Geometric data analysis, beyond convolutions

Jean Feydy
Imperial College London

OPIS Inria team, online — January 2021.

Joint work with B. Charlier, J. Glaunès (numerical foundations),
T. Séjourné, F.-X. Vialard, G. Peyré (optimal transport theory),
P. Roussillon, P. Gori, A. Trouvé (applications to computational anatomy),
F. Sverrisson, B. E. Correia, M. Bronstein (applications to protein sciences).

Who am I?

2012–2016 ENS Paris, **mathematics** and applications.

2015 MVA thesis with **Siemens Healthcare** in Princeton.

2016–2019 PhD thesis with Alain Trouvé, **computational anatomy**;
TA/tutor in applied maths at the ENS Paris.

2019–2022 PostDoc with Michael Bronstein, **geometric deep learning**.

Family of medical doctors (radiologist, haematologist, GPs...):
strong motivation to work towards **clinical solutions**.

Make life easier for engineers and researchers in the field:
two libraries (KeOps, GeomLoss) to **speed up geometric methods**,
with new guarantees of **robustness**.

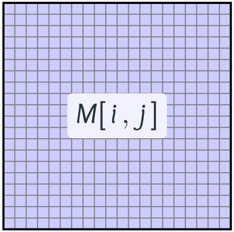
Outline of the presentation

Today, we will talk about:

1. KeOps: fast geometry with **symbolic matrices**.
2. **Applications** to machine learning, proteins, maths...
3. GeomLoss: fast, robust and scalable **optimal transport**.
4. Scientific context, **future works**.

Symbolic matrices?

Machine learning libraries represent most objects as tensors

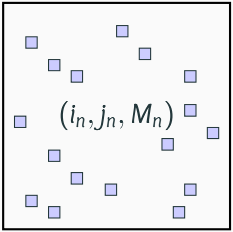


Dense matrix
Coefficients only

Dense matrices – large, contiguous **arrays** of numbers:

+ **Convenient** and well supported.
– Heavy load on the **memories** of our GPUs, with **time-consuming transfers** that take place between compute units.

Machine learning libraries represent most objects as tensors



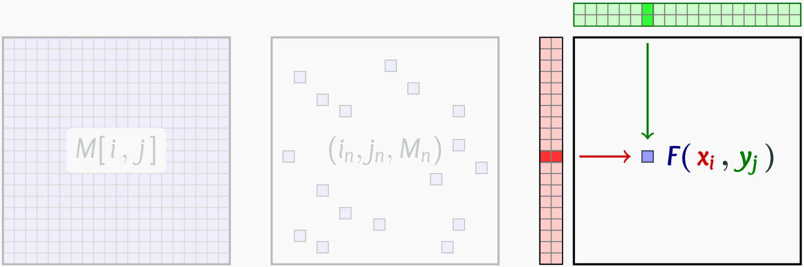
Dense matrix
Coefficients only

Sparse matrix
Coordinates + coeffs

Sparse matrices – tensors that have **few non-zero entries**:

+ Represent **large tensors** with a small memory footprint.
– Outside of **graph** processing, few objects are **sparse enough** to really benefit from this representation.

Machine learning libraries represent most objects as tensors



Dense matrix
Coefficients only

Sparse matrix
Coordinates + coeffs

Symbolic matrix
Formula + data

Distance and **kernel** matrices, **point** convolutions, **attention** layers:

+ **Linear** memory usage: no more **memory** overflows.
+ We can optimize the use of registers for a $\times$**10**-$\times$**100 speed-up** vs. a standard PyTorch GPU baseline.

We provide support for this “new abstraction” on the GPU

Our library comes with all the perks of a deep learning toolbox:

+ Transparent **array-like** interface.
+ Full support for automatic **differentiation**.
+ Comprehensive collection of **tutorials**, available online.

Under the hood: combines an optimized **C++** engine with high-level binders for **PyTorch**, **NumPy**, Matlab and R (thanks to Ghislain Durif). (We welcome **contributors** for JAX, Julia and other frameworks!)

To get started:

$\Longrightarrow$ `pip install pykeops` $\Longleftarrow$

`www.kernel-operations.io`

A first example: efficient nearest neighbor search in dimension 50

Create large point clouds using **standard PyTorch syntax:**

```
import torch
N, M, D = 10**6, 10**6, 50
x = torch.rand(N, 1, D).cuda()  # (1M,  1, 50) array
y = torch.rand(1, M, D).cuda()  # ( 1, 1M, 50) array
```

Turn **dense** arrays into **symbolic** matrices:

```
from pykeops.torch import LazyTensor
x_i, y_j = LazyTensor(x), LazyTensor(y)
```

Create a large **symbolic matrix** of squared distances:

```
D_ij = ((x_i - y_j)**2).sum(dim=2)  # (1M, 1M) symbolic
```

Use an `.argmin()` **reduction** to perform a nearest neighbor query:

```
indices_i = D_ij.argmin(dim=1)  # -> standard torch tensor
```

The KeOps library combines performance with flexibility

Script of the previous slide = efficient nearest neighbor query,
on par with the bruteforce CUDA scheme of the **FAISS** library...
And can be used with **any metric**!

```
D_ij = ((x_i - x_j) ** 2).sum(dim=2)     # Euclidean
M_ij =  (x_i - x_j).abs().sum(dim=2)     # Manhattan
C_ij = 1 - (x_i | x_j)                   # Cosine
H_ij = D_ij / (x_i[...,0] * x_j[...,0])  # Hyperbolic
```

