Analyse de données géométriques, au-delà des convolutions

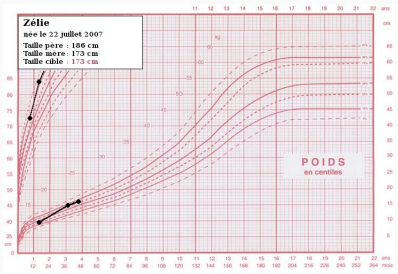
Jean Feydy,
sous la direction d'Alain Trouvé et Michael Bronstein.

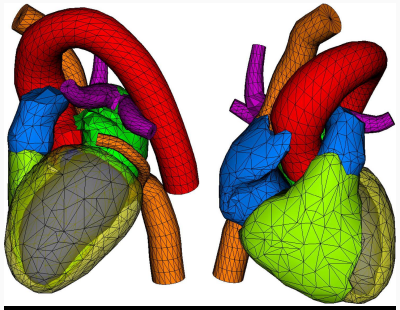
Centre de Recherche des Cordeliers, en ligne — 5 Octobre 2020.

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

En collaboration avec B. Charlier, J. Glaunès (fondations numériques),
T. Séjourné, F.-X. Vialard, G. Peyré (transport optimal),
P. Roussillon, P. Gori (applications en neuro-anatomie).

Traiter une image médicale [Ptr19, EPW^{+}11]





Information pertinente

Statistiques

Description de haut niveau

Anatomie computationnelle

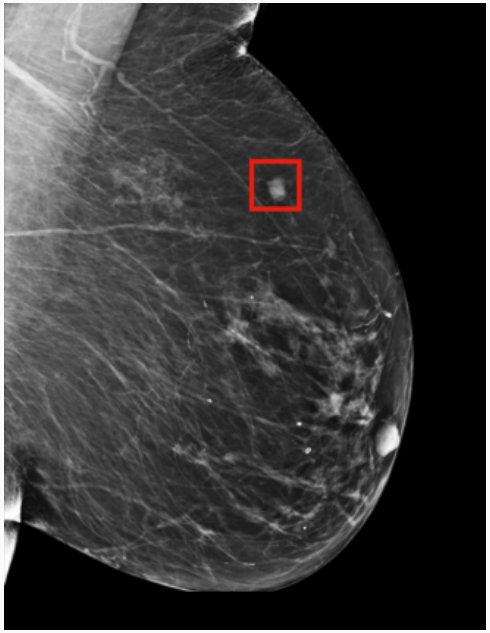
Image brute

Traitement du signal

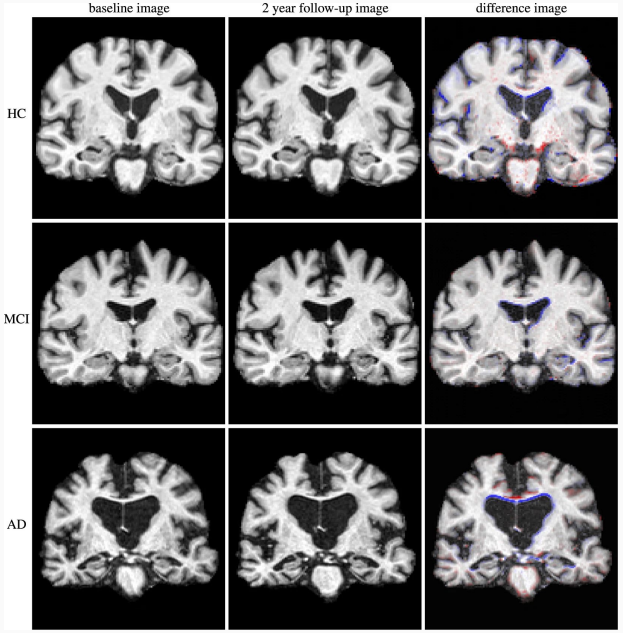
Senseurs

L’anatomie computationnelle [CSG19, LSG+18, CMN14]

Trois grands problèmes :



Détecter un objet



Analyser une variation

Recaler un modèle

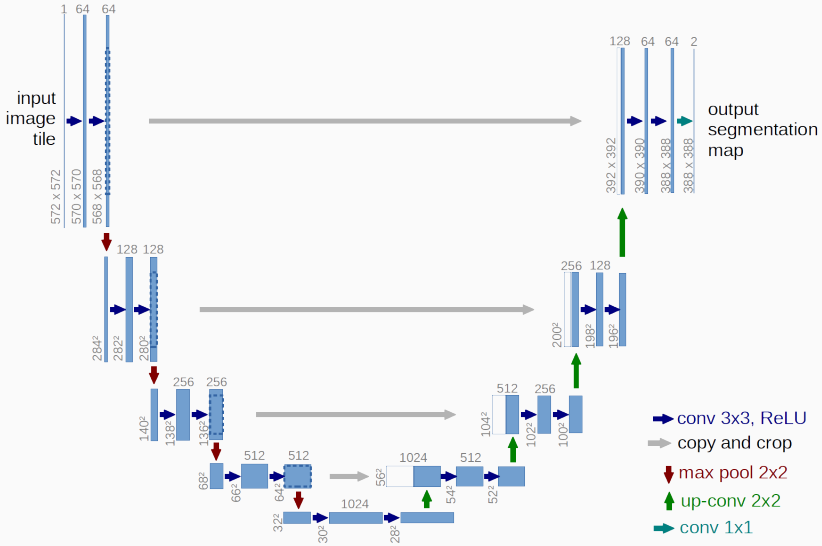
2010–2020 : la révolution du deep learning

Architectures en ondelettes/radiomiques + optimisation sur les données

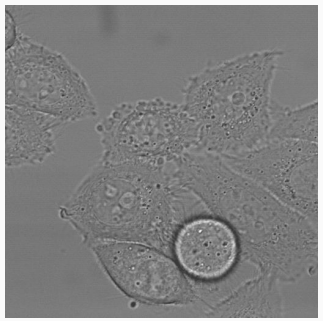
$\Longrightarrow$ Réseaux de neurones à convolutions.

Révolution pour la détection de **motifs** et le traitement de **textures**.

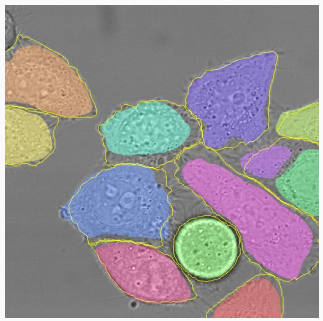
Segmentation avec un U-net [RFB15] :



Architecture



Entrée



Sortie

Les problèmes géométriques gagnent en pertinence

Questions **géométriques** sur des formes segmentées :

- Ce **cœur** bat-il correctement ?
- Comment reconstruire cette **machoire** ?
- Ce **cerveau** a-t-il rapetissé depuis l'année dernière ?
- Peut-on lier ces changements anatomiques à d'**autres signaux** ?

Les problèmes géométriques gagnent en pertinence

Questions **géométriques** sur des formes segmentées :

- Ce **cœur** bat-il correctement ?
- Comment reconstruire cette **machoire** ?
- Ce **cerveau** a-t-il rapetissé depuis l'année dernière ?
- Peut-on lier ces changements anatomiques à d'**autres signaux** ?

Au cours des 30 dernières années, des **méthodes robustes** ont été développées pour répondre à ces questions.

Aujourd'hui, nous voulons les améliorer avec nos **bases de données**.
C'est un problème difficile.

Pour transposer la révolution "ondelettes $\rightarrow$ CNNs" à notre domaine, nous devons **moderniser nos outils numériques**.

Motivations

Analyse de données géométriques, au-delà des convolutions :

- Dédié aux **données géométriques** :
  masques de segmentation, nuages de points, surfaces 3D, etc.
- Met l'accent sur les **méthodes géométriques** :
  K-plus proches voisins, noyaux, transport optimal, etc.
- Fournit de nouvelles **routines numériques**
  pour étendre la boîte à outils standard en sciences des données.

Nous travaillons avec 10^3-10^6 points en dimension 2 à 10,
en nous concentrant sur la géométrie et la rapidité.

Plan de l'exposé

Aujourd'hui, nous parlerons de :

1. **Géométrie rapide** avec des matrices symboliques.
2. **Transport optimal** à grande échelle.
3. Applications et **références**.

Faire de la géométrie avec des matrices symboliques.

Benjamin Charlier

Joan Glaunès

Les librairies de *deep learning* débloquent l'utilisation des GPUs

TensorFlow et PyTorch sont des librairies qui combinent :

+ Une **interface Python** de haut niveau.
+ Des routines pour processeurs et **cartes graphiques**.
+ Un moteur de **différenciation automatique**.
+ Bon support pour l'imagerie (convolutions) et l'algèbre linéaire.

$\Longrightarrow$ **Outils de référence** en machine learning,
devenus indispensables à la recherche en imagerie,
souvent utilisés commes des **boîtes noires** un peu magiques.

Les méthodes efficaces reposent toujours sur des fondations en C++

Implémentations en **C++/CUDA** avec une **interface Python** pour :

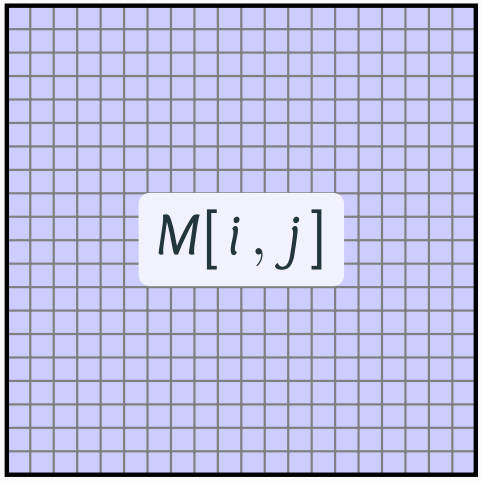
- L'algèbre linéaire (cuBLAS).
- Les convolutions (cuDNN).
- Les transformées de Fourier (cuFFT) et en ondelettes (Kymatio).

Les méthodes géométriques ne bénéficient pas d'un tel niveau d'intégration. Les chercheurs doivent :

- Travailler en C++/CUDA – ce qui est peu commode.
- Se reposer sur des **matrices de distances explicites**.

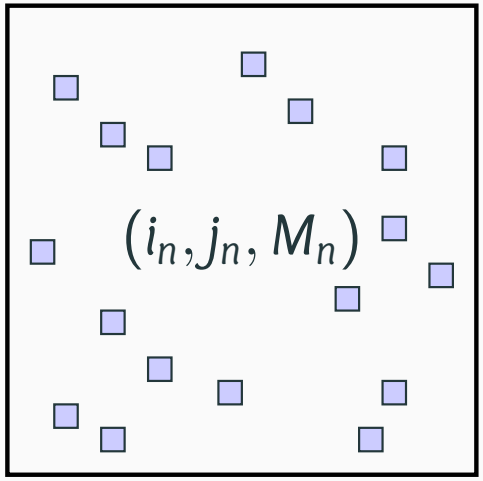
```
RuntimeError: cuda runtime error (2) : out of memory at
/opt/conda/.../THCStorage.cu:66
```

Nous fournissons un support efficace pour les matrices symboliques



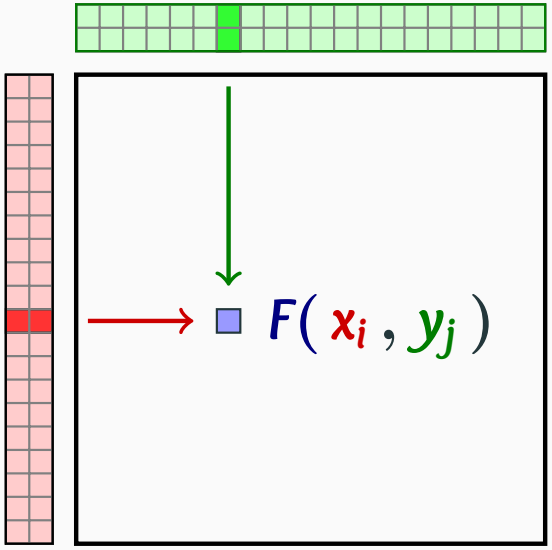
Matrice dense

Coefficients



Matrice creuse

Coordonnées + coeffs



Matrice symbolique

Formule + données

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

KeOps supporte PyTorch, NumPy, Matlab et R

```python
# Nuage de points dans ℝ^50:
import torch
N, D = 10**6, 50
x = torch.rand(N, D).cuda()  # tableau (1M, 50)

# Plus proche voisin pour chaque point:
from pykeops.torch import LazyTensor
x_i = LazyTensor(x.view(N, 1, D))   # x_i = "colonne"
x_j = LazyTensor(x.view(1, N, D))   # x_j = "ligne"
D_ij = ((x_i - x_j)**2).sum(dim=2)  # (N, N) symbolique
indices_i = D_ij.argmin(dim=1)      # -> (N,) dense
```

Performances $\simeq$ codes C++/CUDA de référence (FAISS-GPU).

KeOps combine performances et flexibilité

Travaillons avec des formules arbitraires :

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

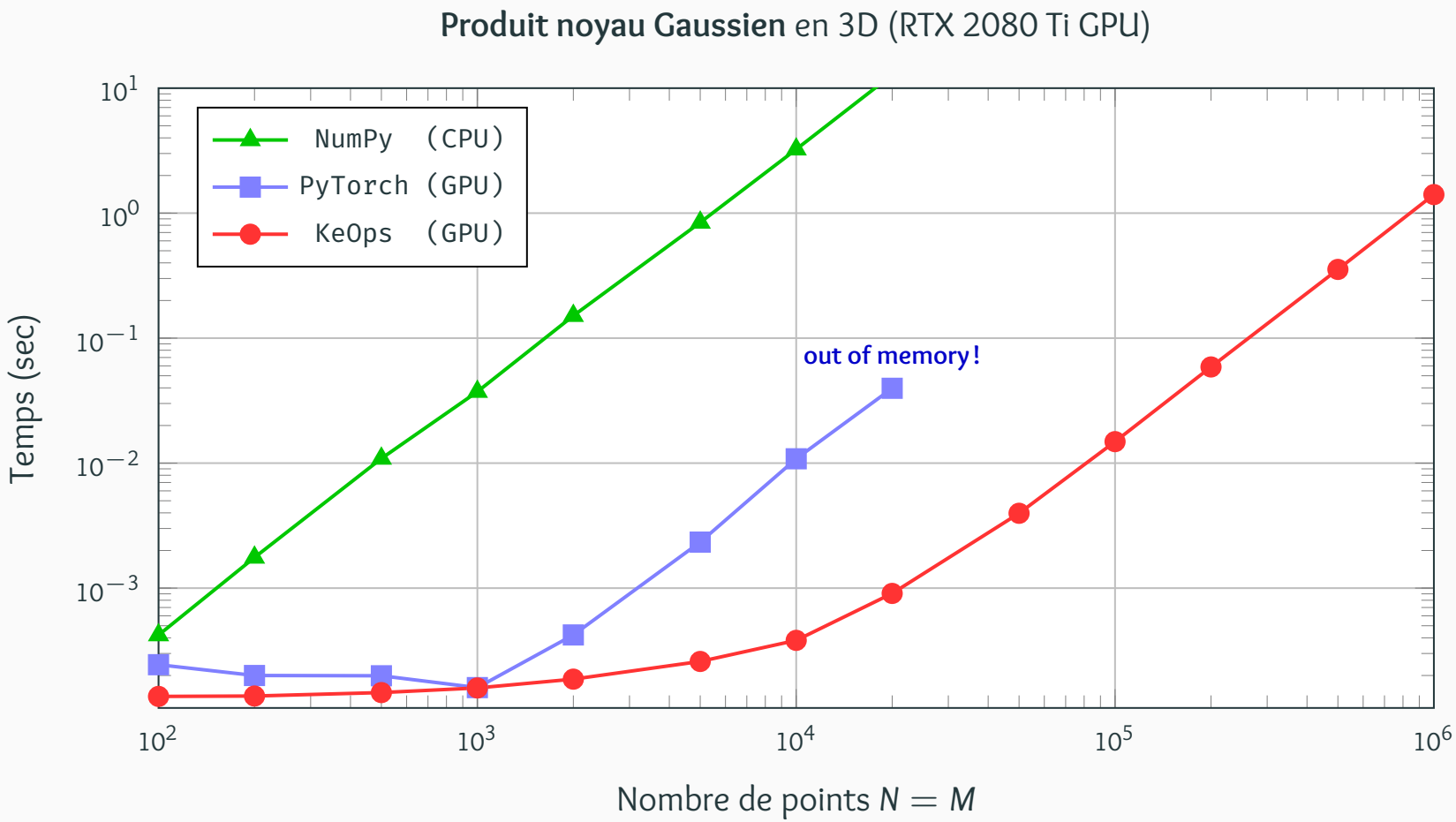
$\Longrightarrow$ accélération $\times 200$ pour UMAP sur les espaces hyperboliques.

