Geometric data analysis, beyond convolutions

Jean Feydy,

under the supervision of Alain Trouvé.

March 21, 2020

Abstract

To model interactions between points, a simple option is to rely on weighted sums known as *convolutions*. Over the last decade, this operation has become a building block for deep learning architectures with an impact on many applied fields. We should not forget, however, that the convolution product is far from being the be-all and end-all of computational mathematics.

To let data scientists explore new research directions, we present robust, efficient and principled implementations of three underrated operations:

1. Generic manipulations of *distance-like matrices*, including kernel matrix-vector products and nearest-neighbor searches.
2. *Optimal transport*, which generalizes sorting to spaces of dimension $\mathrm{D} > 1$.
3. Hamiltonian *geodesic shooting*, which replaces linear interpolation when no relevant algebraic structure can be defined on a metric space of features.

Our `PyTorch`/`NumPy` routines fully support automatic differentiation and scale up to millions of samples in seconds. They generally outperform baseline GPU implementations with `x10` to `x1,000` speed-ups and keep linear instead of quadratic memory footprints. These new tools are packaged in the `KeOps` (kernel methods) and `GeomLoss` (optimal transport) libraries, with applications that range from machine learning to medical imaging. Documentation is available at:

`www.kernel-operations.io/keops` and `/geomloss`.

This work intends to level the playing field between mainstream convolutional architectures and other methods. To complement the standard textbooks in data sciences, which cover statistics and optimization theory very well, we focus our presentation on geometric intuitions and computational efficiency. Appendices left aside, this manuscript should be accessible to all researchers, students and engineers with a background in mathematics, physics, data sciences or imaging.

Foreword

This manuscript sums up three years of work at the interface between medical imaging, machine learning and optimal transport theory. It is written with a strong emphasis put on geometric intuitions and low-level programming, to the detriment of analytic proofs. As we push this "editorial line" to an extent that is rather unusual for a thesis in applied mathematics, let us briefly explain our reasoning and motivations.

We're all studying the same algorithms. When students skim through the literature in data sciences, they often get overwhelmed by the amount of ideas that proliferate in researchers' minds. Scratching beyond the surface though, most impactful works implement variations around ever-present algorithmic themes: kernel matrix algebra in statistics, cascading convolutions in image processing, auction-like iterations for optimal transport, etc.

This numerical stability is mostly due to the design of silicon chips. The hardware industry puts a strong emphasis on *parallel* schemes and *contiguous* memory accesses, effectively creating a *computational bottleneck* that shapes the research landscape. Scientific programming languages follow suit: since the very first `Fortran` releases, they prioritize the support of *tensorized* computations, sparse matrices and convolutions to the detriment of other algorithmic structures. Put together, these constraints bias research in data sciences towards a small set of *scalable* (and nearly-algebraic) methods.

One method, ten interpretations. Surprisingly, in sharp contrast to the relative stability of the underlying numerics, the theorical literature in our field presents an ever-changing landscape. As described in the introduction of (Mallat, 1999), schools of thought generally emerge, discover a few pearls and disband (or move on to greener pastures) in cycles that span a decade or two. Throughout the literature, the theoretical frameworks that are devised to study similar numerical schemes thus *vary greatly from one applied field to another*. Critical insights often get scattered across non-overlapping communities, confusing most outsiders and students.

Exaggerating a little bit, we could say that theoretical data sciences is all about shedding *new lights* on *well-known algorithms* to unleash their full potential. As statisticians, geometers and computer scientists work on different aspects of the same problem, they progressively converge towards optimal implementations. At the end of each cycle, the main contributions of a research community are then summarized in well-documented toolboxes and companion textbooks, which provide building blocks for higher-level projects.

Where do we stand? In the last decade, *medical shape analysis* and *computational optimal transport* have expanded in a bubbling way. Going way beyond the handful of seminal works that achieved widespread recognition – e.g. (Beg et al., 2005; Mérigot, 2011; Cuturi, 2013) –

strings of remarkable papers have brought key contributions. Unfortunately though, with the notable exception of (Younes, 2010; Pennec et al., 2019) and (Peyré and Cuturi, 2017), a lack of accessible textbooks has also made our literature harder to understand by outsiders. Usually written with specific applications in mind (neuroanatomy, fluid mechanics, two-samples testing, etc.), most theoretical advances have been overlooked by the general public.

State-of-the-art codebases have followed the same dynamics, and are now too intricate to be modified without weeks of preliminary training. Unable to catch up with an ever-growing list of (essential) computational tricks, most researchers in the field have given up on *genuine* performance: a thriving literature is dedicated to the study of theoretical convergence rates or to the tuning of high-level `Python` scripts, but few papers ever discuss the substantial gains that can be brought by *dedicated* C++ *implementations*. This is unfortunate, as a trick which enables a `x10` speed-up is just as valuable to end-users as a key theoretical contribution.

Acknowledging these harmful trends in the community, the present work is all about *lowering the barrier of entry* to state-of-the-art results. In order to do so, we first take the time to write down some of the oral "folklore" that glues together advances in the field, without ever being made explicit in research papers. Second, leveraging new intuitions and cross-field insights, we propose improved algorithms and back them with theoretical guarantees that put an emphasis on *robustness*. Finally, and most crucially, we provide efficient implementations of our methods and endow them with *well-documented, user-friendly interfaces*.

Reaching out engineers is our priority. As we package our methods in a way that is most convenient for practitioners, we hope to bridge the gap between cutting-edge research and real-life applications. But should academics really spend so much time popularizing their work instead of focusing on what is supposedly their core business: proving theorems?

Distilling years of expertise, some research teams have succeeded in pushing industry boundaries "on their own". In neurosciences alone, we may cite the `Insight ToolKit` (Yoo et al., 2002), `FreeSurfer` (Fischl, 2012), `Dartel` (Ashburner, 2007), `Elastix` (Klein et al., 2009b), `Ants` (Avants et al., 2009) or the `Scikit-learn` package (Pedregosa et al., 2011).

Realistically though, in medical imaging as in most mature fields, mathematicians and computer scientists cannot have a meaningful impact if they stay by themselves. Developing and *maintaining* software that tackles real-life challenges is a job which is hardly compatible with an academic career. In this context, if new ideas are to make their way from our blackboards to genuine MRI scans, *they must first be understood and accepted by engineers*. Focusing our message on concepts that are valuable to the healthcare industry – sometimes to the detriment of mathematical good taste and conventions – is a necesary compromise.

Personal note. The general tone of this manuscript is a direct consequence of personal interactions with engineers and radiologists. After months of internship at Siemens Healthcare, numerous contacts with the booming Parisian tech scene (Therapixel, Owkin, etc.) and key discussions at Miccai conferences, I am slowly starting to understand the needs of our colleagues in the industry, who work under severe time constraints but always impress me with their scientific endeavour.

Unfortunately, after three years of PhD, I still lack the writing skills and scientific maturity that are needed to write an accessible textbook. I hope, nevertheless, that this introduction to our field will be valuable to readers from all backgrounds.

Extended abstract

Convolutions: strengths and limitations. Geometric data analysis is all about leveraging information on proximity and distance between samples. In a way that mimics Newton's gravitational laws, the simplest way of doing so is to rely on *convolutions* and model interactions between particles as sums of weighted contributions. These are parameterized by a *kernel* function that is set according to a model or estimated from the data.

Implemented with efficient low-level routines, the *convolution product* is at the heart of the deep learning revolution. Thanks to a massive investment from industry players (Nvidia, Google, Facebook, etc.), high-level `Python` libraries now allow researchers to write convolution-based algorithms – *neural networks* – easily, without compromising on performances. In the last decade, thousands of authors have thus proposed ways of *combining* and *tuning* convolution kernels to reach state-of-the-art performances in imaging or natural language processing.

Crucially though, **other operations are also worth studying**. Sorting (and its high-dimensional generalization, optimal transport), nearest-neighbor search, shape or image deformation are all relevant to many applied fields. Allowing them to reach a wide audience is the main purpose of this work: we provide robust, highly efficient and easy-to-use implementations of these fundamental operations, with new theoretical guarantees.

Map-reduce computations. In the first part of this manuscript, we review the foundations of deep learning numerics: GPU programming and automatic differentiation. Acknowledging the limitations of mainstream libraries, we introduce the new concept of semi-symbolic `LazyTensor`: distance- or kernel-like matrices that are not *sparse* in the traditional sense, but can nevertheless be encoded efficiently using a *mathematical formula* and small data arrays. Our C++ `KeOps` (Kernel Operations) library provides a comprehensive support for this new abstraction and comes with transparent `PyTorch`, `NumPy`, `R` and `Matlab` interfaces.

The `KeOps` engine allows `Python` scripts to scale up to graphics-like performances. It relies on a collection of efficient CUDA schemes that we combine with a custom just-in-time compiler and a versatile math engine that fully supports automatic differentiation. Scripts that encode distance or kernel matrices with `KeOps LazyTensors` outperform standard `TensorFlow` and `PyTorch` GPU implementations by *one or two orders of magnitude* while keeping a *linear instead of quadratic memory footprint*. As an example of application to Gaussian Process regression, the seamless switch to a `KeOps` backend by the developers of the `GPytorch` library resulted in a `x30` to `x100` speed-up.

The `KeOps` package is freely available on the `PyPi` repository (`pip install pykeops`). An extensive documentation and numerous tutorials are online at:

`www.kernel-operations.io.`

Optimal transport. Having laid down the computational foundations of our work, we focus on *measures*: a mathematical abstraction that unifies soft sets, weighted point clouds, segmentation maps and random vectors within a common framework. We start by a cross-field presentation of four major families of tools that can be used to compare measures with each other:

1. Pointwise f-divergences, such as the *relative entropy* and the *total variation*.
2. Projection-based *Hausdorff* and *chamfer distances*.
3. Convolution-based *Sobolev*, *kernel norms* and *maximum mean discrepancies*.
4. *Optimal transport* costs, also known as *Wasserstein* or *earth mover's distances*, which rely on the solutions of generalized sorting problems.

Focusing on the entropic regularization of optimal transport, we show that de-biased *Sinkhorn divergences* define convex, positive and definite loss functions that metrize the convergence in law and behave as *low-frequency Wasserstein distances*. To solve the associated transportation problems, we propose a symmetric, multiscale Sinkhorn loop that can be understood as a smooth and high-dimensional generalization of the *Quicksort algorithm*.

Our implementation outperforms the standard Auction, Sinkhorn and SoftAssign algorithms by *one to three orders of magnitude* (for applications to machine learning and shape analysis, respectively), scaling up to millions of samples in seconds. It allows us to define affordable loss functions that exhibit desirable *geometric* properties for the processing of random vectors, point clouds, curves, meshes and brain tractograms.

We package our solvers and functionals as simple `PyTorch` layers in the companion `GeomLoss` (Geometric Loss functions) library, which is freely available on the `PyPi` repository (`pip install geomloss`). Documentation and tutorials are online at:

`www.kernel-operations.io/geomloss`.

Riemannian geometry for shape analysis. Finally, we focus on the topology-aware metrics that can be defined on spaces of (anatomical) shapes. Reviewing the theories that underly registration pipelines in medical imaging, we identify some key shortcomings of the standard LDDMM framework. To tackle these issues in a way that is compatible with the advent of machine learning in computational anatomy, we propose an affordable and principled way of *normalizing* shape metrics to enforce *geometric axioms*.

Thanks to the `KeOps` and `GeomLoss` packages, scalable shape analysis pipelines can now be implemented using modular `Python` scripts. As the processing of *texture* is progressively being solved through the use of convolutional neural networks, these new tools will be valuable to researchers who work on the next open challenge in (medical) image analyis: the data-driven processing of *shapes*.

Limitations and future works. Please note that all the tools presented in this manuscript rely on the *extrinsic* metric of an ambient feature space, with pairwise distances given through explicit formulas. We may leverage mesh structures to compute feature vectors, such as triplets of (position, orientation, curvature) coordinates. But crucially, we never consider the *intrinsic* mesh distance that is central to elastic models and other graph-based methods. In years to come, we plan to work primarily on the interplay between these two descriptions of geometric data, with applications to medical imaging.

Glossary

$\alpha, \beta \in \mathcal{M}^+(\mathcal{X})$	Positive measures – see Section 3.1.3.
$\mathbf{C}(x, y)$	Ground cost function. Typically, $\mathbf{C}(x, y) = \frac{1}{2}\|x - y\|^2$.
Cometric	Inverse of a metric tensor. Fundamental quantity for geodesic shooting.
Compiling	Translating some human-readable program into an optimized binary code.
CPU	Central Processing Unit, standard sequential processor.
C++	Low-level programming language, standard for high-performance computing.
$\delta_x \in \mathcal{M}_1^+(\mathcal{X})$	Dirac mass at location $x \in \mathcal{X}$, defined Eq. (3.9).
$(\Delta_k) \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 3}$	List of triangle faces that make up a surface mesh.
$\mathrm{d}_x F(x_0) \cdot h$	Differential of F at x_0 applied to h, as discussed Eq. (2.3)
$\mathrm{d}_x^\top F(x_0)$	Gradient operator of F at x_0, defined Eq. (2.8)
f, g	Dual potentials for the OT problem – see Eq. (3.170).
F, G	De-biased potentials for the OT_ε problem – see Eq. (3.209).
Feature space $\mathcal{X}$	Domain that contains our samples; typically, a subset of $\mathbb{R}^{\mathrm{D}}$.
g_x	Riemannian metric.
Geodesic	Locally straight curve.
`GeomLoss`	Package for computational optimal transport, presented in Section 3.3.4.
GPU	Graphics Processing Unit, massively parallel computing chip.
$H(q, p)$	Hamiltonian function, defined Eq. (5.21).
K_q	Kernel matrix or Riemannian cometric.
`KeOps`	Kernel Operations library, presented in Chapter 2.
Kernel k	Function that attributes a weight to a distance or to a pair of points.
$\mathrm{KL}(\alpha, \beta)$	Kullback-Leibler divergence, discussed in Section 3.2.1.
$(\Lambda_k) \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 2}$	List of line segments that make up a discrete curve.
`LazyTensor`	Semi-symbolic matrix, encoded using a formula and small data arrays.
$\mathrm{Loss}(A, B)$	Penalty that measures the discrepancy between two objects.
Map-reduce	Combination of a function and a reduction (e.g. a distance and a minimum).
Measure	Distribution of mass. Abstraction for random vector, segmentation map, etc.
$\min_\varepsilon$	SoftMin operator, defined Eq. (A.33).
Optimal transport	Generalization of sorting to high-dimensional spaces.
$\mathrm{OT}_\varepsilon(\alpha, \beta)$	Regularized optimal transport cost, defined in Section 3.3.1.

$\pi \simeq (\pi_{i,j}) \in \mathbb{R}_{\geqslant 0}^{\mathrm{N}\times\mathrm{M}}$	Optimal transport plan, solution of Eq. (3.168,3.187,3.214).
`Python`	High-level programming language, popular among data scientists.
`PyTorch`	Deep learning `Python` library, popular among academics.
$\mathbb{R}^{\mathrm{D}}$	Vector space of dimension D.
Riemannian metric	Point-dependent scalar product that encodes an adaptive geometry.
$\mathbb{S}^{\mathrm{D}}$	Unit sphere of dimension D, embedded in $\mathbb{R}^{\mathrm{D}+1}$.
$\mathrm{S}_\varepsilon(\alpha, \beta)$	De-biased Sinkhorn divergence, defined in Section 3.3.2.
`TensorFlow`	Deep learning `Python` library, industry standard.

Remerciements

Mon directeur. À mes yeux comme à ceux de tous ses élèves, Alain Trouvé est un modèle. D'intégrité, d'abord : en m'imposant de toujours digérer une idée avant d'en faire la publicité, il m'a permis de faire mûrir une succession de bidouilles en un projet scientifique construit. D'ouverture, ensuite : en dépit d'une remarquable carrière, qui aurait pu le pousser vers un certain esprit de clocher, il m'a toujours encouragé à me projeter vers l'extérieur et à profiter de la diversité de notre discipline. De générosité, enfin : je regretterai longtemps nos journées de débat à Cachan – avec petit chocolat au Pavillon des Jardins. Plus encore que pour son apport direct à notre travail, je serai donc toujours reconnaissant à Monsieur Trouvé de m'avoir appris à *prendre le temps de faire de la science*.

Mes collègues. Non content de bénéficier des conseils de M. Trouvé, j'ai eu la chance de partager ces trois années de thèse avec un quatuor de chercheurs sympathiques et extrêmement compétents : Benjamin Charlier, Joan Alexis Glaunès, François-Xavier Vialard et Gabriel Peyré.

Mes aînés de dix ou quinze ans, ces co-encadrants de luxe ont eu la gentillesse de me traiter en collègue – tout en m'enseignant les ficelles du métier. Travailler avec eux et mes trois autres proches collaborateurs – Thibault Séjourné, Pietro Gori et Pierre Roussillon – aura été un plaisir quotidien. Comme indiqué en tête de chaque chapitre, les travaux exposés dans ce manuscrit sont autant dus à leurs efforts qu'aux miens.

Plus généralement, cette thèse a été pour moi l'occasion de découvrir les communautés du transport optimal et de l'imagerie médicale, avec l'analyse de formes comme point de rencontre. J'y ai découvert un environnement scientifique sain, des collègues formidables, à mille lieues du coupe-gorge que l'on décrit souvent: Jean-David Benamou et Marc Niethammer, qui ont gentiment accepté de relire ce manuscrit, en sont de parfaites illustrations. En imagerie médicale surtout, la rencontre d'expertises variées dans un environnement bien structuré, aux enjeux définis, permet des échanges d'une grande qualité. Sauver une grand-mère à l'aide d'un scanner IRM qui combine le travail de physiciens, d'ingénieurs, de mathématiciens, d'informaticiens et, surtout, de manipulateurs et de médecins est un défi qui vaut *vraiment* la peine d'être relevé.

Je ne saurais donc trop conseiller aux jeunes étudiants de l'ENS et du MVA qui liraient ces lignes de considérer un parcours au voisinage de la médecine : on n'y manquera jamais de problèmes mathématiquement intéressants, véritablement utiles et correctement financés.

Les médecins. Enfin… Venant d'une famille de médecins, je ne suis pas complètement impartial à ce sujet. Grâce à mon père, professeur de radiologie à Paris 5 (Descartes – Hôpital Cochin), j'ai eu la chance de pouvoir interagir à de nombreuses reprises avec des spécialistes qui m'impressionnent toujours par leur intelligence et leur expérience de la vie. Je souhaite notamment remercier Alexandre Krainik, Francis Besse, Gaspard d'Assignies, Olivier Vignaux et Patrice Coudray de m'avoir fait confiance en me permettant d'intervenir dans leurs conférences et séminaires : tout recommandé (ou pistonné) que je sois par Papa, je ne reste qu'un petit mathématicien, pas même diplômé. Plus proches de ma spécialité, Mariappan Nadar et Boris Mailhe (mes anciens patrons chez les Siemens Healthineers), Yaroslav Nikulin (de Therapixel), Stéphanie Allassonnière (de Paris 5) et Tom Boeken (de l'Hôpital Pompidou) ont eu la gentillesse de me faire découvrir l'indispensable tissu intermédiaire, entre ingénierie et statistiques, qui permet aux géomètres d'apporter leur pierre à notre système de santé.

Des JFR à la "Masterclass IA-Radiologie", ces rencontres m'ont permis de garder les pieds sur terre. De garder bien en tête la finalité de nos recherches. Influencé par mes (arrière-)grand-pères, mon oncle ou mon père – auxquels se joint maintenant ma sœur – j'ai toujours vu la médecine comme la filière des super-héros... Sans avoir jamais eu le courage de m'y lancer : les gardes deux fois par semaine pendant l'internat, non merci !

Pouvoir aujourd'hui contribuer à cet effort commun sans quitter mon (très confortable) bureau de la rue d'Ulm est une chance inespérée. Pour tâcher d'en être à la hauteur, j'ai beaucoup œuvré à faciliter l'utilisation de nos outils par les ingénieurs du domaine : implémentation, test, packaging, documentation... Un travail fastidieux, qui m'a souvent détourné des maths, mais qui me semble indispensable à la diffusion de nos théorèmes auprès du monde hospitalier. J'espère sincèrement qu'à moyen terme, cet investissement permettra à nos idées d'apporter une plus-value clinique concrète.

Les Écoles Normales Supérieures. À l'heure où je déménage pour Queensway, il me semble important de remercier une dernière fois l'ENS pour ces sept (longues) années qui m'ont vu grandir et peu à peu devenir adulte. De mes études entre Cachan et Paris, je retiendrai l'enthousiasme de MM. Hills, Werner, Mallat, Filliâtre puis Tanré, Veltz, Delingette et Pennec, soutenus par l'extrême compétence de Mmes. Vincent, Trémeau, Auffray et Elmir.

Je dois notamment une fière chandelle aux directeurs du DMA: Messieurs Biquard, Bernard et Viterbo. Après m'avoir judicieusement détourné de la logique vers les maths apps' en fin de première année, puis vers le MVA de l'ENS Cachan en troisième année, ces trois professeurs ont – à ma très grande surprise – insisté pour me recruter comme *caïman* à la sortie de l'École. En me donnant carte blanche sur ce poste de "matheux appliqué proche des élèves", ils m'ont fait cadeau d'une opportunité en or pour partager de belles idées avec une ribambelle d'élèves sympathiques, studieux et intéressés. Je garderai toujours un excellent souvenir de nos séances d'*Introduction à la géométrie Riemannienne par l'étude des espaces de formes*, des TDs de *Sciences des données* pour Gabriel Peyré et, surtout, des cours de *Culture mathématique pour les non-matheux* où l'on se sera vraiment éclaté du début à la fin.

Ma famille. Le mot de la fin, le plus important, est bien sûr pour ma famille. Une petite dédicace à ma mère, d'abord : pour mes treize ans, l'achat d'un premier ordinateur HP familial avec autorisation de bidouille illimitée – plutôt que la GameCube tant désirée – a sans doute été à l'origine de ma vocation pour les maths computationnelles. Plus sérieusement, rester disponible et proche des Allain–Feydy–Song sera toujours ma première priorité. Grâce à Michael Bronstein, Pierre Degond et Dominique Bonnet qui ont accepté de nous recruter à l'Imperial College de Londres, Anna et moi avons aujourd'hui la chance d'habiter ensemble, pas bien loin de Paris ; j'espère pouvoir toujours en dire autant.

Je terminerai notre instant francophone en dédiant ce manuscrit à mes grand-parents, qui m'ont (presque) tout appris de la vie. Le désir de passer de longues années en leur compagnie sous-tend mon travail depuis quatre ans déjà.

Contents

Chapter 1

Introduction

Key points – Shape analysis is relevant to medical imaging:

1. This work is motivated by applications to computational anatomy: we aim to generalize standard statistical tools for population study to anatomical (shape) data.
2. Over the last decade, the data-driven optimization of convolution filters has revolutionized imaging sciences. **Pattern detection and tissue segmentation** are now tackled by algorithms that leverage the **texture** of input images.
3. Unfortunately, the "deep learning revolution" has not yet happened for **geometry processing and shape analysis**: in computational anatomy, data-driven approaches do not yield significantly better results than decades-old baseline models.
4. To make a breakthrough in the field, researchers have to combine statistical learning methods with domain-specific algorithmic structures.

Contributions – Enabling large-scale machine learning for geometric data:

5. We provide first-rate support for point clouds on the GPU, with a differentiable `Python` interface. Our routines allow researchers and data scientists to reach **graphics-like performances** with a convenient **deep learning interface**.
6. Throughout this work, we put an emphasis on **robustness**. To enforce an **invariance to re-sampling and re-meshing**, we encode our shapes as weighted point clouds – *measures* – and restrict ourselves to well-defined, homogeneous operations. This prevents our algorithms from paying too much attention to contingent encoding choices, such as the discrete triangulation of a surface mesh, and let them focus instead on meaningful geometric features.
7. We discuss the different types of metrics that can be defined to compare shapes with each other – from Procrustes analysis to diffeomorphic registration. Hausdorff, kernel and Wasserstein distances now scale up to high-resolution shapes in fractions of a second. Going further, we propose new methods to make sure that data-driven deformation models satisfy key geometric axioms.

Chapter 1 – Introduction:

1.1 Medical imaging and geometry

Medical imaging. Providing quality healthcare at an affordable price is a concern for all societies. Over the last forty years, in Western countries, a key force driving the increase of health standards has been the sustained improvement of imaging devices. Benefitting from decades of research, medical doctors now acquire 3D pictures of their patients' inner workings in minutes.

Major industry players produce thousands of MRI, CT and ultrasound scanners per year: gradually, modern imaging techniques have thus become available to an ever-increasing number of patients. Unfortunately, the number of skilled radiologists able to interpret these images cannot grow accordingly: as far as global access to healthcare is concerned, the scarcity of human resources is now the main bottleneck to resolve.

As mathematicians, can we help? Researchers working towards a semi-automation of clinical exams encounter numerous challenges: from the acquisition of MRI volumes to the global estimation of population trends, turning raw sensor data into a meaningful piece of information is an arduous process, illustrated Figure 1.1. In this thesis, we discuss a specific segment of this considerable pipeline: the **analysis of anatomical data**.

Starting from clean 3D scans provided by our colleagues upstream, we focus on extracting geometric information in a way that is relevant for ulterior analyses. Our job is to provide **meaningful, reproducible, high-level representations of a patient's anatomy**, to be used by medical doctors and statisticians. In this specific branch of imaging sciences, we can identify three major types of problems – illustrated Figure 1.2:

1. **Feature detection:** spot tumors, calcifications or fractures on 2D and 3D images.
2. **Shape analysis:** quantify anatomical deformations and variations of organs' geometries.
3. **Biomechanical simulation:** leverage prior knowledge of the human body to extract physiological information out of a mere grayscale image.

Computational anatomy. This thesis is dedicated to the "intermediate" problem of medical shape analysis, illustrated Figure 1.2.b. We may sometimes tackle general questions related to multiple research fields; but ultimately, our efforts always lean towards the settings and ranges of problems that are relevant to the processing of medical data.

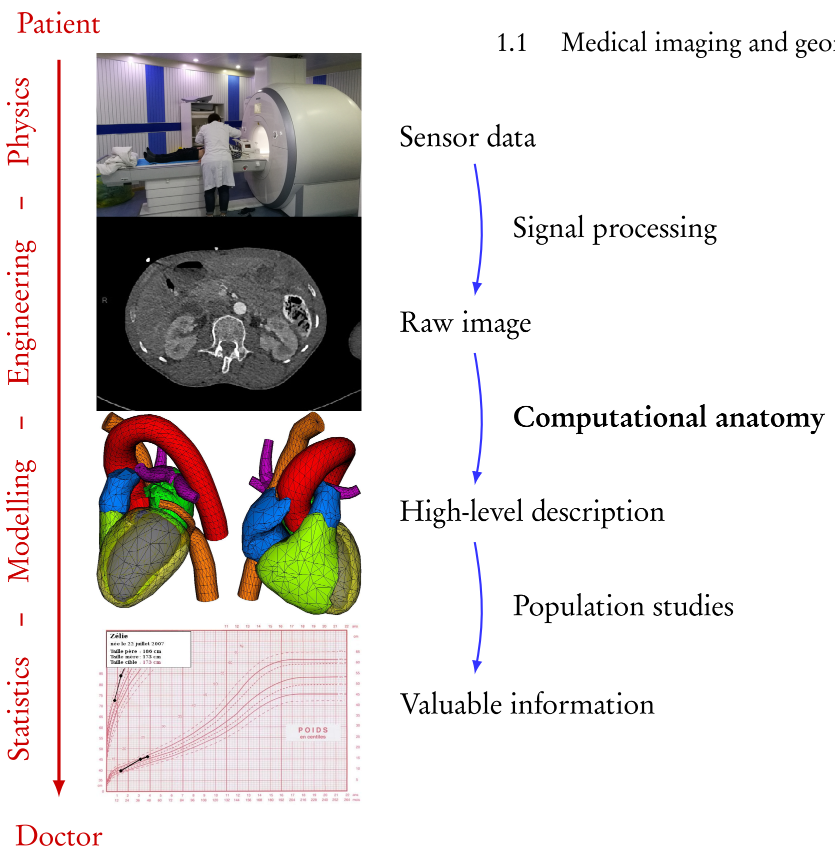
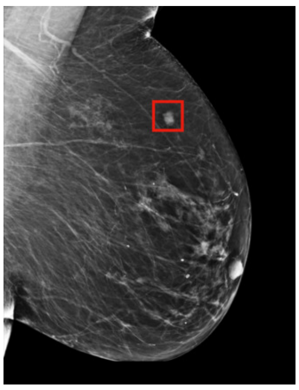
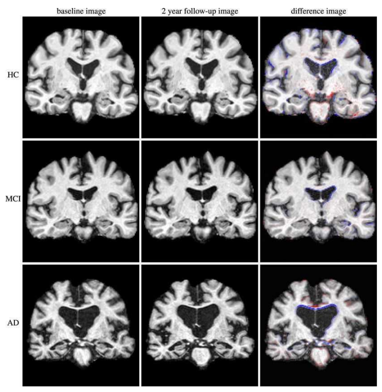


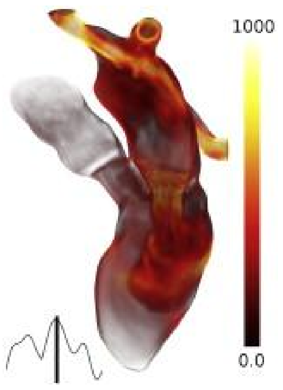
Figure 1.1: Illustrating the medical imaging pipeline with images courtesy of Tom Boeken from the Pompidou Hospital and (Ptrump16, 2019; Ecabert et al., 2011). Medical imaging is a mature and structured field, with well-defined specialities. Physicists, engineers, computer scientists, mathematicians and technicians all team up to bridge the gap between a patient's anatomy and a doctor's diagnosis.



(a) Spot patterns.



(b) **Analyze geometric variations.**



(c) Fit models.

Figure 1.2: Three challenges for computational anatomy, from (Conner-Simons and Gordon, 2019; Ledig et al., 2018; Chnafa et al., 2014). (a) In a medical setting, **feature detection** algorithms may be used to spot fractures, screen for breast cancer and semi-automate countless other clinical exams. (b) Assessing the **shape** of a patient's organs with respect to population trends is a key step in the diagnosis of many pathologies, including Alzheimer's disease. (c) Finally, geometric information can be combined with a strong physiological prior to create **biomechanical models** of a patient's anatomy. Over the last decade, this ambitious strategy has been successfuly applied to blood flow simulation: it now powers major medical tech companies such as Arterys and Heartflow, with applications to the detection of cardiac diseases and surgical planning.

1.1.1 The deep learning revolution

In the wake of (Krizhevsky et al., 2012), statistical learning techniques have had a major impact on imaging sciences. Before introducing the questions that motivate this thesis, we discuss briefly the main features of the "deep learning" framework that currently shapes the landscape in biomedical research.

Supervised learning. Most applications of learning theory to medical imaging stem from a simple remark: instead of tuning the parameters of our algorithms by hand, we should leave this task to standard optimization routines – and focus instead on the global *architectures* of our programs.

In practice, scientists *train* their models by choosing a vector of parameters that optimizes an empirical performance score. Let us assume that we are to regress a functional relationship "$x \mapsto y$" between two variables x and y: say, an image and a medical diagnosis. If $(x_i, y_i)_{i \in [\![1,\mathrm{N}]\!]}$ is a *dataset* of input-output pairs labelled by experts, and if "$\mathrm{F} : (\theta\,;x) \mapsto y$" is a suitable parametric model, we use gradient-based descent schemes to pick a value of the vector of parameters θ that roughly minimizes the *training error*:

$$\mathrm{Cost}(\theta) ~=~ \tfrac{1}{\mathrm{N}} \textstyle\sum_{i=1}^{\mathrm{N}} \mathrm{Loss}(\,\mathrm{F}(\theta\,;x_i)\,,\,y_i\,) ~+~ \mathrm{Reg}(\theta)\,, \qquad (1.1)$$

where "Loss" and "Reg" are the so-called *data attachment* and *regularization* penalties.

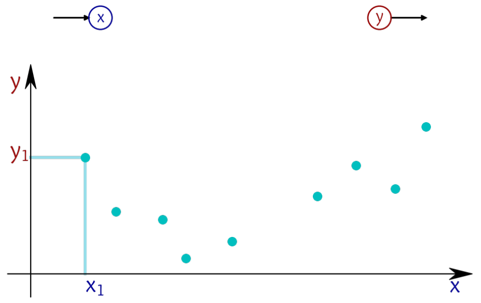
As illustrated Figure 1.3, the usual setting of *polynomial regression* corresponds to the case where $\mathrm{F}(\theta\,;x)$ is a polynomial function of x with coefficients in θ, and $\mathrm{Loss}(x,y) = |x-y|^2$ is a quadratic penalty. Alternatively, when $\mathrm{F}(\theta\,;x)$ is given as a composition of matrix-vector products and pointwise non-linearities, we speak of *multi-layer perceptrons* or *fully connected neural networks*.

The curse of dimensionality. Generic regression or interpolation strategies work best when the training database $(x_i, y_i)_{i \in [\![1,\mathrm{N}]\!]}$ covers well the space of input configurations. Without any prior knowledge of the problem's structure, statistical learning algorithms can then create efficient "hash tables" that approximate any given function.

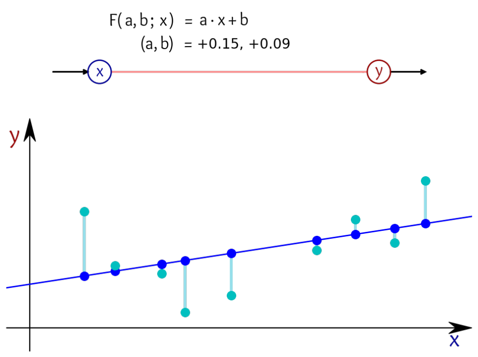
Unfortunately, such strategies can *not* be applied directly to imaging problems. In these settings, the high-dimensional space of input configurations is way too large to be sampled comprehensively: no dataset will ever contain a *dense* sampling of the set of all brain MRIs, as measured by the standard Euclidean distance. Structuring modelling hypotheses – *priors* – are therefore required to let programs extrapolate – *generalize* – outside of training databases in a sensible way.

Convolutional neural networks. Since the early days of imaging research, an algorithmic structure stands out: the composition of cascading convolutions. Iterated filterings can be used to define feature maps at all scales, through affordable combinations of neighboring pixel values. Over the years, variants of this idea have been described as e.g. *Laplacian pyramids* (Burt and Adelson, 1983), *part-based models* (Felzenszwalb et al., 2009) or *wavelet transforms* (Mallat, 1989), as illustrated Figure 1.4.

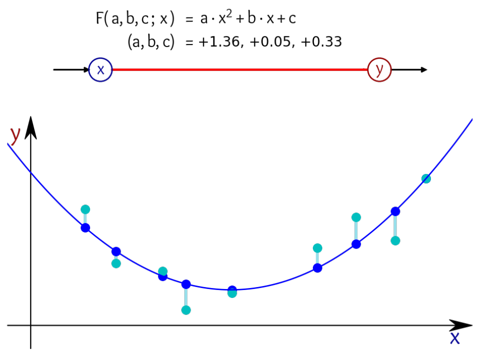
This algorithmic framework strikes a good balance between power and simplicity. At an affordable cost, it allows researchers to "hard-code" fundamental priors in image processing pipelines: translation-invariance, locality, multiscale integration of intermediate representations. As illustrated in Figures 1.5 and 1.6, the key idea behind *convolutional neural networks* is then to **combine the regression paradigm of Eq. (1.1) with this domain-specific architecture**.



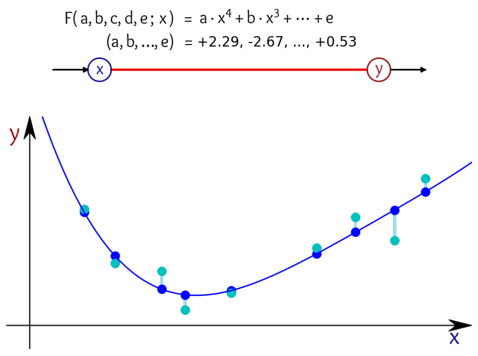
(a) Dataset.



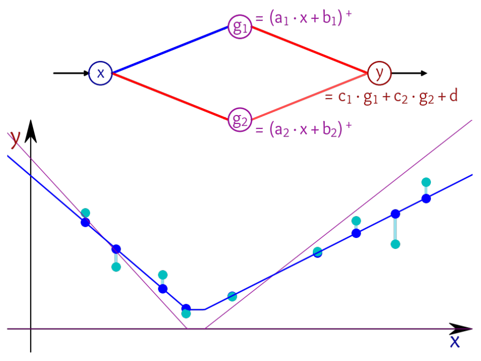
(b) Linear model.



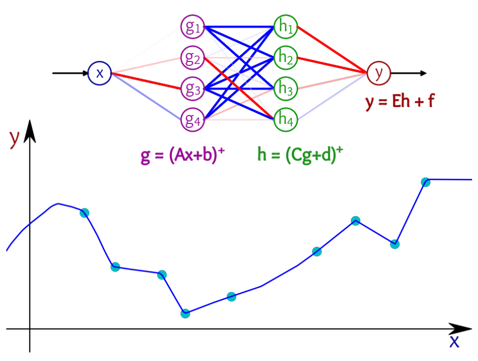
(c) Quadratic model.



(d) Quartic model.



(e) Two hidden variables.



(f) Multi-layer perceptron.

Figure 1.3: "Supervised learning" generalizes "linear regression" to complex models.
(a) We depict a dataset of input-output pairs (x_i, y_i) in $\mathbb{R} \times \mathbb{R}$. As discussed Eq. (1.1), supervised learning is all about fitting a parametric model "$\text{F}(\theta\,; x) \simeq y$" to the data by finding a suitable value of the vector of parameters θ. A simple way of doing so is to minimize the mean squared error $\text{Cost}(\theta) = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} |\text{F}(\theta\,; x_i) - y_i|^2$ by gradient descent with respect to θ.
(b) In the simplest of all settings – *linear regression* – we consider a linear model parameterized by a slope a and an offset b. We represent the positive correlation between x and y as a red segment linking the input node to the output. (c-d) A simple way of getting a closer fit to the data is to consider higher-order polynomials: quadratic parabolas, cubic or quartic curves, etc.
(e) Alternatively, we may introduce **intermediate parametric variables**, which allow us to generate complex behaviors using simple operators – say, affine scalings and pointwise applications of the "REctified Linear Unit" or "positive part" ReLU : $x \mapsto x^+$. (f) These computer-friendly models are easy to extend and combine with each other. Out-of-the-box, the so-called *multi-layer perceptrons* encode piecewise-linear models with a prescribed number of bendings.

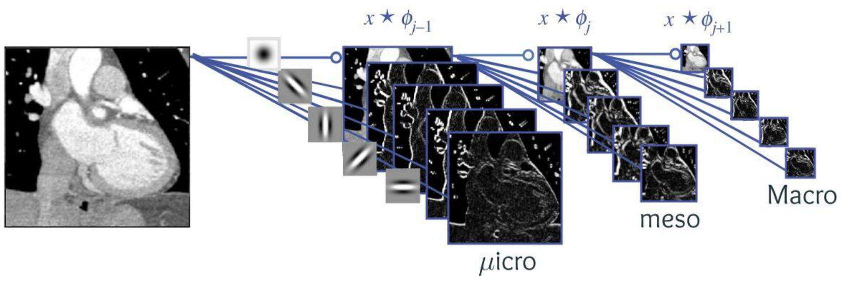


Figure 1.4: A typical wavelet decomposition, adapted from (Mallat, 2016). Filtering – *convolution* – and sub-sampling – *pooling* – have a rich history in image processing. The theory of wavelets, discussed at length in (Mallat, 1999), describes efficient multiscale transforms with prescribed mathematical properties. These rely on finely engineered high- and low-pass convolution filters to produce sparse feature maps at all resolutions. To keep things simple in this introduction, we speak here of "micro-", "meso-" and "macro-scopic" scales, that correspond to superficial and deeper layers of the transform.

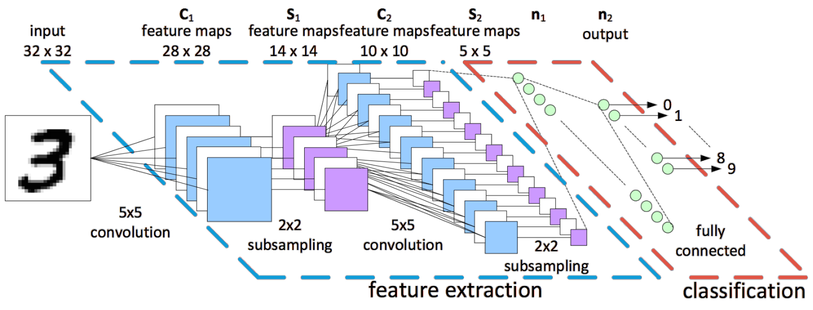


Figure 1.5: Architecture of a convolutional neural network, from (Peemen et al., 2011). For applications to medical imaging and computer vision, *data-driven* convolution filters outperform *explicit* wavelet coefficients by a wide margin. In practice, researchers define imaging pipelines parameterized by tunable convolution filters. The latter are then optimized by gradient descent on classification or segmentation tasks, and provide high-quality feature maps at all scales. For historical reasons, these algorithms are usually described using a pseudo-biological vocabulary: convolution filters are called *neural weights*, while *neural networks* refer to parametric transforms. As discussed Figure 1.3, the *training process* refers to the gradient-based resolution of a regression problem.

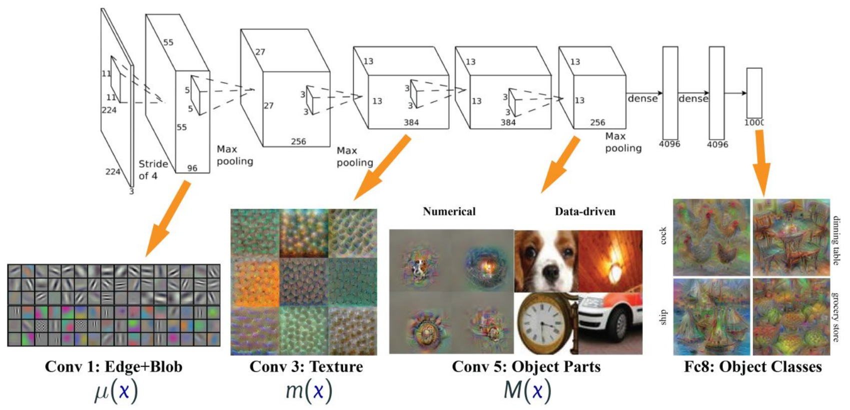


Figure 1.6: Visualization of CNN features at different scales, from (Wei et al., 2017). Today, researchers scale up the training of convolutional neural networks to billions of images. Modern models involve thousands of convolution filters and are usually described using concise block diagrams. As discussed for instance in (Olah et al., 2017), the features associated to these filters can be visualized and still roughly fit within the multiscale framework of Figure 1.4. Please note, however, that generating "good-looking" visualizations of CNN features requires a fair amount of specific regularization tricks. These appealing images allow researchers to get an intuition of their models, but are *not* faithful reflections of their inner structures.

1.1.2 Strengths and limitations of convolutional neural networks

The first convincing model for texture. In computer vision, data-driven CNN features have now replaced hand-made descriptors for object detection and recognition. In biomedical imaging, as illustrated Figure 1.9, segmentation networks outperform traditional methods in most settings and open a whole new range of applications. Most impressively, as illustrated in Figures 1.7 and 1.8, CNNs can advantageously replace wavelet transforms for many processing tasks – including the synthesis of texture.

A major culture shift. An essential contribution of deep learning research is the new-found emphasis put on **cross-field interactions** and **expert-labeled datasets**, as tedious manual computations (gradients, explicit filter coefficients, etc.) get abstracted away progressively. Applying the same blueprint to medical fields, we can hope to combine four types of expertise:

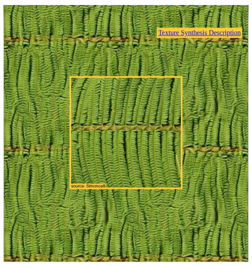
1. **Medical doctors** provide valuable input through well-curated *datasets*.
2. **Statisticians** select *data-driven* values for the parameters of a given pipeline.
3. **Software engineers** implement the building blocks of research codes on Graphics Processing Units (GPUs) to leverage the power of *modern hardware*.
4. **Mathematicians and computer scientists** hard-code their insights on the problems to solve in the *architectures* of their programs. This modelling work allows algorithms to generalize well even in high-dimensional settings.

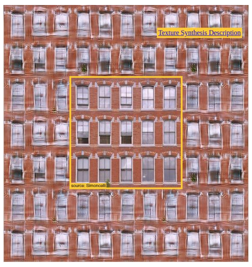
Artificial intelligence? The bio-inspired vocabulary which is prevailing in our field comes from the strong historical ties between research on convolutional "neural networks" and actual neurosciences (Fukushima, 1980). The design of CNNs is partly motivated by biological insights on the structure of the visual cortex (Hubel and Wiesel, 1962, 1968), and many researchers still dream of emulating, one day, full human brains on silicon chips. We should refrain, however, from attributing human-like qualities to algorithms which are little more than finely tuned compositions of filtering operations.

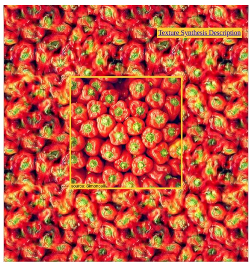
From a computational perspective, convolutional networks are closely related to classic algorithms, such as the *fast wavelet transform* that powers the JPEG-2000 compression standard for digital cinema (Skodras et al., 2001). The pyramidal structures of CNNs allow researchers to make sure their images are *not* processed as generic vectors of dimension 512x256 (Ulyanov et al., 2018). But expecting high-level behaviours to "emerge" out of the simple algorithms in use today would be unreasonable: stacking convolution layers on top of each other has not paved the way towards *general* intelligence.

Limitations of convolution-based algorithms. Image processing and computer vision have gone a long way since the first works on Laplacian pyramids (Burt and Adelson, 1983) and SIFT descriptors (Lowe, 1999). Ultimately though, even after an expensive tuning of their coefficients, CNNs remain heavily biased towards **texture analysis** and **pattern detection**. As illustrated Figure 1.10, state-of-the-art models for image classification still roughly behave as advanced *bag-of-features* models (Sivic and Zisserman, 2003; Nowak et al., 2006) and pay little attention to the *shapes* that structure input images.

Realistically, to tackle the next generation of open problems in medical imaging, researchers are thus going to need *new geometric ideas* alongside compute power and curated datasets.


(a) Leaves.


(b) Windows.


(c) Peppers.

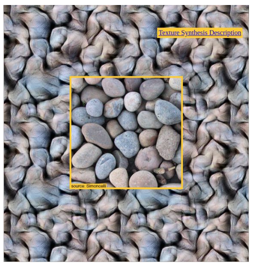
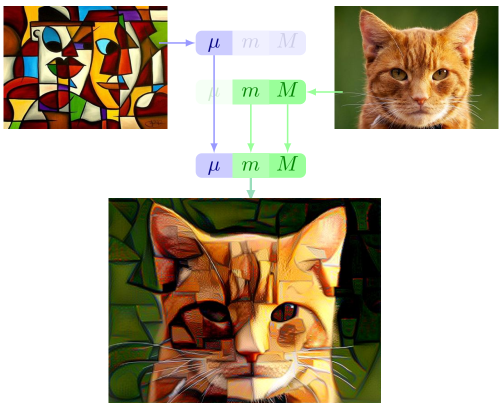
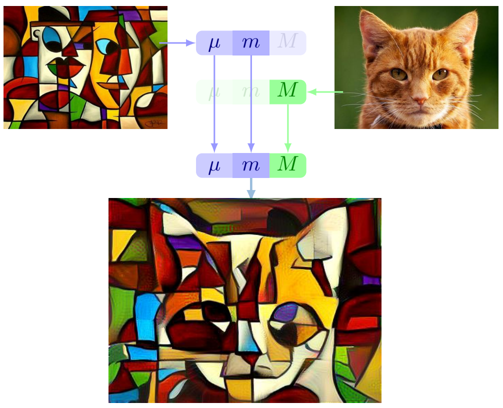

(d) Pebbles.

Figure 1.7: Texture synthesis with wavelets, from `www.cns.nyu.edu/~lcv/texture`. In a landmark paper, (Portilla and Simoncelli, 2000) proposed to compute *texture signatures* as statistical correlation scores between wavelet feature maps. Starting from a reference image (center squares), we can compute its "texture id" and optimize a noisy background (border) to make it fit this prescribed signature. The process allows researchers to generate textured wall-papers using nothing but explicit convolution filters. This example does not involve any kind of "learning" procedure, and illustrates the **close historical connection between convolutional architectures and the processing of texture**.



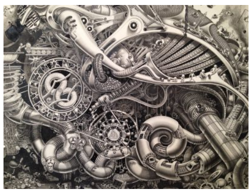
(a) Style signature computed from "micro" layers.



(b) With "micro" and "meso" (deeper) layers.

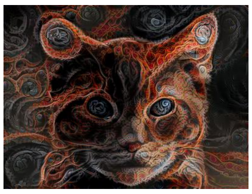
(c) Style 1.

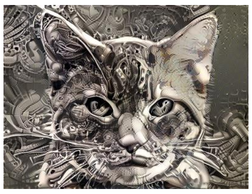
(d) Style 2.


(e) Style 3.

(f) Style 4.

(g) Cat 1.


(h) Cat 2.


(i) Cat 3.

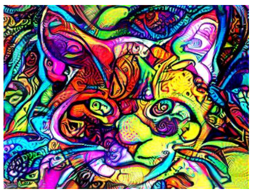
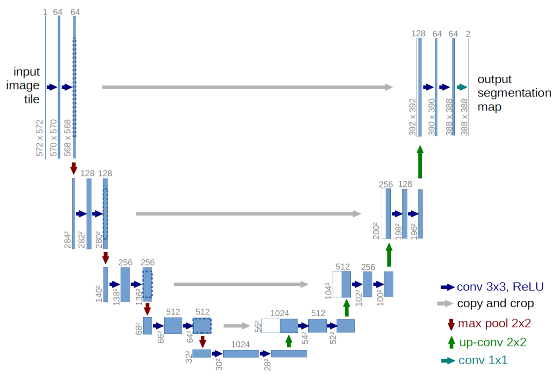
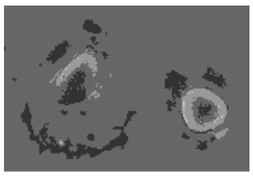

(j) Cat 4.

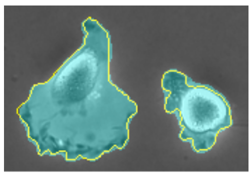
Figure 1.8: The deep art algorithm adapts the Portilla-Simoncelli method to the deep learning era (Gatys et al., 2016). All cat and style images come from (Nikulin and Novak, 2016). (a-b) A well-trained CNN provides high-quality feature maps. Starting from a painting and a photo, we can thus create a synthetic signature that encodes the same "style" as the former and the same "content" as the latter. Optimizing by gradient descent an image to make it fit the prescribed signature, we retrieve a good-looking synthetic "painting". (c-j) This algorithm generates impressive results, which played a great part in the development of a media bubble around "artificial intelligence" from 2015 onwards.



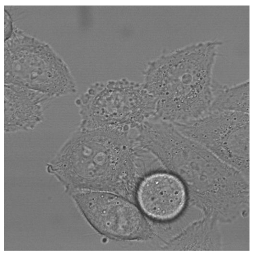
(a) Architecture.



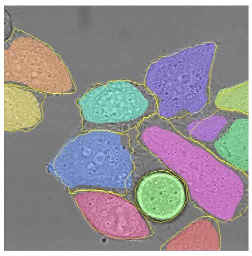
(b) Input.



(c) Output.



(d) Input.



(e) Output.

Figure 1.9: The U-net segmentation network (Ronneberger et al., 2015). (a) The fine-to-coarse-to-fine architecture of the U-net allows this model to compute and re-use feature maps at all scales. It can be trained to segment biomedical slices from pairs of raw images and expert-labelled color masks: here, glioblastoma-astrocytoma brain cancer cells (b-c) and HeLa cells (d-e). In practice, this model works off-the-shelf in all settings where the objects to segment have a texture which is distinct from that of the background. As far as biomedical imaging is concerned, this application of CNNs to segmentation tasks is, without a shadow of a doubt, the **most significant advance of the last decade**.

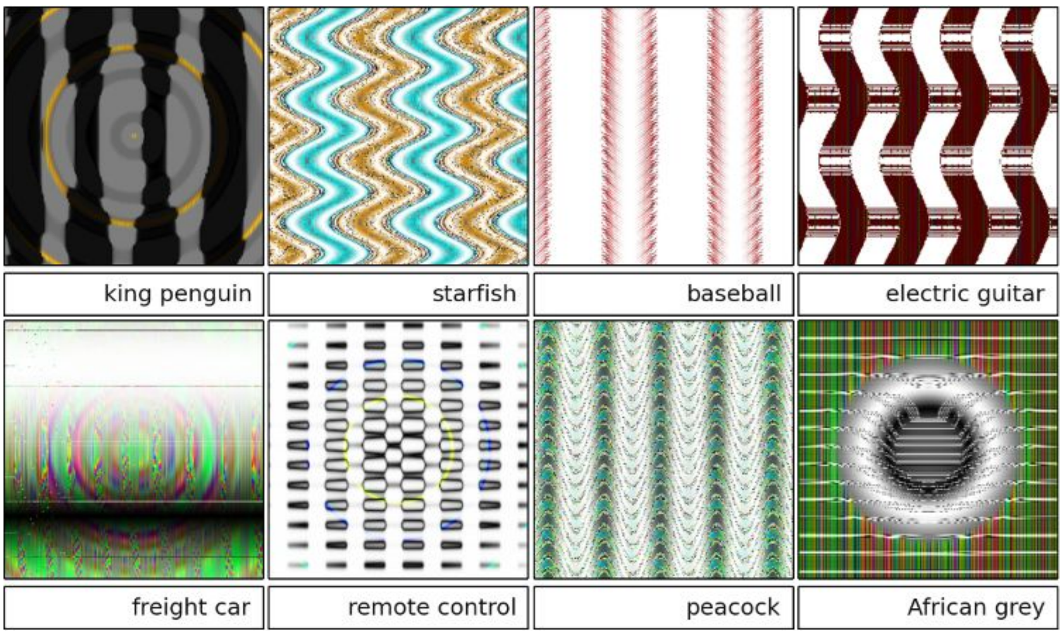
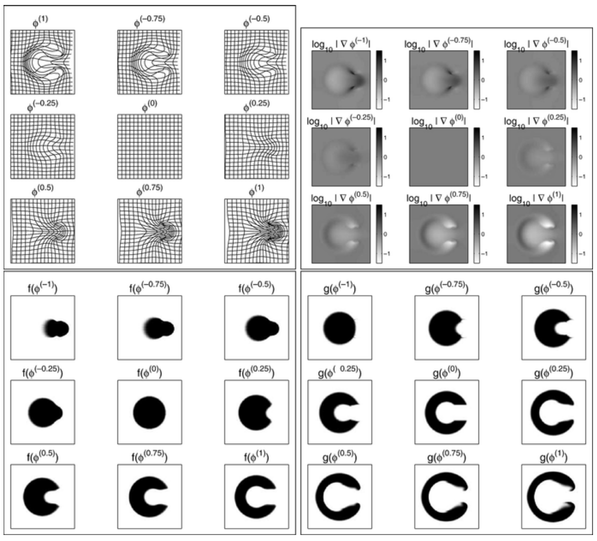
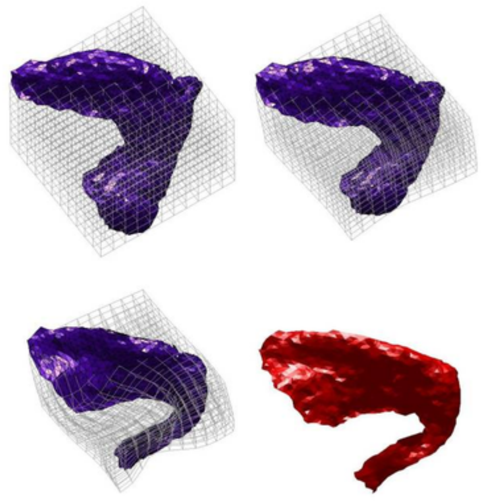


Figure 1.10: Images mis-classified as "king penguin", "starfish" and other standard image classes by a well-trained CNN, from (Nguyen et al., 2015). The multiscale, convolution-based architecture of a CNN is ideally suited to the processing of texture but is **not relevant to geometry or shape analysis**. As a consequence, modern image classification algorithms tend to rely on the detection of patterns and can be fooled with meaningless images known as *adversarial examples*.
Understanding this phenomenon precisely is a major open problem in computer vision and image processing. As for us, we interpret this "texture bias" as a fundamental limitation of convolution-based networks, that **motivates the study of other algorithmic structures**.



(a) Morphing an image.



(b) Morphing a surface mesh.

Figure 1.11: Deformations are easy to encode with point clouds (Ashburner, 2007; Glaunes, 2005). (a) Implementing the free-form deformation of a bitmap image comes with many caveats: combining data-driven insights with an accurate advection scheme is no mean feat. (b) In sharp contrast, deformations of point clouds or surface meshes can be implemented using **simple operations on coordinates**. This encoding is most convenient for shape analysis and is discussed throughout this thesis.

1.1.3 Working with point clouds, meshes and curves

Implicit vs explicit coordinates. Unfortunately, doing geometry with "bitmap" images is relatively cumbersome: simple operations such as rotation and scaling rely on ad hoc interpolation schemes. Going further, free-form deformation or *advection* routines must be implemented with care: naive schemes may quickly destroy the topology of a picture or introduce interpolation artifacts (Staniforth and Côté, 1991; Beg et al., 2005). As illustrated Figure 1.11, encoding anatomic data with explicit point clouds is a sensible choice: geometry is easier to study with *Langrangian* than with *Eulerian* coordinates.

A somewhat neglected area. The large-scale study of anatomical shapes should be tractable: *Graphics* Processing Units were initially designed for the processing of surface meshes, and the support of convolutional architectures only came as an afterthought to hardware constructors. As showcased on a daily basis by modern video games, efficient geometry management has been on the cards since the first "Toy Stories". Surprisingly though, throughout the last decade, geometric problems have been relatively neglected by the deep learning community.

A reason for this is the focus of engineering curriculums on scripting languages: most researchers are not familiar with the intricacies of low-level GPU programming, and are unable to *develop their own tools*. Naturally, the literature thus leans towards problems that can be tackled with off-the-shelf libraries whose developers – Google and Facebook – have always prioritized imaging and natural language processing to the detriment of other fields.

This picture is evolving slowly. A bit confusingly though, modern works on *geometric deep learning* (Bronstein et al., 2017) focus on the extraction of features on *graphs* and often ignore the challenges encountered in the processing of anatomical data: the questions of robustness to re-meshing, topological noise and heterogeneous sampling, the generation of realistic deformations, etc. As of 2020, the field is not yet mature enough to fulfill its potential in medical imaging: addressing these concerns will be a necessary first step.

1.2 Geometric data analysis

Shapes are not vectors. Coming from data sciences and statistics, the main specificity of shape analysis is the absence of the usual algebraic operations "+" and "×". Taking the *sum* of two brains is essentially meaningless, and no canonical heart model can truly replace the neutral element "0" of a vector space.

A geometric theory for geometric data. Fortunately, however, *distances* can still be defined: saying that two skulls are "close" or "far away" from each other could definitely make sense. A primary problem of interest in computational anatomy is thus to define **metric structures on spaces of shapes** which are:

1. **Anatomically relevant**, and make sense from a medical perspective.
2. **Mathematically principled**, to enable robust analyses downstream.
3. **Computationally affordable**, scaling up do real-life 3D data.

1.2.1 Kendall's sphere of triangles

Procrustes analysis. The simplest of all such metrics is the one that deals with *labeled* polygons defined *up to a similarity*. If $x = (x_1, x_2, \dots, x_\mathrm{N})$ and $y = (y_1, y_2, \dots, y_\mathrm{N})$ in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ are two point clouds which are non-degenerate (i.e. not collapsed to a single location), we say that x and y have the same *shape* in the sense of Procrustes if there exists a translation vector $\tau \in \mathbb{R}^\mathrm{D}$, a rotation matrix $\mathrm{R} \in \mathrm{O}(\mathrm{D}) \subset \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$ and a scaling coefficient $\lambda > 0$ such that:

$$y \;=\; \lambda \cdot \mathrm{R}x + \tau\,, \qquad \text{i.e.} \qquad \forall\, i \in [\![1,\mathrm{N}]\!]\,,\; y_i \;=\; \lambda \cdot \mathrm{R}x_i + \tau\,. \tag{1.2}$$

We then define the full **Procrustean distance** between two non-degenerate point clouds as:

$$\mathrm{d}_{\text{Procrustes}}(x,y) \overset{\text{def.}}{=} \frac{\min_{\lambda,\mathrm{R},\tau} \|\lambda\mathrm{R}x + \tau - y\|_2}{\text{Scale}(y)} \overset{\text{def.}}{=} \frac{\min_{\lambda,\mathrm{R},\tau} \sqrt{\sum_{i=1}^{\mathrm{N}} \|\lambda\mathrm{R}x_i + \tau - y_i\|_2^2}}{\sqrt{\sum_{i=1}^{\mathrm{N}} \|y_i - \overline{y}\|_2^2}}\,, \tag{1.3}$$

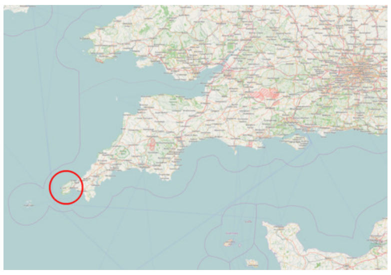
where $\overline{y} = \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}} y_i$ is the mean value of y in $\mathbb{R}^\mathrm{D}$. This geometry is associated to the simple notion of *rigid alignment* or *registration*, a common pre-processing in imaging sciences. It is symmetric, satisfies the triangle inequality and vanishes to zero if and only if x and y have the same shape. Going further, if x' and y' can be put in correspondance with x and y using similarities, then:

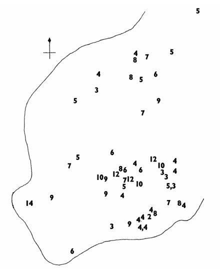
$$\mathrm{d}_{\text{Procrustes}}(x',y') = \mathrm{d}_{\text{Procrustes}}(x,y)\,. \tag{1.4}$$

The sphere of triangles. This metric endows the *quotient* space of Procrustean shapes with a remarkable structure (Kendall, 1977, 1984, 1989). A first example is that of triangles in the Euclidean plane ($\mathrm{N} = 3$ and $\mathrm{D} = 2$): there exists an explicit isometry f between the space of non-degenerate triangles and a sphere of radius 1/2 in $\mathbb{R}^3$, with:

$$\forall\, x,y \in \mathbb{R}^{3\times 2}\setminus\{(p,p,p),\; p\in\mathbb{R}^2\}, \qquad \mathrm{d}_{\text{Procrustes}}(x,y) \;=\; \|f(x) - f(y)\|_{\mathbb{R}^3}\,. \tag{1.5}$$

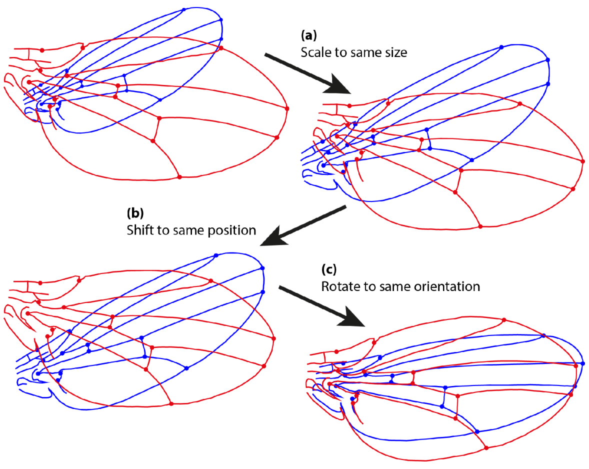
As illustrated from Figure 1.12 to 1.15, this canonical embedding makes sense from topological, geometric *and* statistical perspectives. The theory can then be generalized to large point clouds in $\mathbb{R}^2$ or $\mathbb{R}^3$ and has become a staple of biomedical imaging (Dryden and Mardia, 1998).


(a) Cornwall.


(b) Land's End.

(c) Menhirs.

Figure 1.12: Simulating the ley hunter (Broadbent, 1980; Kendall and Kendall, 1980). The first motivation for the development of the modern theory of shapes laid, quite remarkably, with archeology. (a) At the western-most tip of Great Britain, in Cornwall, lies the peninsula of "Land's End". (c) A famous tourist attraction in the UK, this 10km-by-15km piece of land is home to a considerable collection of megaliths. (b) Looking at their coordinates on a map, archeologists soon started to wonder if these standing stones had been placed "at random" by pre-historic Britons, or if we could try to read more into alignments of stones – *ley lines* – that evoked, to some enthusiasts, a cosmic compass of sorts.
To provide an answer to this question, Simon Broadbent proposed to consider all triplets of menhirs on the map. He could then compare this set of triangle shapes with a "random" uniform distribution, and detect (or not) an abnormally large number of quasi-flat triangles: structured alignments. Overall, this research did not truly settle the debate in archeology… but had a significant legacy in biomedical imaging, which is fine too!



(a) Procrustes analysis.