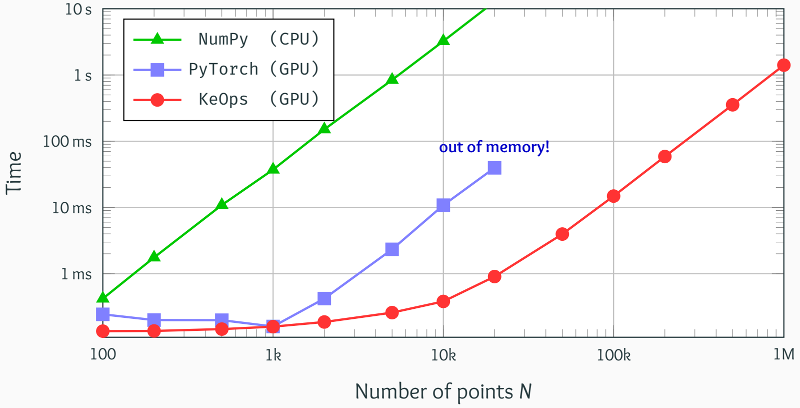
KeOps supports arbitrary **formulas** and **variables** with:

- **Reductions:** sum, log-sum-exp, K-min, matrix-vector product, etc.
- **Operations:** $+$, $\times$, sqrt, exp, neural networks, etc.
- **Advanced schemes:** batch processing, block sparsity, etc.
- **Automatic differentiation:** seamless integration with PyTorch.

KeOps lets users work with millions of points at a time

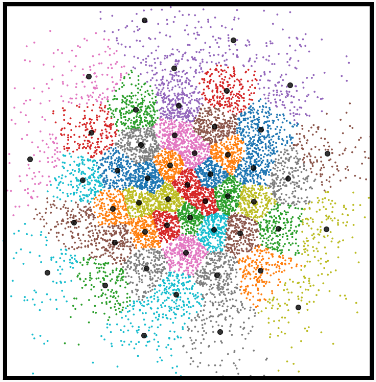
Benchmark of a matrix-vector product with a N-by-N Gaussian kernel matrix between 3D point clouds.

We run NumPy, PyTorch and KeOps on a RTX 2080 Ti GPU.

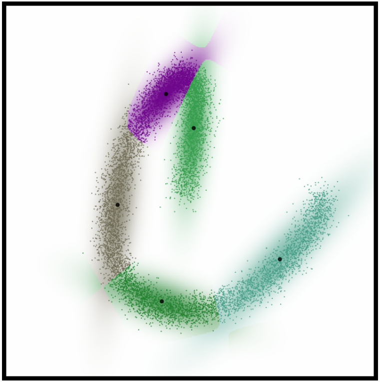


Applications

KeOps is a good fit for machine learning research



K-Means.

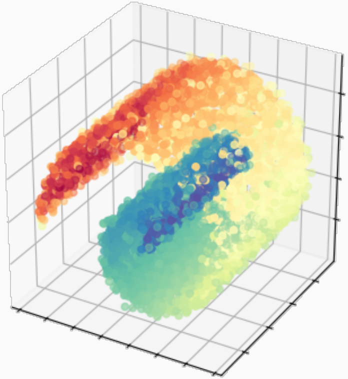


Gaussian Mixture Model.

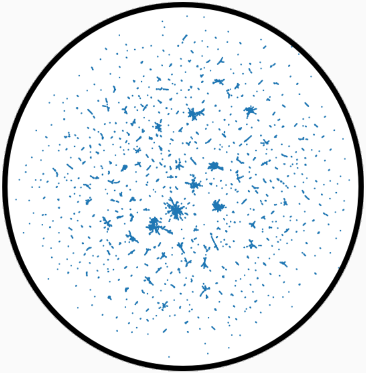
Use any kernel, metric or formula you like!

$\Longrightarrow$ More tutorials coming up soon.

KeOps is a good fit for machine learning research



Spectral analysis.



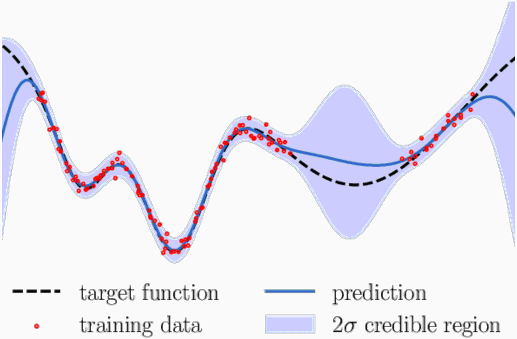
UMAP in hyperbolic space.

Use any kernel, metric or formula you like!

$\Longrightarrow$ More tutorials coming up soon.

Applications to Kriging, spline, Gaussian process, kernel regression

A standard tool for regression [Lec18]:



Under the hood, solve a **kernel linear system**:

$$(\lambda \, \mathrm{Id} + K_{xx}) \, a \;=\; b \qquad \text{i.e.} \qquad a \;\leftarrow\; (\lambda \, \mathrm{Id} + K_{xx})^{-1} b$$

where $\lambda \geqslant 0$ and $(K_{xx})_{i,j} = k(x_i, x_j)$ is a positive definite matrix.

