

.

# Fast geometric libraries for vision and data sciences

---

Jean Feydy  
INRIA Paris

2nd of December, 2021  
AI and digital healthcare junior seminar  
Centre de Recherche des Cordeliers

# 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).

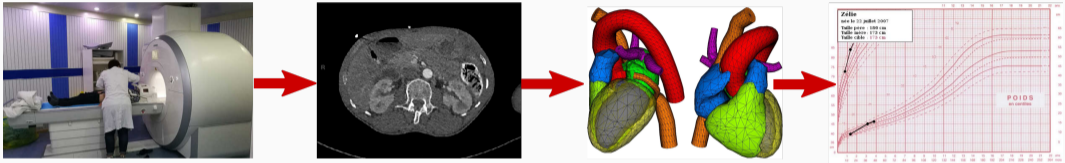
Close ties with **healthcare**:

2015 Image denoising with **Siemens Healthcare** in Princeton.

2019+ MasterClass AI–Imaging, for **radiology interns** in the University of Paris.

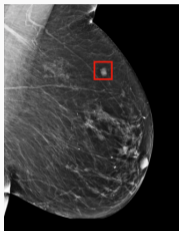
2020+ Colloquium on **Medical imaging in the AI era** at the Paris Brain Institute.

# My motivation: medical data analysis

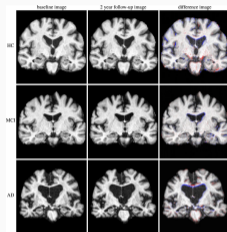


## Three main characteristics:

- **Heterogeneous data:** patient history, images, etc.
- Small stratified samples: 10 – 1 000 patients per group.
- Dealing with **outliers** and the **heavy tails** of our distributions is a priority.



Detect a pattern.



Analyze a variation.

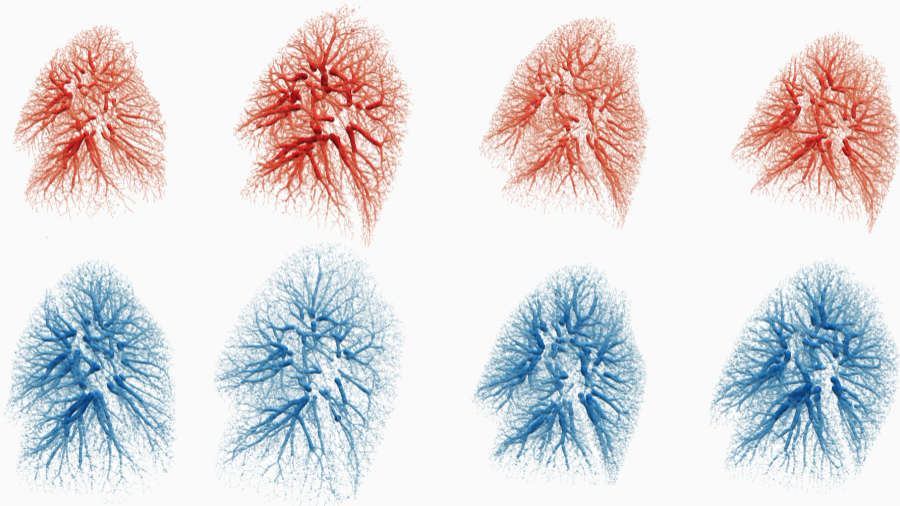


Register a model.

Some characteristics, in the wider context of computer vision research:

- **Standard acquisitions**, without occlusions.
- **Precision** work (at millimeter scale).
- Need for **guarantees** of robustness and regularity.

# Our main focus today: lung registration “Exhale – Inhale”



Complex deformations, high resolution (50k–300k points), high accuracy ( $< 1\text{mm}$ ).

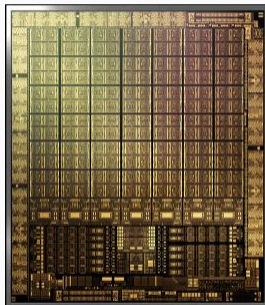
## A field that is moving fast

**Target.** Design models that combine medical **expertise** with modern **datasets**.

**Challenge.** The advent of **Graphics Processing Units (GPU)**:

- Incredible **value for money**:  
 $1\,000\text{€} \simeq 1\,000 \text{ cores} \simeq 10^{12} \text{ operations/s.}$
- **Bottleneck**: constraints on **register** usage.