KeOps supporte :

- **Réductions :** somme, log-sum-exp, produit matrice-vecteur, *etc.*
- **Opérations :** $+$, $\times$, exp, réseaux de neurones, *etc.*
- **Schémas avancés :** sparsité par blocs, stabilité numérique, *etc.*
- **Différenciation automatique :** intégration avec PyTorch.

Passer à l'échelle sur de grands jeux de données

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



La bibliothèque KeOps

+ **Tout-terrain** : C++, R, Matlab, NumPy *et* PyTorch.
+ **Versatile** : de nombreuses opérations, variables, réductions.
+ **Efficace** : $O(N)$ en mémoire, temps compétitifs.
+ **Puissante** : différenciation automatique, sparsité par blocs, etc.
+ **Transparente** : interface avec **SciPy**, GPytorch, etc.
+ **Complètement documentée** :

`www.kernel-operations.io`

→ Krigeage, splines, **processus gaussiens**, méthodes à **noyaux**.

→ Analyse de formes, deep learning **géométrique**.

(Plus d'illustrations en fin d'exposé !)

Transport optimal computationnel

Thibault Séjourné

F.-X. Vialard

Gabriel Peyré

Nous avons besoin de critères d'erreur robustes

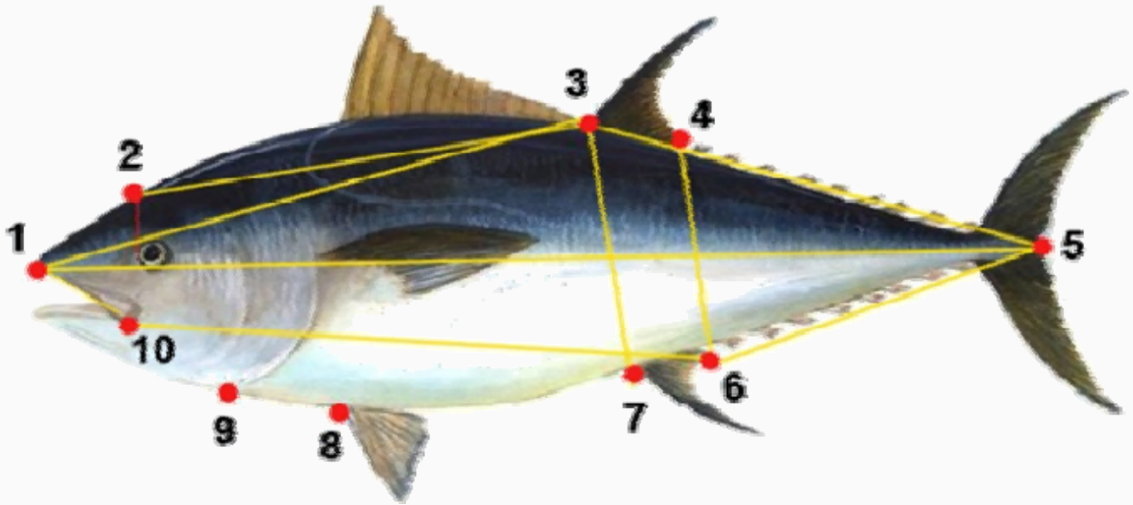
Faire de la géométrie est **plus facile que jamais**.
De nouvelles méthodes peuvent être testées en quelques minutes.

Mais comment **mesurer les réussites** et **les erreurs** ?

$\Longrightarrow$ Nous devons développer des **critères d'erreur géométriques**
pour calculer des distances entre formes.

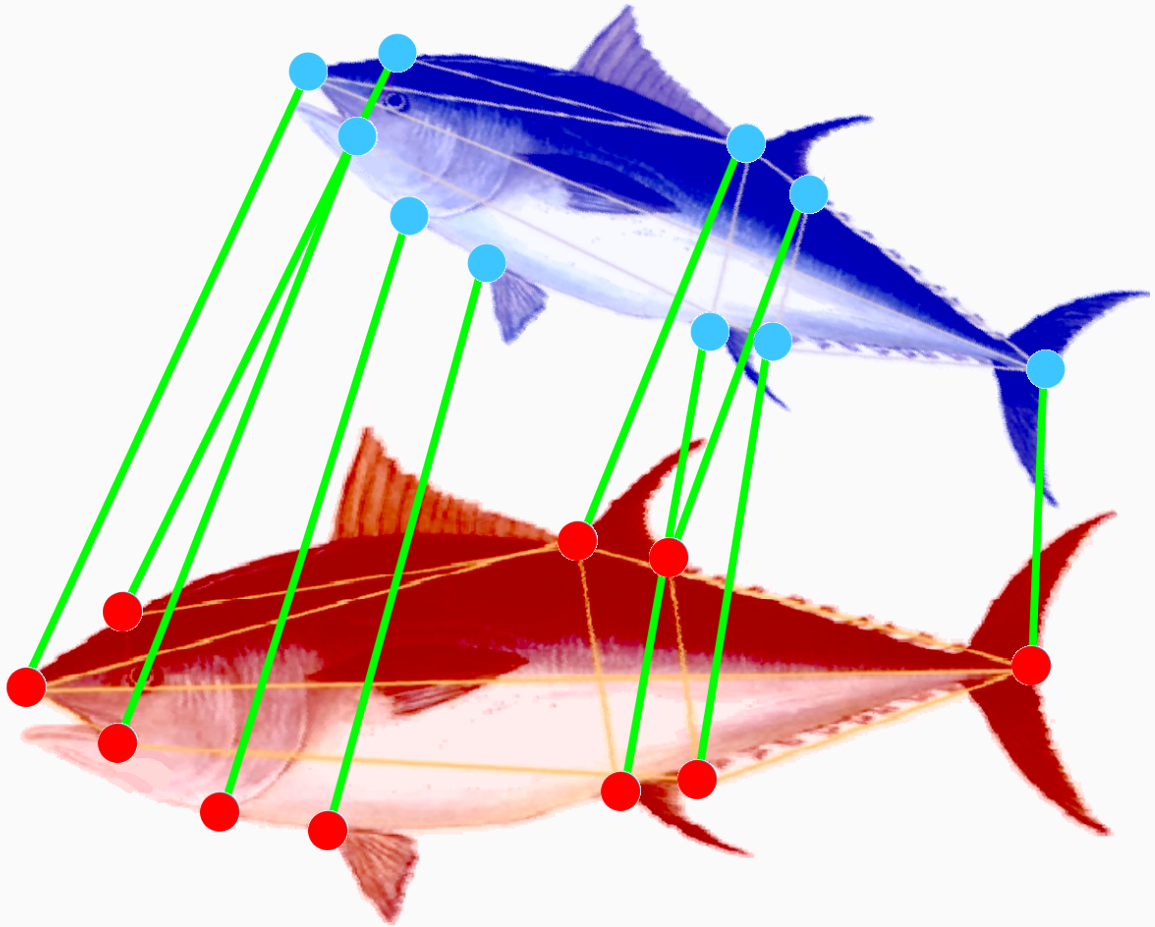
Des gradients de bonne qualité améliorent la **robustesse**
des méthodes de recalage ou d'apprentissage
et nous permettent de nous **concentrer sur nos modèles**.

Avec des points de contrôle, tout est facile...



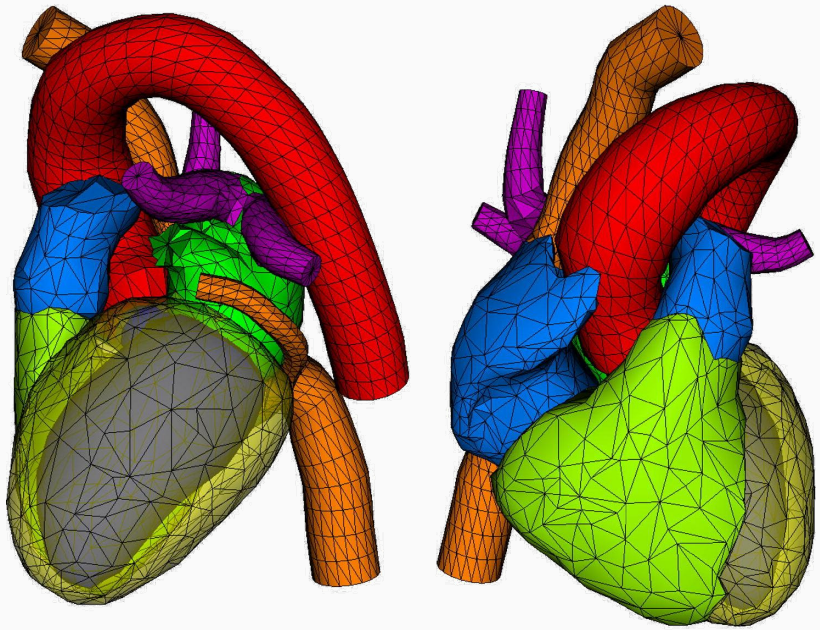
Points de contrôle anatomiques dans *A morphometric approach for the analysis of body shape in bluefin tuna*, Addis et al., 2009.

Avec des points de contrôle, tout est facile...

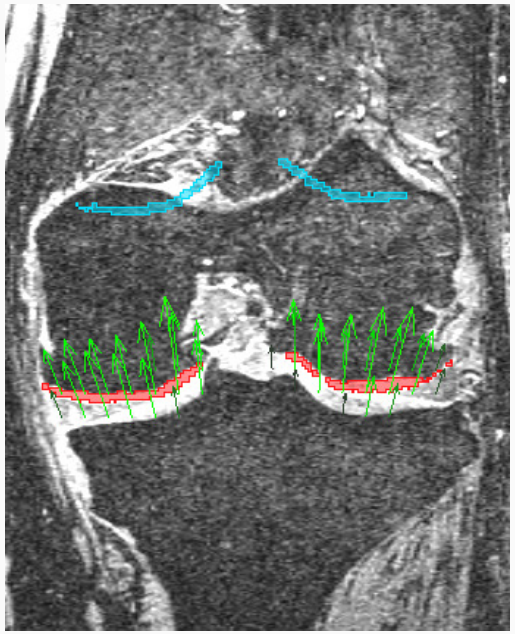


Points de contrôle anatomiques dans *A morphometric approach for the analysis of body shape in bluefin tuna*, Addis et al., 2009.

Malheureusement, les données anatomiques sont souvent *faiblement* étiquetées [EPW+11]



Surfaces

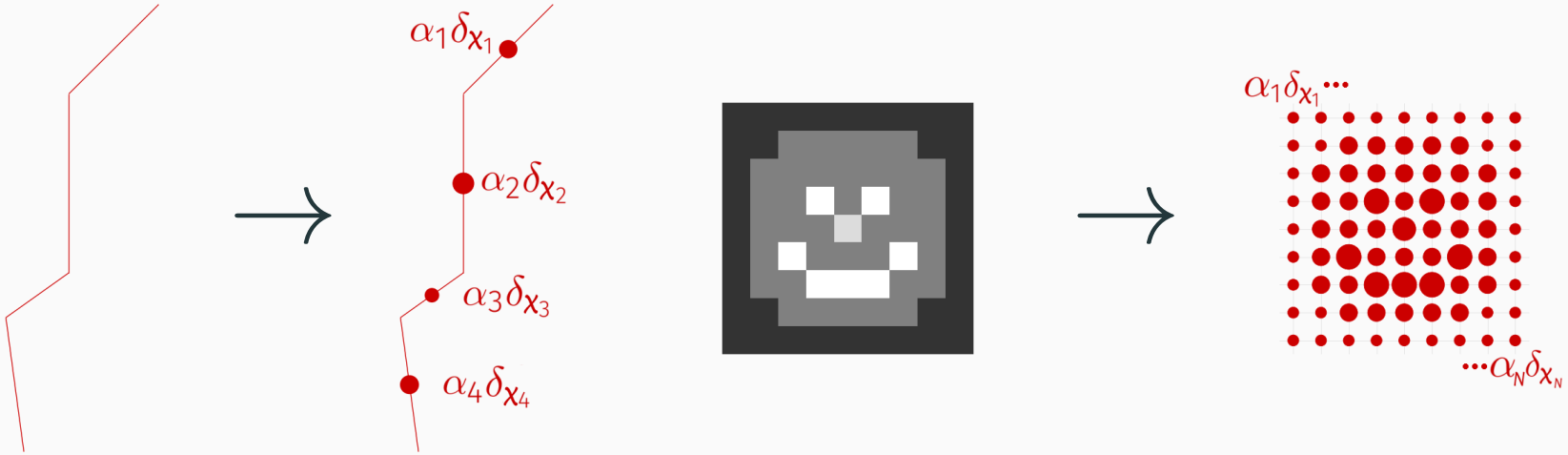


Masques de segmentation

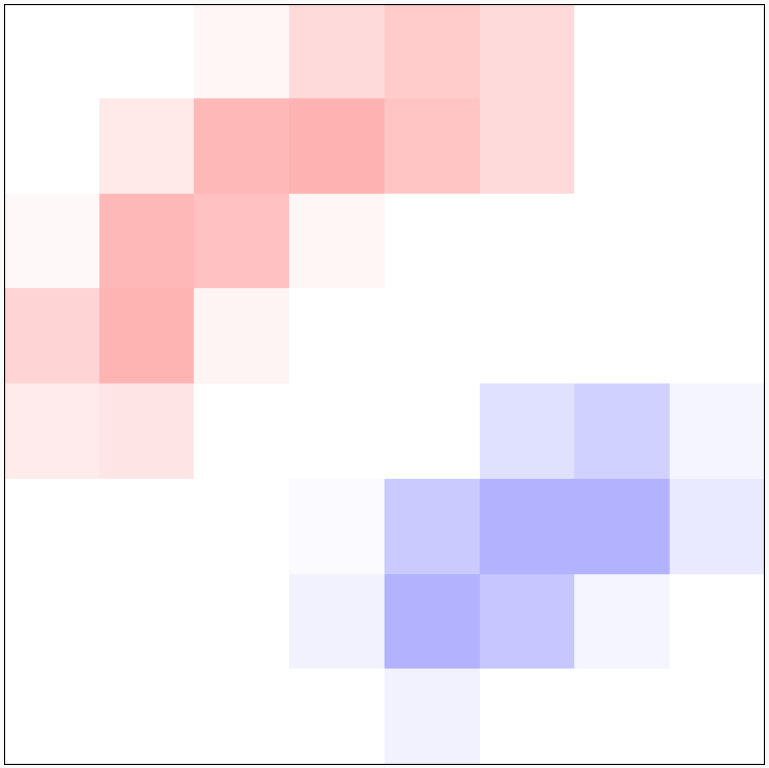
Encodons nos formes comme des mesures

Pour garantir l'invariance aux ré-échantillonnages :

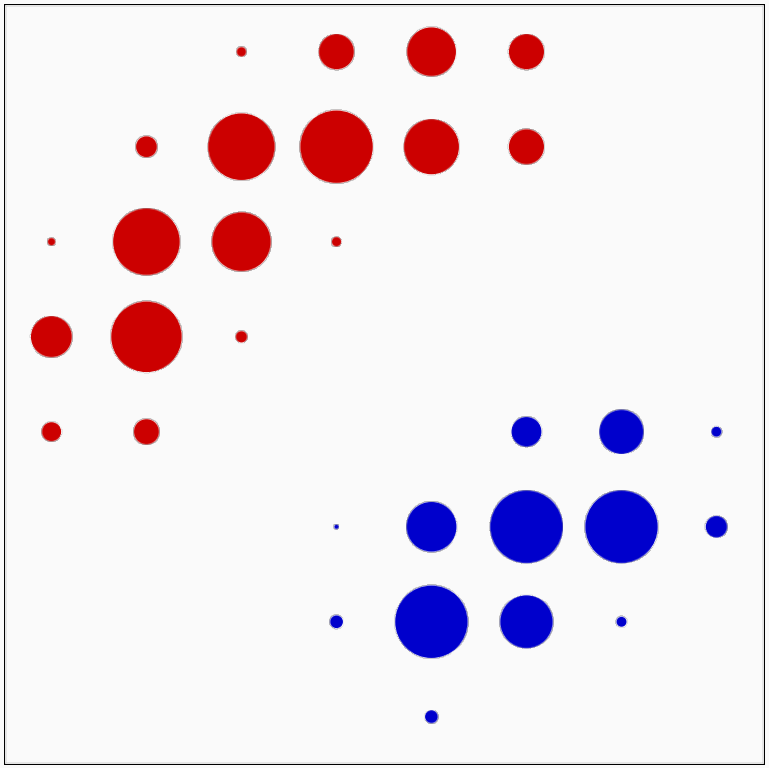
$$A \longrightarrow \alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \qquad B \longrightarrow \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$