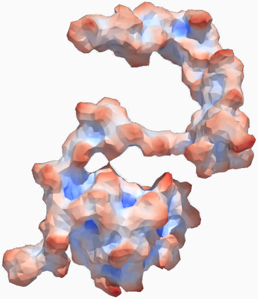
KeOps symbolic tensors:

- Can be fed to **standard solvers**: SciPy, GPytorch, etc.
- GPytorch on the 3DRoad dataset ($N = 278k$, $D = 3$):
  7h with 8 GPUs $\rightarrow$ **15mn** with 1 GPU.
- Provide a **fast backend for research codes**: see e.g. *Kernel methods through the roof: handling* ***billions of points*** *efficiently*, by G. Meanti, L. Carratino, L. Rosasco, A. Rudi (2020).

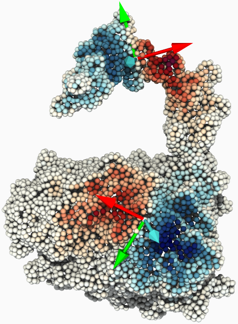
Geometric deep learning w. F. Sverrisson, B. Correia and M. Bronstein

Data-driven methods on **point clouds** and **proteins**:

+ **Fast K-NN search:** local interactions.
+ **Fast N-by-N computations:** global interactions.
+ Heterogeneous **batches**, Octree-like acceleration.

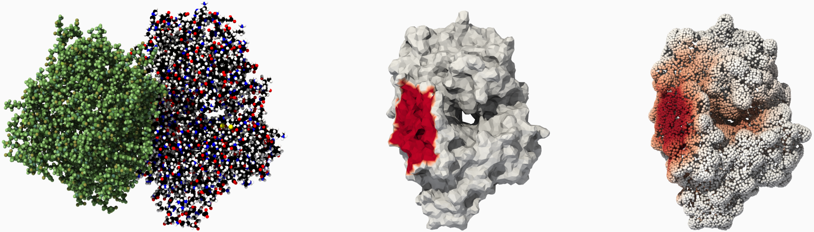


Curvatures at all scales.

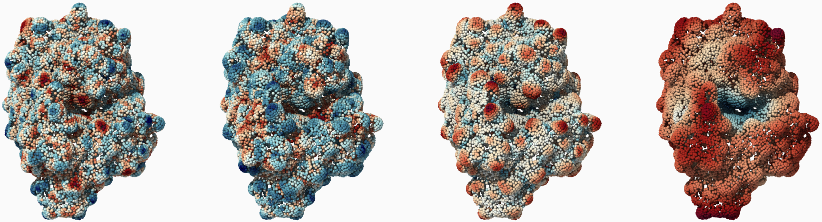


Quasi-geodesic **convolutions**.

Applications to protein sciences [SFCB20]

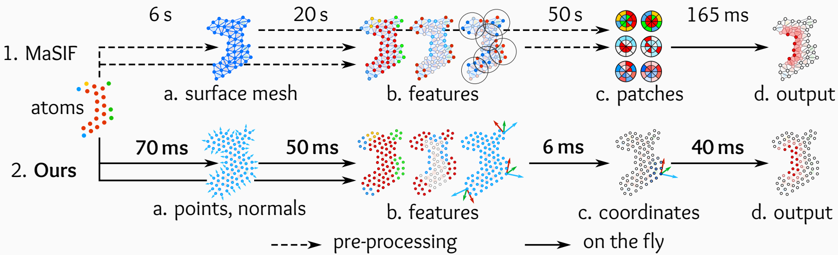


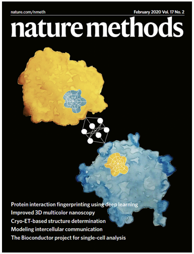
(a) Raw protein data. **(b)** Interface. **(c)** Prediction.



(d) Chem. 1. **(e)** Chem. 2. **(f)** K at 1 Å. **(g)** H at 10 Å.

Fast end-to-end learning on protein surfaces

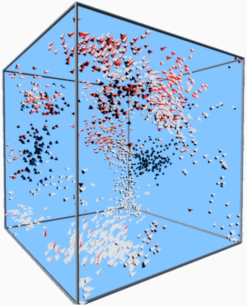




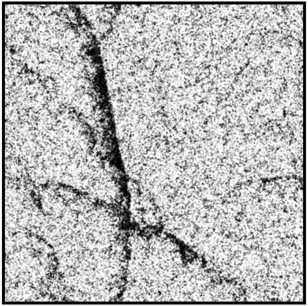
⟶ $\times$100 - $\times$1,000 faster, lighter and fully differentiable.

KeOps lets you focus on your models, results and theorems

Some applications to **dynamical systems** [DM08, DFMAT17] and **statistics** [CDF19] with A. Diez, G. Clarté and P. Degond:



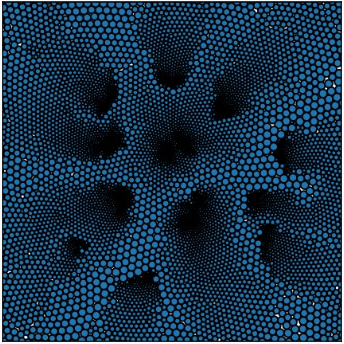
3D Vicsek model with orientation, interactive demo with 2k **flyers**.



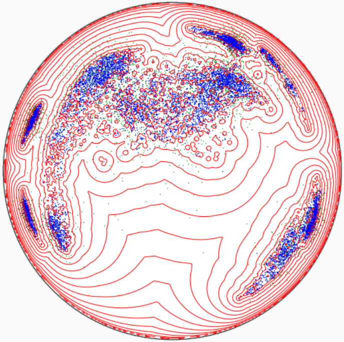
2D Vicsek model on the torus, in real-time with 100k **swimmers**.