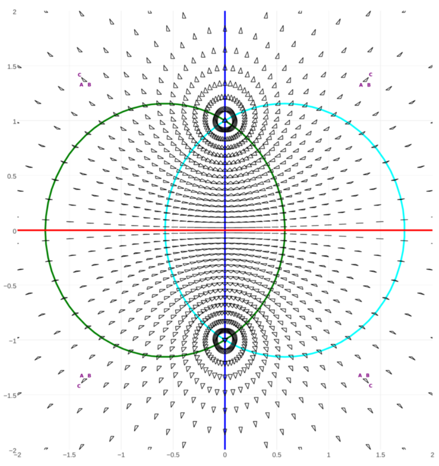
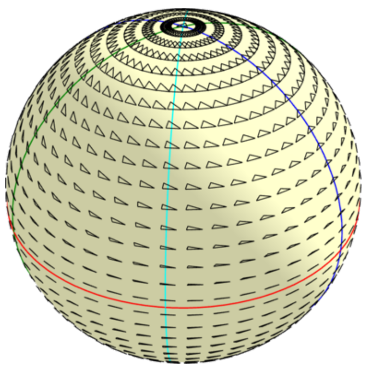
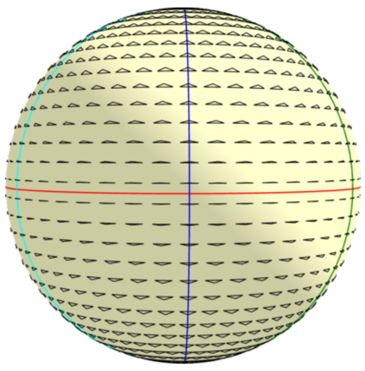

(b) The plane of triangles.

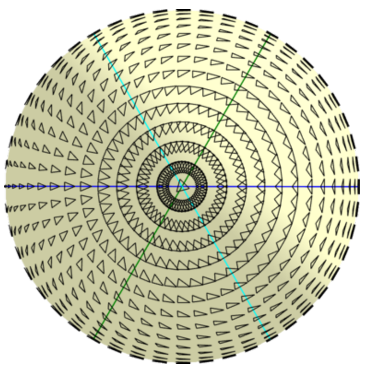
Figure 1.13: Working with shapes defined "up to similarities" in the 2D plane.
(a) As showcased in this illustration from (Klingenberg, 2015), *Procrustes analysis* is all about quotienting out the degrees of freedom associated to scale, position and orientation. There are 4 of them in dimension D = 2, and 7 in dimension D = 3.
(b) The space of triangles ABC in the plane can be identified with $\mathbb{C}^3 \simeq \mathbb{R}^6$. If we remove the degenerate triangle ($A = B = C$) and identify with each other all triangles that can be put in correspondence using a similarity (translation + rotation + scaling), we end up with the two-dimensional space of **triangle shapes**. In this simple chart, we use small glyphs to display, at every location $z \in \mathbb{C}$, the triangle ($A = -1, B = 1, C = z$). This allows us to represent exactly one specimen for every shape of triangle, bar the special case of the semi-degenerate triangle shape "$A = B \neq C$", which is rejected at infinity. We highlight in red the set of flat triangles ($\text{Im}(z) = 0$); in regal blue, the set of triangles which are isosceles at vertex C ($\text{Re}(z) = 0$); in cyan, the set of triangles which are isosceles at vertex B ($|z - 1| = 2$); in green, the set of triangles which are isosceles at vertex A ($|z + 1| = 2$).



(a) Global view.

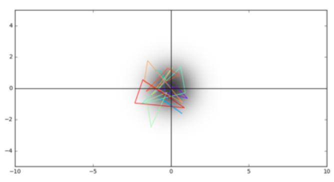


(b) From the equator.

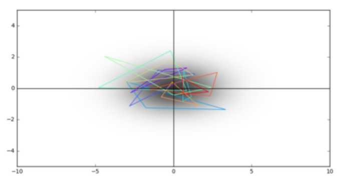


(c) From the North pole.

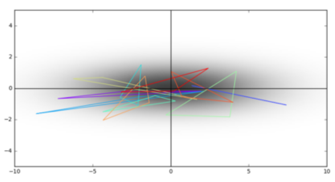
Figure 1.14: Kendall's sphere of triangles. The set of triangle shapes, charted Figure 1.13.b, can be identified with a **sphere of radius** $1/2$. As displayed here, there exists an explicit correspondence between the space of triangle shapes "up to similarities" and the sphere $\mathbb{S}^2(0, 1/2) \subset \mathbb{R}^3$, with at least three important theoretical properties.
First, all the symmetries of the space of triangles have natural geometric counterparts. The "North" and "South" poles correspond to the direct and indirect equilateral triangles, respectively; the set of flat triangles is identified with the equator (in red); the three sets of isosceles triangles correspond to three equidistant meridians (in blue, cyan, green). These four great circles cut the sphere in 12 "identical" domains, which can be identified with each other under the action of the 6 permutations of the vertices A, B, C, plus the reflexion with respect to any straight line in the plane.
Second, as detailed Eq. (1.5), the chord distance on the sphere (embedded in $\mathbb{R}^3$) corresponds to the natural quotient of the Euclidean metric on $\mathbb{C}^3 \simeq \mathbb{R}^6$ under the action of the similarity group.



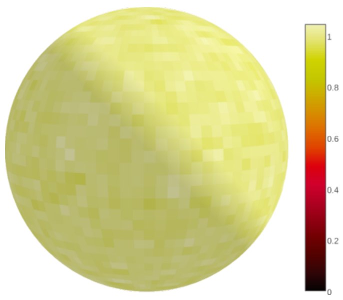
(a) Isotropic Gaussian.



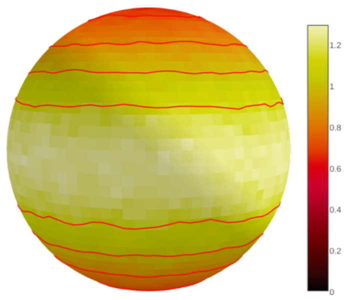
(c) Slightly anisotropic.



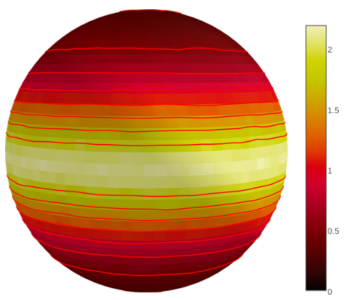
(e) Very anisotropic.



(b) Isotropic Gaussian.



(d) Slightly anisotropic.



(f) Very anisotropic.

Figure 1.15: Canonical distributions on Kendall's sphere of triangles. Finally, and perhaps most surprisingly, this canonical representation of the **shape space of triangles** also has remarkable statistical properties. (a) When the three vertices A, B, C are drawn independently according to an isotropic Gaussian law, the shape of the triangle ABC follows a **uniform law on the sphere**. (b) In other words, if the draw $3 \cdot 10$k vertices according to a normal Gaussian law, the corresponding empirical histogram on the sphere has constant density up to statistical fluctuations: each bin gathers a number of triangles ABC that is roughly proportional to its area. (c-d) If we draw the vertices A, B, C according to an *anisotropic* Gaussian law, the corresponding density on the sphere still retains some structure: it is an explicit function of the distance to the equator. (e-f) With vertices drawn according to a Gaussian bell that becomes more and more degenerate, the law of our random triangle shape converges towards a uniform distribution on the equator of flat triangles. Plot rendered with `Plot.ly` (Inc., 2015).

1.2.2 Endowing spaces of shapes with a metric structure

The ideal theory of triangles acts as a strong motivation for mathematical research in the field. The dream pursued by the "shapes" community over the last thirty years has been to define distances on spaces of brains or bones that could be as elegant as the Procrustes metric, but also relevant from a *medical* perspective (Grenander and Miller, 1998).

Procrustes analysis. How do researchers proceed? First of all, we notice that thanks to its strong algebraic structure, Kendall's theory can be interfaced with standard statistical tools (Jayasumana et al., 2013; Ben Tanfous et al., 2018) or extended to spaces of continuous, parameterization-free curves (Srivastava and Klassen, 2016). As illustrated Figure 1.16, performing dictionary learning "up to similarities" is now perfectly do-able (Song et al., 2019). Crucially though, Procrustes analysis remains focused on position, orientation and scale: in most real-life settings, we must go further.

Linear deformation models. When shape data is *labeled* and *concentrated* around a mean template shape, a good baseline is provided by linear deformation models. If $x^1, \dots, x^\mathrm{P}$ is a collection of pre-registered point clouds in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, we may compute an explicit average template $x^* = \frac{1}{\mathrm{P}}\sum_{k=1}^{\mathrm{P}} x^k$ and identify each subject with the deviation to the mean $v^k = x^k - x^*$, a vector field in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ supported by the vertices of x^*. We can then compare subjects with each other using a Euclidean metric in the mould of:

$$\mathrm{d}_{x^*}(x^i, x^j) \overset{\text{def.}}{=} \|v^i - v^j\|_{x^*} \overset{\text{def.}}{=} \sqrt{(v^i - v^j)^\top \mathrm{K}_{x^*}^{-1}(v^i - v^j)}\,, \tag{1.6}$$

where K_{x^*} is some relevant symmetric and positive N-by-N or ND-by-ND matrix.

As discussed Section 5.1.2, a standard choice for the *cometric* K_{x^*} is the *Thin Plate Spline* (TPS) kernel matrix associated to the template x^* (Bookstein, 1991): it penalizes tearings and is fully invariant to affine changes of coordinates. Alternatively, letting K_{x^*} be equal to (a robust modification of) the covariance matrix of the v^i's in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is also a sensible choice (Mahalanobis, 1936). This method was discussed as early as (Mahalanobis, 1925) for anthropometric population studies, and is now commonly understood through the normalized "PCA" space associated to a *Principal Component Analysis* of the population.

Main challenges in the field. As illustrated Figure 1.17, the "Procrustes + Splines/PCA" toolbox is good enough for simple population studies in biology. In practice, research in computational anatomy has thus focused on providing answers to the two following problems:

1. **Parameterization invariance.** Can we define metrics between *unlabeled* point clouds? Robust algorithms should process surface meshes and volumes without ever being biased by local variations of the sampling density: defining landmark-free metrics on spaces of shapes is a priority.

2. **Non-linear deformations.** Linear PCA-like models tend to break down as soon as datasets become a bit too wide. If diverse poses or classes are present in a population, topological properties may stop being preserved and a large number of PCA modes become required to describe the data. Consequently, we strive to define metrics and models that can deal with *large deformations* while preserving a low intrinsic dimensionality.

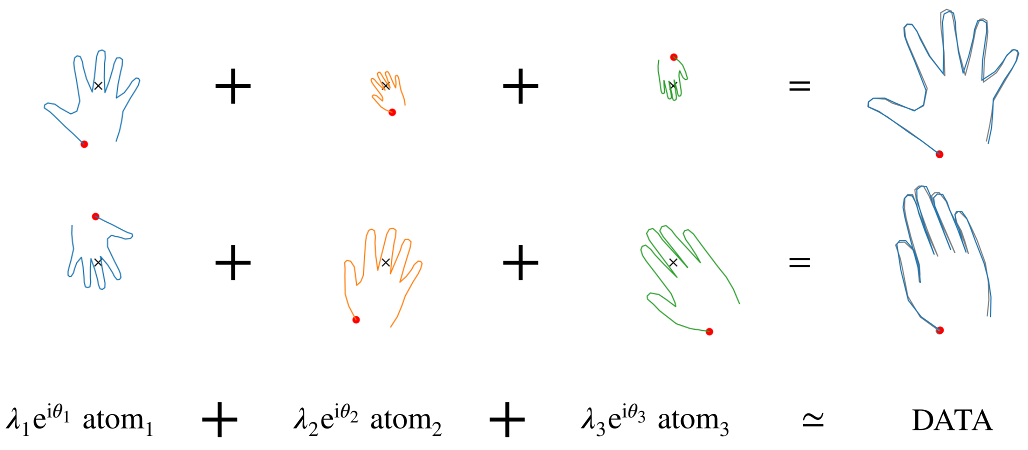
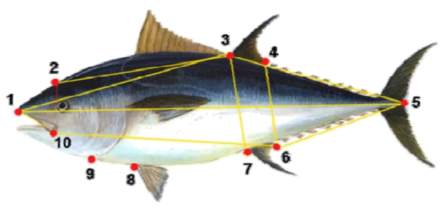
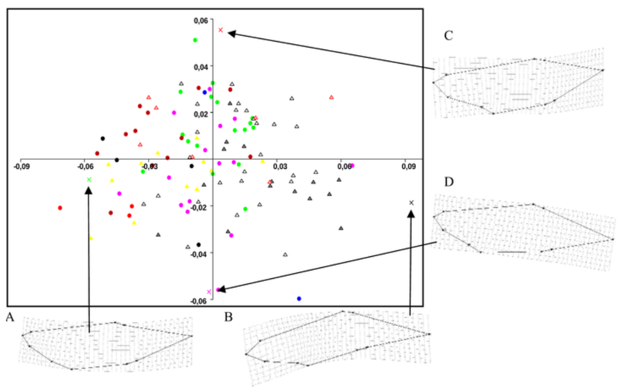

+
=
$\lambda_1 e^{i\theta_1}$ atom$_1$
$\lambda_2 e^{i\theta_2}$ atom$_2$
$\lambda_3 e^{i\theta_3}$ atom$_3$
$\simeq$
DATA

Figure 1.16: Dictionary learning in Kendall's shape space, from (Song et al., 2019). In the 2D plane, landmarks can be represented as complex numbers; similarities are then simply given through affine transformations $z \mapsto \lambda e^{i\theta} z + \tau$. (Song et al., 2019) takes advantage of this algebraic structure to extend the common concept of dictionary to the non-linear space of Procustes shapes: dictionary learning with a Procustes loss can be tackled using standard matching pursuit algorithms (Mairal et al., 2014).
This example is more or less at the limit of what quasi-linear models can handle: research on *non-linear* deformations is motivated by the need to process datasets with a wide variety of modes or poses… as encountered for instance with 3D scans of the hand (Von-Tycowicz et al., 2015)!



(a) Landmarks.



(b) Standard PCA with TPS.

Figure 1.17: A typical morphometric study, from (Addis et al., 2010). The *Thin Plate Splines* framework for shape analysis, discussed at length in (Bookstein, 1991), is now part of the standard toolbox in biomedical imaging. (a) In favourable cases, well-identified anatomical landmarks can be located on the input shapes: they are encoded as point clouds in $\mathbb{R}^{N\times 2}$ or $\mathbb{R}^{N\times 3}$. We then strive to define **relevant metric structures** on these ad hoc parameters.
(b) Linear deformation models are usually called *splines* and parameterized by a *kernel* function. The Thin Plate Spline (TPS) kernel is a canonical choice in 2D and 3D, which penalizes tearings and makes the processing invariant to affine deformations. Off-the-shelf `R` and `Python` toolboxes now allow practitioners to perform Principal Component Analyses (PCA) with strong guarantees, and identify the main modes of variation of their datasets. In this specific instance, a significant discrepancy between tunas fished in 2008 (circles) and 2009 (triangles).

1.2.3 Statistics without a "+"

Answers to these two questions are discussed throughout this work. As detailed for instance in (Pennec, 2008), this research is primarily motivated by the needs of statisticians: a high-quality metric $\mathrm{d}(x, y)$ on a shape space $\mathcal{S}$ can act as a substitute for a vector structure $(0, +, \times)$, paving the way for anatomical population studies.

Fréchet means. To see this, we first come back to the definition of the mean as a solution of a least-square problem (Legendre, 1805; Gauss, 1809). If $x^1, \dots, x^\mathrm{N}$ is a collection of shapes, we define its Fréchet mean of order p through:

$$x^* \;\in\; \arg\min_{x\in\mathcal{S}} \; \tfrac{1}{\mathrm{N}} \textstyle\sum_{i=1}^{\mathrm{N}} \mathrm{d}(x, x^i)^p \,. \tag{1.7}$$

The usual *mean* and *median* respectively correspond to the cases where "$p = 2$" and "$p = 1$" (Fletcher et al., 2008)... But beware: in general, such minimizers may not be unique or even well-defined – we refer to (Charlier, 2011) for an overview.

Geodesics. Going further, geodesic curves are defined as continuous paths $\gamma : t \in [0, 1] \mapsto \gamma(t) \in \mathcal{S}$ that minimize length locally – but not necessarily between their end points, due to curvature and non-uniqueness. Simply put: geodesics are **generalized straight lines**.

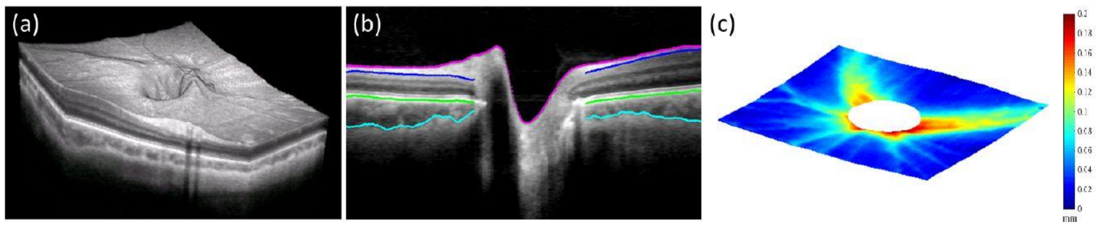
Assuming that $(\mathcal{S}, \mathrm{d})$ is a *Riemannian manifold* – i.e. that it is locally equivalent to a Euclidean vector space, just like a sphere is locally equivalent to its tangent planes – we can then show that all geodesics satisfy an ordinary differential equation of order 2 (Lee, 2006). As discussed in detail Section 5.2.1, this implies that a geodesic is entirely described by its value and derivative at time "$t = 0$": its initial position and velocity, as in Newtonian mechanics.

Geodesic regression, longitudinal models. Identifying geodesics with straight curves, we can then generalize linear regression to spaces of shapes (Fletcher, 2011). Going further, we can parameterize the geodesic segment γ^i between a template x^* and a subject x^i through its initial velocity $v^i = \frac{\mathrm{d}}{\mathrm{d}t}\gamma^i(t = 0)$. This allows us to perform statistical studies such as PCA in the tangent space $\mathrm{T}_{x^*}\mathcal{S}$ to the shape space $\mathcal{S}$ at location x^*, a well-understood vector space of *latent codes* for our dataset (Fletcher et al., 2004; Sommer et al., 2010).

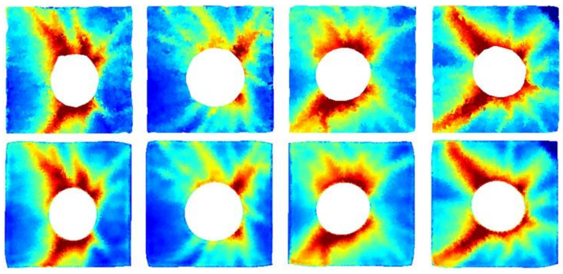
A typical morphometric study is showcased Figure 1.18. Note that in many clinical settings, the problem of interest is to compare the *trajectory* of a patient with the global trend of its group. Consequently, as illustrated Figure 1.19, the study of time-dependent longitudinal models on a Riemannian manifold is bound to become an important topic: we refer to (Durrleman et al., 2013; Schiratti et al., 2015; Koval et al., 2017) for an introduction.

Interfacing geometry with statistics. Overall, the field of computational anatomy is now pretty well structured. As *geometers*, our main job is to provide *statisticians* with reliable and effective metrics on spaces of shapes: at a mild theoretical cost, anatomical data can then be put in the same framework as other physiological measurements.

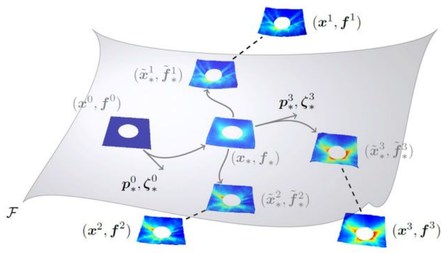
We must stress that in this context, **robustness and reproducibility trump accuracy**. If our work is to bring real value to clinical applications, it should uphold the same standards of reliability as any commercial drug or medical device. Acknowledging our limitations as far as clinical trials are concerned, we see our field as a team effort that must involve engineers, statisticians and medical doctors on an equal footing. To enable these interactions, interpretable pipelines are thus preferable to end-to-end blackboxes.



(a) Optical Coherence Tomography (Huang et al., 1991) is a common imaging technique for ophthalmology. By measuring light interferences using a sort of "optical echograph", medical doctors can now acquire 3D images of the retina at a micrometric resolution.

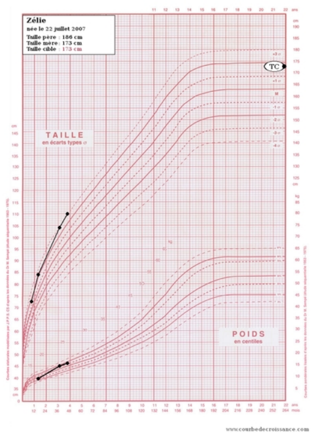


(b) Typical thickness maps from an OCT dataset.

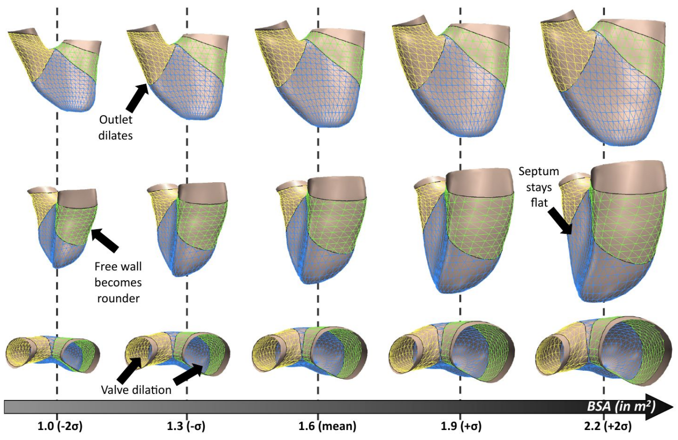


(c) An explicit manifold model.

Figure 1.18: An application of metamorphoses to the detection of glaucoma (Lee et al., 2017). (a) Modern imaging devices allow us to acquire micrometric thickness maps of a patient's retina in a neighborhood of the optic nerve. (b) Glaucoma is a widespread disease, associated to a slow deterioration of the retina. It can be diagnosed by the observation of a thinning of the tissue along the main blood vessels of the ocular globe, following a pattern that varies with the vascularization of each patient. (c) To quotient-out this "innate" anatomic variability and perform a meaningful population study, we can decompose the variability of a dataset into **geometric** and **functional** deviations to a mean template. In practice, this processing allows us to perform population studies in a **robust** and **interpretable** linear space of "latent codes", identified with a tangent space to the manifold of retinas.



(a) Scalar data.



(b) Shape data, from (Mansi et al., 2011).

Figure 1.19: Computational anatomy brings together statistics and shape analysis.
(a) Throughout the XX$^{\text{th}}$ century, statisticians have developed principled methods to process scalar and vector data. Say, the height and weight of a child, tracked over the years and compared with an expected growth curve. (b) Computational anatomy is all about **generalizing this approach to geometric data**, which does not come with a canonical vector structure. As illustrated Figure 1.18, a typical strategy is to endow spaces of shapes with a non-linear – *Riemannian* – metric and reformulate standard statistical tools using nothing but geometric primitives. We can then find significant correlations between anatomical features: here, the Body Surface Area (BSA) and the shape of the right ventricle, in a population study focused on the tetralogy of Fallot (Mansi et al., 2011).

1.3 Outline of this thesis

Shape analysis and registration. Throughout the last twenty years, a thriving community has gathered around the idea that *shape metrics* can be defined in relation to *registration algorithms*. If $\mathrm{D} = 2$ or 3 and $\varphi : \mathbb{R}^\mathrm{D} \rightarrow \mathbb{R}^\mathrm{D}$ is a mapping that turns a source shape A into a target $\varphi(A) = B$, the discrepancy between φ and the identity mapping $\mathrm{Id} : x \in \mathbb{R}^\mathrm{D} \mapsto x \in \mathbb{R}^\mathrm{D}$ can be used as a tractable shape distance $\mathrm{d}(A, B)$. Conversely, plausible deformations φ can be understood as *geodesics* that gradually turn $\mathrm{Id}(A) = A$ into $\varphi(A) = B$ by following, in some sense, a least-effort trajectory in a space of deformations.

As discussed Chapter 5, standard shape models either rely on elastic meshes for biomedical simulation (Montagnat et al., 2001; Srivastava et al., 2010) and computer graphics (Kilian et al., 2007; Von-Tycowicz et al., 2015) or on advanced imaging techniques. In the fields of cardiac- and neuro-anatomy, standard baselines are provided by the B-splines (Rueckert et al., 1999) and diffeomorphic SVF (Arsigny et al., 2006) or LDDMM (Beg et al., 2005) frameworks.

A reference textbook on the subject has recently been compiled, with applications to statistics (Pennec et al., 2019): we recommend this well-curated collection of tutorials for background on the field.

A pivotal moment. Crucially though, as of 2020, the community seems to have reached the limits of what can be achieved using nothing but explicit equations and `Matlab` or C++ codebases. If we are to perform large-scale population studies on modern datasets, such as the long-awaited UK Biobank (Ollier et al., 2005), we must resolve the following bottlenecks:

1. **Speed.** Using iterative algorithms to register volumetric images takes minutes or at best, seconds (Brunn et al., 2019). The impressive `Dartel` (Ashburner, 2007) and `Ants` (Avants et al., 2009) packages have become standard pre-processing tools in neurology… but if we are to use them as atomic sub-routines in higher-level pipelines, we should speed up their runtimes by at least two or three orders of magnitude.

2. **Anatomical relevance.** Today, few models can *extrapolate* outside of training datasets in a way that is meaningful: splines, SVF or LDDMM geodesics rely on simple regularization priors that do not make much sense from a medical perspective. Meanwhile, biomechanical models rely on clean meshes that are notoriously hard to combine with noisy clinical data. Improving our geometric models would have a major impact on statistical analyses downstream – but this is easier said than done!

Going beyond explicit models. To bring an answer to these questions, researchers have mostly focused on multiscale (Durrleman et al., 2014) and low-frequency implementations (Zhang and Fletcher, 2019) of standard algorithms, implicit regularization priors (Arguillere et al., 2016; Gris et al., 2018) and hybrid CNN-SVF architectures (Yang et al., 2017; Krebs et al., 2019; Balakrishnan et al., 2019; Shen et al., 2019).

In favourable settings such as neuroanatomy, computing sensible deformations in fractions of a second now seems within reach. Unfortunately though, no fully satisfying answer has yet been provided to the problem of anatomical relevance: the quest for **data-driven yet robust** shape metrics remains the Holy Grail for researchers in the field.

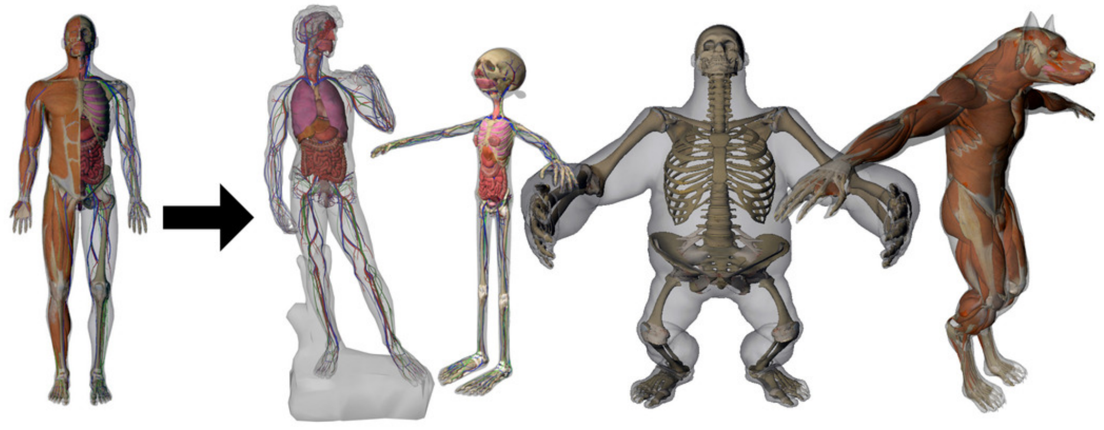


Figure 1.20: Atlas registration or anatomy transfer, from (Ali-Hamadi et al., 2013). A task of interest in medical imaging is to transfer knowledge from a high-quality reference atlas (left) to the body silhouettes of patients (right) – statues and cartoon characters, in this specific occurence! Putting patients in a reference frame of coordinates is a necessary first step for population studies in neurology (Klein et al., 2009a). It can also allow researchers to build meaningful representations out of a collection of raw segmentation masks.
Throughout this work, we discuss the theory that underlies state-of-the-art registration methods in neuro-anatomy. The main ambition of this thesis is to provide algorithmic and theoretical tools for the next generation of **data-driven deformation models**, without ever compromising on **robustness** and **interpretability**.

1.3.1 An accessible introduction to the field.

Written under the supervision of Alain Trouvé, this thesis presents algorithmic and theoretical tools that ease the development of shape models. Combining deep learning methods with registration algorithms pushes existing frameworks to the limit, and revisiting the foundations of computational anatomy is a necessary first step.

Keeping it simple. Clarity is our priority: this manuscript should read as an introduction to the field for a new generation of students. Back in the 90's, pioneers of medical imaging tended to have a background in electrical engineering or mathematics… But the demographics of the community is evolving fast. Having graduated in the deep learning era, a majority of students now come into the field with a skillset focused on data sciences and computer vision.

This culture shift brings fresh ideas to medical imaging but comes with a new challenge: the preservation of expert knowledge. Classic papers in the field assume familiarity with mathematical concepts that generally confuse modern students: Sobolev norms, spectral decompositions, etc. If they are not quickly re-formulated in a simpler way, key advances of the last decades run the risk of drifting out of the mainstream toolbox.

Acknowledging this danger, we refrain from using jargon whenever possible. We introduce all relevant theoretical concepts "from scratch", and generally favour eloquent examples over abstract definitions. We hope that this thesis will help our colleagues to blend together old and new ideas for the future of computational anatomy.

1.3.2 Designing robust geometric methods

Our contributions are detailed at the very beginning of this manuscript, in the extended abstract. To conclude this introduction, we now put our results in their context, at the intersection between machine learning and computational anatomy.

Chapter 2: computational tools. First of all, we focus on laying the numerical foundations of our work: deep learning libraries have revolutionized the way we code, but only support a limited range of operations. In order to let researchers tackle *geometric problems* with a convenient interface, we improve them with a first-rate support for distance-like matrices.

Our routines are packaged in the `KeOps` library, an extension for `PyTorch`, `NumPy`, `R` and `Matlab`: `KeOps` modules are differentiable, easy to integrate in existing codebases and fully documented. They have been developed in collaboration with Joan Glaunès and Benjamin Charlier and provide sizeable performance boosts to many applications.

Chapter 3: from kernels to optimal transport theory. We then proceed to study weighted sets: *measures*, in mathematical jargon. This abstraction may be used to work with segmentation maps in medical imaging, surface meshes in computer graphics or random variables in data sciences.

After a gentle introduction to measure theory, we discuss the *metrics* that can be used to compare these objects with each other. Focusing on optimal transport distances, we present a new multiscale Sinkhorn algorithm that can be understood as a smooth, high-dimensional Quicksort. This work results from a collaboration with Thibault Séjourné, François-Xavier Vialard and Gabriel Peyré: it is packaged in the `GeomLoss` library for `PyTorch` and comes with theoretical guarantees. In practice, Wasserstein distances are now as affordable and easy-to-use as standard kernel norms – with improved geometric properties.

Chapter 4: applications to shape analysis. The theory of measures is well suited to geometry processing: the encoding of a surface as a discrete measure (i.e. as a *weighted* point cloud) is intrinsically robust to re-parameterizations and re-meshings.

Building upon the general results of the previous chapter, we explain how our progresses in optimal transport theory can benefit the medical imaging community. In collaboration with Pierre Roussillon and Pietro Gori, we show how to use the `GeomLoss` routines to drive registration algorithms, transfer labels between two distributions or compute geometric shape templates at an affordable computational cost.

Chapter 5: continuous deformations. A key limitation of optimal transport is that it only models the displacement of *independent* particles. In most clinical settings, we must consider stronger metrics that guarantee the preservation of our shapes' topologies.

As explained throughout this introduction, designing metrics on spaces of shapes that hit a sweet spot between robustness, affordability and clinical relevance is an important problem in medical imaging. To conclude this thesis, we provide an overview of the classic approaches to this question – from elastic meshes to fluid mechanics. We present a model-agnostic method to enforce geometric axioms on shape metrics, and discuss the impact of modern numerical tools on the future of computational anatomy.

1.3.3 Future works

Two solid results. The work presented in Chapters 2 and 3 is now relatively mature. Some theoretical questions have been left unanswered, but our main results are now well-packaged, documented and already bring value to practitioners. Our implementations are bound to be improved upon, but the `LazyTensor` abstraction and multiscale Sinkhorn architecture will certainly stand the test of time.

Real-life applications. Contrastingly, the ideas presented in Chapter 4 are still very much work in progress. Our experiments on toy datasets suggest that optimal transport could provide answers to several problems in medical imaging… but validating these theoretical claims in a clinical setting will take at least a couple of years. We stop short of making any overzealous statement, and look forward to working with translational researchers on these questions.

Long-term target. Going further, the quest for data-driven shape models is still wide open. We deem it to be the key challenge of the 2020's for computational anatomy, and wrote this thesis accordingly. Relieving our colleagues from the burden of low-level programming has been our priority, motivating our focus on numerical routines and loss functions. From 2020 onwards, we should now be free to focus on the design of sensible anatomical models.

With the advent of fully automatic segmentation networks, extracting a clean "biomechanical" mesh out of a 3D image is becoming easier by the day. This key advance should enable the widespread use of pseudo-physical models (Montagnat et al., 2001) and intrinsic feature extractors (Bronstein et al., 2017) in computational anatomy. Adaptive models are bound to supersede homogeneous metrics, as in e.g. (Shen et al., 2019): we believe that the tools presented in this thesis will prove worthwhile to this research program, and intend to work on the topic in years to come.

Chapter 2

Designing efficient computational tools

in collaboration with Benjamin Charlier (University of Montpellier) and Joan Alexis Glaunès (Paris 5 University).

Key points – The untapped potential of GPUs, or why `PyTorch` is not the panacea:

1. Initially designed for video game consoles, Graphics Processing Units (GPUs) have revolutionized imaging sciences. Scientists can now take advantage of their **parallel computing** power with efficient "CUDA" codes and experience x100-10,000 speed-ups compared with baseline implementations.

2. Modern deep learning frameworks (`Theano`, `TensorFlow`, `PyTorch`...) provide a convenient interface for scientific computing. These `Python` libraries combine a **GPU backend** with a semi-symbolic engine for **automatic differentiation** and have the potential to supersede `Matlab` and `NumPy` in the applied maths community.

3. However, as of 2020, these major frameworks still keep a **narrow focus on (convolutional) neural networks**. As they only provide superficial support for mathematical operations outside of bitmap convolutions and linear algebra routines, `TensorFlow` and `PyTorch` can be vastly outperformed by custom CUDA codes on a wide collection of problems.

Contributions – Graphics-like performances with a transparent interface:

4. Our `KeOps` CUDA library implements **efficient map-reduce schemes** on the GPU, for arbitrary mathematical formulas: large quadratic operations such as nearest neighbors searches and kernel dot products now scale up to millions of samples in seconds, with a **linear memory footprint**. `KeOps` also supports **automatic differentiation**.

5. The `PyKeOps` package provides **a transparent `NumPy/PyTorch` interface** for the `KeOps` routines and is freely available on the `PyPi` repository (`pip install pykeops`). `Matlab` and `R` bindings are also available – with a `Julia` interface coming soon.

6. `KeOps` brings **graphics-like performances** to the machine learning and medical imaging communities: it allows users to benefit from a **x30 speed-up** compared with vanilla `PyTorch` GPU implementations – i.e. x10,000 compared with standard `NumPy` CPU routines. An in-depth documentation, numerous examples and tutorials are provided on our website:
   `www.kernel-operations.io`

Chapter 2 – Designing efficient computational tools:

2.1 Autodiff and GPU: the winning combination

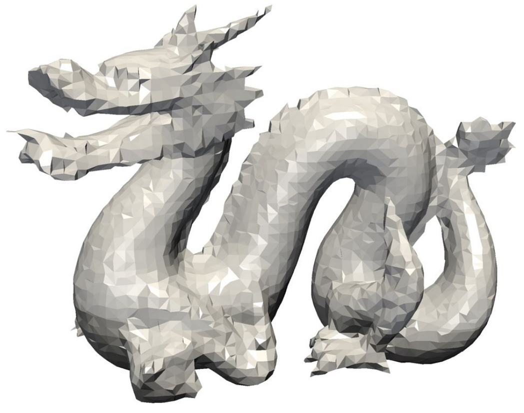
Modern deep learning frameworks. As discussed in section 1.1.1 (The deep learning revolution), recent advances in "artificial intelligence" have been driven by the diffusion of two pieces of software:

1. **Automatic differentiation.** Computing libraries rely on symbolic "historical records" that are attached to the program's variables to implement transparent `.grad()` operators.

2. **GPU backends for tensor-based computations.** Benefitting from the long-term investment of Nvidia, recent frameworks provide backends for *convolutions* and *linear algebra* operators that harness the parallel computing power of modern hardware.

These components enable the large scale tuning of the weights that parameterize *convolutional neural networks*. They were first paired together in a `Python` package by the `Theano` library (Al-Rfou et al., 2016), developed between 2007 and 2018 by the MILA institute. Today, using the Google and Facebook-backed `TensorFlow` and `PyTorch` libraries (Abadi et al., 2015; Paszke et al., 2017), tens of thousands of users routinely optimize massive objective functions using gradient descent strategies.

In less than ten years, these frameworks have allowed "GPU" and "backpropagation" to become buzzwords in applied sciences. However, outside of the graphics (Fernando et al., 2004) and autodiff (Hascoët and Pascual, 2013) communities, few researchers make the effort of understanding the inner workings of these convenient black-boxes. In a world where fast runtimes make or break the popularity of research fields, this oversight has effectively surrendered most of the scientific initiative in machine learning and image processing to the lead developers of `TensorFlow` and `PyTorch`.


(a) Subsampled model, with 11,102 triangles.

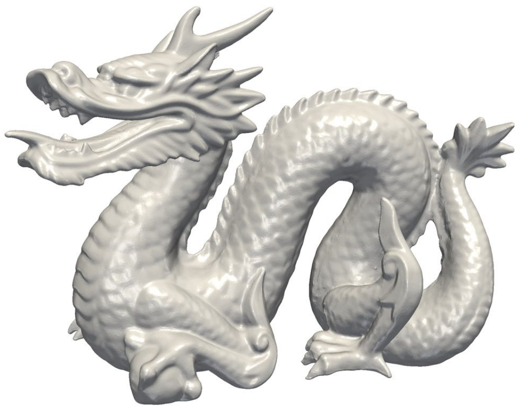

(b) Full model, with 871,414 triangles.

Figure 2.1: Illustrating the gap between the performances of machine learning and graphics routines with subsampled copies of the Stanford dragon (Curless and Levoy, 1996). (a) Due to intrinsic limitations of the tensor-centric paradigm implemented by `TensorFlow` and `PyTorch`, modern Gaussian processes packages cannot scale to datasets with more than 10,000 samples without making significant approximations (Gardner et al., 2018) or mobilizing high-end GPU chips for days (Wang et al., 2019). (b) Relying on a tailor-made CUDA backend, the `KeOps` library allows mathematicians to catch-up with the state-of-the-art and handle large datasets (i.e. point clouds) with a convenient interface.

A key limitation: the narrow focus on CNNs. Since the days of `Theano` and its `Lasagne` extension, deep learning frameworks have always prioritized the support of stacked *convolution* and *fully connected* layers – to the detriment of other algorithmic structures. Among mathematicians, this lack of investment in general purpose frameworks has led to a strong underrating of modern hardware: let's just cite the common belief, held in the machine learning community, that the ceiling for exact kernel methods on a single device lies around 10^4 samples (Hensman et al., 2013)… At a time when off-the-shelf graphical engines render millions of triangles at 60 frames per second on gaming laptops – see Figure 2.1.

Our contribution: stepping outside of the tensor-centric paradigm. Bringing graphics-like performances to our fellow mathematicians is the main goal of this thesis: in 2020, researchers should be allowed to stay *creative* without having to compromise too much on *performances*.

After a brief, high-level crash-course on backpropagation and the intricacies of GPU programming, we present our most important contribution to the field: the `KeOps` library, developed in collaboration with Benjamin Charlier and Joan Alexis Glaunès from 2017 onwards.

Through a convenient symbolic abstraction, the "`LazyTensor`" wrapper, this `Python-Matlab-R` package provides efficient support for kernel and distance-like matrices, *without ever compromising on usability*. Allowing researchers to play with algorithms that do not *solely* rely on convolutions and tensor manipulations, this toolbox lies at the heart of our progresses in optimal transport theory and computational anatomy.

2.1.1 What is a GPU?

Before going any further, we take some time to answer three questions on Graphics Processing Units, the workhorses of modern data sciences:

1. What is, precisely, a GPU?
2. How does GPU programming differ from standard (sequential) coding?
3. How much time and money does it take to benefit from this hardware revolution?

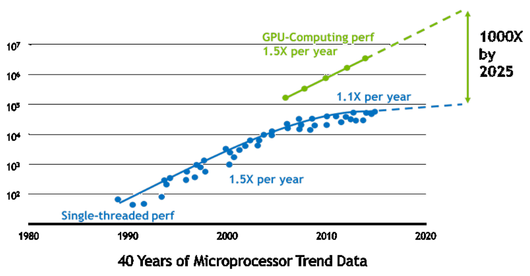
Parallel computing. At first glance, GPUs can be described as clusters of cheap but efficient workers that are sold by the thousands on affordable chips. Modern devices come with 4,000+ computing cores, 10+ Gigabytes of memory and can be bought for less than 1,500$: they are able to compute large matrix-vector products at a fraction of the cost of a traditional high performance platform. As shown Figure 2.2, hardware constructors focus their communication around a simple message: the "GPU revolution" is that of an *economy of scale*.

Memory management. As soon as we start diving into the specialized literature, however, things become murkier: extracting peak performances from a pool of 4,000+ workers is no mean feat. Crucially, just like administrators of XIX$^{\text{th}}$ century bureaus, hardware designers have to enforce guidelines on the behavior of their cores and hard-code *structuring constraints* in the circuitry of their devices. Abstracted through the *CUDA memory model*, the main rules of GPU programming are illustrated in Figure 2.3 and can be summarized as follows:

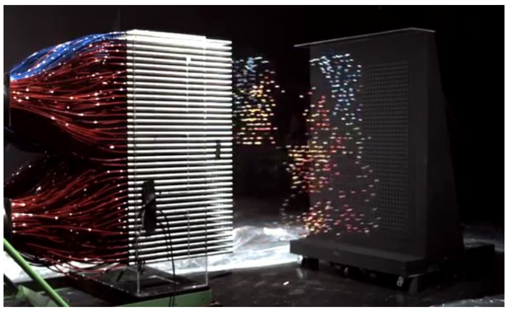
1. GPU threads are organized in interchangeable **blocks** of up to 1,024 workers, which can be identified to the many teams of a large State department.

2. Far from lying scattered in the device memory, information is finely managed and stored in several layers of hardware. In practice, pushing aside some technicalities, scientific CUDA programs may rely on **four different types of memory**:

    1. The **Host memory**, i.e. the usual RAM of the computer that is managed by the main CPU program. It is located far away from the GPU's computing circuits, which cannot access it directly. In our XIX$^{\text{th}}$ century analogy, it would be represented by the heaps of documents stored in *other* State offices, possibly overseas.
    2. The **Device memory**, which plays the role of a *local* RAM chip embedded on the GPU. This is the *library* of our State office, where information is stored before being processed by workers: depending on the model, recent GPUs may be able to store up to 32 Gigabytes of data in this convenient storage location.
    3. The **Shared memory** which is, well, *shared* by all threads in a CUDA block. This small buffer may only store up to ~96 Kilobytes of data and can be compared to the *office shelf* of Figure 2.3.c. Crucially, its latency is *much lower* than that of the Device memory: optimizing its usage in the `KeOps` library was the key to a x50 speed-up for all kernel-related operations.
    4. The **Register** or **Thread memory**, a very low-latency buffer that is uniquely attributed to each worker – a bit like sheets of scrap paper on a desk. A maximum of

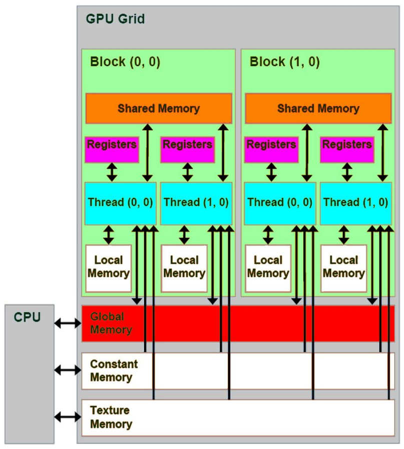


(a) *The time for GPU computing has come*, from the Nvidia website `www.nvidia.com`.

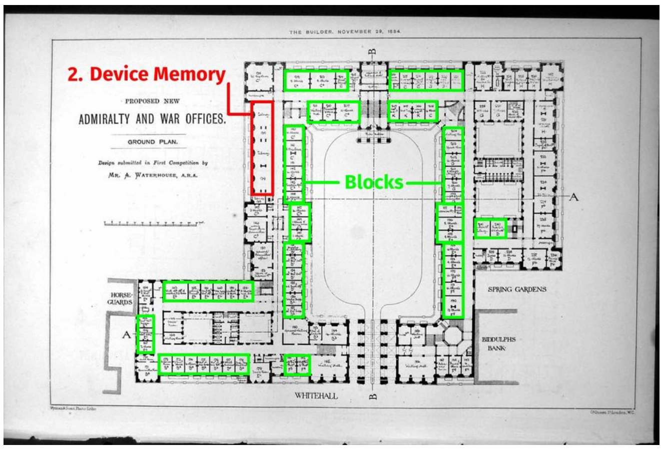


(b) *Mythbusters demo GPU versus CPU*, from the Nvidia YouTube channel.

Figure 2.2: Promotional material taken from the Nvidia website. Most sensibly, hardware constructors focus their marketing strategy on the raw computing power of GPUs and brush under the carpet the key specificity of CUDA programming: **finely grained memory management.**

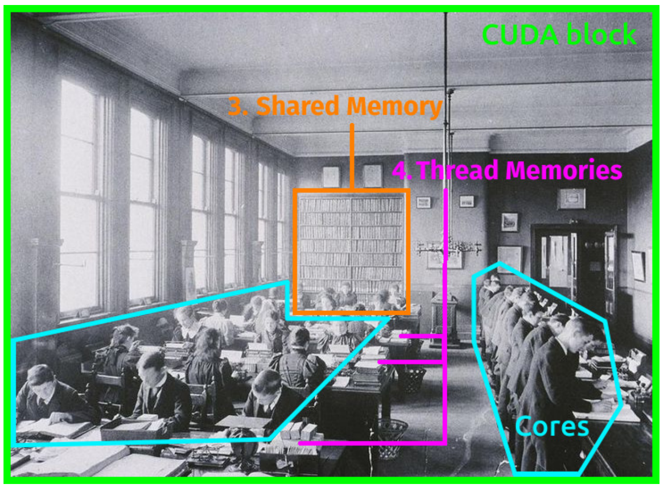


(a) The CUDA memory model, adapted from (Kirk and Wen-Mei, 2010).



(b) Ground plan of the *Admiralty and War Offices*, built in 1884.

Figure 2.3: Abiding by the same constraints, the architecture of modern GPUs (a) closely resembles that of XIX$^{\text{th}}$ century State departments (b-c) which had to keep track of accounting records for large overseas empires. Organizing their workers in identical teams of a few dozen cores, massively parallel devices rely on a finely grained management of information transfers (3,4) to prevent traffic jams at the gates of the Device memory (2).



(c) Inside view of a computational block.

∼256 Kilobytes per block may be attributed this way: values in excess are stored in the (high-latency) Local memory by the compiler.

3. To leverage modern GPUs to the best of their abilities, efficient algorithms should thus obey to **four guiding principles**:

    1. **Promote block-wise parallelism.** Threads can interact with each other during the execution of a program, but may only do so *inside their own CUDA block*.
    2. **Reduce Host↔Device memory transfers.** Incessant back-and-forth copies between the "CPU" and "GPU" RAMs may quickly become the bottleneck of modern research codes. Fortunately, the `TensorFlow` and `PyTorch` APIs now allow users to store and manipulate their data on the device, without ever coming back to the Host memory chip.
    3. **Reduce Device↔Shared/Register memory transfers.** Due to the (relatively) high latency of the Device memory, *programmers should refrain from storing and handling intermediate buffers* outside of the Shared memory of a CUDA block. Unfortunately, the high-level APIs of modern deep learning libraries do *not* allow users to get such a fine-grained control on their computations: this is the main limitation that the `KeOps` package strives to mitigate, with minimal impact on users' existing codebases.
    4. **Promote block-wise memory accesses.** GPUs' memory circuits are wired in a way that promotes *contiguous*, page-wise exchanges between the Device and the Shared memories. Initially designed to process triangles and textures at ever faster rates, GPUs are thus somewhat ill-suited to the processing of *sparse matrices* which rely on rare but *random* memory accesses.

The CUDA development toolkit. Once these constraints are understood and taken into account, CUDA programming is suprisingly easy. Well aware that the promising "AI computing" GPU market would never boom without a strong investment, Nvidia devoted an impressive amount of effort to the creation of a comfortable development environment: an efficient compiler, good profiling tools, robust libraries... and a comprehensive documentation! To get started with GPU programming, a perfect introduction is the series of tutorials written by Mark Harris on the Nvidia devblog:

`devblogs.nvidia.com/even-easier-introduction-cuda`

In a nutshell, typical CUDA C++ files look like:

```
// Import your favorite libraries:
#include <iostream>
#include <math.h>

// The __global__ keyword indicates that the following code is to
// be executed on the GPU by blocks of CUDA threads, in parallel.
// Pointers refer to arrays stored on the Device memory:
```

```
__global__
void My_CUDA_kernel(int parameter, float *device_data, float *device_output) {

    // The indices of the current thread and CUDA block should be
    // used to assign each worker to its place in the computation plan:
    int i = blockIdx.x * blockDim.x + threadIdx.x;

    // The Shared memory is accessed through a raw C++ pointer:
    extern __shared__ float shared_mem[];

    // Local variables may be declared as usual.
    // They'll be stored in the Thread memory whenever possible:
    float some_value = 0;

    // Transfers of information are handled with a transparent interface:
    some_value    = device_data[i];  // Thread memory <- Device memory
    shared_mem[i] = device_data[i];  // Shared memory <- Device memory

    // Whenever required, programmers may create checkpoints for all threads
    // in a CUDA block. Needless to say, this may impact performances.
    __syncthreads();

    // Computations are written in standard C++ and executed in parallel:
    for(int k = 0; k < parameter; k++) {
        // Blablabla
    }

    // Finally, results can be written back to the Device memory with:
    device_output[i] = some_value;  // Device memory <- Thread memory
}


// The main C++ program, executed by the CPU:
int main(void) {
    int N = 1024; float *host_data, *host_out, *device_data, *device_out;

    // Allocate memory on the device - the API is a bit heavy:
    cudaMalloc((void**) &device_data,  N*sizeof(float));
    cudaMemcpy(device_data, host_data, N*sizeof(float), cudaMemcpyHostToDevice);

    // Set the parameters of the CUDA block and run our kernel on the GPU:
    int block_size = 128; int grid_size  = N / block_size;
    int shared_mem_size = 2 * block_size * sizeof(float);
    My_CUDA_kernel<<<grid_size, block_size, shared_mem_size>>>(...);

    cudaDeviceSynchronize(); // Wait for the GPU to finish its job...
    cudaMemcpy(host_out, device_out, N*sizeof(float), cudaMemcpyDeviceToHost);
    ... // Do whatever you want with the result "output array"...
    cudaFree(device_data);  // And don't forget to free the allocated memory!
    return 0;
}
```

How much is this going to cost? Assuming some level of familiarity with C++ programming, designing a CUDA application is thus relatively easy. Thanks to the recent availability of modern – and incredibly convenient – `Python/C++` interfaces such as `Pybind11` (Jakob et al., 2017), the path that takes scientists from CUDA 101 tutorials to fully-fledged open source libraries is now well trodden. But how expensive are these solutions for academic users?

Nvidia's de facto monopoly. Due to the costly nature of hardware design, the GPU market is an oligopoly with no more than three constructors in business:

1. Intel, which produces integrated graphics chips for the mass-consumer market;
2. Nvidia, the established producer of high-end devices;
3. AMD, the eternal competitor of Nvidia on the gaming and cryptocurrency markets.

Unfortunately for academics, out of those three players, Nvidia is the only one that invests seriously in the "AI" and "scientific computing" segments, backing up its hardware with state-of-the-art computing libraries. As far as researchers are concerned, GPU computing is thus a *captive market*, with two ranges of products to pick from:

1. The GeForce **gaming** range, with a flagship model sold for $\sim$1,500$ and slightly defective or more compact chips marketed at lower prices. As of 2020, the GeForce RTX 2080 Ti provides the best value for money for generic academic purposes.

2. The **data center** series, whose slightly more versatile chips are typically sold for $\sim$10,000$ per unit. This higher price is justified by a larger Device memory (from 11 Gb to 32 Gb), efficient support of `float64` computations, marginal improvements in the circuits' architectures... and a recently updated license agreement (2018+) for the CUDA drivers, which forbids data centers from relying on GeForce devices.

Cloud solutions. Dedicated machines are must-buys for deep learning research teams who intend to use their GPUs full-time for the training of neural architectures. However, for theorists and mathematicians who only ever need to use the latest hardware once a month to produce figures and benchmarks, a smarter option may be to rely on cloud rental services. At affordable rates of $\sim$1-3$ per hour – which correspond to amortization periods of one or two months of 24/7 usage – Google, Amazon or Microsoft let customers access their latest machines, free of any maintenance hassle.

Google Colab. Most interestingly, Google provides free GPU sessions to all "GMail" accounts at `colab.research.google.com`. The constraints that are put on these sessions are clear: 12 hours shelf-life of the virtual machines, privacy concerns when working with real data... But they're absolutely worth trying out for "casual" students and researchers.

This work. For the sake of reproducibility and ease of use, we made sure that **all the packages and experiments presented in this thesis run out-of-the-box on free Colab sessions**. Usually, typing "`!pip install pykeops[full]`" in a Colab cell is everything it takes to try our software online: so please play around with these tools, they're *free as in freedom* for everything that's explained here, and *free as in beer* for the rest!

2.1.2 Automatic differentiation: the backpropagation algorithm

Now that the main rules of GPU programming have been exposed, let us recap the fundamentals of *backpropagation* or *reverse accumulation AD*, the algorithm that allows Automatic Differentiation (AD) engines to differentiate scalar-valued computer programs $F : \mathbb{R}^n \rightarrow \mathbb{R}$ efficiently. As we uncover the methods that are hidden behind the transparent "`.grad()`" calls of modern libraries, we will hopefully allow the reader to understand the *rules* and *limitations* of automatic differentiation engines.

Differential. First of all, let us make our notations precise by recalling the mathematical definition of the *differential* of a smooth function.

Definition 2.1 (Differential)**.** Let $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ be two normed vector spaces. A function $F : X \rightarrow Y$ is said to be (Fréchet) differentiable at $x_0 \in X$ if there exists a continuous linear operator $\mathcal{L} : X \rightarrow Y$ such that:

$$\forall\, \delta x \in X, \quad F(x_0 + \delta x) = F(x_0) + \mathcal{L}(\delta x) + o(\|\delta x\|_X) \tag{2.1}$$

or equivalently:

$$\lim_{\delta x \rightarrow 0} \frac{\|F(x_0 + \delta x) - F(x_0) - \mathcal{L}(\delta x)\|_Y}{\|\delta x\|_X} = 0\,. \tag{2.2}$$

If it exists, such a linear operator $\mathcal{L}$ is unique. It is called the **differential** of F at x_0 and is denoted by $\mathcal{L} = \mathrm{d}_x F(x_0)$. We use a dot symbol to denote the application of $\mathcal{L}$, as in

$$\mathcal{L}(\delta x) = \mathrm{d}_x F(x_0) \cdot \delta x\,. \tag{2.3}$$

Jacobian matrix. Let us consider the spaces $X = \mathbb{R}^n$ and $Y = \mathbb{R}^m$ endowed with their usual (Euclidean) metric structures. Given a function $F = (F^1, \dots, F^m)$ that is differentiable at a location x_0 in $\mathbb{R}^n$, the matrix of the linear operator $\mathrm{d}_x F(x_0)$ in the canonical basis is the *Jacobian matrix* of partial derivatives:

$$\begin{pmatrix} \frac{\partial F^1}{\partial x^1}(x_0) & \cdots & \frac{\partial F^1}{\partial x^n}(x_0) \\ \vdots & \frac{\partial F^i}{\partial x^j}(x_0) & \vdots \\ \frac{\partial F^m}{\partial x^1}(x_0) & \cdots & \frac{\partial F^m}{\partial x^n}(x_0) \end{pmatrix}. \tag{2.4}$$

Gradient vector. When F is scalar-valued ($m = 1$), the Jacobian matrix is a *line*: to retrieve a column *gradient vector* in $\mathbb{R}^n$, one usually considers its *transpose*. To define this manipulation in a way that is **coordinate-free**, independent of the choice of a reference basis, we must assume that X is a *Hilbert space*, i.e. that its metric structure is *complete* and comes from an inner product $\langle\,\cdot\,,\,\cdot\,\rangle_X : X \times X \rightarrow \mathbb{R}$. Then, we can write:

Definition 2.2 (Gradient vector)**.** Let $(X, \langle\,\cdot\,,\,\cdot\,\rangle_X)$ be a Hilbert space, x_0 a reference location in X and $F : X \rightarrow \mathbb{R}$ a function that is differentiable at x_0. As its differential $\mathrm{d}_x F(x_0) : X \rightarrow \mathbb{R}$ at x_0 is a *continuous linear form*, the **Riesz representation theorem** ensures that there exists a unique vector $\nabla_x F(x_0) \in X$, the **gradient** of F at x_0 such that

$$\forall\, \delta x \in X, \quad \mathrm{d}_x F(x_0) \cdot \delta x = \langle\, \nabla_x F(x_0)\,,\, \delta x\, \rangle_X\,. \tag{2.5}$$

Computing gradients: finite differences are *not* a good solution. A naive approach to the computation of gradient vectors, the so-called *finite differences* scheme, is to use a Taylor expansion of F around x_0 and write that for small enough values of δt,

$$\nabla_x F(x_0) \;=\; \begin{pmatrix} \partial_{x^1} F(x_0) \\ \partial_{x^2} F(x_0) \\ \vdots \\ \partial_{x^n} F(x_0) \end{pmatrix} \;\simeq\; \frac{1}{\delta t} \begin{pmatrix} F(x_0 + \delta t \cdot (1,0,0,\dots,0)) - F(x_0) \\ F(x_0 + \delta t \cdot (0,1,0,\dots,0)) - F(x_0) \\ \vdots \\ F(x_0 + \delta t \cdot (0,0,0,\dots,1)) - F(x_0) \end{pmatrix}. \tag{2.6}$$

This idea is simple to implement, but also requires $(n+1)$ evaluations of the function F to compute a *single* gradient vector! As soon as the dimension n of the input space exceeds 10 or 100, this stops being tractable: just like inverting a full matrix A is not a sensible way of solving the linear system "$Ax = b$", one should not use finite differences – or any equivalent *forward* method – to compute the gradient of a scalar-valued objective.

Generalized gradient. To go beyond this simple scheme, we need to work with the gradient of *vector-valued* applications. Once again, coordinate-free definitions rely on scalar products:

Definition 2.3 (Generalized gradient). Let $(X, \langle \cdot\,, \cdot \rangle_X)$ and $(Y, \langle \cdot\,, \cdot \rangle_Y)$ be two Hilbert spaces, and let $F : X \to Y$ be a function that is differentiable at $x_0 \in X$. The adjoint:

$$(\mathrm{d}_x F)^*(x_0) : Y^* \to X^* \tag{2.7}$$

of the differential induces a continuous linear map:

$$\mathrm{d}_{\boldsymbol{x}}^{\top} \boldsymbol{F}(\boldsymbol{x}_0) : \boldsymbol{Y} \to \boldsymbol{X} \tag{2.8}$$

through the Riesz representation theorem, called the **generalized gradient** of F at x_0 with respect to the Hilbertian structures of X and Y.

Calculus. The generalized gradient appears in the infinitesimal development of scalar quantities computed from $F(x)$ around a reference location x_0. Let $\alpha \in Y^*$ be a *continuous* linear form on Y, identified with a vector $a \in Y$ through the Riesz representation theorem:

$$\forall\, y \in Y,\ \langle\, \alpha\,,\, y\,\rangle \overset{\text{def.}}{=} \alpha(y) \overset{\text{def.}}{=} \langle\, a\,,\, y\,\rangle_Y\,. \tag{2.9}$$

Then, for any increment $\delta x \in X$, we can write that:

$$\begin{aligned} \langle \alpha, F(x_0 + \delta x) \rangle &= \langle \alpha, F(x_0) \rangle &&+ \langle\ \ \alpha,\ \ \mathrm{d}_x F(x_0) \cdot \delta x\ \rangle &&+ o(\|\delta x\|_X) && (2.10) \\ &= \langle \alpha, F(x_0) \rangle &&+ \langle\, (\mathrm{d}_x F)^*(x_0) \cdot \alpha,\ \delta x\ \rangle &&+ o(\|\delta x\|_X) && (2.11) \\ \text{i.e. } \langle a, F(x_0 + \delta x) \rangle_Y &= \langle a, F(x_0) \rangle_Y &&+ \langle\ \ \ \mathrm{d}_x^{\top} F(x_0) \cdot a,\ \delta x\ \rangle_X &&+ o(\|\delta x\|_X)\,. && (2.12) \end{aligned}$$

Fundamental example. If X and Y are respectively equal to $\mathbb{R}^n$, $\mathbb{R}$ and are endowed with the standard L^2-Euclidean dot products:

$$\langle x, x' \rangle_{\mathbb{R}^n} \;=\; \sum_{i=1}^{n} x_i x_i' \qquad \text{and} \qquad \langle y, y' \rangle_{\mathbb{R}} \;=\; y y'\,, \tag{2.13}$$

the matrix of $\mathrm{d}_x^\top F(x_0) : \mathbb{R} \to \mathbb{R}^n$ in the canonical basis is equal to the vector $\nabla_x F(x_0)$ of directional derivatives:

$$\nabla_x F(x_0) \;=\; \mathrm{d}_x^\top F(x_0) \,\cdot\, 1\,. \tag{2.14}$$

Going further, the matrix of the generalized gradient in the canonical basis coincides with the transpose of the Jacobian matrix whenever the scalar products considered are equal to the "canonical" ones. Everything is consistent.

Generalized gradients stress the influence of the metric structure. Chain rule. This generalized "metric" definition of the gradient has two major advantages over the simple notion of "vector of partial derivatives":

1. It stresses the fact that a gradient is always defined with respect to a *metric structure*, not a basis. In high-dimensional settings, as the equivalence of norms stops being effective, the choice of an appropriate *descent metric* becomes a key regularization prior for first-order optimization schemes. Encoded through a change of variables on the parameters that we strive to optimize, this modelling choice usually has a strong impact on machine learning pipelines (Amari, 1998; Ulyanov et al., 2018; Surace et al., 2018); we discuss it in section 5.2.3, for shape registration.

2. It allows us to *compose* gradients without reserve. Indeed, if X, Y, Z are three Hilbert spaces and if $F = H \circ G$ with $G : X \to Y$ and $H : Y \to Z$, then for all $x_0 \in X$, the chain rule asserts that

$$\mathrm{d}_x F(x_0) \;=\; \mathrm{d}_y H(G(x_0)) \circ \mathrm{d}_x G(x_0)\,. \tag{2.15}$$

With the usual flip for the composition of adjoint (i.e. transposed) operators, we get:

$$[\mathrm{d}_x F(x_0)]^* \;=\; [\mathrm{d}_x G(x_0)]^* \circ [\mathrm{d}_y H(G(x_0))]^* \tag{2.16}$$

$$\text{i.e.}\qquad \mathrm{d}_x^\top F(x_0) \;=\; \mathrm{d}_x^\top G(x_0) \;\circ\; \mathrm{d}_y^\top H(G(x_0))\,. \tag{2.17}$$

Backpropagation. In practice, the function $F : \mathbb{R}^n \to \mathbb{R}$ to differentiate is defined as a composition $F = F_p \circ \cdots \circ F_2 \circ F_1$ of elementary functions $F_i : \mathbb{R}^{\mathrm{N}_{i-1}} \to \mathbb{R}^{\mathrm{N}_i}$ where $\mathrm{N}_0 = n$ and $\mathrm{N}_p = 1$: the lines of our program.

$$\mathbb{R}^n = \mathbb{R}^{\mathrm{N}_0} \xrightarrow{\;F_1\;} \mathbb{R}^{\mathrm{N}_1} \xrightarrow{\;F_2\;} \mathbb{R}^{\mathrm{N}_2} \xrightarrow{\;\cdots\;} \cdots \xrightarrow{\;F_p\;} \mathbb{R}^{\mathrm{N}_p} = \mathbb{R}$$

To keep the notations simple, we now assume that all the input and output spaces $\mathbb{R}^{\mathrm{N}_i}$ are endowed with their canonical L^2-Euclidean metrics. The gradient vector $\nabla_x F(x_0)$ that we strive to compute, at an arbitrary location $x_0 \in \mathbb{R}^n$, is the image of $1 \in \mathbb{R}$ by the linear map:

$$\mathrm{d}_x^\top F(x_0) : \mathbb{R} \to \mathbb{R}^n. \tag{2.18}$$

Thanks to the chain rule, we can write that:

$$\begin{aligned}\mathrm{d}_x^\top F(x_0) \cdot 1 &= \mathrm{d}_x^\top F_1(x_0) \circ \mathrm{d}_x^\top F_2(F_1(x_0)) \circ \cdots \circ \mathrm{d}_x^\top F_p(\,F_{p-1}(\cdots(F_1(x_0)))\,) \cdot 1 && (2.19)\\ &= \mathrm{d}_x^\top F_1(x_0) \,\cdot\, \mathrm{d}_x^\top F_2(\quad x_1 \quad) \cdot \cdots \cdot \mathrm{d}_x^\top F_p(\qquad x_{p-1} \qquad) \cdot 1 && (2.20)\end{aligned}$$

where the x_i's $= F_i \circ \cdots \circ F_1(x)$ denote the intermediate results in the computation of $x_p = F(x_0)$. Assuming that the *forward* and *backward* operators:

$$\begin{array}{rcrcl} F_i & : & \mathbb{R}^{\mathrm{N}_{i-1}} & \to & \mathbb{R}^{\mathrm{N}_i} \\ & & x & \mapsto & F_i(x) \end{array} \qquad (2.21)$$

and

$$\begin{array}{rcrcl} \mathrm{d}_x^\top F_i & : & \mathbb{R}^{\mathrm{N}_{i-1}} \times \mathbb{R}^{\mathrm{N}_i} & \to & \mathbb{R}^{\mathrm{N}_{i-1}} \\ & & (x, a) & \mapsto & \mathrm{d}_x^\top F_i(x) \cdot a \end{array} \qquad (2.22)$$

are known and **encoded as computer programs**, we can thus compute both $F(x_0)$ and $\nabla_x F(x_0) = \mathrm{d}_x^\top F(x_0) \cdot 1$ with a forward-backward pass through the following diagram:

$$\begin{tikzcd}
\mathbb{R}^n & \mathbb{R}^{\mathrm{N}_0} & \mathbb{R}^{\mathrm{N}_1} & \mathbb{R}^{\mathrm{N}_2} & \cdots & \mathbb{R}^{\mathrm{N}_p} = \mathbb{R} \\
\cup & \cup & \cup & \cup & & \cup \\
x_0 \arrow[r, "\text{input}"] & x_0 \arrow[r, "F_1"] \arrow[d] & x_1 \arrow[r, "F_2"] \arrow[d] & x_2 \arrow[r, "\cdots"] \arrow[d] & \cdots \arrow[r, "F_p", "\text{output}"'] \arrow[d] & x_p = F(x_0) \\
\mathrm{d}_x^\top F(x_0) \cdot 1 & \mathrm{d}_x^\top F_1 \arrow[l, "\text{output}"', "\substack{x_0^* \\ \cap \\ \mathbb{R}^{\mathrm{N}_0}}"] & \mathrm{d}_x^\top F_2 \arrow[l, "\substack{x_1^* \\ \cap \\ \mathbb{R}^{\mathrm{N}_1}}"] & \cdots \arrow[l, "\substack{x_2^* \\ \cap \\ \mathbb{R}^{\mathrm{N}_2}}"] & \mathrm{d}_x^\top F_p \arrow[l, "\cdots"] & 1 \arrow[l, "\text{input}"', "\substack{x_p^* \\ \cap \\ \mathbb{R}^{\mathrm{N}_p}}"]
\end{tikzcd}$$

In a nutshell. The *backpropagation* algorithm can be cut in two steps that correspond to the two lines of the diagram above:

1. Starting from $x_0 \in \mathbb{R}^n = \mathbb{R}^{\mathrm{N}_0}$, compute and **store in memory** the successive vectors $x_i \in \mathbb{R}^{\mathrm{N}_i}$. The last one, $x_p \in \mathbb{R}$, is equal to the value of the objective $F(x_0)$.

