Optimal transport with 3D shapes

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

6th of December, 2023
G-Stats seminar
Epione Inria team, Inria Sophia Antipolis
Background in **mathematics** and **data sciences**:

- **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:
   \[\implies\] **pyKeOps** library for distance and kernel matrices, 500k+ downloads.

2. Scale up **pharmacovigilance** to the full French population:
   \[\implies\] **survivalGPU**, a fast re-implementation of the R survival package.

3. Ease access to modern statistical **shape analysis**:
   \[\implies\] **GeomLoss**, truly scalable optimal transport in Python.
   \[\implies\] **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 $A = (x_1, \ldots, x_N)$ and $B = (y_1, \ldots, y_N)$ are two clouds of $N$ points in $\mathbb{R}^D$, we define:

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

Generalizes sorting to metric spaces. **Linear problem** on the permutation matrix $P$:

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

s.t. $P_{i,j} \geq 0$, $\sum_j P_{i,j} = 1$ \quad \sum_i P_{i,j} = 1$.

Each source point is transported onto the target.

Assignment $\sigma : [1, 5] \rightarrow [1, 5]$.
Practical use

Alternatively, we understand OT as:

- Nearest neighbor projection + incompressibility constraint.
- Fundamental example of linear optimization over the transport plan $P_{i,j}$.

This theory induces two main quantities:

- The transport plan $P_{i,j} \simeq$ the optimal mapping $x_i \mapsto y_{\sigma(i)}$.
- The “Wasserstein” distance $\sqrt{\text{OT}(A, B)}$. 
The optimal transport plan

Before

After
The optimal transport plan

Before

After
The optimal transport plan
The optimal transport plan

Before

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

**Gauss** map \( \mathcal{N} : (m, \sigma) \in \mathbb{R} \times [0, \infty) \mapsto \mathcal{N}(m, \sigma) \in \mathbb{P}(\mathbb{R}) \).

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** \((m, \sigma)\)?

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

OT on \( \mathcal{N}(m, \sigma) \)
\[ \rightarrow \] Flat **Euclidean** metric on \((m, \sigma)\).
How should we solve the OT problem?
Duality: central planning with NM variables $\simeq$ outsourcing with $N + M$ variables

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

$\rightarrow$ Assignment

s.t. $\pi \succeq 0$, $\pi 1 = A$, $\pi^T 1 = B$

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

Diagram of $\pi_{i,j}$ values with $\alpha_i \delta x_i$ and $\beta_i \delta y_i$.
Duality: central planning with NM variables \(\simeq\) outsourcing with \(N + M\) variables

\[
\text{OT}(A, B) = \min_{\pi} \langle \pi, C \rangle, \quad \text{with } C(x_i, y_j) = \frac{1}{p} \|x_i - y_j\|^p
\]

\[
\text{s.t. } \pi \geq 0, \quad \pi 1 = A, \quad \pi^T 1 = B
\]

\[
\sum_{i,j} \pi_{i,j} C(x_i, y_j)
\]

\[
\max_{f, g} \langle A, f \rangle + \langle B, g \rangle
\]

\[
\text{s.t. } f(x_i) + g(y_j) \leq C(x_i, y_j),
\]

\[
\sum_i A_i f_i + \sum_j B_j g_j
\]
Duality: central planning with NM variables $\simeq$ outsourcing with N + M variables

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

$\rightarrow$ Assignment

$$\text{s.t. } \pi \geq 0, \quad \pi \mathbf{1} = A, \quad \pi^T \mathbf{1} = B$$

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

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

$s.t.$ $f(x_i) + g(y_j) \leq C(x_i, y_j)$,

$\sum_i A_i f_i + \sum_j B_j g_j$

$\rightarrow$ FedEx
Being too greedy... doesn’t work!