“User-friendly” Python ecosystem, consolidated around a **small number of key operations**.



**7,000 cores**  
in a single GPU.

## My project: a long-term investment in the foundations of our field

**Solution.** Expand the **standard toolbox** in data sciences to deal with the challenges of the healthcare industry.

Ease the development of **advanced models**, without compromising on numerical performance.

In-depth work, numerical **foundations** → **high-level** applications:

1. Efficient manipulation of “**symbolic**” **matrices** (distances, kernel, etc.).
2. **Optimal transport**: generalized sorting methods.
3. Geometric **deep learning**.

**Discussion** about the **future** of these tools and **clinical** perspectives.

# 1. Symbolic matrices

---



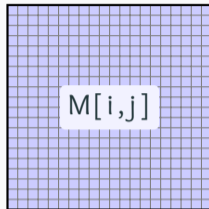
# Computing libraries represent most objects as tensors

**Context.** Constrained **memory accesses** on the GPU:

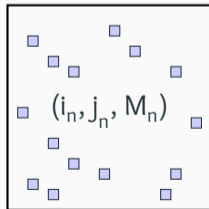
- **Long access times** to the registers penalize the use of large **dense** arrays.
- Hard-wired **contiguous** memory accesses penalize the use of **sparse** matrices.

**Challenge.** In order to reach optimal run times:

- **Restrict** ourselves to operations that are supported by the constructor: convolutions, FFT, etc.
- Develop new routines from scratch in C++/CUDA (FAISS, KPCConv...): **several months of work.**



Dense array



Sparse matrix

# The KeOps library: efficient support for symbolic matrices

**Solution.** KeOps – [www.kernel-operations.io](http://www.kernel-operations.io):

- For PyTorch, NumPy, Matlab and R, on **CPU and GPU**.
- **Automatic differentiation**.
- Just-in-time **compilation** of **optimized C++** schemes, triggered for every new **reduction**: sum, min, etc.

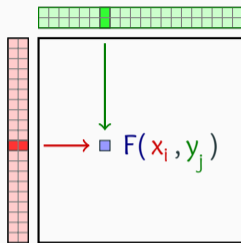
If the formula “F” is simple ( $\leq 100$  arithmetic operations):

“100k  $\times$  100k” computation  $\rightarrow$  10ms – 100ms,

“1M  $\times$  1M” computation  $\rightarrow$  1s – 10s.

Hardware ceiling of  $10^{12}$  operations/s.

$\times 10$  to  $\times 100$  **speed-up** vs standard GPU implementations  
for a wide range of problems.



**Symbolic matrix**

Formula + data

- Distances  $d(x_i, y_j)$ .
- Kernel  $k(x_i, y_j)$ .
- Numerous transforms.

## 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.argmax(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!**

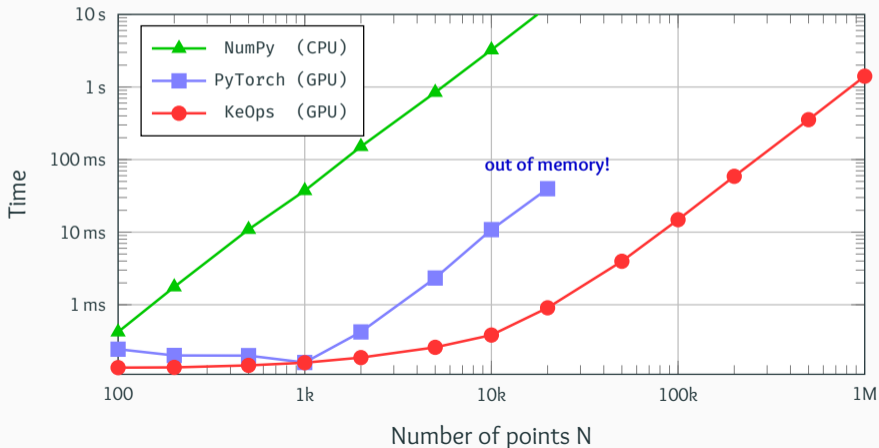
```
D_ij = ((x_i - x_j) ** 2).sum(dim=2)      # Euclidean
M_ij = (x_i - x_j).abs().sum(dim=2)     # Manhattan
C_ij = 1 - (x_i | x_j)                  # Cosine
H_ij = D_ij / (x_i[...,0] * x_j[...,0]) # Hyperbolic
```