2. Starting from the canonical value of $x_p^* = 1 \in \mathbb{R}$, compute the successive **dual vectors**:

$$x_i^* \;=\; \mathrm{d}_x^\top F_{i+1}(x_i) \cdot x_{i+1}^* \,. \qquad (2.23)$$

   The last one, $x_0^* \in \mathbb{R}^n$, is equal to the gradient vector $\nabla_x F(x_0) = \mathrm{d}_x^\top F(x_0) \cdot 1$.

Implementation and performances. This forward-backward procedure can be generalized to all acyclic computational graphs. Hence, provided that the forward and backward operators defined Eqs. (2.21-2.22) are implemented and available, we can compute *automatically* the gradient of any symbolic procedure that is written as a succession of elementary differentiable operations: the F_i's.

In practice, the *backwards* of usual operations are seldom more costly than 4-5 applications of the corresponding *forward* operators: differentiating a polynomial gives us a polynomial, logarithms become pointwise inversions, etc. Ergo, if one has enough memory at hand to store the intermediate results $x_0, \ldots, x_{p-1}$ during the forward pass, **the backpropagation algorithm is an automatic and time-effective way of computing the gradients** of generic scalar-valued functions, with **runtimes that do not exceed that of four or five applications of the forward program.** This statement may come as a shock to first-time users of deep learning frameworks; but as we are about to see, it is both *true* and *effective*.

2.1.3 Working with high-level computational libraries

To provide a transparent interface for the backpropagation algorithm, deep learning libraries rely on three core modules:

1. **A comprehensive list of operations** provided to end-users, with forward and backward routines implemented *next to each other*.
2. **Efficient CPU and GPU backends** for those routines, which allow users to take advantage of their hardware without having to write a single line of C++.
3. **A high-level graph manipulation engine** for symbolic computations, which executes the backpropagation's "backward pass" whenever a gradient value is required.

A minimal working example. Let us illustrate the underlying mechanics of PyTorch – the most popular framework among academics – in a simple case: the computation of the Gaussian squared *kernel norm*:

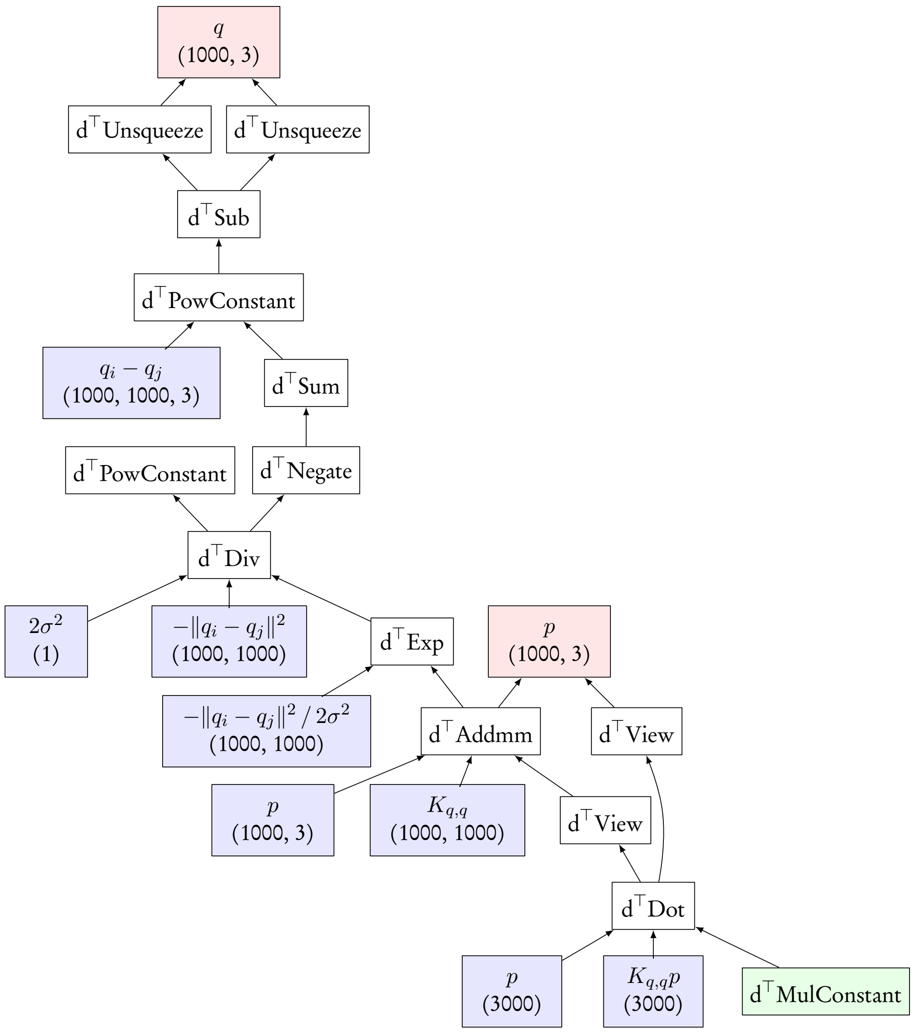
$$H(q,p) = \frac{1}{2}\sum_{i,j=1}^{\mathrm{N}} k(q_i - q_j)\,\langle p_i, p_j\rangle_{\mathbb{R}^{\mathrm{D}}} \qquad \text{where} \qquad k(x) = \exp(-\|x\|^2\,/\,2\sigma^2) \tag{2.24}$$

$$= \frac{1}{2}\langle p,\, K_{q,q}\, p\,\rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}} \qquad \text{where} \qquad (K_{q,q})_{i,j} = k(q_i - q_j)\,, \tag{2.25}$$

and of its gradients with respect to the input arrays $(q_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and $(p_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$. Using the standard (tensorized) PyTorch interface, programmers may write:

```python
import torch                      # GPU + autodiff library
from torchviz import make_dot   # See github.com/szagoruyko/pytorchviz

# With PyTorch, using the GPU is that simple:
use_gpu = torch.cuda.is_available()
dtype   = torch.cuda.FloatTensor if use_gpu else torch.FloatTensor
# Under the hood, this flag determines the backend that is to be
# used for forward and backward operations, which have all been
# implemented both in pure CPU and GPU (CUDA) code.

# Step 1: Define numerical tensors (from scratch or numpy) --------------------
N, D = 1000, 3  # Work with clouds of 1,000 points in 3D
# Generate arbitrary arrays on the host (CPU) or device (GPU):
q = torch.rand(N, D).type(dtype)  # random tensor of shape (N,D)
p = torch.rand(N, D).type(dtype)  # random tensor of shape (N,D)
s = torch.Tensor([2.5]).type(dtype)  # deviation "sigma" of our kernel

# Step 2: Ask PyTorch to keep track of q and p's children ---------------------
# In this demo, we won't try to fine tune the kernel and
# do not need any derivative with respect to sigma:
q.requires_grad = True
p.requires_grad = True
```

```python
# Step 3: Actual computations --------------------------------------------------
# Every PyTorch instruction is executed on-the-fly, but the graph API
# 'torch.autograd' keeps track of the operations and stores in memory
# the intermediate results that are needed for the backward pass.
q_i  = q[:,None,:]  # shape (N,D) -> (N,1,D)
q_j  = q[None,:,:]  # shape (N,D) -> (1,N,D)
D_ij  = ((q_i - q_j) ** 2).sum(dim=2)  # squared distances |q_i-q_j|^2
K_ij = (- D_ij / (2 * s**2) ).exp()    # Gaussian kernel matrix
v    = K_ij @ p  # matrix multiplication. (N,N) @ (N,D) = (N,D)

# Finally, compute the kernel norm H(q,p):
H = .5 * torch.dot( p.view(-1), v.view(-1) ) # .5 * <p,v>

# Display the computational graph in the figure below, annotated by hand:
make_dot(H, {'q':q, 'p':p}).render(view=True)
```



Encoding formulas as tree-like objects. With PyTorch, tensor variables are much more than plain numerical arrays. Any tensor that descends from a differentiable variable (marked with the flag `.requires_grad = True`) possesses two essential attributes:

1. A `.data` pointer which refers to a C++ array that may be stored in either Host (CPU) or Device (GPU) memories.
2. A `.grad_fn` recursive tree-like object, which records the **computational history** of the tensor and can be used whenever a backward pass is required by the `.grad()` operator.

In the picture above, we displayed the "`H.grad_fn`" attribute of our kernel norm using the GraphViz Dot program (Gansner and North, 2000). This acyclic graph is the exact equivalent of the second "backward" line of the backpropagation diagram that we presented page 34:

- **White nodes** stand for backward operators $\mathrm{d}_x^\top F_{i+1} : (x_i, x_{i+1}^*) \mapsto x_i^*$.
- **The green leave** is the first covariable $x_p^* \in \mathbb{R}$, the "gradient with respect to the output" that is initialized to 1 by default.
- **Red leaves** are the covariables "x_0^*", the gradients that we are about to compute.
- **Blue leaves** are the *stored* values x_i that were computed during the forward pass.

A well-packaged backropagation engine. Thanks to the groundwork done by the PyTorch symbolic engine, computing gradients is now as simple as writing:

```
grad_q, grad_p = torch.autograd.grad( H, [q, p] )  # pair of (N,D) tensors
```

That's it – and it goes pretty fast! As should be evident by now, the blend of semi-symbolic calculus and parallel performances that deep learning frameworks provide is a *game changer* for applied mathematicians. Before going any further, we strongly advise readers to try out these scripts on their machines and go through the "Matlab/NumPy to PyTorch" migration guide, which is available at:

pytorch.org/tutorials/beginner/pytorch_with_examples.html

Custom operators, higher-order differentiation. As explained in this tutorial, creating new pairs of (forward, backward) PyTorch operators is easy. The "`torch.autograd.Function`" module allows users to inject their own C++ code in a PyTorch program and is a convenient interface for the developers of PyTorch_Geometric (Fey and Lenssen, 2019), GPytorch (Gardner et al., 2018) and other contributed extensions to the vanilla framework.

Please note that the PyTorch engine also supports the computation of *high-order gradients* through the "`create_graph = True`" optional argument of the `.grad()` operator. Even though full Hessian matrices may *not* be computed efficiently using backprop-like strategies – they're typically way too large anyway – formulas that involve gradients may themselves be understood as "vector computer programs" and differentiated accordingly. In practice, developers of contributed libraries just have to make sure that their *backward* operators rely on well-defined *forward* routines, thus allowing the `autograd` engine to bootstrap the computation of high-order derivatives.

2.2 Going beyond tensor-based computations with the KeOps CUDA library

Memory usage, performances. Out of the box, the tensor-centric interfaces of `PyTorch`, `TensorFlow`, `Matlab` and `NumPy` strike a good balance between power and simplicity: explicit matrices allow users to implement engineering tools with a code that stays close to the maths.

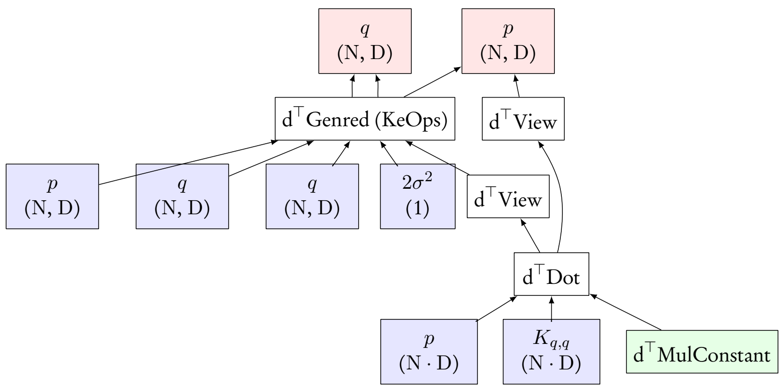
But there is a limit to what full matrices can handle: **whenever the operators involved present some structure, baseline matrix-vector products can be vastly outperformed by domain-specific implementations.** Some examples of this rule are well-known and supported by major frameworks through dedicated methods and "layers":

- In image processing, **convolutions**, **Fourier** and **wavelet transforms** rely on ad hoc schemes that do not involve circulant or Vandermonde matrices.
- On graph or mesh data, **sparse matrices** are encoded as lists of indices plus coefficients and provide support for local operators: graph Laplacians, divergences, etc.

`KeOps`: adding support for symbolic tensors. Surprisingly, though, little to no effort has been made to support generic **mathematical** or **"symbolic" matrices**, which are not *sparse* in the traditional sense but can nevertheless be encoded compactly in memory using a *symbolic formula* and some small data arrays.

Allowing the users of **kernel or distance matrices** to bypass the *transfer* and *storage* of superfluous quadratic buffers is the main purpose of the `KeOps` library. As a bite-sized example of our interface, the program below is a revision of the script presented page 36 that **scales up to clouds of N = 1,000,000 samples in less than a second on modern hardware**, with a linear memory footprint – remark the absence of any N-by-N buffer in the graph.

```python
from pykeops.torch import LazyTensor  # Semi-symbolic wrapper for torch Tensors
q_i  = LazyTensor( q[:,None,:] )  # (N,D) Tensor -> (N,1,D) Symbolic Tensor
q_j  = LazyTensor( q[None,:,:] )  # (N,D) Tensor -> (1,N,D) Symbolic Tensor

D_ij = ((q_i - q_j) ** 2).sum(dim=2)  # Symbolic matrix of squared distances
K_ij = (- D_ij / (2 * s**2) ).exp()    # Symbolic Gaussian kernel matrix
v    = K_ij @ p  # Genuine torch Tensor. (N,N) @ (N,D) = (N,D)

# Finally, compute the kernel norm H(q,p):
H = .5 * torch.dot( p.view(-1), v.view(-1) ) # .5 * <p,v>
# Display the computational graph in the figure below, annotated by hand:
make_dot(H, {'q':q, 'p':p}).render(view=True)
```



2.2.1 The key to high performances: symbolic `LazyTensors`

Current state of the art. This level of performance may surprise readers who grew accustomed to the limitations of tensor-centric frameworks. As discussed in Section 2.1, common knowledge in the machine learning community asserts that "kernel" computations can *not* scale to large point clouds with the CUDA backends of modern libraries: N-by-N kernel matrices stop fitting contiguously on the Device memory as soon as N exceeds some chip-dependent threshold in the `10,000`–`50,000` range.

Focusing on the key operation involved, the *kernel dot product* or *discrete convolution*:

$$\begin{array}{rcccl} \mathrm{KP} & : & \mathbb{R}^{\mathrm{M}\times\mathrm{D}} \times \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \times \mathbb{R}^{\mathrm{N}\times\mathrm{E}} & \to & \mathbb{R}^{\mathrm{M}\times\mathrm{E}} \\ & & ((x_i),(y_j),(b_j)) & \mapsto & (a_i) \;\text{ with }\; a_i = \sum_{j=1}^{\mathrm{N}} k(x_i,y_j)\, b_j \;, \end{array} \tag{2.26}$$

most authors are tempted to introduce an M-by-N kernel matrix $K_{ij} = k(x_i, y_j)$ and implement the operation above as a matrix dot product

$$\mathrm{KP}((x_i),(y_j),(b_j)) \;=\; (K_{ij}) \cdot (b_j)\;. \tag{2.27}$$

To accelerate computations, a flourishing literature has then focused on the construction of **low-rank approximations** of the linear operator (K_{ij}) as detailed Eq. (2.56). Common methods rely on random sampling schemes, multiscale decompositions of the data or take advantage of specific properties of the kernel function k – in our case, a convenient Gaussian blob.

Our focus: exact Map-Reduce computations. As discussed in detail in Section 2.3.3 (Future works) these approximation strategies have a long history and a clear intrinsic value. Nevertheless, acknowledging the fact that progresses can *also* be made through low-level software engineering, we decide to tackle this problem in a completely different way. Brushing aside the elegant but *inefficient* matrix decomposition of Eq. (2.27), the `KeOps` package directly optimizes the kernel sum of Eq. (2.26), understanding it as a Map-Reduce composition of the operators:

$$\begin{array}{rcccl} \mathrm{Map} & : & \mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{E}} & \to & \mathbb{R}^{\mathrm{E}} \\ & & (x,y,b) & \mapsto & k(x,y)\, b \end{array} \tag{2.28}$$

$$\text{and}\quad \begin{array}{rcccl} \mathrm{Reduce} & : & \mathbb{R}^{\mathrm{E}} \times \mathbb{R}^{\mathrm{E}} & \to & \mathbb{R}^{\mathrm{E}} \\ & & (a,a') & \mapsto & a + a' \end{array} \tag{2.29}$$

over the *indexing* indices $i \in [\![1,\mathrm{M}]\!]$ and *reduction* indices $j \in [\![1,\mathrm{N}]\!]$.

Parenthesis: are we re-packaging the wheel? This approach is common in the computer graphics literature but tends to be strictly limited to C++/CUDA programming guides: with an emphasis put on real-time rendering and explicit models, the graphics community never felt the need to develop high-level libraries that would be suited to machine learning research.

In this context, our scientific contribution does not lie in any new theorem or algorithm. Described in the next few pages, the tools on which our package relies (the backpropagation algorithm, online Map-Reduce CUDA schemes and symbolic variadic templating) are all well-known in their respective communities. Our original effort is to *combine* them in a versatile framework, endowed with a transparent interface and a comprehensive documentation: this allows them to reach a much wider audience and have, hopefully, a fertilizing impact.

A generic framework that suits the needs of researchers. The seminal Theano library combined the flexibility of high-level Python frameworks with a first-class support of convolutional architectures on the GPU. In the same vein, the KeOps package puts the spotlight on Map-Reduce schemes for (off-grid) sampled data, an algorithmic structure that we deem to be relevant in many fields that are related to data sciences and shape analysis.

Removing all the Python sugar coating, the workhorse of our library is a **Generic Reduction** (Genred) operator that supports a wide family of formulas. Let us assume that we have:

1. A collection $p^1, p^2, \ldots, p^P$ of vectors.
2. A collection $x_i^1, x_i^2, \ldots, x_i^X$ of vector sequences, indexed by an integer i in $[\![1, \mathrm{M}]\!]$.
3. A collection $y_j^1, y_j^2, \ldots, y_j^Y$ of vector sequences, indexed by an integer j in $[\![1, \mathrm{N}]\!]$.
4. A vector-valued formula $F(p^1, p^2, \ldots, x_i^1, x_i^2, \ldots, y_j^1, y_j^2, \ldots)$ on these input vectors.
5. A Reduction operation that may be a sum, an arg-min, a log-sum-exp, etc.

Then, referring to the p's as **parameters**, the x_i's as i**-variables** and the y_j's as j**-variables**, a single KeOps "Genred" call allows users to compute efficiently the expression

$$a_i = \operatorname*{Reduction}_{j=1,\ldots,\mathrm{N}} \left[F(p^1, p^2, \ldots, x_i^1, x_i^2, \ldots, y_j^1, y_j^2, \ldots) \right] \qquad \text{for } i = 1, \ldots, \mathrm{M}, \tag{2.30}$$

alongside its **derivatives** with respect to all variables and parameters. As discussed in Section 2.3.2 (Applications), this level of generality allows KeOps to handle K-nearest-neighbors classification, K-means clustering, Gaussian mixture model-fitting and many other tasks.

The LazyTensor abstraction. Implementation details are covered in the next few pages but probably won't interest most mathematicians. Wary of making users step outside of the convenient tensor-centric paradigm, we give a matrix-like interface to the unusual computation of Eq. (2.30): through a new "LazyTensor" wrapper for NumPy arrays and PyTorch tensors, users can specify formulas F without ever leaving the comfort of a NumPy-like interface.

As discussed page 38, KeOps LazyTensors embody the concept of "symbolic" tensors that are not sparse in the traditional sense, but can nevertheless be handled more efficiently than large M-by-N arrays if we use:

1. **A symbolic mathematical formula** F, the ".formula" attribute that is encoded as a well-formed string, manipulated with Python operations and parsed at reduction time.
2. **A collection of "small" data arrays** p, x **and** y, the ".variables" list of *parameters*, i- and j*-variables* that are needed to evaluate the formula F.

Coming back to the example of page 38, we can display the LazyTensor "K_ij" using:

```
>>> print(K_ij)
KeOps LazyTensor
    formula: Exp((Minus(Sum(Square((Var(0,3,0) - Var(1,3,1))))) / Var(2,1,2)))
    shape: (1000, 1000)
```

Here, the "Var(index, dimension, [i|j|parameter])" placeholders refer to the data arrays q_i, q_j and 1/(2*s**2) that are stored in the list of K_ij.variables. As we call a supported reduction operator such as the matrix dot-product "@" on K_ij, this information is fed to the Genred engine and a result is returned as a genuine, differentiable PyTorch tensor: things just work smoothly, with full support of *operator broadcasting* and *batch dimensions*.

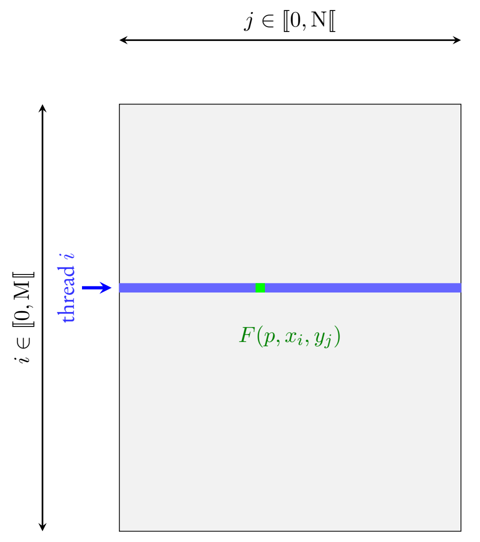
2.2.2 Efficient map-reduce schemes on the GPU

But how does KeOps handle symbolic formulas on the GPU? How can its routines outperform the CUDA backends of deep learning frameworks by such a wide margin?

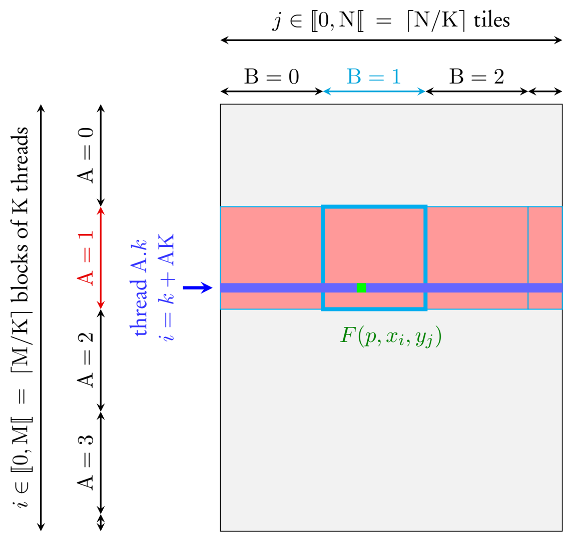
Architecture of the KeOps repository. To answer these questions, we need to dive into the mixed C++/Python/Matlab codebase of the KeOps package, whose structure can be summarized as follows:

- The **pykeops/** folder, with **common/**, **numpy/** and **torch/** subfolders contains our Python wrappers and relies on the fantastic PyBind11 library (Jakob et al., 2017).
- The **keopslab/** folder provides a collection of entry points for Matlab scripts.
- The **keops/** folder contains our C++ files and the associated compilation scripts. The generic KeOps engine that we are now about to discuss is implemented in the **core/** subfolder which contains:
    - The **link_autodiff.cpp** and **link_autodiff.cu** "main" C++ files, which define the methods that binding libraries may use to create high-level modules.
    - The **pack/** subfolder, which defines abstract types for lists and tuples within the C++ templating system. Using advanced concepts that were introduced with the C++11 revision, this file allows us to drive the nvcc compiler with declarative "variadic templating" and generate routines that manipulate an arbitrary number of parameters, i- and j-variables.
    - The **autodiff/** subfolder, which defines the primitives of the KeOps symbolic syntax: variables, abstract unary and binary operations, indexing methods.
    - The **mapreduce/GpuConv*_*.cu** CUDA files, which implement our massively parallel Map-Reduce schemes. **These files contain the core logic of the KeOps library.**
    - The **mapreduce/CpuConv*_*.cpp** C++ files, which implement simple Map-Reduce schemes using "for" loops. They can be used to test the correctness of our parallel implementations and provide a fall-back mode to users who do not have access to GPU chips on their machines.
    - The **reductions/** subfolder, which implement the supported Reduction operations: sum, arg-min, log-sum-exp, etc.
    - The **formulas/** subfolder, which implement the atomic operations that users may combine to define vector-valued formulas F. These headers define the parsing grammar for the ".formula" attribute of LazyTensors, which is understood as an abstract recursive type by the C++ compiler.

As evidenced here, the KeOps engine is heavily reliant on modern features of the C++ language: every time Genred encounters a new instance of Eq. (2.30) (up to the values of M, N and the data arrays which are free to change between every call), the string that specifies a generic formula is parsed by the compiler and a new ".dll" or ".so" shared object is generated before being executed on the relevant Python, R or Matlab tensors.



(a) Simple, ideal scheme: each thread i computes one of the a_i's by looping over the reduction index j and eating-up the values of F **on-the-fly**.


$j \in [\![0, \mathrm{N}[\![\ = \ \lceil \mathrm{N}/\mathrm{K} \rceil$ tiles
B = 0
B = 1
B = 2
$i \in [\![0, \mathrm{M}[\![\ = \ \lceil \mathrm{M}/\mathrm{K} \rceil$ blocks of K threads
A = 0
A = 1
A = 2
A = 3
thread A.k
$i = k + \mathrm{AK}$
$F(p, x_i, y_j)$

(b) Due to the importance of the Shared memory and block-wise memory accesses, (a) is cut in K-by-K tiles to ensure an optimal management of the y_j's.

Figure 2.4: The default 1D Map-Reduce scheme used by the `KeOps Genred` engine can be described as a simple loop over the reduction index "j" (a) that is optimized for GPU chips (b).

1D Map-Reduce scheme. The most important piece of code in the `KeOps` package is the one-dimensional, heavily templated Map-Reduce scheme that can be found in the `GpuConv1D.cu` CUDA file. Used as a default backend by the `Genred` operator, this distributed algorithm relies on principles that are exposed in the reference CUDA programming guide:

`docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#shared-memory`

In a nutshell, this scheme can be described as a **tiled "`for`" loop on the reduction index** j, parallelized over the sole index i – hence the "1D" denomination – which reduces the computed values of $F(p^1, \dots, x_i^1, \dots, y_j^1, \dots)$ **on-the-fly, without ever storing or sending them to the Device memory.**

The algorithm. More precisely, as illustrated in Figure 2.4 with the standard C++ convention of indexes that range "from 0 to $\mathrm{N} - 1$", we can decompose the instructions executed by our CUDA blocks as follows:

1. Each block of K threads is attributed an index A that ranges between 0 and $\lceil \mathrm{M}/\mathrm{K} \rceil - 1$. This number may exceed the physical number of blocks that run simultaneously on the GPU chip, but the `nvcc` compiler abstracts these technicalities away.
2. In every block, the K threads are indexed by an integer $k \in [\![0, \mathrm{K}[\![\ = [\![0, \mathrm{K} - 1]\!]$. The k-th thread is assigned to a fixed value of $i = k + \mathrm{AK}$. It loads the relevant values of $p^1, \dots, p^P$ and $x_i^1, \dots, x_i^X$ **from the Device to the Thread memory** or *register*, taking advantage of the speed-up for contiguous memory accesses: threads in the same block read neighboring memory adresses. Once again, the compiler handles the distribution of K virtual workers on a fixed number of physical CUDA cores.

3. Each thread is instructed to compute a single value $a_i = a_{k+\text{AK}}$ through a "`for`" loop on the values of j in $[\![1, \text{N}]\!]$. To minimize the transfers between the Device and Shared memories while maximizing the amount of contiguous memory accesses (as discussed page 28), this j-loop is cut in blocks of size K: **the large** M**-by-**N **plane of** (i, j) **indices is effectively cut in small** K**-by-**K **tiles**, following a standard CUDA procedure. Having initialized a temporary buffer "a" (in the Thread memory) to the neutral element of the Reduction – 0 if it is a sum, $+\infty$ if it is a minimum, etc. – the k-th thread of the block **loops** over values of the **tile index** B in $[\![0, \lceil \text{N}/\text{K} \rceil - 1]\!]$:

   a. Being assigned to an index $j_k = k + \text{BK}$, the worker loads the relevant values of $y^1_{j_k}, \dots, y^Y_{j_k}$ **from the Device to the Shared memory**. This task is performed in conjunction with the other threads of the block and comes down to a contiguous transfer of a slice "$j \in [\![\text{BK}, \text{BK} + \text{K}[\![$" of the y-data arrays from the "library" of the State department to the shared "office shelf" of Figure 2.3.c.

   b. The thread waits for latecomers and **synchronizes** with all workers in the same block: we don't want to start the computing job if some of the y_j's have not yet been loaded properly in the Shared memory!

   c. **Making a loop** over the reduction index j in $[\![\text{BK}, \text{BK} + \text{K}[\![$, the worker:

      i. Loads the relevant values of the y_j's **from the Shared to the Thread memory**.

      ii. **Computes** the value of $F(p^1, \dots, x^1_i, \dots, y^1_j, \dots)$, with all variables standing close to the computing core in the Thread memory.

      iii. **Reduces** this value onto the running buffer a, in the Thread memory.

   d. Once again, the thread **synchronizes** with the other workers.

4. Once this large outer loop has been completed, the buffer a associated to the k-th thread contains our final value $a_{k+\text{AK}}$. It is then saved **from the Thread to the Device memory** in an appropriate "output" array, alongside the other values in the "$i \in [\![\text{AK}, \text{AK} + \text{K}[\![$" range that have been computed by the block.

Performances. As most efficient CUDA programs, the algorithm presented above is pretty verbose: a full page of tedious memory transfers surrounds what is, at heart, a good old "`for-for`" loop. Crucially though, our efforts pay off: as evidenced by the benchmarks of Figure 2.12, `KeOps` typically provides a **`x30/x10,000` speed-up** when compared with tensorized `PyTorch-GPU/NumPy-CPU` implementations of the same kernel dot product, while keeping a **linear (instead of quadratic) memory footprint**.

This efficiency mostly comes down to the fact that instead of storing the M-by-N computed values of $F(p^1, \dots, x^1_i, \dots, y^1_j, \dots)$ in superfluous quadratic buffers (such as the "kernel matrix"), generating at least 2MN high-latency transfers between **the Thread and the Device memories**, `KeOps` maximizes the use of the Shared memory and consumes the relevant values of $F(p^1, \dots, x^1_i, \dots, y^1_j, \dots)$ on-the-spot, in the registers of the CUDA cores.

This level of performance could *not* have been achieved with high-level `Python` code: `PyTorch` and `TensorFlow` variables always refer to arrays that are stored in the **Device memory**. Writing C++ CUDA programs is the only way of getting an explicit access to the **Shared and Thread memories**. As discussed in Section 2.2.4, supporting *generic* formulas and reductions with `KeOps` thus required the implementation of a fully-fledged symbolic math engine, *within the CUDA framework*, that could be executed *inside* our loop at steps 3.c.ii-iii.

2.2.3 2D parallelization scheme, block-sparse reductions

2D scheme. The "`GPU_1D`" algorithm that we just presented is efficient whenever M is larger than the number of CUDA cores available on the chip: no thread stays idle. This is generally the case in shape analysis and data sciences, where the support of *batch processing* by `KeOps` allows programmers to fully saturate their GPUs with large input tensors.

Nevertheless, to provide cover for cases where the number of "indexing lines" M is much smaller than the size of the "reduction range" N, `KeOps` also implements a 2D Map-Reduce scheme in the **`GpuConv2D.cu`** CUDA file. Assigning the K-by-K *tiles* of Figure 2.4.b one-by-one to the CUDA blocks – instead of using a line-wise grouping method – this algorithm requires the allocation of intermediate buffers but makes sure that no block stays idle during the computation.

Complexity of the `KeOps` routines. Notwithstanding their clever management of memory, the "`GPU_1D`" and "`GPU_2D`" schemes have a **quadratic time complexity**: if M and N denote the numbers of i and j variables, the time needed to perform a generic reduction scales asymptotically in $O(\text{MN})$. This is most evident in the benchmark displayed Figure 2.12, where all the kernel coefficients:

$$K_{ij} \;=\; k(x_i, y_j) \;=\; \exp(-\|x_i - y_j\|^2 \,/\, 2\sigma^2) \qquad (2.31)$$

are computed to implement the Gaussian convolution of Eq. (2.26):

$$(a_i) \;=\; (K_{ij}) \cdot (b_i) \qquad \text{i.e.} \qquad a_i \;=\; \sum_{j=1}^{\text{N}} k(x_i, y_j) \cdot b_j \qquad \text{for } i \in [\![1, \text{M}]\!] \,. \qquad (2.32)$$

Can we do better? To break through this quadratic lower bound, a simple idea is to *skip some computations*, using a *sparsity prior* on the kernel matrix. For instance, we could decide to skip kernel computations when points x_i and y_j are far away from each other. But can we do so *efficiently*?

Sparsity on the CPU. On CPUs, a standard strategy is to use sparse matrices and encode our operators through lists of non-zero coefficients and indices. Schematically, this comes down to endowing each index $i \in [\![1, \text{M}]\!]$ with a set $J_i \subset [\![1, \text{N}]\!]$ of j-neighbors and to restrict ourselves to the computation of:

$$a_i \;=\; \sum_{j \in J_i} k(x_i, y_j) \cdot b_j, \qquad \text{for } i \in [\![1, \text{M}]\!] \,. \qquad (2.33)$$

This approach is well suited to matrices which only have a handful of nonzero coefficients per line, such as the intrinsic Laplacian of a 3D mesh. But on large, densely connected problems, sparse encodings run into a major issue: as they rely on **non-contiguous** memory accesses, they scale poorly on GPUs.

Block-sparse reductions. As explained page 28, GPU chips are wired to rely on *coalesced memory operations* which load blocks of dozens of contiguous bytes at once. Instead of allowing the use of arbitrary indexing sets J_i for all lines of our sparse kernel matrix, we should thus restrict ourselves to computations of the form:

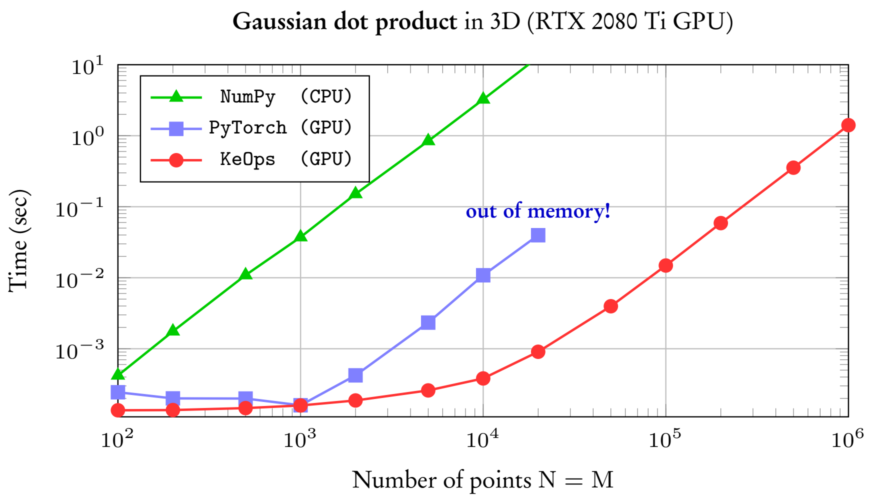
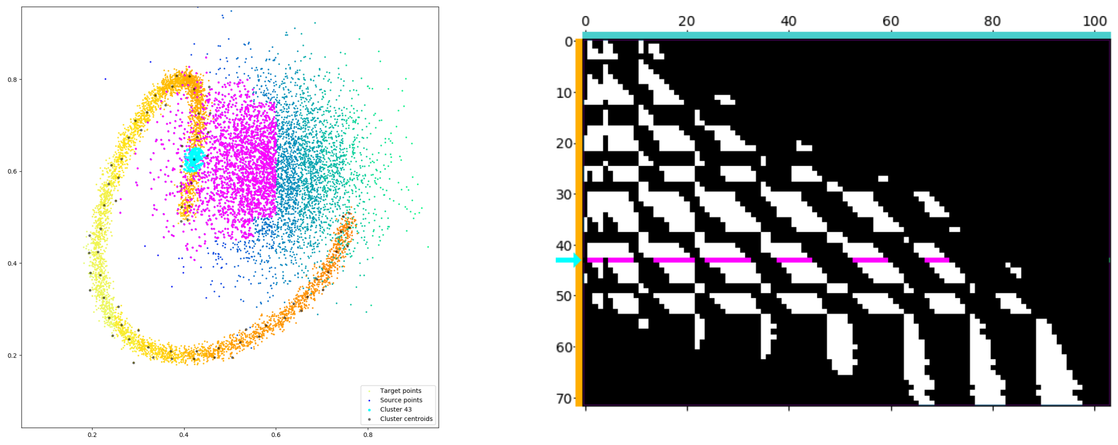


Figure 2.5: Benchmarking `NumPy`, `PyTorch` and `KeOps` on the Gaussian kernel product of Eq. (2.26) in dimension D = 3 with a scalar signal of dimension E = 1. As we let the number N = M of points increase, we observe a `10,000` and `30` to `1` ratio in favor of the `KeOps` routines, which also keep a **linear memory footprint**. As evidenced here, the default "`GPU_1D`" backend of `KeOps` is extremely competitive for datasets of 10^3 to 10^6 samples. With a GPU chip bought for $\sim$1,500$ in 2020 – the GeForce RTX 2080 Ti – large 10^5-by-10^5 quadratic operations can now be performed in around 10ms, **without making any approximation** and with a native support of **automatic differentiation**. Extended benchmarks are provided page 56.



(a) Clouds (x_i) – spiral – and (y_j) – Gaussian.

(b) Coarse boolean mask of cluster-to-cluster interactions.

Figure 2.6: Illustrating **block-sparse reductions** with 2D point clouds. When using an M-by-N "kernel" matrix to compute an interaction term between two datasets, a common approximation strategy is to skip terms which correspond to clusters of points that are far away from each other. Through a set of helper routines and optional arguments, `KeOps` allows users to implement these pruning strategies efficiently, on the GPU. (a) Putting our points in square bins, we compute the centroid of each cluster. Simple thresholds on centroid-to-centroid distances allow us to decide that the 43rd "cyan" cluster of target points (x_i) should only interact with neighboring cells of source points (y_j), highlighted in magenta, etc. (b) In practice, this decision is encoded in a coarse boolean matrix that is processed by `KeOps`, with each line (*resp.* column) corresponding to a cluster of x (*resp.* y) variables. Here, we higlight the 43rd line of our mask which corresponds to the cyan-magenta points of (a).

$$a_i \;=\; \sum_{l=1}^{S_q} \sum_{j=\text{start}_l^q}^{\text{end}_l^q-1} k(x_i, y_j) \cdot b_j, \qquad \text{for } i \in [\![\text{start}_q, \text{end}_q[\![\;\text{ and }\; q \in [\![1, \text{Q}]\!]\,, \tag{2.34}$$

where:

1. The $[\![\text{start}_q, \text{end}_q[\![$ intervals form a **partition** of the set of i-indices $[\![1, \text{M}]\!]$:

$$[\![1, \text{M}]\!] \;=\; \bigsqcup_{q=1}^{\text{Q}} [\![\text{start}_q, \text{end}_q[\![\ . \tag{2.35}$$

2. For every segment $q \in [\![1, \text{Q}]\!]$, the S_q intervals $[\![\text{start}_l^q, \text{end}_l^q[\![$ encode a set of *neighbors* as a **finite collection of contiguous ranges of indices**:

$$\forall\, i \in [\![\text{start}_q, \text{end}_q[\![\,,\ J_i \;=\; \bigsqcup_{l=1}^{S_q} [\![\text{start}_l^q, \text{end}_l^q[\![\ . \tag{2.36}$$

By encoding our sparsity patterns as **block-wise binary masks** made up of tiles

$$T_l^q \;=\; [\![\text{start}_q, \text{end}_q[\![\times [\![\text{start}_l^q, \text{end}_l^q[\![\;\subset\; [\![1, \text{M}]\!] \times [\![1, \text{N}]\!]\ , \tag{2.37}$$

we can leverage coalesced memory operations for maximum efficiency on the GPU. As long as our index ranges are wider than the CUDA blocks, we should get close to optimal performances.

Going further. This scheme can be generalized to *generic* formulas and reductions. For reductions with respect to the i axis, we simply have to define *transposed* tiles

$$U_l^q \;=\; [\![\text{start}_l^q, \text{end}_l^q[\![\times [\![\text{start}_q, \text{end}_q[\![\;\;\subset\; [\![1, \text{M}]\!] \times [\![1, \text{N}]\!] \tag{2.38}$$

and restrict ourselves to computations of the form:

$$b_j \;=\; \sum_{l=1}^{S_q} \sum_{i=\text{start}_l^q}^{\text{end}_l^q-1} k(x_i, y_j) \cdot a_i, \qquad \text{for } j \in [\![\text{start}_q, \text{end}_q[\![\text{ and } q \in [\![1, \text{Q}]\!]\ . \tag{2.39}$$

A decent trade-off. This block-wise approach to sparse reductions may seem a bit too coarse, as some negligible coefficients get computed with little to no impact on the final result... But in practice, the **GPU speed-ups** on contiguous memory operations more than make up for it: implemented in the `GpuConv1D_ranges.cu` CUDA file, our block-sparse Map-Reduce scheme is the workhorse of the *multiscale Sinkhorn algorithm* showcased in Section 3.3.3.

As explained on our website, the main user interface for `KeOps` block-sparse reduction is an optional "`.ranges`" attribute for `LazyTensors` which encodes block-sparsity masks. In practice, as illustrated in Figure 2.6, helper routines allow users to specify tiled sparsity patterns from *clustered* arrays of samples x_i, y_j and coarse *cluster-to-cluster* boolean matrices. Implementing Barnes-Hut-like strategies (Barnes and Hut, 1986) and other approximation rules is thus relatively easy, up to a preliminary sorting pass which ensures that all clusters are stored *contiguously in memory*.

2.2.4 Generic formulas, automatic differentiation

The three previous sections have higlighted the need for efficient Map-Reduce GPU routines in data sciences. To complete our guided tour of the inner workings of the KeOps library, we now explain how *generic* reductions and formulas are encoded within our C++ codebase.

Developing versatile CUDA libraries. As discussed at the end of Section 2.2.2, **supporting generic reductions and automatic differentiation within KeOps was a daunting challenge**. Let us briefly explain why.

The overwhelming majority of deep learning frameworks and contributed packages follow a simple **"one Python operation = one CUDA program"** development paradigm – with the notable exception of projects such as TensorComprehensions (Vasilache et al., 2018). Following the usual practice, each PyTorch operation is linked to a pre-compiled binary program that implements a *specific* computation: a sum, a matrix-vector product or whatever. With every Python instruction, these routines are simply executed on the ".data" attributes (raw C++ arrays) of the relevant variables.

Back in 2017, in early KeOps releases, this is how we first implemented the Gaussian kernel dot product and its derivatives of order 1 and 2 : with explicit "gaussian_dot.cu", "gaussian_dot_grad_x.cu", "gaussian_dot_grad_xx.cu" CUDA files. Once the basics are understood, writing by hand an *ad hoc* CUDA program for every instance of Eq. (2.30) is not too difficult.

The KeOps symbolic engine. On the other hand, allowing KeOps users to define and reduce their *own* formulas without having to write a single line of C++ code is a *much* more challenging target. One way or another, the specification of a new *symbolic* computation in a Python (LazyTensor) script has to result in the injection of lightweight, efficient C++ code right *inside* the CUDA loop of page 43, at steps 3.c.ii-iii.

As of 2020, supporting the **dynamic** generation of efficient CUDA code for deep learning scripts is the major challenge that the TensorFlow (Google) and PyTorch (Facebook) development teams strive to tackle. Usually referred to as Just In Time (JIT) compilation, this feature is now partially supported by the latest versions of PyTorch and TensorFlow, through an "Accelerated Linear Algebra" (XLA) backend. Asking users to accept some restrictions on the structure of their Python scripts, experimental JIT compilers attempt to *fuse* consecutive CUDA routines in order to optimize the use of resources and buffers.

With limited means – three part-time developers, mathematicians by trade – KeOps achieves this target in the controlled setting of symbolic Map-Reduce schemes, defined by Eq. (2.30). The well-documented constraints that are put on the development of academic software projects guide our main design decisions:

- To ensure the **portability** and **long-term maintainability** of the KeOps library, we perform the syntactic analysis of user-defined formulas without any dependency on external "math processing" engines.
- To make sure that KeOps is able to reach users outside of our own microcosm, its core logic must be written in **pure C++** – without any hard dependency on Python. Today, most KeOps users access it through its PyTorch and NumPy interfaces... But future R or Julia bindings may well have a more lasting impact.

Working with variadic templates. To achieve our goals whilst abiding by these constraints, we chose to rely on the power of modern C++/CUDA compilers. Leveraging expressive meta-programming instructions that were introduced by the C++11 revision, the **`keops/core/`** folder effectively implements a small but robust math engine **within the C++ templating system**.

Letting a general-purpose tool such as `nvcc` or `clang` handle the parsing and (most of) the low-level optimization of our code may look like a rash decision. But in practice, the graph-based optimizers of modern C++ compilers are now so efficient that we never felt limited in any way: with the help of a few simplification rules – encoded as template specializations – the binaries generated by the `KeOps` generic engine perform just as well as clever hand-written CUDA kernels. With foundations that rely on standard, well-tested tools that are now *too big to fail*, we expect to be able to maintain `KeOps` for many years to come.

Key files. As detailed in page 41, our parsing grammar for symbolic formulas is described in terms of abstract C++ types in the **`keops/core/formulas/*/*.h`** headers. These files provide a comprehensive list of mathematical operators and rely on the primitives implemented in the **`keops/core/pack/`** and **`keops/core/autodiff/`** subfolders: abstract unary and binary operators, tuples of variables and parameters, integer constants, indexing methods.

In practice. To give a glimpse of how `KeOps` works under the hood, let us present an excerpt from the **`formulas/maths/Log.h`** header – the declaration of the `Log<...>` operator:

```cpp
// 1. Declare a new Unary Operation - the pointwise logarithm:
template < class F >
struct Log : UnaryOp<Log,F> {

    // 2. Declare a new attribute: dimension of Log(F) = dimension of F:
    static const int DIM = F::DIM;

    // 3. Utility method: pointwise Logarithm should be displayed as "Log":
    static void PrintIdString(std::stringstream& str) { str << "Log"; }

    // 4. Actual C++ implementation of our operator:
    static HOST_DEVICE INLINE void Operation(__TYPE__ *out, __TYPE__ *outF) {
        for(int k = 0; k < DIM; k++)
            out[k] = log( outF[k] );
    }

    // 5. Define a new alias for the "backward" operator of F...
    template < class V, class GRADIN >
    using DiffTF = typename F::template DiffT<V,GRADIN>;

    // 6. And use it to implement the "backward" of Log(F):
    template < class V, class GRADIN >
    using DiffT = DiffTF<V, Mult< Inv<F>, GRADIN> >;
};
```

As evidenced here, the implementation of a new operator goes through six compulsory steps:

1. The **declaration** of a new operation as an instance of the abstract `UnaryOp` or `BinaryOp` templates. These are defined in the **`keops/core/autodiff.h`** header with a set of standard methods and attributes. The operand `F` of `Log<F>` is an arbitrary formula, recursively encoded as a templated structure.

2. The specification of a few standard **attributes**. Here, the dimension of the vector `Log(F)` – accessed as `Log<F>::DIM` in typical C++ fashion – is equal to that of `F`. Our logarithm is applied pointwise and does not affect the shape of the underlying vector.

3. The specification of some **utility methods**. Here, the string identifier `PrintIdString` may be used to access the formula that is encoded within any `KeOps` C++ template.

4. The actual **implementation** of our operator, that is to be executed within the Thread memory of each CUDA core. As specified in the abstract definition of `UnaryOp`, the inline method `Operation` takes as input a C++ array `outF`, the vector-valued output of our operand `F`. It computes the pointwise logarithms using a standard CUDA routine and stores them in a new `out` buffer of size `Log<F>::DIM`. In practice, modern C++ compilers may simplify this operation as an in-place modification of the values stored in `outF`.

5. **Prepare the chain rule** by defining an alias for the adjoint "backward" operator of the operand `F` with respect to an arbitrary differentiation variable `V`. As explained in Eq. (2.22), the new operator $\mathrm{d}_{\mathtt{V}}^{\top} F$ is a formal expression that takes as input the variables "$x = (p^1, \dots, x_i^1, \dots, y_j^1, \dots)$" of `F` and a new vector "a" of size `F::DIM`, the gradient vector `GRADIN` or "x_i^*" that is backpropagated through the whole computational graph. Understood as the adjoint or "transpose" of the differential of `F`, the application of this operator is encoded within `KeOps` as a new templated expression `F::DiffT<V,GRADIN>` that should implement the computation of $\mathrm{d}_{\mathtt{V}}^{\top} F \cdot \mathtt{GRADIN}$.

6. **Implement the chain rule** recursively, using the templated expression above: `DiffTF = F::DiffT<V,GRADIN>`. Here, the C++ declaration:

   ```
   Log<F>::DiffT<V,GRADIN> = F::DiffT<V, Mult< Inv<F>, GRADIN> > (2.40)
   ```

   simply encodes the well-known fact that with pointwise computations,

$$\mathrm{d}_{\mathtt{V}}^{\top} \left[\log \circ F \right](p, x_i, y_j) \cdot \mathtt{GRADIN} = \mathrm{d}_{\mathtt{V}}^{\top} F(p, x_i, y_j) \cdot \frac{\mathtt{GRADIN}}{F(p, x_i, y_j)} . \qquad (2.41)$$

Contributing with a new operation. Advanced users may wish to extend the existing engine with home-made operators, injecting their C++ code within the `KeOps` Map-Reduce kernel. Doing so is now relatively easy: having implemented a custom instance of the `UnaryOp` or `BinaryOp` templates in a new **`keops/core/formulas/*/*.h`** header, contributors should simply remember to add their file to the list of includes **`keops/keops_includes.h`** and write a `LazyTensor` method in the **`pykeops/common/lazy_tensor.py`** module. To get merged in the main `KeOps` repository, which is hosted on GitHub, writing a simple unit test in the **`pykeops/test/`** folder and a description in the pull request should then be enough.

Reductions. Following the same design principles, Reduction operators are implemented in the **`keops/core/reductions/*.h`** headers. Taking as input an arbitrary symbolic formula `F`, `Reduction<F>` templates encode Map-Reduce schemes in the mould of Eq. (2.30) and implement a few standard routines. In the case of the simple **Sum** reduction (**`sum.h`** header), these can be described as:

1. An `InitializeReduction` method, which fills up the running buffer "a" of page 43 – a vector of size `F::DIM` – with zeros before the start of the loop on the reduction index j.
2. A `ReducePair` method, which takes as input a pointer to the running buffer a, a pointer to the result $F_{i,j} = F(p^1, \dots, x_i^1, \dots, y_j^1, \dots)$ and implements the in-place reduction:
$$a \leftarrow a + F_{i,j} \,. \tag{2.42}$$
3. A `FinalizeOutput` method, which post-processes the buffer a before saving its value in the output array. This is a useful step for **argmin**-like reductions; but in the case of the **sum**, no post-processing is needed.

The online Log-Sum-Exp trick. `KeOps` also supports accurate summation schemes on floating-point numbers (Kahan, 1965). Going further, the **`max_sumshiftexp.h`** header implements an online version of the **Log-Sum-Exp** trick: a factorization of the maximum in:

$$\log \sum_{j=1}^{\mathrm{N}} \exp(F_{i,j}) \;=\; m_i \;+\; \log \sum_{j=1}^{\mathrm{N}} \exp(F_{i,j} \,-\, m_i), \;\; \text{with } m_i \;=\; \max_{j=1}^{\mathrm{N}} F_{i,j} \tag{2.43}$$

that ensures the computation of this important quantity – the linchpin of maximum likelihood estimators and entropic optimal transport solvers – without numeric overflows.

Merging the content of our C++ header and of the `Python` post-processing step implemented in **`pykeops/common/operations.py`**, assuming that $F_{i,j} = F(p^1, \dots, x_i^1, \dots, y_j^1, \dots)$ is a scalar quantity, we may describe its behaviour as follows:

1. The `InitializeReduction` method ensures that our running buffer a is a vector of size 2 that encodes the current value of the inner summation as an explicit (exponent, mantissa) or "(maximum, residual)" pair of `float` numbers: at any stage of the computation, the pair (m, r) encodes the positive number $e^m \cdot r$ with the required precision. We initially set the value of a to $(-\infty, 0) \simeq e^{-\infty} \cdot 0$.
2. The `ReducePair` method takes as input a pointer to the result $F_{i,j}$ of the computation, a pointer to the running buffer $a = (m, r) \simeq e^m \cdot r$ and implements the in-place update:
$$(m, r) \;\leftarrow\; \begin{cases} (\, m \;,\; r + \quad e^{F_{i,j} - m}) & \text{if } m \geqslant F_{i,j} \\ (F_{i,j},\; 1 + r \cdot e^{m - F_{i,j}}) & \text{otherwise.} \end{cases} \tag{2.44}$$
This is a numerically stable way of writing the sum reduction:
$$e^m \cdot r \;\leftarrow\; e^m \cdot r \,+\, e^{F_{i,j}} \;=\; \begin{cases} e^{m} \;\cdot (r + \quad e^{F_{i,j} - m}) & \text{if } m \geqslant F_{i,j} \\ e^{F_{i,j}} \cdot (1 + r \cdot e^{m - F_{i,j}}) & \text{otherwise.} \end{cases} \tag{2.45}$$
3. `FinalizeOutput` post-processes the buffer $a = (m, r) \simeq e^m \cdot r$ by applying the final "log" operation: it returns a value of $m + \log(r)$ for the full reduction.

Backpropagation through a `KeOps` call. Last but not least, **KeOps fully supports automatic differentiation**. Most of the magic required is implemented by the `F::DiffT<V,GRADIN>` attributes of `KeOps` formulas and reductions, as discussed in previous pages.

Then, to implement the `PyTorch backward` of the `KeOps Genred` operator, we simply have to remember that if $(g_i) \in \mathbb{R}^{\mathrm{M}\times\mathrm{E}}$ is a "gradient to backpropagate" with respect to the output $(a_i) \in \mathbb{R}^{\mathrm{M}\times\mathrm{E}}$ of a `Genred` call with a **Sum reduction**, we can write that for all variations $(\delta p, \delta x_i, \delta y_j)$ of the parameters, i- and j-variables, at order 1:

$$\Big\langle \sum_{j=1}^{\mathrm{N}} F(p+\delta p, x_i+\delta x_i, y_j+\delta y_j) - F(p,x_i,y_j)\ ,\ g_i \Big\rangle_{\mathbb{R}^{\mathrm{M}\times E}} \tag{2.46}$$

$$= \sum_{i=1}^{\mathrm{M}}\sum_{j=1}^{\mathrm{N}} \Big(\Big\langle \mathrm{d}_p^\top F(\dots)\cdot g_i, \delta p \Big\rangle + \Big\langle \mathrm{d}_{x_i}^\top F(\dots)\cdot g_i, \delta x_i \Big\rangle + \langle \mathrm{d}_{y_j}^\top F(\dots)\cdot g_i, \delta y_j \rangle \Big)\,. \tag{2.47}$$

Consequently, performing the appropriate permutations of sums:

$$\mathrm{d}_{x_i}^\top \Big[\sum_{j=1}^{\mathrm{N}} F(p,x_i,y_j)\Big]\cdot(g_i) \;=\; \sum_{j=1}^{\mathrm{N}} \Big(\mathrm{d}_{x_i}^\top \Big[F(p,x_i,y_j)\Big]\cdot g_i\Big)\,, \tag{2.48}$$

$$\mathrm{d}_{y_j}^\top \Big[\sum_{j=1}^{\mathrm{N}} F(p,x_i,y_j)\Big]\cdot(g_i) \;=\; \sum_{i=1}^{\mathrm{M}} \Big(\mathrm{d}_{y_j}^\top \Big[F(p,x_i,y_j)\Big]\cdot g_i\Big)\,, \tag{2.49}$$

$$\mathrm{d}_{p}^\top \Big[\sum_{j=1}^{\mathrm{N}} F(p,x_i,y_j)\Big]\cdot(g_i) \;=\; \sum_{i=1}^{\mathrm{M}}\sum_{j=1}^{\mathrm{N}} \Big(\mathrm{d}_{p}^\top \Big[F(p,x_i,y_j)\Big]\cdot g_i\Big)\,. \tag{2.50}$$

Similarly, when (a_i) is given through a **Log-Sum-Exp reduction**:

$$a_i \;=\; \log \sum_{j=1}^{\mathrm{N}} \exp F(p,x_i,y_j)\,, \tag{2.51}$$

straightforward computations show that:

$$\mathrm{d}_{x_i}^\top \Big[\log\sum_{j=1}^{\mathrm{N}} \exp F(p,x_i,y_j)\Big]\cdot(g_i) \;=\; \sum_{j=1}^{\mathrm{N}} e^{F(p,x_i,y_j)-a_i}\cdot \Big(\mathrm{d}_{x_i}^\top \Big[F(p,x_i,y_j)\Big]\cdot g_i\Big)\,, \tag{2.52}$$

$$\mathrm{d}_{y_j}^\top \Big[\log\sum_{j=1}^{\mathrm{N}} \exp F(p,x_i,y_j)\Big]\cdot(g_i) \;=\; \sum_{i=1}^{\mathrm{M}} e^{F(p,x_i,y_j)-a_i}\cdot \Big(\mathrm{d}_{y_j}^\top \Big[F(p,x_i,y_j)\Big]\cdot g_i\Big)\,, \tag{2.53}$$

$$\mathrm{d}_{p}^\top \Big[\log\sum_{j=1}^{\mathrm{N}} \exp F(p,x_i,y_j)\Big]\cdot(g_i) \;=\; \sum_{i=1}^{\mathrm{M}}\sum_{j=1}^{\mathrm{N}} e^{F(p,x_i,y_j)-a_i}\cdot \Big(\mathrm{d}_{p}^\top \Big[F(p,x_i,y_j)\Big]\cdot g_i\Big)\,. \tag{2.54}$$

In other words, a backward pass through a `Genred` call that involves a Sum or a Log-Sum-Exp reduction can always be written as a symbolic Map-Reduce computation, fitting Eq. (2.30).

Bootstrapping derivatives of arbitrary order. Applying these commutation rules between the differential operator $\mathrm{d}_{\mathtt{V}}^\top$ and the Sum or Log-Sum-Exp reductions, the **`pykeops/torch/generic/generic_red.py`** module provides full compatibility between `KeOps LazyTensors` and the `torch.autograd` package. Thanks to recursive calls to the `Genred` operator and to our symbolic math engine, everything works just fine – even high-order derivatives.

2.3 The PyKeops package: a powerful tool with a transparent interface

The previous section uncovered the inner workings of the LazyTensor module. After fifteen pages of technical derivations, time has now come to reap the reward of this year-long investment in low-level software engineering and present the user interface of the KeOps package.

Our starting point: computational anatomy. Back in 2017, we started working on the KeOps library to give to our colleagues of the (medical) shape processing community an easy access to the CUDA routines of the fShapes toolkit (Charlier et al., 2017a) – a Matlab toolbox that relies extensively on Gaussian kernel products. This initial target was reached pretty quickly: today, the reference Deformetrica software (Bône et al., 2018) – maintained by the Aramis Inria team at the ICM Institute for Brain and Spinal Cord, www.deformetrica.org – is fully reliant on the PyTorch+KeOps framework. Most of our collaborators use one of the KeOps bindings to implement their shape processing pipelines.

As discussed in Section 5.2.3, modern "LDDMM" codebases for statistical shape modelling are ten times slimmer (and easier to maintain!) than they were just three years ago: graduate students can now get started in days instead of months. We expect to witness many progresses in the field as research teams get relieved from the burden of low-level C++ development. As far as our specialized community of mathematicians is concerned, with more than 1,000 downloads per month on the PyPi repository, KeOps is already a success.

Reaching a wider audience. In 2018-2019, after several interactions with colleagues in machine learning and optimal transport conferences, we realized that our generic Map-Reduce engine could be used to solve problems that go way beyond neuro-anatomy. Provided that some effort was made to improve the general user experience, KeOps LazyTensors could be a game changer for engineers and researchers in many applied fields.

Today, after months of patient re-packaging and documentation, KeOps is a fully-fledged open source library (MIT License) whose development can be tracked on GitHub (github.com/getkeops/keops). **It fully supports Matlab, R, NumPy and PyTorch.** The Python bindings are easy to install through the PyPi repository (pip install pykeops), with numerous examples available on our website:

www.kernel-operations.io

2.3.1 Supported reductions and formulas

As discussed in our introductory tutorials, LazyTensors can be built from any valid NumPy array or PyTorch tensor and support a wide range of mathematical operations. Generic, broadcasted computations define valid programs:

```python
import torch
from pykeops.torch import LazyTensor

A, B, M, N, D = 7, 3, 100000, 200000, 10
x_i = LazyTensor( torch.randn(A, B, M, 1, D) )  # "i"-variable
l_i = LazyTensor( torch.randn(1, 1, M, 1, D) )  # "i"-variable
y_j = LazyTensor( torch.randn(1, B, 1, N, D) )  # "j"-variable
s   = LazyTensor( torch.rand( A, 1, 1, 1, 1) )  # parameter
```

```
F_ij = (x_i ** 1.5 + y_j / l_i).cos()          # Algebraic expression
F_ij = F_ij - (x_i | y_j)                       # Scalar product
F_ij = F_ij + (x_i[:,:,:,:,2] * s.relu() * y_j)  # Indexing, ReLU activation

a_j = F_ij.sum(dim=2)  # a_j.shape = [7, 3, 200000, 10]
```

LazyTensors fully support automatic differentiation – up to arbitrary orders – as well as a decent collection of reduction operations. On top of the .sum(), "@" (matrix multiplication) and .logsumexp() operators which have already been discussed in depth, users may rely on .min(), .argmin(), .min_argmin(), .max(), .argmax(), .max_argmax(), .Kmin(K=...), .argKmin(K=...) or .min_argKmin(K=...) methods to implement their algorithms. We refer interested readers to our website, where tutorials and examples cover most use cases.

Linear solver. KeOps provides support for the resolution of large "mathematical" linear systems – a critical operation in geology (Kriging), imaging (splines), statistics (Gaussian process regression) and data sciences (kernel regression). Assuming that the LazyTensor "K_xx" encodes a symmetric, positive definite matrix K_{xx}, the .solve() method:

```
a_i = K_xx.solve(b_i, alpha=alpha)
```

returns the solution:

$$a^\star \;=\; \operatorname*{argmin}_a \|\,(\alpha\,\mathrm{Id} + K_{xx})\,a \;-\; b\,\|_2^2 \;=\; (\alpha\,\mathrm{Id}\;+\;K_{xx})^{-1}b\,, \tag{2.55}$$

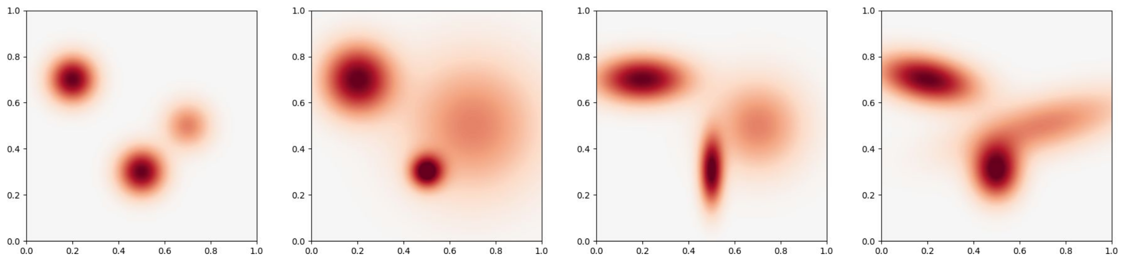
of the linear system "$(\alpha\,\mathrm{Id}\;+\;K_{xx})\,a = b$", computed with a conjugate gradient scheme.