KeOps lets you focus on your models, results and theorems

$\Longrightarrow$ Scale up to **millions/billions of agents** with Python scripts.



Packing problem in 2D
with 10k repulsive balls.



Collective Monte Carlo **sampling**
on the hyperbolic Poincaré disk.

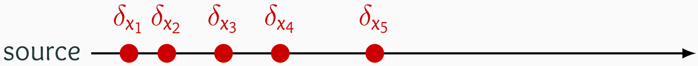
Fast, scalable and robust optimal transport solvers

The Wasserstein, Earth Mover's distance

Sorting points in 1D :

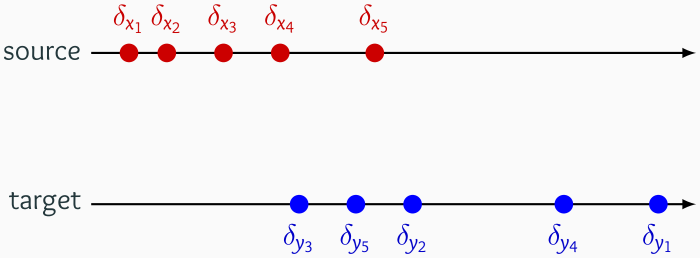
The Wasserstein, Earth Mover's distance

Sorting points in 1D :



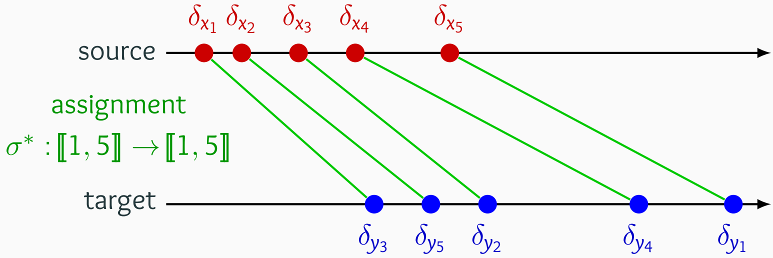
The Wasserstein, Earth Mover's distance

Sorting points in 1D :



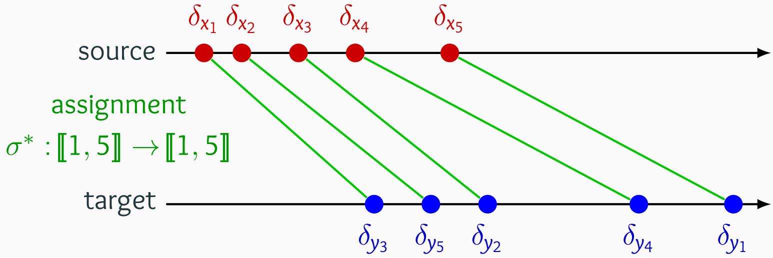
The Wasserstein, Earth Mover's distance

Sorting points in 1D :



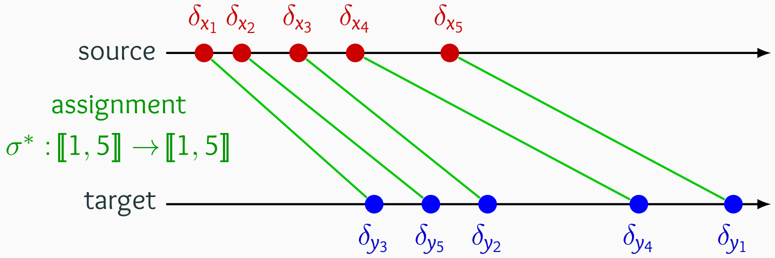
The Wasserstein, Earth Mover's distance

Sorting points in 1D :



The Wasserstein, Earth Mover's distance

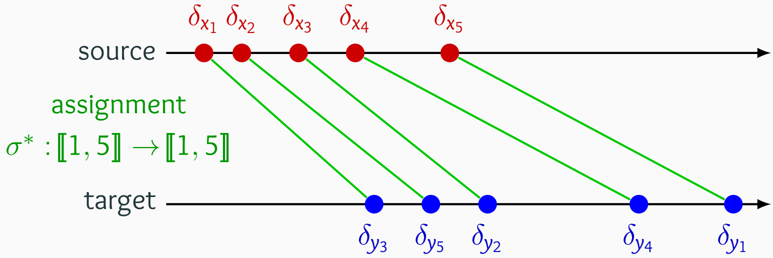
Sorting points in 1D :



$$\text{OT}(\alpha, \beta) \;=\; \frac{1}{2N} \sum_{i=1}^{N} |x_i - y_{\sigma^*(i)}|^2$$

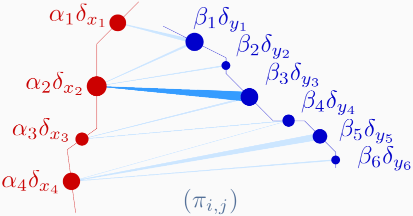
The Wasserstein, Earth Mover's distance

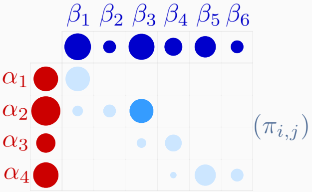
Sorting points in 1D :



$$\text{OT}(\alpha, \beta) = \frac{1}{2N} \sum_{i=1}^{N} |x_i - y_{\sigma^*(i)}|^2 = \min_{\sigma \in \mathcal{S}_N} \frac{1}{2N} \sum_{i=1}^{N} |x_i - y_{\sigma(i)}|^2$$