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

- **Reductions:** sum, log-sum-exp, K-min, matrix-vector product, etc.
- **Operations:** +, ×, 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

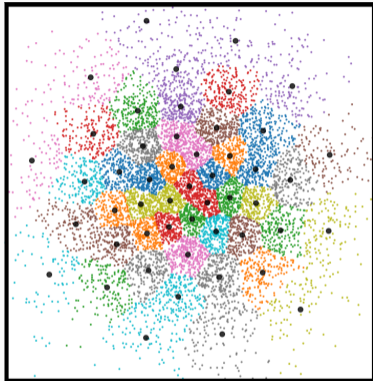
Benchmark of a Gaussian **convolution**  
between **clouds** of  $N$  3D points on a RTX 2080 Ti GPU.



# Applications

---

# KeOps is a good fit for machine learning research



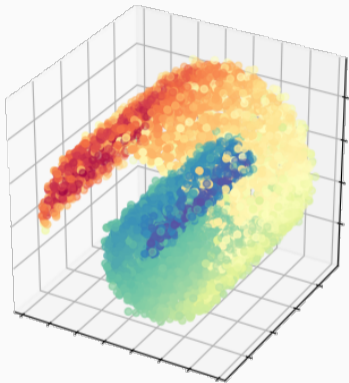
K-Means.



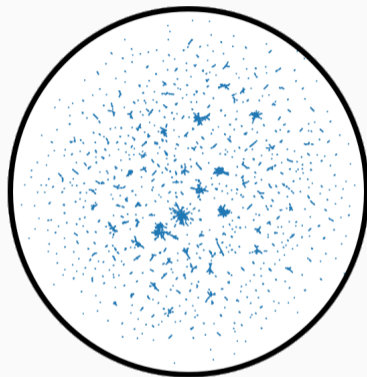
Gaussian Mixture Model.

Use **any** kernel, metric or formula **you** like!

# KeOps is a good fit for machine learning research



Spectral analysis.

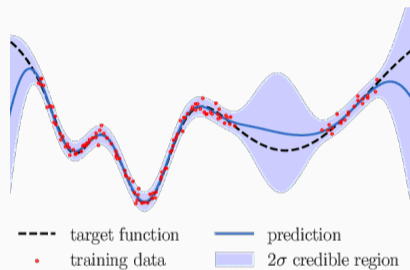


UMAP in hyperbolic space.

Use **any** kernel, metric or formula **you** like!



A standard tool for regression [Lec18]:



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

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

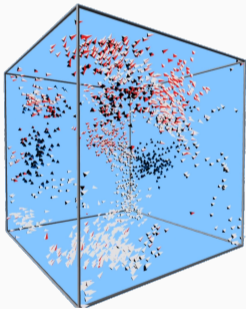
where  $\lambda \geq 0$  et  $(K_{xx})_{i,j} = k(x_i, x_j)$  is a positive definite matrix.

KeOps symbolic tensors  $(K_{xx})_{i,j} = k(x_i, x_j)$ :

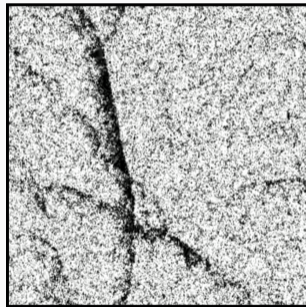
- Can be fed to **standard solvers**: SciPy, GPyTorch, etc.
- GPytorch on the 3DRoad dataset (N = 278k, D = 3):  
**7h avec 8 GPUs** → **15mn avec 1 GPU.**
- Provide a **fast backend for research codes**:  
see e.g. *Kernel methods through the roof: handling **billions of points** efficiently*,  
by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

# 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 et P. Degond:



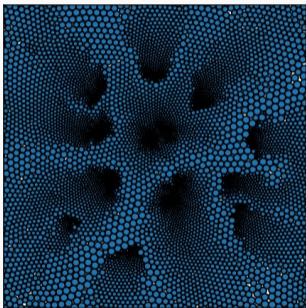
3D Vicsek model with orientation,  
interactive demo with 2k **flyers**.