Un cadre simple : le recalage de densités

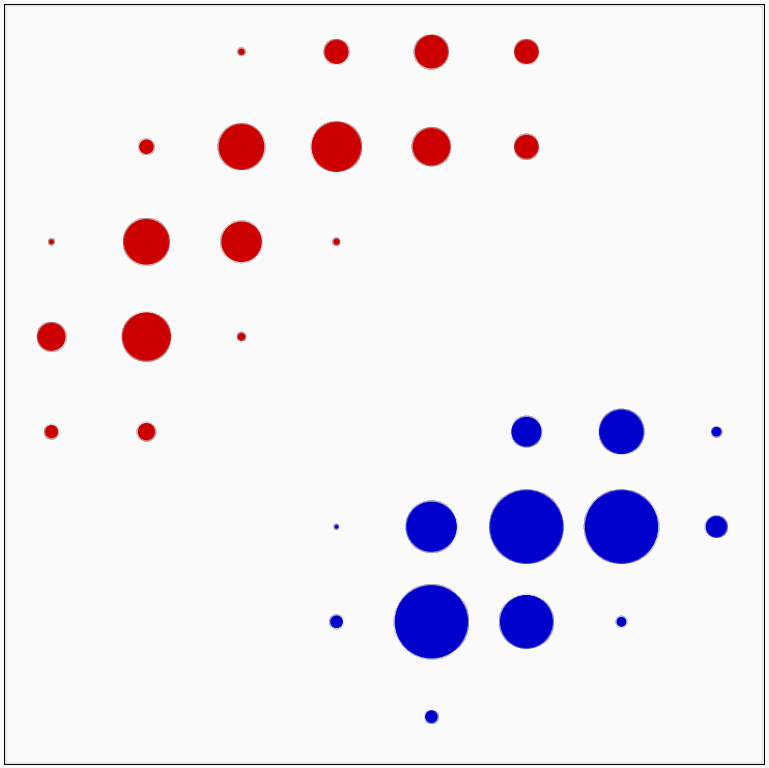


Un cadre simple : le recalage de densités



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

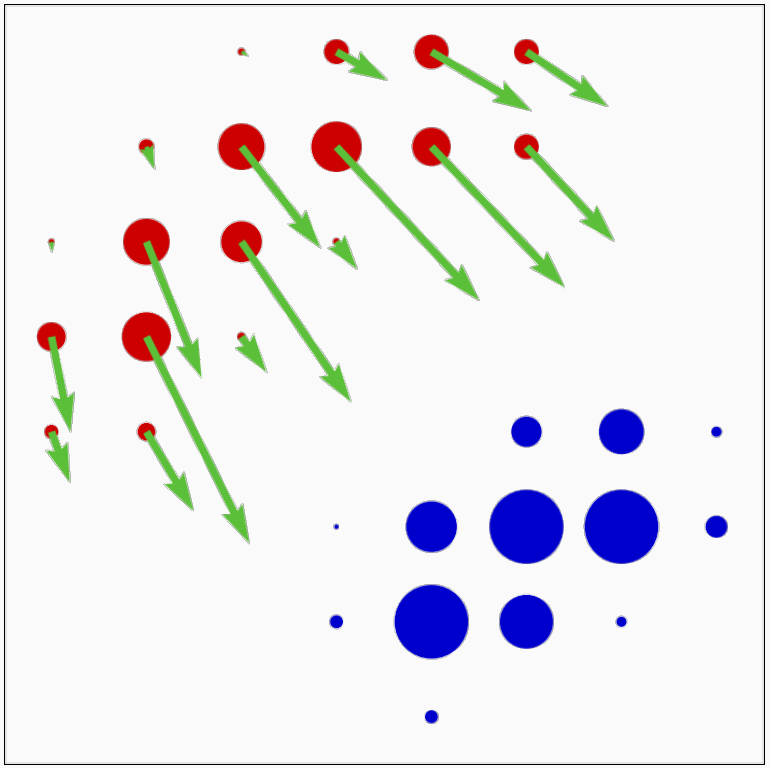
Un cadre simple : le recalage de densités



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

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

Un cadre simple : le recalage de densités

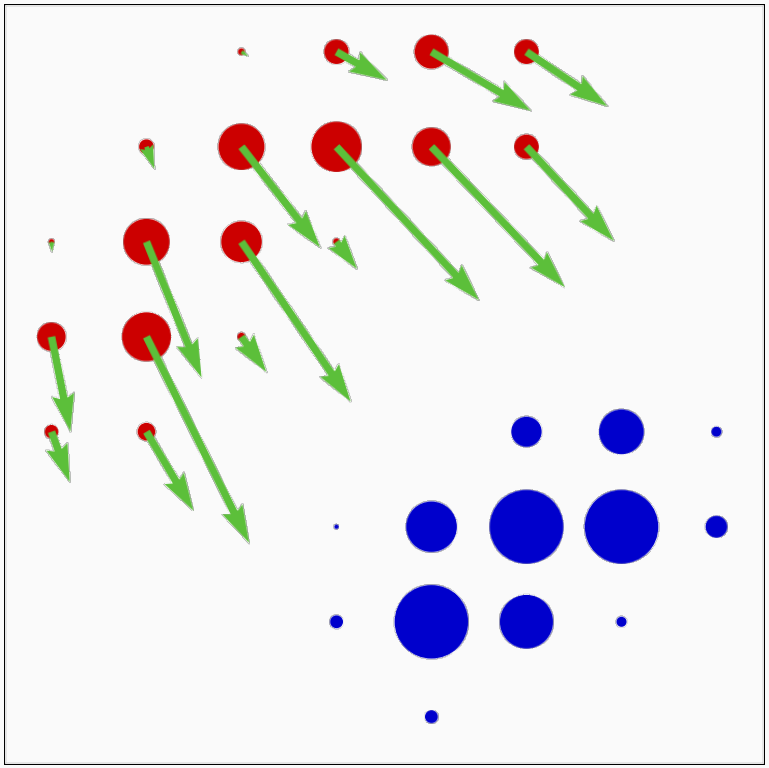


$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i} , \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j} .$$

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

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

Un cadre simple : le recalage de densités



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

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

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

Extensions :

- $\sum_i \alpha_i \neq \sum_j \beta_j$, données aberrantes [CPSV18],
- courbes et surfaces [KCC17],
- poids variables α_i.

La distance de Wasserstein

On veut des **gradients propres**, sans artefacts.

La distance de Wasserstein

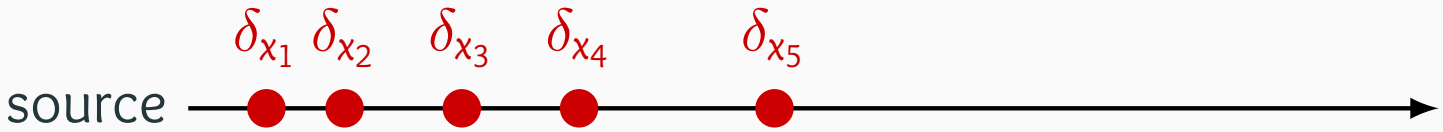
On veut des **gradients propres**, sans artefacts.

Un exemple jouet en dimension 1 :

La distance de Wasserstein

On veut des **gradients propres**, sans artefacts.

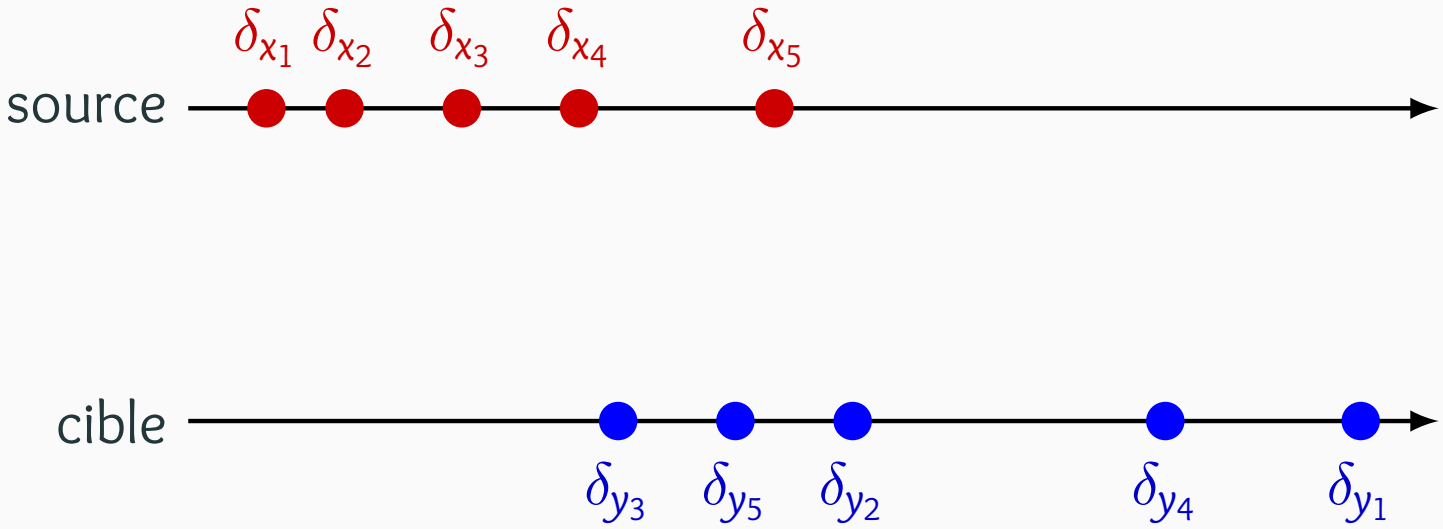
Un exemple jouet en dimension 1 :



La distance de Wasserstein

On veut des **gradients propres**, sans artefacts.

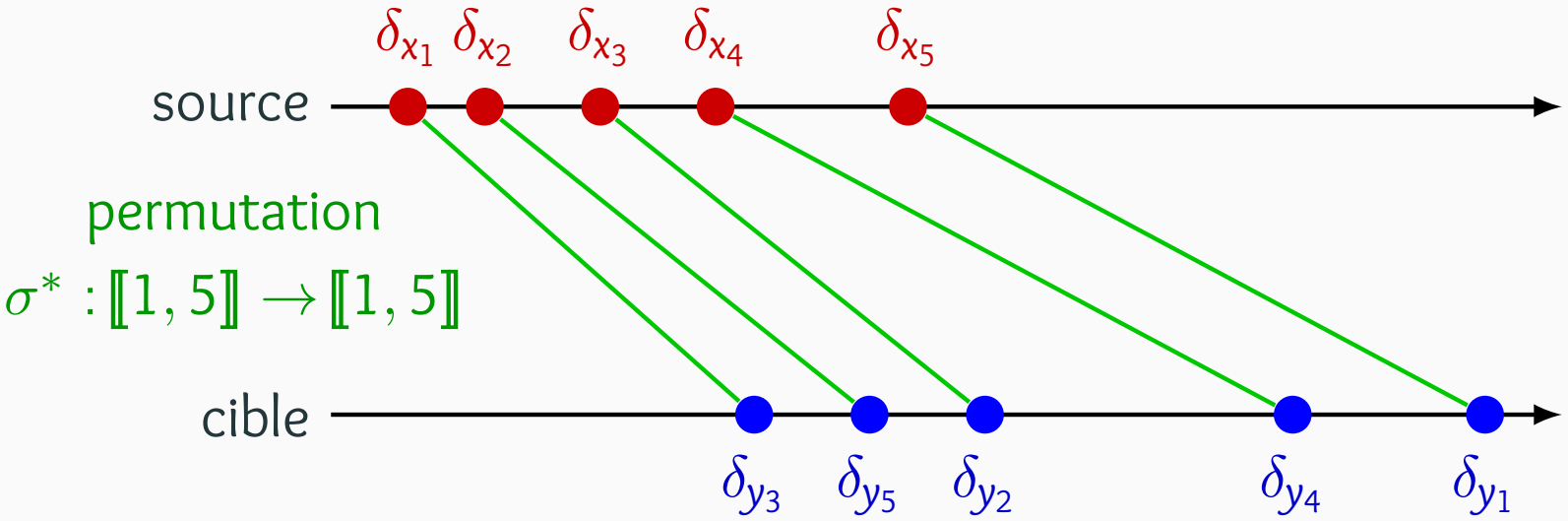
Un exemple jouet en dimension 1 :



La distance de Wasserstein

On veut des **gradients propres**, sans artefacts.

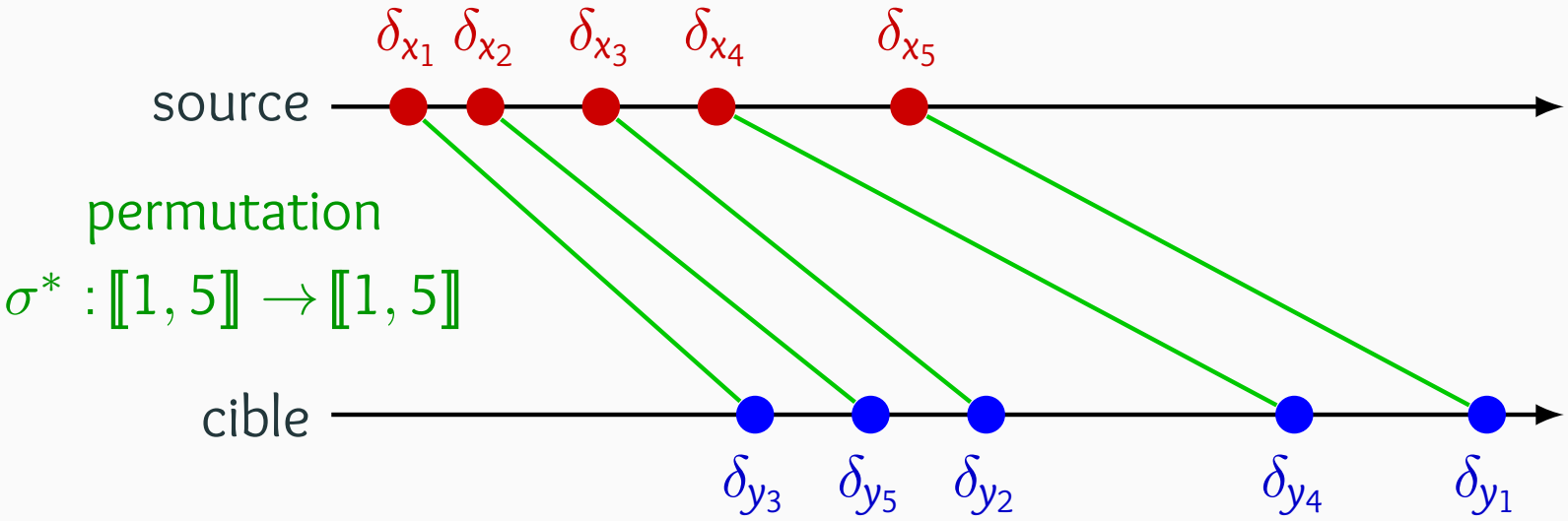
Un exemple jouet en dimension 1 :



La distance de Wasserstein

On veut des **gradients propres**, sans artefacts.

Un exemple jouet en dimension 1 :

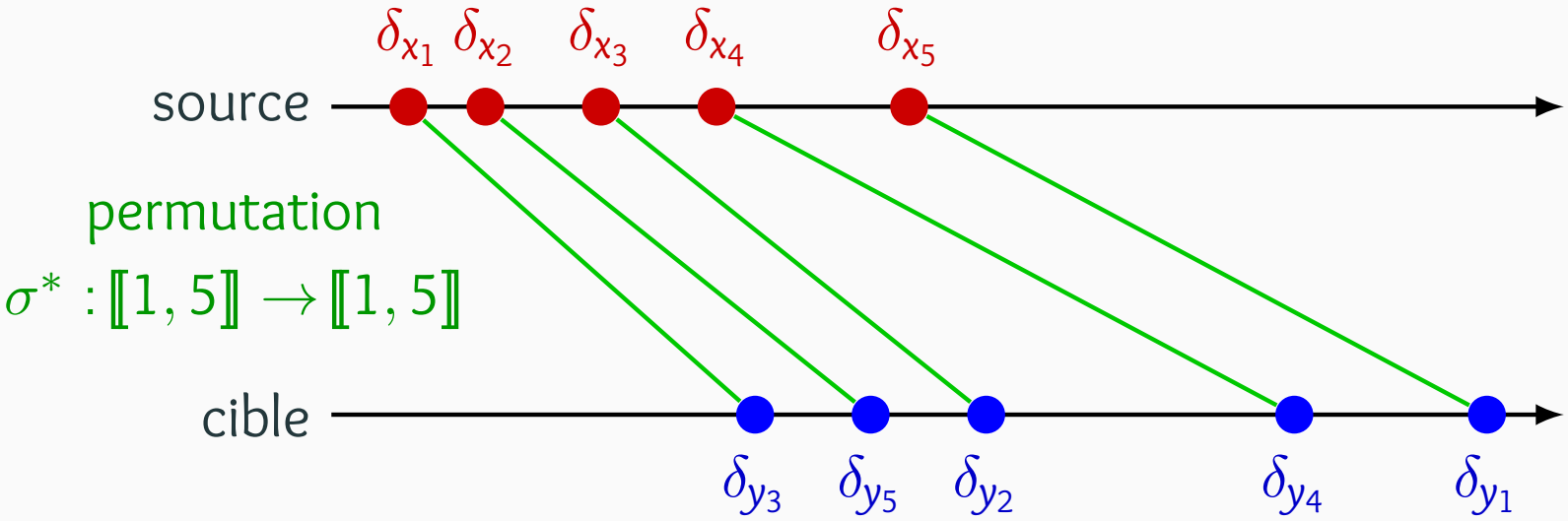


$$\mathrm{OT}(\alpha, \beta) = \frac{1}{2N} \sum_{i=1}^{N} |x_i - y_{\sigma^*(i)}|^2$$

La distance de Wasserstein

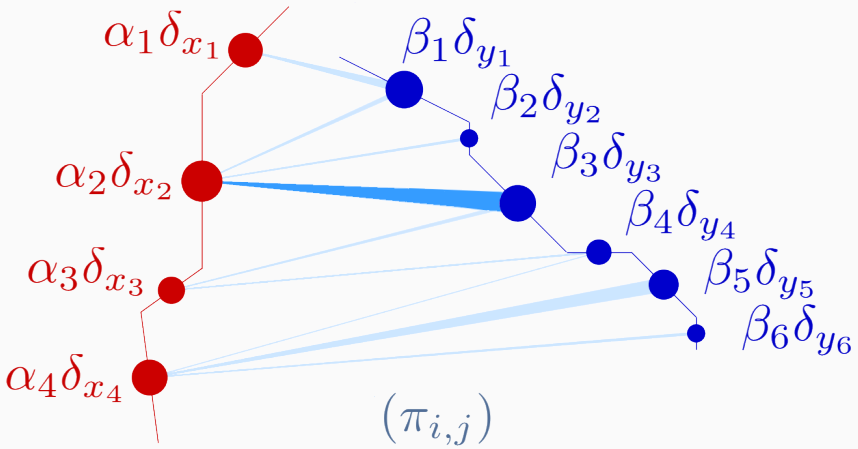
On veut des **gradients propres**, sans artefacts.