Optimal transport generalizes sorting to D > 1





Minimize over N-by-M matrices (transport plans) π :

$$\text{OT}(\alpha, \beta) = \min_{\pi} \underbrace{\sum_{i,j} \pi_{i,j} \cdot \tfrac{1}{2}|x_i - y_j|^2}_{\text{transport cost}}$$

subject to $\pi_{i,j} \geqslant 0$,

$$\sum_j \pi_{i,j} = \alpha_i, \quad \sum_i \pi_{i,j} = \beta_j.$$

Key properties [Bre91]

The Wasserstein loss $\text{OT}(\alpha, \beta)$ is:

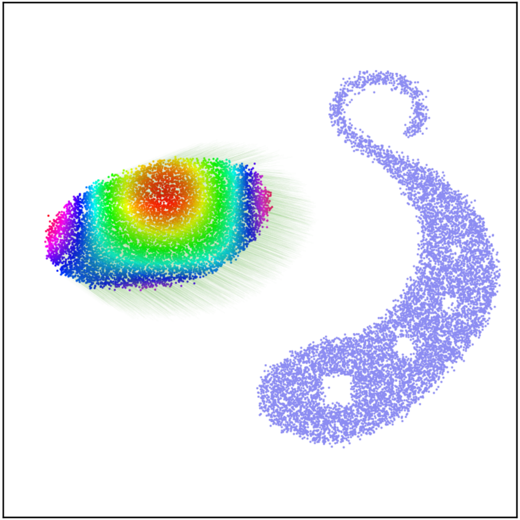
- **Symmetric:** $\text{OT}(\alpha, \beta) = \text{OT}(\beta, \alpha)$.
- **Positive:** $\text{OT}(\alpha, \beta) \geqslant 0$.
- **Definite:** $\text{OT}(\alpha, \beta) = 0 \iff \alpha = \beta$.
- **Translation-aware:** $\text{OT}(\alpha,\ \text{Translate}_{\vec{v}}(\alpha)\,) = \frac{1}{2}\|\,\vec{v}\,\|^2$.
- More generally, OT retrieves the unique **gradient of a convex function** $T = \nabla\varphi$ that maps α onto β :

$$\text{In dimension 1,} \qquad (x_i - x_j) \cdot (y_{\sigma(i)} - y_{\sigma(j)}) \geqslant 0$$

$$\text{In dimension D,} \qquad \langle\, x_i - x_j\ ,\ T(x_i) - T(x_j)\,\rangle_{\mathbb{R}^D} \geqslant 0 .$$

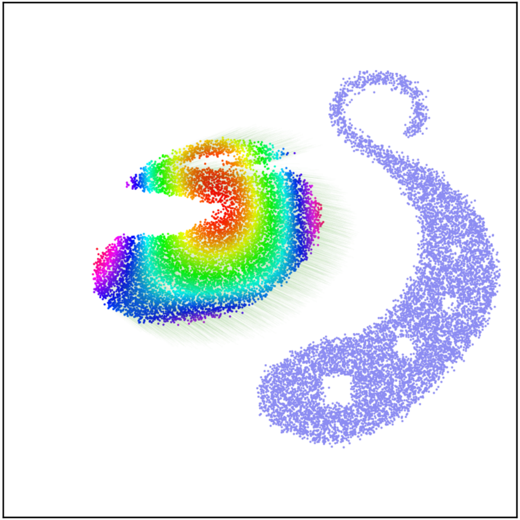
$\Longrightarrow$ Appealing generalization of an **increasing mapping.**

