Geometric data analysis, beyond convolutions

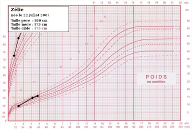
Jean Feydy,
under the supervision of Alain Trouvé and Michael Bronstein.

Geometry, ML and Health workshop at UCL, online — September 25, 2020.

ENS Paris, ENS Paris-Saclay, Imperial College London.

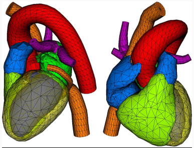
Joint work with B. Charlier, J. Glaunès (numerical foundations),
T. Séjourné, F.-X. Vialard, G. Peyré (optimal transport theory),
P. Roussillon, P. Gori (applications to neuroanatomy).

The medical imaging pipeline [Ptr19, EPW+11]



Valuable information

Statistics



High-level description

Computational anatomy

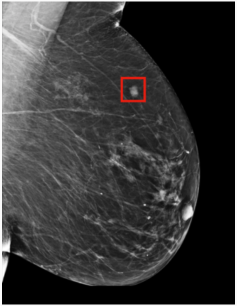
Raw image

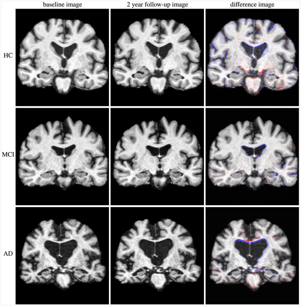
Signal processing

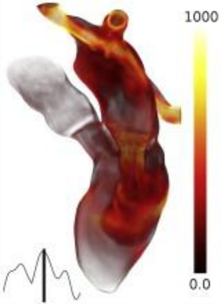
Sensor data

Computational anatomy [CSG19, LSG+18, CMN14]

Three main problems:







Spot patterns

Analyze variations

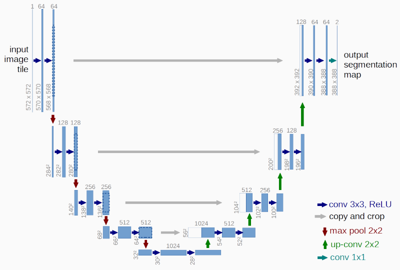
Fit models

Wavelet/Radiomics-like architectures + data-driven optimization

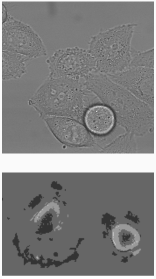
$\Longrightarrow$ Convolutional Neural Networks.

A revolution for **feature** detection and **texture** models.

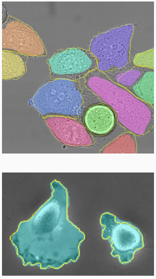
Segmentation with U-nets [RFB15]:



Architecture



Input



Output

Geometric problems are becoming increasingly relevant

Geometric questions on segmented **shapes**:

- Is this **heart** beating all right?
- How should we reconstruct this **mandible**?
- Has this **brain** grown or shrunk since last year?
- Can we link these anatomical changes to **other signals**?

Geometric problems are becoming increasingly relevant

Geometric questions on segmented **shapes**:

- Is this **heart** beating all right?
- How should we reconstruct this **mandible**?
- Has this **brain** grown or shrunk since last year?
- Can we link these anatomical changes to **other signals**?

Over the last 30 years, **robust methods** have been designed to answer these questions.

Today, we want to improve them with **data-driven** insights. This is challenging.

To replicate the *deep learning* revolution in this field, we need to **revamp our numerical toolbox**.

Motivations

Geometric data analysis, beyond convolutions:

- Focus on **geometric data**:
  segmentation maps, point clouds, surface meshes, etc.
- Focus on **geometric methods**:
  K-nearest neighbors, kernel methods, optimal transport, etc.
- Provide new **computational routines**:
  expand the toolbox for data sciences.

We usually work with 10^3-10^6 points in dimension 2 to 10.
We focus on geometry and speed.

Outline of the talk

Today, we will talk about:

1. **Fast geometry** with symbolic matrices.
2. Scalable **optimal transport**.
3. Applications and **references**.

Fast geometry with symbolic matrices.

Benjamin Charlier

Joan Glaunès

Deep learning frameworks: unlocking GPUs for research

TensorFlow and PyTorch combine:

- $+$ Array-centric **Python interface**.
- $+$ CPU and **GPU** backends.
- $+$ **Automatic differentiation** engine.
- $+$ Excellent support for imaging (convolutions) and linear algebra.

$\Longrightarrow$ Ideally suited for research.

Efficient algorithms still rely on C++ foundations

Explicit **C++/CUDA implementations** with a **Python interface** for:

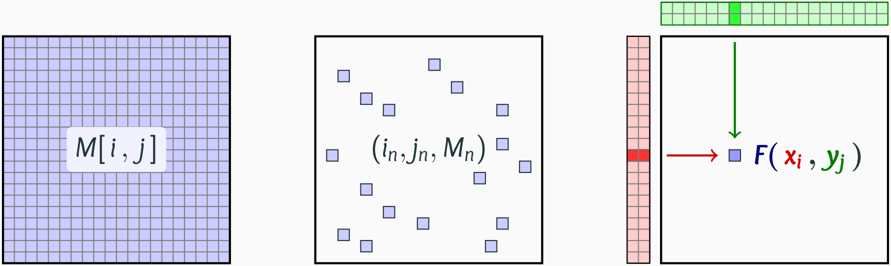
- Linear algebra (cuBLAS).
- Convolutions (cuDNN).
- Fourier (cuFFT) and wavelet transforms (Kymatio).

Geometric algorithms do not benefit from the same level of integration. Researchers can either:

- Work directly in C++/CUDA – **cumbersome** for data sciences.
- Rely on sparse matrices and graphs with **small neighborhoods**.
- Rely on **explicit distance matrices**.