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

s.t. \( \forall i, j, f_i + g_j \leq C(x_i, y_j) \)

**Algorithm 3.1:** Naive greedy algorithm

1. \( f_i, g_j \leftarrow 0_{\mathbb{R}^N}, 0_{\mathbb{R}^M} \)
2. repeat
3. \( f_i \leftarrow \min_{j=1}^{M} \left[ C(x_i, y_j) - g_j \right] \)
4. \( g_j \leftarrow \min_{i=1}^{N} \left[ C(x_i, y_j) - f_i \right] \)
5. until convergence.
6. return \( f_i, g_j \)
The auction algorithm: take it easy with a slackness $\varepsilon > 0$

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

s.t. $\forall i, j, f_i + g_j \leq C(x_i, y_j)$

**Algorithm 3.2:** Pseudo-auction algorithm

1. $f_i, g_j \leftarrow 0_{\mathbb{R}^N_{+}}, 0_{\mathbb{R}^M}$
2. repeat
3. $f_i \leftarrow \min_{j=1}^{M} \left[ C(x_i, y_j) - g_j \right] - \varepsilon$
4. $g_j \leftarrow \min_{i=1}^{N} \left[ C(x_i, y_j) - f_i \right]$
5. until $\forall i, \exists j, f_i + g_j \geq C(x_i, y_j) - \varepsilon$.
6. return $f_i, g_j$
The Sinkhorn algorithm: use a softmin, get a well-defined optimum

\[
\text{OT}(\alpha, \beta) = \max_{(f_i) \in \mathbb{R}^N, (g_j) \in \mathbb{R}^M} \sum_{i=1}^{N} \alpha_i f_i + \sum_{j=1}^{M} \beta_j g_j \\
\text{s.t. } \forall i, j, f_i + g_j \leq C(x_i, y_j)
\]

Algorithm 3.3: Sinkhorn or “soft-auction” algorithm

1. \(f_i, g_j \leftarrow 0_{\mathbb{R}^N}, 0_{\mathbb{R}^M}\)
2. repeat
3. \(f_i \leftarrow -\varepsilon \log \sum_{j=1}^{M} \beta_j \exp \frac{1}{\varepsilon} [g_j - C(x_i, y_j)]\)
4. \(g_j \leftarrow -\varepsilon \log \sum_{i=1}^{N} \alpha_i \exp \frac{1}{\varepsilon} [f_i - C(x_i, y_j)]\)
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**:

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

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

Fast implementations with **pyKeOps**:

- If \( C(x_i, y_j) \) is a closed formula: **bruteforce** scales to \( N, M \approx 100k \) 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 \( C(x_i, y_j) = \frac{1}{2} \| x_i - y_j \|^2 \):
  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 $\varepsilon$ at each (early) step.
- Reach an $\varepsilon$-optimal solution with $(\max C) / \varepsilon$ steps.

Simple heuristic: run the optimization with **decreasing values** of $\varepsilon$.

$\varepsilon$-scaling

\[ \begin{align*}
\varepsilon & \text{-scaling} \\
= & \text{simulated annealing} \\
= & \text{multiscale strategy} \\
= & \text{divide and conquer}
\end{align*} \]
Remark 3: the curse of dimensionality

In low dimension:
- \( \|x - y\| \) takes large and small values.
- The OT objective is **peaky** wrt. \( \pi \).
- \( \varepsilon \)-optimal solutions are **useful**.
- \( \text{OT}\)(discrete samples) \( \simeq \) \( \text{OT}\)(underlying distributions)

In high dimension:
- \( \|x - y\| \) gets closer to a constant.
- The OT objective is **flat** wrt. \( \pi \).
- \( \varepsilon \)-optimal solutions are **random**.
- \( \text{OT}\)(discrete samples) \( \neq \) \( \text{OT}\)(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 $O(N^3)$.
- [Ber79]: **Auction** algorithm in $O(N^2)$.
- [KY94]: **SoftAssign** = Sinkhorn + simulated annealing, in $O(N^2)$.
- [GRL+ 98, CR00]: **Robust Point Matching** = Sinkhorn as a loss.
- [Cut13]: Start of the **GPU era**.
- [Mér11, Lév15, Sch19]: **multi-scale** solvers in $O(N \log N)$.