Clean gradients for registration and measure-fitting problems



$t = .00$

Clean gradients for registration and measure-fitting problems

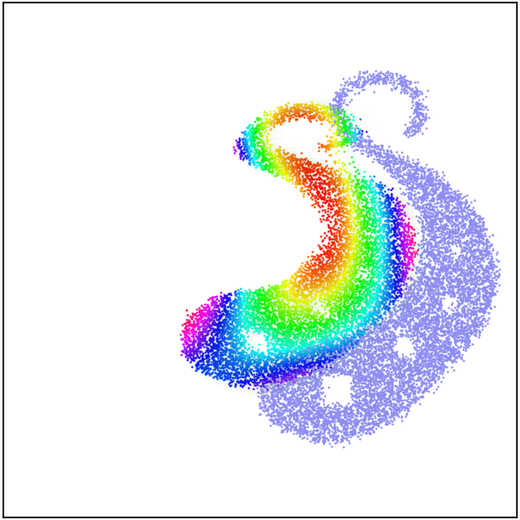


$t = .25$

Clean gradients for registration and measure-fitting problems

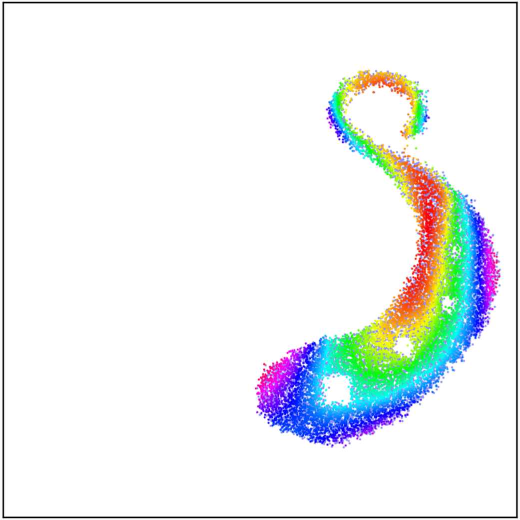
$t = .50$

Clean gradients for registration and measure-fitting problems



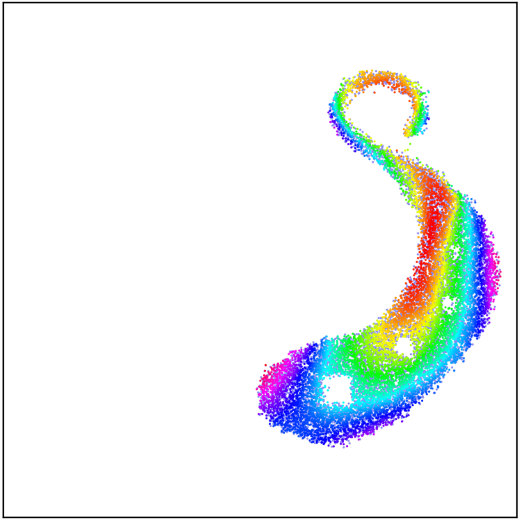
$t = 1.00$

Clean gradients for registration and measure-fitting problems



$t = 5.00$

Clean gradients for registration and measure-fitting problems



$t = 10.00$

Robust optimal transport: softening the bijectivity constraints

Standard OT: minimize over N-by-M transport plans π,

$$\text{OT}(\alpha, \beta) \;=\; \min_{\pi} \,\langle \tfrac{1}{2}|x_i - y_j|^2, \pi \rangle$$

$$\text{s.t.} \quad \pi \geqslant 0\,, \;\; \pi \mathbf{1} \;=\; \alpha, \;\; \pi^\intercal \mathbf{1} \;=\; \beta\,.$$

When dealing with **real-life data**, we'd rather work with:

$$\text{OT}_{\sigma,\rho}(\alpha, \beta) \;=\; \min_{\pi} \,\langle \tfrac{1}{2}|x_i - y_j|^2, \pi \rangle$$

$$+ \underbrace{\sigma^2\, \text{KL}(\pi \,|\, \alpha \otimes \beta)}_{\pi \text{ is fuzzy at scale } \sigma} + \underbrace{\rho^2\, \text{D}(\pi \mathbf{1} \,|\, \alpha) \;+\; \rho^2\, \text{D}(\pi^\intercal \mathbf{1} \,|\, \beta)}_{\pi \text{ tries to match } \alpha \text{ with } \beta \text{... up to a distance } \rho} \quad .$$

In the formula above:

- **KL** is the relative entropy.
- **D** may be the relative entropy, the total variation, etc.

Robust optimal transport: fast algorithms, with guarantees

We define the **Sinkhorn divergence**:

$$\mathrm{S}_{\sigma,\rho}(\alpha,\beta) = \mathrm{OT}_{\sigma,\rho}(\alpha,\beta) - \tfrac{1}{2}\mathrm{OT}_{\sigma,\rho}(\alpha,\alpha) - \tfrac{1}{2}\mathrm{OT}_{\sigma,\rho}(\beta,\beta)$$

$$\simeq \mathrm{OT}_{\text{“lazy-}\rho\text{”}}(\, k_\sigma \star \alpha \,,\, k_\sigma \star \beta \,)\,,$$

where k_σ is a Gaussian kernel of deviation σ and our “lazy” particles do not move beyond a distance ρ.

Theorem 1 (geometry): $S_{\sigma,\rho}$ is suitable for gradient descent. It is **positive**, definite, **convex** and metrizes the convergence in law.

Theorem 2 (algorithm): We can **implement** $S_{\sigma,\rho}$ efficiently, **on GPUs.** Two main ingredients: **log-convolution** with the Gaussian kernel k_σ and a **proximal operator** that is related to $\rho^2\,\mathrm{D}(\cdot\,|\,\cdot)$.

How should we solve the OT problem?

Key dates for discrete optimal transport with N points:

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

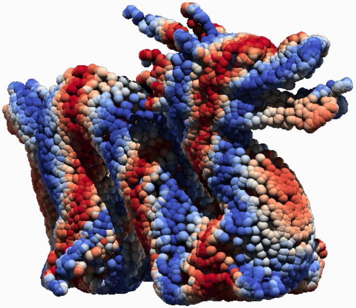
$\Longrightarrow$ Generalized **QuickSort** algorithm.

Scaling up optimal transport to anatomical data

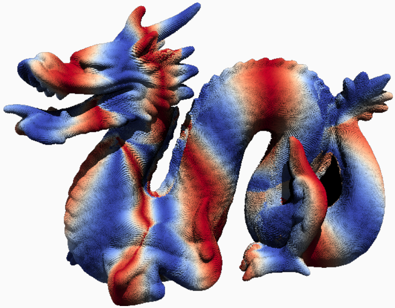
Progresses of the last decade add up to a $\times$**100 - $\times$1000 acceleration:**

Sinkhorn GPU $\xrightarrow{\times 10}$ + KeOps $\xrightarrow{\times 10}$ + Annealing $\xrightarrow{\times 10}$ + Multiscale