`RuntimeError: cuda runtime error (2) : out of memory at /opt/conda/.../THCStorage.cu:66`

We provide efficient support for distance-like matrices



Dense matrix
Coefficients only

Sparse matrix
Coordinates + coeffs

Symbolic matrix
Formula + data

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

"KErnel OPerationS"

KeOps works with PyTorch, NumPy, Matlab and R

```
# Large point cloud in ℝ⁵⁰:
import torch
N, D = 10**6, 50
x = torch.rand(N, D).cuda()  # (1M, 50) array

# Compute the nearest neighbor of every point:
from pykeops.torch import LazyTensor
x_i = LazyTensor(x.view(N, 1, D))   # x_i is a "column"
x_j = LazyTensor(x.view(1, N, D))   # x_j is a "line"
D_ij = ((x_i - x_j)**2).sum(dim=2)  # (N, N) symbolic
indices_i = D_ij.argmin(dim=1)      # -> (N,) dense
```

On par with reference C++/CUDA libraries (FAISS-GPU).

Combining performance and flexibility

We can work with arbitrary formulas:

```
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
```

$\Longrightarrow$ $\times$200 acceleration for UMAP on hyperbolic spaces.

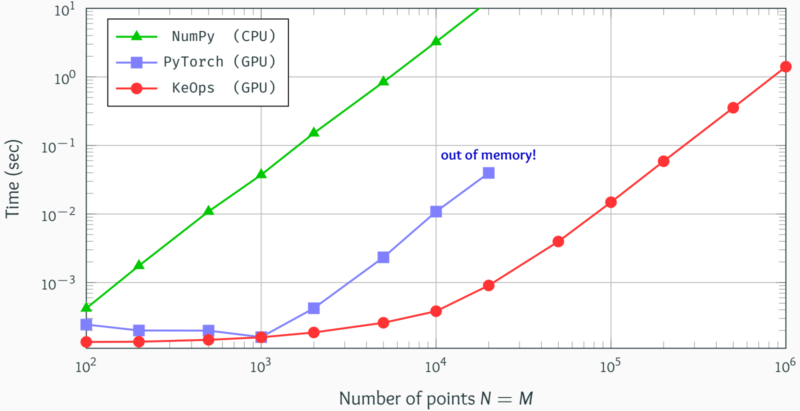
KeOps supports:

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

Scaling up to large datasets

$$a_i \leftarrow \sum_{j=1}^{M} \underbrace{\exp(-\|x_i - y_j\|^2/2\sigma^2)}_{k(x_i,y_j)}\ b_j, \quad \forall i \in [\![1, N]\!]$$

Gaussian kernel product in 3D (RTX 2080 Ti GPU)



The KeOps library

+ **Cross-platform**: C++, R, Matlab, NumPy and PyTorch.
+ **Versatile**: many operations, variables, reductions.
+ **Efficient**: $O(N)$ memory, competitive runtimes.
+ **Powerful**: automatic differentiation, block-sparsity, etc.
+ **Transparent**: interface with **SciPy**, GPytorch, etc.
+ **Fully documented**:

`www.kernel-operations.io`

→ Kriging, splines, Gaussian processes, **kernel** methods.

→ Geometry processing, **geometric** deep learning.

(More illustrations at the end of the talk!)

Computational optimal transport

Thibault Séjourné

F.-X. Vialard

Gabriel Peyré

We need robust loss functions for shape analysis

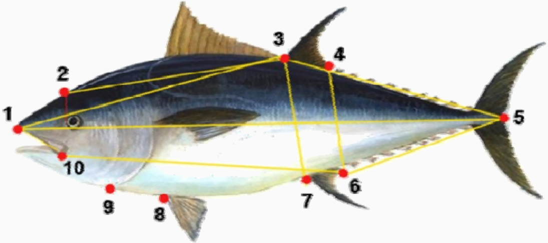
Working with point clouds is now **easier than ever**.
We can protoype new geometric algorithms in minutes.

But how should we **measure success** and **errors**?

$\Longrightarrow$ We must develop **geometric loss functions**
to compute distances between shapes.

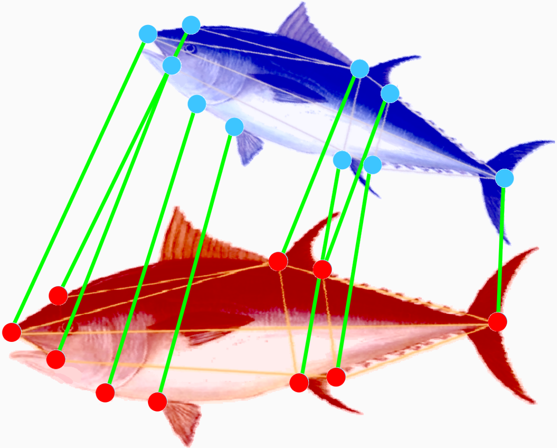
High-quality gradients will improve the **robustness**
of registration or training algorithms
and allow us to **focus on our models**.

Life is easy when you have landmarks...



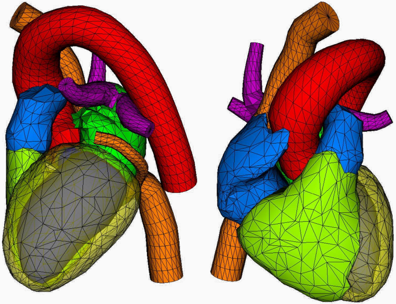
Anatomical landmarks from *A morphometric approach for the analysis of body shape in bluefin tuna*, Addis et al., 2009.

Life is easy when you have landmarks...