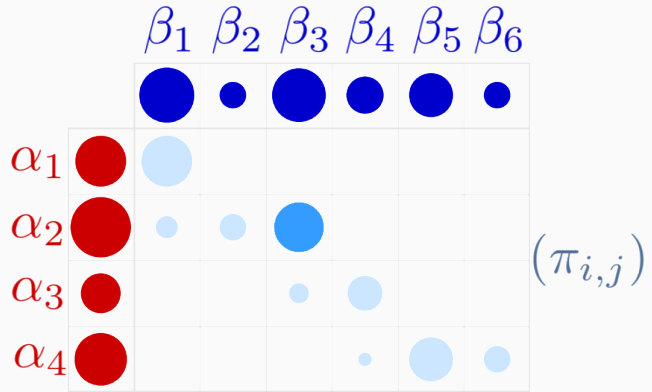
Un exemple jouet en dimension 1 :



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

Le transport optimal généralise le tri aux espaces de dimension D > 1





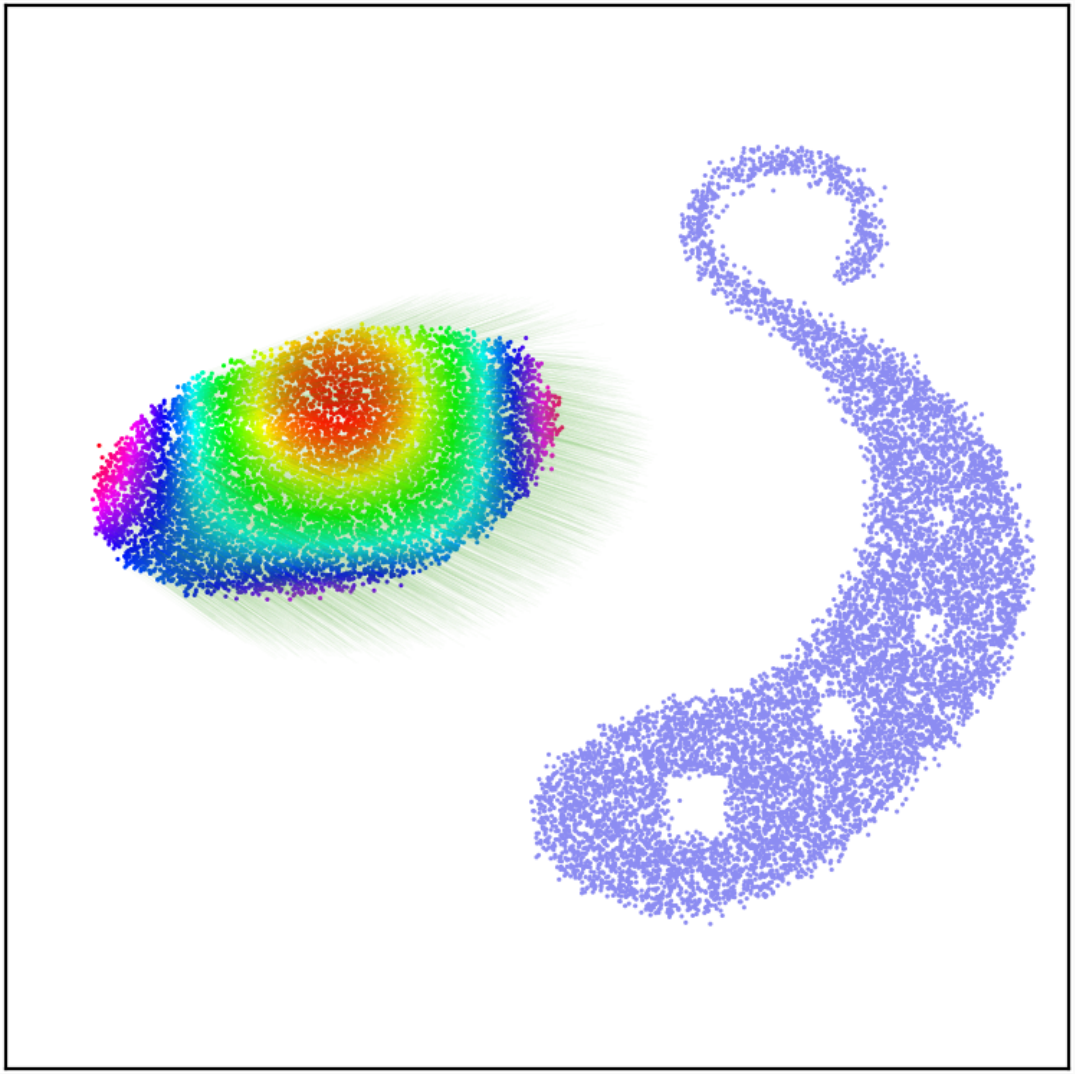
Minimisation sur une matrice N-par-M (plan de transport) π :

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

sous contraintes : $\pi_{i,j} \geqslant 0$,

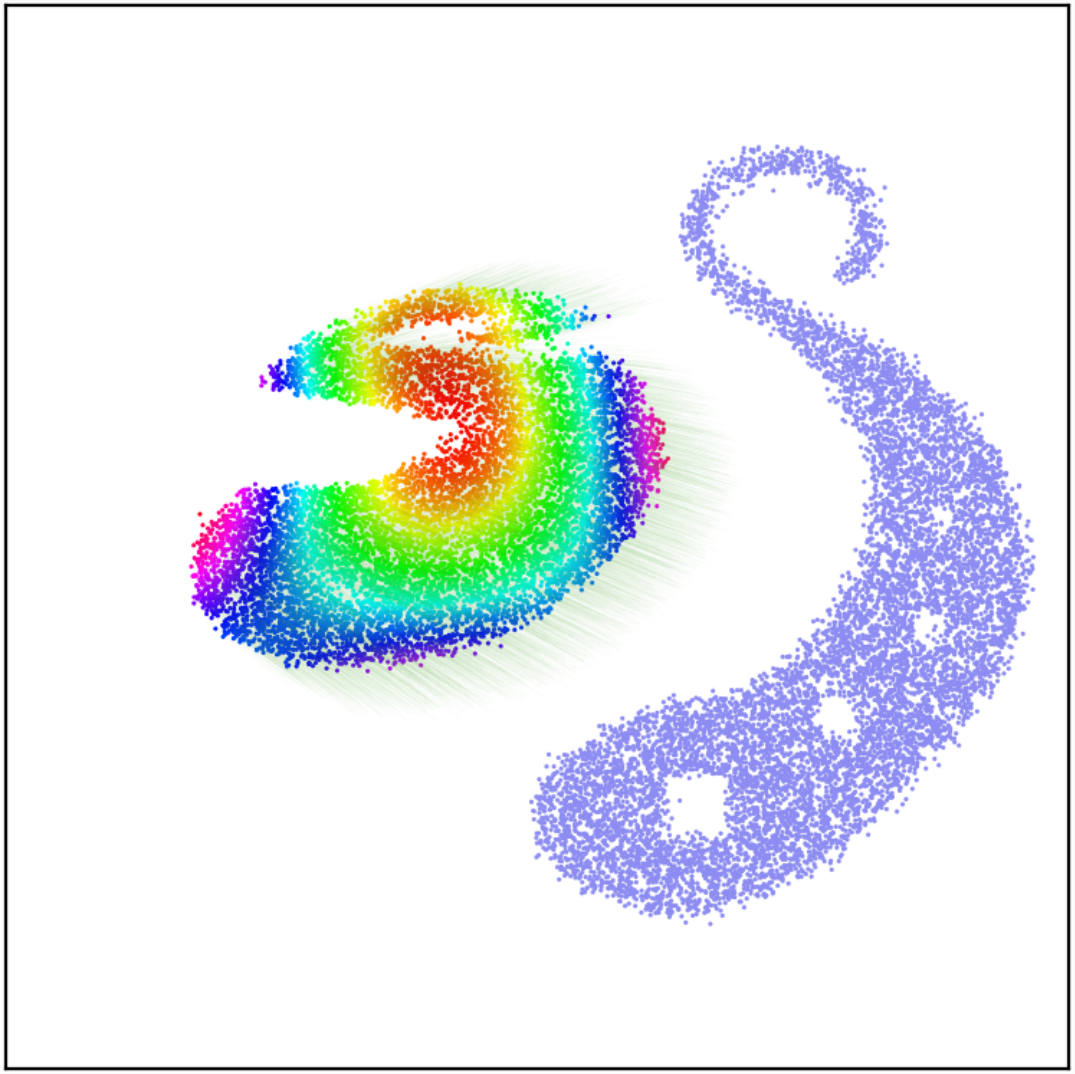
$$\sum_j \pi_{i,j} = \alpha_i, \quad \sum_i \pi_{i,j} = \beta_j.$$

Un recalage jouet : le flot gradient $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



$t = .00$

Un recalage jouet : le flot gradient $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$

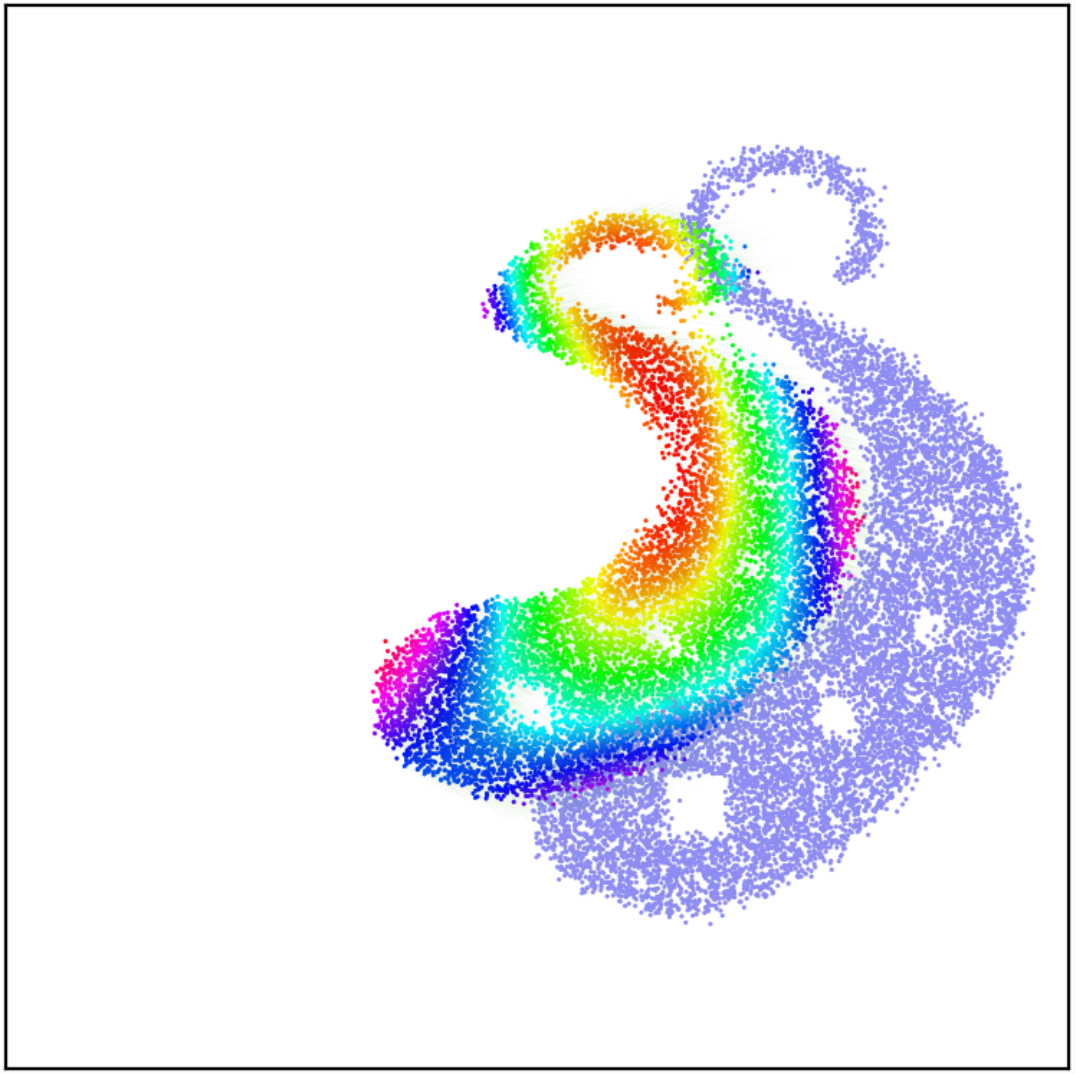


$t = .25$

Un recalage jouet : le flot gradient $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$

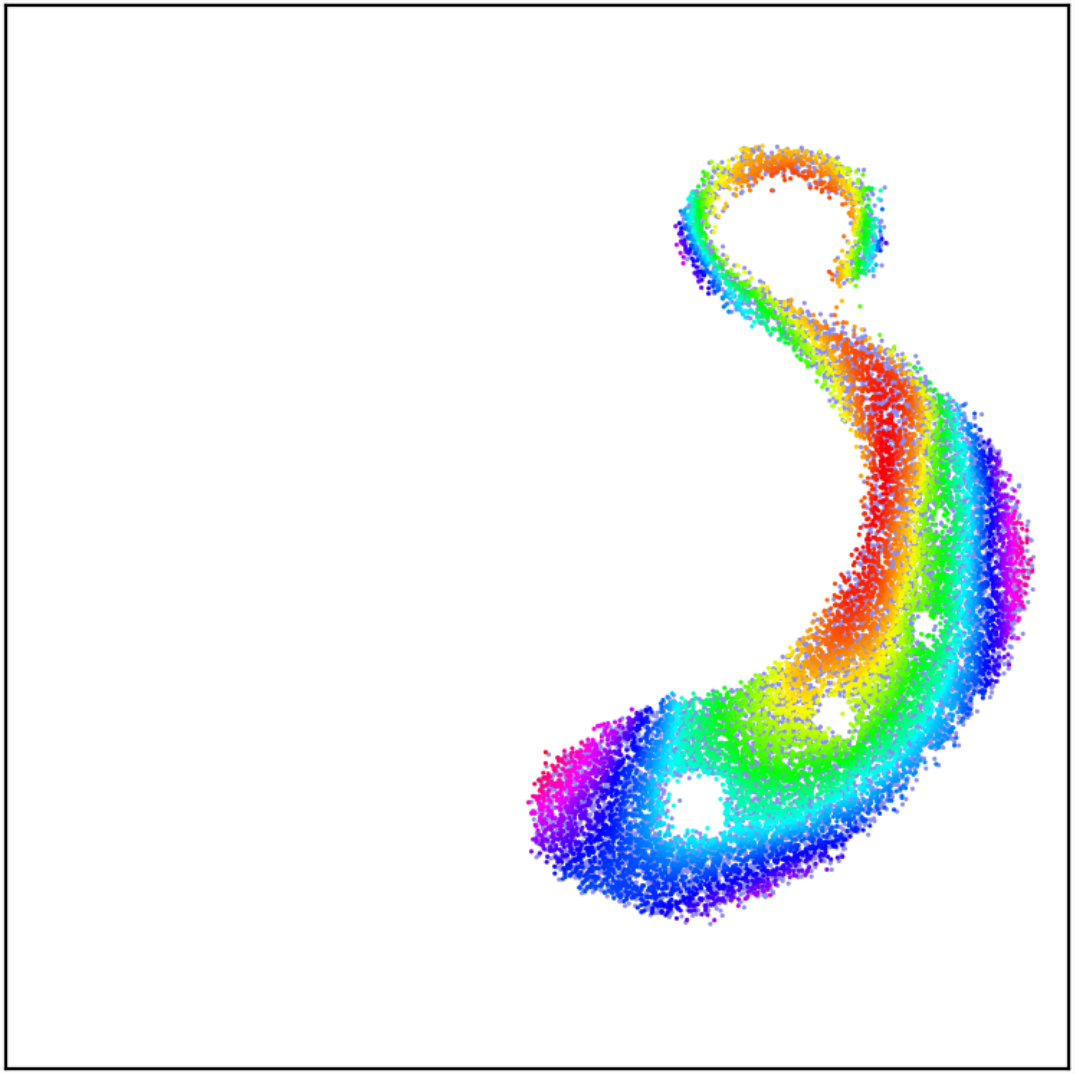
$t = .50$

Un recalage jouet : le flot gradient $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