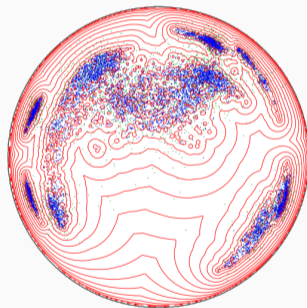
2D Vicsek model on the torus,  
in real-time with 100k **swimmers**.

# KeOps lets you focus on your models, results and theorems

⇒ 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.

## 2. Back to shapes: optimal transport

---

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

**Context.** 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:

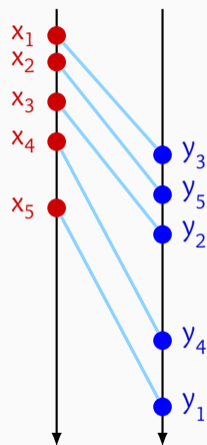
$$\text{OT}(A, B) = \min_{\sigma \in \mathcal{S}_N} \frac{1}{2N} \sum_{i=1}^N \|x_i - y_{\sigma(i)}\|^2$$

Generalizes **sorting** to metric spaces.

**Linear problem** on the permutation matrix  $P$ :

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

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



assignment  
 $\sigma : [1, 5] \rightarrow [1, 5]$

## Key properties of this distance “up to permutations”

The Wasserstein distance  $\sqrt{\text{OT}}(\mathbf{A}, \mathbf{B})$  is:

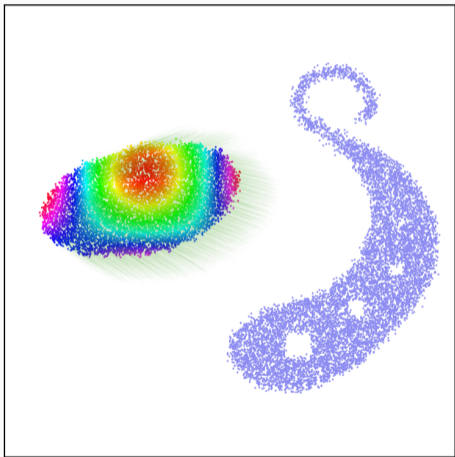
- **Symmetric:**  $\text{OT}(\mathbf{A}, \mathbf{B}) = \text{OT}(\mathbf{B}, \mathbf{A})$ .
- **Positive:**  $\text{OT}(\mathbf{A}, \mathbf{B}) \geq 0$ .
- **Definite:**  $\text{OT}(\mathbf{A}, \mathbf{B}) = 0 \iff \mathbf{A} = \mathbf{B}$ .
- **Translation-aware:**  $\text{OT}(\mathbf{A}, \text{Translate}_{\vec{v}}(\mathbf{A}))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

$$\text{In dimension 1, } (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{y}_{\sigma(i)} - \mathbf{y}_{\sigma(j)}) \geq 0$$

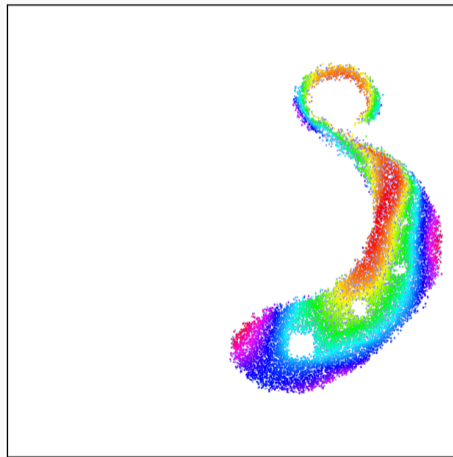
$$\text{In dimension D, } \langle \mathbf{x}_i - \mathbf{x}_j, T(\mathbf{x}_i) - T(\mathbf{x}_j) \rangle_{\mathbb{R}^D} \geq 0.$$