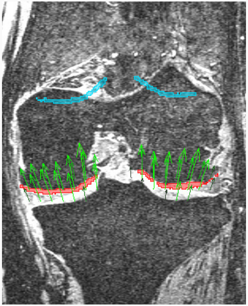


Anatomical landmarks from *A morphometric approach for the analysis of body shape in bluefin tuna*, Addis et al., 2009.

Unfortunately, medical data is often weakly labeled [EPW+11]



Surface meshes

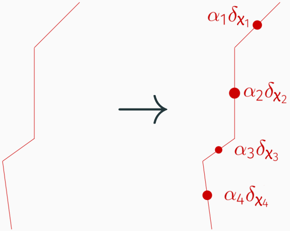


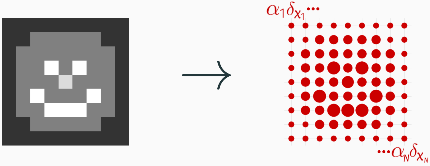
Segmentation masks

Encoding unlabeled shapes as measures

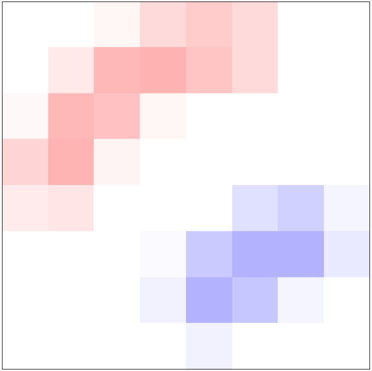
Let's enforce sampling invariance:

$$A \longrightarrow \alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \qquad B \longrightarrow \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

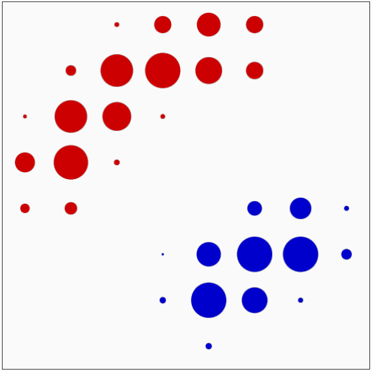




A baseline setting: density registration

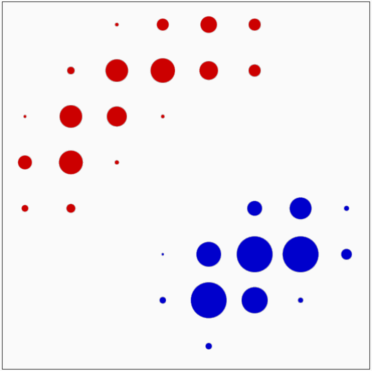


A baseline setting: density registration



$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i} , \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j} .$$

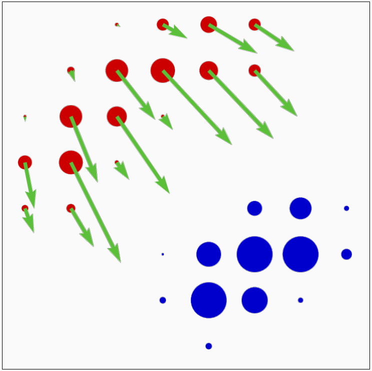
A baseline setting: density registration



$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

$$\sum_{i=1}^{N} \alpha_i \;=\; 1 \;=\; \sum_{j=1}^{M} \beta_j$$

A baseline setting: density registration

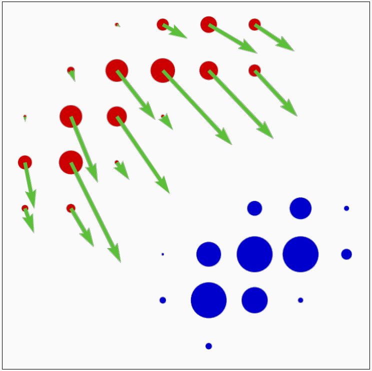


$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}.$$

$$\sum_{i=1}^{N} \alpha_i = 1 = \sum_{j=1}^{M} \beta_j$$

Display $v_i = -\frac{1}{\alpha_i} \nabla_{x_i} \text{Loss}(\alpha, \beta)$.

A baseline setting: density registration



$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

$$\sum_{i=1}^{N} \alpha_i \;=\; 1 \;=\; \sum_{j=1}^{M} \beta_j$$

Display $v_i \;=\; -\frac{1}{\alpha_i} \nabla_{x_i} \text{Loss}(\alpha, \beta)$.

Seamless extensions to:

- $\sum_i \alpha_i \neq \sum_j \beta_j$, outliers [CPSV18],
- curves and surfaces [KCC17],
- variable weights α_i.

The Wasserstein distance

We need **clean gradients**, without artifacts.

The Wasserstein distance

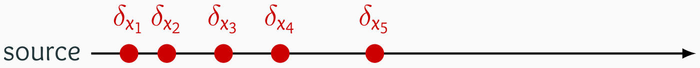
We need **clean gradients**, without artifacts.

Simple toy example in 1D :

The Wasserstein distance

We need **clean gradients**, without artifacts.

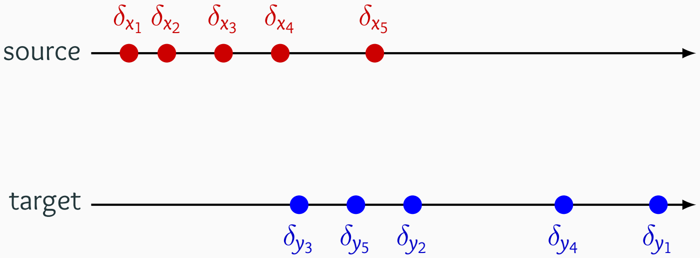
Simple toy example in 1D :



The Wasserstein distance

We need **clean gradients**, without artifacts.