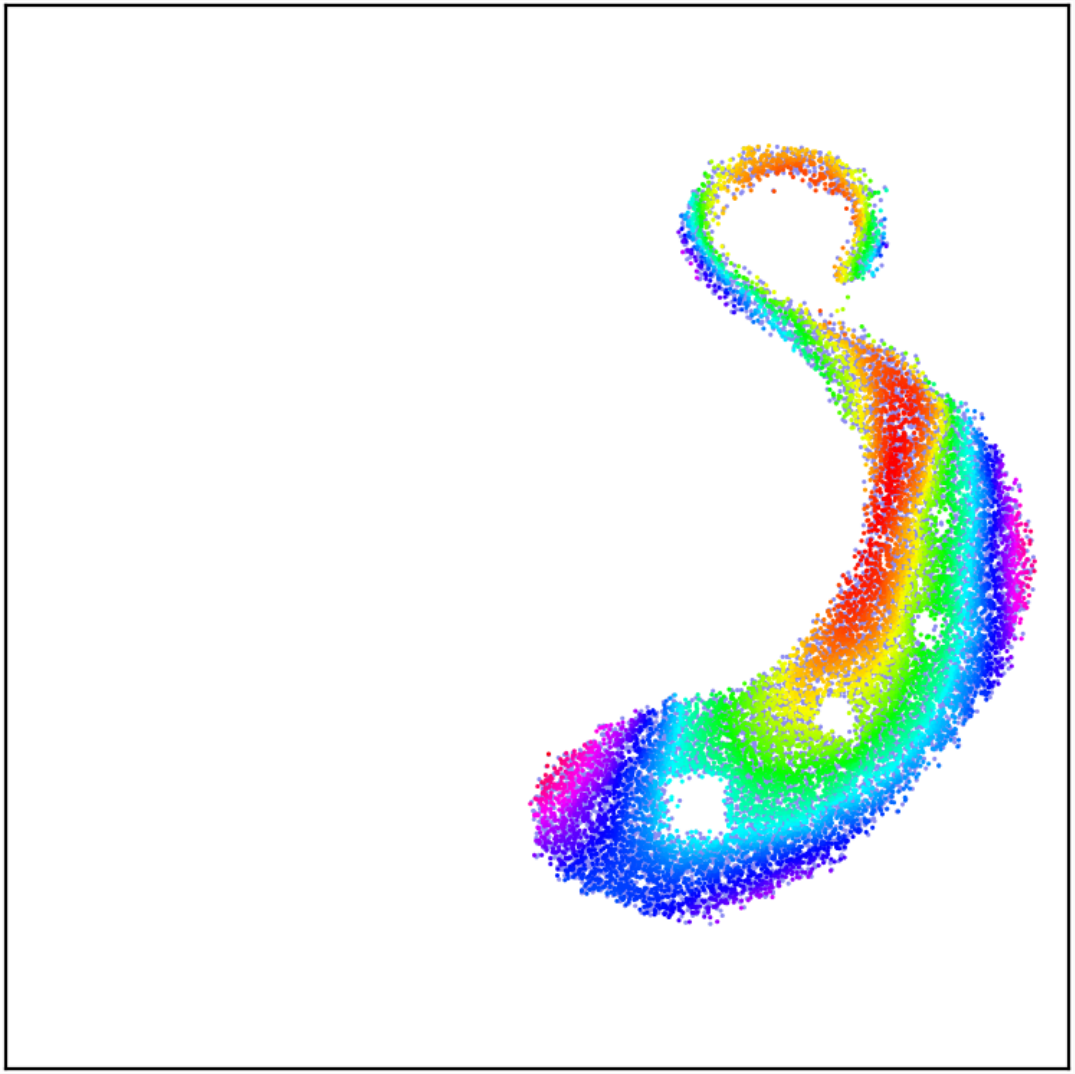
$t = 1.00$

Un recalage jouet : le flot gradient $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



$t = 5.00$

Un recalage jouet : le flot gradient $x_i \leftarrow x_i - \delta t \frac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta)$



$t = 10.00$

Comment résoudre le problème de transport ?

Dates importantes pour le transport optimal discret avec N points :

- [Kan42] : Problème **dual**.
- [Kuh55] : Méthode **hongroise** en $O(N^3)$.
- [Ber79] : Algorithme des **enchères** en $O(N^2)$.
- [KY94] : **SoftAssign** = Sinkhorn + recuit simulé, en $O(N^2)$.
- [GRL+98, CR00] : **Robust Point Matching** = erreur via Sinkhorn.
- [Cut13] : Début de l'**ère des GPUs**.
- [Mér11, Lév15, Sch19] : Solveurs **multi-échelles** en $O(N \log N)$.
- Aujourd'hui : **Méthode de Sinkhorn multi-échelle, sur le GPU**.

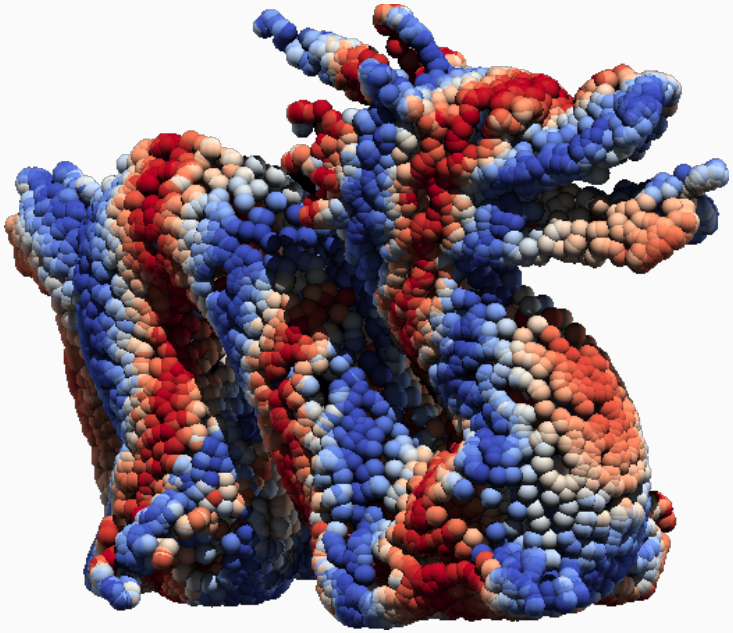
$\Longrightarrow$ Algorithme de **tri rapide** généralisé.

Passage à l'échelle sur des données anatomiques

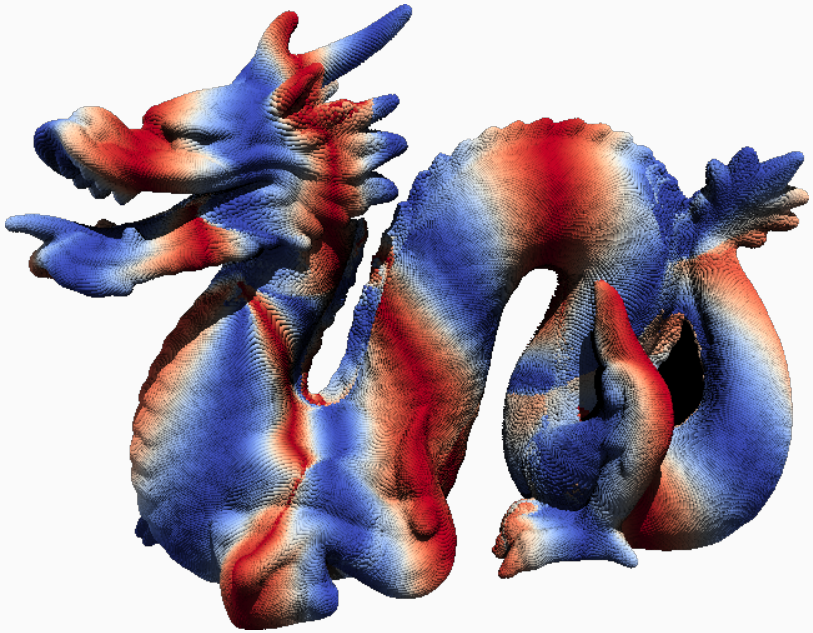
Ces progrès cumulés permettent une accélération $\times 100$ - $\times 1000$:

Sinkhorn GPU $\xrightarrow{\times 10}$ + KeOps $\xrightarrow{\times 10}$ + Recuit $\xrightarrow{\times 10}$ + Multi-échelle

Avec une précision de 1%, sur une machine de gamer :



10k points en 30-50ms



100k points en 100-200ms

Fonctions d'erreur géométriques pour PyTorch

Notre site web : `www.kernel-operations.io/geomloss`

$$\Longrightarrow \texttt{pip install geomloss} \Longleftarrow$$

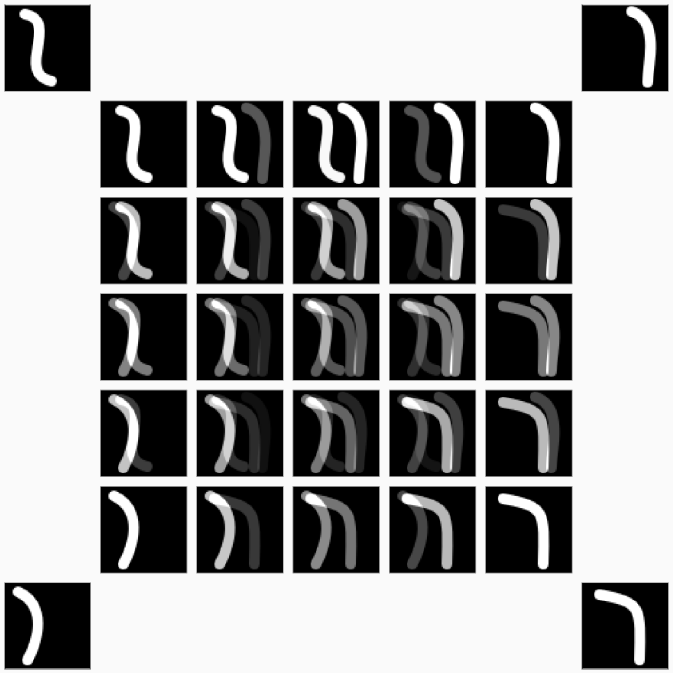
```
# Nuages de points dans [0,1]^3
import torch
x = torch.rand(100000, 3, requires_grad=True).cuda()
y = torch.rand(200000, 3).cuda()

# Coût de transport entre deux mesures discrètes:
from geomloss import SamplesLoss
loss = SamplesLoss(loss="sinkhorn", p=2, blur=.05)

L = loss(x, y)  # Poids constants, par défaut
# GeomLoss supporte l'autodiff, les batches, etc.
g_x, = torch.autograd.grad(L, [x])
```

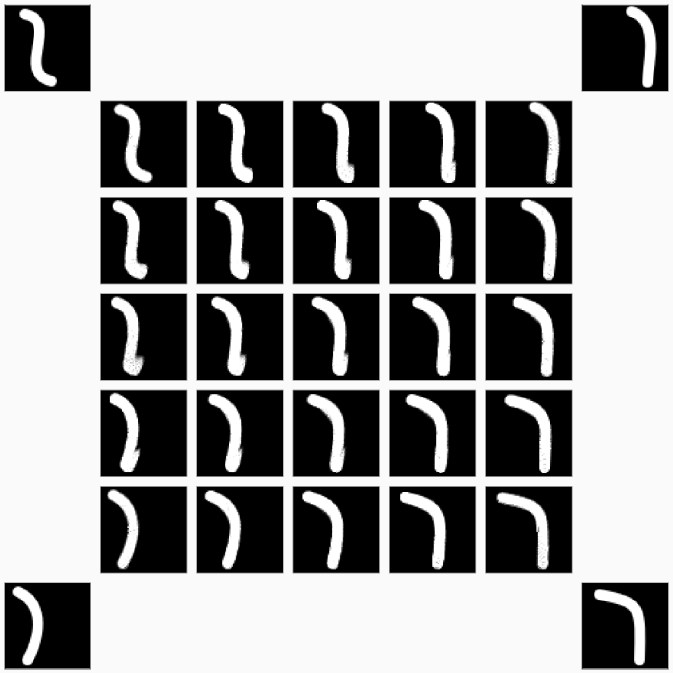
Une interpolation géométrique à petit prix [AC11]

$$\text{Barycentre } \alpha^* = \arg\min_{\alpha} \sum_{i=1}^{N} \lambda_i \, \text{Erreur}(\alpha, \beta_i) .$$



Barycentres linéaires

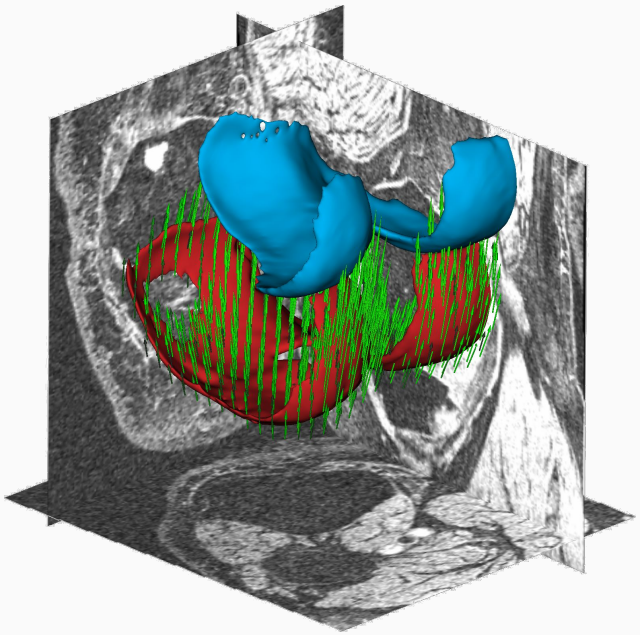
$\text{Erreur}(\alpha, \beta) = \|\alpha - \beta\|_{L^2}^2$