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

# An efficient model... but beware of tears!



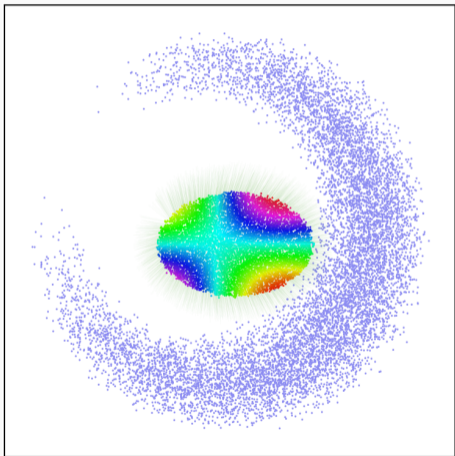
Before



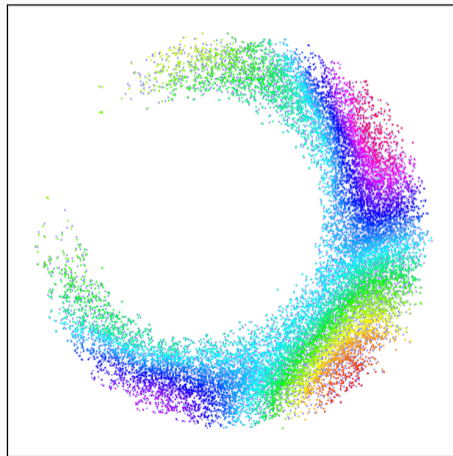
After



# An efficient model... but beware of tears!

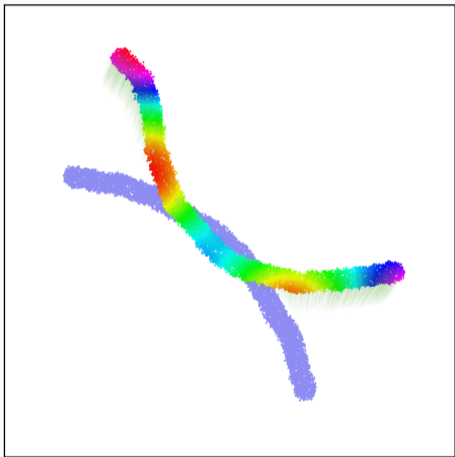


Before

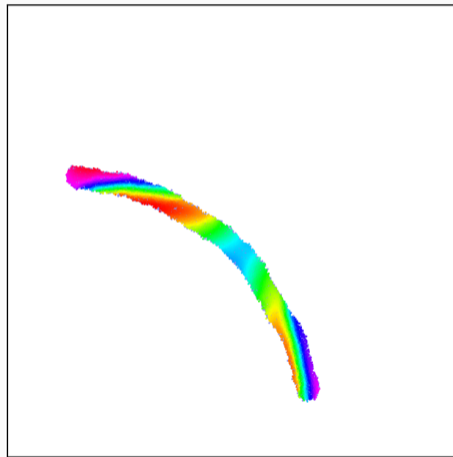


After

# An efficient model... but beware of tears!

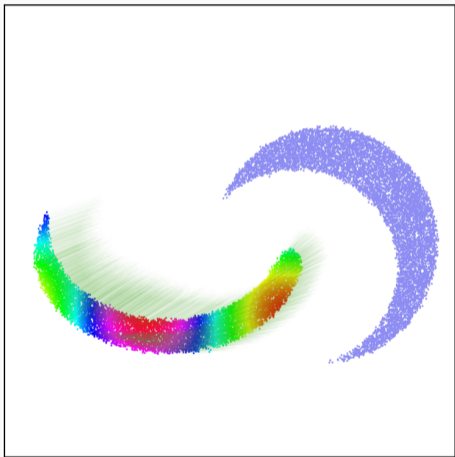


Before

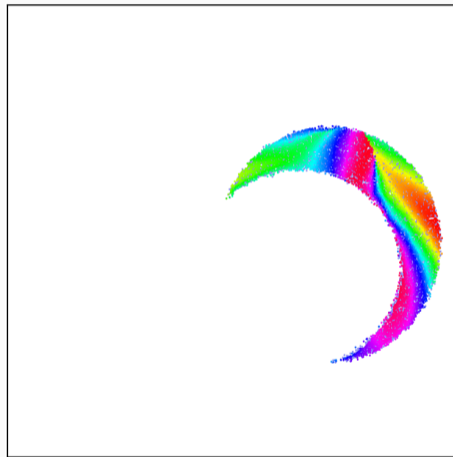


After

# An efficient model... but beware of tears!



Before



After

## How should we solve the OT problem?

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<sup>+</sup>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**.  
     $\implies$  Generalized **QuickSort** algorithm.

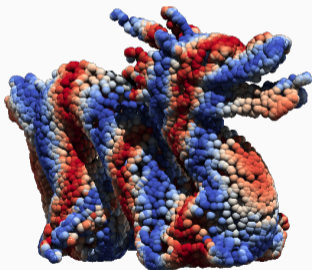
## Scaling up optimal transport to anatomical data

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