Using KeOps as a backend for high-level libraries. Going further, as discussed in Figure 2.9.c and Figure 2.11.b, LazyTensors can be neatly interfaced with the high-quality solvers of the Scipy (Jones et al., 2001) and GPytorch (Gardner et al., 2018) libraries. Preliminary results with the maintainers of the latter already show remarkable improvements to the state-of-the-art: re-running the benchmarks of (Wang et al., 2019) with a new KeOps backend, exact Gaussian process regressions that took **7 hours** to train on a cluster of 8 top-drawer V100 GPUs (3DRoad dataset, N = 278,319, D = 3) can now be performed in **15 minutes** on a single gaming chip, the GeForce RTX 2080 Ti.

2.3.2 Gallery of examples

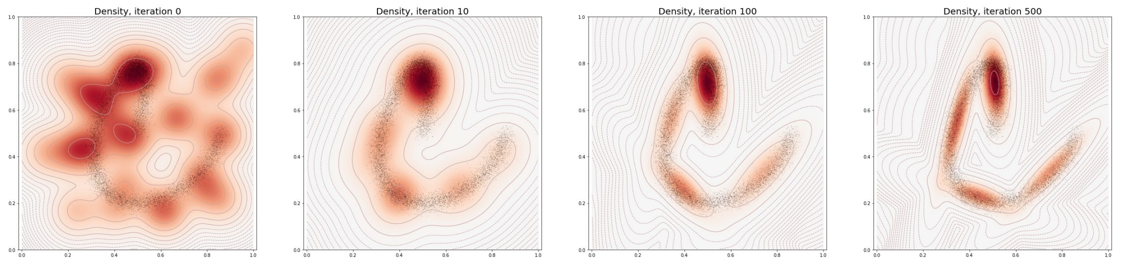
Displayed on our website, in Figures 2.7 to 2.11 and in the subsequent chapters of this manuscript, our gallery of tutorials showcases an eclectic collection of applications to machine learning, statistics, optimal transport theory and computational anatomy.

We carry on working towards a closer integration with the Python scientific stack (Van Der Walt et al., 2011; Hunter, 2007; Pedregosa et al., 2011) and plan to implement Julia bindings in months to come. By making our routines freely available to the general public, we hope to help the applied maths community to catch up with the state-of-the-art in computer science: in 2020, bruteforce quadratic algorithms should have no problem scaling up to millions of samples in minutes; clever approximation schemes are only needed if users intend to perform real-time analysis or scale to Gigabytes of data.



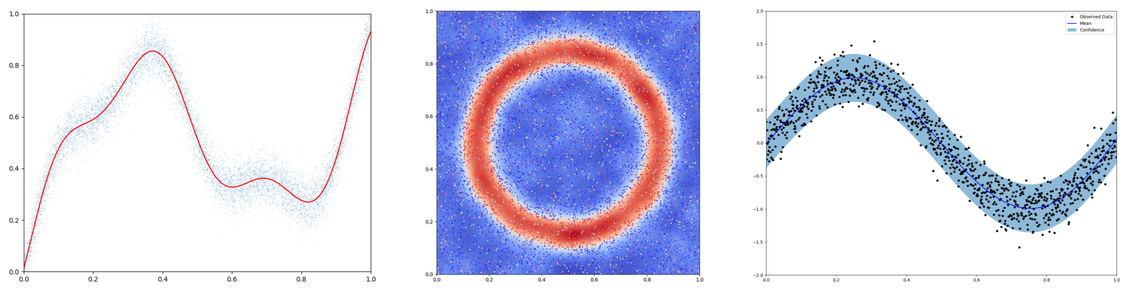
(a) $\sigma^{-2} \in \mathbb{R}$ (b) $(\sigma_j^{-2}) \in \mathbb{R}^{\mathrm{N}}$ (c) $(\Sigma_j^{-1}) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ (d) $(\Sigma_j^{-1}) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}^2}$

Figure 2.7: Thanks to an embryonic support of tensor variables (and not just vectors), `KeOps` provides a clean interface for the specification of **generic mixture models**. Depending on the shape of the "inverse covariance matrix" Σ^{-1} of multivariate Gaussian laws, users may specify: (a) uniform kernel products as in Eq. (2.32); (b) kernel products with a *scalar radius* σ_j that depends on the source point y_j; (c) kernel products with a *diagonal* covariance matrix that varies with the source point y_j; (d) kernel products with arbitrary *symmetric positive definite* covariance matrices that change with y_j.



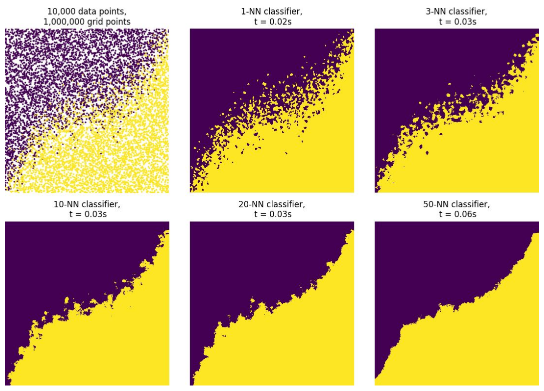
(a) `it = 0` (b) `it = 10` (c) `it = 100` (d) `it = 500`

Figure 2.8: Combining the expressive syntax of Fig. 2.7 with the power of automatic differentiation, `KeOps` users can fit generic mixture models to large datasets using off-the-shelf optimization schemes. In this example, a **Gaussian mixture model with a sparsity-inducing penalty** on the class weights is fitted to a 2D point cloud using the Adam optimizer (Kingma and Ba, 2014).

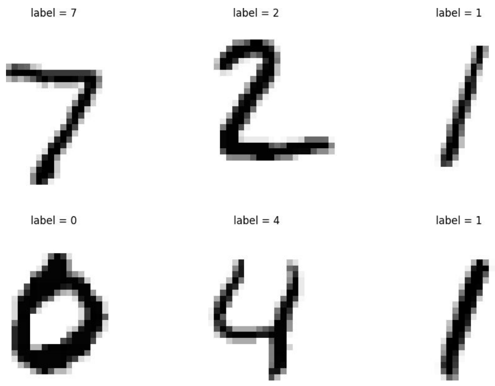


(a) Kernel regression in 1D. (b) Kriging in 2D. (c) `GPytorch` integration.

Figure 2.9: Kriging, also known as **kernel** or **Gaussian process regression** is a fundamental tool in data sciences that relies on the resolution of large kernel linear systems – see page 89. (a-b) Out-of-the-box, `KeOps` provides a conjugate gradient solver for `LazyTensors` that allows users to scale up to datasets with 10^4 to 10^6 high-dimensional samples in seconds or minutes. (c) Going further, we are working with the authors of the `GPytorch` library (Gardner et al., 2018), and provide a `KeOps` backend for their collection of efficient pre-conditioned solvers.

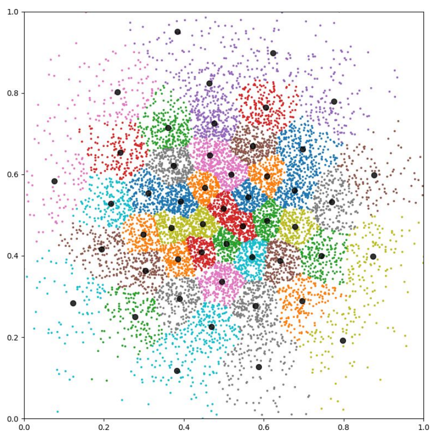


(a) K-NN classifier in 2D.

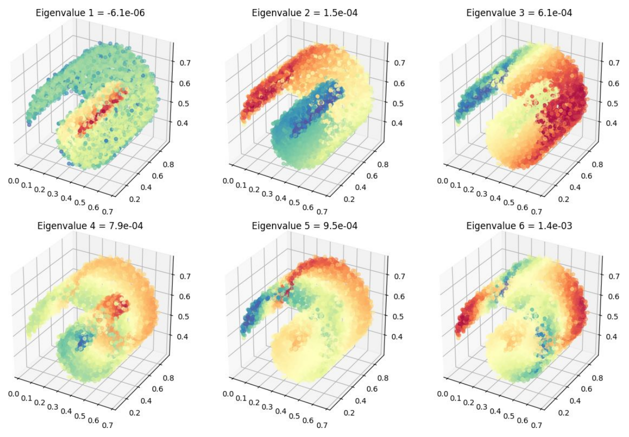


(b) K-NN on the MNIST dataset.

Figure 2.10: Supported by `KeOps LazyTensors`, the `.argKmin(K=...)` reduction allows users to implement bruteforce **K-nearest neighbors classification** in no more than four lines of code. (a) Thanks to the linear memory footprint of `KeOps` routines, users may compute large 1,000,000-by-10,000 queries without having to worry about memory overflows. (b) The `KeOps` engine has no problem scaling up to **high-dimensional feature spaces**. Performing a 3-NN classification on the full MNIST dataset (LeCun and Cortes, 1998) – a 10,000-by-60,000 NN-search in dimension 728 – yields a 3% error rate and takes roughly 10s on a RTX 2080 Ti chip, *without any pre-processing*.

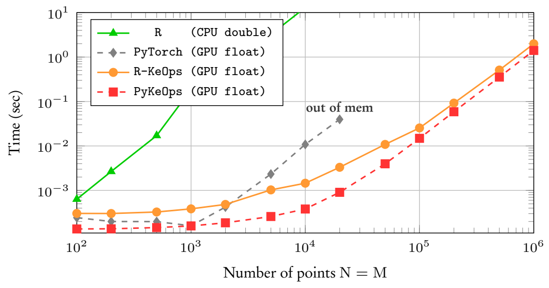


(a) K-means in 2D.

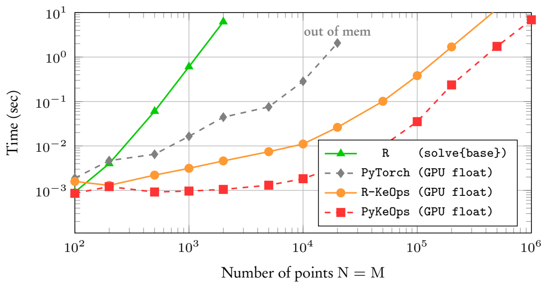


(b) Spectral coordinates in 3D.

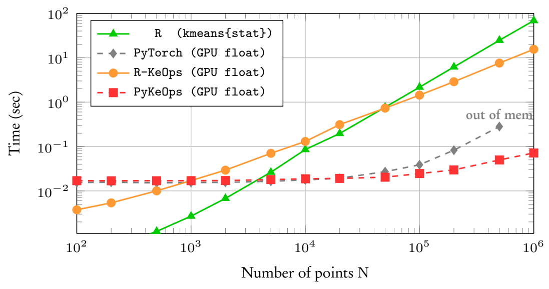
Figure 2.11: With `KeOps`, users can quickly implement classic machine learning (ML) algorithms using a code that is both *scalable* and *modular*. (a) With five lines of high-level `Python` code, **K-means clustering** can scale up to large datasets without any pre-processing. As displayed here, performing `10` iterations of the K-means loop on `N = 10,000` points in dimension `D = 2`, with `K = 50` clusters takes `10`·2ms on a RTX 2080 Ti chip. In a setting that is closer to standard ML applications, performing `10` iterations of the K-means loop with `N = 1,000,000` points in dimension `D = 100` with `K = 1,000` clusters takes `10·1.8s`. (b) To help users implement advanced schemes such as **spectral clustering**, **`KeOps LazyTensors` provide a clean interface to the solvers of the `scipy.sparse.linalg` package:** eigenvalue problems, linear systems, etc. Combined with the support of block-sparsity masks, this feature allows users to compute spectral coordinates on very large point clouds (`N = 1,000,000`) in minutes, without having to introduce arbitrary cutoffs on the number of neighbours per sample.



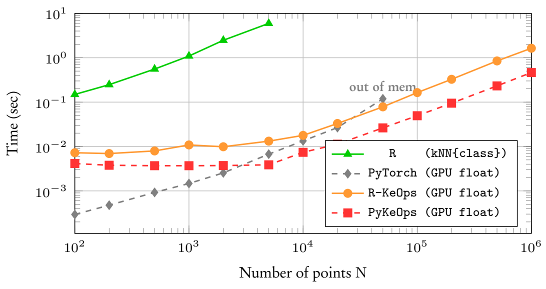
(a) Matrix-vector products with N-by-N Gaussian kernel matrices built from point clouds in dimension D = 3.



(b) Solving an N-by-NGaussian kernel linear system with ridge regularization (constant diagonal weights).



(c) 10 iterations of K-means (Lloyd's algorithm) with N points in dimension D = 10 and K = $\lfloor\sqrt{N}\rfloor$ clusters.



(d) Exact (K = 10)-nearest neighbor search: 10k queries in dimension D = 100 with a database of N samples.

Figure 2.12: Benchmarking `KeOps` on common machine learning problems.
Timings performed on a gaming Nvidia RTX 2080 Ti GPU, available for `< $1,500` as of 2020. `R-KeOps` scripts for kernel and geometric applications generally outperform their standard `R` counterparts (Venables and Ripley, 2002) by several orders of magnitude. On modern hardware, they **scale up to clouds of** N **`> 1,000,000` samples in seconds** and keep a linear memory footprint. Unfortunately though, `R` still stores data on the "CPU" Host memory: peak performances are obtained with our `PyKeOps` interface for `PyTorch` which allows users to define Device-only processing pipelines on the GPU.

	`PyTorch`	`PyTorch-TPU`	`TF-XLA`	`Halide`	`TVM`	`KeOps`	`CUDA`
N = 10k	34 ms	10 ms	23 ms	5 ms	6 ms	**1.8 ms**	1.4 ms
N = 100k	*	*	1,062 ms	360 ms	282 ms	**107 ms**	106 ms
N = 1M	*	*	*	41.3 s	26.5 s	**10.3 s**	10.1 s
Lines of code	5	5	5	15	17	**5**	55
Interface	arrays	arrays	arrays	C++	low-level	**arrays**	C++

Figure 2.13: Benchmarking `KeOps` against similar frameworks. Average runtimes for an N-by-N Gaussian kernel product in dimension D = 3 over 100 iterations – "*" stand for "out of memory" errors. For the sake of reproducibility, these timings are performed on a fresh Google Colab session, with a free Tesla K80 GPU (checked with `nvidia-smi`). As showcased Figure 2.12, timings with recent gaming hardware would be ~`10x` faster across the board.

As discussed page 47, a growing trend in the *systems for machine learning* literature has been to develop *just in time* compilation frameworks that turn high-level scripts into optimized executables. In this benchmark, run in December 2019, we compare the performances of our `KeOps` routines against other notable frameworks: a vanilla `PyTorch` code run on the GPU and on a *Tensor Processing Unit* provided by Google Colab ; a `TensorFlow` script with *Accelerated Linear Algebra* compilation (Leary and Wang, 2017) ; a `Halide` high-level C++ code (Ragan-Kelley et al., 2013) ; a `TVM Python` script (Chen et al., 2018) ; a reference `CUDA` implementation.

In the specific context of kernel-related operations, `KeOps` is extremely competitive. These timings are bound to evolve: we expect the impressive `TVM`, `Halide` and `XLA` libraries to catch-up with `KeOps` in years to come. Nevertheless, they allow us to illustrate the difference of focus between generalist frameworks and our library. Developed by mathematicians, `KeOps` only does one thing – but it does it well, with a transparent interface and full cross-platform compatibility.

2.3.3 Future works

Place of KeOps in the scientific ecosystem. The KeOps package has no claim to set the state-of-the-art in high performance computing: when implemented properly, hand-written CUDA schemes always outperform naive GPU loops, be it for (approximate) nearest neighbor search or B-spline interpolation.

However, as it combines a reasonable level of performance with the *flexibility* of a deep learning interface, KeOps can unlock research programs by increasing the productivity of developers. The main ambition of this work was to allow our colleagues in medical imaging to benefit from the "deep learning revolution" without having to focus exclusively on convolutional neural networks; we now hope that this localized success can be replicated in other fields.

Long-term goal: fast approximation schemes. In months to come, we plan to implement boilerplate features such as row- and column-wise indexing, block-wise definition of LazyTensors and a full support of tensor variables. Additional low-level profiling should also help us to converge towards optimal runtimes.

Long-term, our main challenge will be to reconcile KeOps with the rich literature in numerical mathematics that focuses on fast approximation schemes for kernel dot products, often referred to as *discrete convolutions* in computational geometry or *discrete integral operators* in physics. To perform efficiently the kernel matrix-vector product of Eq. (2.27), a most sensible idea is to compute a rank-R approximation of the linear operator $K_{x,y}$:

$$K_{x,y} \overset{\text{def.}}{=} \begin{bmatrix} & & \\ & k(x_i,y_j) & \\ & & \end{bmatrix} \simeq \begin{bmatrix} \\ A \\ \\ \end{bmatrix} \cdot \begin{bmatrix} B \end{bmatrix} \cdot \begin{bmatrix} \quad C \quad \end{bmatrix}, \tag{2.56}$$

where A, B and C are M-by-R, R-by-R and R-by-N matrices respectively. Such decompositions reduce the complexity of a matrix-vector product with K from a $O(\text{MN})$ to a $O((\text{M}+\text{R}+\text{N})\,\text{R})$.

In favorable cases, we should be able to build a decent approximation of $K_{x,y}$ with a small rank R, thus securing a dramatic speed-up. But how can we find relevant factors A, B and C? Historically, five major types of strategies have been proposed to tackle this problem:

1. **Singular value decompositions** and *adaptive cross-approximation* algorithms (Bebendorf, 2000; Zhao et al., 2005) iteratively pick the leading rows or eigenvectors of the kernel matrix $K_{x,y}$ to yield explicit numerical arrays A, B and C.

2. **Quadrature methods** sub-sample the point clouds x_i and y_j and rely on simple algebraic rules to correct for over- or under-sampling artifacts. For instance, if we pick R points $\tilde{x}_i$ among the x_i's, the *Nyström* rule asserts that:

$$K_{x,x} \simeq K_{x,\tilde{x}}\, K_{\tilde{x},\tilde{x}}^{-1}\, K_{\tilde{x},x}\,. \tag{2.57}$$

Random sampling strategies for the $\tilde{x}_i$'s have been studied extensively in the machine learning literature (Zhang et al., 2008; Yang et al., 2012).

3. **Spline-based decompositions**, discussed e.g. in (Cambier and Darve, 2019), are quadrature methods that rely on Lagrange polynomials to bypass the costly inversion of the

Nyström rule. The R control points $\tilde{x}_i$ and $\tilde{y}_j$ are generally placed at Chebyshev nodes (Mastroianni and Occorsio, 2001), and we end up making the approximation that:

$$K_{x,y} \simeq L_{x\leftarrow\tilde{x}}\, K_{\tilde{x},\tilde{y}}\, L^{\top}_{y\leftarrow\tilde{y}} , \tag{2.58}$$

where $L_{x\leftarrow\tilde{x}} = [L_1(x)|\cdots|L_\mathrm{R}(x)]$ denotes the M-by-R matrix of sampled values on the point cloud x of the polynomial interpolation basis $(L_1, \dots, L_\mathrm{R})$ associated to the $\tilde{x}_i$'s.

4. **Spectral strategies** can be applied whenever $k : (x,y) \mapsto k(x-y)$ is a translation-invariant kernel: they leverage the Fourier convolution theorem discussed Figure 3.2. Efficient implementations generally rely on non-uniform FFTs (Dutt and Rokhlin, 1993; Greengard and Lee, 2004) or random Fourier features (Rahimi and Recht, 2008).

5. **Multipole** decompositions rely on truncated Taylor developments of the kernel function. For instance, if:

$$k(x,y) \simeq \sum_{p=0}^{\mathrm{P}-1}\sum_{q=0}^{\mathrm{Q}-1} T_{p,q}\, x^p y^q , \tag{2.59}$$

we may approximate the kernel product $a_i = \sum_{j=1}^{\mathrm{N}} k(x_i, y_j)\, b_j$ as:

$$a_i \simeq \sum_{j=1}^{\mathrm{N}}\sum_{p=0}^{\mathrm{P}-1}\sum_{q=0}^{\mathrm{Q}-1} T_{p,q}\, x_i^p y_j^q\, b_j = \sum_{p=0}^{\mathrm{P}-1} x_i^p \sum_{q=0}^{\mathrm{Q}-1} T_{p,q} \sum_{j=1}^{\mathrm{N}} y_j^q\, b_j , \tag{2.60}$$

which is equivalent to working with the M-by-P and Q-by-N matrices of moments $A = [x_i^0|x_i^1|\cdots|x_i^{\mathrm{P}-1}]$ and $C = [y_j^0|y_j^1|\cdots|y_j^{\mathrm{Q}-1}]^\top$, with a P-by-Q matrix of Taylor coefficients $B = (T_{p,q})$ in-between. These strategies have been studied extensively for applications to physics, e.g. N-body simulation: we refer to (Greengard, 1988; Beatson and Greengard, 1997; Yang et al., 2003) for an introduction.

Multiscale strategies. When a coarse-to-fine decomposition of the input data is combined with one of these compression methods, we retrieve efficient block-wise approximations of the kernel matrix (Barnes and Hut, 1986; Beatson and Greengard, 1997; Hackbusch, 2015). Dramatic speed-ups can then be achieved even in full-rank cases, as encountered for instance when dealing with Coulomb or Helmholtz kernels:

$$k(x,y) = \frac{1}{\|x-y\|} \qquad \text{and} \qquad k(x,y) = \frac{e^{iw\|x-y\|}}{\|x-y\|} . \tag{2.61}$$

`KeOps` interface. In line with recent works in scientific computing (Aussal and Bakry, 2019), we should be able to reach an $O(\mathrm{N}\log\mathrm{N})$ time complexity with $O(\mathrm{N})$ memory usage in a wide range of favorable cases. Long-term, we expect to provide an interface to some of these algorithms through a simple "tolerance" parameter for `KeOps LazyTensors`: a transparent "`K.tol = 1e-3`" statement would be a dream come true!

Chapter 3

Geometry on a space of measures

in collaboration with François-Xavier Vialard (Paris-Est University), Thibault Séjourné and Gabriel Peyré (École Normale Supérieure).

Key points – Measure theory is a central topic in data sciences:

1. Studying **unlabeled distributions of mass or probability** is a major problem in applied sciences. Described in a language that encompasses both weighted point clouds and continuous probability laws, **"measures"** can be used to model shapes, random vectors or data samples in arbitrary feature spaces.

2. Restricting ourselves to operations that are **parameterization-invariant** and homogeneous with respect to the weights, we may try to define **principled distances between measures**. Used as loss functions in model-fitting pipelines, these routines provide **gradients** that can be used to match distributions with each other.

3. A general way of metrizing spaces of measures is to rely on **dual norms**, known as integral probability metrics or adversarial costs in statistics and machine learning. Historically, three families of geometries have attracted a considerable interest since the 1950's: **Hausdorff** distances, which rely on nearest-neighbor projections; **Kernel** norms, which rely on (off-grid) convolutions; **Optimal Transport** (OT) costs, which rely on the solutions of generalized sorting problems.

Contributions – Putting scalable Wasserstein distances on the shelf, with guarantees:

4. We provide a **unified overview** of Hausdorff, Kernel and OT fidelities from a **geometric perspective**. Transport-based loss functions (also known as Wasserstein or earth mover's distances) exhibit desirable behaviours, at a high computational cost: designing **tractable approximations of optimal transport** is a key problem for geometric data analysis.

5. Focusing on entropic regularization, we show that **de-biased Sinkhorn divergences** define convex, positive and definite loss functions that behave as **low-frequency** Wasserstein distances. Extensions to the **unbalanced** setting are handled in a clean, idiomatic fashion.

6. Benefiting from ten years of research on multiscale OT solvers, we propose a symmetrized, de-biased, multiscale Sinkhorn loop that can be understood as a **generalized Quicksort algorithm**. Implemented with the `PyKeOps` package, our code outperforms standard implementations of the Sinkhorn or Auction algorithms by **three orders of magnitude** and scales up to millions of samples in a matter of seconds. It is freely available on the `PyPi` repository (`pip install geomloss`) and on our reference website:
`www.kernel-operations.io/geomloss`

Chapter 3 – Geometry on a space of measures:

3.1 Working with measures – weighted point clouds

The language of measures, or "spatial distributions of mass", is a central part of the undergraduate curriculum in mathematics. Formalized at the turn of the XX$^{\text{th}}$ century by Borel, Lebesgue and others, it allows mathematicians to study *discrete sums* and *continuous integrals* within the same framework. Since the foundational work of Kolmogorov in the 30's, measure theory is also the pillar on which relies the modern axiomatization of probabilities.

The two cultures. Unfortunately, the study of this fundamental tool is mostly neglected outside of pure maths programs. Computer scientists tend to focus on *discrete* objects, which seem more suited to the digital age: combinatorics, formal grammars or graph theory. Meanwhile, *calculus* and *linear algebra* make up the bulk of engineering textbooks to the detriment of other topics: the existence of the Lebesgue integral is often accepted as an axiom of calculus.

Modern progresses in information technologies vindicate our track-based teaching system: down to earth classes ensure that engineers learn the fundamentals of their trade without getting lost in technicalities. However, in cross-disciplinary fields such as data sciences and medical imaging, the lack of a common vocabulary for structures that can *not* be simply understood as vectors – such as random variables or 3D meshes – leads to a great deal of misunderstandings between neighboring sub-communities.

More often than not, theorists and practitioners work on related problems and politely cite each other, but have a hard time understanding papers written at the other end of their field. Discussed in Section 3.3.4, an example of the confusion that can then prosper is the uncertainty that surrounds the "Wasserstein" keyword in the recent machine learning literature.

Reaching out computer scientists and engineers. This manuscript attempts to bridge the gap between elegant theorems and efficient implementations. To ensure that our theoretical *and* practical results on optimal transportation are understood by all readers, we choose to focus this chapter on the study of *discrete* measures in a *continuous* vector space. Simply put: **weighted point clouds**. This restriction allows us to bypass most of the difficulties that come with continuous optimization, while preserving the overall structure and geometry of the problem.

Going further, general (and fully rigorous) proofs of our main statements can be found in the Chapter A of the appendices. They will be of interest to our colleagues, but can be safely ignored by most readers. By setting up an explicit distinction between geometric *intuitions* and analytic *proofs*, we hope to make our work more palatable to the general public. In applied mathematics just as in C++ programming, the informal *documentation* that comes with a new result matters as much as its *demonstration*.

3.1.1 A parameterization-invariant encoding of data

Geometers and data scientists work with collections of samples:

$$x_1, x_2, \dots, x_\mathrm{N} \in \mathcal{X} \tag{3.1}$$

that belong to a domain-dependant *space of features* $\mathcal{X}$: typically, a subset of a vector space $\mathbb{R}^\mathrm{D}$. The x_i's may encode the positions of points in the ambient space $\mathbb{R}^3$, the values of a multi-variate signal on a population of N subjects, or any other type of vector data. In all cases, the input dataset can be thought of as a large N-by-D array $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.

Parameterization invariance. In most settings, the ordering of the x_i's can *not* be relied upon. The rows of "Excel spreadsheets" are often ordered at random, and data scientists take care to design algorithms that discard this misleading piece of information.

Permutation invariance is usually achieved by restricting computations to *symmetric* functions of the x_i's (Qi et al., 2017) or by keeping a strict focus on operations that are well-defined with respect to the un-ordered *set* of points:

$$\{x_1, x_2, \dots, x_\mathrm{N}\} \subset \mathcal{X} . \tag{3.2}$$

A typical example is the distance to the point cloud $\{(x_i)\}$:

$$\mathrm{d}(\,\cdot\,, \{x_1, \dots, x_\mathrm{N}\}) \;:\; x \in \mathbb{R}^\mathrm{D} \;\mapsto\; \min_{i=1}^{\mathrm{N}} \|x - x_i\| , \tag{3.3}$$

used by the celebrated *iterative closest point* algorithm (Besl and McKay, 1992).

Working with measures. From this perspective, a convenient way of taking *weights* and *multiplicity* into account is to introduce the concept of *measure*: additive distributions of mass on the feature space $\mathcal{X}$, that can be understood as **generalized "soft" subsets** of $\mathcal{X}$.

Formally, a (positive) measure μ on $\mathcal{X}$ is defined as a function:

$$\mu : S \subset \mathcal{X} \;\mapsto\; \mu(S) \in \mathbb{R} \cup \{+\infty\} , \tag{3.4}$$

that attributes a positive mass to subsets S of $\mathcal{X}$ and satisfies four axioms:

1. **The domain Σ of μ is a σ-algebra.** Unfortunately, some important measures can *not* be defined rigorously on the full collection $\mathcal{P}(\mathcal{X})$ of subsets of $\mathcal{X}$ and must be restricted to a collection Σ of "measurable" sets. We always assume that the domain Σ of a measure μ contains the full feature space $\mathcal{X}$, is closed under the complement and is closed under countable unions.
2. **Positivity.** For all measurable subset S of $\mathcal{X}$ in Σ, $\mu(S) \geqslant 0$. We may consider functions μ that do not satisfy this property, but will explicitely refer to them as "signed" measures.
3. **Null measure of the empty set.** If $\emptyset$ denotes the empty set $\{\,\} \subset \mathcal{X}$, then $\emptyset \in \Sigma$ and
$$\mu(\emptyset) \;=\; 0\,. \tag{3.5}$$
4. **Additivity.** If $(S_k)_{k\in\mathbb{N}}$ is a countable collection of *disjoint* subsets of $\mathcal{X}$ in Σ,
$$\mu\Big(\bigsqcup_{k=0}^{+\infty} S_k\Big) \;=\; \sum_{k=0}^{+\infty} \mu(S_k)\,. \tag{3.6}$$

The axioms of measure theory encode the intuition that *the mass is an extensive quantity*, distributed over the feature space $\mathcal{X}$ according to discrete or continuous laws "μ". Put together, these properties imply that a positive measure is always **non-decreasing**:

$$\forall\, S, T \in \Sigma,\;\; S \subset T \;\Longrightarrow\; \mu(S) \leqslant \mu(T)\,. \tag{3.7}$$

As a gentle introduction to measure theory, let us now present the fundamental intuitions and examples that can be associated to the formal definition of Eqs. (3.4-3.7).

Probability theory. Probabilists and statisticians focus on measures μ that sum up to 1 (i.e. are such that $\mu(\mathcal{X}) = 1$), generally understood as distributions of random variables X that take their values in the feature space $\mathcal{X}$: we say that X follows the law μ, or simply write that "$X \sim \mu$". For any *event* $S \in \Sigma$, $\mu(S)$ is identified with the probability that the random vector X takes its values in the measurable subset $S \subset \mathcal{X}$:

$$\mu(S) \;=\; \mathbb{P}_{X\sim\mu}(X \in S) \;\in\; [0,1]\,. \tag{3.8}$$

Dirac measures. The simplest example of measure is the **Dirac distribution** δ_x, fully concentrated at an arbitrary location $x \in \mathcal{X}$ and formally defined through:

$$\delta_x \;:\; S \subset \mathcal{X} \;\mapsto\; \begin{cases} 1 & \text{if } x \in S\,, \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

In words, δ_x only puts weight on the singleton $\{x\}$ and larger subsets of $\mathcal{X}$. From a probabilistic perspective, this "atomic" distribution is associated to a deterministic behaviour: $X \sim \delta_x$ if and only if $X = x$ almost surely; $\mathbb{P}_{X\sim\delta_x}(X = x) = 1$.

Lebesgue measures. A more refined example is the **Lebesgue or "volume" measure** on the ambient space $\mathbb{R}^{\mathrm{D}}$, intuitively defined through:

$$\lambda_{\mathbb{R}^{\mathrm{D}}} \;:\; S \subset \mathbb{R}^{\mathrm{D}} \;\mapsto\; \int_{x\in S} \mathrm{d}x\,. \tag{3.10}$$

Note that presenting a *rigorous* definition of the Lebesgue measure and of its σ-algebra $\mathcal{B}(\mathbb{R}^{\mathrm{D}})$ of measurable Borel sets takes a few days of work in undergraduate classes of mathematics. Defining a functional that generalizes the notions of *length*, *area* or *volume* of simple geometric sets while avoiding the pitfalls of the Banach-Tarski paradox is no mean feat.

From a probabilistic perspective, the Lebesgue measure encodes the notion of *uniform* distribution over a continuous domain. For instance, a univariate random variable X that follows the law of:

$$\lambda_{[0,1]} \;:\; S \subset \mathbb{R} \;\mapsto\; \int_{x \in S \cap [0,1]} \mathrm{d}x \tag{3.11}$$

is said to be *uniformly distributed* over the unit interval.

Gaussian measures. Our last fundamental example is the **Normal** or **Gaussian distribution**, informally defined in dimension D through:

$$\mathcal{N}_{\mathbb{R}^{\mathrm{D}}} \;:\; S \subset \mathbb{R}^{\mathrm{D}} \;\mapsto\; \frac{1}{(2\pi)^{\mathrm{D}/2}} \int_{x \in S} \exp(-\|x\|^2/2)\,\mathrm{d}x\,. \tag{3.12}$$

This distribution appears in the central limit theorem and provides a reference point for most theories in statistics and data sciences. As with all continuous distributions, its rigorous definition relies on the Lebesgue measure and associated σ-algebra of measurable Borel sets.

Algebraic manipulations. Unlike vectors and functions, measures can *not* be described in terms of coordinates in a canonical basis. In a Euclidean feature space $\mathcal{X} \subset \mathbb{R}^{\mathrm{D}}$, the notions of "continuous density" (Lebesgue) and "atomic distribution of mass" (Dirac) can only fit within the same framework through the language of set theory.

Nevertheless, if α and β are two measures that are respectively defined on collections Σ and Σ' of subsets of $\mathcal{X}$, and if a, b are two scalar weights in $\mathbb{R}$, we may define the **linear combination** $a\alpha + b\beta$ as the (signed) measure:

$$a\alpha + b\beta \;:\; S \in \Sigma \cap \Sigma' \;\mapsto\; a \cdot \alpha(S) \;+\; b \cdot \beta(S) \;\in\; \mathbb{R}\,. \tag{3.13}$$

This operation lets us combine heterogeneous objects: $\delta_0 + \lambda_{[0,1]} + \mathcal{N}_{\mathbb{R}}$ is a well-defined (positive) measure on the real line. In natural sciences, measures allow physicists to describe *pointwise*, *surface* or *volume* charge densities with clean and uniform notations.

Weighted point clouds. This chapter is mostly concerned with the study of **discrete measures**, written as (finite) linear combinations of atomic Dirac masses. Combining Eq. (3.9) with Eq. (3.13), we can write that:

$$\sum_{i=1}^{\mathrm{N}} \mu_i \delta_{x_i} \;:\; S \subset \mathcal{X} \;\mapsto\; \sum_{x_i \in S} \mu_i \;\in\; \mathbb{R} \tag{3.14}$$

is a well-defined measure for any collection $x_1, \dots, x_{\mathrm{N}}$ of samples in the feature space $\mathcal{X}$ and weights $\mu_1, \dots, \mu_{\mathrm{N}}$ in $\mathbb{R}$. In the special case where the μ_i's are equal to each other, the probability measure $\frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \delta_{x_i}$ can be understood as a **uniform distribution** over the discrete sample $(x_1, \dots, x_{\mathrm{N}})$ that takes multiplicity into account while being invariant to permutations.

3.1.2 Encoding discrete measures and continuous functions

Integration. The fundamental operation of measure theory is the **integration** of measurable functions f, denoted by:

$$\mu(f) \overset{\text{def.}}{=} \int f \mathrm{d}\mu \in \mathbb{R} . \tag{3.15}$$

This scalar number should be understood as the total value of the function f evaluated on the weighted set μ. It coincides with the usual integral of calculus when μ is the Lebesgue measure. In a simpler setting, if $\mu = \sum_{i=1}^{\mathrm{N}} \mu_i \delta_{x_i}$ is a discrete measure, we write that:

$$\int f \mathrm{d}\mu \overset{\text{def.}}{=} \sum_{i=1}^{\mathrm{N}} \mu_i \, f(x_i) . \tag{3.16}$$

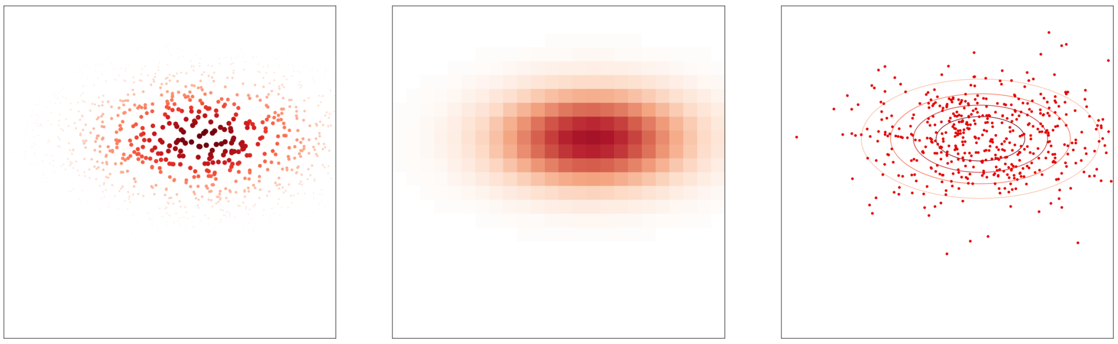
In functional analysis and probability theory (where measures always have unit mass), the integral operator is usually called the **duality bracket** or **expected value** and is alternatively denoted by:

$$\langle \mu , f \rangle \overset{\text{def.}}{=} \int f \mathrm{d}\mu \overset{\text{def.}}{=} \mathbb{E}_{X\sim\mu} [f(X)] . \tag{3.17}$$

In this manuscript, we favor the left-hand notation "$\langle \mu, f \rangle$" which higlights the bi-linearity of the integration with respect to both operands, measures *and* functions:

$$\langle \sum_{i=1}^{\mathrm{N}} \mu_i \delta_{x_i}, f \rangle = \sum_{i=1}^{\mathrm{N}} \mu_i \underbrace{\langle \delta_{x_i}, f \rangle}_{f(x_i)} \quad \text{and} \quad \langle \mu, f+g \rangle = \langle \mu, f \rangle + \langle \mu, g \rangle . \tag{3.18}$$

Duality. A key insight from functional analysis is that the measure-function duality bracket "$\langle \mu, f \rangle$" is the correct generalization of the "line · column" dot product to spaces of continuous functions. In a sense that is made rigorous by the many variants of the Riesz–Markov–Kakutani representation theorem, measures are the "lines" or "co-vectors" that correspond to continuous functions seen as "column" vectors in a space of infinite dimension.



(a) Weighted point cloud. (b) Density map. (c) Parametric generator.

Figure 3.1: Three different ways of encoding a multivariate Gaussian law.

Encoding measures and functions on a computer. This geometric point of view on integration and measure theory underlies most mathematical works in the field, including the proofs that we present in the Chapter A of this manuscript. For practitioners, these abstract considerations have one major consequence: **general measure-theoretic algorithms can be implemented in any setting that provides compatible encodings for *continuous functions* and *distributions of mass***, plus a well-defined *integral operator* or "measure · function dot product".

Skimming through the literature, most applied works focus on one out of three archetypal encodings – illustrated Figure 3.1:

1. **In geometry and data sciences,** the feature space $\mathcal{X}$ is a subset of $\mathbb{R}^{\mathrm{D}}$ and measures are encoded as weighted point clouds:

$$\mu \;=\; \sum_{i=1}^{\mathrm{N}} \mu_i \delta_{x_i} \;, \tag{3.19}$$

where $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and $(\mu_i) \in \mathbb{R}^{\mathrm{N}}$ are large N-by-D and N-by-1 arrays. Functions are encoded as programs that can be evaluated efficiently at arbitrary locations $x \in \mathbb{R}^{\mathrm{D}}$: a good example is the distance function of Eq. (3.3) that can be implemented efficiently using the `KeOps` library of Chapter 2. The duality bracket is performed as:

$$\langle \mu, f \rangle \;=\; \sum_{i=1}^{\mathrm{N}} \mu_i f_i \;, \tag{3.20}$$

where $(f_i) \in \mathbb{R}^{\mathrm{N}}$ is the vector of sampled values $(f(x_i))_{i\in[\![1,\mathrm{N}]\!]}$ evaluated in parallel.

2. **In imaging and signal processing,** the feature space is a discrete grid of pixels – say, $\mathcal{X} = [\![1,\mathrm{W}]\!] \times [\![1,\mathrm{H}]\!]$ – endowed with a reference counting measure:

$$\mathrm{Count}_{\mathcal{X}} \;\overset{\text{def.}}{=}\; \sum_{x\in\mathcal{X}} \delta_x \;:\; S \subset \mathcal{X} \;\mapsto\; \mathrm{Cardinal}(S) \;. \tag{3.21}$$

Since $\mathcal{X}$ is a countable (finite) set, the additivity axiom of Eq. (3.6) ensures that measures μ on $\mathcal{X}$ are entirely determined by their singleton function:

$$m \;:\; x \in \mathcal{X} \;\mapsto\; \mu(\{x\}) \in \mathbb{R} \;, \tag{3.22}$$

with $\mu = m\,\mathrm{Count}_{\mathcal{X}}$, i.e.:

$$\mu = \sum_{x\in\mathcal{X}} m(x)\,\delta_x \;:\; S \subset \mathcal{X} \;\mapsto\; \sum_{x\in S} m(x) \;. \tag{3.23}$$

Thanks to the finiteness of the feature space $\mathcal{X}$, measure theory on discrete grids can thus be implemented using standard vector operations. Measures μ and functions f are both encoded through "images" m and f in $\mathbb{R}^{\mathrm{W}\times\mathrm{H}}$, as the integral operator reads:

$$\langle\, \mu \,,\, f \,\rangle \;=\; \sum_{x\in\mathcal{X}} m(x)\, f(x) \;. \tag{3.24}$$

3. **In machine learning,** measures μ are understood through random variables $X \sim \mu$, encoded by oracles $(X_i)_{i\in\mathbb{N}}$ that can be evaluated at will. This is generally achieved by picking random lines in a large N-by-D data array, as we implement *online* variants of classical algorithms developed in the "geometry and data sciences" setting.

   More interestingly, authors who focus on generative modelling and adversarial networks tend to implement their random variables (X_i) as the *pushforwards* of reference oracles $X^{\text{ref}} \sim \mu^{\text{ref}}$ through parametric functions – the so-called *generative networks*. If $(X_i^{\text{ref}})_{i\in\mathbb{N}}$ is a process that samples a known probability law – say, a Gaussian $\mathcal{N}_{\mathbb{R}^d}$ on a low-dimensional *latent space* $\mathbb{R}^d$ – and if:

$$f_\theta ~:~ x \in \mathbb{R}^d \mapsto f_\theta(x) \in \mathbb{R}^{\text{D}} \tag{3.25}$$

   defines an embedding of $\mathbb{R}^d$ into the full feature space $\mathcal{X} = \mathbb{R}^{\text{D}}$, parameterized by a vector of weights θ, the parametric measure $\mu_\theta = \mu \circ f_\theta^{-1}$ is represented by the random values of the sampler:

$$X_i ~\overset{\text{def.}}{=}~ f_\theta(X_i^{\text{ref}}) ~\in~ \mathcal{X}~. \tag{3.26}$$

   Usually, continuous functions are also encoded with parametric functions:

$$g_\psi : \mathcal{X} = \mathbb{R}^{\text{D}} \to \mathbb{R} \tag{3.27}$$

   known as *adversarial neural networks* or *discriminators*, and stochastic Monte-Carlo approximations are used to estimate the values of integrals. If $\text{B} > 0$ is the *batch size* of the algorithm,

$$\langle\, \mu_\theta\,,\, g_\psi\,\rangle ~\simeq~ \frac{1}{\text{B}}\sum_{i=1}^{\text{B}} g_\psi(X_i) ~=~ \frac{1}{\text{B}}\sum_{i=1}^{\text{B}} g_\psi(f_\theta(X_i^{\text{ref}}))~. \tag{3.28}$$

This thesis. This manuscript is written with a focus on the first of these three encodings, which is well suited to computational anatomy and goes hand-in-hand with our `KeOps` library. As discussed in Chapter 4, the manipulation of weighted point clouds fits naturally with the geometric processing of curves and 3D meshes.

Promoting scientific interactions. Note, however, that most of the ideas presented in these pages are **independent of implementation details** and could be relevant in other applied fields. Our work relies extensively on insights and methods that were introduced by remarkable papers (Mérigot, 2011; Lévy, 2015; Schmitzer, 2019) in physically motivated settings (fluid mechanics, PET denoising, etc.), and we strongly believe in the cross-field nature of the topics that we are about to study.

The (abstract) language of measures is all about letting researchers focus on the big picture, freed from the burden of low-level programming. We hope that a gentle introduction to the field will help readers to get a clear understanding of (geometric) measure theory and promote cross-pollination between physics, computer vision, (medical) imaging and data sciences.

3.1.3 Notations, technical hypotheses

Before getting to the meat of this chapter, we now state clearly our main assumptions and provide a comprehensive reference for notations.

Bounded feature space. To ensure that all the quantities that we are about to define are *finite* without having to introduce cumbersome hypotheses on the moments of our distributions, we assume that our feature space $\mathcal{X}$, endowed with a distance function d, is *compact*.

In practice, we work with measures that have support in some **bounded** region of a finite-dimensional feature space: $\mathbb{R}^\mathrm{D}$ endowed with its standard Euclidean metric $\mathrm{d}(x,y) = \|x-y\|$. We sometimes refer to $\mathcal{X}$ as being "equal" to the full Euclidean space $\mathbb{R}^\mathrm{D}$, with the understanding that speaking about explicit balls or hyper-cubes of bounded diameter would be more appropriate.

Continuous functions, discrete measures. In this work, we draw a clear line between *functions* and *measures* on the feature space. The former, denoted by regular letters such as f, g, a or b always belong to the set $\mathcal{C}(\mathcal{X})$ of *continuous functions*.

On the other hand, measures are denoted by Greek letters α, β, π or μ and are always assumed to belong to the set $\mathcal{M}^+(\mathcal{X})$ of *finite Borel (and thus Radon) measures* on the *compact* metric space $(\mathcal{X}, \mathrm{d})$. To limit technical digressions, we focus this chapter on the simple case of *discrete* measures on a vector space $\mathcal{X} = \mathbb{R}^\mathrm{D}$. α and β can be written as weighted point clouds:

$$\alpha ~=~ \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i} \qquad \text{and} \qquad \beta ~=~ \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}~, \tag{3.29}$$

with non-negative weights $(\alpha_i) \in \mathbb{R}^\mathrm{N}_{\geqslant 0}$ and $(\beta_j) \in \mathbb{R}^\mathrm{M}_{\geqslant 0}$ associated to sampling locations $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and $(y_j) \in \mathbb{R}^{\mathrm{M}\times\mathrm{D}}$.

Outside of Section 3.3.2 (where we briefly discuss extensions of optimal transport theory to the *unbalanced* setting), we often assume that our measures α and β belong to the set of *unit mass*, probability measures $\mathcal{M}_1^+(\mathcal{X})$. Otherwise said, that:

$$\langle \alpha,\, 1\rangle ~=~ \sum_{i=1}^{\mathrm{N}} \alpha_i ~=~ 1 \qquad \text{and} \qquad \langle \beta,\, 1\rangle ~=~ \sum_{j=1}^{\mathrm{M}} \beta_j ~=~ 1. \tag{3.30}$$

Support of a measure. The support of a positive Radon measure $\alpha \in \mathcal{M}^+(\mathcal{X})$ is defined as the complement of the largest open set with null α measure. In other words, provided that the α_i's are *positive*,

$$\mathrm{Supp}\left(\sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}\right) ~=~ \{x_1, \dots, x_\mathrm{N}\} ~\subset~ \mathcal{X}. \tag{3.31}$$

Assuming that our feature space $\mathcal{X}$ is compact, we can generalize the notion of *distance* to compact sets: for any reference location $x \in \mathcal{X}$ and continuous function $\mathbf{C} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined on pairs of points (x, y), we write that:

$$\mathbf{C}(\, x,\, \mathrm{Supp}(\alpha)\,) ~\stackrel{\text{def.}}{=}~ \min_{y \in \mathrm{Supp}(\alpha)} \mathbf{C}(x,y)~.$$

Densities. When the support of a measure α is included in that of a reference measure β, we may sometimes re-write the relationship between the two distributions as a weighted equality:

$$\alpha \;=\; \tfrac{\mathrm{d}\alpha}{\mathrm{d}\beta}\,\beta\,, \qquad \text{i.e.} \qquad \int_{x\in\mathcal{X}} f(x)\,\mathrm{d}\alpha(x) \;=\; \int_{x\in\mathcal{X}} f(x)\,\tfrac{\mathrm{d}\alpha}{\mathrm{d}\beta}(x)\,\mathrm{d}\beta(x) \tag{3.32}$$

for all measurable function f. If it exists, such a function $\frac{\mathrm{d}\alpha}{\mathrm{d}\beta} : \mathcal{X} \rightarrow \mathbb{R}$ is uniquely defined β-almost everywhere and is usually called the *Radon-Nikodym derivative* or *density* of α with respect to β. In this favorable case, α is said to be *absolutely continuous* with respect to β – a fact denoted by $\alpha \ll \beta$ – and we may perform pointwise comparisons between the two measures. A fundamental example is the density of the normal Gaussian law of Eq. (3.12) with respect to the Lebesgue measure of Eq. (3.10) :

$$\frac{\mathrm{d}\mathcal{N}_{\mathbb{R}^\mathrm{D}}}{\mathrm{d}\lambda_{\mathbb{R}^\mathrm{D}}} \;:\; x\in\mathbb{R}^\mathrm{D} \;\mapsto\; \frac{1}{(2\pi)^{\mathrm{D}/2}}\,\exp(-\|x\|^2/2) \;\in\; \mathbb{R}\,. \tag{3.33}$$

Note that most relationships between measures can *not* be handled with density functions. Sometimes, α simply has no density with respect to β: this is for instance what happens when one tries to compare discrete measures with continuous distributions.

Convergence in law, weak-$\star$ topology. Even though measures do not always share a common support, notions of proximity and “convergence” can still be defined: intuitively, the sequence of Dirac atoms $\delta_{1/n}$ gets “closer to δ_0” as n tends to infinity. Going further, picking large amounts of identically distributed random samples should allow us to “approximate” continuous probability laws with discrete objects.

To formalize this idea, mathematicians rely on the *convergence in law* – also known as weak-$\star$ topology – and write that:

$$\alpha_n \rightharpoonup \alpha_\infty \quad\Longleftrightarrow\quad \forall\, f\in\mathcal{C}(\mathcal{X}),\;\; \langle\alpha_n, f\rangle \rightarrow \langle\alpha_\infty, f\rangle \;\text{ i.e. } \int_{\mathcal{X}} f\,\mathrm{d}\alpha_n \rightarrow \int_{\mathcal{X}} f\,\mathrm{d}\alpha_\infty\,. \tag{3.34}$$

In words: a sequence of measures $(\alpha_n)_{n\in\mathbb{N}}$ converges *weakly* towards a limit distribution α_∞ if and only if we can observe a convergence of the associated integrals with respect to *continuous* test functions. Since $\langle\delta_x,\, f\rangle = f(x)$, the definition above is enough to ensure that:

$$\delta_{1/n} \;\xrightharpoonup{n\rightarrow+\infty}\; \delta_0\,. \tag{3.35}$$

More interestingly, using the well-known theory of Riemann integrals, we can show that:

$$\frac{1}{n}\sum_{i=1}^{n}\delta_{i/n} \;\rightharpoonup\; \lambda_{[0,1]}\,, \quad \text{i.e. that} \quad \forall\, f\in\mathcal{C}([0,1]),\;\; \frac{1}{n}\sum_{i=1}^{n} f(i/n) \rightarrow \int_0^1 f(x)\,\mathrm{d}x\,. \tag{3.36}$$

The notion of *weak* convergence of discrete measures towards (continuous) limit distributions is at the heart of statistics (central limit theorem, etc.) and computational geometry (mesh refinement, etc.). A functional $\mathrm{F} : \mathcal{M}^+(\mathcal{X}) \rightarrow \mathbb{R}$ is said to be *continuous with respect to the convergence in law* if:

$$\alpha_n \rightharpoonup \alpha \quad \text{implies that} \quad \mathrm{F}(\alpha_n) \rightarrow \mathrm{F}(\alpha) \tag{3.37}$$

for any sequence of measures in $\mathcal{M}^+(\mathcal{X})$. This notion of regularity encodes the idea of **stability with respect to deformations of the measures’ supports** and should be satisfied by all *geometric* quantities that are computed from measures.

Product space. To compare measures with each other, we need to consider pairs of points (x, y) that belong to the Cartesian product $\mathcal{X}^2 = \mathcal{X} \times \mathcal{X}$: let us now recall some standard notations that link the feature space $\mathcal{X}$ to the set of pairs $\mathcal{X}^2$.

Tensor sum. If f and g are two functions in $\mathcal{C}(\mathcal{X})$, their **tensor sum** $f \oplus g$ is defined as:

$$f \oplus g \ : \ (x, y) \in \mathcal{X} \times \mathcal{X} \ \mapsto \ f(x) + g(y) \ .$$

If $\mathcal{X} = \{x_1, \dots, x_\mathrm{N}\}$ is a discrete space, functions f and g on $\mathcal{X}$ are usually encoded through vectors $(\boldsymbol{f}_i)$ and $(\boldsymbol{g}_j)$ in $\mathbb{R}^\mathrm{N}$. In this setting, $f \oplus g$ is understood as a large N-by-N array,

$$(\boldsymbol{f} \oplus \boldsymbol{g})_{i,j} \ = \ \boldsymbol{f}_i \ + \ \boldsymbol{g}_j \qquad \text{i.e.} \qquad \boldsymbol{f} \oplus \boldsymbol{g} \ = \ \boldsymbol{f} + \boldsymbol{g}^\top \ , \tag{3.38}$$

with a broadcasted "column + line = square" addition.

Tensor product. Similarly, if α and β are measures on $\mathcal{X}$, their **tensor product** $\alpha \otimes \beta$ is a measure on the product space $\mathcal{X} \times \mathcal{X}$ defined through:

$$\forall\, f \in \mathcal{C}(\mathcal{X} \times \mathcal{X}), \ \int_{\mathcal{X} \times \mathcal{X}} f(x, y) \,\mathrm{d}(\alpha \otimes \beta)(x, y) = \int_{\mathcal{X}} \int_{\mathcal{X}} f(x, y) \,\mathrm{d}\alpha(x) \,\mathrm{d}\beta(y) \ . \tag{3.39}$$

In the discrete setting discussed above, measures are encoded as vectors $(\boldsymbol{\alpha}_i)$ and $(\boldsymbol{\beta}_j)$ in $\mathbb{R}^\mathrm{N}$. Their tensor product is identified with an N-by-N array:

$$(\boldsymbol{\alpha} \otimes \boldsymbol{\beta})_{i,j} \ = \ \boldsymbol{\alpha}_i \boldsymbol{\beta}_j \qquad \text{i.e.} \qquad \boldsymbol{\alpha} \otimes \boldsymbol{\beta} \ = \ \boldsymbol{\alpha} \boldsymbol{\beta}^\top \ . \tag{3.40}$$

Marginals. Finally, if π is a measure on the product space $\mathcal{X} \times \mathcal{X}$, we refer to its two **marginals** as π_1 and π_2. If π is encoded as a large N-by-N array $\boldsymbol{\pi}$, π_1 and π_2 respectively correspond to the vectors of row- and column-wise sums:

$$\boldsymbol{\pi}_1 \ = \ \boldsymbol{\pi} \mathbf{1} \qquad \text{and} \qquad \boldsymbol{\pi}_2 \ = \ \boldsymbol{\pi}^\top \mathbf{1} \ . \tag{3.41}$$

If π has density with respect to a tensor product $\alpha \otimes \beta$, i.e. $\pi = p(\,\cdot\,,\,\cdot\,) \cdot (\alpha \otimes \beta)$, we have that:

$$\mathrm{d}\pi_1(x) \ = \ \Big(\int_{y \in \mathcal{X}} p(x, y) \,\mathrm{d}\beta(y) \Big) \cdot \mathrm{d}\alpha(x) \tag{3.42}$$

$$\text{and} \quad \mathrm{d}\pi_2(y) \ = \ \Big(\int_{x \in \mathcal{X}} p(x, y) \,\mathrm{d}\alpha(x) \Big) \cdot \mathrm{d}\beta(y) \ . \tag{3.43}$$

Smoothing, convolution. If α is a (positive or signed) measure on $\mathcal{X}$ and if:

$$k \ : \ (x, y) \in \mathcal{X} \times \mathcal{X} \ \mapsto \ k(x, y) = k(y, x) \in \mathbb{R} \tag{3.44}$$

is a symmetric, real-valued continuous function on the product space $\mathcal{X}^2$, the **smoothing** $k \star \alpha \in \mathcal{C}(\mathcal{X})$ is defined through:

$$k \star \alpha \ : \ x \in \mathcal{X} \ \mapsto \ \int_{y \in \mathcal{X}} k(x, y) \,\mathrm{d}\alpha(y) \ . \tag{3.45}$$

If $\mathcal{X} = \mathbb{R}^\mathrm{D}$ is a vector space and if $k(x, y) = k(x - y)$ is a translation-invariant kernel parameterized by a "filter" $k : \mathbb{R}^\mathrm{D} \to \mathbb{R}$, the definition above coincides with the well-known **convolution** operator:

$$\forall\, x \in \mathbb{R}^\mathrm{D}, \quad (k \star \alpha)(x) \ \overset{\text{def.}}{=} \ \int_{y \in \mathbb{R}^\mathrm{D}} k(x - y) \,\mathrm{d}\alpha(y) \ = \ \int_{y \in \mathbb{R}^\mathrm{D}} k(y) \,\mathrm{d}\alpha(x - y) \ . \tag{3.46}$$

Implementing a convolution. As illustrated Figure 3.2, a convolution $k \star \alpha$ can either be understood as a weighted sum of translated copies of the kernel k, or through a simple scaling of coefficients in the (spectral) Fourier domain.

Mirroring the discussion of Section 3.1.2, this fundamental "(function, measure) $\rightarrow$ function" operator can be encoded in (at least) three different ways – illustrated Figure 3.3:

1. **If α is encoded as a weighted point cloud,** the convolution can be understood as a matrix-vector product with a *kernel matrix*. Assuming that $\alpha = \sum_{j=1}^{\mathrm{N}} \alpha_j \delta_{x_j}$, the values of the continuous function $f = k \star \alpha$ on a point cloud $(y_i) \in \mathcal{X}^{\mathrm{M}}$ can be computed through:
$$f(y_i) \;=\; (k \star \alpha)(y_i) \;=\; \sum_{j=1}^{\mathrm{N}} k(y_i - x_j)\, \alpha_j \quad \text{i.e.} \quad (f_i) \;=\; (K_{y_i,x_j})(\alpha_j)\,, \tag{3.47}$$
where (K_{y_i,x_j}) is the M-by-N matrix of kernel values $k(y_i, x_j)$. Going further, as discussed in Chapter 2, this operation can be implemented efficiently using approximation schemes or the GPU routines of the `KeOps` library.

2. **If α is encoded as a density map a on a grid of pixels,** the convolution "image" $k \star \alpha$ is made up of local averages of the values of the density map $(a[i,j])$ weighted by values of k stored in the so-called *convolution filter* $(k[i,j])$. Efficient implementations generally rely on explicit evaluations of the sum:
$$(k \star \alpha)[\,i_0,\, j_0\,] \;=\; \sum_{i,j} k[\,i,\, j\,]\, a[\,i_0 - i,\, j_0 - j\,] \tag{3.48}$$
when the filter k has a small support, and leverage fast Fourier transforms otherwise.

3. **If α is encoded through a random sampler $X \sim \alpha$,** kernel matrices or `KeOps` routines can provide Monte-Carlo estimations of the values of $k \star \alpha$ at arbitrary sampling locations: with a *batch size* $\mathrm{B} > 0$,
$$(k \star \alpha)(x) \;\simeq\; \frac{1}{\mathrm{B}} \sum_{i=1}^{\mathrm{B}} k(x - X_i)\,. \tag{3.49}$$
Alternatively, in settings where functions are best encoded as parametric programs – e.g. in the GAN literature – authors tend to refrain from including (expensive) smoothing operators in their *neural architectures*. Regularization of the probability measure α is directly performed through the addition of an independent noise to the samples' values: if $k = k_\sigma$ is a Gaussian kernel of deviation $\sigma > 0$
$$k_\sigma \;:\; x \in \mathbb{R}^{\mathrm{D}} \;\mapsto\; \frac{1}{(2\pi\sigma^2)^{\mathrm{D}/2}}\, \exp(-\|x\|^2/2\sigma^2)\,, \tag{3.50}$$
adding an independent Gaussian noise $B_i \sim \mathcal{N}_{\mathbb{R}^{\mathrm{D}}}$ to the parametric sampler $(X_i)_{i \in \mathbb{N}}$ with:
$$Y_i \;=\; X_i \;+\; \sigma B_i \tag{3.51}$$
allows us to sample the probability law whose density with respect to the Lebesgue measure $\lambda_{\mathbb{R}^{\mathrm{D}}}$ is equal to $k_\sigma \star \alpha$.

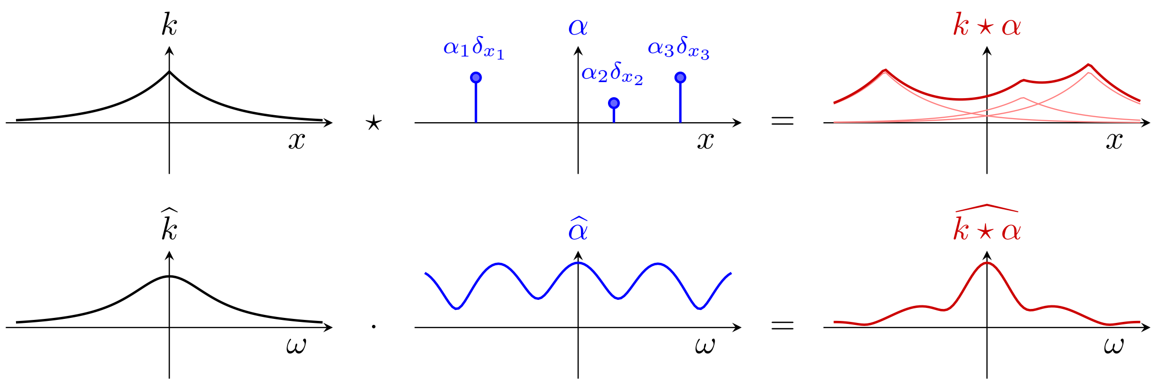
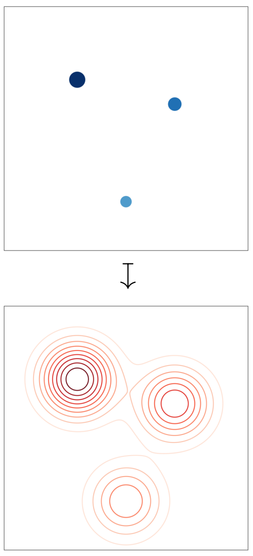
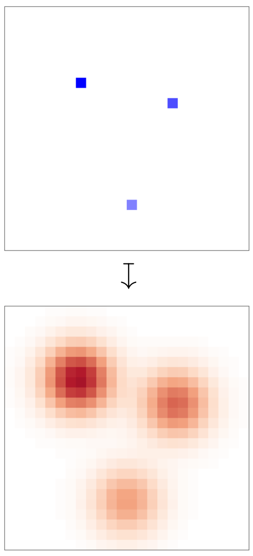


Figure 3.2: The convolution operator. (first line) Assuming that α is a finite, discrete measure, the convolution $k \star \alpha$ is a superposition of weighted and translated copies of the kernel function k. (second line) Spectral analysis allows us to write functions k and measures α as superpositions of pure harmonics $e_\omega : x \in \mathbb{R}^{\mathrm{D}} \mapsto \exp(i\omega \cdot x) \in \mathbb{C}$ indexed by wave vectors $\omega \in \mathbb{R}^{\mathrm{D}}$: the Fourier transforms $\widehat{k}$ and $\widehat{\alpha}$ of our objects can be understood as (infinite) collections of coordinates $(\widehat{k}(\omega))_{\omega\in\mathbb{R}^{\mathrm{D}}}$ and $(\widehat{\alpha}(\omega))_{\omega\in\mathbb{R}^{\mathrm{D}}}$ in the orthonormal (Hilbert) basis of trigonometric wave functions. Crucially, in this convenient system of coordinates, translation-invariant linear operators such as the convolution are simple *diagonal scalings*: the Fourier transform of $k \star \alpha$ is given by the pointwise product $\widehat{k \star \alpha}(\omega) = \widehat{k}(\omega)\,\widehat{\alpha}(\omega)$. When k is a continuous kernel, the convolution product $\alpha \mapsto k \star \alpha$ can thus be understood as a **lowpass** operator that turns discrete (peaked) measures into functions that are "as smooth as k" by attenuating the high-frequency components of $\widehat{\alpha}$.


(a) Geometry.


(b) Image processing.

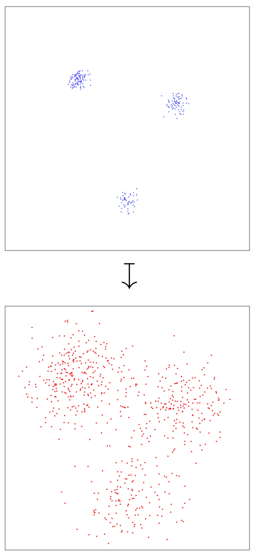

(c) Machine learning.