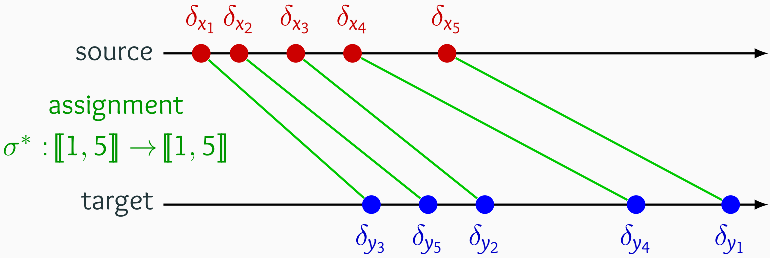
Simple toy example in 1D :



The Wasserstein distance

We need **clean gradients**, without artifacts.

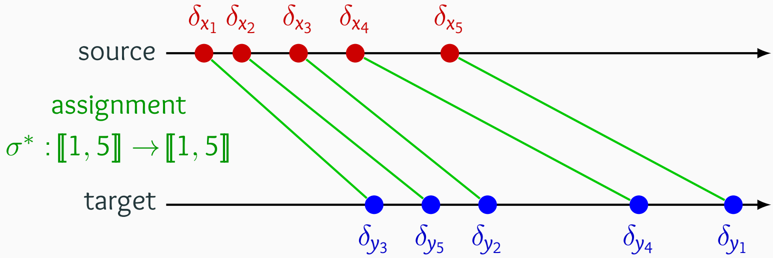
Simple toy example in 1D :



The Wasserstein distance

We need **clean gradients**, without artifacts.

Simple toy example in 1D :

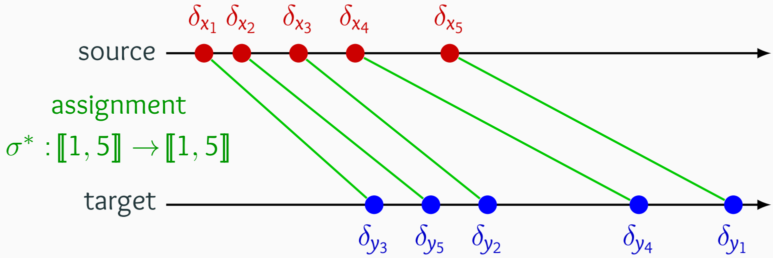


$$\mathrm{OT}(\alpha, \beta) \;=\; \frac{1}{2N} \sum_{i=1}^{N} \left| x_i - y_{\sigma^*(i)} \right|^2$$

The Wasserstein distance

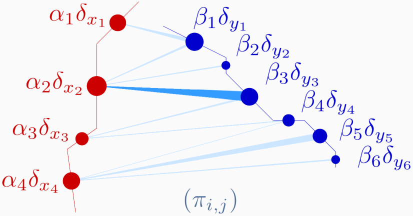
We need **clean gradients**, without artifacts.

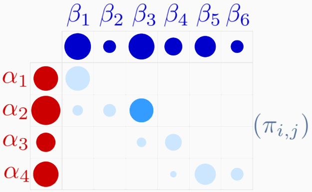
Simple toy example in 1D :



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

Optimal transport generalizes sorting to D > 1





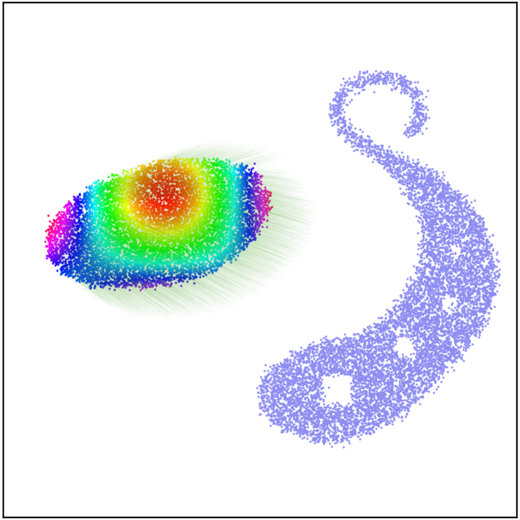
Minimize over *N*-by-*M* matrices (transport plans) π :

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

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

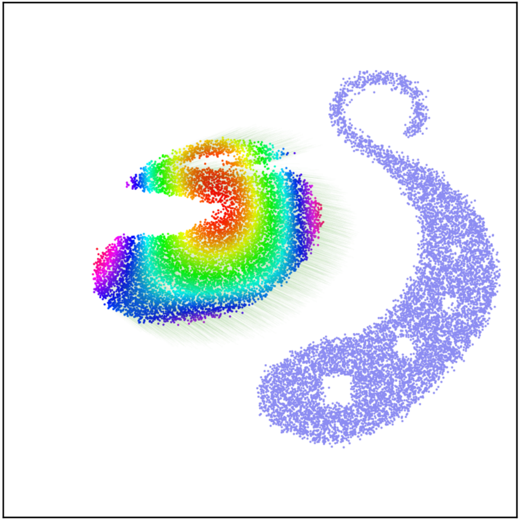
$$\sum_j \pi_{i,j} = \alpha_i, \quad \sum_i \pi_{i,j} = \beta_j.$$

Gradient flow as a toy registration: $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



$t = .00$

Gradient flow as a toy registration: $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$

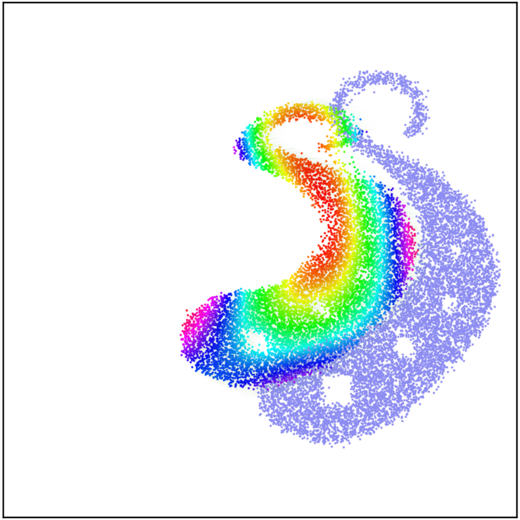


$t = .25$

Gradient flow as a toy registration: $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$