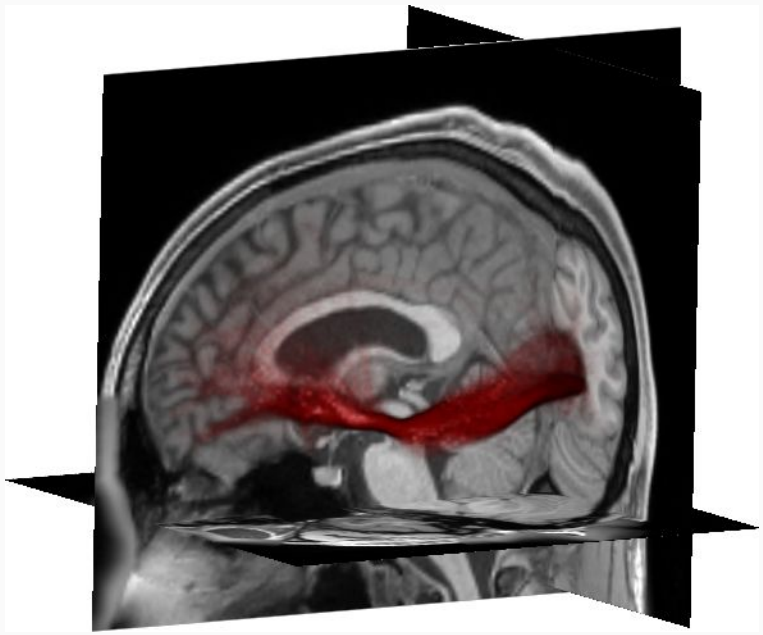
Barycentres de Wasserstein

$\text{Erreur}(\alpha, \beta) = \text{OT}(\alpha, \beta)$

Applications en imagerie médicale

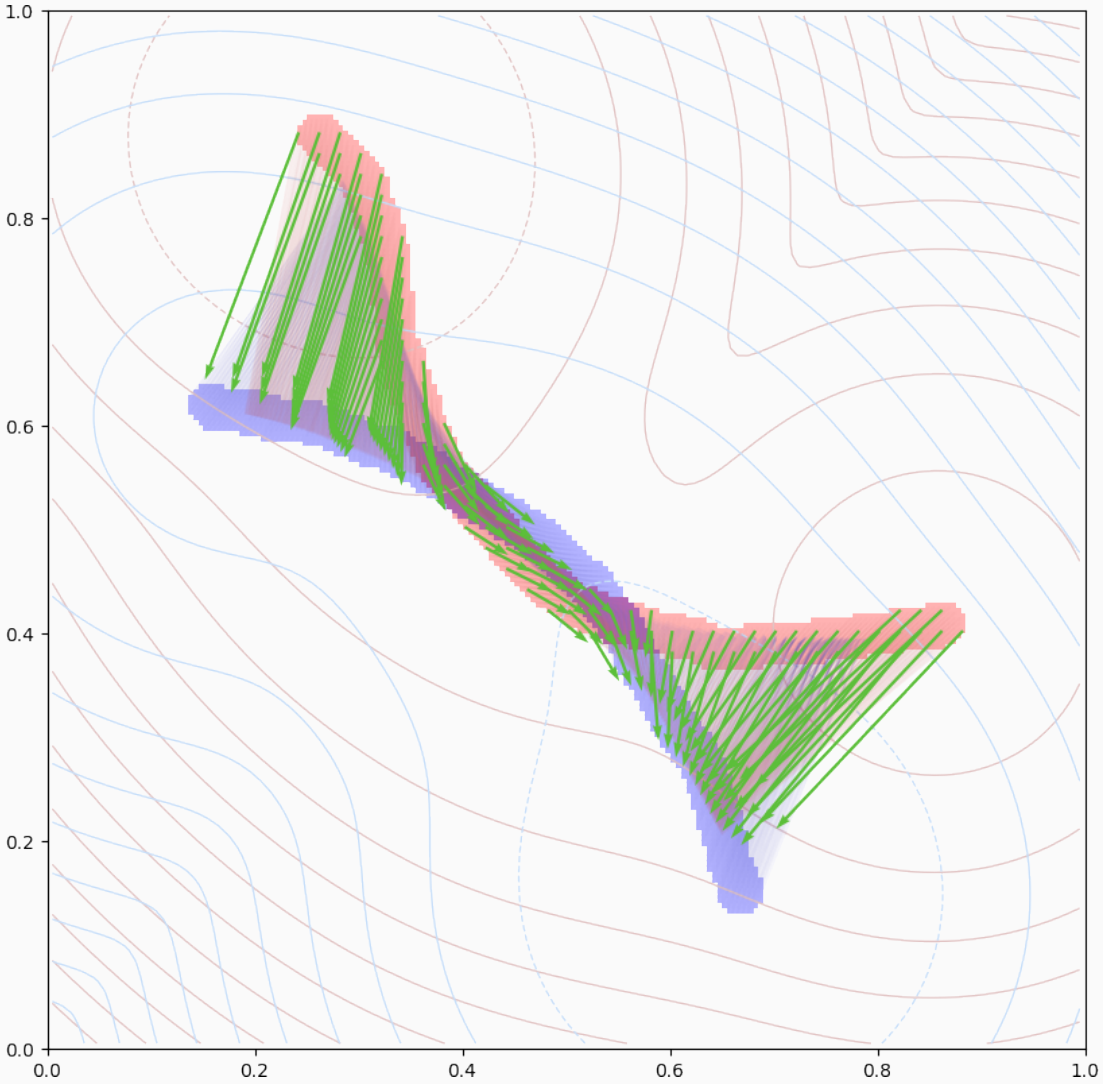


Cartilage du genou.



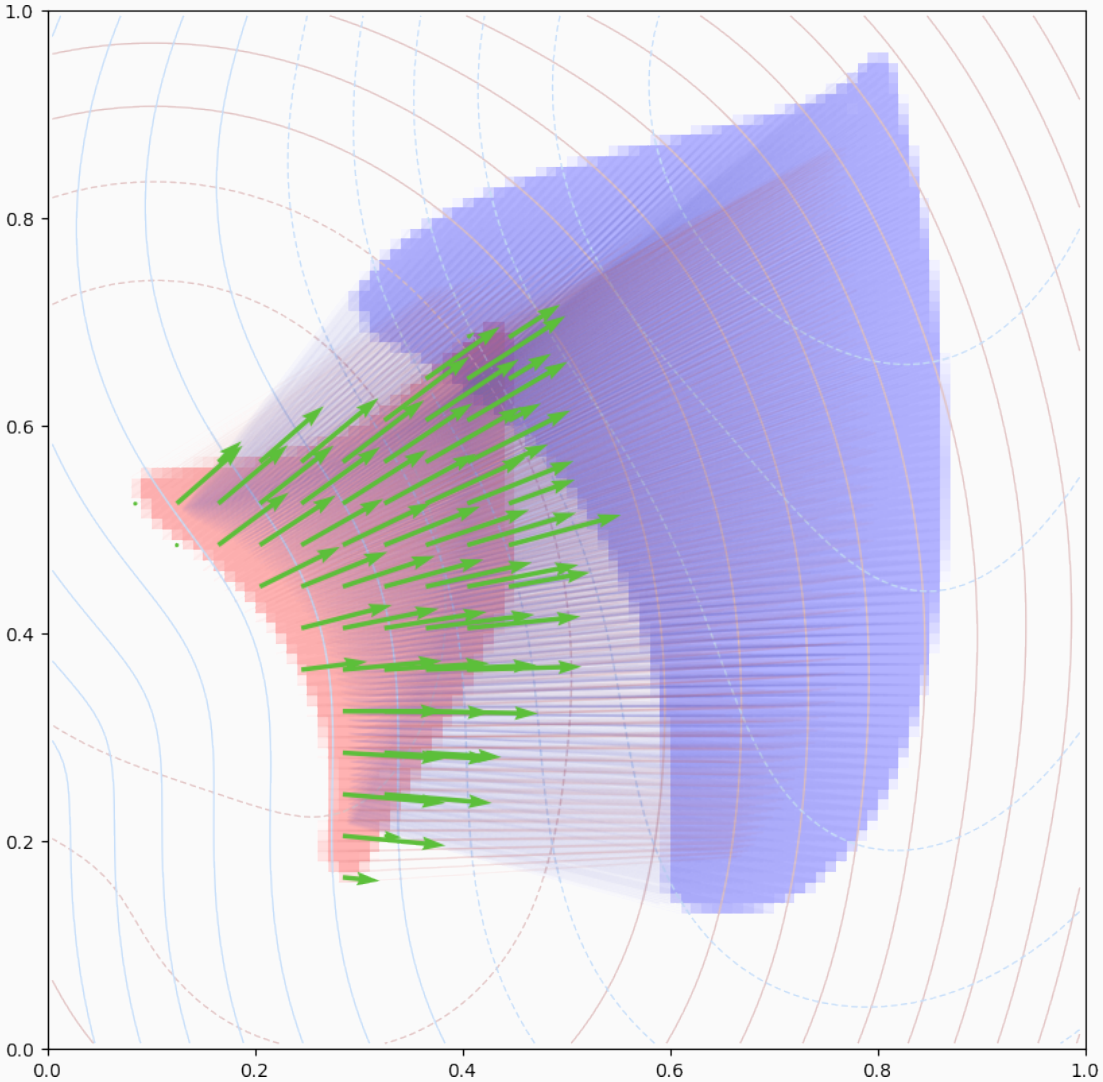
Faisceau de matière blanche.

Une fonction d'erreur globale et géométrique



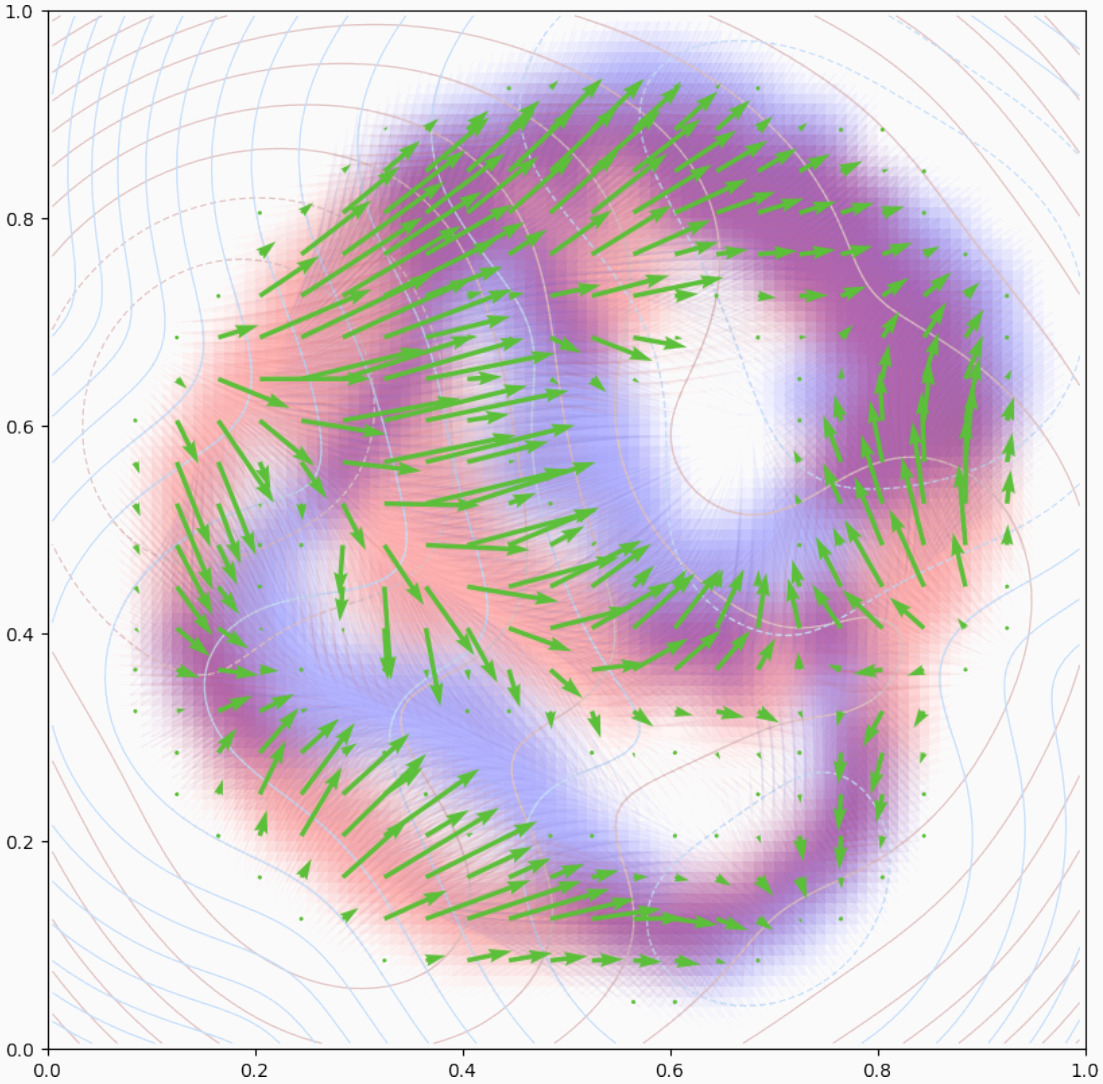
Un gradient de bonne qualité...

Une fonction d'erreur globale et géométrique



Un gradient de bonne qualité...

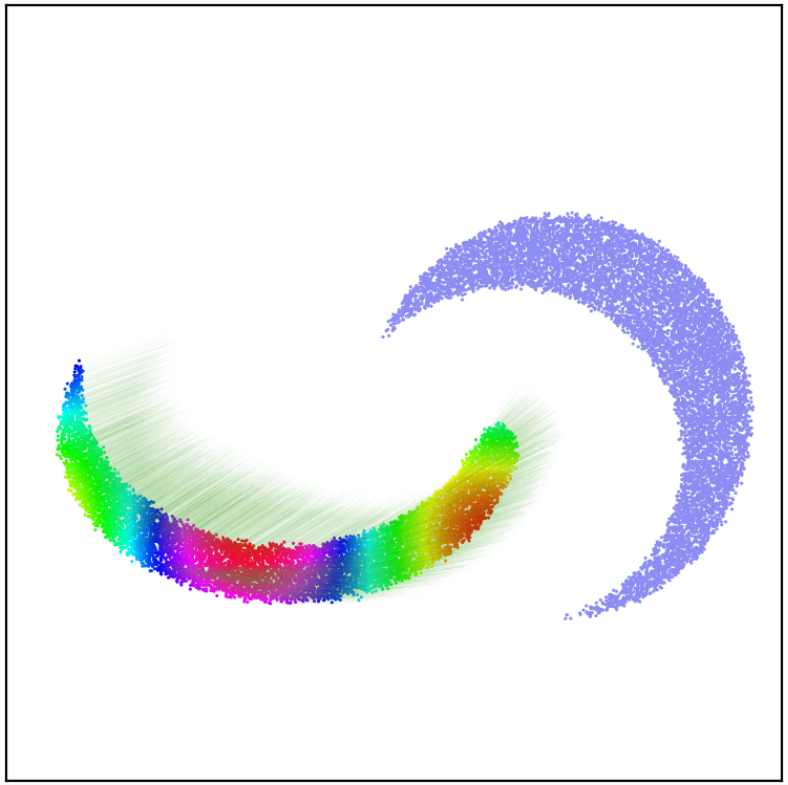
Une fonction d'erreur globale et géométrique



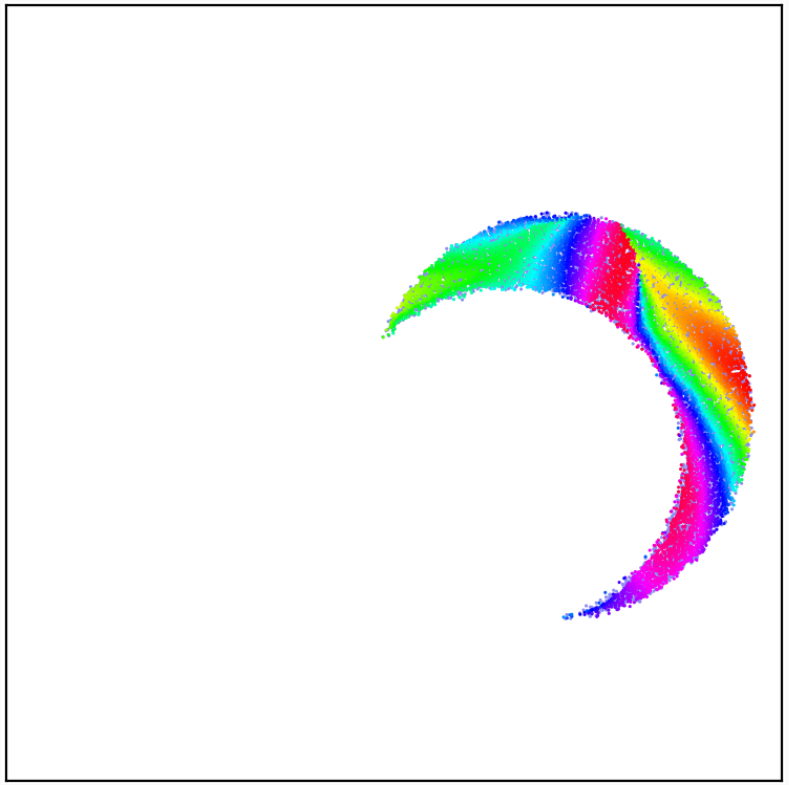
Un gradient de bonne qualité...

Mais pas de préservation de la topologie !

Transport optimal = recalage à petit prix ? Attention !



Avant



Après

$\Longrightarrow$ Garantir la préservation de la topologie est **bien plus difficile** :
voir le Chapitre 5 de ma thèse de doctorat.

