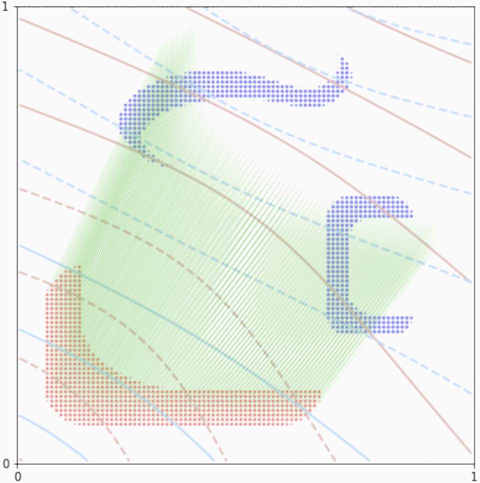
Iteration 0, blur $\sigma = \sqrt{\varepsilon} = 2^0$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha,\beta)$



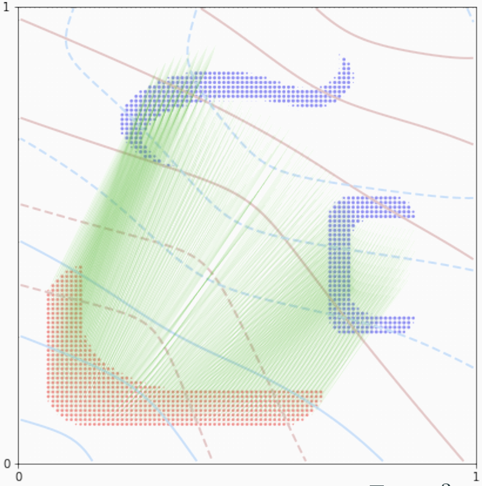
Iteration 1, blur $\sigma = \sqrt{\varepsilon} = 2^{-1}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



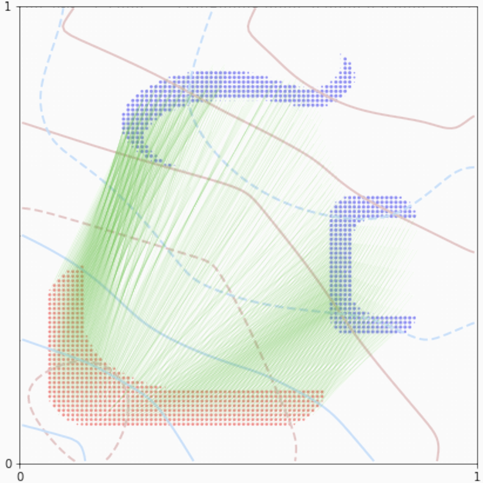
Iteration 2, blur $\sigma = \sqrt{\varepsilon} = 2^{-2}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



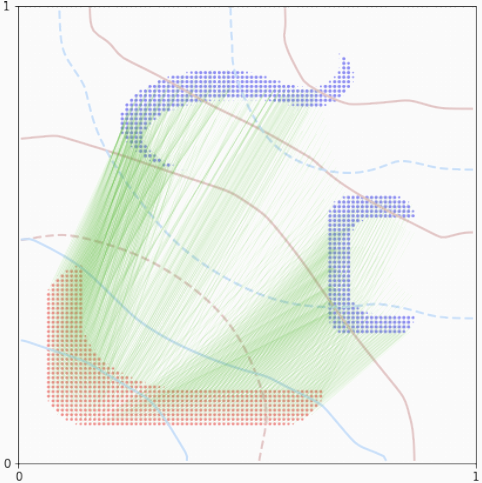
Iteration 3, blur $\sigma = \sqrt{\varepsilon} = 2^{-3}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}$ OT(α, β)



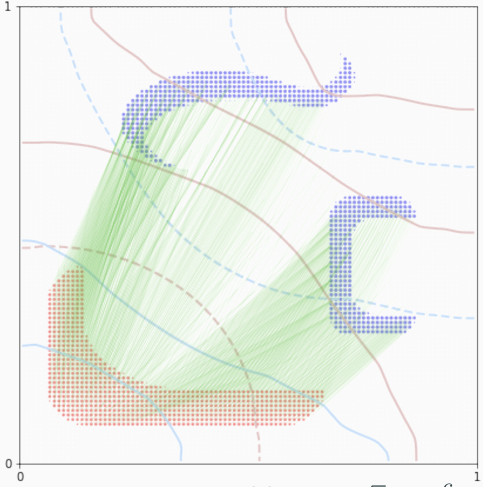
Iteration 4, blur $\sigma = \sqrt{\varepsilon} = 2^{-4}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