Figure 3.3: Three different ways of **implementing a convolution**. (a) When α is given as a weighted point cloud, kernel dot products allow us to evaluate $k \star \alpha$ at any location x in the feature space $\mathcal{X}$. In this manuscript, such implicit functions are represented using level sets known as *metaballs* in computer graphics (Blinn, 1982). (b) When α is encoded as a density on a grid of pixels, the values of $k\star\alpha$ sampled at the same pixel locations can be computed using efficient *convolution layers*. (c) When a probability distribution α is encoded through a random sampler (X_i), the addition of an independent perturbation whose law has density k with respect to the Lebesgue measure is a simple way of introducing the convolution $k \star \alpha$ in a stochastic algorithm.

3.2 Defining distances between measures

As discussed in the previous section, **measures encode the notion of weighted set** and usually refer to:

1. **Weighted point clouds** or un-labeled datasets in shape analysis and data sciences.
2. **Density maps** or segmentation masks in computer vision and medical imaging.
3. **Random vectors** in statistics and machine learning.

These objects have less structure than raw numerical arrays, but can still be manipulated efficiently. Using encodings that best fit the constraints of real-life applications, programmers may *combine*, *integrate* and *convolve* measures to let them interact with continuous test functions.

Metrizing a space of measures. Crucially, theorists and practitioners also need to measure *errors* between parametric models and empirical datasets. From the 50's onwards, a major problem in applied mathematics has thus been to define loss functions that could quantify the discrepancy between any two measures α and β at an affordable computational cost.

Desirable properties. In order to legitimize geometric intuitions, researchers tend to focus on functionals:

$$\text{Loss} \;:\; (\alpha,\beta)\in\mathcal{M}^+(\mathcal{X})\times\mathcal{M}^+(\mathcal{X}) \;\mapsto\; \text{Loss}(\alpha,\beta)\in\mathbb{R} \tag{3.52}$$

that are related to *distances* on the space of measures: ideally, Loss or $\sqrt{\text{Loss}}$ should satisfy the triangle inequality. Failing that, suitable loss functions should at the very least be **positive** and **definite**:

$$\forall\,\alpha,\beta\in\mathcal{M}^+(\mathcal{X}),\;\; \text{Loss}(\alpha,\beta)\;\geqslant\;0 \qquad \text{and} \qquad \text{Loss}(\alpha,\beta)\;=\;0\;\Leftrightarrow\;\alpha=\beta\,. \tag{3.53}$$

Stability. As discussed around Eqs. (3.34-3.37), theoretical properties that are related to the **convergence in law** allow mathematicians to guarantee that their formulas *stay consistent* when practitioners improve their discretizations. A loss function is **weakly continuous** if:

$$\text{Loss}(\alpha,\beta) \;=\; \text{Loss}\Big(\sum_{i=1}^{\text{N}}\alpha_i\delta_{x_i}\,,\;\sum_{j=1}^{\text{M}}\beta_j\delta_{y_j}\Big) \tag{3.54}$$

is **stable** with respect to the measures' weights and sampling locations. This definition covers the splitting of Dirac masses in geometry, the up- and down-sampling of density maps in imaging or changes of the batch sizes N and M in statistics.

Going further, we say that a loss **metrizes the convergence in law** if:

$$\alpha_n \xrightharpoonup{n\rightarrow+\infty} \alpha_\infty \iff \text{Loss}(\alpha_n,\alpha) \xrightarrow{n\rightarrow+\infty} 0\,, \tag{3.55}$$

i.e. if it can be relied upon to assess the convergence of discrete samplers to their underlying continuous distributions. Note that the properties above are minimal requirements, which do *not* guarantee that a loss formula behaves in a way that is compatible with the geometry of the ambient feature space $\mathcal{X}$. As we are about to see, the choice of a suitable loss function – among the large collection of formulas that are positive, definite and numerically stable – can have a massive impact on the behavior of applied pipelines.

Gradient of a function defined on a space of measures. As discussed in Section 3.1.2, *signed measures* are in duality with *continuous functions*: the associated vector spaces $\mathcal{M}(\mathcal{X})$ and $\mathcal{C}(\mathcal{X})$ interact with each other through the integration or *duality bracket* "$\langle \mu, f \rangle$". As we are about to see, this operation plays the same role as the dot products of Definition 2.3 in the definition of (generalized) gradients for measure-theoretic functionals:

Definition 3.1 (Differentiability on the space of Radon measures)**.** A functional $\mathrm{F} : \mathcal{M}(\mathcal{X}) \rightarrow \mathbb{R}$ is said to be differentiable at $\alpha \in \mathcal{M}(\mathcal{X})$ if there exists a continuous function, the *gradient* of F at α denoted by $\nabla \mathrm{F}(\alpha) \in \mathcal{C}(\mathcal{X})$ such that:

$$\forall \xi \in \mathcal{M}(\mathcal{X}),\, \forall t \in \mathbb{R},\; \mathrm{F}(\alpha + t\xi) \;\overset{\text{def.}}{=}\; \mathrm{F}(\alpha) \,+ t \,\langle\, \xi \,,\, \nabla \mathrm{F}(\alpha)\,\rangle \,+ o(t) \tag{3.56}$$

$$= \;\mathrm{F}(\alpha) + t \int_{\mathcal{X}} \nabla \mathrm{F}(\alpha)\, \mathrm{d}\xi + o(t)\;. \tag{3.57}$$

If F is merely defined on the space of *probability* measures $\mathcal{M}_1^+(\mathcal{X})$, variations ξ are such that $\int_{\mathcal{X}} \mathrm{d}\xi = 0$ and $\nabla \mathrm{F}(\alpha)$ is only defined up to an additive constant – we refer to (Santambrogio, 2015) for a detailed explanation.

Encoding the gradient. In practice, as discussed Section 3.1.2, measures are encoded through discrete objects such as vector of weights $(\alpha_i) \in \mathbb{R}^{\mathrm{N}}_{\geqslant 0}$ and tables of sampling locations $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$. The theoretical continuous gradient $f = \nabla \mathrm{F}(\alpha)$ is linked to the actual vectors "$\nabla_{\alpha_i} \mathrm{F}(\alpha)$" and "$\nabla_{x_i} \mathrm{F}(\alpha)$" needed by practitioners as follows:

1. **In geometry and data sciences.** If α is encoded as a weighted point cloud $\sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$,

$$\nabla_{\alpha_i} \mathrm{F}\big(\textstyle\sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}\big) \;=\; f(x_i) \quad \text{and} \quad \nabla_{x_i} \mathrm{F}\big(\textstyle\sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}\big) \;=\; \alpha_i \,\nabla f(x_i)\;. \tag{3.58}$$

   Note that the second identity involves the spatial gradient $\nabla f(x_i) \in \mathbb{R}^{\mathrm{D}}$ of the function $f = \nabla \mathrm{F}(\alpha) : \mathcal{X} \rightarrow \mathbb{R}$ on the feature space. It can be shown formally using a permutation of limits, which is legitimate in all practical use cases.

2. **In imaging and signal processing.** When α is given through its density $a(x)$ on a grid of pixels $\mathcal{X} = [\![1, \mathrm{W}]\!] \times [\![1, \mathrm{H}]\!]$, $\nabla_a \mathrm{F}$ is also encoded as an image, with:

$$\nabla_{a(x)} \mathrm{F}(\, a \cdot \mathrm{Count}_{\mathcal{X}}\,) \;=\; f(x)\;. \tag{3.59}$$

3. **In machine learning.** Finally, if α is encoded as the parametric push-forward α_θ of a reference measure, f appears as an intermediate step in the chain rule for $\theta \mapsto \mathrm{F}(\alpha_\theta)$. Using the notations of Section 2.1.2 :

$$\nabla_\theta \mathrm{F}(\alpha_\theta) \;=\; \mathrm{d}_\theta^\top \alpha_\theta \,\cdot\, \mathrm{d}_\alpha^\top \mathrm{F} \,\cdot\, 1 \;=\; \mathrm{d}_\theta^\top \alpha_\theta \,\cdot\, f\;. \tag{3.60}$$

3.2.1 Pointwise divergences: total variation and relative entropy

Total variation. Formally, as detailed in Eq. (3.4), measures are defined as *functions* μ that attribute a positive mass $\mu(S)$ to subsets S of the feature space $\mathcal{X}$. From this perspective, the simplest notion of distance that can be defined between two positive measures α and β is therefore the **Total Variation**:

$$\text{TV} \;:\; (\alpha,\beta) \in \mathcal{M}^+(\mathcal{X}) \times \mathcal{M}^+(\mathcal{X}) \;\mapsto\; \sup_{S \subset \mathcal{X}} \;|\,\alpha(S) - \beta(S)\,| \;\in\; \mathbb{R}_{\geqslant 0}\;, \tag{3.61}$$

which is the maximal discrepancy between the values of α and β on measurable subsets – or *events* – of $\mathcal{X}$. Focusing on the measure-function duality, we can write the Total Variation as:

$$\text{TV}(\alpha,\beta) \;=\; \sup_{\|f\|_\infty \leqslant 1} \;\langle \alpha - \beta,\, f \rangle\;, \tag{3.62}$$

where *competitors* or *adversarial* test functions $f : \mathcal{X} \to \mathbb{R}$ are measurable and bounded:

$$\|f\|_\infty \;\overset{\text{def.}}{=}\; \sup_{x\in\mathcal{X}} |f(x)| \;\leqslant\; 1\;. \tag{3.63}$$

Relative entropy. Assuming that α has density $\frac{\mathrm{d}\alpha}{\mathrm{d}\beta}$ with respect to β, we can go further and define the *relative entropy* or **Kullback-Leibler divergence** through:

$$\text{KL}(\alpha,\beta) \;\overset{\text{def.}}{=}\; \langle\, \alpha\,,\, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\beta}\,\rangle \;-\; \langle \alpha,\, 1\rangle \;+\; \langle \beta,\, 1\rangle\;. \tag{3.64}$$

If $\frac{\mathrm{d}\alpha}{\mathrm{d}\beta}$ cannot be defined, the value of $\text{KL}(\alpha,\beta)$ is set to $+\infty$. As discussed in Section A.3.1, this (assymetric) loss function is positive, definite and can be written in dual form as:

$$\text{KL}(\alpha,\beta) \;=\; \sup_{f\in\mathcal{C}(\mathcal{X})} \;\langle \alpha,\, f\rangle \;-\; \langle \beta,\, e^f - 1\rangle\;. \tag{3.65}$$

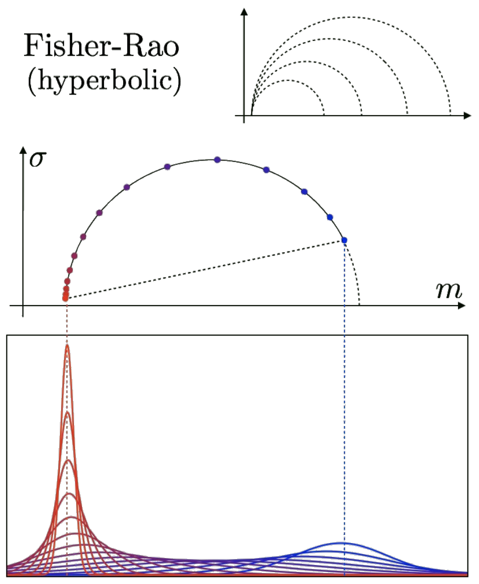
Invariance to the feature space. Thanks to their *point-wise* definitions which do not rely on pair-wise quantities such as the distance $\|x-y\|$, the TV and KL losses are invariant to the parameterization of the feature space $\mathcal{X}$: applying a change of coordinates to the features x_i and y_j of our measures has no impact on the values of $\text{TV}(\alpha,\beta)$ and $\text{KL}(\alpha,\beta)$. As far as the latter is concerned, this even holds in the continuous case (Bauer et al., 2016).

The robustness of these metrics to the parameterization of the feature space prevents them from being continuous with respect to the convergence in law, but makes them ideally suited to the processing of generic *histograms*. When measures have support on a pre-defined set of un-related labels such as:

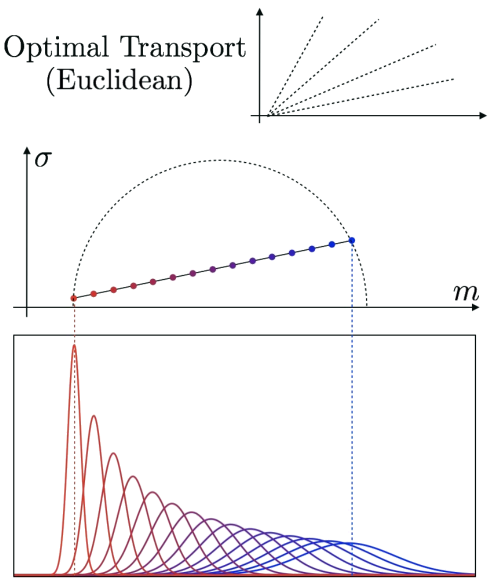
$$\mathcal{X} \;=\; \{\text{“dog”},\ \text{“cat”},\ \text{“bird”}\}\;, \tag{3.66}$$

which is typical for classification taks in machine learning and computer vision, using the TV, KL or general Csiszár f-divergences (Csiszár et al., 2004) losses is a most sensible choice.

Endowed with a remarkable intrinsic structure, the relative entropy KL lies at the heart of *information theory*, with numerous applications to computer science (Shannon, 1948) and statistics (Kullback, 1997). Derived from the theory of *entropic coding*, the “`.zip`” algorithm (Ziv and Lempel, 1978) is an ubiquitous standard for the compression of binary files, to be used in situations where the structure of the problem can *not* be leveraged. This is in constrast with e.g. the processing of natural images, where the Fourier- and wavelet-based JPEG and JPEG-2000 algorithms are industry standards (Wallace, 1992; Skodras et al., 2001).



(a) Fisher-Rao metric.



(b) Wasserstein-2 metric.

Figure 3.4: Geodesics on the statistical manifold of univariate Gaussian laws $\mathcal{N}(m, \sigma)$.
(a) When the metric structure on the ambient space of probability measures is given by the KL divergence, the 2-dimensional *statistical manifold* of Gaussian laws is isometric to the **Poincaré upper half-plane**. This standard model of hyperbolic geometry corresponds to a local scaling in $1/\sigma$ of the Euclidean distance around each point (m, σ), which promotes trajectories that pass through high-variance states. (b) When the space of probability measures is endowed with the Wasserstein-2 metric discussed in Section 3.2.4, Gaussian laws can be identified with points (m, σ) in the **Euclidean upper half-plane**. Images taken from the textbook (Peyré and Cuturi, 2017), where detailed computations can be found.

Statistical manifolds. Going further, the field of *information geometry* (Amari and Nagaoka, 2007) introduces *geometric ideas* in statistics by restricting the KL divergence on the "ambient space" $\mathcal{M}^+(\mathcal{X})$ to parametric families of probability distributions, seen as surfaces or submanifolds. A motivating example is to consider the family of univariate Gaussian laws $\mathcal{N}(m, \sigma)$, parameterized by a scalar *mean value* $m \in \mathbb{R}$ and a positive deviation $\sigma > 0$:

$$\forall x \in \mathbb{R}, \quad \frac{\mathrm{d}\mathcal{N}(m,\sigma)}{\mathrm{d}\lambda_{\mathbb{R}}} \;=\; \frac{1}{\sigma\sqrt{2\pi}} \exp\big(-\|x-m\|^2 \,/\, 2\sigma^2 \big)\,. \tag{3.67}$$

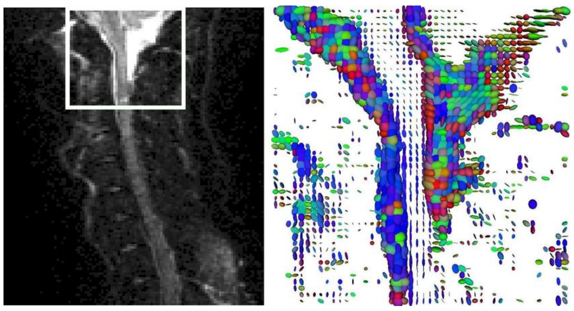
Linearizing the KL formula around a reference measure $\mathcal{N}(m, \sigma)$, we find that for sufficiently small deviations $(\Delta m, \Delta\sigma)$ of the values of the parameters m and σ:

$$\mathrm{KL}\Big(\mathcal{N}(m+\Delta m, \sigma+\Delta\sigma), \mathcal{N}(m,\sigma)\Big) \;=\; \frac{\frac{1}{2}|\Delta m|^2 + |\Delta\sigma|^2}{\sigma^2} \;+\; o((\Delta m, \Delta\sigma)^2)\,. \tag{3.68}$$

Remarkably, up to a benign rescaling of m into $m/\sqrt{2}$, this quantity coincides with the well-know **hyperbolic Poincaré metric** on the upper half-plane $\mathbb{R} \times \mathbb{R}_{>0}$, described in (Cannon et al., 1997; Charpentier et al., 2010). The "Gaussian mapping" :

$$\mathcal{N} \;:\; (m,\sigma) \in \mathbb{R} \times \mathbb{R}_{>0} \;\mapsto\; \mathcal{N}(m,\sigma) \in \mathcal{M}^+(\mathcal{X}) \tag{3.69}$$

can thus be understood as an **isometry** between the Poincaré model and the family of Gaussian distributions, endowed with the intrinsic **Fisher-Rao** metric induced locally by the relative entropy KL.


(a) Diffusion Tensor Imaging.

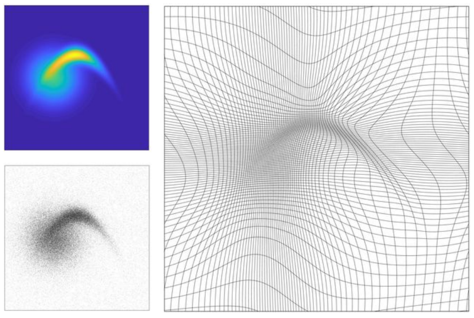

(b) Sampling arbitrary distributions.

Figure 3.5: Unexpected applications of information geometry to medical imaging and sampling. Figure (a) is taken from (Pennec, 2008), and illustrates the need for efficient denoising algorithms on tensor-valued images – here, a DTI view of the rachis. The restriction of the Fisher-Rao metric to multi-variate Gaussian laws is affine invariant, and underlies the reference algorithms in the field. Figure (b) comes from (Bauer et al., 2017) and illustrates Optimal Information Transport. By computing explicit diffeomorphisms (right) that match a uniform law with an arbitrary density map (top), researchers can sample continuous distributions with high sample rates (bottom).

Applications. In higher dimensions, multi-variate Gaussian laws are parameterized by mean *vectors* and covariance *matrices*. By restricting the Fisher-Rao metric to this family of probability measures, researchers can endow the cone of symmetric, positive definite matrices with the so-called *affine-invariant metric* (Pennec et al., 2006). Remarkably, this gem of applied geometry is now routinely deployed on MRI scans: related algorithms provide a robust baseline for the processing of **Diffusion Tensor Images** (Pennec, 2008).

Going further, **Optimal Information Transport** lifts the Fisher-Rao geometry to spaces of deformations of the ambient space $\mathcal{X} = \mathbb{R}^\mathrm{D}$ (Bauer et al., 2015). Associated algorithms may be used to find diffeomorphic mappings between densities at an affordable cost: applications to medical imaging and sampling theory are shown in (Bauer et al., 2018). As illustrated in Figure 3.5, the KL formula has thus applications that go way beyond the cross-entropy layer used in machine learning to perform **logistic regressions**.

Geodesics. This is all well and good. But should we pick the KL divergence above any other formula? To understand the metric structures that we define on spaces of positive measures, a sensible starting point is to look at *geodesics*, i.e. continuous paths of *minimal length*:

$$\gamma \;:\; t \in [0,1] \;\mapsto\; \gamma_t \in \mathcal{M}^+(\mathcal{X}) \tag{3.70}$$

that join a source distribution $\gamma_0 = \alpha$ to a target $\gamma_1 = \beta$.

Let us focus on the Fisher-Rao geodesic of Figure 3.4.a. We see that according to the geometry induced by the KL loss, the "least action" transition between confident left- and right-wing distributions passes through a *highly diffuse* medium point. This modelling assumption fits well with e.g. social sciences, where measures could be used to encode personal opinions in the feature space $\mathcal{X}$ of political ideas.

Discrete measures and relative entropy. Going further, the invariance of the KL formula to affine changes of coordinates in the feature space $\mathcal{X} = \mathbb{R}$ allows us to show that:

$$\begin{aligned}
\mathrm{KL}(\mathcal{N}(m,\sigma),\, \mathcal{N}(m,\sigma/2)) &= \mathrm{KL}(\mathcal{N}(m,\sigma/2),\, \mathcal{N}(m,\sigma/4)) && (3.71)\\
&= \mathrm{KL}(\mathcal{N}(m,\sigma/4),\, \mathcal{N}(m,\sigma/8)) && (3.72)\\
&= \ldots && (3.73)
\end{aligned}$$

At the limit, following Zeno's paradox, this property implies that **discrete Dirac measures are rejected outside of the domain**, infinitely far away from continuous distributions. This is, at heart, the reason why we must state that $\mathrm{KL}(\alpha, \beta) = +\infty$ whenever α has no density with respect to β.

Geometric loss functions. Putting degenerate distributions out of reach is acceptable for researchers who only ever need to consider diffuse probability laws. Geometers, however, have been working with idealized points, curves and surfaces since the days of Euclid: they need to rely on loss functions that can handle discrete *and* continuous measures alike.

In the same vein, promoting a default behaviour that is compatible with the linear structure on the feature space $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$ is often more sensible than dealing with the uneven behaviour of the Fisher-Rao geodesics, illustrated Figure 3.4.a. We say that a loss function **lifts the distance on the feature space** $\mathcal{X}$ to the general family of measures $\mathcal{M}^+(\mathcal{X})$ if for all Dirac masses δ_x and δ_y :

$$\mathrm{Loss}(\delta_x, \delta_y) \;=\; \|x-y\| \;\text{ or }\; \tfrac{1}{2}\|x-y\|^2\,, \qquad (3.74)$$

with geodesic $\gamma_{x\to y} : t \in [0,1] \mapsto \delta_{(1-t)x+ty} \in \mathcal{M}^+(\mathcal{X})$. In words: if a Loss takes "geometric" values on pairs of atomic measures, identified with points x and y of the ambient space $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$.

As discussed in Figure 3.4.b, the Wasserstein-2 metric introduced in Section 3.2.4 is arguably the most appealing of all **geometric loss functions**: it provides intuitive linear interpolations and stays defined through an explicit mathematical expression. It should be understood as a well-defined L^2-Euclidean metric on the *material*, *Lagrangian* particles that make up our distributions.

3.2.2 Hausdorff distances: iterative closest points and mixture models

A first idea: nearest-neighbor projections. From a pragmatic perspective though, no fancy mathematical theory is needed to satisfy the lifting condition of Eq. (3.74). The simple formula:

$$\mathrm{ICP}(\alpha,\beta) \overset{\text{def.}}{=} \tfrac{1}{4}\sum_{i=1}^{\mathrm{N}} \alpha_i \min_{j=1}^{\mathrm{M}} \|x_i - y_j\|^2 \;+\; \tfrac{1}{4}\sum_{j=1}^{\mathrm{M}} \beta_j \min_{i=1}^{\mathrm{N}} \|x_i - y_j\|^2\,, \qquad (3.75)$$

used by the *iterative closest point* algorithm (Besl and McKay, 1992) and its many variants works just fine. Ubiquitous in 3D point cloud processing, this well-known functional relies on **nearest-neighbor projections**. It can be re-cast in the measure-function duality paradigm by introducing the *distance fields to the supports*:

$$a(\,x\,) \overset{\text{def.}}{=} \tfrac{1}{2}\,\mathrm{d}(\,x,\, \mathrm{Supp}(\alpha))^2 \qquad \text{and} \qquad b(\,x\,) \overset{\text{def.}}{=} \tfrac{1}{2}\,\mathrm{d}(\,x,\, \mathrm{Supp}(\beta))^2\,, \qquad (3.76)$$

two continuous functions $a, b : \mathcal{X} \to \mathbb{R}$ which allow us to write that:

$$\text{ICP}(\alpha,\beta) \;=\; \tfrac{1}{2}\langle \alpha,\, b\rangle \;+\; \tfrac{1}{2}\langle \beta,\, a\rangle \;=\; \tfrac{1}{2}\langle \alpha-\beta,\, b-a\rangle \;, \tag{3.77}$$

since $\langle \alpha,\, a\rangle = 0 = \langle \beta,\, b\rangle$. We retrieve a good-looking, quadratic-like formula that involves the *difference of functions* $(b-a)$ integrated with respect to the *difference of measures* $(\alpha-\beta)$. The expression above is usually called the **chamfer distance** in computer vision, where the computation of distance maps a and b – *chamfer transforms* – to the support of measures α and β using fast marching methods is a critical operation (Borgefors, 1984).

SoftMin regularization. Nearest-neighbor projections can be softened by introducing the SoftMin operator "$\min_\varepsilon$", defined for any continuous expression $f : \mathcal{X} \to \mathbb{R}$ by:

$$\underset{x\sim\alpha}{\min{}_\varepsilon}\; f(x) \;\overset{\text{def.}}{=}\; -\varepsilon\, \log \int_{\mathcal{X}} \exp\Big(-\tfrac{1}{\varepsilon} f(x)\Big)\, \mathrm{d}\alpha(x)\;. \tag{3.78}$$

Implemented using the efficient and stable Log-Sum-Exp reduction presented around Eq. (2.43), this operation **interpolates between a minimum and a sum**. As discussed in Section A.3.2, we can show that if α is a probability law:

$$\underset{x\sim\alpha}{\min{}_\varepsilon}\; f(x) \xrightarrow{\varepsilon\to 0} \min_{x\in\text{Supp}(\alpha)} f(x) \tag{3.79}$$

$$\xrightarrow{\varepsilon\to+\infty} \langle \alpha,\, f\rangle\;. \tag{3.80}$$

Gaussian Mixture Models. If $\sigma > 0$ is a positive regularization scale, the *soft distance fields*:

$$a_\sigma(\,y\,) \;\overset{\text{def.}}{=}\; \underset{x\sim\alpha}{\min{}_\varepsilon}\; \tfrac{1}{2}\|x-y\|^2 \qquad \text{and} \qquad b_\sigma(\,x\,) \;\overset{\text{def.}}{=}\; \underset{y\sim\beta}{\min{}_\varepsilon}\; \tfrac{1}{2}\|x-y\|^2 \tag{3.81}$$

associated to a *temperature* $\varepsilon = \sigma^2$ are proportional to the negative log-likelihoods of mixture models, built from our two distributions using a **Gaussian kernel** k_σ of deviation σ:

$$a_\sigma \;\propto\; -\log(k_\sigma \star \alpha) \qquad \text{and} \qquad b_\sigma \;\propto\; -\log(k_\sigma \star \beta)\;. \tag{3.82}$$

Generalizing Eq. (3.77) in a principled way, formulas such as:

$$\text{GMM-log}(\alpha,\beta) \;\overset{\text{def.}}{=}\; \tfrac{1}{2}\langle \alpha-\beta,\, b_\sigma - a_\sigma\rangle \;=\; \tfrac{1}{2}\Big\langle\, \alpha-\beta\,,\, \log\frac{k_\sigma\star\alpha}{k_\sigma\star\beta}\,\Big\rangle \tag{3.83}$$

– or asymetric variations in the mould of $\langle \alpha,\, b_\sigma\rangle$ – are extremely popular in statistics and computational geometry: they appear whenever researchers try to **maximize the likelihood of a mixture model** or express their algorithms within an **Expectation-Maximization** framework. Note, however, that they do *not* define positive loss functions.

Hausdorff divergences. Following an established tradition in computer graphics (Bouaziz et al., 2016), we refer to the formulas discussed in Eqs. (3.75,3.83) as soft- or integrated-**Hausdorff** loss functions. Detailed in Section A.2, our main proofs on entropic optimal transport rely on a positive and definite variation of Eq. (3.83), introduced in Eq. (A.40).

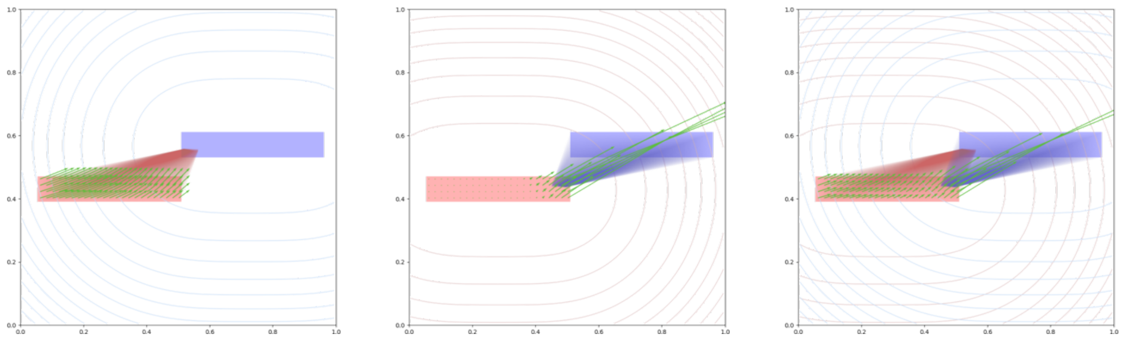

(a) Loss $= \frac{1}{2}\langle \alpha,\, b \rangle$. (b) Loss $= \frac{1}{2}\langle \beta,\, a \rangle$. (c) Loss $= \frac{1}{2}\langle \alpha - \beta,\, b - a \rangle$.

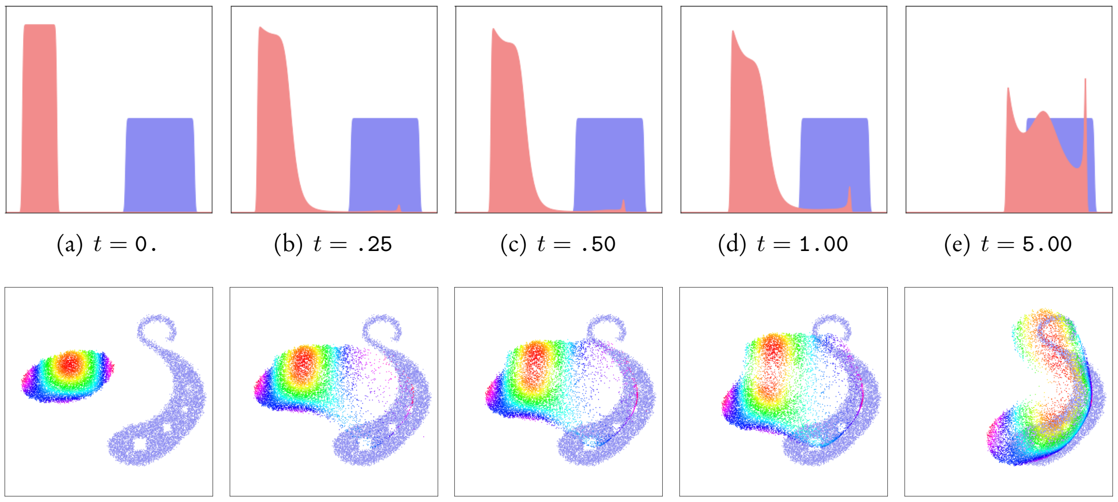
Figure 3.6: Projections in the Soft-Hausdorff loss function produce localized gradients.
In these pictures, our measures α – in red – and β – in blue – are displayed as solid shapes in the unit square $\mathcal{X} = [0,1] \times [0,1]$. Distance fields a – in red – and b – in blue – are displayed using contour lines and computed using Eq. (3.81), which generalizes Eq. (3.76). The (opposite) gradients $-\frac{1}{\alpha_i}\nabla_{x_i}\text{Loss}(\alpha,\beta)$ of our loss functions with respect to the particles that make up the red measure α are displayed as green vector fields, while red and blue lines figuratively represent the “springs” that link points x_i and y_j to their nearest neighbors in the other point cloud, as suggested by Eq. (3.75).
(a) The first term $\langle \alpha,\, b \rangle$ of Hausdorff-like formulas is the integral of α in the distance field generated by β. Its gradient $\nabla b(x_i)$ urges particles x_i to run straight towards their **nearest neighbors** in the target measure β.
(b) The second term $\langle \beta,\, a \rangle$ is the integral of β in the distance field generated by α. Its gradient has a strong influence on the points x_i of α that are **close to the target** β, but leaves the other ones untouched.
(c) By combining these two (simple) behaviors, researchers can define affordable Loss functions. Unfortunately though, as illustrated in Figure 3.7, the resulting gradient is of **very low quality** and can only be used in optimization pipelines after a heavy-handed **regularization** step.

Geometric intuitions, gradient flows. As detailed in Figure 3.6, Hausdorff loss functions rely on nearest-neighbor projections that induce heterogeneous, degenerate gradient fields. **Unfortunately, they are thus somewhat ill-suited to generic measure-fitting problems.** Can we make this statement more specific?

To design illustrative experiments and compare geometric loss functions with each other, we rely on *unregularized*, particle-based, “Wasserstein” gradient flows (Santambrogio, 2017). Working in a reference feature space – the **unit interval** $\mathcal{X} = [0,1] \subset \mathbb{R}$ or the **unit square** $\mathcal{X} = [0,1] \times [0,1] \subset \mathbb{R}^2$ – we sample points (x_i) and (y_j) according to known probability laws: in all our experiments, uniform distributions over shape-like domains of $\mathcal{X}$. In practice, picking $\text{N} = \text{M} =$ `10,000` points per shape, we work with the discrete probability measures:

$$\alpha = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \delta_{x_i} \qquad \text{and} \qquad \beta = \frac{1}{\text{M}} \sum_{j=1}^{\text{M}} \delta_{y_j}\,, \tag{3.84}$$

endowed with uniform weights $\alpha_i = 1/\text{N}$ and $\beta_j = 1/\text{M}$. We then focus on a typical optimization problem: the **minimization of Loss**(α, β) **with respect to the positions** (x_i) of the samples that make up the model measure α, as we try to *fit* a model measure α to the fixed target distribution β using an arbitrary loss function.


(a) $t = 0.$ (b) $t = .25$ (c) $t = .50$ (d) $t = 1.00$ (e) $t = 5.00$

(f) $t = 0.$ (g) $t = .25$ (h) $t = .50$ (i) $t = 1.00$ (j) $t = 5.00$

Figure 3.7: Gradient flow of the Hausdorff loss function defined Eq. (3.83) in 1D and 2D. The blurring scale is set to $\sigma = 0.10$, at around 10% of the configurations' diameters. Source and target measures are discretized with $\mathrm{N} = \mathrm{M} = 10{,}000$ samples as we iterate the Euler scheme of Eq. (3.85). (first line) We start with discrete measures α and β sampled uniformly on the intervals $[0.0, 0.2]$ and $[0.6, 1.0]$. Both measures are displayed using *kernel density estimations* – i.e. convolutions with a small unit-mass kernel – over the real line and are respectively associated with the red and blue colors. (second line) We start with discrete measures α and β sampled from an ellipse and a saxophone-like shape in the unit square $\mathcal{X} = [0,1] \times [0,1]$. The samples $x_i(t)$ that make up our model distribution α are displayed using a rainbow colormap, allowing us to track the trajectories of individual particles.

A reference toy problem: Wasserstein gradient flows. In line with the recent literature in machine learning, we tackle this problem by performing **gradient descent** on the x_i's and update the positions of the samples iteratively, starting from our initial configuration at time $t = 0$. As discussed in Section 2.1.2 (Automatic differentiation), computing gradients $\nabla_{x_i}\mathrm{Loss}(\alpha, \beta)$ is now a mere formality.

Keep in mind, though, that each point x_i is associated to a Dirac atom $\alpha_i\delta_{x_i} = \frac{1}{\mathrm{N}}\delta_{x_i}$ whose influence in the source measure is proportional to the positive weight $\alpha_i = 1/\mathrm{N}$. To define trajectories which do *not* vary with the number of samples N, we normalize the velocity fields by $1/\alpha_i$ and iterate the loop:

$$\forall t \geqslant 0, \qquad x_i(t+\delta t) \;\leftarrow\; x_i(t) \;-\; \delta t\, \tfrac{1}{\alpha_i} \nabla_{x_i}\mathrm{Loss}\Big(\frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}} \delta_{x_i(t)}\,,\, \frac{1}{\mathrm{M}}\sum_{j=1}^{\mathrm{M}} \delta_{y_j} \Big)\,, \qquad (3.85)$$

with a fixed *learning rate* $\delta t = 0.01 > 0$. These conventions enforce parameterization invariance and are well-suited to a theoretical analysis. As far as machine learning is concerned, this toy minimization algorithm corresponds to a **model- or network-free optimization**, where a *reconstruction error* is directly minimized with respect to the samples' positions.

Referred to as **Wasserstein gradient flows** in the continuous case, evolution equations in the mould of Eq. (3.85) have recently become a standard tool for the study of partial differential equations (Jordan et al., 1998): we refer to (Santambrogio, 2015, 2017) for an overview.

Hausdorff divergences provide low-quality gradients. Illustrated in Figure 3.7, this simple experiment allows us to visualize the information encoded by the usual gradient vector:

$$\nabla_{x_i}\mathrm{Loss}(\alpha,\beta) ~=~ \frac{\partial \mathrm{Loss}(\alpha,\beta)}{\partial x_i}~, \tag{3.86}$$

without any external influence.

What does it say? Unfortunately, as evidenced by the erratic evolution of our densities, the raw vector field induced by Hausdorff-like formulas cannot be relied upon to produce smooth, intuitive interpolations between geometric measures. Following the qualitative analysis of Figure 3.6, the model measure α tends to be **torn apart** in two stages: first, some extremal points are attracted quickly towards the support of β, in a bid to minimize the "α to β" interaction term $\langle\beta,\, a\rangle$; second, regular points x_i follow through and go in straight line towards their nearest neighbors y_j at the risk of splitting α into pieces.

In a real pipeline: Riemmanian gradient flows. Needless to say, this naive dynamics is of little practical interest… But the results of this toy experiment do not imply that Hausdorff distances are useless. Far from it. Crucially, in most real-life applications, the x_i's are *not* updated using the simple rule of Eq. (3.85) but generated as the output of a parametric transform:

$$\Phi ~:~ (p_k) \in \mathbb{R}^{\mathrm{K}} ~\mapsto~ \Phi(p_k) ~=~ (x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}~, \tag{3.87}$$

as we perform gradient descent on the vector of **parameters** (p_k) :

$$\forall\, t \geqslant 0,~~ p_k(t+\delta t) ~\leftarrow~ p_k(t) ~-~ \delta t\, \nabla_{p_k}\mathrm{Loss}\circ\Phi(p_k(t))~. \tag{3.88}$$

Sub-Riemannian gradient descent. Let us remark that if Φ is differentiable, the chain rule reads – with the notations of Eq. (2.20) :

$$\begin{aligned}
\nabla_{p_k}\mathrm{Loss}\circ\Phi(p_k) ~&=~ \mathrm{d}_{p_k}^{\top}\mathrm{Loss}\circ\Phi(p_k)~\cdot~1 && (3.89)\\
&=~ \mathrm{d}_{p_k}^{\top}\Phi(p_k)~\cdot~\mathrm{d}_{x_i}^{\top}\mathrm{Loss}~\cdot~1 && (3.90)\\
&=~ \mathrm{d}\Phi(p_k)^{\top}\,\nabla_{x_i}\mathrm{Loss}(x_i)~, && (3.91)
\end{aligned}$$

where $\mathrm{d}\Phi$ denotes the Jacobian matrix of the parametric model Φ. Consequently, at order 1 in δt, the updates on the array $(x_i) = \Phi(p_k)$ read:

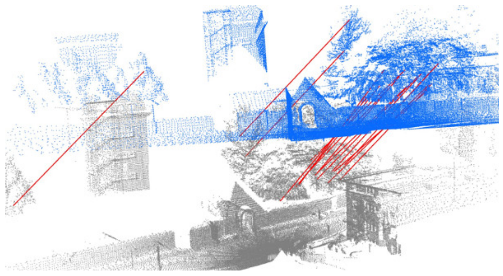
$$\forall\, t \geqslant 0, \qquad x_i(t+\delta t) ~\leftarrow~ x_i(t) ~-~ \delta t\, \underbrace{\mathrm{d}\Phi(p_k)\,\mathrm{d}\Phi(p_k)^{\top}}_{\mathbf{K}(p_k)}\,\nabla_{x_i}\mathrm{Loss}(x_i)~. \tag{3.92}$$

Real-life gradient descent is all about **regularizing the naive gradient vector** through the application of a symmetric, positive, semi-definite operator $\mathbf{K}$ that is encoded within the differential of the generative map Φ. In some favorable cases – say, if Φ is injective – the tensor $\mathbf{K}(p_k)$ can be expressed as a function of the measure α encoded by the x_i's, and we interpret Eq. (3.92) as a sub-Riemannian gradient descent scheme for the semi-definite pseudo-metric:

$$\|\delta x_i\|_{x_i}^2 ~\overset{\text{def.}}{=}~ (\delta x_i)^{\top}\,(\mathbf{K}(x_i))^{-1}\,\delta x_i~. \tag{3.93}$$

As discussed in Section 5.2.3, we can indeed remark that in the small-δt limit:

$$x_i - \delta t\,\mathbf{K}(x_i)\nabla_{x_i}\mathrm{Loss}(x_i) ~=~ \arg\min_{x_i'}~\big[\mathrm{Loss}(x_i') ~+~ \tfrac{1}{2\delta t}\|x_i'-x_i\|_{x_i}^2\,\big] ~+~ o(\delta t)~. \tag{3.94}$$


(a) Point Cloud Library (Rusu and Cousins, 2011).

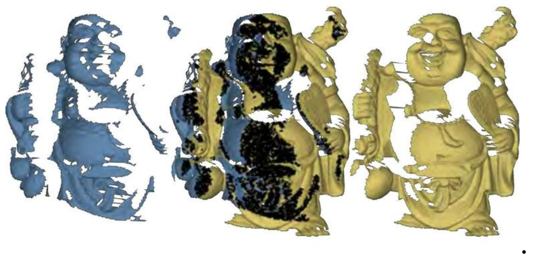

(b) Sparse ICP (Bouaziz et al., 2013)

Figure 3.8: Hausdorff-like distances are ubiquitous for 3D affine and rigid registration.

Affine shape registration. The regularizing operator $\mathbf{K}$ encodes the *modelling prior* that drives descent algorithms towards relevant minima and prevents the generated measure α from leaving the set of *acceptable models*. Understanding its behavior in complex situations – e.g. when Φ is a convolutional neural network – is a challenging problem (Amari, 1998; Ulyanov et al., 2018). In simple explicit settings however, the analysis can be straightforward and provide critical insights: see, for instance, the archetypal example of **shape registration with affine transformations.**

In this setting, the β_j's and y_j's are set and describe some fixed *target shape* β discretized with M weighted points. The measure α, on the other hand, is generated using a low-rank transform. A *template shape* is encoded by a reference set of weights (α_i) and points $\widetilde{x} = (\widetilde{x}_i)$, which allows us to generate the moving samples (x_i) using:

$$\Phi_{\text{aff}} \;:\; (A, B) \,\in\, \mathbb{R}^{\mathrm{D}\times\mathrm{D}} \times \mathbb{R}^{1\times\mathrm{D}} \;\mapsto\; (x_i) \;=\; \widetilde{x}\,A + \mathbf{1}B \,\in\, \mathbb{R}^{\mathrm{N}\times\mathrm{D}}\,, \tag{3.95}$$

an affine mapping parameterized by a vector $p = (A, B)$ of size $\mathrm{D}^2 + \mathrm{D}$ – here, $\mathbf{1}$ denotes the constant N-by-1 array whose entries are equal to 1. Computations show that the operator $\mathbf{K}$ associated to this *deformation layer* has a simple analytic form:

$$\mathbf{K} \;:\; v \,\in\, \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \;\mapsto\; \widetilde{x}\,(\widetilde{x}^\top v) \,+\, \mathbf{1}\,(\mathbf{1}^\top v) \,\in\, \mathbb{R}^{\mathrm{N}\times\mathrm{D}}\,. \tag{3.96}$$

Crucially, $\mathbf{K} = \widetilde{x}\widetilde{x}^\top + \mathbf{1}\mathbf{1}^\top$ is a *low-rank* symmetric matrix which discards most of the information encoded within the raw descent direction $v = -\nabla_{x_i}\text{Loss}(x_i)$. With at most $\mathrm{D}^2 + \mathrm{D}$ non-zero eigenvalues, it can be described as a *projection plus scaling* operator onto the low-dimensional space of affine deformations of $\widetilde{x}$. As long as the gradient vector field v points roughly in the correct direction, iterative affine registration algorithms should thus be able to improve their matchings without ever tearing the template shape apart – the deformation model Φ_{aff} is just way too constrained.

Conclusion. As illustrated in Figure 3.8, Hausdorff-like loss functions provide the reference baseline for pose estimation and mesh reconstruction methods. Thanks to the intrinsic robustness of affine and rigid deformation models, authors in the field are now able to post-process Hausdorff gradients using outlier detection methods (Aiger et al., 2008; Ma et al., 2014) and tackle extremely challenging configurations – with noise or partial acquisitions – in real-time.

Unfortunately though, performances break down as soon as the generative model Φ stops being as constrained as a low-rank affine deformation. The ideal, unregularized setting of Eq. (3.85) – where $\mathbf{K} \propto \text{Id}_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}}$ – should act as a strong motivation for the development of **high-quality geometric loss functions** whose gradients can be relied upon to drive flexible, possibly *learned* generative models in situations that are closer to Figure 3.7 than to Figure 3.8.b.

3.2.3 Kernel methods: Sobolev metrics, MMDs and charged particles

A second idea: using convolutions. How should we proceed? To retrieve a smooth, non-degenerate gradient flow from the good-looking formula of Eq. (3.77), a simple idea is to **replace distance fields by convolutions**. That is, to pick a symmetric *kernel* function:

$$k \,:\, (x,y) \in \mathcal{X} \times \mathcal{X} \,\mapsto\, k(x,y) = k(x-y) \in \mathbb{R} \tag{3.97}$$

and work with the *influence* fields or *potential*:

$$a_k \overset{\text{def.}}{=} -k \star \alpha \qquad\qquad \text{and} \qquad\qquad b_k \overset{\text{def.}}{=} -k \star \beta\,. \tag{3.98}$$

Our quadratic loss function then reads:

$$\text{Kernel}_k(\alpha,\beta) \overset{\text{def.}}{=} \tfrac{1}{2}\langle \alpha - \beta,\, b_k - a_k\rangle \;=\; \tfrac{1}{2}\langle \alpha-\beta,\, k\star(\alpha-\beta)\rangle \overset{\text{def.}}{=} \tfrac{1}{2}\|\alpha-\beta\|_k^2\,. \tag{3.99}$$

Such a formula can be implemented using the tools and routines presented around Figure 3.3. Assuming that $\alpha = \sum_{i=1}^{\text{N}} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{\text{M}} \beta_j \delta_{y_j}$ are two discrete measures, we can simply develop the loss value as a combination of three double sums:

$$\text{Kernel}_k(\alpha,\beta) \;=\; \tfrac{1}{2}\langle \alpha,\, k\star\alpha\rangle - \langle \alpha,\, k\star\beta\rangle + \tfrac{1}{2}\langle \beta,\, k\star\beta\rangle \tag{3.100}$$

$$= \tfrac{1}{2}\sum_{i=1}^{\text{N}}\sum_{j=1}^{\text{N}} \alpha_i\alpha_j\, k(x_i,x_j) \;-\; \sum_{i=1}^{\text{N}}\sum_{j=1}^{\text{M}} \alpha_i\beta_j\, k(x_i,y_j) \;+\; \tfrac{1}{2}\sum_{i=1}^{\text{M}}\sum_{j=1}^{\text{M}} \beta_i\beta_j\, k(y_i,y_j)\,, \tag{3.101}$$

performed efficiently using the `KeOps` routines of Chapter 2. Alternatively, when $\mathcal{X} = \mathbb{R}^{\text{D}}$, we can leverage the Fourier transform and convolution theorem of Figure 3.2 to write that:

$$\text{Kernel}_k(\alpha,\beta) \;=\; \frac{1}{2}\int_{\omega\in\mathbb{R}^{\text{D}}} \widehat{k}(\omega)\,\big|\,\widehat{\alpha}(\omega) - \widehat{\beta}(\omega)\big|^2\,\mathrm{d}\omega\,. \tag{3.102}$$

As discussed below, this identity will incite us to focus on kernel functions k whose Fourier transform $\widehat{k}$ is positive on the spectral domain $\mathbb{R}^{\text{D}}$, thus ensuring that Kernel_k is a positive and definite quantity.

Kernel methods. Formulas in the mould of Eqs. (3.99-3.101) are **ubiquitous in applied sciences**: from physics to machine learning, applying a convolution is the simplest way of modelling spatial correlations and pair-wise interactions. Unfortunately though, few papers and textbooks take the time to draw explicit links between fields that have, at first glance, very little in common. Before going any further, we devote a few pages to a short panorama around the six major interpretations of Eq. (3.99). As we identify with each other the theories of:

1. **Newtonian gravitation and electrostatics** in physics,
2. **blurred squared distances** in imaging sciences,
3. **Sobolev norms** in functional analysis,
4. **maximum mean discrepancies** in statistics,
5. **reproducing kernel Hilbert spaces** in machine learning and
6. **Kriging, splines** or **Gaussian processes** in geostatistics, imaging and probabilities,

we will hopefully help the reader to get a deeper understanding of a theory that is central to modern data sciences.

First interpretation: generalized electrostatics. Historically, convolutions were first introduced by Newton in his *Principia Mathematica* to model universal gravitation. When k is the *Coulomb kernel* given by:

$$k(x,y) \overset{\text{def.}}{=} \begin{cases} 0 & \text{if } x = y \,, \\ \frac{1}{\|x-y\|} & \text{otherwise} \,, \end{cases} \tag{3.103}$$

the continuous potential:

$$-k \star \alpha \;:\; x \in \mathbb{R}^{\mathrm{D}} \;\mapsto\; -\sum_{i=1}^{\mathrm{N}} \frac{\alpha_i}{\|x - x_i\|} \tag{3.104}$$

is the *gravitational potential* generated by the distribution of mass α. Its spatial gradient is the opposite of the *gravitational acceleration vector*:

$$\vec{g}(x) \overset{\text{def.}}{=} \nabla[k \star \alpha](x) \;=\; \sum_{i=1}^{\mathrm{N}} \alpha_i \,\frac{x_i - x}{\|x - x_i\|^3} \,, \tag{3.105}$$

whose norm proverbially decays following an *inverse square law* of the distances between the current point x and the mass locations x_i in $\mathcal{X} = \mathbb{R}^3$.

Going further, the classical theory of *electrostatics* generalizes these equations to *signed* distributions. If $+\alpha$ and $-\beta$ are respectively understood as "positive" and "negative" distributions of electric charges (say, protons $+\alpha_i$ and electrons $-\beta_j$ at locations x_i and y_j), the loss formula of Eqs. (3.99-3.101) gives the *potential energy* stored in the global electrostatic interaction between all pairs of particles. The gradient vector:

$$\vec{E}(x_i) \overset{\text{def.}}{=} -\tfrac{1}{\alpha_i}\nabla_{x_i}\text{Kernel}_k(\alpha,\beta) \;=\; -\nabla\big[k \star (\alpha - \beta)\big](x_i) \tag{3.106}$$

$$= \underbrace{\sum_{j=1}^{\mathrm{N}} \alpha_j \frac{x_i - x_j}{\|x_i - x_j\|^3}}_{-\partial_{x_i}\frac{1}{2}\langle \alpha, k\star\alpha\rangle} \;+\; \underbrace{\sum_{j=1}^{\mathrm{M}} \beta_j \frac{y_j - x_i}{\|y_j - x_i\|^3}}_{+\partial_{x_i}\langle \alpha, k\star\beta\rangle} \tag{3.107}$$

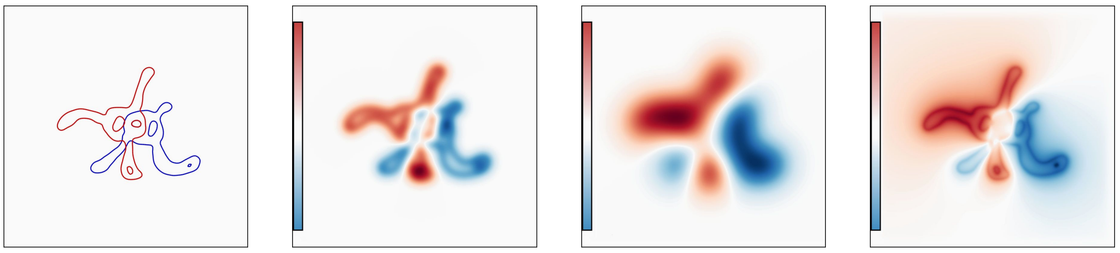
is the *electric field* $\vec{E}(x_i)$ applied by the total distribution of charge $(\alpha - \beta)$ on the particle x_i of charge $+\alpha_i$. It is the superposition of *repulsive* and *attractive* terms, respectively generated by the distributions $+\alpha$ and $-\beta$.

In practice, using a kernel loss in the mould of Eq. (3.99) thus amounts to working with a **generalized electrostatic model**: the spatial decay of the interaction is encoded by the profile of the potential k. For instance, a Gaussian kernel:

$$k(x-y) \overset{\text{def.}}{=} e^{-\|x-y\|^2/2\sigma^2} \quad \text{with gradient} \quad \nabla_x k(x-y) \;=\; \tfrac{x-y}{\sigma^2} e^{-\|x-y\|^2/2\sigma^2} \tag{3.108}$$

models an interaction that is strong when $\|x - y\| \simeq \sigma$ and negligible otherwise.

Applying the particle-based gradient flow of Eq. (3.85) to a kernel loss is akin to simulating the evolution of a dampened system of charged particles. Known as *electrostatic halftoning* (Schmaltz et al., 2010) in some communities, this method provides a simple yet effective baseline for image stippling. Coupled with higher-level generative models for the x_i's, it is a fundamental method for shape registration and machine learning, as discussed at the end of this section.



(a) Raw data. (b) Small kernel. (c) Large kernel. (d) Peak + Heavy tail.

Figure 3.9: Influence of the blurring kernel g on the convolution $g \star (\alpha - \beta)$ (b-d) computed from the signed measure $(\alpha - \beta)$ (a). (b) When the kernel has a small support, $g \star \alpha$ and $g \star \beta$ barely interact with each other: the kernel Loss is little more than a glorified pointwise distance. (c) On the other hand, large smooth kernels can over-blur the data and effectively prevent measure-fitting or *registration* algorithms from reaching a satisfying accuracy level. (d) As far as *geometric* applications are concerned, good baselines are provided by kernels that are both pointy and with a wide support: they preserve the high-frequency content of the input data and allow samples to interact with each other at long range.

Second interpretation: blurred sum of squared distances. Far away from physics, a second interpretation of the kernel loss comes from *imaging*. As seen in Figure 3.9, if $g : \mathbb{R}^\mathrm{D} \rightarrow \mathbb{R}$ is a convolution kernel, the continuous function $g \star (\alpha - \beta)$ provide a blurry view on the sharp input measures α and β. Its standard L^2 norm is given by the *sum of squared distances* formula:

$$\tfrac{1}{2}\| \, g \star (\alpha - \beta) \|^2_{L^2(\mathbb{R}^\mathrm{D})} \;=\; \int_{x \in \mathbb{R}^\mathrm{D}} \big| \, [g \star (\alpha - \beta)](x) \big|^2 \, \mathrm{d}x \tag{3.109}$$

$$=\; \int_{\omega \in \mathbb{R}^\mathrm{D}} \big| \, \widehat{g}(\omega) \big|^2 \, \big| \, \widehat{\alpha}(\omega) - \widehat{\beta}(\omega) \big|^2 \, \mathrm{d}\omega \; . \tag{3.110}$$

We can thus identify this quantity with the kernel norm associated to the symmetric kernel:

$$k \;=\; (g \circ (x \mapsto -x)) \, \star \, g \, , \tag{3.111}$$

whose Fourier transform:

$$\widehat{k}(\omega) \;=\; |\widehat{g}(\omega)|^2 \;=\; \overline{\widehat{g}(\omega)} \, \widehat{g}(\omega) \tag{3.112}$$

is real-valued and non-negative. In practice, the introduction of a *point spread function* g allows practitioners to create *overlap* between neighboring samples in α and β. As image registration algorithms strive to minimize loss fidelity terms in the mould of Eq. (3.109), they typically align structures who “see” each other thanks to these cheap $(\alpha - \beta) \mapsto g \star (\alpha - \beta)$ filtering passes.

Third interpretation: dual of a Sobolev norm. Kernel losses are fundamental quantities in physics and imaging. Remarkably, they can also be cast in the measure-function duality paradigm using sets of *smooth adversarial test functions*: unit balls in generalized Sobolev spaces.

What does this jargon implies? In functional analysis, for the study of partial differential equations, mathematicians tend to rely on functions whose derivatives are square-integrable –

of finite energy – and therefore bounded in a global sense. For instance, on the real line $\mathcal{X} = \mathbb{R}$, the *Sobolev space* $H^1 = W^{1,2}$ of finite-energy functions with finite-energy derivatives is made up of all (weakly) differentiable functions $f : \mathbb{R} \to \mathbb{R}$ such that:

$$\int_{-\infty}^{+\infty} |f(x)|^2 \,\mathrm{d}x \;<\; +\infty \qquad \text{and} \qquad \int_{-\infty}^{+\infty} |f'(x)|^2 \,\mathrm{d}x \;<\; +\infty\,. \tag{3.113}$$

It is endowed with the H^1-Sobolev norm defined through:

$$\|f\|_{H^1}^2 \;=\; \int_{-\infty}^{+\infty} |f(x)|^2 \;+\; |f'(x)|^2 \,\mathrm{d}x \;<\; +\infty\,, \tag{3.114}$$

which penalizes large values *and* large derivatives. The fundamental *Sobolev embedding theorem* shows that generalized functions of finite H^1 norm can be represented as *continuous* functions: relying on pointwise evaluations, their integrals with respect to Dirac masses are well-defined quantities. Mimicking the construction of the Total Variation written Eq. (3.62), we can thus define the *dual H^{-1} norm* through:

$$\|\alpha - \beta\|_{H^{-1}} \;=\; \max_{\|f\|_{H^1} \leqslant 1} \langle \alpha - \beta,\, f\,\rangle\,, \tag{3.115}$$

for any positive Radon measures α and β in $\mathcal{M}^+(\mathbb{R})$.

Intuitively, restricting the maximization problem above to *continuous* test functions is a way of retrieving a *geometric* behaviour. If we compare a model $\alpha = \delta_{1/n}$ with a target $\beta = \delta_0$, the Total Variation norm can always pick a step function:

$$f \;:\; x \in \mathbb{R} \;\mapsto\; \begin{cases} -1 & \text{if } x < 1/2n\,, \\ +1 & \text{otherwise}\,. \end{cases} \tag{3.116}$$

It then saturates at:

$$\mathrm{TV}(\delta_{1/n}, \delta_0) \;=\; \langle \delta_{1/n} - \delta_0,\, f\rangle \;=\; f(1/n) - f(0) \;=\; 1 - (-1) \;=\; 2 \tag{3.117}$$

for any value of n, which is useless. More interestingly, the H^{-1} norm defined above can only pick test functions f whose derivative is square integrable, with:

$$\int_{-\infty}^{+\infty} |f(x)|^2 \;+\; |f'(x)|^2 \,\mathrm{d}x \;\leqslant\; 1\,. \tag{3.118}$$

Such functions can *not* make sharp jumps between $1/n$ and 0 : $\|\delta_{1/n} - \delta_0\|_{H^{-1}}$ should thus somewhat reflect the *geometric* proximity between the two Dirac masses, as n tends to infinity.

These hand-waving explanations can be made rigorous using Fourier analysis: the H^{-1} norm has a **simple closed-form expression** as a Kernel$_k$ loss. To derive it, we first remark that using Parseval's identity:

$$\begin{aligned} \|f\|_{H^1}^2 \;&=\; \int_{-\infty}^{+\infty} |f(x)|^2 \;+\; |f'(x)|^2 \,\mathrm{d}x && (3.119)\\ &\propto\; \int_{-\infty}^{+\infty} |\widehat{f}(\omega)|^2 \;+\; |\,\widehat{f'}(\omega)|^2 \,\mathrm{d}\omega && (3.120)\\ &=\; \int_{-\infty}^{+\infty} |\widehat{f}(\omega)|^2 \;+\; |\,i\omega\widehat{f}(\omega)|^2 \,\mathrm{d}\omega && (3.121)\\ &=\; \int_{-\infty}^{+\infty} (1+\omega^2)\,|\widehat{f}(\omega)|^2 \,\mathrm{d}\omega\,. && (3.122) \end{aligned}$$

Then, using the Fourier expressions of kernel norms from Eq. (3.102), we can re-cast the optimization problem of Eq. (3.115) as:

$$\|\alpha - \beta\|_{H^{-1}} \;=\; \max_{f:\mathbb{R}\to\mathbb{R}} \int_{-\infty}^{+\infty} \overline{[\widehat{\alpha}(\omega) - \widehat{\beta}(\omega)]}\,\widehat{f}(\omega)\,\mathrm{d}\omega \tag{3.123}$$

$$\text{subject to} \quad \int_{-\infty}^{+\infty} (1+\omega^2)\,|\widehat{f}(\omega)|^2\,\mathrm{d}\omega \;\leqslant\; 1\,. \tag{3.124}$$

Introducing a Lagrange multiplier or leveraging the Cauchy-Schwarz inequality, we see that the optimal Fourier transform $\widehat{f}(\omega)$ is necessarily proportional to $(\widehat{\alpha}(\omega) - \widehat{\beta}(\omega))/(1+\omega^2)$. Direct computations then show that:

$$\|\alpha - \beta\|^2_{H^{-1}} \;=\; \int_{-\infty}^{+\infty} \frac{1}{1+\omega^2}\,|\,\widehat{\alpha}(\omega) - \widehat{\beta}(\omega)|^2\,\mathrm{d}\omega\,. \tag{3.125}$$

Simply put, the H^{-1} norm on the real line $\mathcal{X} = \mathbb{R}$ is thus **equal to the kernel norm** associated to the exponential kernel:

$$k(x) \;=\; e^{-|x|} \qquad \text{whose Fourier transform is given by} \qquad \widehat{k}(\omega) \;=\; \frac{1}{1+\omega^2}\,, \tag{3.126}$$

up to the usual multiplicative constants for Fourier transforms.

In the general case, mathematicians work with Sobolev spaces that are defined:

1. For arbitrary differentiation orders m, with high-order terms appended to the high-pass polynomial in $1 + \omega^2 + \cdots + \omega^{2m}$.
2. In high-dimensional settings, as $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$ is some arbitrary vector space. Generally, $\mathrm{D} = 2$ or 3 for applications in fluid mechanics.

Anyway: as long as a Sobolev space H^m satisfies the hypotheses of the embedding theorem (i.e. if $m > \mathrm{D}/2$), its dual norm H^{-m} can be understood through the lens of Eq. (3.99), with a continuous kernel k that becomes smoother as m increases.

Fourth interpretation: maximum mean discrepancies. The theory of Sobolev spaces is at the heart of modern analysis... But in practice, data scientists have little use for its calculus-centric formalism. Pragmatically, statisticians prefer to specify the profile of admissible test functions f using explicit kernel functions k on the feature space $\mathcal{X}$.

Assuming that a kernel k is well-behaved – e.g. continuous, square-integrable and with a positive Fourier transform – the space of k-smooth functions V_k is that of all functions $f = k\star\mu$ such that $\langle \mu,\, k\star\mu\rangle$ is finite, for arbitrary signed distributions μ. It is endowed with the dual metric induced by the *deconvolution* operator $k^{(-1)} \star \cdot$:

$$\|f\|^2_{k^{(-1)}} \;\overset{\text{def.}}{=}\; \langle k^{(-1)} \star f,\, f\rangle \;\overset{\text{def.}}{=}\; \langle \mu,\, k\star\mu\rangle \;\text{ with }\; f = k\star\mu\,. \tag{3.127}$$

Computations similar to those of the previous paragraph then show that the kernel norm of Eq. (3.99) is given through the expected adversarial formulation:

$$\|\alpha - \beta\|_k \;=\; \max_{\|f\|_{k^{(-1)}} \leqslant 1} \langle \alpha - \beta,\, f\rangle\,. \tag{3.128}$$

Following a terminology that was coined by (Gretton et al., 2012), kernel norms are generally called *(kernel) Maximum Mean Discrepancies* (MMDs) in the machine learning literature: the *maximum* refers to the dual optimization problem, the *mean* to the integral operator in the duality bracket and the *discrepancy* to the difference of input measures $(\alpha - \beta)$.

The most iconic example of a non-Sobolev MMD is the *Gaussian kernel norm.* If $\sigma > 0$ is the standard deviation of a Gaussian kernel:

$$k_\sigma \;:\; x \in \mathbb{R}^{\mathrm{D}} \;\mapsto\; \frac{1}{(\sigma\sqrt{2\pi})^{\mathrm{D}}}\, e^{-\|x\|^2/2\sigma^2}\;, \tag{3.129}$$

the Fourier transform $\widehat{k_\sigma}$ of k_σ is simply equal to $k_{1/\sigma}$. Consequently, the canonical dual norm on the set of k_σ-smooth function reads:

$$\|f\|^2_{k_\sigma^{(-1)}} \;=\; \int_{-\infty}^{+\infty} e^{+\|\sigma\omega\|^2/2}\, |\widehat{f}(\omega)|^2\,\mathrm{d}\omega\;, \tag{3.130}$$

with a penalization of high frequencies in:

$$\frac{1}{\widehat{k_\sigma}(\omega)} \;\propto\; e^{+\|\sigma\omega\|^2/2} \tag{3.131}$$

that grows (much) faster than any polynomial in $1+\|\omega\|^2+\cdots+\|\omega\|^{2m}$ induced by the differentiations of a Sobolev norm. In practice, Gaussian kernel norms all but prevent their test functions from expressing frequencies outside of an admissible lowpass band in the $[-3/\sigma, +3/\sigma]$ range.

Fifth interpretation: kernel methods. The theory of Reproducing Kernel Hilbert Spaces (RKHS) generalizes these computations even further, by *removing* the assumption that the smoothing operator $k \star \cdot$ acts through a translation-invariant kernel $k(x, y) = k(x - y)$.