Applications

Bilan

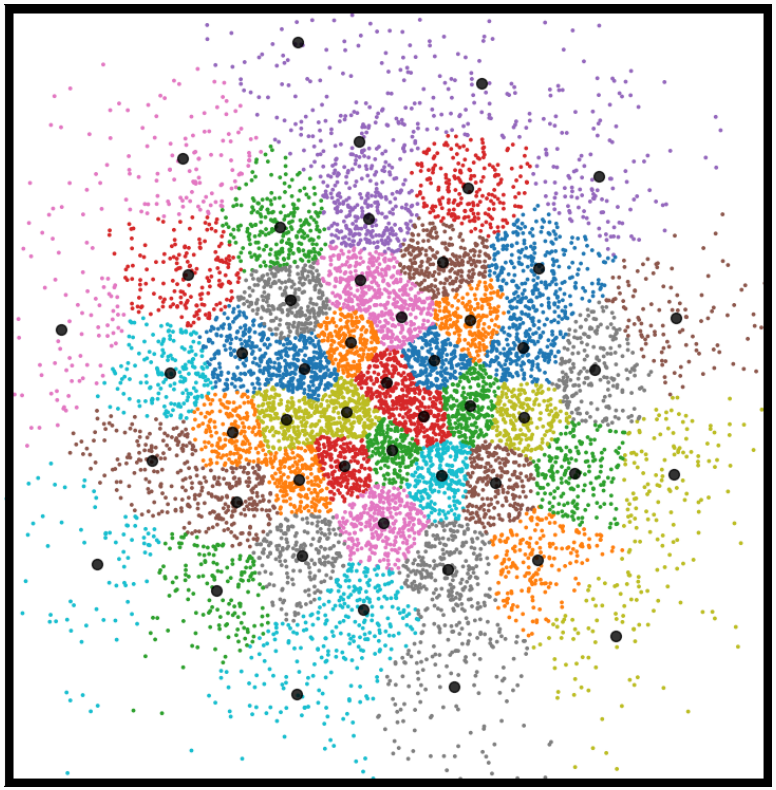
Motivation principale : **rendre facile l'analyse de formes.**
En 2020, travailler avec des **formes** devrait être
aussi simple que manipuler des **vecteurs.**

Deux outils robustes pour débloquer la recherche dans ce domaine :

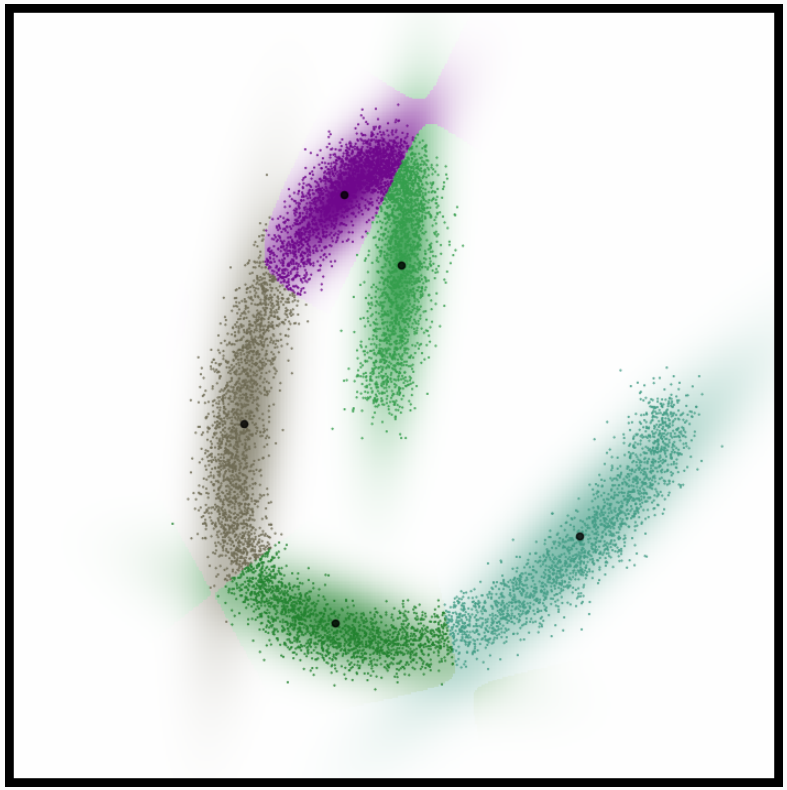
+ **Matrices symboliques** : rapides et versatiles.
+ **Fonctions d'erreur géométriques** : bons gradients.

$\Longrightarrow$ Très utiles en dehors de l'imagerie médicale !

KeOps est un outil adapté au machine learning



K-Means.

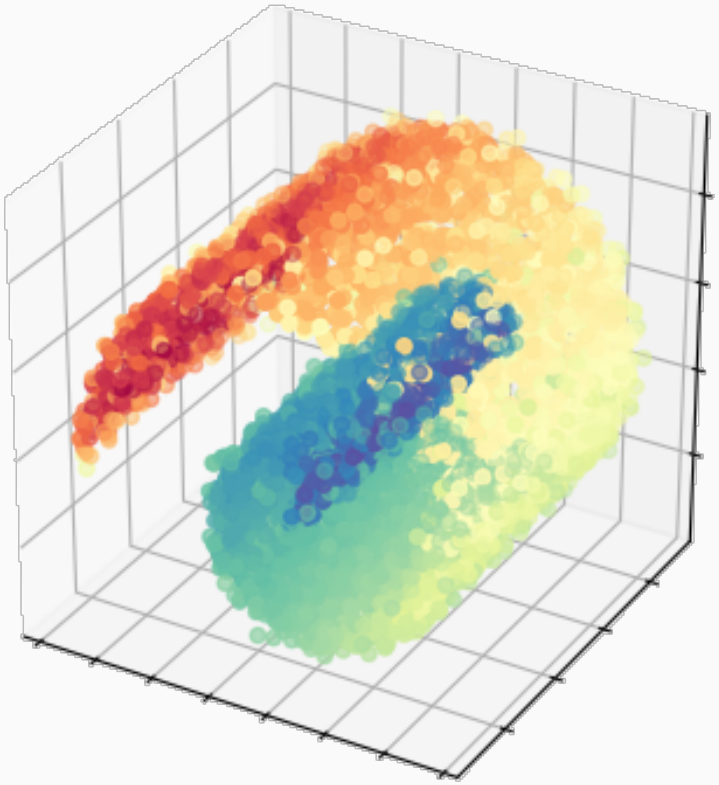


Modèle de Mixture de Gaussiennes.

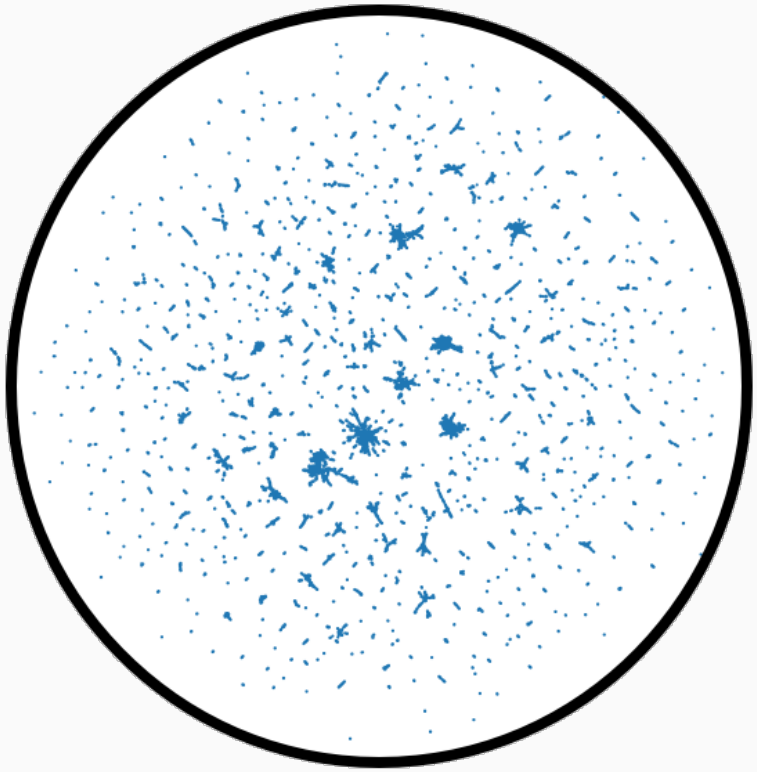
Compatible avec vos noyaux, métriques et formules préférées !

$\Longrightarrow$ De nouveaux tutoriaux arrivent en Octobre.

KeOps est un outil adapté au machine learning



Analyse spectrale.



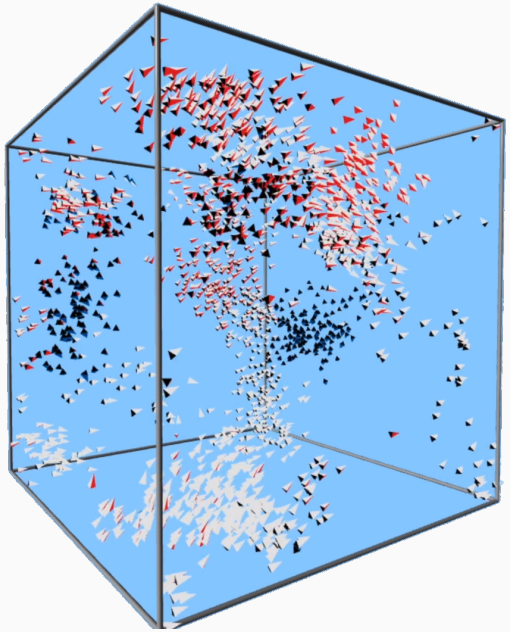
UMAP sur un espace hyperbolique.

Compatible avec vos noyaux, métriques et formules préférées !

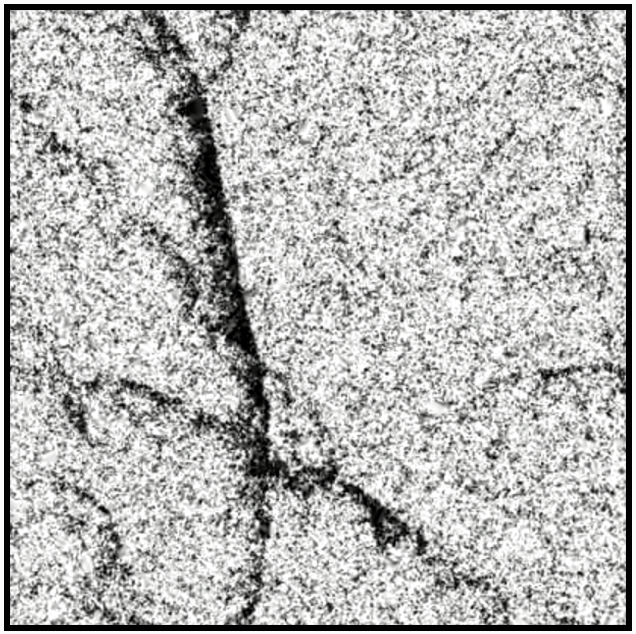
$\Longrightarrow$ De nouveaux tutoriaux arrivent en Octobre.

KeOps laisse les chercheurs se concentrer sur leurs modèles

Applications à l'étude des **systèmes dynamiques** [DM08, DFMAT17] et aux **statistiques** [CDF19] avec A. Diez, G. Clarté et P. Degond :



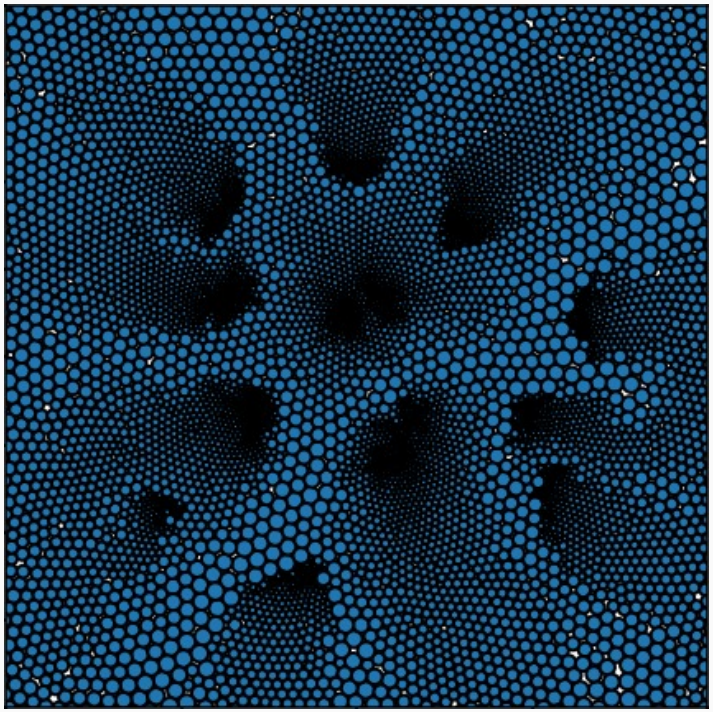
Modèle Vicsek 3D avec orientations, démo interactive avec 2k **oiseaux**.



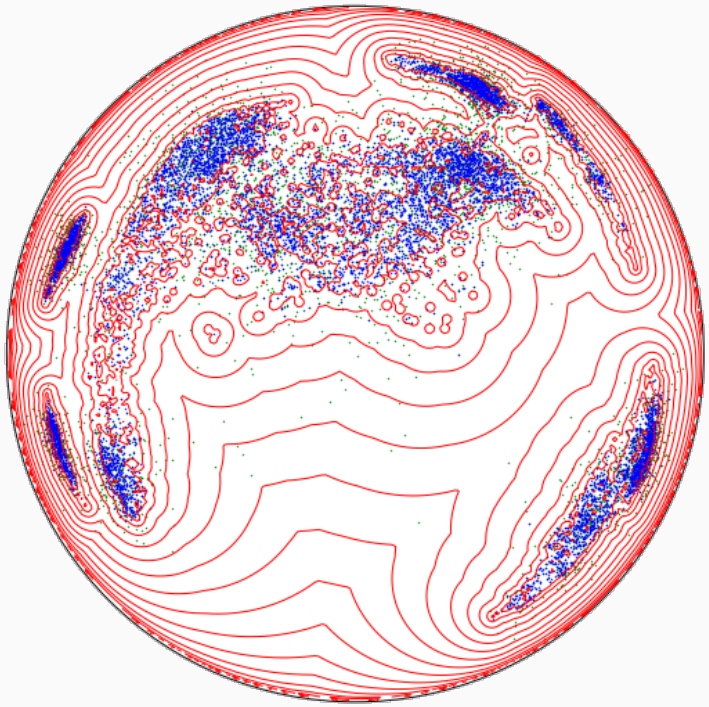
Modèle Vicsek 2D sur le tore, en temps réel avec 100k **nageurs**.

KeOps laisse les chercheurs se concentrer sur leurs modèles

$\Longrightarrow$ Travailler sur des **millions/milliards d'agents** en Python.

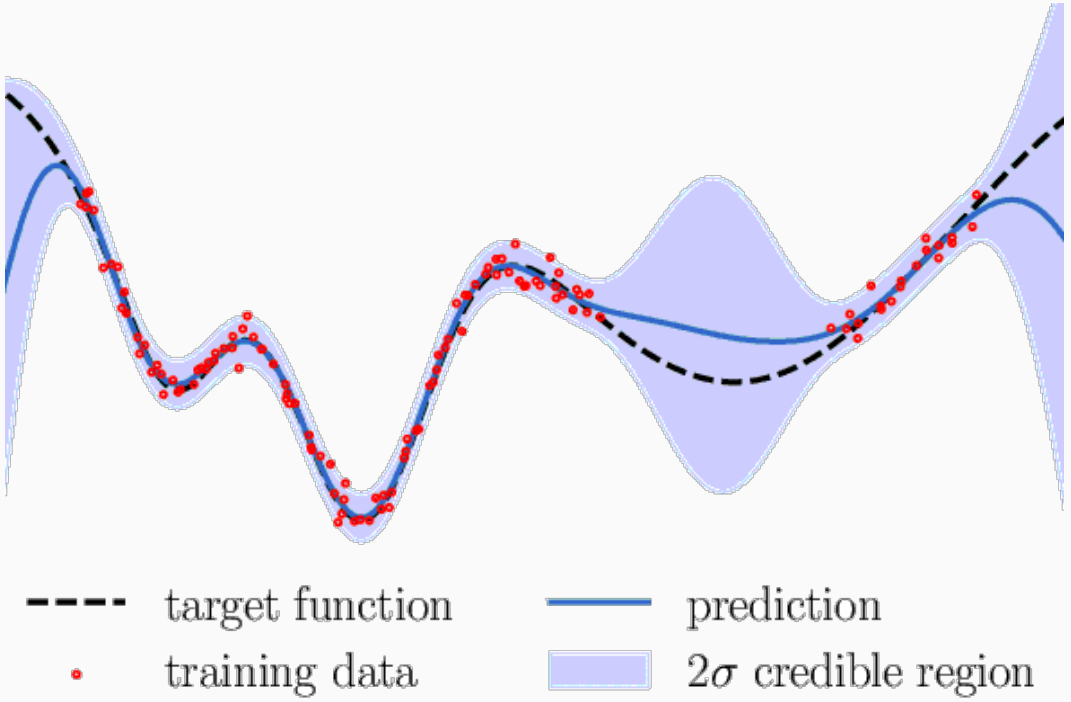


Problème de **packing** en 2D avec 10k disques.



Échantillonnage de Monte Carlo Collectif sur le disque de Poincaré.

Un outil standard pour la régression [Lec18] :



Sous le capot, résolution d'un **système linéaire à noyaux** :

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

où $\lambda \geqslant 0$ et $(K_{xx})_{i,j} = k(x_i, x_j)$ est une matrice symétrique positive.