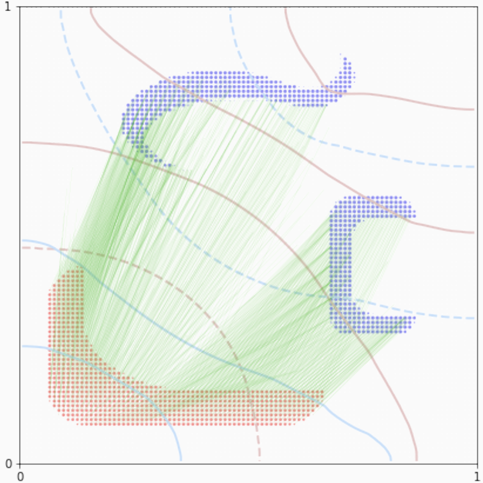
Iteration 5, blur $\sigma = \sqrt{\varepsilon} = 2^{-5}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\textbf{OT}(\alpha, \beta)$



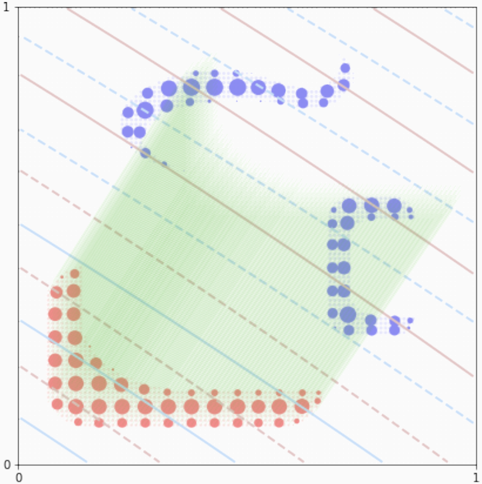
Iteration 6, blur $\sigma = \sqrt{\varepsilon} = 2^{-6}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathbf{OT}(\alpha, \beta)$



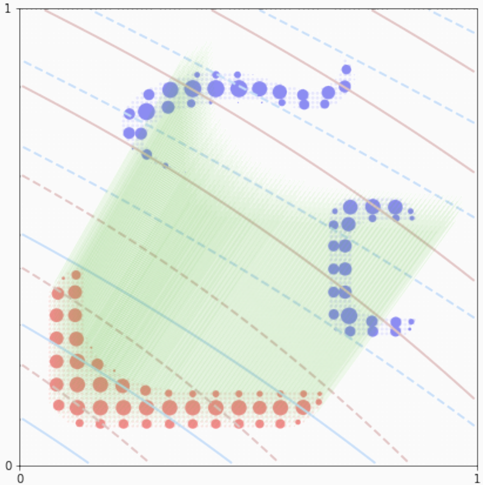
Iteration 7, blur $\sigma = \sqrt{\varepsilon} = \texttt{.01}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}$ OT(α, β)



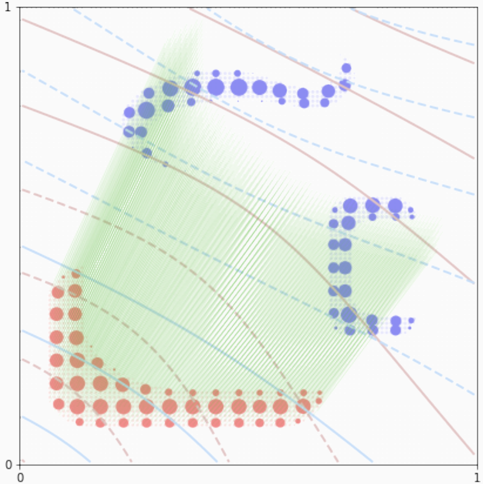
Iteration 0, blur $\sigma = 2^0$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



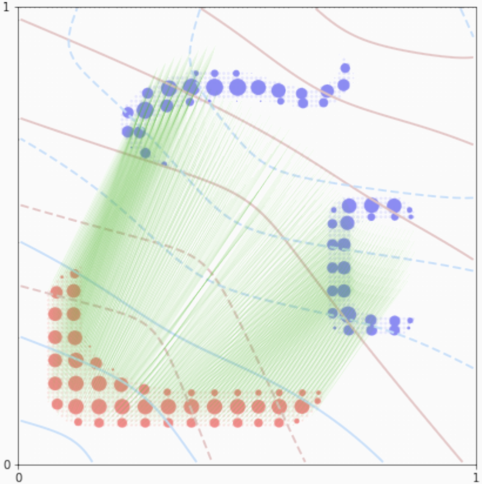
Iteration 1, blur $\sigma = 2^{-1}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha,\beta)$