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



10k points in 30-50ms



100k points in 100-200ms

Geometric Loss functions for PyTorch

Our website: www.kernel-operations.io/geomloss

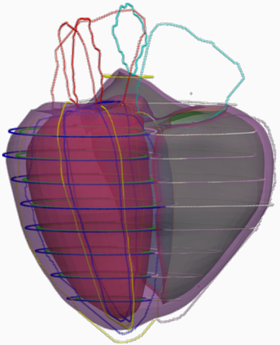
$\Longrightarrow$ pip install geomloss $\Longleftarrow$

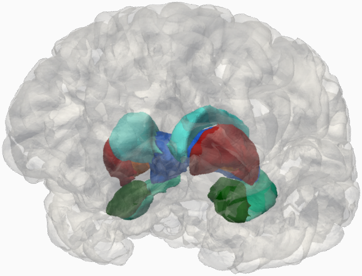
```
# Large point clouds in [0,1]^3
import torch
x = torch.rand(100000, 3, requires_grad=True).cuda()
y = torch.rand(200000, 3).cuda()

# Define a Wasserstein loss between sampled measures
from geomloss import SamplesLoss
loss = SamplesLoss(loss="sinkhorn", p=2)
L = loss(x, y)  # By default, use constant weights
```

Soon: efficient support for **images**, **meshes** and generic metrics.

Ongoing work: computational anatomy





Fast OT-based registration with
S. Joutard, X. Hao, A. Young from KCL,
Z. Shen, M. Niethammer from UNC.

Diffeomorphic and spline registration
e.g. Deformetrica LDDMM software
with the Aramis Inria team.

Scientific context, future works

Genuine team work

Alain Trouvé

Thibault Séjourné

F.-X. Vialard

Gabriel Peyré

Benjamin Charlier

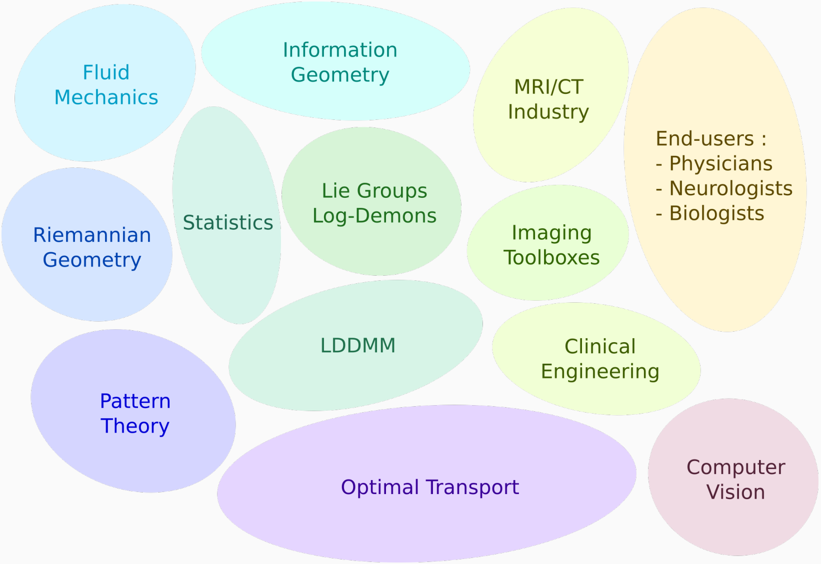
Joan Glaunès

Pierre Roussillon

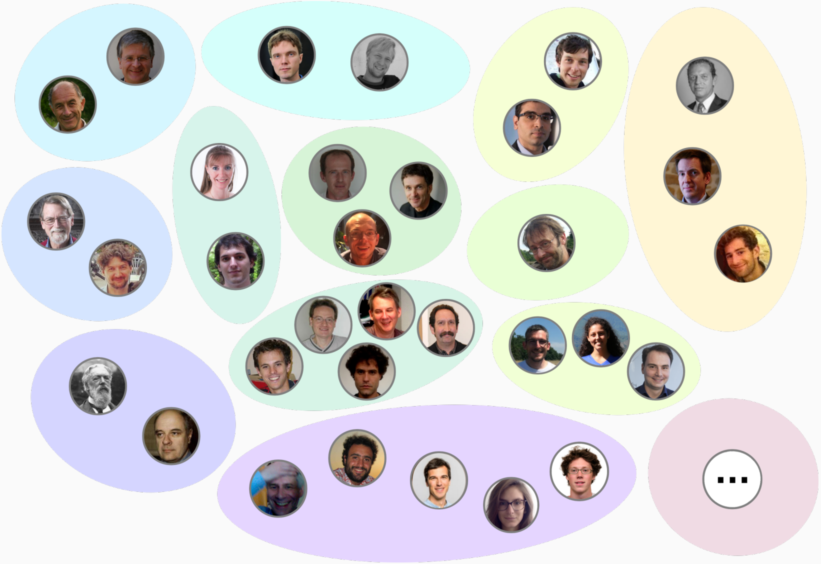
Pietro Gori

+ Freyr Sverrisson, Bruno Correia, Michael Bronstein, ...

Promoting cross-field interactions



Promoting cross-field interactions



The Python revolution

The emergence of an open and **modular** ecosystem of scientific tools has been a **boon** to the community.

Deep learning frameworks have put **GPU computing** and **automatic differentiation** in the hands of every student. (Incredible!)

These libraries have attracted significant backing from **industry** players (Google, Facebook, ...) and allowed the field to **boom** over the last decade.

The Python revolution

Interacting with other researchers, doctors
and engineers has never been so **easy**.