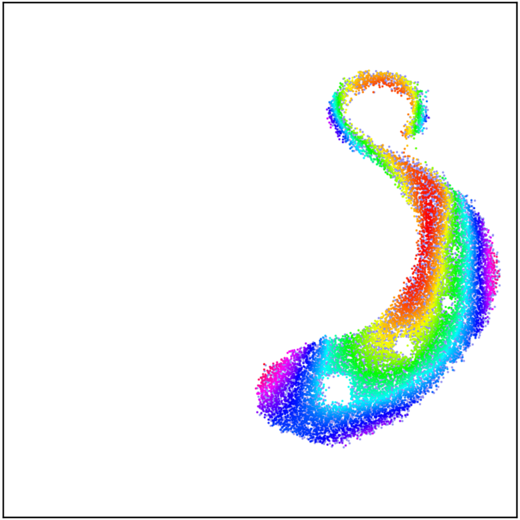
$t = .50$

Gradient flow as a toy registration: $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



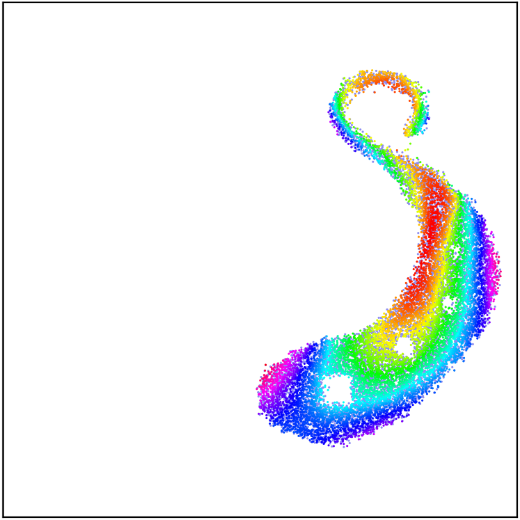
$t = 1.00$

Gradient flow as a toy registration: $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



$t = 5.00$

Gradient flow as a toy registration: $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



$t = 10.00$

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.**

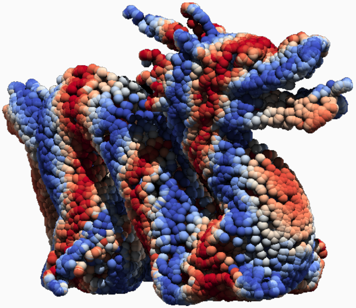
$\Longrightarrow$ Generalized **QuickSort** algorithm.

Scaling up optimal transport to anatomical data

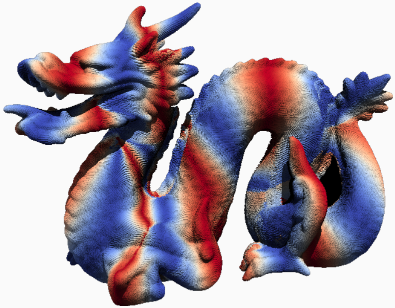
These progresses add up to a $\times 100$ - $\times 1000$ **acceleration:**