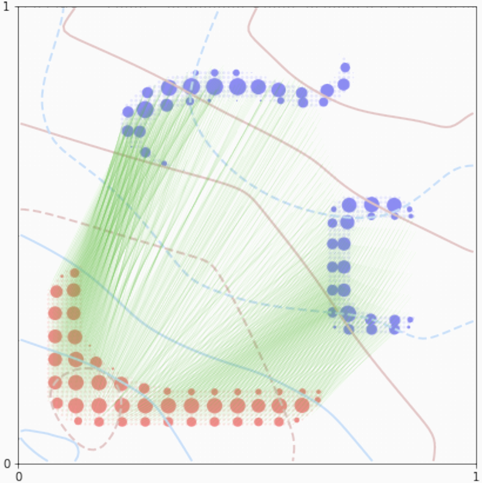
Iteration 2, blur $\sigma = 2^{-2}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



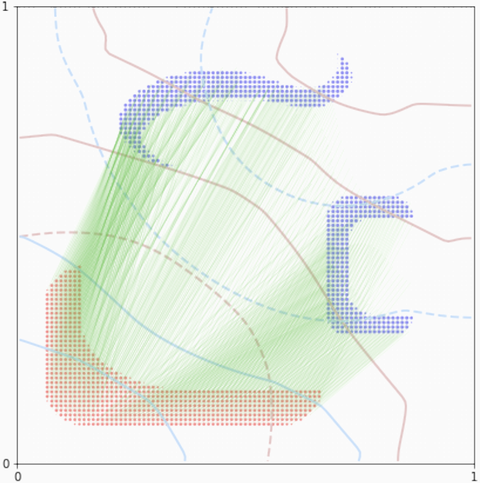
Iteration 3, blur $\sigma = 2^{-3}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



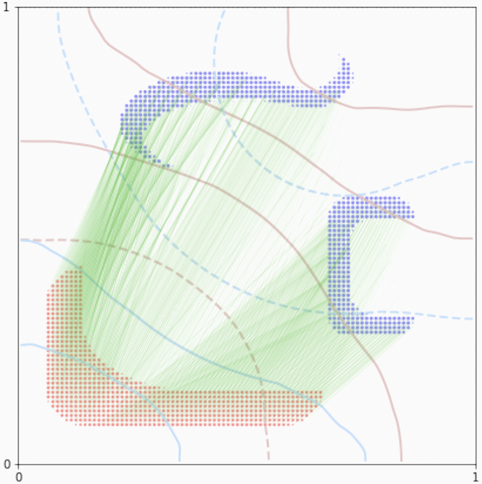
Iteration 4, blur $\sigma = 2^{-4}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha, \beta)$



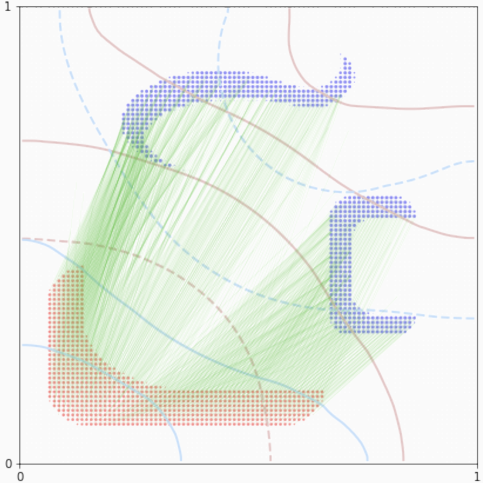
Iteration 5, blur $\sigma = 2^{-5}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha,\beta)$



Iteration 6, blur $\sigma = 2^{-6}$

Visualizing F, G and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{OT}(\alpha,\beta)$



Iteration 7, blur σ = `.01`

Scaling up optimal transport to anatomical data

Progresses of the last decade add up to a $\times$**100** - $\times$**1000** acceleration:

Sinkhorn GPU $\xrightarrow{\times 10}$ + KeOps $\xrightarrow{\times 10}$ + Annealing $\xrightarrow{\times 10}$ + Multi-scale

With a precision of 1%, on a modern gaming GPU:

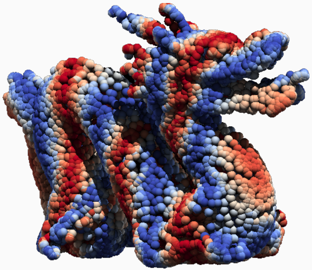
`pip install geomloss`

+

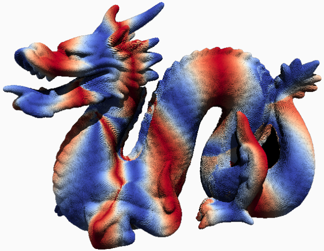
modern GPU

(1 000 €)

$\Longrightarrow$



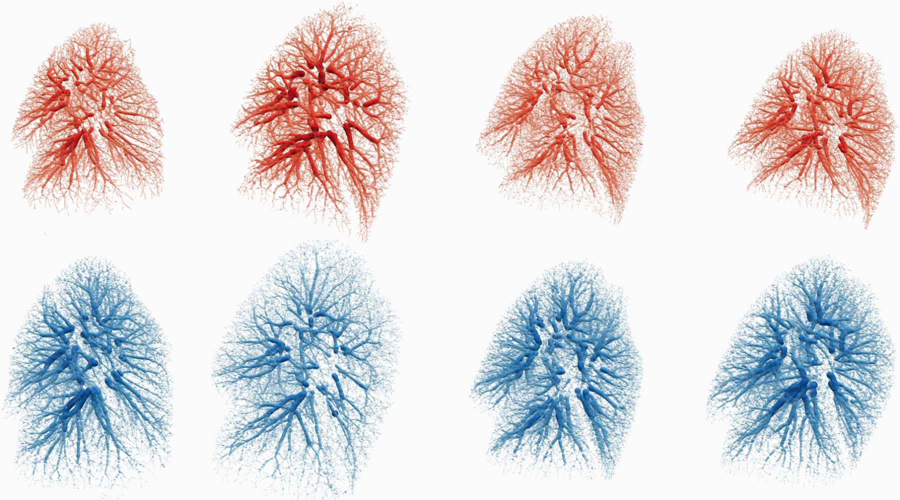
10k points in 30-50ms



100k points in 100-200ms

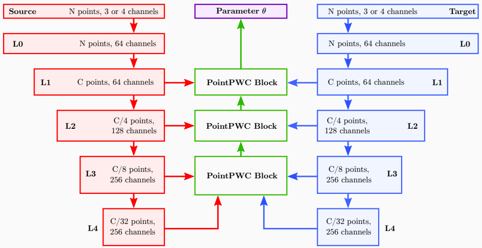
Applications

A typical example in anatomy: lung registration “Exhale – Inhale”



Complex deformations, high **resolution** (50k–300k points), high **accuracy** ($<$ 1mm).

State-of-the-art networks – and their limitations



Multi-scale convolutional point neural network.

Point neural nets, **in practice**:

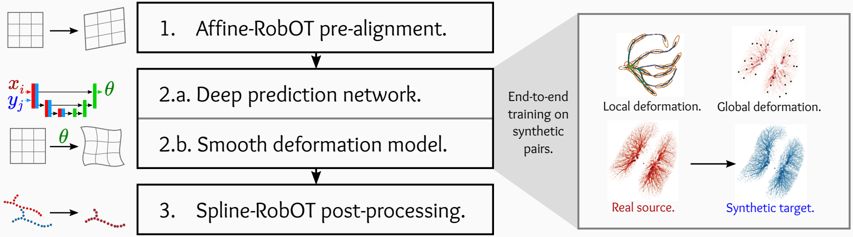
- Compute **descriptors** at all scales.
- **Match** them using geometric layers.
- Train on **synthetic** deformations.

Strengths and weaknesses:

- Good at **pairing** branches.
- Hard to train to high **accuracy**.

$\Longrightarrow$ **Complementary** to OT.