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

  $\Rightarrow$ Generalized **QuickSort** algorithm.
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i$ \text{OT}(\alpha, \beta)$
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

Iteration 1, blur $\sigma = \sqrt{\varepsilon} = 2^{-1}$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i$ OT($\alpha$, $\beta$)

Iteration 2, blur $\sigma = \sqrt{\varepsilon} = 2^{-2}$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

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($\alpha, \beta$)

Iteration 4, blur $\sigma = \sqrt{\varepsilon} = 2^{-4}$
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i$ OT($\alpha, \beta$)

Iteration 5, blur $\sigma = \sqrt{\varepsilon} = 2^{-5}$
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

Iteration 6, blur $\sigma = \sqrt{\varepsilon} = 2^{-6}$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT} (\alpha, \beta)$

Iteration 7, blur $\sigma = \sqrt{\varepsilon} = 0.01$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial_x x_i \mathcal{OT} (\alpha, \beta)$
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

Iteration 1, blur $\sigma = 2^{-1}$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{ OT}(\alpha, \beta)$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i^T \text{OT}(\alpha, \beta)$

Iteration 3, blur $\sigma = 2^{-3}$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

Iteration 4, blur $\sigma = 2^{-4}$
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{ OT}(\alpha, \beta)$
Visualizing $F, G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

Iteration 6, blur $\sigma = 2^{-6}$
Visualizing $F$, $G$ and the Brenier map $\nabla F(x_i) = -\frac{1}{\alpha_i} \partial x_i \text{OT}(\alpha, \beta)$

Iteration 7, blur $\sigma = .01$
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 €)
```

$10k$ 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).
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**.

[Multi-scale convolutional point neural network](#)
Three-steps registration

1. Affine-RobOT pre-alignment.
2.a. Deep prediction network.
2.b. Smooth deformation model.

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

0. Input data
1. Pre-alignment
2. Deep registration
3. Fine-tuning
Barycenter $A^* = \arg \min_A \sum_{i=1}^{4} \lambda_i \text{Loss}(A, B_i)$.

**Euclidean** barycenters.
\[
\text{Loss}(A, B) = \|A - B\|^2_{L^2}
\]

**Wasserstein** barycenters.
\[
\text{Loss}(A, B) = \text{OT}(A, B)
\]
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.

⇒ We can now **easily** do manifold learning with e.g. UMAP in Wasserstein spaces of **2D and 3D** distributions.
An example: Anna Song’s exploration of 3D shape textures [Son22]
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**:
  ─→ Super-fast solvers on **simple domains** (esp. 2D/3D spaces).
  ─→ Simple registration for shapes that are close to each other.
  ─→ **Fundamental tool** at the intersection of geometry and statistics.
  ─→ Can we extend recent computational advances to **topology-aware** metrics?

• GPUs are more **versatile** than you think.
  ─→ 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

www.kernel-operations.io

www.jeanfeydy.com/geometric_data_analysis.pdf
M. Agueh and G. Carlier.

**Barycenters in the Wasserstein space.**


Dimitri P Bertsekas.

**A distributed algorithm for the assignment problem.**

Haili Chui and Anand Rangarajan.

A new algorithm for non-rigid point matching.


Marco Cuturi.

Sinkhorn distances: Lightspeed computation of optimal transport.

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

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

Leonid V Kantorovich.  

**On the translocation of masses.**  
Harold W Kuhn.

The Hungarian method for the assignment problem.


Jeffrey J Kosowsky and Alan L Yuille.

The invisible hand algorithm: Solving the assignment problem with statistical physics.

Bruno Lévy.

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


Quentin Mérigot.

**A multiscale approach to optimal transport.**

Gabriel Peyré and Marco Cuturi.

**Computational optimal transport.**


Bernhard Schmitzer.

**Stabilized sparse scaling algorithms for entropy regularized transport problems.**

Anna Song.

*Generation of tubular and membranous shape textures with curvature functionals.*