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` $\Longleftarrow$

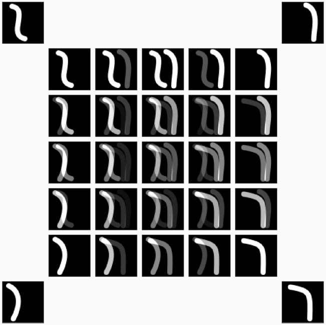
```python
# Large point clouds in [0,1]^3
import torch
x = torch.rand(100000, 3, requires_grad=True).cuda()
y = torch.rand(200000, 3).cuda()

# 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 **bitmaps**, **meshes** and generic metrics.

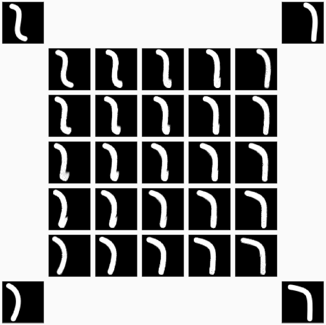
Affordable geometric interpolation [AC11]

$$\text{Barycenter } {\color{red}\alpha^*} = \arg\min_{\color{red}\alpha} \sum_{i=1}^{N} \lambda_i \,\text{Loss}(\,{\color{red}\alpha}\,,\, {\color{blue}\beta_i}\,)\,.$$



Linear barycenters

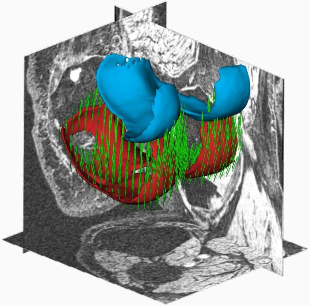
$\text{Loss}({\color{red}\alpha}, {\color{blue}\beta}) = \|{\color{red}\alpha} - {\color{blue}\beta}\|^2_{L^2}$



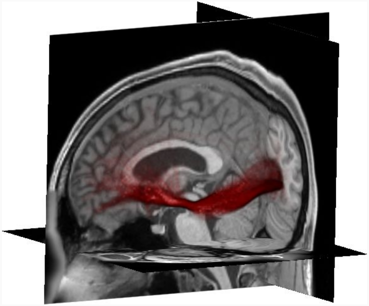
Wasserstein barycenters

$\text{Loss}({\color{red}\alpha}, {\color{blue}\beta}) = \text{OT}({\color{red}\alpha}, {\color{blue}\beta})$

In medical imaging: fast interpolation between "simple" shapes

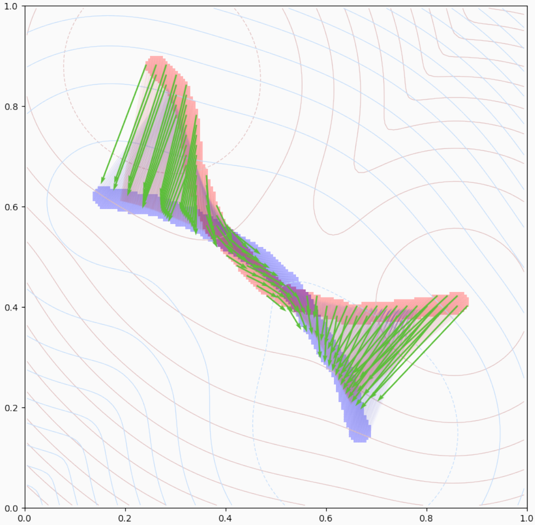


Knee caps



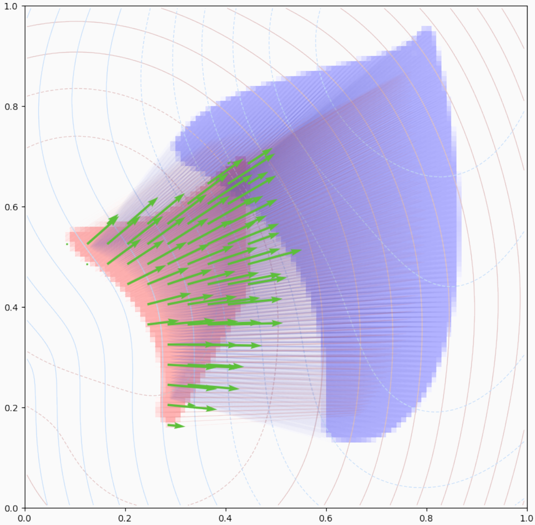
White matter bundles

A global and geometric loss function



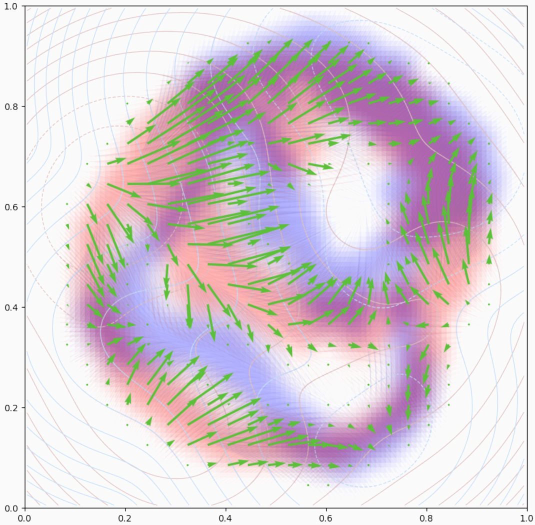
A high-quality gradient...

A global and geometric loss function



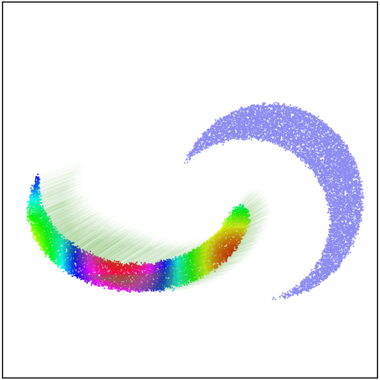
A high-quality gradient...

A global and geometric loss function

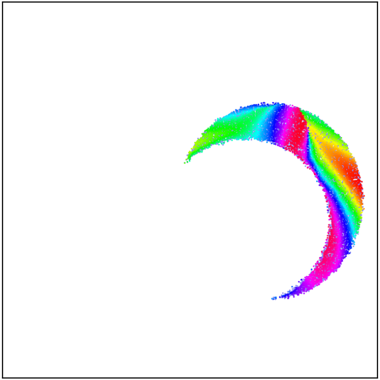


A high-quality gradient... But no preservation of topology!

Optimal transport = cheap’n easy registration? Beware!



Before



After

$\Longrightarrow$ Guaranteeing the preservation of topology is **much harder:**
see Chapter 5 of my PhD thesis.

Applications

Overview

Main motivation: **make shape analysis easy**.

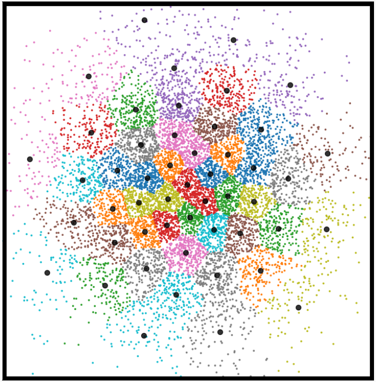
Working with **shapes** ought to be as simple as dealing with **vectors.**

Two modern and robust tools to unlock research in the field:

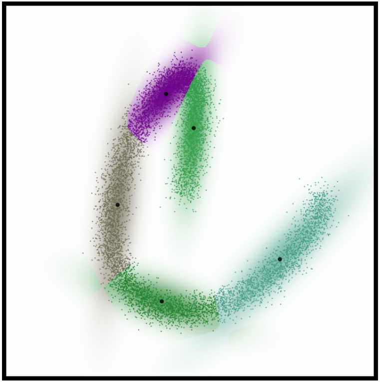
+ **Symbolic matrices:** fast and versatile.
+ **Geometric Loss functions:** high-quality gradients.

$\Longrightarrow$ Very useful outside of medical imaging too!

KeOps is a good fit for machine learning research



K-Means.

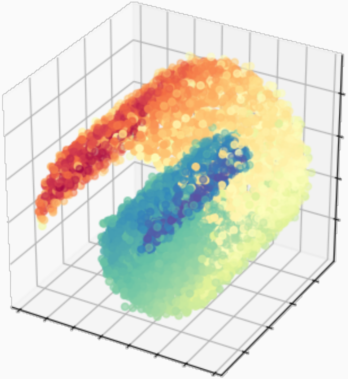


Gaussian Mixture Model.

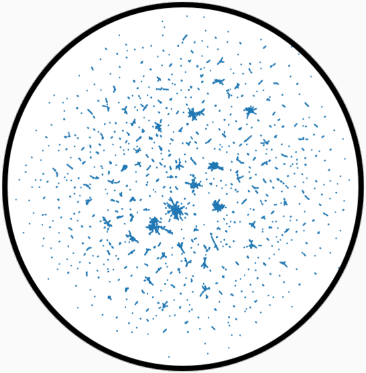
Use any kernel, metric or formula you like!

$\Longrightarrow$ More tutorials coming up in October.

KeOps is a good fit for machine learning research



Spectral analysis.



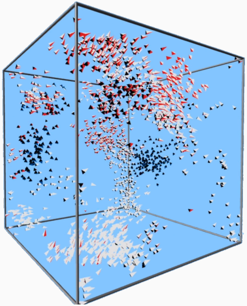