But on the other hand, PyTorch and TensorFlow have also **biased**
the field towards a **small set** of **well-supported** operations:
convolutions and matrix-matrix products, mostly.

This design choice is **not** due to an intrinsic limitation of GPUs:
our hardware is more than capable of **simulating** large,
open **3D worlds** in real-time!

As academic researchers, we must strive to keep **other paths open**.
Foster the development of a full range of methods,
from **robust** convex baselines
to **expensive** deep learning pipelines.

Our contribution to the community

KeOps and GeomLoss are:

- + **Fast:** $\times 10$ - $\times 1{,}000$ speedup vs. naive GPU implementations.
- + **Memory-efficient:** $O(N)$, not $O(N^2)$.
- + **Versatile, with a transparent interface**: freedom!
- + **Powerful and well-documented:** research-friendly.
- − Slow with **large vectors** of dimension $D > 100$.

Our contribution to the community

KeOps and GeomLoss are:

+ **Fast:** $\times 10$ - $\times 1{,}000$ speedup vs. naive GPU implementations.
+ **Memory-efficient:** $O(N)$, not $O(N^2)$.
+ **Versatile, with a transparent interface**: freedom!
+ **Powerful and well-documented:** research-friendly.
− Slow with **large vectors** of dimension $D > 100$.

First half of 2021:

→ **Approximation strategies** (Nyström, etc.) in KeOps.
→ Wasserstein **barycenters** and **grid images** in GeomLoss.

An ongoing research project

Roadmap for KeOps + GeomLoss:

2017–18 **Proof of concept** with conference papers, online codes.
Get first feedback from the community.

2019–20 **Stable library** with solid theorems, a well-documented API.
KeOps backends for high-level packages.

2021–22 **Mature library** with focused application papers, full tutorials.
Works out-of-the-box for students and engineers.

2022+ **A standard toolbox**, with genuine clinical applications?
That's the target!

Conclusion

Key points

- **Symbolic** matrices are to **geometric** ML what
  sparse matrices are to **graph** processing:
  $\longrightarrow$ KeOps, **x30 speed-up** vs. PyTorch, TF and JAX.
  $\longrightarrow$ Useful in a wide range of settings.

- Optimal Transport = **generalized sorting**:
  $\longrightarrow$ Geometric gradients.
  $\longrightarrow$ Super-fast $O(N \log N)$ solvers.

- These tools open **new paths** for geometers and statisticians:
  $\longrightarrow$ 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.

Conclusion

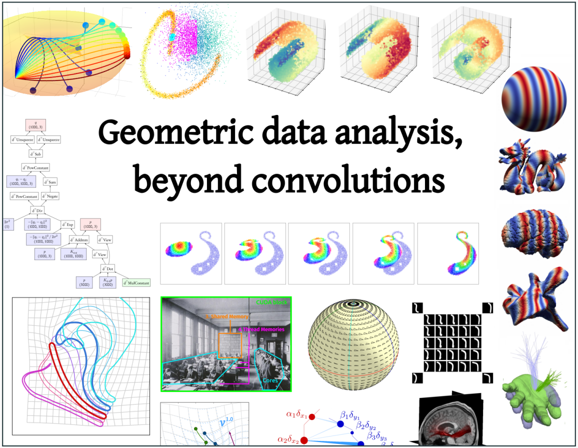
We believe that **KeOps** and **GeomLoss** will stimulate research on:

- **Clustering** methods: fast K-Means and EM iterations.
- Data **representation**: UMAP, fast KNN graphs with any metric.
- **Kernel** methods: kernel matrices.
- **Gaussian** processes: covariance matrices.
- **Geometric** deep learning: point convolutions.
- **Medical imaging**: computational anatomy.
- Geometric **statistics**: going beyond Euclidean models.
- Natural **language** processing: transformer networks?

What do you think?

Documentation and tutorials are available online

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



www.jeanfeydy.com/geometric_data_analysis.pdf

Dimitri P Bertsekas.
A distributed algorithm for the assignment problem.
Lab. for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA, 1979.

Y. Brenier.
Polar factorization and monotone rearrangement of vector-valued functions.
Comm. Pure Appl. Math., 44(4):375–417, 1991.

Grégoire Clarté, Antoine Diez, and Jean Feydy.
Collective proposal distributions for nonlinear MCMC samplers: Mean-field theory and fast implementation.
arXiv preprint arXiv:1909.08988, 2019.

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.

Pierre Degond, Amic Frouvelle, Sara Merino-Aceituno, and Ariane Trescases.
Alignment of self-propelled rigid bodies: from particle systems to macroscopic equations.
In *International workshop on Stochastic Dynamics out of Equilibrium*, pages 28–66. Springer, 2017.

Pierre Degond and Sébastien Motsch.
Continuum limit of self-driven particles with orientation interaction.
Mathematical Models and Methods in Applied Sciences, 18(supp01):1193–1215, 2008.

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.

Florent Leclercq.
Bayesian optimization for likelihood-free cosmological inference.
Physical Review D, 98(6):063511, 2018.

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.

Bernhard Schmitzer.
Stabilized sparse scaling algorithms for entropy regularized transport problems.
SIAM Journal on Scientific Computing, 41(3):A1443–A1481, 2019.

Freyr Sverrisson, Jean Feydy, Bruno E. Correia, and Michael M. Bronstein.
Fast end-to-end learning on protein surfaces.
bioRxiv, 2020.