Formalized by analysts such as Aronszajn and Schwartz in the 1950's, the theory of RKHS can be described as the study of Hilbert spaces of functions for whom the pointwise evaluation is a well-defined, continuous linear form. Tanks to the Riesz representation theorem, this is equivalent to the study of dot products $\langle\,\cdot\,,\,\cdot\,\rangle_{V_k}$ that induce a complete metric structure $\|\cdot\|_{V_k}$ on spaces V_k of functions $f : \mathcal{X} \rightarrow \mathbb{R}$ and are such that:

$$\forall\, x \in \mathcal{X},\; \exists\, g_x \in V_k,\; \forall\, f \in V_k,\; \langle \delta_x, f\rangle \overset{\text{def.}}{=} f(x) \;=\; \langle\, g_x,\, f\,\rangle_{V_k}\,. \tag{3.132}$$

This axiom implies the existence of a symmetric, continuous function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ that characterizes the dot product $\langle\,\cdot\,,\,\cdot\,\rangle_{V_k}$ and is such that:

$$\forall\, x, y \in \mathcal{X},\;\; g_x(y) \;=\; g_y(x) \overset{\text{def.}}{=} k(x, y)\,. \tag{3.133}$$

The *kernel* k satisfies the so-called *reproducing* property:

$$\forall\, x, y \in \mathcal{X},\;\; \langle\, k(\,\cdot\,, x),\, k(\,\cdot\,, y)\,\rangle_{V_k} \;=\; \langle\, \delta_x,\, k(\,\cdot\,, y)\,\rangle \;=\; k(x, y)\,. \tag{3.134}$$

In practice, this abstract theory allows researchers to generalize results proved on $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$ to general feature spaces, e.g. permutation groups for genomics. From the 90's onwards, *kernel methods* have become a staple of the machine learning literature: we refer to (Shawe-Taylor et al., 2004) for an extensive overview.

Sixth interpretation: Gaussian processes. With an ever-widening scope, kernel theory can be used to specify the "typical regularity" of functions in Hilbert spaces V_k, whose unit balls then induce the family of Kernel$_k$ losses through the dual formulation of Eq. (3.128).

In applications that range from geology – **Kriging** – to shape analysis – **spline registration** – the simplest way of leveraging such an assumption of *regularity* is to specify a *Gaussian prior* on the space of admissible signals. Formally, this comes down to endowing the space $L^2(\mathbb{R}^\mathrm{D})$ of square-integrable functions with a probability distribution whose density with respect to the "volume" measure is proportional to:

$$\frac{\mathrm{d}\mathcal{N}(0, \|\,\cdot\,\|_{V_k}^2)}{\mathrm{d}\lambda_{L^2(\mathbb{R}^\mathrm{D})}}(f) \;\propto\; e^{-\|f\|_{V_k}^2/2\sigma^2}\,. \tag{3.135}$$

In the infinite-dimensional Hilbert space $L^2(\mathbb{R}^\mathrm{D})$, the iso-likelihood sets of this distribution are given, for $t \geqslant 0$, by the equation:

$$t \;=\; \|f\|_{V_k}^2 \;=\; \int_{-\infty}^{+\infty} \frac{1}{\widehat{k}(\omega)}\,\big|\widehat{f}(\omega)\big|^2\,\mathrm{d}\omega\,. \tag{3.136}$$

Assuming that $\widehat{k}$ is positive, we recognize an ellipsoid whose principal directions are colinear to the Fourier harmonics $e_\omega : x \in \mathbb{R}^\mathrm{D} \mapsto e^{i\omega\cdot x} \in \mathbb{C}$, with axes lengths proportional to $\sqrt{\widehat{k}(\omega)}$.

Kernel regression. In practice, data is often provided as a collection of labelled points $(x_i, f_i) \in \mathbb{R}^\mathrm{D} \times \mathbb{R}$ for i in $[\![1,\mathrm{N}]\!]$, understood as the sampled values f_i at locations x_i of an underlying signal $f : \mathbb{R}^\mathrm{D} \to \mathbb{R}$ that we strive to infer. Under the regularity assumption of Eq. (3.135), the most likely candidate is the solution of the quadratic optimization problem:

$$f^* \;=\; \arg\min_{f\in V_k} \|f\|_{V_k}^2 \qquad \text{subject to} \qquad \forall\, i \in [\![1,\mathrm{N}]\!]\,,\; f(x_i) \;=\; f_i\,. \tag{3.137}$$

The optimal competitor is thus a function which is "as k-smooth as possible" while taking the prescribed values f_i at the sampling locations x_i.

From a geometric perspective, f^* is the *orthogonal projection* of $0 \in V_k$ onto the affine subspace defined by the constraints of Eq. (3.137). Using the representation formula of Eqs. (3.132-3.133), this domain can be rewritten as an intersection of (affine) hyperplanes:

$$\text{Constraints}(x_i, f_i) \;=\; \bigcap_{i=1}^{\mathrm{N}} \{\, f \in V_k,\ \langle\, k(x_i,\,\cdot\,),\, f\,\rangle_{V_k} \;=\; f_i\,\}\,. \tag{3.138}$$

Since f^* belongs to its orthogonal for the $\langle\,\cdot\,,\,\cdot\,\rangle_{V_k}$ dot product:

$$[\text{Constraints}(x_i, f_i)]^{\perp V_k} \;=\; \text{Vect}(\, k(x_i,\,\cdot\,),\ i \in [\![1,\mathrm{N}]\!]\,)\,, \tag{3.139}$$

it can thus be written as a linear combination of the N elementary functions $k(x_i,\,\cdot\,)$. Otherwise said, there exists a vector of weights $(\mu_i) \in \mathbb{R}^\mathrm{N}$ such that:

$$\forall\, x \in \mathbb{R}^\mathrm{D},\;\; f^*(x) \;=\; \sum_{i=1}^{\mathrm{N}} \mu_i k(x, x_i) \qquad \text{i.e.} \qquad f^* \;=\; k \star \sum_{i=1}^{\mathrm{N}} \mu_i\,\delta_{x_i}\,. \tag{3.140}$$

The statement above is known as the *representer theorem* in the machine learning literature. If we introduce the N-by-N **kernel matrix** $(K_{x,x})_{i,j} = k(x_i, x_j)$, the equality constraints of Eq. (3.137) imply that:

$$(f_i) \;=\; K_{x,x}\,(\mu_i) \;\in\; \mathbb{R}^{\mathrm{N}}\,. \tag{3.141}$$

We can thus solve the *kernel* or *Gaussian process regression problem* in two steps:

1. Solve the linear system of Eq. (3.141) to compute the optimal vector of weights:
$$(\mu_i) \;=\; K_{x,x}^{-1}\,(f_i)\,. \tag{3.142}$$
2. Evaluate f^* at any sampling location $x \in \mathbb{R}^{\mathrm{D}}$ using the closed-form solution of Eq. (3.140).

Ridge kernel regression. Alongside N-body simulation, this method has been the *major motivation* behind the development of kernel-related routines since the 1950's, when it was introduced for geostatistics by Krige and Matheron. Note however that in practice, fitting a perfectly smooth model to a real-world dataset is not a sensible thing to do: the linear operator $K_{x,x}$ is often badly conditioned. Practitioners generally assume that their data is *noisy* and strive to solve the *regularized* interpolation problem:

$$f^* \;=\; \arg\min_{f\in V_k} \quad \|f\|_{V_k}^2 \;+\; \frac{1}{\alpha}\sum_{i=1}^{\mathrm{N}} |\, f(x_i) - f_i\,|^2 \tag{3.143}$$

$$\text{or } \mu^* \;=\; \arg\min_{\|\mu\|_k<+\infty} \langle \mu,\, k\star\mu\rangle \;+\; \frac{1}{\alpha}\sum_{i=1}^{\mathrm{N}} |\,[k\star\mu](x_i) - f_i\,|^2\,, \tag{3.144}$$

for some positive value of the regularization parameter α, as we replace the hard equality constraints of Eq. (3.137) with a smooth quadratic penalty.

This correction was historically introduced to model a *nugget effect* and allow smooth terrain models to handle irregular real-life samples. In practice, adjusting the computations of the last few paragraphs, we can show that solving the regression problem of Eq. (3.143) with discrete samples is just a matter of replacing Eq. (3.142) in the first step of our algorithm by:

$$(\mu_i) \;=\; (\alpha\,\mathrm{Id} + K_{x,x})^{-1}\,(f_i)\,. \tag{3.145}$$

In Chapter 2, around Eq. (2.55) and Figure 2.9, we discussed some of the *computational* aspects of solving these kernel linear systems. Notably, we explained how the `KeOps` library could provide a `x30` to `x100` speed-up to state-of-the-art solvers in the general case. In some favorable settings, however, there is no need for such machinery. For instance, when k is a B-spline kernel sampled on a regular grid of x_i's, the kernel matrix $K_{x,x}$ has a band-diagonal structure that can be leveraged with $O(\mathrm{N})$ solvers: we refer to classic papers such as (Takeda et al., 2007) for an extensive discussion.

Feature maps. The explanations above have allowed us to introduce an important quantity: the discrete *kernel matrix* $K_{x,x}$ of pair-wise values $k(x_i, x_j)$. In probability theory, statistics and machine learning, a large body of work tends to interpret these scalar quantities as *correlations* or *dot products* in a convenient feature space. Indeed, under technical assumptions on the kernel k, it is often possible to build a *feature map*:

$$\varphi\ : x \in \mathcal{X} \mapsto \varphi(x) \in F \qquad \text{such that} \qquad k(x,y) = \langle \varphi(x), \varphi(y) \rangle_F\ , \tag{3.146}$$

where F is an arbitrary Hilbert space endowed with a dot-product $\langle \cdot\,, \cdot \rangle_F$.

This **kernel trick** allows researchers to understand the kernel matrix $K_{x,x}$ as the *Gram matrix* of a point cloud $(\varphi(x_i))_{i\in[\![1,\mathrm{N}]\!]}$, image of our dataset by a domain-dependent feature extractor. Plugging a kernel matrix into a *geometric* algorithm to act as a "non-linear Gram matrix" is thus a legitimate operation. As evidenced by the popularity of kernel-SVM and other kernelized machine learning algorithms, the combination of robust statistical routines with custom kernel matrices strikes a good balance between power and simplicity.

Please note that in probability theory and (geo)statistics, F is often a space of random variables "X" or "Y" endowed with the standard dot product:

$$\langle X, Y \rangle_F = \mathbb{E}[X \cdot Y]\ . \tag{3.147}$$

In these fields, $K_{x,x}$ is thus generally called the *covariance matrix* while k is the *covariance function* or *variogram*: we refer to (Rasmussen, 2003) for an introduction.

Positivity, universality. All the results and identities presented in this section hold for symmetric, translation invariant, continuous and square-integrable kernels whose Fourier transform is *positive*. As discussed in Eq. (3.102), the associated Kernel$_k$ loss is a positive, definite functional. And as detailed around Eq. (3.109), it can be understood as a simple L^2 norm seen through the lens of the feature mapping:

$$\varphi_k\ :\ x \in \mathbb{R}^{\mathrm{D}} \mapsto g \star \delta_x \in L^2(\mathbb{R}^{\mathrm{D}}) \qquad \text{with} \qquad \widehat{g}(\omega) = \sqrt{\widehat{k}(\omega)}\ . \tag{3.148}$$

An overwhelming majority of papers in data sciences focus on Gaussian, exponential or Cauchy kernels that are respectively defined through:

$$k(x) \propto e^{-\|x\|^2/2\sigma^2}\ , \qquad k(x) \propto e^{-\|x\|/\sigma}\ , \qquad k(x) \propto \frac{1}{1+\|x/\sigma\|^2} \tag{3.149}$$

and satisfy these hypotheses.

More generally, most of the interesting results in kernel theory hold for continuous, symmetric functions $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that are **"positive definite"**, i.e. are such that:

$$\forall\, \mathrm{N} > 0,\ \forall\, (x_i) \in \mathcal{X}^{\mathrm{N}},\ \ K_{x,x} = \big(k(x_i, x_j)\big) \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}} \ \text{ is a positive-definite matrix.} \tag{3.150}$$

Under this assumption, **Mercer's theorem** is indeed able to provide an explicit feature map φ and legitimize kernel analysis as an application of Euclidean geometry in some (infinite-dimensional) Hilbert space F of feature vectors. A weaker but common assumption on k is **conditional positivity**, which is only imposed on the space of zero mean vectors and is most relevant to the study of *probability* measures.

Finally, researchers typically assume that their kernels are **universal**, i.e. that finite kernel expansions of the form $\sum_{i=1}^{\mathrm{N}} \mu_i k(\cdot, x_i)$ – for any number $\mathrm{N} > 0$ of points $(x_i)_i \in \mathcal{X}^{\mathrm{N}}$ and weights $(\mu_i)_i \in \mathbb{R}^{\mathrm{N}}$ – are dense in $\mathcal{C}(\mathcal{X})$ for the $\|\cdot\|_\infty$ norm (Micchelli et al., 2006). In the translation-invariant setting, this condition is roughly equivalent to asking the Fourier transform of k to be *positive* instead of being merely *non-negative*, and ensures that the associated kernel norm metrizes the convergence in law.

A flat metric structure. As discussed in the last ten pages or so, kernel norms are backed by an impressive body of work. In subsequent chapters of this manuscript, we will heavily rely on this influential theory to lay the foundations of computational anatomy. But **are kernel losses really suited to *geometric* applications?**

Following the guidelines of Figure 3.4, we first remark that kernel geodesics have a straightforward expression. Since kernel norms are induced by a *linear* convolution operator $k \star \cdot$, they are compatible with the algebraic structure on the vector space of signed measures $\mathcal{M}(\mathcal{X})$: the shortest path between any two distributions α and β is given by:

$$\gamma_{\alpha\rightarrow\beta} \;:\; t \in [0,1] \;\mapsto\; (1-t)\alpha \;+\; t\beta\,. \tag{3.151}$$

At heart, kernel norms thus induce a **pointwise**, **Eulerian** geometry on spaces of measures. Unlike the relative entropy KL, discussed in Section 3.2.1, Kernel_k losses can handle continuous *and* discrete distributions alike. They metrize the convergence in law. But out-of-the-box, they interpolate between Dirac atoms δ_x and δ_y through manipulations on *weights*, instead of *moving* samples in the feature space as prescribed around Eq. (3.74). Showcased in Figure 3.14.a, the *fading* interpolations induced by kernel losses do *not* leverage the metric structure of the feature space.

Electrostatic screening, vanishing gradients. Fortunately, the gradient flow of Eq. (3.85) provides a simple way of retrieving a particle-based dynamics out of any loss function. As discussed in Section 4.2.2, relying on a *kernel norm* or *MMD* to drive a measure-fitting pipeline is a standard procedure in shape analysis, image processing and machine learning. But **are all kernels created equal?**

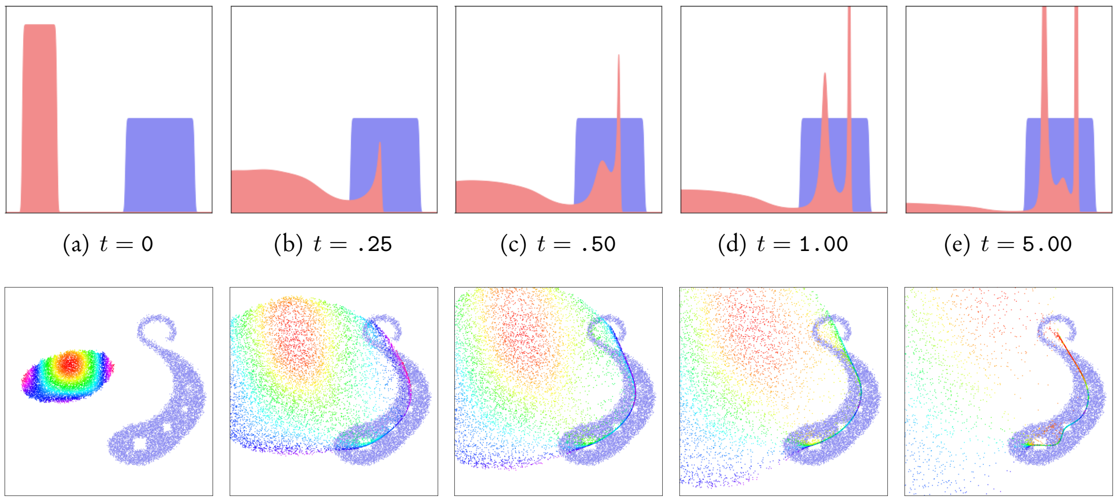
Naively, we could be inclined to believe so. After all, from a theoretical perspective, any universal, positive definite kernel induces a Kernel_k loss that metrizes the convergence in law. In practice though, as evidenced by the catastrophic dynamics of Figure 3.10, combining gradient descent with a kernel norm can be a dramatic mismatch.

To understand this phenomenon, we must cast aside the refined theories of RKHS or Sobolev spaces and come back to our first physical intuitions: Kernel_k losses are **generalized electrostatic energies**. As we minimize $\frac{1}{2}\|\alpha-\beta\|_k^2$ with respect to the x_i's, "protons" $+\alpha_i\delta_{x_i}$ naturally repulse each other while being attracted to the "electrons" $-\beta_j\delta_{y_j}$. Unsurprisingly, the repulsive term generated by close neighbors in the expression of the electric field $\vec{E}(x_i)$, Eq. (3.106), often dominates the feeble attractive force generated by the far-away target β.

Picking a sensible kernel. To mitigate this weakness, the choice of a suitable kernel k is essential. For geometric applications, practitioners should focus on potentials whose attractive strength $-\nabla k(x)$ does not vanish too early: a **heavy tail** is a desirable property. On the other hand, geometric kernels should **not be too smooth**: as evidenced in Figure 3.9.c, large regular kernels can quickly blur-out the small-scale (i.e. high-frequency) details present in the input data.

From this perspective, the ever-popular Gaussian kernel is **one of the worst possible choices**: the Gaussian bell combines an extreme smoothness – discussed in Eqs. (3.129-3.131) – with a compact support, as for all practical purposes:

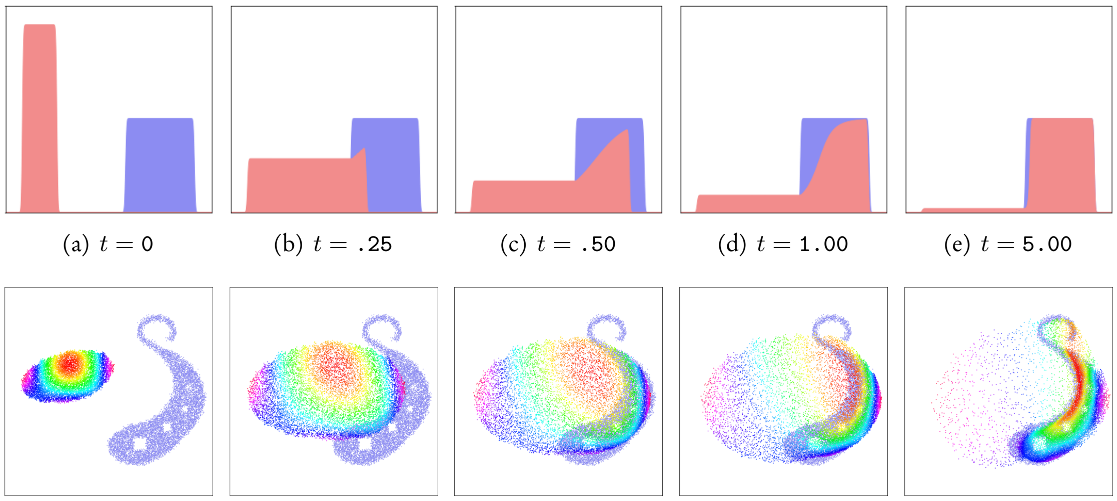
$$1 \;=\; e^{-0^2/2} \;>>\; e^{-5^2/2} \;=\; 0\,. \tag{3.152}$$



(a) $t = 0$ (b) $t = .25$ (c) $t = .50$ (d) $t = 1.00$ (e) $t = 5.00$

(f) $t = 0$ (g) $t = .25$ (h) $t = .50$ (i) $t = 1.00$ (j) $t = 5.00$

Figure 3.10: Gradient flow of a Gaussian Kernel$_k$ loss in the unit interval (a-e) and the unit square (f-j). Here, the setting is the same as that of Figure 3.7 and $k(x, y) = \exp(-\|x-y\|^2/2\sigma^2)$, with $\sigma = 0.1$. As evidenced by these two dynamics, following the gradient of a positive and definite loss function does **not** necessarily allow measure-fitting pipelines to converge towards satisfying alignments. The "explosion" showcased here is due to the repulsive interaction exerced by the "positive charges" $+\alpha_i \delta_{x_i}$ onto another. Eventually, most of the samples x_i end up outside of the effective support of the target's potential field $b_k = -k \star \beta$ and lay around motionless, scattered across the domain.



(a) $t = 0$ (b) $t = .25$ (c) $t = .50$ (d) $t = 1.00$ (e) $t = 5.00$

(f) $t = 0$ (g) $t = .25$ (h) $t = .50$ (i) $t = 1.00$ (j) $t = 5.00$

Figure 3.11: Gradient flow of the "Energy Distance" Kernel$_k$ loss in the unit interval (a-e) and the unit square (f-j). Here, the setting is the same as that of Figure 3.7 and $k(x, y) = -\|x-y\|$. Thanks to its pointiness and (very) heavy tail, the Energy Distance kernel provides an **excellent baseline** for geometric applications. At the cost of a single convolution per evaluation, it allows practitioners to retrieve a *monotonic* convergence of the model α towards the blue target β. Its main weakness is common to all kernel losses: **electrostatic screening**. Due to the combination of attractive and repulsive terms in the gradient of Eq. (3.106), *vanishing gradients* are always present: some points x_i (here, on the top-left) converge extremely slowly towards their targets, as the measure α gets stretched without good reason.

A Gaussian kernel of deviation σ is essentially blind to all points beyond a maximum *reach* of $\sim 3\sigma$ and to all details which are smaller than a $\sim \sigma/2$ *blur scale*. Taken separately, these properties could be guarantees of *robustness* with respect to *outliers* and discrete *sampling locations*. But as they are tied together through a single parameter σ, these two features severely impact the **reliability** of Gaussian kernel losses in applied settings: catastrophic failures along the lines of Figure 3.10 are a common occurence in computational anatomy.

This state of things is most unfortunate: Gaussian kernels are generally well-understood by practitioners, well-suited to other applications (e.g. clustering) and supremely affordable on grid images and volumes, where they can be implemented using *separable* convolution filters. A common strategy is therefore to rely on *sums of Gaussians*:

$$k(x-y) \;=\; \sum_{i=1}^{\mathrm{K}} w_i \, e^{-\|x-y\|^2/2\sigma_i^2}\,, \tag{3.153}$$

with new hyper-parameters to tune: the scales $(\sigma_i) \in \mathbb{R}^{\mathrm{K}}_{>0}$ and weights $(w_i) \in \mathbb{R}^{\mathrm{K}}$.

The Energy Distance. This strategy can provide satisfying results, but is generally brittle and a bit cumbersome. As a robust and reliable baseline for practitioners, we would like to recommend the (underrated) Energy Distance kernel:

$$k(x,y) \;=\; -\|x-y\|\,. \tag{3.154}$$

It is well-known is statistics (Szekely and Rizzo, 2005) and generally induces good-looking monotonic flows, displayed Figure 3.11, **without any parameter to tune**. The associated potential:

$$a_{-\|\cdot\|}(x) \;=\; -[\,-\|\cdot\|\star\alpha\,](x) \;=\; \sum_{i=1}^{\mathrm{N}} \alpha_i\|x-x_i\| \;=\; \mathbb{E}_{A\sim\alpha}[\,\|x-A\|\,] \tag{3.155}$$

is a natural generalization of the distance field to a measure, and we always observe that:

$$\text{Kernel}_{-\|\cdot\|}(\delta_x,\delta_y) \;=\; \|x-y\|\,. \tag{3.156}$$

Note, however, that the "minus norm" kernel is only *conditionally positive*: the positivity of the associated $\text{Kernel}_{-\|\cdot\|}$ loss is guaranteed only if α and β have the same mass. The $\text{Kernel}_{-\|\cdot\|}$ loss should thus be used on the *normalized* probability distributions $\alpha/\langle\alpha,\,1\rangle$ and $\beta/\langle\beta,\,1\rangle$.

Conclusion. All in all, robust kernel norms such as the Energy Distance provide sensible gradient vectors at an affordable computational cost: a mere convolution through the data. Their properties have been a strong motivation for the development of efficient numerical schemes, from Fast Multipole Methods (Greengard, 1988; Beatson and Greengard, 1997) to the `KeOps` library.

Nevertheless, as far as geometric applications are concerned, kernel norms suffer from **two important limitations**: their geodesic interpolating curves rely on *pointwise* fadings instead of *geometric* deformations; and due to *electrostatic screening*, their *gradients vanish* or become inhomogoneous in ways that often mislead finely grained generative models.

Removing these artifacts without giving up on fast runtimes is the main motivation for the work presented in the next few pages.

3.2.4 Optimal Transport: Wasserstein distance and generalized Quicksort

A third idea: sorting points. To go beyond the limitations of convolutions and kernel norms, we rely on the theory of **Optimal Transport** (OT) which generalizes **sorting** to feature spaces $\mathcal{X}$ of dimension $\mathrm{D} > 1$. Also known as **linear assignment**, OT is all about casting point set registration as a linear optimization problem. Pairwise correspondences between weighted point clouds can then be used to endow the space of probability measures $\mathcal{M}_1^+(\mathcal{X})$ with a particle-based geometry, known as the **Wasserstein metric** or **Earth Mover's Distance**.

A principled body of work. Over the last 40 years, this framework has progressively entered the standard toolbox in applied mathematics as strong French and Italian schools gathered around the topic. **Excellent textbooks** on the subject are available: albeit slightly outdated, (Villani, 2003) and (Villani, 2008) are still reference works in pure mathematics while (Santambrogio, 2015) and (Peyré and Cuturi, 2017) respectively cover the standard results for applications to functional analysis and data sciences. We highly recommend these monographs to readers who would like to get a solid background on the topic, and refer to them as well as to the lecture notes (Vialard, 2019) for standard proofs.

Keep in mind, however, that **all authors have their biases**. Staying close to its author's main interests, (Santambrogio, 2015) is written with a strong focus on PDE flows; meanwhile, (Peyré and Cuturi, 2017) is mostly concerned with *theoretical* convergence rates and the popularization of OT in a wide range of applied settings.

In this manuscript, which is all about **real-life performances for geometric applications**, we focus on the link between Optimal Transport and sorting. This connection is especially relevant for low-dimensional feature spaces and allows us to highlight four points:

1. OT is *not* a generic optimization problem. Whenever possible, leveraging its geometric structure is the key to unlock efficient $O(\mathrm{N}\,\log \mathrm{N})$ schemes that can be understood as high-dimensional generalizations of the **Quicksort** algorithm.

2. OT can be highly relevant, but is **not a magic solution** to all registration problems. Just like sorting algorithms, OT matches samples according to the metric structure of the feature space: nothing more, nothing less. If features are not properly engineered, OT is thus no more relevant than alphabetical or random sorting. Keeping this in mind is most important in machine learning and image processing, where high-dimensional embeddings can be of dubious quality.

3. Crucially, **OT does not preserve the topology** of the input measures. Computational efficiency is achieved at the cost of **simplistic assumptions** on what practitioners may regard as "optimal" matchings: users get what they pay for, and should always keep in mind the counter-examples of Figure 3.31.

4. OT has a **rich history** and naturally appears in many applied settings, from physics to combinatorics. Talented mathematicians and computer scientists have been working on the subject since World War 2, with varying vocabularies and motivations. To stop re-inventing the wheel and contribute to the wider scientific community, getting familiar with the literature *outside* of data sciences is a necessary first step.

Optimal Transport in dimension $\mathrm{D} = 1$**.** With these precautions in mind, we first assume that our measures α and β sample *univariate* probability distributions with an equal number of points. That is, we assume that $\mathrm{M} = \mathrm{N}$ and $\mathcal{X} = \mathbb{R}$ with:

$$\alpha = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \delta_{x_i} \qquad \text{and} \qquad \beta = \frac{1}{\mathrm{N}} \sum_{j=1}^{\mathrm{N}} \delta_{y_j}, \tag{3.157}$$

as (x_i), $(y_j) \in \mathbb{R}^{\mathrm{N}}$ and $\alpha_i = \beta_j = 1/\mathrm{N}$. Up to a re-labelling of the samples, we may assume that the x_i's are sorted and find a permutation $\sigma : [\![1, \mathrm{N}]\!] \rightarrow [\![1, \mathrm{N}]\!]$ such that:

$$x_1 \leqslant x_2 \leqslant \cdots \leqslant x_{\mathrm{N}} \qquad \text{and} \qquad y_{\sigma(1)} \leqslant y_{\sigma(2)} \leqslant \cdots \leqslant y_{\sigma(\mathrm{N})} \,. \tag{3.158}$$

The Wasserstein distance. Optimal Transport theory is all about using this canonical ordering of the samples to **pair the** x_i**'s and the** y_j**'s with each other.** On the real line, we define the **Wasserstein-2** loss function through:

$$\mathrm{OT}(\alpha, \beta) \overset{\text{def.}}{=} \frac{1}{2\mathrm{N}} \sum_{i=1}^{\mathrm{N}} |x_i - y_{\sigma(i)}|^2 \,. \tag{3.159}$$

This formula generalizes the squared Euclidean distance on $\mathbb{R}^{\mathrm{N}}$ to clouds of *unlabeled* samples. Direct computations show that $\sqrt{\mathrm{OT}}$ satisfies the triangular inequality and is a distance, usually named after Leonid Vaseršteĭn. Following the recommendations of Eq. (3.74), OT *lifts the distance on the feature space* as:

$$\mathrm{OT}(\delta_x, \delta_y) = \tfrac{1}{2}|x - y|^2 \,, \tag{3.160}$$

with *geometric* shortest paths between Dirac masses:

$$\gamma_{\delta_x \rightarrow \delta_y} \;:\; t \in [0, 1] \;\mapsto\; \delta_{(1-t)x+ty} \in \mathcal{M}^+(\mathbb{R}) \,. \tag{3.161}$$

Remarkably, the OT loss is **differentiable** almost everywhere. Generically, we have that:

$$-\tfrac{1}{\alpha_i} \nabla_{x_i} \mathrm{OT}(\alpha, \beta) = -\tfrac{\mathrm{N}}{2\mathrm{N}} \nabla_{x_i} \big[\, |x_i - y_{\sigma(i)}|^2 \,\big] = y_{\sigma(i)} - x_i \,. \tag{3.162}$$

Otherwise said, with a learning rate δt that is equal to 1, **a single gradient descent step** on $\mathrm{OT}(\alpha, \beta)$ as in Eq. (3.85) is enough to ensure a **perfect matching** of the x_i's onto the y_j's.

Generalization to higher dimensions. These properties are undoubtedly appealing: thanks to a re-ordering of the x_i's and the y_j's, the Wasserstein-2 distance endows the set of unlabeled N-samples in $\mathbb{R}$ with a **purely geometric** Euclidean-like structure. But can we generalize this method to higher dimensions, where no ordering "$\leqslant$" is available? Following the steps of (Monge, 1781), we cast **the sorting operation as an optimization problem on permutations** and define the Wasserstein-2 loss between point clouds (x_i) and (y_j) in $\mathbb{R}^{\mathrm{N} \times \mathrm{D}}$ through:

$$\mathrm{OT}(\alpha, \beta) \overset{\text{def.}}{=} \min_{\substack{\sigma : [\![1,\mathrm{N}]\!] \rightarrow [\![1,\mathrm{N}]\!] \\ \text{permutation}}} \frac{1}{2\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \|x_i - y_{\sigma(i)}\|^2 \,. \tag{3.163}$$

In dimension $\mathrm{D} = 1$, direct computations show that the optimal labelling σ is, indeed, the one that corresponds to a non-decreasing coupling between the x_i's and the y_j's. Picking the "least action" assignment σ is a sensible way of generalizing sorting to arbitrary feature spaces.

Generalization to weighted point clouds. Going further, Kantorovich remarked in (Kantorovich, 1942) that the Monge assignment problem could be generalized to any pair of discrete positive measures:

$$\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i} \qquad \text{and} \qquad \beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j} \tag{3.164}$$

that have the same total mass. If the **cost function** on the feature space $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$ is given by:

$$\mathbf{C} : (x, y) \in \mathbb{R}^{\mathrm{D}} \times \mathbb{R}^{\mathrm{D}} \mapsto \mathbf{C}(x, y) = \tfrac{1}{2}\|x - y\|^2 \in \mathbb{R}_{\geqslant 0} , \tag{3.165}$$

we define the Wasserstein-2 loss between α and β through:

$$\mathrm{OT}(\alpha, \beta) \overset{\text{def.}}{=} \min_{(\pi_{i,j}) \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}_{\geqslant 0}} \sum_{i=1}^{\mathrm{N}} \sum_{j=1}^{\mathrm{M}} \pi_{i,j}\, \mathbf{C}(x_i, y_j) \tag{3.166}$$

$$\text{subject to} \quad \forall\, i, j,\ \pi_{i,j} \geqslant 0, \quad (\pi \mathbf{1})_i \overset{\text{def.}}{=} \sum_{j=1}^{\mathrm{M}} \pi_{i,j} = \alpha_i, \quad (\pi^\top \mathbf{1})_j \overset{\text{def.}}{=} \sum_{i=1}^{\mathrm{N}} \pi_{i,j} = \beta_j \,.$$

Illustrated Figure 3.13.(a,b,d), this equation describes the minimization of the **linear objective** $\pi \mapsto \langle \pi, \mathbf{C} \rangle$ over the set of non-negative **transport plans** $(\pi_{i,j}) \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}_{\geqslant 0}$ whose lines sum-up to α and whose columns sum-up to β.

Formally, these constraints define a **simplex** of co-dimension $(\mathrm{N} + \mathrm{M})$ in the $(\mathrm{N} \cdot \mathrm{M})$-dimensional cone $\mathbb{R}^{\mathrm{N}\times\mathrm{M}}_{\geqslant 0}$. More intuitively, the weighted point cloud α was historically introduced to describe a collection of resources α_i, produced in factories at locations x_i, that were to be allocated to satisfy the needs β_j of Soviet cities at locations y_j. The values of $\mathbf{C}(x_i, y_j)$ describe transportation costs per unit of mass: π should thus be understood as a large **factories$\times$cities allocation plan** that we strive to optimize globally.

Needless to say, the weighted Kantorovich formulation is consistent with the combinatorial Monge problem of Eq. (3.163). When $\mathrm{N} = \mathrm{M}$ and $\alpha_i = \beta_j = 1/\mathrm{N}$ for all i, j in $[\![1, \mathrm{N}]\!]$, standard computations show that any permutation $\sigma : [\![1, \mathrm{N}]\!] \rightarrow [\![1, \mathrm{N}]\!]$ solution of the combinatorial problem of Eq. (3.163) induces a solution of the Kantorovich problem of Eq. (3.166) given by the permutation matrix:

$$\pi_{i,j} = \tfrac{1}{N} \delta_{\sigma(i)=j} = \begin{cases} 1/\mathrm{N} & \text{if } \sigma(i) = j \\ 0 & \text{otherwise.} \end{cases} \tag{3.167}$$

Generalization to continuous measures. Today, mathematicians define the Wasserstein-2 distance between any pair $\alpha, \beta \in \mathcal{M}^+(\mathcal{X})$ of same-mass positive measures through:

$$\mathrm{OT}(\alpha, \beta) \overset{\text{def.}}{=} \min_{\pi \in \mathcal{M}^+(\mathcal{X}\times\mathcal{X})} \langle \pi, \mathbf{C} \rangle \quad \text{subject to} \quad \pi \geqslant 0,\ \pi_1 = \alpha,\ \pi_2 = \beta \,. \tag{3.168}$$

As detailed page 69, these compact notations describe a *linear optimization problem* with respect to a Radon measure π defined on the product space $\mathcal{X} \times \mathcal{X}$. The linear constraints "$\pi_1 = \alpha$" and "$\pi_2 = \beta$" fix the marginals of π, which can thus be understood as a *probabilistic mapping* between the source and target distributions. These constraints imply that the optimal coupling π has density with respect to the product measure $\alpha \otimes \beta$: when α and β are discrete, Eq. (3.168) coincides exactly with Eq. (3.166).

A sensible generalization of the sorting problem. From the **1D sorting** of Eq. (3.159) to the **multi-dimensional** Monge problem of Eq. (3.163) and the **fully general**, continuous Kantorovitch formulation of Eq. (3.168), modern Optimal Transport theory proposes a **consistent** way of generalizing "sorting" to arbitrary feature spaces. Showcased in Figure 3.12, the resulting assignments allow researchers to define particle-based algorithms on (unlabeled) measures at a low computational cost.

Continuous OT retains the **geometric properties** of the discrete sorting operation. Assuming that $\mathbf{C}(x,y) = \frac{1}{2}\|x-y\|^2$ is the quadratic cost of Eq. (3.165), we can show that the **lifting conditions** of Eqs. (3.74,3.160) stay satisfied. Even better: as discussed in Figure 3.4.b, OT defines sensible **linear interpolations** between Gaussian laws and can be used to define an elegant notion of harmonicity for measure-valued applications (Lavenant, 2019a,b).

The dual Kantorovitch formulation. The OT loss can be cast in the measure-function duality paradigm using the adversarial formulations:

$$\mathrm{OT}(\alpha,\beta) = \max_{\substack{(f_i)\in\mathbb{R}^{\mathrm{N}}\\(g_j)\in\mathbb{R}^{\mathrm{M}}}} \sum_{i=1}^{\mathrm{N}} \alpha_i f_i + \sum_{j=1}^{\mathrm{M}} \beta_j g_j \quad \text{s.t. } \forall\, i,j,\ f_i + g_j \leqslant \mathbf{C}(x_i,y_j) \tag{3.169}$$

$$= \max_{f,g\in\mathcal{C}(\mathcal{X})} \langle \alpha,\, f\rangle + \langle \beta,\, g\rangle \quad \text{s.t.} \quad f\oplus g \leqslant \mathbf{C} \tag{3.170}$$

which respectively correspond to the discrete and general cases.

The **equivalence** between the **primal** problem of Eqs. (3.166,3.168) and the **dual** problem above is the fundamental theorem of linear programming. In 1975, Leonid Kantorovitch was awarded the "Nobel" prize in economics for this result, whose traditional interpretation in terms of "external contractors" and market equilibrium is illustrated Figure 3.13.

Dual potentials. Strictly speaking, the dual problem above only interacts with the dual potentials f and g on the **supports of the input measures**, through the vectors of sampled values:

$$f_i = f(x_i) \qquad \text{and} \qquad g_j = g(y_j)\,, \tag{3.171}$$

which are unique up to an additive constant: for all $K\in\mathbb{R}$,

$$(f,g) \text{ is an optimal pair } \Leftrightarrow\ (f+K, g-K) \text{ is also solution of the dual problem.} \tag{3.172}$$

We can, however, canonically **extend** any pair of optimal dual potentials to the **whole domain** $\mathcal{X}=\mathbb{R}^{\mathrm{D}}$ by using the (non-sufficient) optimality equations:

$$\forall\, x\in\mathcal{X},\ f(x) = \min_{y\in\mathrm{Supp}(\beta)} \big[\mathbf{C}(x,y) - g(y)\big] = \min_{j=1}^{\mathrm{M}}\big[\mathbf{C}(x,y_j) - g(y_j)\big]\,, \tag{3.173}$$

$$\forall\, y\in\mathcal{X},\ g(y) = \min_{x\in\mathrm{Supp}(\alpha)} \big[\mathbf{C}(x,y) - f(x)\big] = \min_{i=1}^{\mathrm{N}}\big[\mathbf{C}(x_i,y) - f(x_i)\big]\,, \tag{3.174}$$

which correspond to a saturation of the constraints of the dual problem. The rules of linear programming – *complementary slackness* – ensure that any optimal pair (f,g) satisfies these conditions for $x=x_i$, $y=y_j$. Extending them to the whole domain is a consistent way of ensuring that important quantities such as ∇f and ∇g are **properly defined** almost everywhere. In line with the literature on the topic, we refer to optimal potentials (f,g) that satisfy these equations for all values of x and y in $\mathcal{X}$ as to the **"unique" optimal dual potentials** associated to the transport problem $\mathrm{OT}(\alpha,\beta)$, defined up to an additive constant.

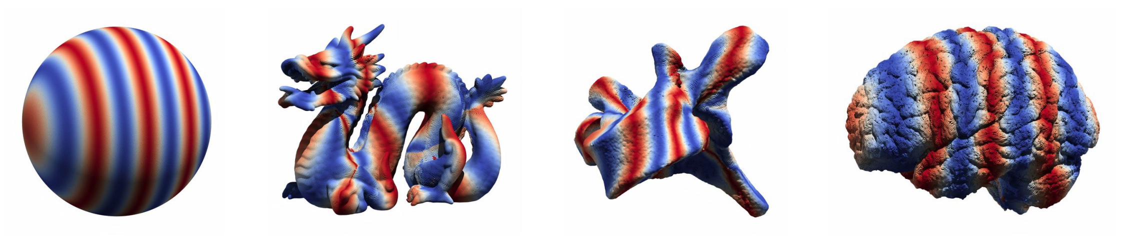


Figure 3.12: Optimal Transport generalizes sorting to spaces of dimension $\mathrm{D} > 1$**.** Here, all measures are sampled from 3D meshes with about `200,000` triangles each, with one weighted Dirac mass $|\text{area}| \cdot \delta_{\text{center}}$ per triangle. As we transport a stripe color scheme from the unit sphere α to the Stanford dragon β, a vertebra β' – courtesy of `bio1260` on `SketchFab` – and a human brain β'', we can observe the structure of the optimal transport plans $\pi_{\alpha\leftrightarrow\beta}$, $\pi_{\alpha\leftrightarrow\beta'}$ and $\pi_{\alpha\leftrightarrow\beta''}$ solutions of the Kantorovitch problem of Eq. (3.166). As discussed from Figure 3.27 to Figure 3.30, allowing practitioners to compute such mappings in **fractions of a second** is the major contribution of this work.

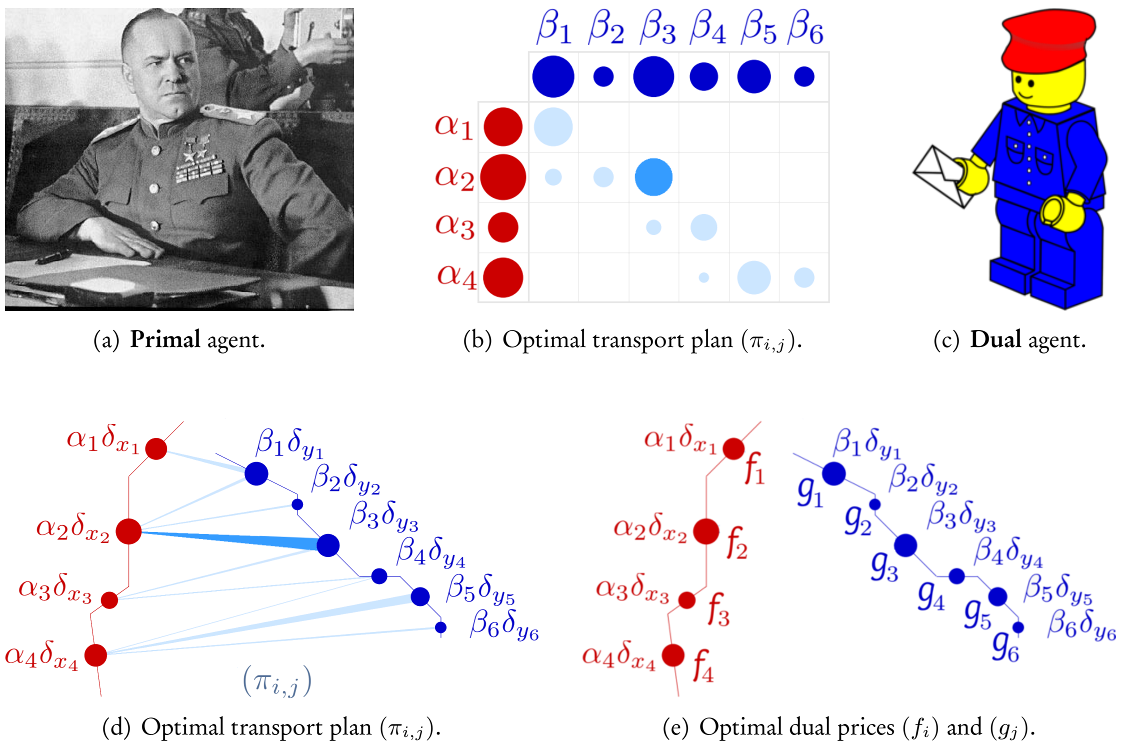


Figure 3.13: Illustrating Kantorovitch's duality. The **primal** problem $\mathrm{OT}(\alpha,\beta)$ of Eqs. (3.166,3.168) models the decision process of a Soviet marshall (a) who wishes to **minimize** the cost of allocating resources from factories x_i to cities y_j (d) using a large N-by-M transport plan (b). On the other hand, the **dual** problem $\mathrm{OT}(\alpha,\beta)$ of Eqs. (3.169,3.170) models the aspirations of an external contractor (c) whose profit margin is directly proportional to the stamp prices f_i and g_j at locations x_i and y_j. The postal network (e) should be set up to **maximize** the total sending + retrieval price $\sum_i \alpha_i f_i + \sum_j \beta_j g_j$ under a **competitive market** constraint: for all pair (x_i, y_j) of destinations, the "FedEx rate" $f_i + g_j$ should stay below the baseline cost $\mathbf{C}(x_i, y_j)$ of sending an army truck from one place to the other. Crucially, Kantorovitch showed in 1942 that the two problems are **equivalent**: there exists a "marshall vs postman" market equilibrium around the optimal value $\mathrm{OT}(\alpha,\beta)$ of both competing problems.

A compact encoding. The most important consequence of the primal-dual equivalence is that the **full information** about an *optimal* transport plan $\pi \in \mathcal{M}^+(\mathcal{X} \times \mathcal{X})$ – an N-by-M matrix – can be summarized within a **single dual potential** $f \in \mathcal{C}(\mathcal{X})$ – a vector of $\mathbb{R}^\mathrm{N}$. Indeed, since $\beta \geqslant 0$, we know that for any set value of f, Eq. (3.174) gives the best possible competitor g that saturates the constraint "$f \oplus g \leqslant \mathbf{C}$"; standard computations then show that the support of $\pi = (\pi_{i,j})$ is concentrated on pairs of points (x_i, y_j) such that $f(x_i) + g(y_j) = \mathbf{C}(x_i, y_j)$. A large N-by-M transportation problem can thus be reduced to the optimization of a single vector of potential *prices* $f_i = f(x_i)$.

The Monge map. In (Brenier, 1991), motivated by his research on the incompressible Euler equation, Yann Brenier pushed this analysis further. Working under the assumption that **C** is, indeed, the quadratic cost function of Eq. (3.165), he showed that OT is all about computing the **gradients of convex potentials**. As detailed for instance in (Peyré and Cuturi, 2017, Remark 2.24), a simple assumption of continuity on the input measures α and β is enough to ensure that if (f, g) is an optimal dual pair for $\mathrm{OT}(\alpha, \beta)$, the **Monge map**:

$$T \,:\, x \in \mathbb{R}^\mathrm{D} \,\mapsto\, x - \nabla f(x) \,=\, \nabla\big[\,\tfrac{1}{2}\|\cdot\|^2 - f\,\big](x) \tag{3.175}$$

coincides with the optimal transport plan π of α onto β. More precisely: the support of $\pi \in \mathcal{M}^+(\mathcal{X}\times\mathcal{X})$ is concentrated on pairs $(x, T(x)) \in \mathcal{X}\times\mathcal{X}$ and β is equal to the pushforward measure $\alpha \circ T^{-1}$ of α under the action of T. Remarkably, the continuous potential:

$$\varphi \,:\, x \in \mathbb{R}^\mathrm{D} \,\mapsto\, \tfrac{1}{2}\|x\|^2 \,-\, f(x) \tag{3.176}$$

is necessarily **convex**, which allows us to ensure that:

$$\forall\, x, x' \in \mathbb{R}^\mathrm{D},\ \langle\, x' - x,\, T(x') - T(x)\,\rangle_{\mathbb{R}^\mathrm{D}} = \langle\, x' - x,\, \nabla\varphi(x') - \nabla\varphi(x)\,\rangle_{\mathbb{R}^\mathrm{D}} \geqslant 0\,. \tag{3.177}$$

This property of quadratic OT is a **strong argument** in favor of the theory. It generalizes to high-dimensional settings the monotonicity of the optimal re-ordering σ of Eq. (3.158):

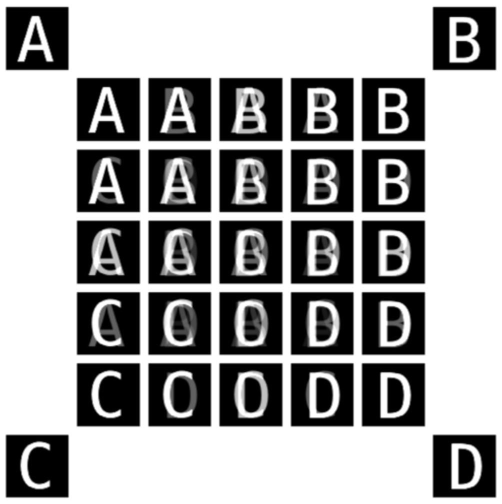
$$\forall\, i, i' \in [\![1,\mathrm{N}]\!]\,,\ \ (x_{i'} - x_i)\cdot(y_{\sigma(i')} - y_{\sigma(i)}) \,\geqslant\, 0\,. \tag{3.178}$$

The Brenier map. We should keep in mind that formally, the results above only hold when α is continuous, i.e. has density with respect to the Lebesgue measure $\lambda_{\mathbb{R}^\mathrm{D}}$. As illustrated Figure 3.13.d, transport between weighted point clouds may involve a *splitting* of mass that can not be represented using a deterministic Monge map.

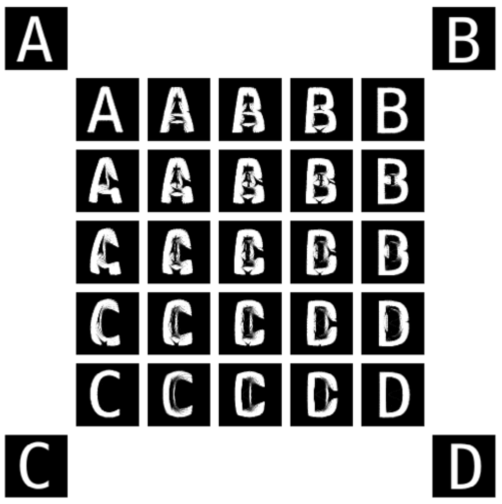
In applications, however, most practitioners are satisfied with the *barycentric mapping*:

$$T \,:\, x_i \in \mathbb{R}^\mathrm{D} \,\mapsto\, x_i - \nabla f(x_i) \,=\, x_i - \tfrac{1}{\alpha_i}\nabla_{x_i}\mathrm{OT}(\alpha,\beta) \,=\, \frac{\sum_{j=1}^{\mathrm{M}} \pi_{i,j} y_j}{\sum_{j=1}^{\mathrm{M}} \pi_{i,j}} \tag{3.179}$$

that matches each source sample $\alpha_i\delta_{x_i}$ of α with a weighted barycenter of its targets $\beta_j\delta_{y_j}$ in β. In subsequent pages of this manuscript, we *generalize* this construction to regularized variants of OT and give it a central role in applied algorithms. To avoid any confusion, we propose to call **Brenier maps** the generalized variants $T : x \mapsto x - \nabla f(x)$ of the classical Monge map, which may **not exactly send** α **onto** β but always provide good approximations of the OT matching under a given time or complexity budget.



(a) Kernel, linear barycenters.



(b) Wasserstein barycenters.

Figure 3.14: Optimal Transport induces a particle-based geometry on the space of measures. In both figures, four reference probability measures A, B, C and D are represented using density maps on the unit square $\mathcal{X} = [0,1] \times [0,1]$ – black is zero and denotes the absence of mass. The 25 interpolating density maps represent the **Fréchet barycenters** $\gamma_{s,t} = \arg\min_{\gamma \in \mathcal{M}_1^+(\mathcal{X})} (1-s) \cdot (1-t) \cdot \text{Loss}(A,\gamma) + (1-s) \cdot t \cdot \text{Loss}(B,\gamma) + s \cdot (1-t) \cdot \text{Loss}(C,\gamma) + s \cdot t \cdot \text{Loss}(D,\gamma)$ for values of s and t sampled in $\{$`0., .25, .50, .75, 1.00`$\}$.
(a) The geometry induced by a Kernel$_k$ loss function acts purely on **weights**. As discussed Eq. (3.151), $\gamma_{s,t}$ is simply given through a bi-linear interpolation of density maps.
(b) On the other hand, when Loss is the Wasserstein-2 OT formula of Eq. (3.168), $\gamma_{s,t}$ interpolates between measures using a purely geometric, **particle-based** method (Agueh and Carlier, 2011). The difference is most striking in the $A \leftrightarrow C$ geodesic interpolation. Note that using approximations described Section 4.3.3, this image was generated in less than a second on a free Google Colab session.

Choosing a cost function: beyond quadratic OT. Throughout this introduction, we kept a focus on the quadratic cost function $\mathbf{C}(x,y) = \frac{1}{2}\|x-y\|^2$, which induces the **Wasserstein-2 distance** on the space of probability measures. Endowed with the rich structure discussed around Eqs. (3.175-3.179), this specific metric is the one that most faithfully generalizes the sorting operation to general feature spaces. As discussed around Eq. (3.85), it enables the study of **Wasserstein gradient flows** and is most suited to **geometric** applications. Understanding its (algorithmic) properties in low-dimensional feature spaces is our main priority.

In many applied fields, however, the cost function **C** is **arbitrary** and given as a generic matrix $(\mathbf{C}_{i,j}) \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$. This was the setting originally studied by Kantorovitch, whose cost matrix typically encoded the structure of the Soviet railway system. Since World War 2, solving *exactly* the discrete OT problem in this *un-structured* setting has been a major problem in **operations research** and **network optimization**. As discussed at the beginning of Section 3.3, the literature dedicated to solving the **linear assignment problem** in this generic setting has mostly revolved around **combinatorial** algorithms. These are typically sub-optimal when the cost function presents a structure that can be leveraged, or when practitioners are only interested in getting a rough approximate matching. Nevertheless, they still provide strong baselines and essential concepts for the study of geometric OT problems.

Wasserstein-p, earth mover's distance. In between the fully *geometric* and *generic* extremes, the family of **Wasserstein-p** problems is associated to the power cost functions:

$$\mathrm{C}(x,y) \;=\; \frac{1}{|p|}\,\|x-y\|^p\,. \tag{3.180}$$

The case of the Coulomb penalty "$p=-1$" has connections with density functional theory and has become a topic of interest in quantum chemistry (Gori-Giorgi et al., 2009; Buttazzo et al., 2012; Cotar et al., 2013). Meanwhile, concave costs such as the square-root penalty "$p=1/2$" are relevant in economics, but notoriously hard to handle (Delon et al., 2012).

Crucially, the simple distance penalty obtained for "$p=1$" is the one that was originally studied by Monge to formulate his **earth mover's problem**. The associated theory is just as significant as the one developed in the quadratic case "$p=2$", with a rich history and deep connections to many applied fields. A fundamental remark is that when $\mathrm{C}(x,y)=\|x-y\|$, the Kantorovitch dual formulation of Eq. (3.170) is equivalent to a simplified problem where "$f=-g$" : the OT loss becomes the dual norm associated to the set of 1-Lipschitz test functions:

$$\mathrm{OT}(\alpha,\beta) \;=\; \sup_{f\text{ is 1-Lipschiz}} \;\langle\,\alpha-\beta,\,f\,\rangle \;\overset{\text{def.}}{=}\; \sup_{|f(x)-f(y)|\leqslant\|x-y\|} \;\langle\,\alpha-\beta,\,f\,\rangle\,. \tag{3.181}$$

In recent years, this alternative definition of the Wasserstein-1 distance has inspired the creation of a wide family of dual – *adversarial* – Loss functions, loosely related to OT theory and computed through gradient ascent on a family of parametric test functions – *discriminative newtorks*. We discuss the so-called *Wasserstein GANs* at the end of this chapter, but wish to stress that the results presented in these pages only hold with respect to the *genuine* OT problem. Generalizing them to the weakly structured settings encountered in machine learning and imaging is still very much an open question.

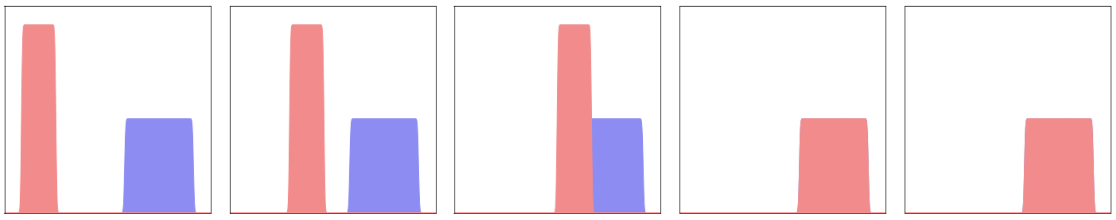
Finally, as detailed in (Santambrogio, 2015, Section 4.2) an equivalent definition of the Wasserstein-1 distance is given by Beckmann's minimal flow problem (Koopmans and Beckmann, 1957):

$$\mathrm{OT}(\alpha,\beta) \;=\; \min_{\psi:\mathbb{R}^\mathrm{D}\to\mathbb{R}^\mathrm{D}} \int_{x\in\mathbb{R}^\mathrm{D}} \|\psi(x)\|\,\mathrm{d}x \qquad \text{s.t.} \qquad \operatorname{div}\psi \;\overset{\text{def.}}{=}\; \nabla\cdot\psi \;=\; \alpha-\beta\,, \tag{3.182}$$

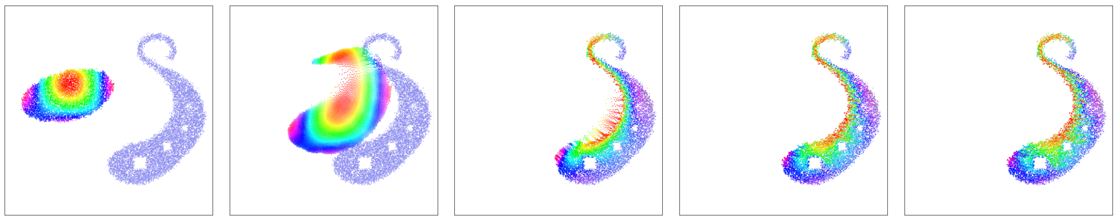
which is of considerable interest in economics and can be conveniently generalized to graphs.

Geometric properties. As illustrated in Figures 3.15 and 3.16, the choice of an appropriate cost function C has a sensible influence on the dynamics induced by the $\mathrm{OT}(\alpha,\beta)$ loss on measure-fitting pipelines. In the remainder of this manuscript, **we mostly focus on the Wasserstein-2 distance** which provides the most "geometric" gradients and is thus **best suited to computational anatomy**. Nevertheless, we keep discussing the major trade-offs associated to other cost functions and hope that the advances presented in the next few pages will prove useful in a wide range of settings.

Conclusion. OT theory is an essential tool for geometric data processing, which allows us to retrieve the intuitive dynamics of Figure 3.16 with a simple mathematical formula. In practice though, the resolution of generic linear programs along the lines of Eqs. (3.169-3.170) is known to be a hard combinatorial problem. The quest for **scalable** and **principled** algorithms that tackle this challenge in structured settings – e.g. when $\mathrm{C}(x,y)=\frac{1}{2}\|x-y\|^2$ or $\|x-y\|$ – is thus a topic of utmost interest in data sciences.

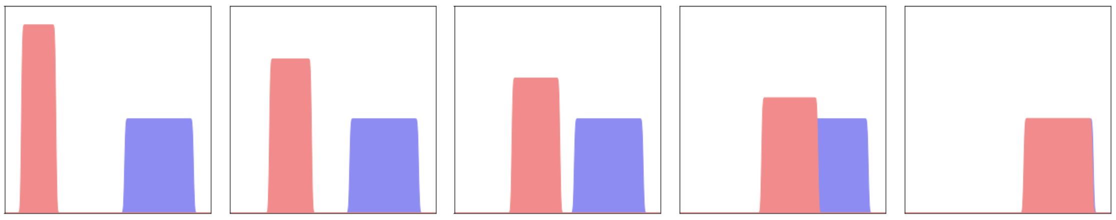


(a) $t = 0$ (b) $t = .25$ (c) $t = .50$ (d) $t = 1.00$ (e) $t = 5.00$

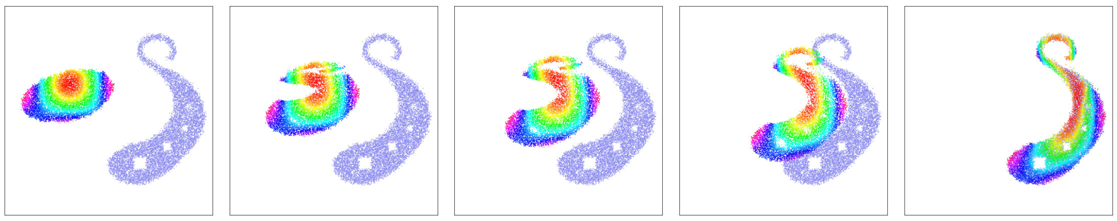


(f) $t = 0$ (g) $t = .25$ (h) $t = .50$ (i) $t = 1.00$ (j) $t = 5.00$

Figure 3.15: Gradient flow of the Wasserstein-1 loss in the unit interval (a-e) and the unit square (f-j). Here, the setting is the same as that of Figure 3.7 and $\mathbf{C}(x, y) = \|x - y\|$. Since the gradient of $\mathbf{C}$ is given by the unit-norm vector $\nabla_x \mathbf{C}(x, y) = (x - y)/\|x - y\|$ whenever $x \neq y$, source particles $\alpha_i \delta_{x_i}$ generally travel at unit speed until reaching a full covering of the target β.



(a) $t = 0$ (b) $t = .25$ (c) $t = .50$ (d) $t = 1.00$ (e) $t = 5.00$



(f) $t = 0$ (g) $t = .25$ (h) $t = .50$ (i) $t = 1.00$ (j) $t = 5.00$

Figure 3.16: Gradient flow of the Wasserstein-2 loss in the unit interval (a-e) and the unit square (f-j). Here, the setting is the same as that of Figure 3.7 and $\mathbf{C}(x, y) = \frac{1}{2}\|x - y\|^2$. A standard theorem on the structure of Wasserstein-2 geodesics ensures that each source particle $\alpha_i \delta_{x_i}$ goes in straight line towards its target $T(x_i)$, which stays constant troughout the whole evolution. The gradient of the quadratic cost $\mathbf{C}$ is given by the simple formula $\nabla_x \mathbf{C}(x, y) = (x - y)$, which vanishes as x gets close to y. We thus observe an *asymptotic* convergence in e^{-t} of the model α towards the target β, as we recover a dynamics that is comparable to that of the linear ODE "$f'(t) = -f(t)$". In applications, practitioners may reach the final matching at "$t = +\infty$" with a **single gradient descent step** by using a large learning rate $\delta t = 1$, instead of the small "infinitesimal" value of $\delta t = .01$ showcased since Figure 3.7.

3.3 Sinkhorn entropies and divergences

In dimension $\mathrm{D} = 1$, the Optimal Transport – *sorting* – problem can be solved using efficient and robust methods. Relying on coarse-to-fine – *divide and conquer* – strategies, Quicksort-like algorithms position iterative *median pivots* and find optimal permutations with a generic $O(\mathrm{N} \log \mathrm{N})$ complexity. Can we emulate these performances in higher-dimensional settings?

How should we solve the Monge-Kantorovitch problem? At first glance, scaling up the resolution of the OT problem of Eq. (3.166) is a daunting task: the transport plan $\pi \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ that we strive to optimize can hardly fit in memory as soon as M and N exceed `10-50,000`. Fortunately though, as discussed page 98, the OT problem can always be reduced to a tractable *dual* search for optimal potentials that have the same memory footprint as the input data.

Since World War 2, numerous methods have been proposed to find optimal dual pairs (f, g) solution of Eqs. (3.169-3.170) that maximize the linear form:

$$(f,g) \;\in\; \mathcal{C}(\mathcal{X}) \times \mathcal{C}(\mathcal{X}) \;\mapsto\; \langle \alpha,\, f\rangle \;+\; \langle \beta,\, g\rangle \;\in\; \mathbb{R} \tag{3.183}$$

under the constraint that $f \oplus g \leqslant \mathbf{C}$. Usually developed for applications to economics and operations research, standard solvers find *exact* solutions of the linear assignment problem in *cubic* or at least *sup-quadratic* time: we refer to the introduction of Bernhard Schmitzer's PhD thesis (Schmitzer, 2014) or to (Peyré and Cuturi, 2017, Chapter 3) for an overview.

Alternate maximization on the dual variables. Unfortunately, these classical linear programming solvers are *combinatorial algorithms* which do not stream well on GPUs. As we look for methods that can leverage the parallel computing power of modern hardware, a tempting solution is to iterate the *optimality equations* of Eqs. (3.173-3.174) until reaching convergence. That is, to saturate alternatively the separable constraint:

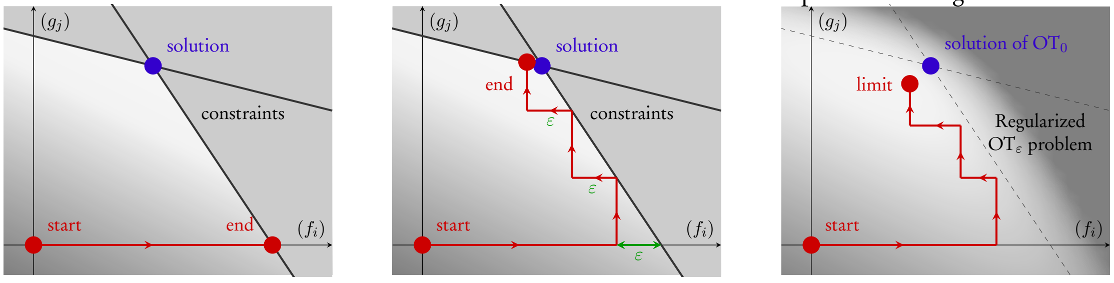
$$\forall\, i \in [\![1, \mathrm{N}]\!]\,,\; \forall\, j \in [\![1, \mathrm{M}]\!]\,,\;\; f_i \,+\, g_j \;\leqslant\; \mathbf{C}(x_i, y_j) \tag{3.184}$$

with respect to the variables $(f_i) \in \mathbb{R}^{\mathrm{N}}$ and $(g_j) \in \mathbb{R}^{\mathrm{M}}$, until no more progress can be made.