UMAP in hyperbolic space.

Use any kernel, metric or formula you like!

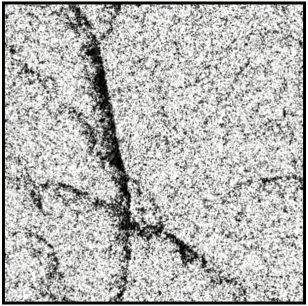
$\Longrightarrow$ More tutorials coming up in October.

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:



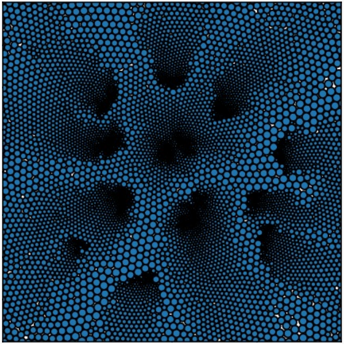
3D Vicsek model with orientation, interactive demo with 2k **flyers**.



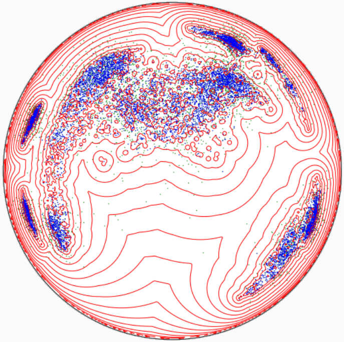
2D Vicsek model on the torus, in real-time with 100k **swimmers**.

KeOps lets you focus on your models, results and theorems

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



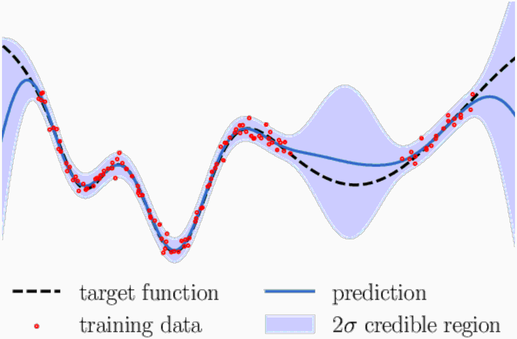
Packing problem in 2D
with 10k repulsive balls.



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

Applications to Kriging, spline, Gaussian process, kernel regression

A standard tool for regression [Lec18]:



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

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

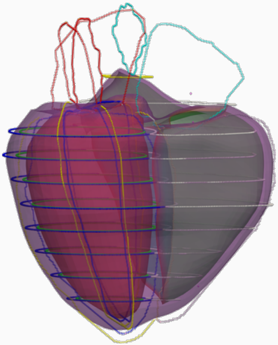
where $\lambda \geqslant 0$ and $(K_{xx})_{i,j} = k(x_i, x_j)$ is a positive definite matrix.

Applications to Kriging, spline, Gaussian process, kernel regression

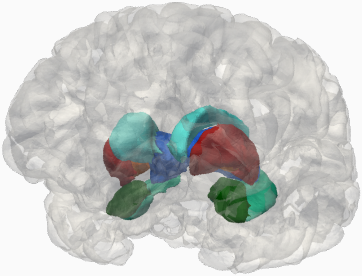
KeOps symbolic tensors:

- 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).

My first motivation: computational anatomy



Fast OT-based registration
with Samuel Joutard, Xu Hao
and Alistair Young from KCL.

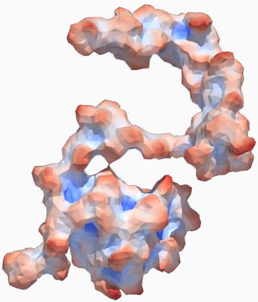


Diffeomorphic and spline registration
e.g. Deformetrica LDDMM software
with the Aramis Inria team.

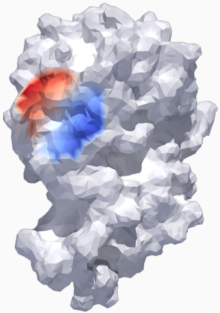
Geometric deep learning w. Freyr Sverrisson and Michael Bronstein

Data-driven geometric methods on **point clouds**:

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



Mean **curvature** at all scales.



Tangent **convolutions.**

Future improvements

KeOps and GeomLoss are:

- $+$ **Fast:** $\times 10$ - $\times 1{,}000$ speedup vs. naive GPU implementations.
- $+$ **Memory-efficient:** $O(\mathrm{N})$, not $O(\mathrm{N}^2)$.
- $+$ **Versatile, with a transparent interface**: freedom!
- $+$ **Powerful and well-documented:** research-friendly.
- $-$ Slow with **large vectors** of dimension $\mathrm{D} > 50$.

Future improvements

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 $D > 50$.

Christmas 2020: fix register spilling, add support for **Tensor cores.**

- $\rightarrow$ **Dramatic speed-ups** when $16 \leqslant D \leqslant 1{,}000$.
- $\rightarrow$ **Applications to NLP:** attention layers, Word Mover's Distance.