Applications au Krigeage, splines, noyaux, processus Gaussiens

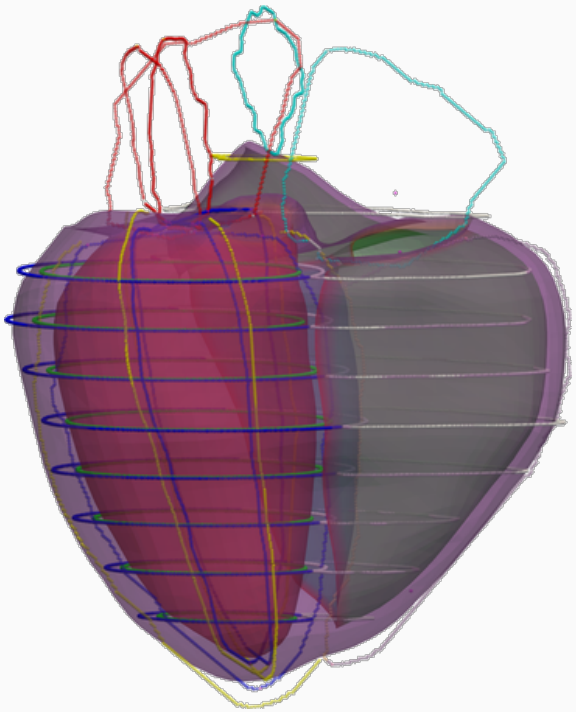
Les matrices symboliques de KeOps :

- Supportent les **solveurs standards** : SciPy, GPytorch, etc.
- Sur le jeu de données 3DRoad ($N = 278k$, $D = 3$) :

  7h avec 8 GPUs $\rightarrow$ **15mn** avec 1 GPU.

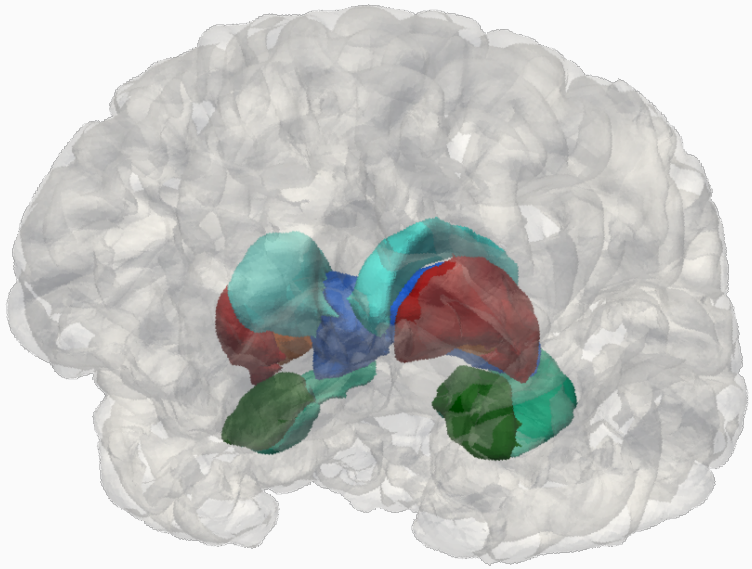
- Fournit un **backend rapide pour les chercheurs** : par ex.
  *Kernel methods through the roof : handling **billions of points** efficiently*, par G. Meanti, L. Carratino, L. Rosasco, A. Rudi (2020).

Ma première motivation : l'anatomie computationnelle



Recalage rapide par transport optimal

avec Samuel Joutard, Xu Hao
et Alistair Young de KCL.



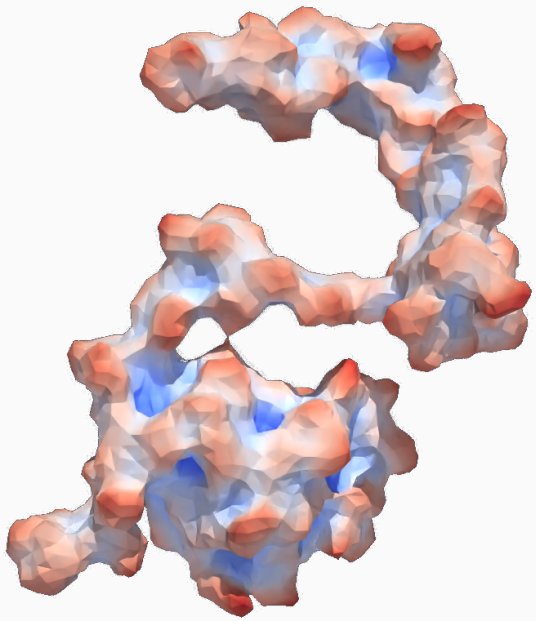
Recalage spline et difféomorphique

e.g. le logiciel LDDMM Deformetrica
avec l'équipe Inria Aramis.

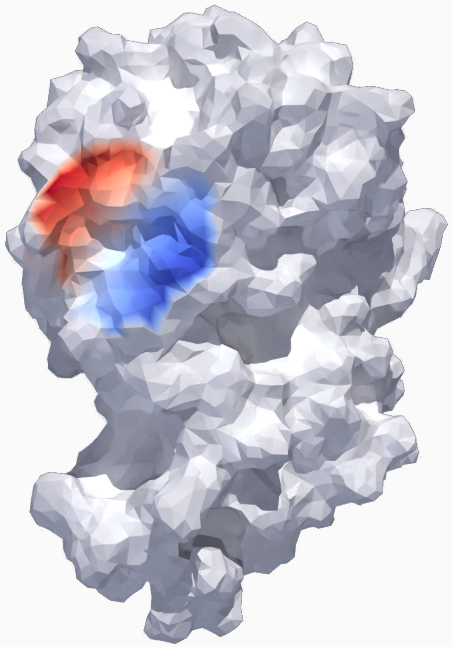
Deep learning géométrique avec F. Sverrisson et M. Bronstein

Méthodes géométriques et data-driven sur des **nuages de points** :

+ **K-NN rapides** : interactions locales.
+ **Calculs N-par-N rapides** : interactions globales.
+ **Batchs** hétérogènes, schémas rapides en octree.



Courbure moyenne à toutes les échelles.



Convolutions tangentes.

Améliorations à venir

KeOps et GeomLoss sont :

- \+ $\times$**10 -$\times$1,000 plus rapides** que des implémentations GPU standards.
- \+ **Peu gourmandes** : empreinte mémoire en $O(N)$ au lieu de $O(N^2)$.
- \+ **Versatiles**, avec une interface transparente.
- \+ **Puissantes et bien documentées** : idéales pour la recherche.
- − Lentes avec **des grands vecteurs** en dimension $D > 50$.

Améliorations à venir

KeOps et GeomLoss sont :

- $+$ $\times$**10**-$\times$**1,000 plus rapides** que des implémentations GPU standards.
- $+$ **Peu gourmandes** : empreinte mémoire en $O(N)$ au lieu de $O(N^2)$.
- $+$ **Versatiles**, avec une interface transparente.
- $+$ **Puissantes et bien documentées** : idéales pour la recherche.
- $-$ Lentes avec **des grands vecteurs** en dimension $D > 50$.

Noël 2020 : améliorations C++, intégration des **Tensor cores**.

- $\rightarrow$ **Accélération significative** quand $16 \leqslant D \leqslant 1{,}000$.
- $\rightarrow$ **Applications au traitement de langage naturel** : modules d'attention, Word Mover's Distance.

Un projet de recherche au long cours

Plan de route pour KeOps et GeomLoss :

2017–18 **Preuves de concept**, codes en ligne.
Premiers retours de la communauté.

2019–20 **Bibliothèque stable** avec des théorèmes solides, une bonne documentation. KeOps est intégrée par des logiciels de plus haut niveau.

2020–22 **Bibliothèque mûre** avec des articles d'application ciblés.
Un outil clé en main pour les étudiants et les ingénieurs.

2022+ **Une boîte à outils standard**, avec de vraies applications cliniques ? C'est l'objectif !

Conclusion

Point clés

- **Les matrices symboliques** jouent un rôle essentiel :
  $\longrightarrow$ KeOps, accélération x30 vs. PyTorch et TF.
  $\longrightarrow$ Utiles pour un grand nombre d'applications.

- Transport Optimal = **tri généralisé** :
  $\longrightarrow$ Gradients géométriques.
  $\longrightarrow$ Solveurs rapides en $O(N \log N)$.

- Pour aller plus loin en analyse de formes, nous devons développer **des modèles adaptés aux données, efficaces et robustes**.
  Nous disposons enfin des **bons outils** pour relever ce défi.

Un véritable travail d'équipe

Alain Trouvé

Thibault Séjourné

F.-X. Vialard

Gabriel Peyré

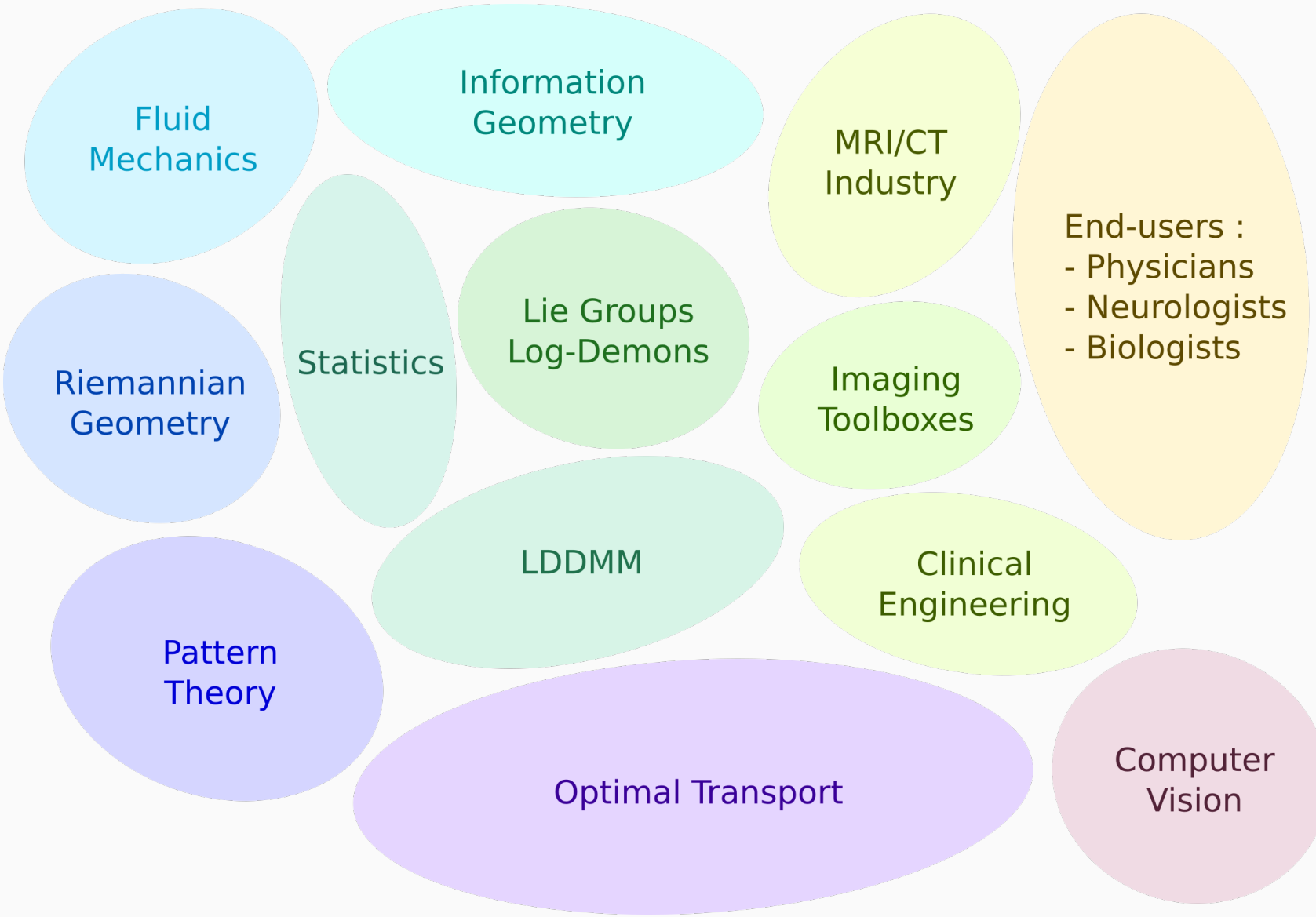
Benjamin Charlier

Joan Glaunès

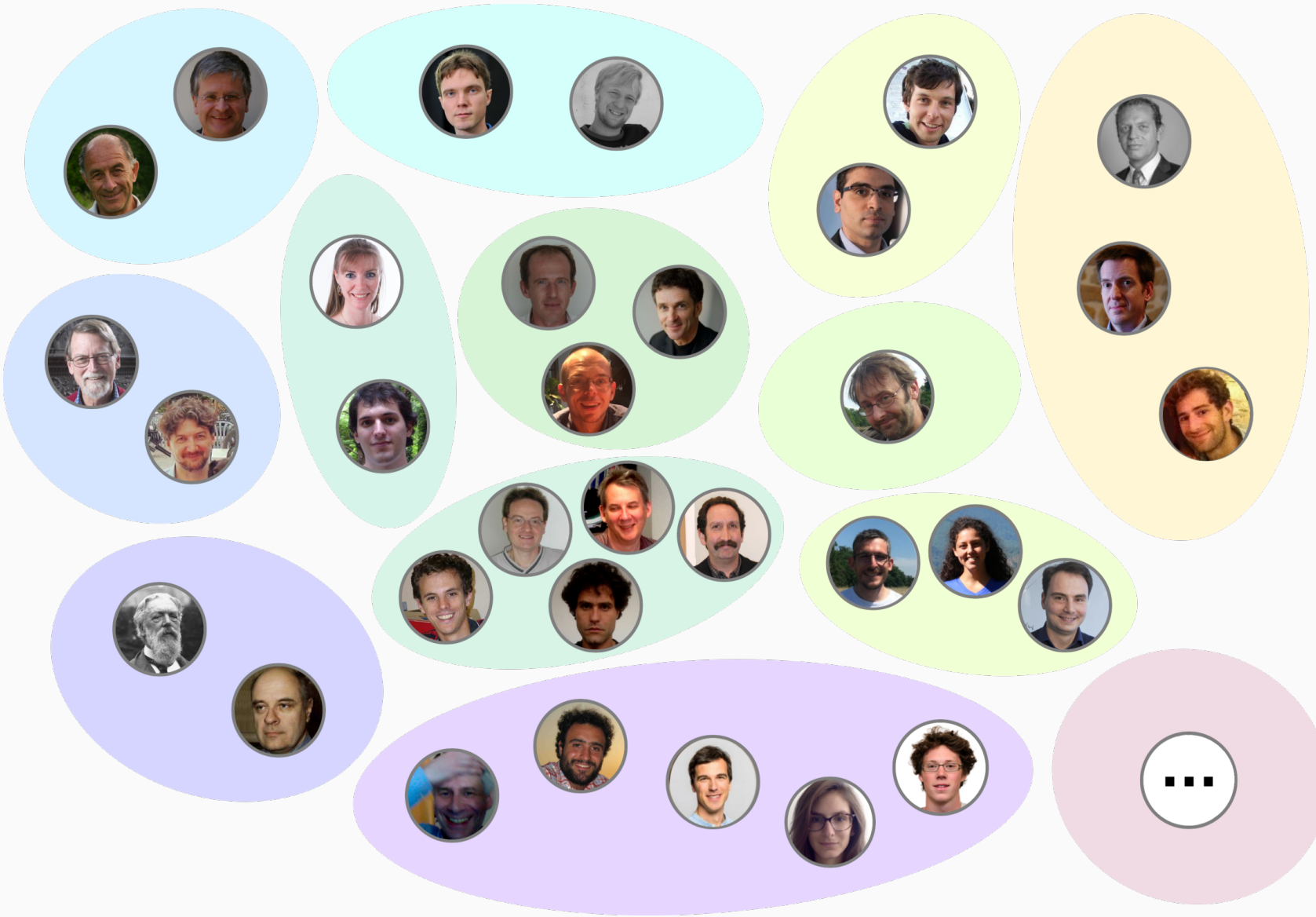
Pierre Roussillon

Pietro Gori

Promouvoir les interactions entre disciplines est essentiel



Promouvoir les interactions entre disciplines est essentiel



Parler aux étudiants et aux ingénieurs est une priorité

Documentation en ligne :

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

Thèse de doctorat, en introduction au domaine :

Geometric data analysis, beyond convolutions

`www.jeanfeydy.com/geometric_data_analysis.pdf`

Merci pour votre attention.

Avez-vous des questions ?

References i

M. Agueh and G. Carlier.

Barycenters in the Wasserstein space.

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

Dimitri P Bertsekas.

A distributed algorithm for the assignment problem.

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

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.

References ii


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.

References iii

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.

References iv

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.

References vi

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.

References viii

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.

References ix

Quentin Mérigot.

A multiscale approach to optimal transport.

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

Ptrump16.

lrm 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.