Algorithm 3.1: Naive greedy algorithm

Parameter: Cost function $\mathbf{C} : (x_i, y_j) \in \mathcal{X} \times \mathcal{X} \mapsto \mathbf{C}(x_i, y_j) \in \mathbb{R}$.
Input: Positive measures $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$ with the same mass.

```
1: $f_i,\ g_j \;\leftarrow\; \mathbf{0}_{\mathbb{R}^{\mathrm{N}}},\ \mathbf{0}_{\mathbb{R}^{\mathrm{M}}}$        ▷ Default initialization for vectors of size N and M.
2: repeat
3:     $f_i \;\leftarrow\; \min_{j=1}^{\mathrm{M}} \left[\, \mathbf{C}(x_i, y_j) \;-\; g_j \,\right]$        ▷ Parallel updates over $i \in [\![1, \mathrm{N}]\!]$.
4:     $g_j \;\leftarrow\; \min_{i=1}^{\mathrm{N}} \left[\, \mathbf{C}(x_i, y_j) \;-\; f_i \,\right]$        ▷ Parallel updates over $j \in [\![1, \mathrm{M}]\!]$.
5: until convergence.
6: return $f_i,\ g_j$        ▷ Pseudo-optimal dual vectors.
```



(a) Greedy, Alg. 3.1. (b) Auction, Alg. 3.2. (c) Sinkhorn, Alg. 3.3.

Figure 3.17: The auction (b) and Sinkhorn (c) algorithms alleviate the main drawbacks of a greedy coordinate ascent scheme on the dual OT problem (a): they converge to sensible ε-approximations of the optimal transport cost $\mathrm{OT}(\alpha, \beta)$ with an $O(\mathrm{NM} \max_{\alpha\otimes\beta} \mathbf{C}/\varepsilon)$ complexity.

A good-bad idea. Most appealingly, this algorithm can be parallelized using the `KeOps` routines and streams well on modern hardware. Since α and β are positive measures, maximizing f and g alternatively leads to a non-decreasing sequence of dual objective values… But unfortunately, in general, **these greedy iterations do not converge towards a solution of the transport problem**: they get stuck after no more than two updates in configurations that satisfy both Eqs. (3.173-3.174) but are *not* optimal solutions of the dual OT problem.

This phenomenon is studied in detail in (Santambrogio, 2015, Section 1.6) (**C**-cylical monotonicity) and is a consequence of the non-differentiability of the dual OT problem. As we optimize f and g aggressively, we converge towards "stalemate" configurations on *faces* of the set of admissible dual pairs which do not necessarily maximize the linear form $(f, g) \mapsto \langle\alpha,\, f\rangle + \langle\beta,\, g\rangle$ on the domain, as illustrated Fig. 3.17.

The auction algorithm. From times to times, mathematics provide valuable life lessons. In Figure 3.13, we showed that under a "competitive market" assumption, internal planning and external out-sourcing compete around a common equilibrium. Now, we've seen that **greedy updates** generally prevent us from reaching a global optimum, even for the seemingly trivial and monotonous maximization of a linear functional over a convex polytope.

To retrieve a well-behaved algorithm, we should **relax the updates** on the prices f_i and g_j. Most pragmatically, the theory of *auctions* proposes to introduce a positive **temperature** or *margin* $\varepsilon > 0$ and iterate "sub-optimal" updates along the lines of:

Algorithm 3.2: Pseudo-auction algorithm

Parameters: Cost function $\mathbf{C} : (x_i, y_j) \in \mathcal{X} \times \mathcal{X} \mapsto \mathbf{C}(x_i, y_j) \in \mathbb{R}$,
Temperature $\varepsilon > 0$.
Input: Positive measures $\alpha = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \delta_{x_i}$ and $\beta = \frac{1}{\mathrm{M}} \sum_{j=1}^{\mathrm{M}} \delta_{y_j}$ with $\mathrm{N} = \mathrm{M}$.

```
1: $f_i,\ g_j \leftarrow \mathbf{0}_{\mathbb{R}^\mathrm{N}},\ \mathbf{0}_{\mathbb{R}^\mathrm{M}}$          ▷ Default initialization for vectors of size N and M.
2: repeat                                                              ▷ Auction rounds.
3:     $f_i \leftarrow \min_{j=1}^{\mathrm{M}} [\,\mathbf{C}(x_i, y_j) - g_j\,] - \varepsilon$       ▷ Bidding phase over $i \in [\![1, \mathrm{N}]\!]$ .
4:     $g_j \leftarrow \min_{i=1}^{\mathrm{N}} [\,\mathbf{C}(x_i, y_j) - f_i\,]$          ▷ Assignment phase over $j \in [\![1, \mathrm{M}]\!]$ .
5: until $\forall\, i,\ \exists\, j,\ f_i + g_j \geqslant \mathbf{C}(x_i, y_j) - \varepsilon$ .     ▷ $\varepsilon$-complementary slackness.
6: return $f_i,\ g_j$                                                ▷ Pseudo-optimal dual vectors.
```

Properties. First introduced in (Bertsekas, 1979), the standard auction algorithm is traditionally formulated in the *linear assignment* setting: $\alpha_i = 1/\mathrm{N} = 1/\mathrm{M} = \beta_j$. It is meant to optimize a labelling $\sigma : [\![1, \mathrm{N}]\!] \rightarrow [\![1, \mathrm{M}]\!]$, using a single vector of dual prices f_i and asynchronous updates that are slightly more aggressive than those of Algorithm 3.2. Throughout the 80's, a very strong school gathered around Dimitri Bertsekas and extended this method to a wide range of settings: network flows, fully asynchronous iterations, integer weights, etc. We recommend (Bertsekas, 1992, 2009) and their references for an in-depth introduction.

As far as geometric applications are concerned, the main take-away from this remarkable line of work is that if $\mathbf{C} \geqslant 0$, dual pairs which get ε-close to the optimal value can be computed in no more than $(\max_{\alpha\otimes\beta} \mathbf{C}/\varepsilon)$ parallel iterations, with $O(\mathrm{NM})$ complexity. Going further, as discussed Section 3.3.3, annealing or *ε-scaling* strategies can be used to accelerate convergence: the number of iterations is reduced to an $O(\log(\max_{\alpha\otimes\beta} \mathbf{C}/\varepsilon))$, both in theory and in practice.

The Sinkhorn algorithm. Auction iterations are ideally suited to large-scale geometric applications. To emulate their behaviour with *smooth* updates that are **easier to study and extend** from a mathematical perspective, a simple idea is to rely on SoftMax and SoftMin reductions. For any continuous expression φ and temperature $\varepsilon > 0$, these operators are defined through:

$$\max_{\substack{\varepsilon \\ \alpha}} \varphi \;=\; \max_{\substack{\varepsilon \\ x\sim\alpha}} \big[\,\varphi(x)\,\big] \;\overset{\text{def.}}{=}\; +\varepsilon\,\log \int e^{+\varphi(x)/\varepsilon}\,\mathrm{d}\alpha(x) \;=\; +\varepsilon\,\log\langle \alpha,\, \exp(+\varphi/\varepsilon)\rangle\;, \quad (3.185)$$

$$\min_{\substack{\varepsilon \\ \alpha}} \varphi \;=\; \min_{\substack{\varepsilon \\ x\sim\alpha}} \big[\,\varphi(x)\,\big] \;\overset{\text{def.}}{=}\; -\varepsilon\,\log \int e^{-\varphi(x)/\varepsilon}\,\mathrm{d}\alpha(x) \;=\; -\varepsilon\,\log\langle \alpha,\, \exp(-\varphi/\varepsilon)\rangle\;, \quad (3.186)$$

and can be implemented efficiently using the online `KeOps` log-sum-exp scheme of page 50. As discussed Section A.3.2, log-sum-exp formulas allow us to interpolate between a sum with respect to α (when $\varepsilon \rightarrow +\infty$) and a maximum or a minimum on its support (when $\varepsilon \rightarrow 0$).

To get a smooth approximation of the optimal dual potentials f and g at a low computational cost, a sensible strategy is therefore to replace the "minima" of Algorithm 3.1 with SoftMin reductions as we iterate the not-so-greedy "**Sinkhorn**" updates:

Algorithm 3.3: Sinkhorn or "soft-auction" algorithm, in the log-domain

Parameters: Cost function $\mathbf{C} : (x_i, y_j) \in \mathcal{X} \times \mathcal{X} \mapsto \mathbf{C}(x_i, y_j) \in \mathbb{R}$,
Temperature $\varepsilon > 0$.
Input: Positive measures $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$ with the same mass.

```
1: $f_i,\ g_j \leftarrow \mathbf{0}_{\mathbb{R}^\mathrm{N}},\ \mathbf{0}_{\mathbb{R}^\mathrm{M}}$          ▷ Default initialization for vectors of size N and M.
2: repeat
3:     $f_i \leftarrow -\,\varepsilon\,\log \sum_{j=1}^{\mathrm{M}} \beta_j\,\exp \tfrac{1}{\varepsilon}[\,g_j - \mathbf{C}(x_i, y_j)\,]$    ▷ Parallel updates over $i \in [\![1, \mathrm{N}]\!]$.
4:     $g_j \leftarrow -\,\varepsilon\,\log \sum_{i=1}^{\mathrm{N}} \alpha_i\,\exp \tfrac{1}{\varepsilon}[\,f_i - \mathbf{C}(x_i, y_j)\,]$    ▷ Parallel updates over $j \in [\![1, \mathrm{M}]\!]$.
5: until convergence up to a set tolerance.          ▷ Monitor the updates of $f$ and $g$.
6: return $f_i,\ g_j$                                  ▷ Pseudo-optimal dual vectors.
```

3.3.1 Entropic regularization of the Kantorovich problem

The Schrödinger problem. Surprisingly, this simple algorithm **works well** in practice: the iterates converge linearly towards an ε-approximation of the optimal dual potentials f and g, without ever getting stuck in bad "local minima". To understand this, we introduce a strictly convex regularization of the primal OT problem of Eq. (3.170) which reads:

$$\begin{aligned}\mathrm{OT}_\varepsilon(\alpha,\beta) \overset{\text{def.}}{=} \min_{\pi\in\mathcal{M}^+(\mathcal{X}\times\mathcal{X})} \langle\,\pi,\,\mathbf{C}\,\rangle \;+\; \varepsilon\,\mathrm{KL}(\pi,\alpha\otimes\beta) \qquad (3.187)\\ \text{subject to} \qquad \pi\geqslant 0, \qquad \pi_1=\alpha, \qquad \pi_2=\beta\,,\end{aligned}$$

for positive values of the temperature $\varepsilon > 0$. The **entropic penalty**:

$$\mathrm{KL}(\pi,\alpha\otimes\beta) \;=\; \Big\langle\,\pi\,,\,\log\tfrac{\mathrm{d}\pi}{\mathrm{d}\alpha\otimes\beta}\,\Big\rangle \;-\;\langle\,\pi,\,1\,\rangle \;+\;\langle\,\alpha\otimes\beta,\,1\,\rangle\,, \qquad (3.188)$$

or Kullback-Leibler divergence, has already been discussed in detail Section 3.2.1. This **entropic OT_ε problem** was first introduced by Erwin Schrödinger in (Schrödinger, 1932, Section VII, page 302) in a memoir that discussed the physical interpretations of quantum mechanics. It models a Brownian bridge problem between two probability measures $\omega_0=\alpha$ and $\omega_1=\beta$, with an important legacy in probability theory and fluid mechanics (Léonard, 2012; Baradat, 2019).

Remarkably, the entropic *barrier* ($x\mapsto x\log(x/y)-x+y$) has a singular derivative of $-\infty$ at $x=0$, which prevents this value from being attractive. For $\varepsilon>0$, this ensures that the unique solution π of $\mathrm{OT}_\varepsilon(\alpha,\beta)$ has **full support** with respect to the product measure $\alpha\otimes\beta$ and belongs to the *interior* of the simplex defined by the positivity and marginal constraints – see Figure 3.18. Unlike the "genuine" $\mathrm{OT}=\mathrm{OT}_0$ linear program, which collapses into a combinatorial problem on the boundary of the simplex of admissible transport plans, the strictly convex OT_ε problem can thus be solved using *calculus* and vanishing derivatives.

Dual formulations. Thanks to the Fenchel-Rockafellar theorem, which generalizes Kantorovitch's duality from linear to convex programs, we can rewrite the primal problem above as the dual maximization of an **un-constrained** dual objective:

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \max_{f,g\in\mathcal{C}(\mathcal{X})} \quad \langle\alpha,\,f\rangle \;+\;\langle\beta,\,g\rangle \;-\;\varepsilon\,\langle\alpha\otimes\beta,\,\exp\tfrac{1}{\varepsilon}[f\oplus g-\mathbf{C}]\;-\;1\rangle\,. \qquad (3.189)$$

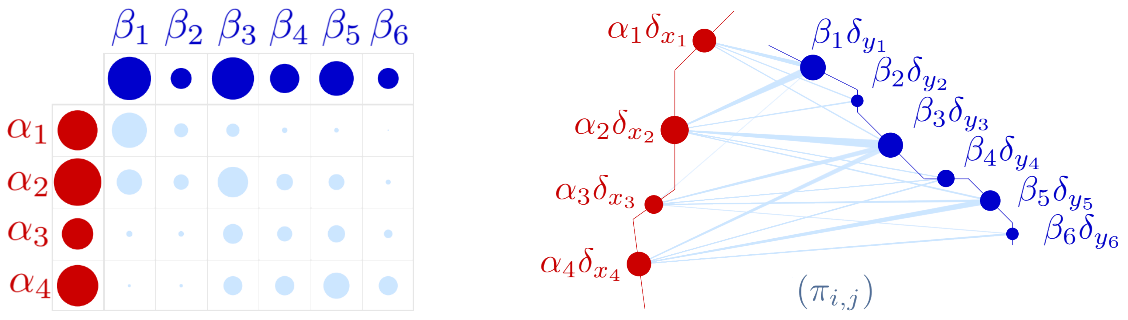
Alternatively, some authors may prefer the Lipschitz formulation of (Genevay et al., 2016):

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \max_{f,g\in\mathcal{C}(\mathcal{X})} \quad \langle\alpha,\,f\rangle \;+\;\langle\beta,\,g\rangle \;-\;\underbrace{\varepsilon\,\log\langle\alpha\otimes\beta,\,\exp\tfrac{1}{\varepsilon}[f\oplus g-\mathbf{C}]\rangle}_{\max_{\varepsilon,\,\alpha\otimes\beta}\,[f\oplus g-\mathbf{C}]}\,. \qquad (3.190)$$

As for us, we generally favour the "elementary" dual problem:

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \max_{f,g\in\mathcal{C}(\mathcal{X})} \quad \langle\alpha,\,f\rangle \;+\;\langle\beta,\,g\rangle \qquad \text{s.t.} \qquad \max_{\varepsilon,\,\alpha\otimes\beta}\,[f\oplus g-\mathbf{C}]\;\leqslant\;0\,, \qquad (3.191)$$

which highlights the similarity with the traditional "FedEx" problem of Eq. (3.170). Anyway: these three dual formulations are all equivalent to the primal OT_ε problem of Eq. (3.187). As illustrated Figure 3.17.c, they can be understood as concave regularizations of the classical dual problem: the hard constraint "$f\oplus g\leqslant\mathbf{C}$" is simply replaced by various soft penalties.



(a) As an N-by-M positive matrix. (b) As a collection of "springs".

Figure 3.18: The entropic transport plan π, solution of the regularized OT_ε problem, has full support with respect to the product measure $\alpha \otimes \beta$. If $\mathbf{C}(x, y) = \frac{1}{p}\|x - y\|^p$, π sends points $\alpha_i\delta_{x_i}$ onto **fuzzy** collections of targets $\beta_j\delta_{y_j}$ whose diameters are roughly proportional to the **blurring scale** $\sigma = \varepsilon^{1/p}$.

A compact encoding of the optimal transport plan. Strong duality is satisfied by the OT_ε problem. The regularized optimal transport plan $\pi \in \mathcal{M}^+(\mathcal{X} \times \mathcal{X})$ and dual potentials $(f, g) \in \mathcal{C}(\mathcal{X})^2$ are linked through the constitutive equation:

$$\pi \;=\; \exp \tfrac{1}{\varepsilon}\big[\, f \oplus g - \mathbf{C} \,\big] \cdot (\alpha \otimes \beta)\,, \tag{3.192}$$

$$\text{i.e.} \quad \pi_{i,j} \;=\; \exp \tfrac{1}{\varepsilon}\big[\, f_i + g_j - \mathbf{C}(x_i, y_j) \,\big] \cdot \alpha_i\beta_j\,, \tag{3.193}$$

which ensures that a significant amount of mass $\pi_{i,j}$ is transported from a point $\alpha_i\delta_{x_i}$ onto a target $\beta_j\delta_{y_j}$ if and only if the constraint "$f_i + g_j \leqslant \mathbf{C}(x_i, y_j)$" is ε-close to being saturated.

Sinkhorn = coordinate ascent on the dual problem. For the three dual objectives $\mathcal{D}_{\alpha,\beta}(f, g)$ of Eqs. (3.189-3.191), simple computations show that for all pair (f, g) of dual potentials:

$$\partial_f \mathcal{D}_{\alpha,\beta}(f, g) \;=\; 0 \iff f(x) \;=\; \min_{\substack{\varepsilon\\ y\sim\beta}} \big[\,\mathbf{C}(x, y) - g(y)\,\big] \qquad \alpha(x)\text{-a.e.} \tag{3.194}$$

$$\text{and} \quad \partial_g \mathcal{D}_{\alpha,\beta}(f, g) \;=\; 0 \iff g(y) \;=\; \min_{\substack{\varepsilon\\ x\sim\alpha}} \big[\,\mathbf{C}(x, y) - f(x)\,\big] \qquad \beta(y)\text{-a.e.} \tag{3.195}$$

Up to the switch between "vector" and "measure" notations, we retrieve exactly the Sinkhorn iterations of Algorithm 3.3. Enforcing them alternatively is thus equivalent to performing a simple **coordinate ascent** on the smooth and concave dual of the Schrödinger problem $\text{OT}_\varepsilon(\alpha, \beta)$. As the latter approximates the Monge-Kantorovitch problem up to an ε-small correction, this explains the surprisingly good empirical properties of Algorithm 3.3.

Matrix scaling. Most researchers interpret the "Sinkhorn" iterations of Algorithm 3.3 through the introduction of auxiliary variables: the **Gibbs kernel** k_ε and **scaling functions** u and v, defined trough:

$$k_\varepsilon(x, y) \;=\; e^{-\mathbf{C}(x,y)/\varepsilon}\,, \qquad u(x) \;=\; e^{f(x)/\varepsilon}\,, \qquad v(y) \;=\; e^{g(y)/\varepsilon} \tag{3.196}$$

and encoded with the **kernel matrix** and **positive scaling vectors**:

$$K_{i,j} \;=\; e^{-\mathbf{C}(x_i,y_j)/\varepsilon} \in \mathbb{R}^{\text{N}\times\text{M}}_{>0}\,, \quad U_i \;=\; e^{f_i/\varepsilon} \in \mathbb{R}^{\text{N}}_{>0}\,, \quad V_j \;=\; e^{g_j/\varepsilon} \in \mathbb{R}^{\text{M}}_{>0}\,. \tag{3.197}$$

In this exponential system of coordinates, the Sinkhorn iterations read:

$$u \leftarrow \frac{1}{k \star (v\beta)} \qquad \text{and} \qquad v \leftarrow \frac{1}{k \star (u\alpha)}\,, \tag{3.198}$$

$$\text{i.e.} \qquad U_i \leftarrow \frac{1}{K(V\beta)} \qquad \text{and} \qquad V_j \leftarrow \frac{1}{K^\top(U\alpha)}\,. \tag{3.199}$$

This is equivalent to **enforcing alternatively the marginal constraints**:

$$(\pi\mathbf{1})_i = \sum_{j=1}^{\mathrm{M}} \pi_{i,j} = \alpha_i \qquad \text{and} \qquad (\pi^\top\mathbf{1})_j = \sum_{i=1}^{\mathrm{N}} \pi_{i,j} = \beta_j\,, \tag{3.200}$$

on the transport plan:

$$\pi_{i,j} = \exp\tfrac{1}{\varepsilon}\big[\, f_i + g_j - \mathrm{C}(x_i, y_j)\,\big] \cdot \alpha_i\beta_j = \alpha_i U_i K_{i,j} V_j \beta_j\,, \tag{3.201}$$

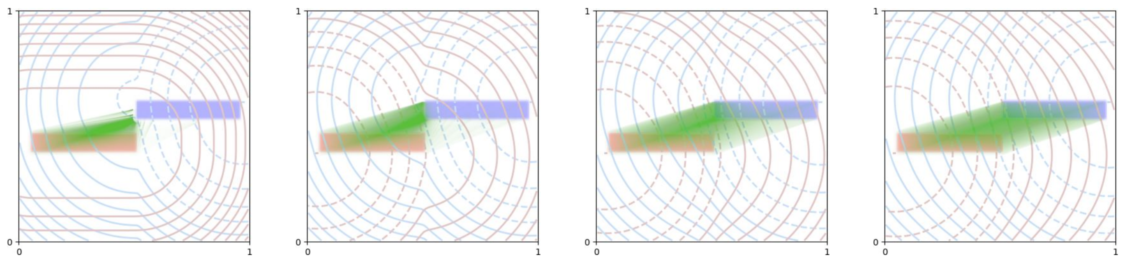
encoded implicitely by the two scaling vectors (U_i) and (V_j).

Historical perspective. Today, Algorithm 3.3 is most often named after Richard Sinkhorn who first showed the convergence of the normalizing procedure of Eq. (3.199) for arbitrary positive matrices K (Sinkhorn, 1964). Easy to implement and study, these iterations and their application to the transport problem have been periodically re-discovered since the days of Schrödinger. Let us discuss some of the most important lines of work, without any pretense of exhaustivity: we value cross-field interactions more than priority disputes over a method which predates the career of any active researcher.

Economics. Unsurprisingly, following Kantorovitch's pioneering work, some of the first major papers on entropic OT came from operations research and economics. In this literature, the Sinkhorn iterations and Schrödinger problem OT_ε were successively referred to as **entropy maximising models** (Wilson, 1969), **gravity models** (Erlander, 1980), or **matching with trade-offs** (Galichon and Salanié, 2010) using the **iterative projection fitting procedure (IPFP)** (Yule, 1912; Deming and Stephan, 1940; Fienberg et al., 1970; Ruschendorf et al., 1995).

Geometry. Independently, (Kosowsky and Yuille, 1994) introduced the Sinkhorn loop to the machine learning community as a modification of the standard auction rounds, inspired by statistical physics and neural networks: the names of **SoftAssign** or **invisible hand algorithm** are attached to this period. As discussed in the next few pages, most of the important algorithmic and theoretical points were already understood in this work, which had a considerable influence in computer vision. Marketed as **robust point matching (RPM)** in two landmark papers (Gold et al., 1998; Chui and Rangarajan, 2000), the regularized loss function OT_ε is now a standard tool for point cloud processing – see e.g. the excellent Wikipedia page on *point set registration*.

Statistics. Finally, (Cuturi, 2013) revived the **Sinkhorn** algorithm in the **GPU era**. Subsequent works in machine learning have kept a strong focus on the high-temperature setting ($\max \mathrm{C}/\varepsilon \leqslant$ 10-30) and brought an important understanding of the *statistical* properties of the OT_ε problem in high-dimensional settings (Genevay et al., 2019). Today, the reference textbook (Peyré and Cuturi, 2017) is probably the best introduction available on the subject: we strongly recommend it to newcomers looking for intuitions, insightful remarks and examples of applications.



(a) $\texttt{it} = \texttt{2}$ (b) $\texttt{it} = \texttt{10}$ (c) $\texttt{it} = \texttt{100}$ (d) $\texttt{it} = \texttt{300}$

Figure 3.19: The Sinkhorn algorithm is simple, but relatively inefficient. We display two translated probability measures α (in red) and β (in blue) using their rectangle-shaped densities on the unit square $\mathcal{X} = [0, 1] \times [0, 1]$. Here, $\mathbf{C}(x, y) = \frac{1}{2}\|x - y\|^2$ and $\sqrt{\varepsilon} = \texttt{0.01}$. In the background, the dual potentials f (in blue) and g (in red) are extended to the ambient space using the canonical Eqs. (3.194-3.195) and displayed as contour lines. After every application of Algorithm 3.3, line 3, the dual vectors $(f(x_i))$ and $(g(y_j))$ encode a transport plan π which satisfies the first marginal contraint ($\pi\mathbf{1} = \alpha$) but not the second one ($\pi^\top\mathbf{1} = \beta$) until convergence : we display it as a green, transparent "spring system".
(a) Initially, at iteration 1 of the Sinkhorn loop, the implicit transport plan π is very close to the naive nearest-neighbor matching discussed Section 3.2.2 (Hausdorff, chamfer distance). (b-c) The Sinkhorn algorithm is all about letting the "high pressure" wavefront on the nearest neighbors of α spread through the support of β. (d) At convergence, π finally maps α onto β. The whole procedure is slow, with incremental steps of size ε. As discussed Section 3.3.3, we now see that the Sinkhorn loop could benefit *massively* from coarse-to-fine multiscale strategies, which handle the translation above in one single step.

Stressing the importance of our conventions. Throughout its long history, entropic OT has been studied from diverse perspectives. Subtle variations between papers occur naturally, but are seldom made explicit... and tend to confuse newcomers to the field: keeping track of the conventions adopted by a loosely connected web of authors is by no mean easy. In the present work, we wish to stress two important variations between our conventions and those of the "standard" Sinkhorn framework adopted for instance in (Cuturi, 2013; Peyré and Cuturi, 2017):

1. As detailed in Algorithm 3.3, we always write our iterations in the **log-domain**. We directly update the dual potentials f_i and g_j instead of the (exponentiated) scaling vectors U_i and V_j. This ensures **numerical stability** for all values of the temperature ε, with little computational overhead: the `KeOps` log-sum-exp routines are just as efficient as standard matrix-vector products with a kernel matrix $K_{i,j}$.

2. Our entropic penalty is defined relatively to the **product measure** $\alpha \otimes \beta$ and does not rely on the ill-defined "entropy" $H(\pi) = \text{KL}(\pi, \text{Count}_{\text{Supp}(\alpha\otimes\beta)})$ with respect to the counting measure on the product of the measures' supports. This change is barely noticeable on the primal problem: for discrete measures, the two conventions are equivalent up to an additive constant. However, it drastically **simplifies the study of the dual problem**: potentials f and g can now be properly extended to the ambient space using Eqs. (3.194-3.195), which generalize Eqs. (3.173-3.174) and stay consistent for both *discrete* and *continuous* measures α and β.

These changes make very little difference in classical statistics, where researchers tend to work with *histograms* and discrete feature spaces. However, as we investigate *geometric* problems in continuous vector spaces, picking canonical and stable conventions is an essential first step towards **robust and efficient algorithms**, as suggested by Figure 3.19.

3.3.2 Removing the entropic bias, with theoretical guarantees

The entropic bias. When $\mathbf{C}(x, y) = \frac{1}{p}\|x - y\|^p$, the Monge-Kantorovitch problem defines a positive and definite loss function. As discussed Section 3.2.4, $\sqrt[p]{\mathrm{OT}}$ satisfies the triangle inequality and is usually called the Wasserstein-p distance. This is most convenient – but beware. These important geometric properties **do not hold** for the entropic loss $\mathrm{OT}_\varepsilon(\alpha, \beta)$, output of the Sinkhorn algorithm, as soon as $\varepsilon > 0$. Most noticeably, as illustrated Figures 3.20 and 3.21.a, minimizing an OT_ε loss is **not a sensible thing to do**. In general, if β is a given target measure, there exists a degenerate measure α such that:

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \;<\; \mathrm{OT}_\varepsilon(\beta, \beta) \;\neq\; 0\,. \tag{3.202}$$

This problem is well-documented in the computer vision literature, without any satisfying answer being provided: letting the temperature ε slowly decrease towards 0 to retrieve a genuine Wasserstein distance is do-able, but generally brittle and expensive – as illustrated Figure 3.20.

De-biased Sinkhorn divergences. As long as researchers kept understanding entropic regularization as a "dirty hack" to compute approximations of the Wasserstein distance, the entropic bias could be accepted as a necessary evil. But in recent years, a major culture shift on entropic regularization has taken place in the community: the **regularizing** influence of the entropic barrier and Gibbs kernel $k_\varepsilon = \exp(-\mathbf{C}/\varepsilon)$ on the input measures is now hailed as a *feature* of the theory, not a bug that should be brushed under the carpet. It could improve statistical robustness and prevent overfitting on the precise samples' locations, which is relevant in stochastic high-dimensional settings where **sampling noise** is a major concern (Genevay et al., 2019).

To retrieve a "good-looking" formula that vanishes when $\alpha = \beta$ even if $\varepsilon > 0$, the statistics paper (Ramdas et al., 2017) thus introduced the **de-biased Sinkhorn divergence**:

$$\mathrm{S}_\varepsilon(\alpha, \beta) \;\overset{\text{def.}}{=}\; \mathrm{OT}_\varepsilon(\alpha, \beta) \;-\; \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha, \alpha) \;-\; \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta, \beta)\;, \tag{3.203}$$

which was quickly adopted in the machine learning community for the training of generative models (Genevay et al., 2018; Salimans et al., 2018; Sanjabi et al., 2018).

The reasoning behind this "de-biased" formula is simple. When $\varepsilon \to +\infty$, the entropic penalty in the OT_ε problem of Eq. (3.187) becomes so strong that π converges towards the product measure $\alpha \otimes \beta$, with a cost value:

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \;\xrightarrow{\varepsilon \to +\infty}\; \langle\, \alpha \otimes \beta, \mathbf{C} \,\rangle \;=\; \langle\, \alpha, \mathbf{C} \star \beta \,\rangle\,. \tag{3.204}$$

Surprise: we do not recognize a squared norm... but a kernel dot product, similar to the ones discussed in Section 3.2.3 ! Subtracting the $\alpha \leftrightarrow \alpha$ and $\beta \leftrightarrow \beta$ auto-correlation terms to retrieve a quadratic-like limit thus makes a lot of sense. More specifically, both (Ramdas et al., 2017) and (Genevay et al., 2018) showed that if the probability measures α and β stay put:

$$\mathrm{S}_\varepsilon(\alpha, \beta) \;\xrightarrow{\varepsilon \to 0}\; \mathrm{OT}(\alpha, \beta) \qquad \text{(Wasserstein)} \tag{3.205}$$

$$\xrightarrow{\varepsilon \to +\infty} \langle\, \alpha, \mathbf{C} \star \beta \,\rangle - \tfrac{1}{2}\langle\, \alpha, \mathbf{C} \star \alpha \,\rangle - \tfrac{1}{2}\langle\, \beta, \mathbf{C} \star \beta \,\rangle$$

$$= \tfrac{1}{2}\langle\, \alpha - \beta, -\mathbf{C} \star (\alpha - \beta) \,\rangle \qquad \text{(kernel MMD)}\,. \tag{3.206}$$

Our contribution. The de-biased Sinkhorn divergence interpolates between two well-known loss functions: the Wasserstein and kernel distances. Can we prove that the appealing geometric properties at the limit – positivity, definiteness, convexity, etc. – also hold for S_ε, with any value of ε ? Going further, couldn't we improve upon the baseline Sinkhorn loop, whose rather poor performances are illustrated Figure 3.19 ? The remainder of this chapter provides an original answer to these questions. After fifty pages of introduction to measure theory, time has now come to present our contributions to the field, developed between 2017 and 2019.

De-biased dual potentials. First of all, we remark that the Sinkhorn divergence S_ε can be neatly expressed in terms of debiased potentials F and $G : \mathcal{X} \mapsto \mathbb{R}$. If $(f^{\beta\to\alpha}, g^{\alpha\to\beta})$ is a solution of the dual problem $\mathrm{OT}_\varepsilon(\alpha,\beta)$, it necessarily satisfies both optimality Equations (3.194-3.195). Simple computations then show that:

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \;=\; \langle\, \alpha,\, f^{\beta\to\alpha} \,\rangle \;+\; \langle\, \beta,\, g^{\alpha\to\beta} \,\rangle\,, \tag{3.207}$$

as the soft penalty vanishes at the optimum. Going further, if $(f^{\alpha\leftrightarrow\alpha}, f^{\alpha\leftrightarrow\alpha})$ and $(g^{\beta\leftrightarrow\beta}, g^{\beta\leftrightarrow\beta})$ are the unique solutions of the symmetric problems $\mathrm{OT}_\varepsilon(\alpha,\alpha)$ and $\mathrm{OT}(\beta,\beta)$ on the diagonal of the space of dual pairs $\mathcal{C}(\mathcal{X}) \times \mathcal{C}(\mathcal{X})$, we see that:

$$\tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) \;=\; \langle\, \alpha,\, f^{\alpha\leftrightarrow\alpha} \,\rangle \qquad \text{and} \qquad \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta) \;=\; \langle\, \beta,\, g^{\beta\leftrightarrow\beta} \,\rangle\,, \tag{3.208}$$

$$\text{so that} \qquad \mathrm{S}_\varepsilon(\alpha,\beta) \;=\; \langle\, \alpha,\, \underbrace{f^{\beta\to\alpha} - f^{\alpha\leftrightarrow\alpha}}_{F} \,\rangle \;+\; \langle\, \beta,\, \underbrace{g^{\alpha\to\beta} - g^{\beta\leftrightarrow\beta}}_{G} \,\rangle\,. \tag{3.209}$$

Gradient. As shown in Section A.4.2, the debiased potentials F and G correspond to the gradients of the Sinkhorn divergence S_ε with respect to the input measures α and β, in the sense of Eq. (3.56). This is consistent with the discussion of page 100 on Brenier potentials. In the discrete setting, assuming that $\alpha = \sum_i \alpha_i \delta_{x_i}$ and $\beta = \sum_j \beta_j \delta_{y_j}$ have the same mass:

$$\partial_{\alpha_i}\mathrm{S}_\varepsilon(\alpha,\beta) \;=\; F(x_i) \qquad \text{and} \qquad \partial_{x_i}\mathrm{S}_\varepsilon(\alpha,\beta) \;=\; \alpha_i\, \nabla F(x_i)\,. \tag{3.210}$$

As discussed page 124, these expressions can be computed "for free" as the output of any solver of the entropic OT problem. No backpropagation through the Sinkhorn loop is ever required.

Positivity, convexity. Crucially, as illustrated in Figures 3.21.b and 3.22, the debiased Sinkhorn divergence is **suitable for measure-fitting applications** and satisfies the geometric axioms of page 72. More precisely, the following result holds:

Theorem 3.1. *Let $\mathcal{X}$ be a compact metric space with a Lipschitz cost function $\mathrm{C}(x,y)$ that induces, for $\varepsilon > 0$, a* positive universal *kernel $k_\varepsilon(x,y) = \exp(-\mathrm{C}(x,y)/\varepsilon)$. Then, S_ε defines a symmetric, positive, definite, smooth loss function that is convex in each of its input variables. It also metrizes the convergence in law: for all probability measures α and $\beta \in \mathcal{M}_1^+(\mathcal{X})$,*

$$0 \;= \mathrm{S}_\varepsilon(\beta,\beta) \;\leqslant\; \mathrm{S}_\varepsilon(\alpha,\beta)\,, \tag{3.211}$$

$$\alpha = \beta \;\iff\; \mathrm{S}_\varepsilon(\alpha,\beta) = 0\,, \tag{3.212}$$

$$\alpha_n \rightharpoonup \alpha \;\iff\; \mathrm{S}_\varepsilon(\alpha_n,\alpha) \to 0\,. \tag{3.213}$$

Notably, these results also hold for measures with bounded support on a Euclidean space $\mathcal{X} = \mathbb{R}^\mathrm{D}$ endowed with ground cost functions $\mathrm{C}(x,y) = \|x-y\|$ or $\mathrm{C}(x,y) = \frac{1}{2}\|x-y\|^2$ – which induce exponential and Gaussian kernels respectively.

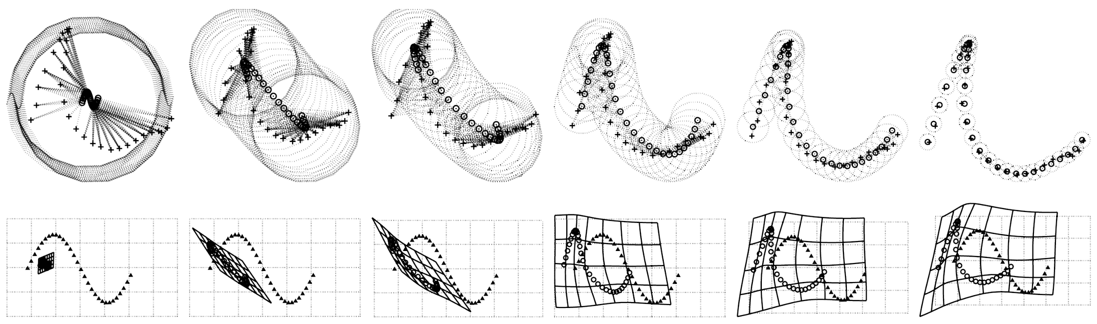
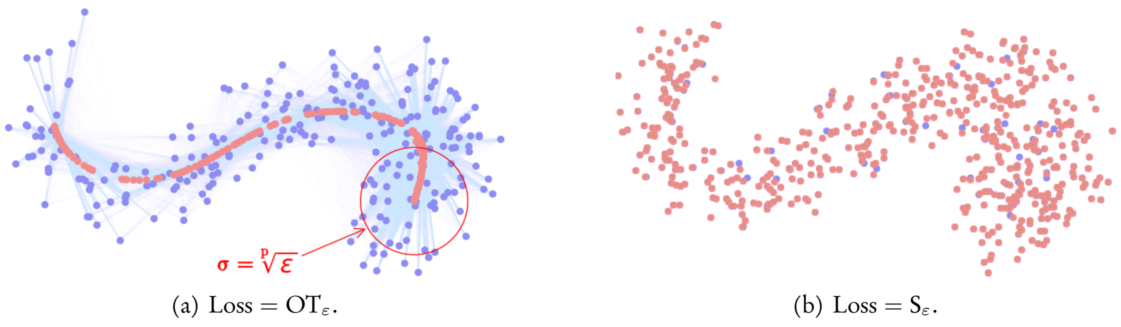


Figure 3.20: The entropic bias in the TPS-RPM paper (Chui and Rangarajan, 2000). In a classic CVPR paper, Chui and Rangarajan propose to use an entropic transport loss OT_ε – *robust point matching* – to drive a *thin plate spline* deformation model. Starting from a rest configuration (triangles and regular grid, second line), their algorithm registers a deformed point cloud (circles x_i, first and second lines) onto a target point cloud (crosses y_j, first line). This is done by alternatively computing entropic transport plans (spring systems, first line) and regularized spline deformations (deformed grid, second line), following a method detailed Section 5.1.2. The temperature $\varepsilon > 0$ is figuratively represented using circles of radius $\sigma = \varepsilon^{1/p}$, where $\mathbf{C}(x, y) = \frac{1}{p}\|x - y\|^p$.
When ε is large (left), the registration **collapses** as source points x_i converge towards local barycenters at scale σ of the target samples y_j. This phenomenon, which we propose to call **entropic bias**, is an immediate consequence of the fuzziness of the entropic transport plan displayed Figure 3.18. To mitigate its effects, Chui and Rangarajan propose to let ε decrease across iterations (left to right), as they slowly recover a solution to the genuine transport problem $\mathrm{OT} = \mathrm{OT}_0$. This solution is not very satisfying: collapsing distributions to slowly unwrap them afterwards is both unstable and cumbersome. As discussed in the remainder of this chapter, **debiased Sinkhorn divergences** provide a convenient and principled answer to this two-decade-old problem.



(a) Loss = OT_ε. (b) Loss = S_ε.

Figure 3.21: Removing the entropic bias. Solution α (in red) of the fitting problem $\min_\alpha \mathrm{Loss}(\alpha, \beta)$ for some β shown in blue. Here, $\mathbf{C}(x, y) = \|x - y\|$ on the unit square $\mathcal{X}$ in $\mathbb{R}^2$ and $\varepsilon = .1$. The positions of the red dots were optimized by gradient descent, starting from a normal Gaussian sample.
(a) The entropic bias in the standard Sinkhorn loss OT_ε promotes **mode collapse** as the source measure α converges towards a structure that looks like a medial axis of the target β at scale $\sigma = \varepsilon^{1/p} = \varepsilon$.
(b) Defined through the **debiased formulation** of Eq. (3.203), the Sinkhorn divergence S_ε is a positive, definite loss function: $\beta = \arg\min_\alpha \mathrm{S}_\varepsilon(\alpha, \beta)$. The exponential kernel $k_\varepsilon(x, y) = \exp(-\|x - y\|/\varepsilon)$ induced by the "distance" cost is **pointy**, with a heavy-tailed Fourier transform. This ensures that the associated Sinkhorn divergence is sensitive to the full spectrum of the input configuration (α, β), promoting a near-perfect overlap between the point clouds. This is desirable in many applications, but could also lead to **overfitting on the samples' locations** in real-life (noisy) settings.

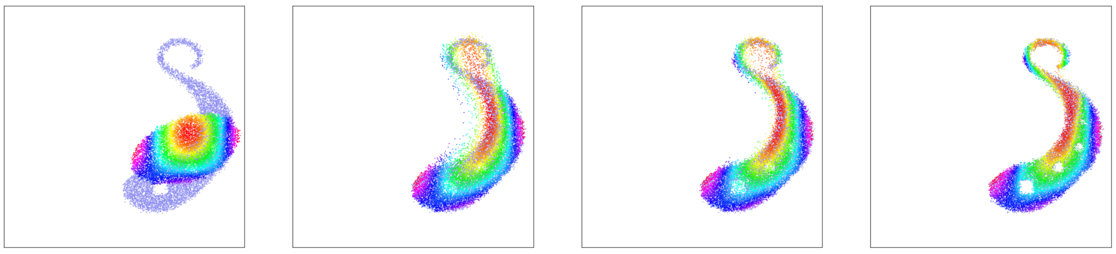
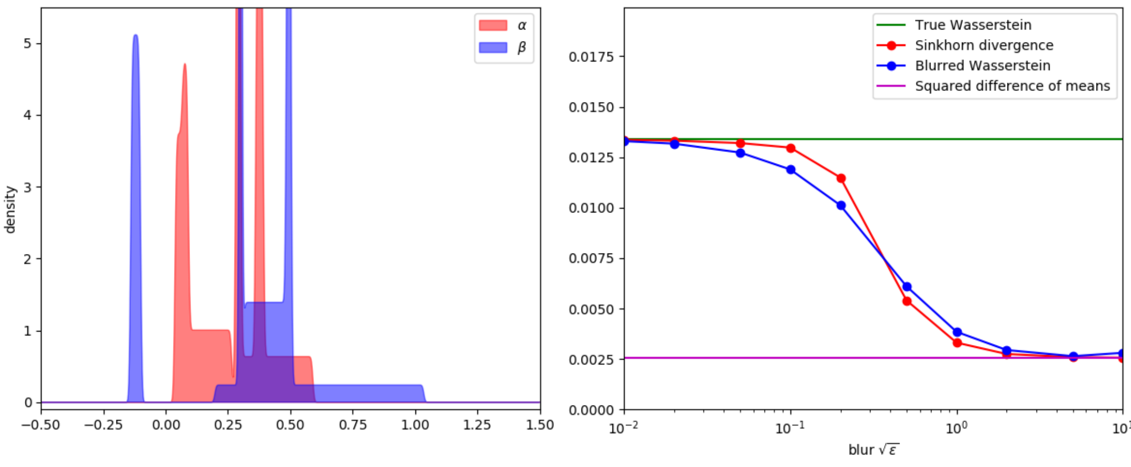

(a) $\sqrt{\varepsilon} = \texttt{1.00}$ (b) $\sqrt{\varepsilon} = \texttt{.10}$ (c) $\sqrt{\varepsilon} = \texttt{.05}$ (d) $\sqrt{\varepsilon} = \texttt{.01}$

Figure 3.22: Influence of the blurring scale $\sigma = \varepsilon^{1/p} = \sqrt{\varepsilon}$ of the Sinkhorn divergence S_ε on measure-fitting pipelines, with a quadratic cost $\mathbf{C}(x, y) = \frac{1}{2}\|x-y\|^2$ on the unit square $\mathcal{X} = [0,1] \times [0,1]$. These pictures are analogous to the last thumbnail (j) of Figure 3.16, with Sinkhorn divergences S_ε used as loss functions instead of the genuine Wasserstein-2 squared distance ($\sqrt{\varepsilon} = 0$). As we let the ratio between the configuration's diameter $\max_{i,j} \|x_i - y_j\|$ and the blurring scale $\sigma = \sqrt{\varepsilon}$ vary between one (a) and a hundred (d), we illustrate the typical behaviour of the Sinkhorn divergence as a loss function for shape registration and generative modelling.

The key operator behind entropic OT is the Gibbs kernel $k_\varepsilon = \exp(-\mathbf{C}/\varepsilon)$. In this situation, since $\mathbf{C}$ is a squared distance on $\mathcal{X}$, k_ε is a **Gaussian kernel of deviation** σ which is smooth and **blurs out the high frequencies** of the input measures – unlike the pointy exponential kernel of Figure 3.21.b.
(a) When σ is larger than the diameter of the measures' supports, S_ε behaves like the degenerate kernel MMD $\langle \alpha - \beta, -\frac{1}{2}\|\cdot\|^2 \star (\alpha - \beta)\rangle = \frac{1}{2}\|\int x \,\mathrm{d}\alpha(x) - \int y \,\mathrm{d}\beta(y)\|^2$ which only takes into account the mean values of both distributions. (b-c) Lowering the blurring scale allows S_ε to capture the finer details of the target distribution β. (d) Finally, as $\sqrt{\varepsilon}$ gets close to zero, S_ε becomes virtually indistinguishable from a genuine Wasserstein distance.
The pictures above show that in practice, in the "Wasserstein-2 setting", S_ε is **blind to all details which are smaller than the blurring scale** σ. This behaviour allows practitioners to regularize their matchings with a simple and interpretable scale parameter, a desirable behaviour in most applications.



(a) Measures α and β. (b) Values.

Figure 3.23: As discussed page 116, the Sinkhorn divergence S_ε essentially behaves like a blurred Wasserstein distance B_ε. (a) To illustrate it, we draw two random samples α and β on the real line and represent them using kernel density estimations on the interval `[-.50, 1.50]`. (b) Then, we display the values of $\mathrm{S}_\varepsilon(\alpha, \beta)$ and $\mathrm{B}_\varepsilon(\alpha, \beta)$ for varying values of ε. As predicted by the theory, both quantities interpolate between the genuine Wasserstein-2 loss $\mathrm{OT}(\alpha, \beta)$ and a degenerate kernel MMD. Remarkably, the curves stay close to each other: S_ε could be used as an affordable proxy for the intuitive blurred Wasserstein loss B_ε.

The theorem above is significant. After twenty years of research on entropic OT for point set registration, it is the first time that a loss derived from the Sinkhorn algorithm is actually **shown to be suitable for optimization and gradient descent**. Its proof, detailed Chapter A, is our main theoretical contribution to the field. The full demonstration is pretty technical, especially for the proof of convexity, and can be skipped by practitioners. Nevertheless, the simple partial proofs of Section A.1.3 already provide relevant insights on the structure of the Sinkhorn divergence S_ε: we recommend them to interested readers.

Extensions to the unbalanced setting. Remarkably, Theorem 3.1 can be generalized to the *unbalanced* theory of OT (Liero et al., 2015; Chizat et al., 2018b), explained for instance in (Peyré and Cuturi, 2017, Section 10.2). This was shown by Thibault Séjourné as part of his first year of PhD, under the supervision of François-Xavier Vialard and Gabriel Peyré: we refer to his paper for an extensive discussion (Séjourné et al., 2019).

Let us simply mention that if $\varepsilon, \rho > 0$ are two regularization parameters, we can **soften the marginal constraints** on the transport plan π and consider the unbalanced cost:

$$\mathrm{OT}_{\varepsilon,\rho}(\alpha,\beta) \overset{\text{def.}}{=} \min_{\pi \in \mathcal{M}^+(\mathcal{X}\times\mathcal{X})} \langle\, \pi,\, \mathbf{C}\,\rangle + \varepsilon\,\mathrm{KL}(\pi, \alpha\otimes\beta) + \rho\,\mathrm{KL}(\pi_1,\alpha) + \rho\,\mathrm{KL}(\pi_2,\beta)\;. \quad (3.214)$$

The problem above is well-defined **even if α and β don't have the same total mass** and coincides with the standard Schrödinger problem OT_ε of Eq. (3.187) when $\rho = +\infty$. The associated dual problem is:

$$\begin{aligned}\mathrm{OT}_{\varepsilon,\rho}(\alpha,\beta) \;=\; &\max_{f,g\in\mathcal{C}(\mathcal{X})} \rho\,\langle \alpha,\, 1 - e^{-f/\rho}\rangle \;+\; \rho\,\langle \beta,\, 1 - e^{-g/\rho}\rangle \\ &\qquad -\; \varepsilon\,\langle \alpha\otimes\beta,\, \exp\tfrac{1}{\varepsilon}[f\oplus g - \mathbf{C}] \;-\; 1\rangle\;,\end{aligned} \quad (3.215)$$

with generalized (dampened) Sinkhorn iterations that read (Chizat et al., 2018a):

$$f(x) \;\leftarrow\; \frac{\rho}{\rho+\varepsilon}\, \min_{y\sim\beta}{}_{\varepsilon} \,\big[\, \mathbf{C}(x,y) - g(y)\,\big] \quad\text{and}\quad g(y) \;\leftarrow\; \frac{\rho}{\rho+\varepsilon}\, \min_{x\sim\alpha}{}_{\varepsilon}\, \big[\, \mathbf{C}(x,y) - f(x)\,\big]\;.$$

Then, if we define the unbalanced Sinkhorn divergence:

$$\mathrm{S}_{\varepsilon,\rho}(\alpha,\beta) = \mathrm{OT}_{\varepsilon,\rho}(\alpha,\beta) - \tfrac{1}{2}\mathrm{OT}_{\varepsilon,\rho}(\alpha,\alpha) - \tfrac{1}{2}\mathrm{OT}_{\varepsilon,\rho}(\beta,\beta) + \tfrac{\varepsilon}{2}\big(\langle\alpha,\,1\rangle - \langle\beta,\,1\rangle\big)^2\;, \quad (3.216)$$

Theorem 3.1 still holds. We detail the practical applications of this result in Chapter 4.

Intuitions on low-frequency Optimal Transport. Sinkhorn divergences rely on a simple idea: by **blurring the transport plan** through the addition of an entropic penalty, we can reduce the effective dimensionality of the transportation problem and compute sensible approximations of the Wasserstein distance at a low computational cost.

As discussed in the next section, the baseline Sinkhorn loop can be symmetrized, de-biased and turned into a genuine multiscale algorithm. Available through highly efficient routines, the de-biased Sinkhorn divergence S_ε is therefore a **tractable** approximation of the Wasserstein distance that **retains its key geometric properties**: positivity, convexity, metrization of the convergence in law for measures with bounded support.

But is it really the best way of smoothing our transport problem? When $\mathbf{C}(x, y) = \frac{1}{2}\|x - y\|^2$, a sensible alternative to Sinkhorn divergences could be the **blurred Wasserstein distance**:

$$\mathrm{B}_\varepsilon(\alpha, \beta) \;=\; \mathrm{OT}(\,(k_{\varepsilon/4} \star \alpha)\,\lambda_{\mathbb{R}^\mathrm{D}},\,(k_{\varepsilon/4} \star \beta)\,\lambda_{\mathbb{R}^\mathrm{D}}\,)\;, \tag{3.217}$$

where OT denotes the **true Wasserstein distance** associated to our cost function **C**,

$$k_{\varepsilon/4}\;:\;(x-y)\;\mapsto\;\exp(-\|x-y\|^2/\tfrac{2}{4}\varepsilon) \tag{3.218}$$

is a Gaussian kernel of deviation $\sigma = \sqrt{\varepsilon}/2$ and $\lambda_{\mathbb{R}^\mathrm{D}}$ is the Lebesgue measure on $\mathcal{X} = \mathbb{R}^\mathrm{D}$. This simple divergence enjoys a collection of desirable properties:

1. It is the **square of a distance** that metrizes the convergence in law.
2. It takes geometric values on atomic Dirac masses, lifting the ground cost function to the space of positive measures:
$$\mathrm{B}_\varepsilon(\delta_x, \delta_y) \;=\; \mathrm{C}(x, y) \;=\; \tfrac{1}{2}\|x-y\|^2 \;=\; \mathrm{S}_\varepsilon(\delta_x, \delta_y)\;. \tag{3.219}$$
3. It has the same **asymptotic properties** as the Sinkhorn divergence, interpolating between the true Wasserstein distance (when $\varepsilon \to 0$) and a degenerate kernel norm (when $\varepsilon \to +\infty$) as discussed Figure 3.22.
4. Thanks to the joint convexity of the Wasserstein distance, $\mathrm{B}_\varepsilon(\alpha, \beta)$ is a **decreasing** function of ε: as we remove small-scale details, we lower the overall transport cost.

An empirical test. To compare the Sinkhorn and blurred Wasserstein divergences, a simple experiment is to display their values on pairs of 1D measures for increasing values of the temperature ε. As illustrated Figure 3.23.a, we first generate random samples (x_i) and (y_j) that respectively parameterize two discrete probability measures α and β on the unit interval. We can then compute $\mathrm{S}_\varepsilon(\alpha, \beta)$ using the multiscale Sinkhorn algorithm of Section 3.3.3, while the blurred Wasserstein loss $\mathrm{B}_\varepsilon(\alpha, \beta)$ can be quickly approximated with the **addition of a Gaussian noise** followed by a **sorting pass** – a perfect OT solver in dimension $\mathrm{D} = 1$.

Results. In practice, as showcased Figure 3.23.b, the Sinkhorn and blurred Wasserstein divergences are **nearly indistinguishable from each other**. But as far as we can tell today, these two Loss functions have very different properties:

1. B_ε is **easy to define**, compute in 1D and analyze from geometric or statistical points of views… But cannot (yet?) be computed efficiently in higher dimensions, where the true OT problem is nearly intractable.
2. S_ε is available through Sinkhorn-like algorithms but has a cumbersome, composite definition and is pretty **hard to study** rigorously: presented in Chapter A or (Genevay et al., 2019), proofs on the topic quickly get technical.

Couldn't we get the best of both worlds? Ideally, we'd like to tweak **efficient** Sinkhorn-like algorithms to compute the **natural** divergence B_ε… but as of today, this still seems out of reach. A realistic target could be to **quantify the difference** between these two loss functions, in order to legitimize the use of the Sinkhorn divergence as a **cheap proxy** for the intuitive and well-understood blurred Wasserstein distance: we leave these questions for future works.

3.3.3 Scaling up to millions of samples with a multiscale strategy

In the last two sections, we have shown that the theory of auctions could be *softened* and *debiased* to define a tractable **low-frequency approximation** of the Wasserstein distance. The so-called *Sinkhorn divergence* S_ε is positive, definite and metrizes the convergence in law. As far as practitioners are concerned, this new loss function fits perfectly within the modern gradient-based programming paradigm for shape analysis and machine learning.

Unfortunately though, implementations of the Sinkhorn algorithm found in the literature are generally *quadratic* both in time and memory. This prevents practitioners from scaling up to large points clouds and datasets, with a bottleneck around 10k samples – which isn't much, as discussed Figure 2.1. In 2020, we should be able to go further: but how?

How should we compute the dual potentials? Since World War 2, solving the Kantorovitch problem of Eqs. (3.169-3.170) has been a major problem in computational mathematics. Tractable approximations such as the Schrödinger problem OT_ε of Eqs. (3.187-3.191) have been proposed, and important works have been published in a wide variety of settings:

1. **Generic linear programming.** In the classic theory of Kantorovitch for economics and operations research, the cost matrix $(\mathbf{C}_{i,j})$ is arbitrary. OT assignment is cast as a **standard linear program**, which can be solved using the Hungarian method (Kuhn, 1955), simplex newtorks (Ahuja et al., 1988), auction iterations or more advanced combinatorial solvers (Schrijver, 2003). Unfortunately, all these algorithms have a quadratic or worse time complexity: every coefficient of the cost matrix is inspected at least once in the optimization procedure.

2. **Structured linear programming.** To bypass this limitation, advanced multiscale solvers leverage priors on the **structure of the cost matrix** (Schmitzer, 2016; Gerber and Maggioni, 2017). In favourable cases – e.g. if the cost $\mathbf{C}(x, y)$ is a strictly convex function of the distance in a low-dimensional feature space $\mathcal{X}$ – these algorithms reach the desired log-linear complexity that is expected in dimension $\text{D} = 1$ for sorting problems. Unfortunately though, these algorithms are notoriously hard to parallelize: no GPU implementation of these methods has ever been made available.

3. **Continuous schemes based on PDE theory.** In the Wasserstein-2 setting, the key insights of (Brenier, 1991) discussed page 100 allow us to link OT with standard problems in **fluid mechanics** that can be solved using variational or finite element-like methods.

   The most well-known of these algorithms is the *dynamical* scheme of (Benamou and Brenier, 2000), whose convergence to the underlying transport problem has recently been shown in the general case (Lavenant, 2019c). Going further, ulterior papers directly solve the Monge-Ampère equation satisfied by the optimal dual potential (Haker et al., 2004; Benamou et al., 2014), minimize explicit variational problems with efficient discretization schemes (Haber et al., 2010) or solve a flow problem formulated in a wavelet basis, possibly on 3D meshes (Dominitz and Tannenbaum, 2009).

   As shown for instance in (Lavenant et al., 2018), these methods are flexible and easy to formulate on geometric domains. Unfortunately, they are also limited to low-dimensional feature spaces and tend to be orders of magnitude slower than other solvers in the standard Euclidean setting.

4. **Semi-discrete OT, with non-degenerate point clouds.** Generalized fluid mechanics has been the major motivation behind the revival of OT theory in the 90's. Eager to simulate the flow of the incompressible Euler equation with Lagrangian schemes inspired by (Arnold, 1966) and (Brenier, 1999), a strong group of authors has focused on the large-scale resolution of semi-discrete OT problems between **"volumetric" point clouds and a weighted Lebesgue measure**. After a seminal paper on the multiscale resolution of OT problems (Mérigot, 2011), application to 3D data were showcased in (Lévy, 2015). Today, elegant simulations (Gallouët and Mérigot, 2016; Mérigot and Mirebeau, 2016) go hand-in-hand with cutting-edge theoretical results in fluid mechanics (Baradat and Monsaingeon, 2018). Albeit not yet fully published, modern codebases in the field are currently moving to GPUs (Ray et al., 2018) and reach an impressive level of performance: we recommend the "`github.com/sd-ot`" repository and the "`Geogram`" software to interested readers.

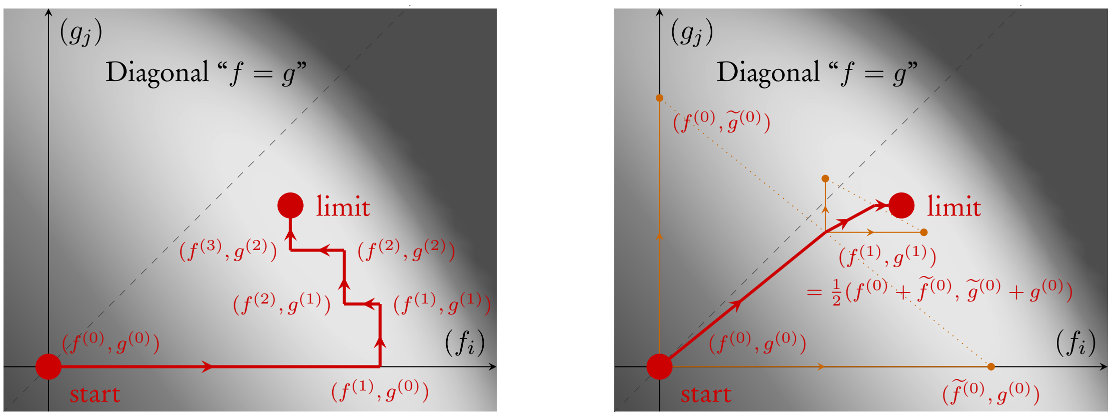
5. **Graph-based algorithms.** Solving a transport problem on *graphs* without ever computing the full matrix of pair-wise geodesic distances is highly relevant to operations research and network analysis. In the Wasserstein-1 setting, efficient algorithms generally rely on Beckmann's formulation presented Eq. (3.182) as in (Ryu et al., 2018; Li et al., 2018). Alternatively, when $\mathbf{C}$ is a quadratic cost function, the kernel convolutions involved in the Sinkhorn loop, Eq. (3.198), can be approximated with a heat diffusion (Solomon et al., 2015). Unfortunately though, such methods become numerically unstable as soon as the blurring scale $\sigma = \sqrt{\varepsilon}$ goes below a tenth of the graph's diameter.

6. **Stochastic, online setting.** Finally, when the input measures α and β are encoded through stochastic random variables $X_i \sim \alpha$ and $Y_j \sim \beta$ (as discussed Section 3.1.2), online variants of classic algorithms can be implemented. We refer to (Genevay et al., 2016) for an introduction.

Our focus: discrete measures in simple feature spaces. In this chapter, we focus on the simplest of all settings: discrete OT in a **vector feature space** $\mathcal{X} = \mathbb{R}^\mathrm{D}$ endowed with an **explicit cost function** such as $\mathbf{C}(x, y) = \frac{1}{p}\|x - y\|^p$. This structured problem is most relevant to computational anatomy and geometric applications, but could also be of interest to a wider audience. Most of the improvements discussed here can be transposed to other settings, and we hope that these pages will inspire our colleagues in all branches of the OT community.

Specifications of a good OT solver. Before describing our solution to the transport problem, we should make an explicit list of desirable properties for OT solvers. If "`OT`" denotes a finite program that approximates the Wasserstein cost $\mathrm{OT}(\alpha, \beta)$ between any two discrete measures α and β, we strive to enforce the following properties:

1. **Positivity.** For all input configurations, $\mathtt{OT}(\alpha, \beta) \geqslant 0$.
2. **Definiteness.** Ideally, $\mathtt{OT}(\alpha, \beta) = 0$ if and only if $\alpha = \beta$. At the very least, the partial derivatives $\nabla_\alpha \mathtt{OT}(\alpha, \beta)$ and $\nabla_\beta \mathtt{OT}(\alpha, \beta)$ should vanish whenever $\alpha = \beta$.
3. **Symmetry.** $\mathtt{OT}(\alpha, \beta)$ should be equal to $\mathtt{OT}(\beta, \alpha)$.
4. **Numerical stability.** No numeric overflow should ever occur.
5. **Robustness, monotonicity.** Simple situations should be handled well.
6. **Speed.** Solvers should have an $O(\mathrm{NM})$ time complexity or lower; log-linear and parallel algorithms should be used whenever possible.
7. **Scalability.** Solver should keep linear or log-linear memory footprints.
8. **Simple parameters.** Good solvers have no more than a couple of interpretable parameters: users should engineer their features, not their optimization routines.



(a) Sinkhorn, Alg. 3. (b) Symmetric Sinkhorn, Alg. 4.

Figure 3.24: The **concave maximization problem** $\mathrm{OT}_\varepsilon(\alpha, \beta)$ becomes nearly symmetric with respect to f and g when α gets close to β. (a) This situation is typical in measure-fitting applications, but hardly suited to a naive coordinate ascent. (b) In the Sinkhorn loop, we recommend the use of *averaged* iterations that interpolate between both ascent directions "$\widetilde{f}$" and "$\widetilde{g}$". The resulting program, detailed Algorithm 3.4, is more robust than the baseline Sinkhorn algorithm and is fully symmetric with respect to the input measures, even after a **finite number of iterations**.

Symmetrized iterations. As discussed in the previous section, Algorithm 3.3 already satisfies some of these axioms. But we can do (much) better.

First of all, we remark that the baseline Sinkhorn loop is not symmetric with respect to α and β, as illustrated Figure 3.24. To alleviate this problem, we advocate the use of *averaged* iterations that interpolate, at each step, between a maximization "over f" and a maximization "over g". The resulting symmetric Sinkhorn updates are showcased Figure 3.24.b and written out in the Algorithm below. An in-depth study of these iterations can be found in (Knight et al., 2014), with applications to the scaling of linear systems. To the best of our knowledge, this simple trick had never been used in the context of entropic OT.