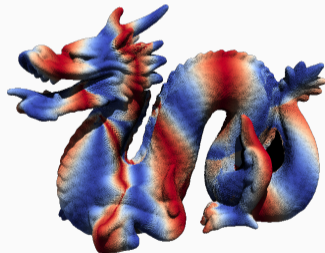
Sinkhorn GPU  $\xrightarrow{\times 10}$  + KeOps  $\xrightarrow{\times 10}$  + Annealing  $\xrightarrow{\times 10}$  + Multi-scale

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

### 3. Geometric deep learning

---

# Design task-specific trainable models

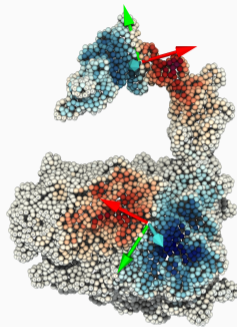
**Context.** Trainable models on **non-Euclidean domains** (point clouds, surfaces, graphs, etc.), beyond 2D/3D images.

**Challenge.** In spite of growing interest in the industry, these models still **lack support** on the numerical side. C++/CUDA is (often) required to reach top performance.

**Solution.** Using KeOps, with a few lines of Python:

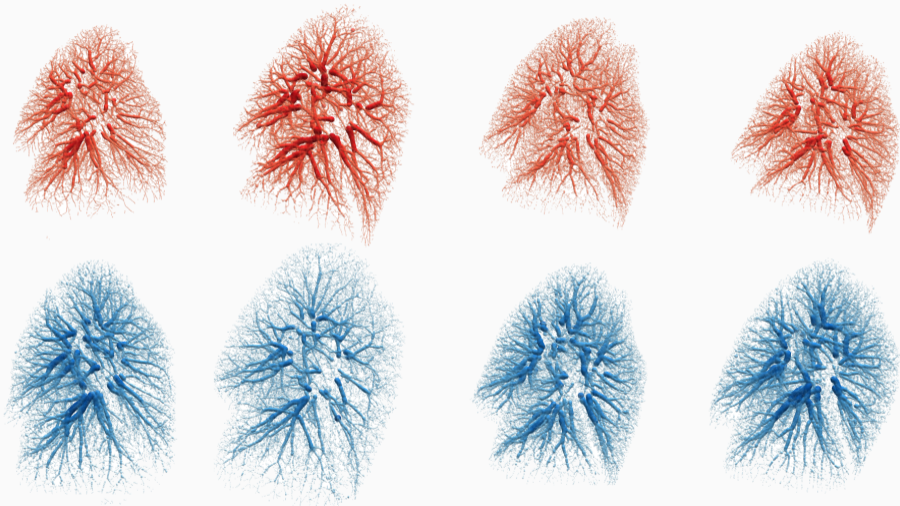
- **Local** interactions: K-nearest neighbors.
- **Global** interactions: generalized convolutions.