Three-steps registration

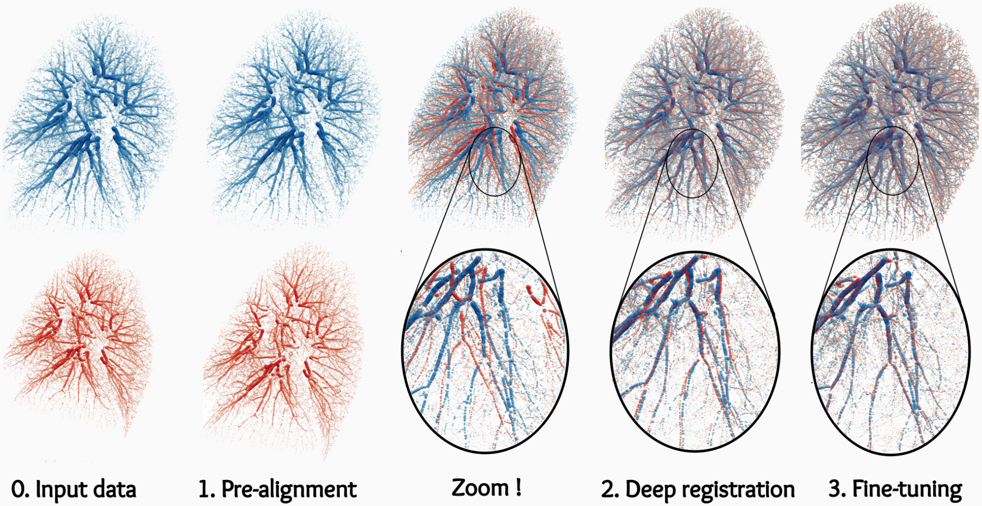


This **pragmatic** method:

- Is **easy to train** on synthetic data.
- Scales up to high-resolution: 100k points in 1s.
- Excellent results: **KITTI** (outdoors scans) and **DirLab** (lungs).

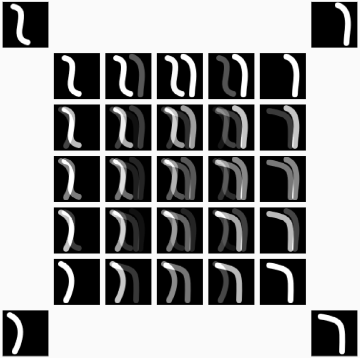
Accurate *point cloud registration with* ***robust*** *optimal transport*,
Shen, Feydy et al., NeurIPS 2021.

Three-steps registration



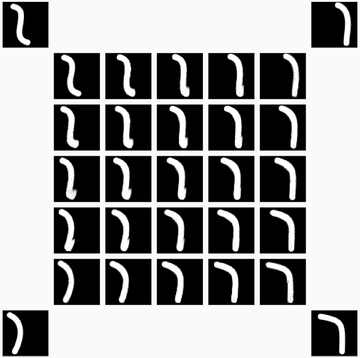
Wasserstein barycenters [AC11]

$$\text{Barycenter } {\color{red}\mathsf{A}}^* \;=\; \arg\min_{\color{red}\mathsf{A}} \sum_{i=1}^{4} \lambda_i \,\text{Loss}(\,{\color{red}\mathsf{A}}\,,\, {\color{blue}\mathsf{B}_i}\,)\;.$$



Euclidean barycenters.

$\text{Loss}({\color{red}\mathsf{A}}, {\color{blue}\mathsf{B}}) = \|{\color{red}\mathsf{A}} - {\color{blue}\mathsf{B}}\|^2_{L^2}$



Wasserstein barycenters.

$\text{Loss}({\color{red}\mathsf{A}}, {\color{blue}\mathsf{B}}) = \text{OT}({\color{red}\mathsf{A}}, {\color{blue}\mathsf{B}})$

Wasserstein barycenters

From a computational perspective:

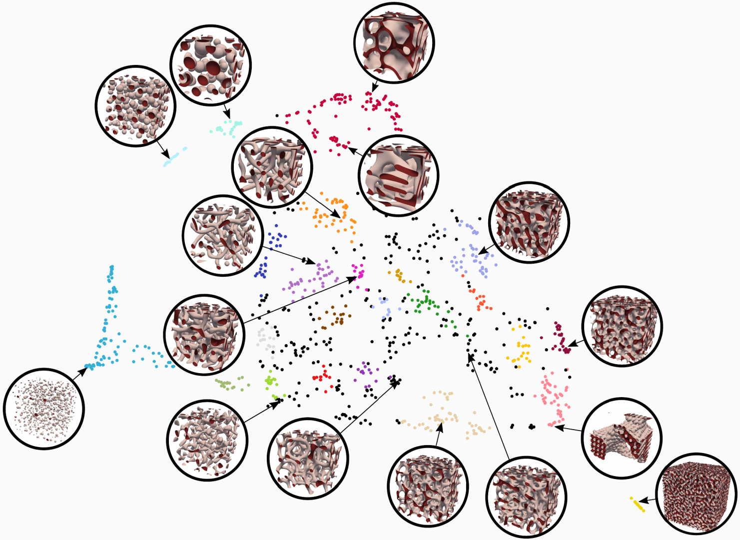
- The problem is **convex** (easy) wrt. the weights.
- The support of the barycenter lies in the **convex hull** of the input distributions.