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.

2020–22 **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 key to performance:
  $\longrightarrow$ KeOps, **x30 speed-up** vs. PyTorch and TF.
  $\longrightarrow$ Useful in a wide range of settings.

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

- Going forward, we must develop **topology-aware**, **data-driven**, efficient yet **robust** shape models.
  This is challenging, but we finally have **the right tools for the job**.

Genuine team work

Alain Trouvé

Thibault Séjourné

F.-X. Vialard

Gabriel Peyré

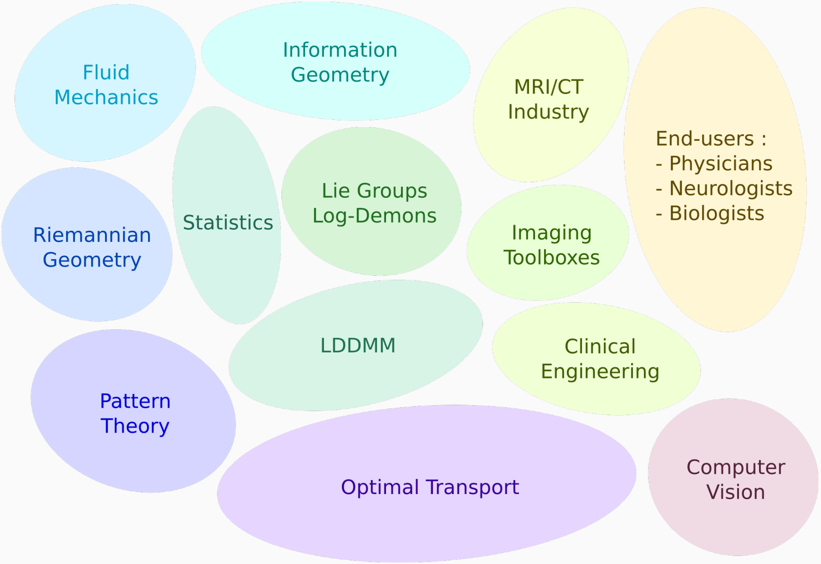
Benjamin Charlier

Joan Glaunès

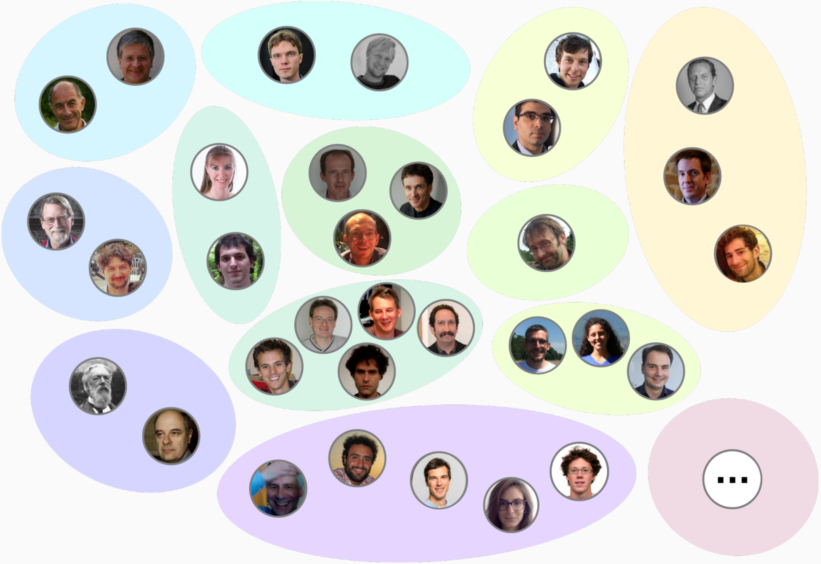
Pierre Roussillon

Pietro Gori

Promoting cross-field interactions



Promoting cross-field interactions



Reaching out to students and engineers is a priority

Online documentation:

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

PhD thesis, written as an introduction to the field:

Geometric data analysis, beyond convolutions

`www.jeanfeydy.com/geometric_data_analysis.pdf`

Thank you for your attention.

Any questions?

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.

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.
- Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard.
  Unbalanced optimal transport: Dynamic and kantorovich formulations.
  Journal of Functional Analysis, 274(11):3090–3123, 2018.
- 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-c`
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.

References v

Olivier Ecabert, Jochen Peters, Matthew J Walker, Thomas Ivanc, Cristian Lorenz, Jens von Berg, Jonathan Lessick, Mani Vembar, and Jürgen Weese.
Segmentation of the heart and great vessels in CT images using a model-based adaptation framework.
Medical image analysis, 15(6):863–876, 2011.

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.

Irene Kaltenmark, Benjamin Charlier, and Nicolas Charon.
A general framework for curve and surface comparison and registration with oriented varifolds.
In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

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.

Ptrump16.
Irm picture.
`https://commons.wikimedia.org/w/index.php?curid=64157788`, 2019.
CC BY-SA 4.0.

References x

Olaf Ronneberger, Philipp Fischer, and Thomas Brox.
U-net: Convolutional networks for biomedical image segmentation.
In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

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