Modelling **freedom**  
⇒ **Domain-specific** priors.



Quasi-geodesic convolution on a protein surface.

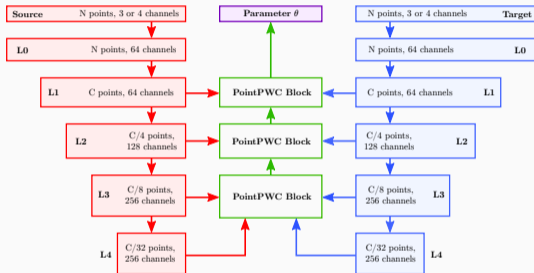
## Lung registration “Exhale – Inhale”



Complex deformations, high resolution (50k–300k points), high accuracy ( $< 1\text{mm}$ ).



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



Multi-scale convolutional  
point neural network.

## Point neural nets, in practice:

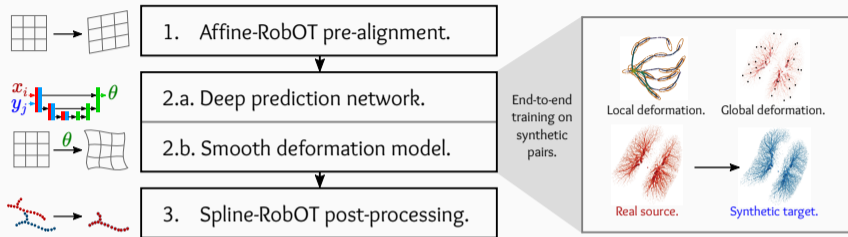
- Compute **descriptors** at all scales.
- **Match** them using geometric layers.
- Train on **synthetic** deformations.

## Strengths and weaknesses:

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

⇒ **Complementary** to OT.

# Three-steps registration

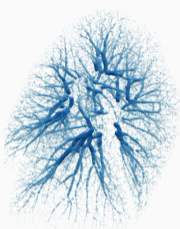


This **pragmatic** method:

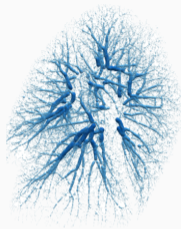
- Is **easy to train** on synthetic data.
- Scales up to high-resolution: 100k points in 1s.
- Excellent results: **KITTI** (outdoors scans) and **DirLab** (lungs).

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

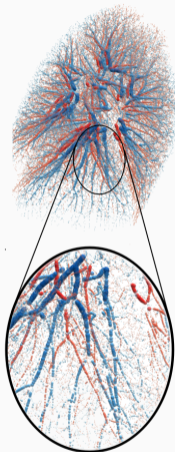
# Three-steps registration



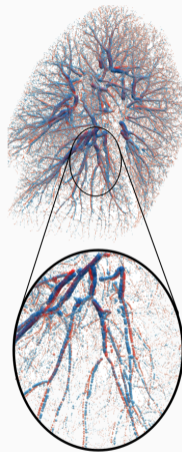
0. Input data



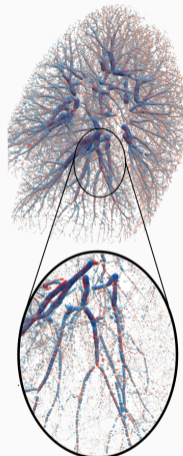
1. Pre-alignment



Zoom !