The **curse of dimensionality** hits hard:

- In high dimension, identifying the support can become **NP-hard**.
- In dimensions 2 and 3, we can just use a grid and recover **super fast** algorithms. Computing OT distances and barycenters between **density maps** is a solved problem.

$\Longrightarrow$ We can now **easily** do manifold learning with e.g. UMAP in Wasserstein spaces of **2D and 3D** distributions.

An example this afternoon: Anna Song's presentation on 3D shape textures



Conclusion

Genuine team work

Benjamin Charlier

Joan Glaunès

Thibault Séjourné

F.-X. Vialard

Gabriel Peyré

Alain Trouvé

Marc Niethammer

Shen Zhengyang

Olga Mula

Hieu Do

Key points

- Optimal Transport = **generalized sorting** :

  $\longrightarrow$ Super-fast solvers on **simple domains** (esp. 2D/3D spaces).

  $\longrightarrow$ Simple registration for shapes that are close to each other.

  $\longrightarrow$ **Fundamental tool** at the intersection of geometry and statistics.

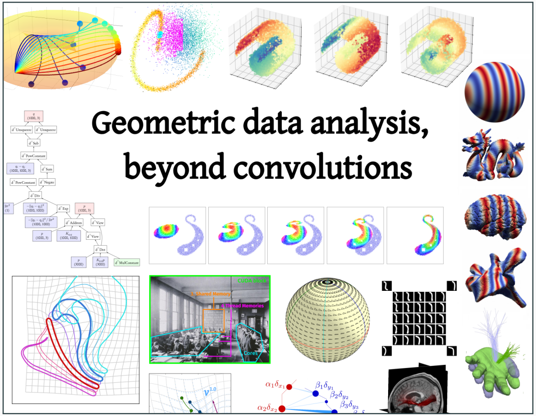
  $\longrightarrow$ Can we extend recent computational advances to **topology-aware** metrics?

- GPUs are more **versatile** than you think.

  $\longrightarrow$ Ongoing work to provide **fast GPU backends** to researchers, going beyond what Google and Facebook are ready to pay for.

Documentation and tutorials are available online

$\Longrightarrow$ www.kernel-operations.io $\Longleftarrow$



www.jeanfeydy.com/geometric_data_analysis.pdf

References

References i

M. Agueh and G. Carlier.

Barycenters in the Wasserstein space.

SIAM Journal on Mathematical Analysis, 43(2):904–924, 2011.

Dimitri P Bertsekas.

A distributed algorithm for the assignment problem.

Lab. for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA, 1979.

References ii

Haili Chui and Anand Rangarajan.

A new algorithm for non-rigid point matching.

In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 44–51. IEEE, 2000.

Marco Cuturi.

Sinkhorn distances: Lightspeed computation of optimal transport.

In *Advances in Neural Information Processing Systems*, pages 2292–2300, 2013.

Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness.

New algorithms for 2d and 3d point matching: Pose estimation and correspondence.

Pattern recognition, 31(8):1019–1031, 1998.

Leonid V Kantorovich.

On the translocation of masses.

In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.

Harold W Kuhn.
The Hungarian method for the assignment problem.
Naval research logistics quarterly, 2(1-2):83–97, 1955.

Jeffrey J Kosowsky and Alan L Yuille.
The invisible hand algorithm: Solving the assignment problem with statistical physics.
Neural networks, 7(3):477–490, 1994.

Bruno Lévy.

A numerical algorithm for l2 semi-discrete optimal transport in 3d.

ESAIM: Mathematical Modelling and Numerical Analysis, 49(6):1693–1715, 2015.

Quentin Mérigot.

A multiscale approach to optimal transport.

In *Computer Graphics Forum*, volume 30, pages 1583–1592. Wiley Online Library, 2011.

Gabriel Peyré and Marco Cuturi.

Computational optimal transport.

arXiv preprint arXiv:1803.00567, 2018.

Bernhard Schmitzer.

Stabilized sparse scaling algorithms for entropy regularized transport problems.

SIAM Journal on Scientific Computing, 41(3):A1443–A1481, 2019.