2. Deep registration



3. Fine-tuning

## Conclusion

---

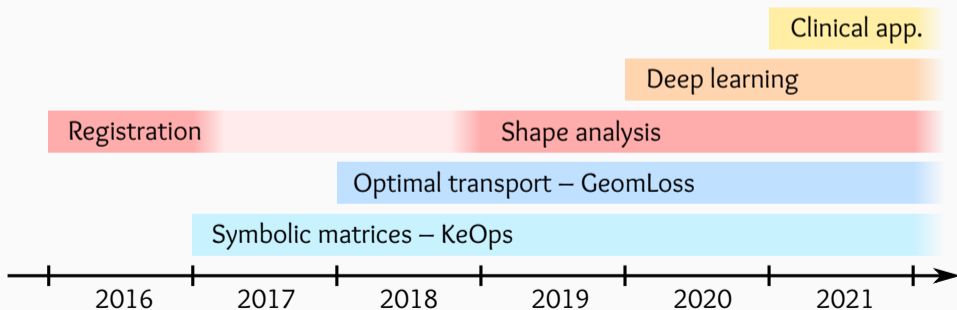
## Key points

- **Symbolic** matrices are to **geometric** ML what **sparse** matrices are to **graph** processing:
  - KeOps: **x30 speed-up** vs. PyTorch, TF et JAX.
  - Useful in a wide range of settings.
- Optimal Transport = **generalized sorting** :
  - Geometric gradients.
  - Super-fast  $O(N \log N)$  solvers.
- These tools open **new paths** for geometers and statisticians:
  - 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.

## Summary: a long-term investment that is starting to bear fruits

Two major evolutions:

- “Big” geometric problem:  $N > 10k \rightarrow N > 1M$ .
- Optimal transport: linear **problem** + generalized **quicksort**.



## Genuine team work



Alain Trouvé



Thibault Séjourné



F.-X. Vialard



Gabriel Peyré



Benjamin Charlier



Joan Glaunès



Freyr Sverrisson

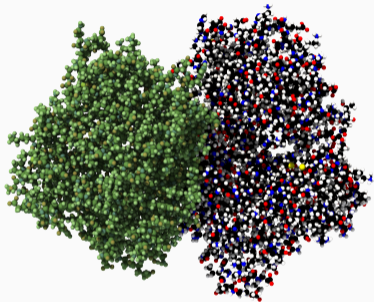


Shen Zhengyang

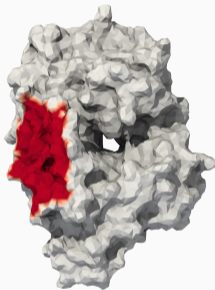
+ Marc Niethammer, Bruno Correia, Michael Bronstein...



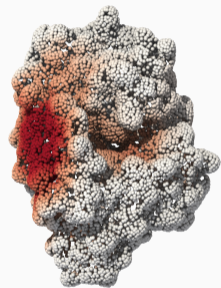




(a) Raw protein data.

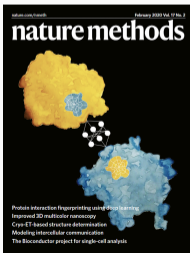
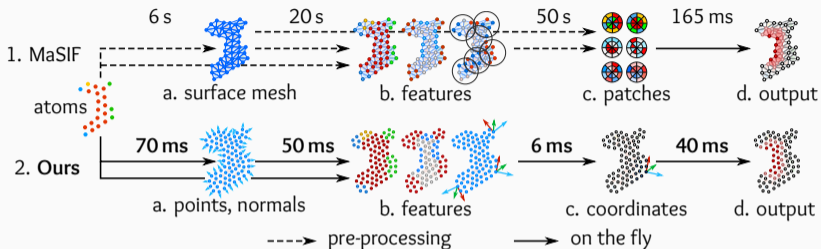


(b) Interface.



(c) Prediction.

# Fast end-to-end learning on protein surfaces



→ ×100 - ×1,000 faster, lighter  
and fully differentiable.

## References

---

 Dimitri P Bertsekas.

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

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

 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.*

 Christophe Chnafa, Simon Mendez, and Franck Nicoud.

**Image-based large-eddy simulation in a realistic left heart.**

*Computers & Fluids*, 94:173–187, 2014.

 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.



Adam Conner-Simons and Rachel Gordon.

**Using ai to predict breast cancer and personalize care.**

<http://news.mit.edu/2019/using-ai-predict-breast-cancer-and-personalize-care-0507>,  
2019.

MIT CSAIL.



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.

 Christian Ledig, Andreas Schuh, Ricardo Guerrero, Rolf A Heckemann, and Daniel Rueckert.



**Structural brain imaging in Alzheimer's disease and mild cognitive impairment: biomarker analysis and shared morphometry database.**

*Scientific reports*, 8(1):11258, 2018.

 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.