To let the structure of our program stand out, we gray-out the lines associated to the de-biasing of Eqs. (3.203,3.209) and use high-level "SoftMin" notations. If $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ is a discrete measure and $(f_i) = (f(x_i))$ is a dual vector in $\mathbb{R}^{\mathrm{N}}$:

$$\min_{x \sim \alpha}{}_{\varepsilon} \left[\, \mathrm{C}(x, y) - f(x) \,\right] \overset{\text{def.}}{=} -\varepsilon \log \textstyle\sum_{k=1}^{\mathrm{N}} \alpha_k \exp \frac{1}{\varepsilon} \left[\, f_k - \mathrm{C}(x_k, y) \,\right] . \tag{3.220}$$

Algorithm 3.4: Symmetric Sinkhorn algorithm, with debiasing

Parameters: Cost function $\mathrm{C} : (x_i, y_j) \in \mathcal{X} \times \mathcal{X} \mapsto \mathrm{C}(x_i, y_j) \in \mathbb{R}$,
Temperature $\varepsilon > 0$.
Input: Positive measures $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$ with the same mass.

```
1: $f_i^{\beta\to\alpha}$, $g_j^{\alpha\to\beta}$, $f_i^{\alpha\leftrightarrow\alpha}$, $g_j^{\beta\leftrightarrow\beta}$ $\leftarrow$ $0_{\mathbb{R}^\mathrm{N}}$, $0_{\mathbb{R}^\mathrm{M}}$, $0_{\mathbb{R}^\mathrm{N}}$, $0_{\mathbb{R}^\mathrm{M}}$   ▷ Dual vectors.
2: repeat   ▷ The four lines below are executed simultaneously.
3:     $f_i^{\beta\to\alpha} \leftarrow \frac{1}{2} f_i^{\beta\to\alpha} + \frac{1}{2} \min_{y\sim\beta,\,\varepsilon} \left[\, \mathrm{C}(x_i, y) - g^{\alpha\to\beta}(y) \,\right]$ ,   ▷ $\alpha \leftarrow \beta$
       $g_j^{\alpha\to\beta} \leftarrow \frac{1}{2} g_j^{\alpha\to\beta} + \frac{1}{2} \min_{x\sim\alpha,\,\varepsilon} \left[\, \mathrm{C}(x, y_j) - f^{\beta\to\alpha}(x) \,\right]$ ,   ▷ $\beta \leftarrow \alpha$
       $f_i^{\alpha\leftrightarrow\alpha} \leftarrow \frac{1}{2} f_i^{\alpha\leftrightarrow\alpha} + \frac{1}{2} \min_{x\sim\alpha,\,\varepsilon} \left[\, \mathrm{C}(x_i, x) - f^{\alpha\leftrightarrow\alpha}(x) \,\right]$ ,   ▷ $\alpha \leftarrow \alpha$
       $g_j^{\beta\leftrightarrow\beta} \leftarrow \frac{1}{2} g_j^{\beta\leftrightarrow\beta} + \frac{1}{2} \min_{y\sim\beta,\,\varepsilon} \left[\, \mathrm{C}(y, y_j) - g^{\beta\leftrightarrow\beta}(y) \,\right]$ .   ▷ $\beta \leftarrow \beta$
4: until convergence up to a set tolerance.   ▷ Monitor the updates on the potentials.
5: return $f_i^{\beta\to\alpha} - f_i^{\alpha\leftrightarrow\alpha}$, $g_j^{\alpha\to\beta} - g_j^{\beta\leftrightarrow\beta}$   ▷ Debiased dual potentials $F(x_i)$ and $G(y_j)$.
```

Annealing strategy. Going further, we tackle the issue of *speed*. As showcased Figure 3.19, the Sinkhorn iterations generally improve the dual cost with small incremental steps that hardly resemble those of optimal sorting algorithms: we should be able to do better. This intuition is made rigorous by the complexity analyses of (Kosowsky and Yuille, 1994) and (Schmitzer, 2019), which link the Sinkhorn algorithm to auction iterations, as well as the PDE-based analysis of (Berman, 2017) and the very recent proof of (Léger, 2020). In typical scenarios, we should expect an approximate convergence of the Sinkhorn loop in:

$$\frac{\max_{\alpha\otimes\beta} \mathrm{C}}{\varepsilon} \simeq \left(\frac{\max_{i,j} \|x_i - y_j\|}{\sigma}\right)^p \qquad \text{iterations,} \tag{3.221}$$

where $\mathrm{C}(x,y) = \frac{1}{p}\|x-y\|^p$ and $\sigma = \varepsilon^{1/p}$ is the blurring scale associated to the Gibbs kernel $k_\varepsilon = \exp(-\mathrm{C}/\varepsilon)$. In the quadratic Wasserstein-2 setting, waiting for the Sinkhorn algorithm to converge becomes cumbersome as soon as the diameter-to-blur ratio exceeds 10 or 20.

Fortunately, we can reduce the number of iterations required to reach a nearly optimal dual pair to a mere $O(\log(\max\|x_i - y_j\|/\sigma))$ by **letting the temperature** ε **decrease across iterations**. After all, each auction or Sinkhorn iteration at temperature ε allows us to improve the dual cost by a step of size $\simeq \varepsilon$, up until reaching an ε-approximation of the optimal transport cost. Making large steps with a high temperature in the beginning, to fine-tune the dual potentials with a small ε in the end-game makes a lot of sense.

In practice, letting σ decay from a large estimation of the diameter towards a small target value is a simple yet efficient way of speeding-up the convergence of the Sinkhorn loop. This heuristic is known as *simulated annealing* in numerical analysis and *ε-scaling* in operations research. It was introduced as early as (Bertsekas, 1979) and (Kosowsky and Yuille, 1994) for the auction and Sinkhorn algorithms, respectively. Combining it with the symmetrization and debiasing introduced in the last few pages, we get an efficient algorithm that reads:

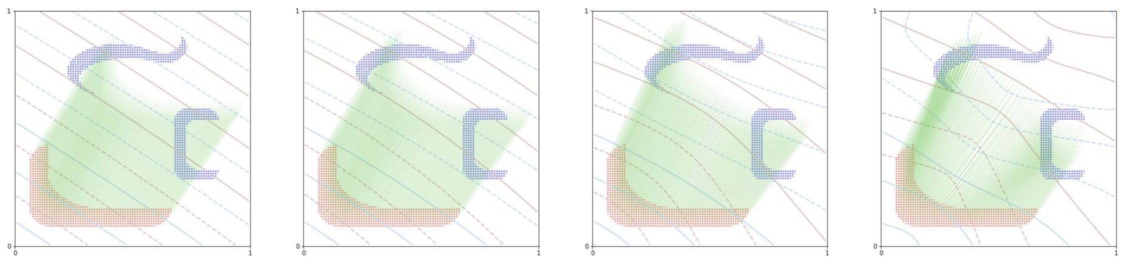
Algorithm 3.5: Symmetric Sinkhorn algorithm, with ε-scaling and debiasing

Parameters: Cost function $\mathrm{C} : (x_i, y_j) \in \mathcal{X}\times\mathcal{X} \mapsto \frac{1}{p}\|x_i - y_j\|^p \in \mathbb{R}$,
Target temperature $\varepsilon > 0$, given as a blurring scale $\sigma = \varepsilon^{1/p}$.

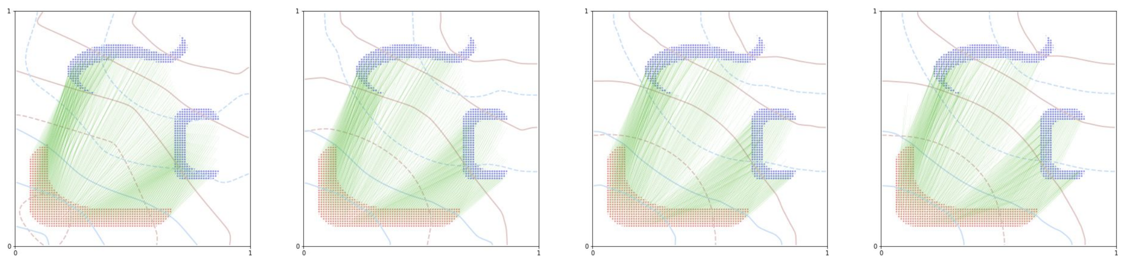
Annealing: Estimation of the diameter $\Delta \simeq \max_{i,j}\|x_i - y_j\|$,
Scaling ratio $q \in (0,1)$, default values of `.5` (fast) or `.9` (safe).

Input: Positive measures $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i\delta_{x_i}$ and $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$ with the same mass.

```
▷ Sensible initializations for $\varepsilon = +\infty$, with $[\mathrm{C}\star\beta](x_i) = \sum_{j=1}^{\mathrm{M}} \mathrm{C}(x_i,y_j)\beta_j$, etc. :
1: $f_i^{\beta\to\alpha},\ g_j^{\alpha\to\beta},\ f_i^{\alpha\leftrightarrow\alpha},\ g_j^{\beta\leftrightarrow\beta} \leftarrow [\mathrm{C}\star\beta](x_i),\ [\mathrm{C}\star\alpha](y_j),\ [\mathrm{C}\star\alpha](x_i),\ [\mathrm{C}\star\beta](y_j)$
2: for $s$ in $[\Delta,\ \Delta\cdot q,\ \Delta\cdot q^2,\ \dots,\ \sigma]$ do      ▷ $\lceil\log(\Delta/\sigma)/\log(1/q)\rceil$ iterations.
3:     $\varepsilon \leftarrow s^p$      ▷ The system cools down across iterations.
4:     $f_i^{\beta\to\alpha} \leftarrow \frac{1}{2} f_i^{\beta\to\alpha} + \frac{1}{2}\min_{y\sim\beta,\varepsilon}[\mathrm{C}(x_i,y) - g^{\alpha\to\beta}(y)]$,      ▷ $\alpha \leftarrow \beta$
       $g_j^{\alpha\to\beta} \leftarrow \frac{1}{2} g_j^{\alpha\to\beta} + \frac{1}{2}\min_{x\sim\alpha,\varepsilon}[\mathrm{C}(x,y_j) - f^{\beta\to\alpha}(x)]$,      ▷ $\beta \leftarrow \alpha$
       $f_i^{\alpha\leftrightarrow\alpha} \leftarrow \frac{1}{2} f_i^{\alpha\leftrightarrow\alpha} + \frac{1}{2}\min_{x\sim\alpha,\varepsilon}[\mathrm{C}(x_i,x) - f^{\alpha\leftrightarrow\alpha}(x)]$,      ▷ $\alpha \leftarrow \alpha$
       $g_j^{\beta\leftrightarrow\beta} \leftarrow \frac{1}{2} g_j^{\beta\leftrightarrow\beta} + \frac{1}{2}\min_{y\sim\beta,\varepsilon}[\mathrm{C}(y,y_j) - g^{\beta\leftrightarrow\beta}(y)]$.      ▷ $\beta \leftarrow \beta$
5: return $f_i^{\beta\to\alpha} - f_i^{\alpha\leftrightarrow\alpha},\ g_j^{\alpha\to\beta} - g_j^{\beta\leftrightarrow\beta}$      ▷ Debiased dual potentials $F(x_i)$ and $G(y_j)$.
```

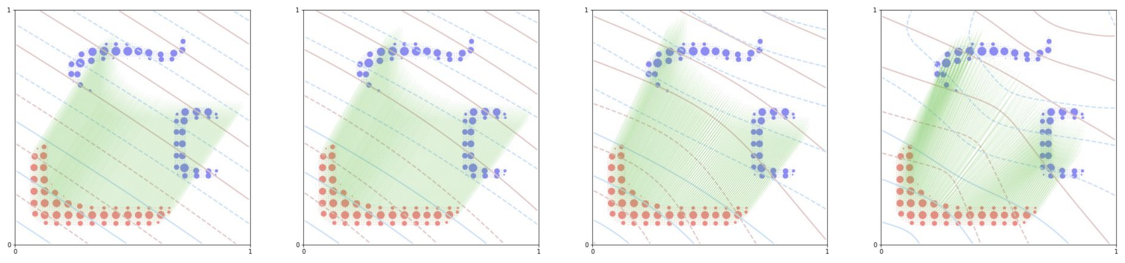


(a) $\texttt{it} = \texttt{0}$, $\sqrt{\varepsilon} = 2^0$. (b) $\texttt{it} = \texttt{1}$, $\sqrt{\varepsilon} = 2^{-1}$. (c) $\texttt{it} = \texttt{2}$, $\sqrt{\varepsilon} = 2^{-2}$. (d) $\texttt{it} = \texttt{3}$, $\sqrt{\varepsilon} = 2^{-3}$.

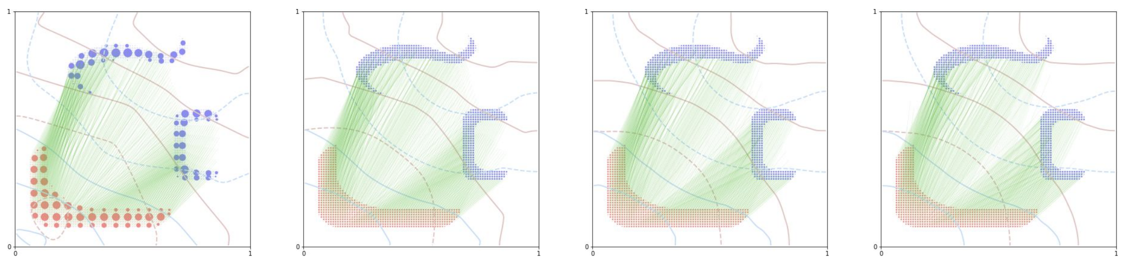


(e) $\texttt{it} = \texttt{4}$, $\sqrt{\varepsilon} = 2^{-4}$. (f) $\texttt{it} = \texttt{5}$, $\sqrt{\varepsilon} = 2^{-5}$. (g) $\texttt{it} = \texttt{6}$, $\sqrt{\varepsilon} = 2^{-6}$. (h) $\texttt{it} = \texttt{7}$, $\sqrt{\varepsilon} = \texttt{.01}$.

Figure 3.25: The symmetric Sinkhorn algorithm, with debiasing and ε-scaling – Algorithm 3.5. The two measures α (in red) and β (in blue) are displayed as weighted point clouds in the unit square $\mathcal{X} = [0,1] \times [0,1]$, endowed with the quadratic cost function $\mathrm{C}(x,y) = \frac{1}{2}\|x-y\|^2$. The de-biased potentials $F = f^{\beta\to\alpha} - f^{\alpha\leftrightarrow\alpha}$ and $G = g^{\alpha\to\beta} - g^{\beta\leftrightarrow\beta}$ are extended to the ambient space and displayed using blue and red contour lines, respectively. The Brenier mapping $x_i \mapsto x_i - \frac{1}{\alpha_i}\nabla_{x_i}\mathrm{S}_\varepsilon(\alpha,\beta) = x_i - \nabla F(x_i)$ is displayed as a green vector field throughout the iterations of Algorithm 3.5.



(a) $\texttt{it} = \texttt{0}$, $\sqrt{\varepsilon} = 2^0$. (b) $\texttt{it} = \texttt{1}$, $\sqrt{\varepsilon} = 2^{-1}$. (c) $\texttt{it} = \texttt{2}$, $\sqrt{\varepsilon} = 2^{-2}$. (d) $\texttt{it} = \texttt{3}$, $\sqrt{\varepsilon} = 2^{-3}$.



(e) $\texttt{it} = \texttt{4}$, $\sqrt{\varepsilon} = 2^{-4}$. (f) $\texttt{it} = \texttt{5}$, $\sqrt{\varepsilon} = 2^{-5}$. (g) $\texttt{it} = \texttt{6}$, $\sqrt{\varepsilon} = 2^{-6}$. (h) $\texttt{it} = \texttt{7}$, $\sqrt{\varepsilon} = \texttt{.01}$.

Figure 3.26: The multiscale Sinkhorn algorithm, symmetrized and debiased – Algorithm 3.6. The ε-scaling heuristic can be combined with multiscale decompositions of the input measures to reach an approximate $O(\mathrm{N}\log\mathrm{N} + \mathrm{M}\log\mathrm{M})$ complexity on the GPU. The configuration here is essentially the same as that of Figure 3.25, but a **coarse representation** of the input measures is leveraged to speed-up computations by an order of magnitude. (a-e) As long as $\sigma = \varepsilon^{1/p} = \sqrt{\varepsilon}$ is larger than the typical distance between coarse sampling locations, the "Sinkhorn" updates are sufficiently smooth to be performed on low-resolution data. (f-h) The last few iterations are computed on the high-resolution measures. We leverage a block-sparsity mask computed at (e) to **prune out** useless computations in the SoftMin reductions: in the "sum" of the "log-sum-exp", we skip the computation of distances between points whose clusters were not interacting with each other at iteration 4.

Results. As showcased Figure 3.25, the ε-scaling heuristic allows practitioners to compute acceptable approximations of the OT plan with no more than a handful of Sinkhorn iterations. Note, however, that no complete theory of simulated annealing for entropic OT exists as of 2020: rigorous proofs of convergence are only available for unrealistically slow temperature decays (Sharify et al., 2011), with gaps and technical quantization hypotheses on the input measures (Schmitzer, 2019) or for the simpler auction algorithm (Bertsekas, 1992). With a new understanding of entropic regularization emerging from our work, we hope to make theoretical advances on the topic in years to come.

Kernel truncation. Can we go further? In the general case, as discussed page 117, optimal transport problems cannot be solved with less than $O(\text{NM})$ operations: the cost function C should be evaluated on all pairs of points! Algorithm 3.5 allows users to compute an approximation of the OT distance with a dozen of optimized KeOps reductions, and should get close to optimal performances in weakly structured settings.

Nevertheless, when the data is intrinsically low-dimensional, we should strive to do better. Targeting the log-linear complexity of sorting algorithms, a line of *multiscale* OT solvers was started by a seminal paper of Quentin Mérigot (Mérigot, 2011): the core idea is to **leverage the structure of the virtual cost matrix** $(\mathrm{C}(x_i, y_j))$ **to prune out useless computations** in quadratic OT solvers. In the simple case of entropic OT, which was best studied from 2015 onwards by Bernhard Schmitzer (Schmitzer, 2019), multiscale schemes rely on **two key observations** made on the ε-scaling descent:

1. When the blurring radius $\sigma = \varepsilon^{1/p}$ is large, the dual potentials f and g define **smooth** functions on the ambient space that can be described accurately with **coarse samples** at scale σ. The first few iterations of the Sinkhorn loop can thus be performed quickly, on **sub-sampled encodings** $\widetilde{\alpha} = \sum \widetilde{\alpha}_i \delta_{\widetilde{x}_i}$ and $\widetilde{\beta} = \sum \widetilde{\beta}_j \delta_{\widetilde{y}_j}$ of our measures computed with an appropriate clustering method – e.g. a grid or K-means clustering.

2. The fuzzy transport plans π_ε, solutions of the primal problem $\text{OT}_\varepsilon(\alpha, \beta)$ for decreasing values of ε typically define a **nested sequence** of measures on the product space $\alpha \otimes \beta$. Informally, we may assume that:

$$\varepsilon < \varepsilon' \implies \text{Support}(\pi_\varepsilon) \subset \text{Support}(\pi_{\varepsilon'}) . \tag{3.222}$$

Meanwhile, if $(f_\varepsilon, g_\varepsilon)$ denotes an optimal dual pair for the coarse problem $\text{OT}_\varepsilon(\tilde{\alpha}, \tilde{\beta})$ at temperature ε, we know that the **effective support** of:

$$\pi_\varepsilon = \exp \tfrac{1}{\varepsilon}[f_\varepsilon \oplus g_\varepsilon - \mathrm{C}] \cdot \tilde{\alpha} \otimes \tilde{\beta} \tag{3.223}$$

is typically restricted to pairs of *coarse points* or clusters $(\tilde{x}_i, \tilde{y}_j)$ such that:

$$f_\varepsilon(\tilde{x}_i) + g_\varepsilon(\tilde{y}_j) \geqslant \mathrm{C}(\tilde{x}_i, \tilde{y}_j) - 5\,\varepsilon . \tag{3.224}$$

The so-called *kernel truncation trick* is all about leveraging this coarse-level information to **prune out useless computations** from SoftMin reductions at a finer level. We skip the computation of point-to-point interactions that would have a negligible impact on the updates of the dual potentials, and solve an entropic OT problem without ever computing the full matrix of pair-wise distances between our samples.

Using the block-sparse KeOps routines discussed Section 2.2.3 and idealized here with a "$\min_\varepsilon^{\text{Mask}}[\,\ldots\,]$" notation, we combine these key insights with Algorithm 3.5 as follows:

Algorithm 3.6: Multiscale Sinkhorn algorithm, with symmetrization and debiasing

Parameters: Cost function $\mathrm{C} : (x_i, y_j) \in \mathcal{X} \times \mathcal{X} \mapsto \frac{1}{p}\|x_i - y_j\|^p \in \mathbb{R}$,
Target temperature $\varepsilon > 0$, given as a blurring scale $\sigma = \varepsilon^{1/p}$,
Constraints' intensity $\rho > 0$, given as a maximum reach scale $r = \rho^{1/p}$
($r = \rho = +\infty$ retrieves balanced Optimal Transport).

Annealing: Estimation of the diameter $\Delta \simeq \max_{i,j} \|x_i - y_j\|$,
Scaling ratio $q \in (0, 1)$, default values of `.5` (fast) or `.9` (safe),
Truncation margin $\tau > 0$, default values of `3` (fast) or `5` (safe).

Input: Coarse-to-fine collections $(\alpha^{(1)}, \dots, \alpha^{(S)})$ and $(\beta^{(1)}, \dots, \beta^{(S)})$ of discrete measures $\alpha^{(k)} = \sum_{i=1}^{\mathrm{N}^{(k)}} \alpha_i^{(k)} \delta_{x_i^{(k)}}$ and $\beta^{(k)} = \sum_{j=1}^{\mathrm{M}^{(k)}} \beta_j^{(k)} \delta_{y_j^{(k)}}$ at scale $s^{(k)}$.
Typically, computed using grid or K-means clustering on high-resolution measures.

```
1:  $k,\ \alpha,\ \beta,\ \mathrm{N},\ \mathrm{M},\ \lambda \leftarrow 1,\ \alpha^{(1)},\ \beta^{(1)},\ \mathrm{N}^{(1)},\ \mathrm{M}^{(1)},\ 1/(1+(\Delta/r)^p)$   ▷ Coarsest scale.
    ▷ Sensible initializations for $\varepsilon = \Delta^p$, with $[\,\mathrm{C} \star \beta\,](x_i) = \sum_{j=1}^{\mathrm{M}} \mathrm{C}(x_i, y_j)\beta_j$, etc. :
2:  $f_i^{\beta\to\alpha}, g_j^{\alpha\to\beta}, f_i^{\alpha\leftrightarrow\alpha}, g_j^{\beta\leftrightarrow\beta} \leftarrow \lambda[\,\mathrm{C}\star\beta\,](x_i), \lambda[\,\mathrm{C}\star\alpha\,](y_j), \lambda[\,\mathrm{C}\star\alpha\,](x_i), \lambda[\,\mathrm{C}\star\beta\,](y_j)$
3:  $\mathcal{M}(\alpha\leftrightarrow\beta), \mathcal{M}(\alpha\leftrightarrow\alpha), \mathcal{M}(\beta\leftrightarrow\beta) \leftarrow \mathbf{1}_{\mathbb{R}^{\mathrm{N}\times\mathrm{M}}}, \mathbf{1}_{\mathbb{R}^{\mathrm{N}\times\mathrm{N}}}, \mathbf{1}_{\mathbb{R}^{\mathrm{M}\times\mathrm{M}}}$   ▷ Full masks.
4:  for $s$ in $[\Delta,\ \Delta\cdot q,\ \Delta\cdot q^2,\ \dots,\ \sigma]$ do   ▷ $\lceil \log(\Delta/\sigma)/\log(1/q) \rceil$ iterations.
5:      $\varepsilon \leftarrow s^p$   ▷ The system cools down across iterations.
6:      $\lambda \leftarrow 1/(1+(s/r)^p)$   ▷ Dampening coefficient, equal to 1 if $r = +\infty$.
7:      $f_i^{\beta\to\alpha} \leftarrow \frac{1}{2} f_i^{\beta\to\alpha} + \frac{1}{2}\lambda \min_{y\sim\beta,\,\varepsilon}^{\mathcal{M}(\alpha\leftrightarrow\beta)} [\,\mathrm{C}(x_i^{(k)}, y) - g^{\alpha\to\beta}(y)\,]$,   ▷ $\alpha^{(k)} \leftarrow \beta^{(k)}$
        $g_j^{\alpha\to\beta} \leftarrow \frac{1}{2} g_j^{\alpha\to\beta} + \frac{1}{2}\lambda \min_{x\sim\alpha,\,\varepsilon}^{\mathcal{M}(\alpha\leftrightarrow\beta)^\top} [\,\mathrm{C}(x, y_j^{(k)}) - f^{\beta\to\alpha}(x)\,]$,   ▷ $\beta^{(k)} \leftarrow \alpha^{(k)}$
        $f_i^{\alpha\leftrightarrow\alpha} \leftarrow \frac{1}{2} f_i^{\alpha\leftrightarrow\alpha} + \frac{1}{2}\lambda \min_{x\sim\alpha,\,\varepsilon}^{\mathcal{M}(\alpha\leftrightarrow\alpha)} [\,\mathrm{C}(x_i^{(k)}, x) - f^{\alpha\leftrightarrow\alpha}(x)\,]$,   ▷ $\alpha^{(k)} \leftarrow \alpha^{(k)}$
        $g_j^{\beta\leftrightarrow\beta} \leftarrow \frac{1}{2} g_j^{\beta\leftrightarrow\beta} + \frac{1}{2}\lambda \min_{y\sim\beta,\,\varepsilon}^{\mathcal{M}(\beta\leftrightarrow\beta)} [\,\mathrm{C}(y, y_j^{(k)}) - g^{\beta\leftrightarrow\beta}(y)\,]$.   ▷ $\beta^{(k)} \leftarrow \beta^{(k)}$
8:      if $k < S$ and $s < s^{(k)}$ then   ▷ Extrapolate to a finer scale.
9:          $\mathcal{M}(\alpha\leftrightarrow\beta)_{i,j} \leftarrow f_i^{\beta\to\alpha} \oplus g_j^{\alpha\to\beta} \geqslant \mathrm{C}(x_i^{(k)}, y_j^{(k)}) - \tau\varepsilon$   ▷ Block-sparsity mask.
10:         $\mathcal{M}(\alpha\leftrightarrow\alpha)_{i,j} \leftarrow f_i^{\alpha\leftrightarrow\alpha} \oplus f_j^{\alpha\leftrightarrow\alpha} \geqslant \mathrm{C}(x_i^{(k)}, x_j^{(k)}) - \tau\varepsilon$   ▷ Block-sparsity mask.
11:         $\mathcal{M}(\beta\leftrightarrow\beta)_{i,j} \leftarrow g_i^{\beta\leftrightarrow\beta} \oplus g_j^{\beta\leftrightarrow\beta} \geqslant \mathrm{C}(y_i^{(k)}, y_j^{(k)}) - \tau\varepsilon$   ▷ Block-sparsity mask.
12:         $f_i^{\beta\to\alpha} \leftarrow \lambda \min_{y\sim\beta,\,\varepsilon}^{\mathcal{M}(\alpha\leftrightarrow\beta)} [\,\mathrm{C}(x_i^{(k+1)}, y) - g^{\alpha\to\beta}(y)\,]$,   ▷ $\alpha^{(k+1)} \leftarrow \beta^{(k)}$
            $g_j^{\alpha\to\beta} \leftarrow \lambda \min_{x\sim\alpha,\,\varepsilon}^{\mathcal{M}(\alpha\leftrightarrow\beta)^\top} [\,\mathrm{C}(x, y_j^{(k+1)}) - f^{\beta\to\alpha}(x)\,]$,   ▷ $\beta^{(k+1)} \leftarrow \alpha^{(k)}$
            $f_i^{\alpha\leftrightarrow\alpha} \leftarrow \lambda \min_{x\sim\alpha,\,\varepsilon}^{\mathcal{M}(\alpha\leftrightarrow\alpha)} [\,\mathrm{C}(x_i^{(k+1)}, x) - f^{\alpha\leftrightarrow\alpha}(x)\,]$,   ▷ $\alpha^{(k+1)} \leftarrow \alpha^{(k)}$
            $g_j^{\beta\leftrightarrow\beta} \leftarrow \lambda \min_{y\sim\beta,\,\varepsilon}^{\mathcal{M}(\beta\leftrightarrow\beta)} [\,\mathrm{C}(y, y_j^{(k+1)}) - g^{\beta\leftrightarrow\beta}(y)\,]$.   ▷ $\beta^{(k+1)} \leftarrow \beta^{(k)}$
13:         $k,\ \alpha,\ \beta,\ \mathrm{N},\ \mathrm{M} \leftarrow k+1,\ \alpha^{(k+1)},\ \beta^{(k+1)},\ \mathrm{N}^{(k+1)},\ \mathrm{M}^{(k+1)}$

14: if $r = +\infty$ then   ▷ Balanced entropic transport: $\mathrm{S}_\varepsilon(\alpha, \beta)$.
15:     return $f_i^{\beta\to\alpha} - f_i^{\alpha\leftrightarrow\alpha},\ g_j^{\alpha\to\beta} - g_j^{\beta\leftrightarrow\beta}$   ▷ Debiased potentials $F(x_i)$ and $G(y_j)$.
16: else   ▷ Unbalanced transport of strength $\rho = r^p$ : $\mathrm{S}_{\varepsilon,\rho}(\alpha, \beta)$.
17:     return $\rho\,(e^{-f_i^{\alpha\leftrightarrow\alpha}/\rho} - e^{-f_i^{\beta\to\alpha}/\rho}),\ \rho\,(e^{-g_j^{\beta\leftrightarrow\beta}/\rho} - e^{-g_j^{\alpha\to\beta}/\rho})$   ▷ $F(x_i)$ and $G(y_j)$.
```

Contributions. A typical run is showcased Figure 3.26. In practice, our algorithm differs significantly from standard machine learning codes or from Bernhard Schmitzer's reference CPU implementation (Schmitzer, 2019). Some modifications were motivated by **mathematical insights** and may be relevant for all entropic OT solvers:

1. Our algorithm computes the **debiased dual potentials** F and G which correspond to the positive and definite Sinkhorn divergence S_ε, as detailed Eq. (3.209). In the unbalanced setting, we refer to Eqs. (3.214-3.216) and (Séjourné et al., 2019) for details.

2. For the sake of **numerical stability**, all computations are performed in the log-domain. We rely on the efficient log-sum-exp routines provided by the `KeOps` library and discussed around Eq. (2.43).

3. For the sake of **symmetry**, we use averaged updates on the dual potentials f and g instead of the standard alternate updates. This allows us to converge (much) faster when the two input measures are close to each other, which is typical in measure-fitting applications. We also make sure that whenever F and G are computed with Algorithm 3.6 and $\mathrm{S}_\varepsilon(\alpha,\beta) = \langle \alpha,\, F\rangle + \langle \beta,\, G\rangle$, even after a *finite* number of iterations:

$$\mathrm{S}_\varepsilon(\alpha,\beta) = \mathrm{S}_\varepsilon(\beta,\alpha)\ ,\ \ \mathrm{S}_\varepsilon(\alpha,\alpha) = 0 \ \text{ and } \ \partial_\alpha \mathrm{S}_\varepsilon(\alpha,\beta=\alpha) = 0\ . \tag{3.225}$$

4. When extrapolating from coarse to fine scales, we use the **genuine, closed-form expressions of Eqs. (3.194-3.195)** of our dual potentials instead of the simplistic piecewise-constant rule presented in (Schmitzer, 2019). In practice, this simple switch allows our code to be extremely aggressive in the descent: we only spend **one iteration per value of the temperature** ε. The importance of using the correct expression of the dual potentials in the multiscale descent was confirmed in private communications by Bernhard Schmitzer, who reported significant improvements to his reference solver.

5. Our gradients are computed using an explicit formula provided at convergence by the envelope theorem. This allows us to **bypass the costly backpropagation** through the Sinkhorn loop that is often advocated in machine learning papers, but wholly unnecessary. In practice, in the "forward" pass of our `PyTorch` implementation, we simply "detach" the dual potentials $(f_i^{\beta\to\alpha})$, $(g_j^{\alpha\to\beta})$, $(f_i^{\alpha\leftrightarrow\alpha})$ and $(g_j^{\beta\leftrightarrow\beta})$ before a final "extrapolation" update that is equivalent to the line 12 of Algorithm 3.6. Without ever writing a single explicit derivative, this trick allows us to use the following mathematical identities:

$$\begin{aligned}
\nabla_{\alpha_i}\mathrm{S}_\varepsilon(\alpha,\beta) \;&=\; F(x_i) \;=\; f_i^{\beta\to\alpha} \;-\; f_i^{\alpha\leftrightarrow\alpha}\,, && (3.226)\\
\tfrac{1}{\alpha_i}\nabla_{x_i}\mathrm{S}_\varepsilon(\alpha,\beta) \;&=\; \nabla F(x_i) \;=\; \tfrac{1}{1+\varepsilon/\rho}\,\frac{\sum_{k=1}^{\mathrm{M}} \exp\tfrac{1}{\varepsilon}\big[g_k^{\alpha\to\beta} - \mathrm{C}(x_i,y_k)\big]\cdot\nabla_{x_i}\mathrm{C}(x_i,y_k)}{\sum_{k=1}^{\mathrm{M}} \exp\tfrac{1}{\varepsilon}\big[g_k^{\alpha\to\beta} - \mathrm{C}(x_i,y_k)\big]} && (3.227)\\
&\qquad -\; \tfrac{1}{1+\varepsilon/\rho}\,\frac{\sum_{k=1}^{\mathrm{N}} \exp\tfrac{1}{\varepsilon}\big[f_k^{\alpha\leftrightarrow\alpha} - \mathrm{C}(x_i,x_k)\big]\cdot\nabla_{x_i}\mathrm{C}(x_i,x_k)}{\sum_{k=1}^{\mathrm{N}} \exp\tfrac{1}{\varepsilon}\big[f_k^{\alpha\leftrightarrow\alpha} - \mathrm{C}(x_i,x_k)\big]}\,.
\end{aligned}$$

It is well-known in the automatic differentiation community and relevant for all iterative algorithms that converge towards a limit stationary point: we refer to Section 2.1.2 for an introduction to the inner workings of modern autodiff engines.

Parallelism. Other tricks are more **hardware-dependent** and result from trade-offs between computation times and memory accesses on the GPU:

6. Multiscale CPU schemes generally rely on lists and sparse matrices which are *not* suited to GPUs. As discussed page 44, we implement the kernel truncation rule efficiently by combining a sorting pass with a **block-sparse truncation scheme** that enforces **contiguity in memory**. Just like we did with the log-sum-exp reduction, we abstracted the relevant CUDA codes in the `KeOps` library: this should allow them to reach a wider audience and improve the maintainability of the whole codebase.

7. For the sake of simplicity, we only implemented a **two-scale** algorithm which performs well when working with 50k-500k samples per measure. On the GPU, (semi) brute-force methods tend to have less overhead than finely crafted tree-like methods: using **a single coarse scale** is a good compromise for this range of problems.

8. **Our implementation is not limited to dimensions 2 and 3.** Users are free to use this program in conjunction with their favorite clustering schemes, and should expect decent speed-ups whenever their data has a low "intrinsic" Hausdorff dimension. In practice, users can specify a multiscale decomposition of their measures using vectors of class labels, output of a grid binning, K-means or any other clustering routine.

A pragmatic implementation. Crucially, our code does not perform any of the **sanity checks** on the truncation masks described in (Schmitzer, 2019): we cannot **guarantee** the correctness of our fast multiscale scheme. Running these tests during the descent would induce a significant overhead on the GPU, for little practical impact: we found the multiscale Sinkhorn loop to be remarkably stable, and precise enough for all practical purposes. The nice coarse-to-fine dynamics of Figure 3.26 is representative of a typical run of Algorithm 3.6.

As of today, our multiscale implementation should thus be understood as a **pragmatic**, GPU-friendly algorithm that provides quick estimates of the Wasserstein distance and gradient on large-scale problems. In the remainder of this chapter, we focus on bringing this highly optimized scheme to a wide audience, with an emphasis put on a *proper packaging* tailored to the needs of data scientists. Rigorous theoretical analysis is left for future works.

Analogy with the Quicksort algorithm. As discussed in Section 3.2.4, Optimal Transport can be understood as a generalized sorting problem. But how far can we go with this analogy? **In dimension** $\mathrm{D} = 1$, when $p \geqslant 1$, the optimal Monge map can be computed using a simple sorting pass on the data with $O(\mathrm{N}\log\mathrm{N} + \mathrm{M}\log\mathrm{M})$ complexity. At the other end of the spectrum, **generic** OT problems on **high-dimensional**, scattered point clouds have little to no structure and cannot be solved with less than $O(\mathrm{NM})$ operations.

From this perspective, multiscale OT solvers should thus be understood as **multi-dimensional Quicksort algorithms**: coarse cluster centroids and their targets play the part of median pivots. Our streamlined GPU implementation delivers on the promise made by (Mérigot, 2011; Gerber and Maggioni, 2017; Schmitzer, 2016, 2019) over the last decade: when the input data is intrinsically low-dimensional, the runtime needed to compute a Wasserstein distance should be closer to a $O(\mathrm{N}\log\mathrm{N} + \mathrm{M}\log\mathrm{M})$ than to a $O(\mathrm{NM})$.

Benchmarks. We now deem important to back our performance claims with rigorous benchmarks. To showcase the speed-ups provided by Algorithms 3.5 and 3.6 in a *realistic* setting, we focus on standard subsampled versions of the Stanford dragon, found in the reference archive (Curless and Levoy, 1996). We measure timings on the simplest of all registration problems: the optimal transport of a sphere onto the dragon, using a quadratic ground cost $\mathrm{C}(x, y) = \frac{1}{2}\|x-y\|^2$ in the ambient space $\mathcal{X} = \mathbb{R}^3$. This task is fairly representative of the needs of practitioners in computer vision and shape analysis, where scaling up OT computations has been a major concern for the best part of the XXI$^{\text{st}}$ century (Gold et al., 1998).

Problem solved. We first start by representing our 3D meshes as discrete probability measures:

$$\alpha \,=\, \sum_{i=1}^{\mathrm{N}} \alpha_i\, \delta_{x_i} \qquad \text{and} \qquad \beta \,=\, \sum_{j=1}^{\mathrm{M}} \beta_j\, \delta_{y_j}\,, \tag{3.228}$$

with **one weighted Dirac mass |area|** $\cdot$ $\delta_{\textbf{center}}$ **per triangle**. As discussed in Section 3.3.1, we then strive to solve the primal-dual entropic OT problem:

$$\begin{aligned}\mathrm{OT}_\varepsilon(\alpha,\beta) \,&=\, \min_{0\leqslant\pi\ll\alpha\otimes\beta} \langle \pi, \mathrm{C}\rangle \,+\, \varepsilon\,\mathrm{KL}(\pi, \alpha\otimes\beta) \quad \text{s.t.}\;\; \pi\,\mathbf{1} = \alpha \;\;\text{and}\;\; \pi^\intercal\mathbf{1} = \beta &(3.229)\\ &=\, \max_{f,g\in\mathcal{C}(\mathcal{X})} \langle \alpha, f\rangle + \langle \beta, g\rangle - \varepsilon\langle \alpha\otimes\beta, \exp\tfrac{1}{\varepsilon}[\,f\oplus g - \mathrm{C}\,] - 1\rangle &(3.230)\end{aligned}$$

as quickly as possible, optimizing on **dual vectors** $f_i = f(x_i)$ and $g_j = g(y_j)$ that encode an implicit transport plan:

$$\begin{aligned}\pi \,&=\, \exp\tfrac{1}{\varepsilon}[\,f\oplus g - \mathrm{C}\,]\;\cdot\;\alpha\otimes\beta\,, &(3.231)\\ \text{i.e.}\qquad \pi_{x_i\leftrightarrow y_j} \,&=\, \exp\tfrac{1}{\varepsilon}[\,f_i + g_j - \mathrm{C}(x_i,y_j)\,]\;\cdot\;\alpha_i\beta_j\,. &(3.232)\end{aligned}$$

Theoretical analysis: choosing a temperature. Understood as a smooth generalization of the standard theory of auctions, entropic regularization allows us to compute tractable approximations of the Wasserstein distance on the GPU.

The level of approximation is set using a single parameter, the temperature $\varepsilon > 0$ which is homogeneous to the cost function **C**. With a number of iterations that scales roughly in:

$$\begin{cases} O(\,\max_{i,j}\mathrm{C}(x_i,y_j)\,/\,\varepsilon\,) & \text{with the Sinkhorn and Auction algorithms}\\ O\Big(\,\log\,(\,\max_{i,j}\mathrm{C}(x_i,y_j)\,/\,\varepsilon)\,\Big) & \text{using an }\varepsilon\text{-scaling annealing strategy,}\end{cases} \tag{3.233}$$

we may compute an approximation OT_ε of the transport cost with precision $\simeq \varepsilon$.

Theoretical analysis: choosing a blurring scale. In practice, when $\mathrm{C}(x,y) = \frac{1}{p}\|x-y\|^p$ is the standard Wasserstein-p cost, the temperature ε is best understood through its p-th root:

$$\sigma \,=\, \sqrt[p]{\varepsilon}\,, \tag{3.234}$$

the **blurring scale** of the (exponential if "$p = 1$", Gaussian if "$p = 2$") Gibbs kernel $k_\varepsilon = \exp(-\mathbf{C}/\varepsilon)$ through which Sinkhorn-like algorithms interact with our weighted point clouds.

According to the heuristics presented page 120, we may expect to solve a regularized OT_ε problem with a number of iterations that scales in:

$$\begin{cases} O((\Delta/\sigma)^p) & \text{with the Sinkhorn and Auction algorithms} \\ O(\log(\Delta/\sigma)) & \text{using an } \varepsilon\text{-scaling annealing strategy,} \end{cases} \tag{3.235}$$

where $\Delta = \max_{i,j} \|x_i - y_j\|$ is the **diameter** of our configuration. The trade-off between speed (large σ) and precision (small σ) is illustrated Figure 3.22 and discussed page 116.

Solvers. We focus on the quadratic Wasserstein-2 setting ($p = 2$), which provides the most useful gradients for geometric applications – as illustrated Figure 3.16. Showcased in Figures 3.28, 3.29 and 3.30, the archetypal "statistics", "shape analysis" and "graphics" regimes correspond to varying values of the blurring scale $\sigma = \sqrt{\varepsilon}$ and of the number of samples $\sqrt{\text{MN}}$. In each of those three settings, we discuss the performances of three distinct algorithms:

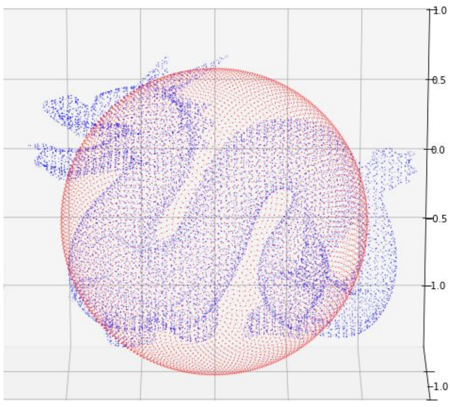
1. Our baseline is provided by Algorithm 3.3, a simple **Sinkhorn loop** implemented in the log-domain on the variables f and g for the sake of numerical stability. We discuss two separate implementations: a **tensorized** `PyTorch` script (which has a quadratic memory footprint) and a **scalable** `KeOps` code (which has a linear memory footprint). Accuracy-vs-time curves reflect an increasing number of iterations spent in the loop.
2. Our second competitor is the **symmetric Sinkhorn algorithm with ε-scaling**, presented in Algorithm 3.5. Once again, both `PyTorch` and `PyTorch+KeOps` implementations are discussed. Accuracy-vs-time values are computed by letting the scaling ratio q vary between `0.5` (fast) and `0.99` (accurate).
3. Finally, we discuss the performances of our **multiscale Sinkhorn solver** – Algorithm 3.6 – with two different truncation thresholds: $\tau = 1$ (fast) and $\tau = 5$ (safe). Coarse decompositions are computed using a cubic binning at scale $s^{(1)} \simeq \Delta/20$. Our `PyTorch` implementation relies on the block-sparse `KeOps` routines, discussed in Section 2.2.3. Once again, we trade time for accuracy by adjusting the value of the scaling ratio q.

In both Algorithms 3.5 and 3.6, we skip the "grayed-out" computations that correspond to the debiasing of Eqs. (3.203,3.209). This allows us to benchmark our solvers on the standard Schrödinger problem $\text{OT}_\varepsilon(\alpha, \beta)$ of Eqs. (3.229-3.230): keep in mind that full debiased runs would take twice as long.

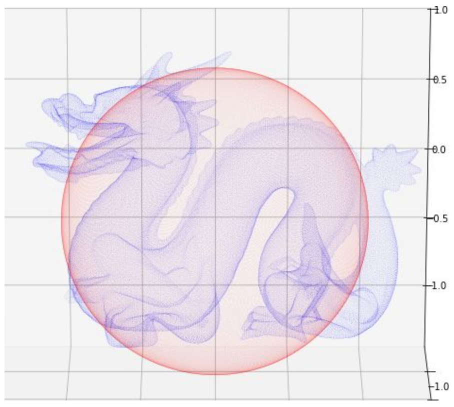
Convergence. We monitor convergence in our entropic OT solvers by computing a simple, meaningful quantity: the relative error made on the entropic Wasserstein "distance" $\text{OT}_\varepsilon^{1/p} = \sqrt{\text{OT}_\varepsilon}$. For any two measures α and β, we approximate the genuine target value of $\text{OT}_\varepsilon(\alpha, \beta)$ by the common value of all solvers run with very high precision settings. Then, if $\texttt{OT}_\varepsilon(\alpha, \beta)$ denotes the output of a given program,

$$\text{Rel}(\alpha, \beta) \;=\; \frac{\left|\sqrt{\texttt{OT}_\varepsilon(\alpha, \beta)} - \sqrt{\text{OT}_\varepsilon(\alpha, \beta)}\right|}{\sqrt{\text{OT}_\varepsilon(\alpha, \beta)}} \tag{3.236}$$

is chosen as our error criterion. This fits well with the needs of practitioners, who are usually interested in computing Wasserstein distances and gradients "up to a `1%` approximation error" as quickly as possible.



(a) $N = \texttt{10,000}$, $M = \texttt{11,102}$.



(b) $N = \texttt{200,000}$, $M = \texttt{202,520}$.

Figure 3.27: Two reference configurations. To compare approximate OT solvers in a fair and reproducible way, we focus on a meaningful transport problem between two reference measures: the unit sphere α of $\mathbb{R}^3$ and the Stanford dragon β, sampled with `10k` (a) or `200k` (b) points each. The typical diameter of this configuration is of order 1. As we compare several entropic OT solvers with each other, we measure convergence through the **relative error** made on the entropic Wasserstein distance $\sqrt{2 \cdot \text{OT}_\varepsilon(\alpha, \beta)}$, with **actual runtimes** – instead of abstract iteration numbers – on the x axis. All benchmarks are perfomed on a Google Cloud machine, with a Tesla V100 GPU that is slightly more efficient than the RTX 2080 Ti of Chapter 2.

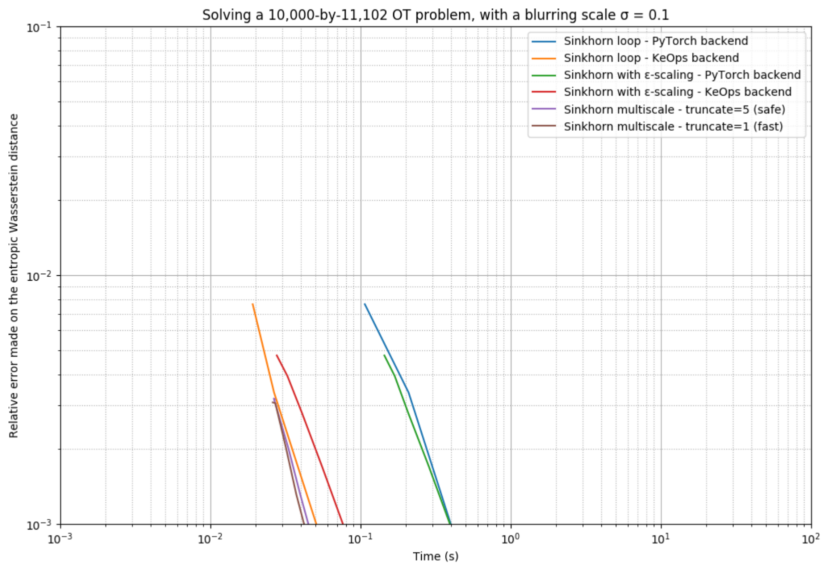


Figure 3.28: Benchmarks in a blurry "Cuturi-like" setting. A current trend in machine learning is to rely on **large blurring scales** to compute low-resolution gradients: **giving up on precision** is understood as a way of becoming robust to **sampling noise** (Genevay et al., 2019). To emulate this regime with our 3D point clouds, we first pick a (very) large blurring scale $\sigma = \sqrt{\varepsilon} = \texttt{0.1}$ and work with the subsampled point clouds of Figure 3.27.a. This corresponds to the setting of Figure 3.22.b. As evidenced here, when the **diameter-to-blur ratio** $\max \|x_i - y_j\|/\sigma$ **is of order** `10`, the baseline Sinkhorn algorithm **works just fine**. Improvements in this regime mostly come down to a clever low-level implementation of the SoftMin reduction, abstracted within the `KeOps` library: switching from `PyTorch` to `KeOps` allows us to get a **`x10` speed-up and a linear memory footprint**, but **annealing strategies are overkill** for this simple high-temperature problem.

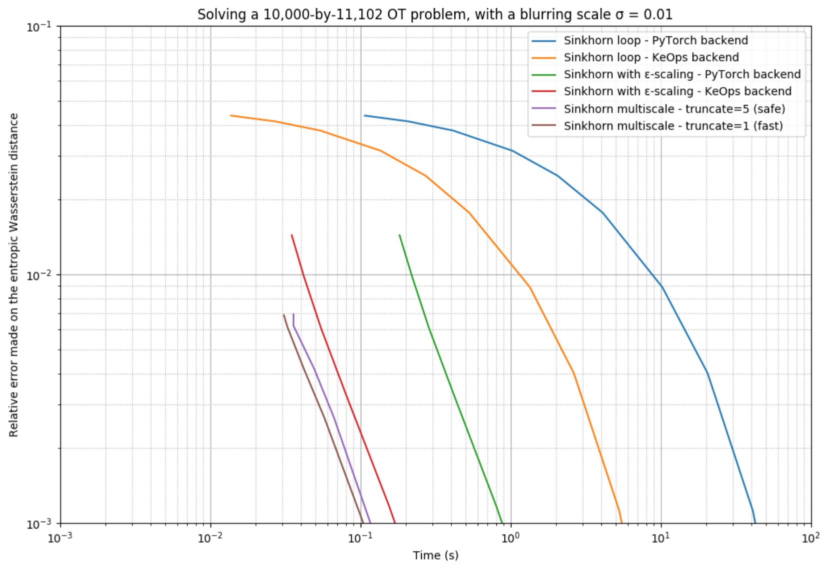


Figure 3.29: Benchmarks in a low-resolution "geometric" setting. Keep in mind, however, that the high-temperature setting of Figure 3.28 is not suited to applications in geometry and shape analysis: as illustrated Figure 3.22, the precision of the Sinkhorn divergence is inversely proportional to the blurring scale $\sigma = \sqrt{\varepsilon}$ and most applications require **diameter-to-blur ratios of order** 50 **or** 100.
As we pick a value of $\sigma = 0.01$ that is ten times smaller than that of Figure 3.28 (with the same low-resolution dataset), the computation time required by the standard Sinkhorn loop is multiplied by 100. On the other hand, the scaling-based routines only experience a x2-3 slow-down: our multiscale algorithm ends up being 400 times faster than a naive implementation of the Sinkhorn loop.

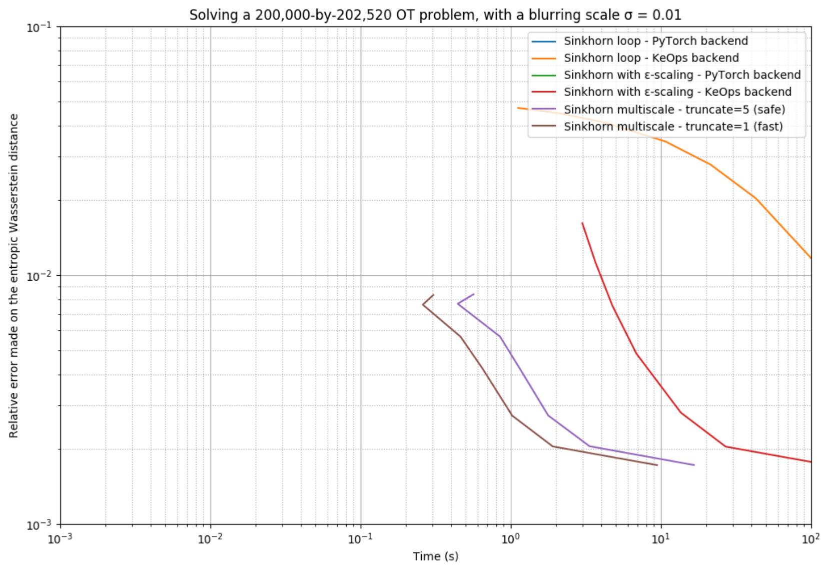


Figure 3.30: Benchmarks in a high-resolution "graphics" setting. As we switch to the high-resolution dataset of Figure 3.27.b, with 200k samples per measure, this trend becomes even more apparent. Tensorized `PyTorch` routines crash due to memory overflows and single-scale, quadratic Sinkhorn implementations experience a x100 slow-down. Remarkably, **the multiscale runtimes stay roughly linear with respect to the number of samples** and only slow-down by a factor 10-20. Even though high precision levels are still out-of-reach, practitioners can now compute Wasserstein distances "up to a 1% approximation error" in 100 to 300 milliseconds on high-resolution datasets.

Results. As shown here, our new multiscale OT solvers scale up to both large point clouds *and* small values of the temperature: taking $\varepsilon = \sigma^2$ of the order of $(\texttt{0.01})^2 = \texttt{1e-4}$ for normalized point clouds is now perfectly do-able. When implemented using our efficient solvers, Sinkhorn divergences define *robust* approximations of the Wasserstein distance that can be used in a wide range of settings, without any problem of numerical stability.

Our primary focus has always been to optimize OT solvers for applications to computational anatomy, in-between the two archetypal settings of Figures 3.29 and 3.30. In practice, in this setting, our multiscale implementation provides a `x100` speed-up compared with the state-of-the-art. As the first multiscale solver for discrete OT ever implemented on the GPU, it is both faster than GPU implementations of the Sinkhorn loop *and* log-linear implementations of multiscale algorithms on the CPU. Keeping a reasonable level of accuracy, we can now compute Wasserstein-2 distances and gradients between heart or brain meshes in less than `0.1s`: this opens a whole new range of applications, discussed in the remainder of this thesis.

Is Sinkhorn a fast algorithm? The Sinkhorn algorithm is known to have a linear convergence rate, with performances that fall off a cliff when the $(\max_{\alpha\otimes\beta} \mathbf{C}/\varepsilon)$ ratio becomes larger than 50 or 100. To improve upon this coordinate ascent scheme, many authors have proposed to use *accelerated* updates inspired by optimization theory (Walker and Ni, 2011): see, for instance, the Nesterov-like extrapolation rule of (Thibault et al., 2017).

This is a sensible approach: after all, the dual maximization problem $\text{OT}_\varepsilon(\alpha, \beta)$ is concave, smooth, and satisfies all the standard hypotheses in the field. Unfortunately though, assuming an appropriate tuning of the hyper-parameters, such strategies can at best provide a `x5-10` speed-up compared with the baseline Sinkhorn loop. This is not enough to compete with the ε-scaling heuristic, which provides excellent estimations of the optimal dual vectors in at most a *dozen* iterations, both for large *and* small values of the temperature ε.

Global vs local heuristics. At a fundamental level, we believe that the success of annealing strategies reflects the *global* structure of the "sorting" OT problem. Standard optimization theory is mostly concerned with solving *generic* convex problems with very high accuracy. Motivated by applications to, say, sparse recovery, most authors in the field focus on improving asymptotic convergence rates – the *end-game* – in a neighborhood of the global optimum, but say little about how we should get in such neighborhoods in the first place.

In the context of entropic OT, this focus on asymptotic results has clear limitations. Indeed, as soon as the $(\max_{\alpha\otimes\beta} \mathbf{C}/\varepsilon)$ ratio starts increasing beyond 20-50, the OT_ε problem becomes badly conditioned: as illustrated Figures 3.17.c, local gradients only encode information about the closest saturated constraints. As suggested by Figure 3.19, order 1 schemes typically improve the dual cost with updates of size of order ε: they cannot be competitive in the mid- and low-temperature settings which are of interest to a majority of users.

This analysis contrasts with the nice coarse-to-fine dynamics of annealing schemes, which let our algorithms match distributions hierarchically – from mean values to small details. This heuristic, which alters the OT_ε *problem* instead of accelerating its *gradient*, allows us to leverage our knowledge of the point clouds' structures. Even though no formal proof of convergence exists for Algorithms 3.5 and 3.6 as of 2020, the intuitive displays of page 121 vindicate, in our opinion, this age-old heuristic from operations research.

Relationship with the machine learning literature. As discussed at the end of this chapter, the recent line of stats-ML papers on entropic OT started by (Cuturi, 2013) has prioritized the quest for statistical robustness over computational efficiency. Consequently, in spite of their impact on fluid mechanics (Mérigot and Mirebeau, 2016), computer graphics (Lévy, 2015) and all fields where a manifold assumption (Gerber and Maggioni, 2017) may be done on the input measures, **works on multiscale methods have been mostly ignored by authors in the machine learning community**.

In recent years, some of the key geometric insights discussed page 122 have been rediscovered in the stats-ML literature. A notable example is (Altschuler et al., 2018a), where authors propose to use low-rank approximations of the Gaussian kernel matrix $(k_\varepsilon(x_i, y_j))$ in the standard Sinkhorn iterations of Eq. (3.199): this is roughly equivalent to working with coarse approximations of the input measures. Unfortunately, these ML-related works keep a focus on very high-temperature scenarios and never discuss ε-scaling strategies, coarse-to-fine schemes or introduce any idea that could allow practitioners to work with smaller amounts of entropic regularization.

For instance, (Altschuler et al., 2018b) only considers data normalized to fit in the unit hyper-cube $\mathcal{X} = [0, 1]^\mathrm{D}$, with a blurring scale $\sigma = \sqrt{\varepsilon}$ that ranges between $1/\sqrt{2 \cdot 5} = \texttt{0.31}$ and $1/\sqrt{2 \cdot 30} = \texttt{0.13}$. This massive amount of regularization may be of interest in settings where sampling noise is a concern. It is, however, hardly suited to any geometric application in shape analysis or computer graphics: as illustrated Figure 3.22, large blurring scales make our losses blind to all details in the input distributions.

We also note that in very high-temperature settings, computing blurry Wasserstein distances up to the third decimal is of little practical interest. The benchmarks of (Altschuler et al., 2018b) include comparisons with our solvers for $\sigma = 1/\sqrt{2 \cdot 15} = \texttt{0.18}$ on the unit cube, but we do not believe that making these runs with a (very) conservative value of the scaling ratio at $q = \texttt{0.95}$ reflects the needs of practitioners. The default value suggested by our "`GeomLoss`" package, $q = \texttt{0.5}$, would likely have been just as precise for all practical purposes – and considerably faster.

Bridging the gap between the ML and OT communities. We hope that the introduction to the *geometric* side of OT presented in these pages will be accessible to all readers. As researchers in the stats-ML community progressively acknowledge the geometric meaning of the temperature ε and the associated blurring scale $\sigma = \varepsilon^{1/p}$, abstract benchmarks will hopefully be replaced by healthy discussions on the trade-offs associated to the behaviours of OT solvers in different settings.

We aim at *bridging the gap* between communities that study OT from "geometric" and "statistical" perspectives by providing a fast OT solver that relies on key ideas from both worlds. More than our efficient (but always perfectible) implementations, we believe that this original introduction to entropic OT can have a fertilizing impact on the literature. Most importantly, the geometric plots of Figures 3.22, 3.25 or 3.26 convey a message that is very different from the standard diagrams of (Peyré and Cuturi, 2017) : we look forward to reading papers that combine these geometric insights with key ideas from other recent works, such as (Benamou et al., 2015), (Berman, 2017) or (Léger, 2020).

3.3.4 The GeomLoss package – future works

Progress over the last few years. Entropic OT has gone a long way since (Kosowsky and Yuille, 1994; Chui and Rangarajan, 2000) and (Cuturi, 2013). As discussed throughout this chapter, we now know how to make the Sinkhorn loop converge in a mere handful of iterations. Going further, optimal log-linear (instead of quadratic) runtimes are now at hand whenever the input data has a low intrinsic dimensionality.

Our single- and multi-scale OT solvers, detailed in Algorithms 3.5 and 3.6, are respectively suited to applications in statistics and geometry. They allow us to define a fully symmetric, positive and definite loss function that satisfies all the axioms of page 118 : the debiased Sinkhorn divergence S_ε. This affordable approximation of the Wasserstein distance is backed with essential theoretical results, presented in Theorem 3.1 and in (Genevay et al., 2019; Séjourné et al., 2019; Mena and Weed, 2019). It provides reliable gradients that define a low-frequency *Brenier mapping* between any two measures and can be tuned with a couple of meaningful, interpretable parameters.

A simple interface. These desirable properties come at the cost of *simplicity*: our solvers are numerically stable, orders of magnitude faster than the baseline Sinkhorn loop… but also harder to implement. This is most unfortunate: the striking simplicity of the Sinkhorn updates, Eq. (3.199), certainly played a major part in its widespread adoption.

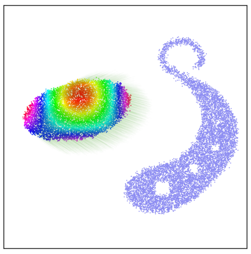
To let users get access to an efficient and well-tested implementation of our methods, we took the time to distribute our solvers in a user-friendly `Python` module: the `GeomLoss` package, which is freely available on `PyPi` (`pip install geomloss`). Numerous examples and tutorials are showcased in our documentation, that is available at:

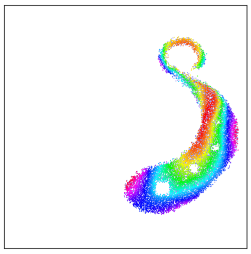
`www.kernel-operations.io/geomloss` .

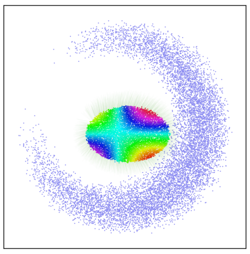
Out-of-the-box, Sinkhorn divergences can be computed using an idiomatic `PyTorch` interface:

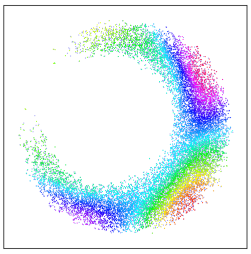
```python
import torch
from geomloss import SamplesLoss  # See also ImagesLoss, VolumesLoss

# Create some large point clouds in 3D
x = torch.randn(100000, 3, requires_grad=True).cuda()
y = torch.randn(200000, 3).cuda()

# Define a Sinkhorn (~Wasserstein) loss between sampled measures
S_e = SamplesLoss(loss="sinkhorn", p=2, blur=.05)

Sxy = S_e(x, y)  # By default, use constant weights = 1/number of samples
g_x, = torch.autograd.grad(Sxy, [x])  # GeomLoss fully supports autograd!
```

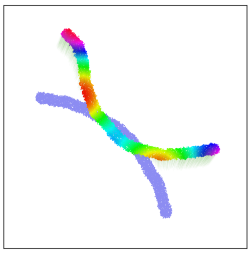
Future of the GeomLoss library. Going further, applications to computational anatomy, image processing or machine learning can be sketched out in a few lines of high-level `Python` code. Some of our demos are displayed in the next two pages, and illustrate the main strengths and weaknesses of optimal transport “on its own”. In months to come, we will work on implementing and packaging properly extensions to the basic framework: solvers for images and volumetric density maps, extensions to meshes and curves with orientation- and curvature-aware loss functions, etc. This work is detailed in Chapter 4.

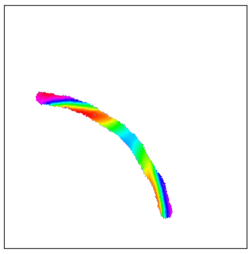

(a) Saxo - before.

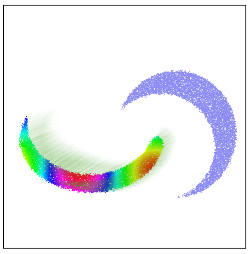

(b) Saxo - after.


(c) Crescent - before.


(d) Crescent - after.


(e) Worms - before.


(f) Worm - after.


(g) Moons - before.

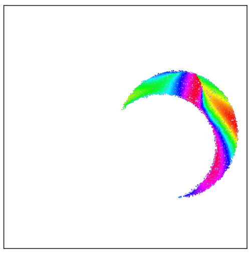
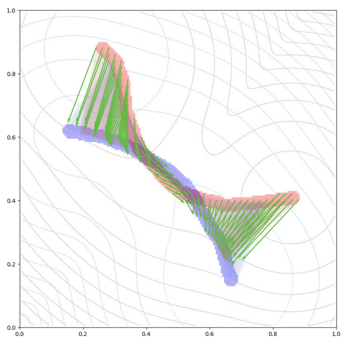
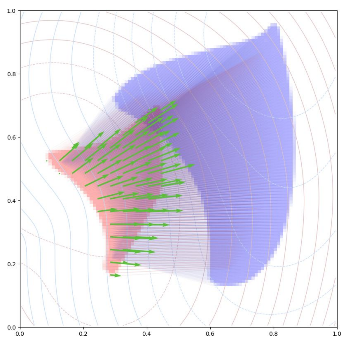

(h) Moons - after.

Figure 3.31: Optimal transport generalizes sorting to arbitrary feature spaces. Here, we showcase some matchings in the unit square $\mathcal{X} = [0,1] \times [0,1]$ computed with a Sinkhorn divergence S_ε, as $\mathrm{C}(x,y) = \frac{1}{2}\|x-y\|^2$ and $\sigma = \sqrt{\varepsilon} = \texttt{0.01}$. They are analogous to the 3D matchings of Figure 3.12, with a target β in blue and a source α displayed using a rainbow colormap. (a-d) OT can match any two measures with each other; for instance, an elliptic blob with a saxophone or a crescent. This may come handy in the literature on generative modelling, where authors strive to match Gaussian samples with generic empirical distributions. (e-f) When the two input measures are close to each other, Wasserstein-2 OT tends to retrieve good-looking monotonic matchings. (g-h) Keep in mind, however, that **OT does not always match salient geometric features with each other** – say, the ends of both crescents in the moons dataset. In challenging settings, OT should be used in feature spaces that take into account the local orientation and curvature or rely on global spectral coordinates. This is discussed in Chapter 4.


(a) Small deformations.


(b) Scalings, translations.

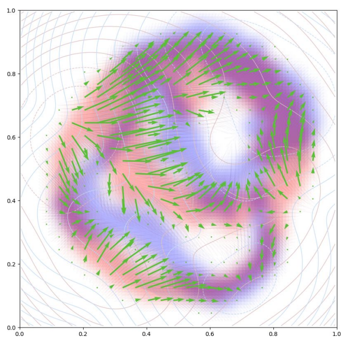
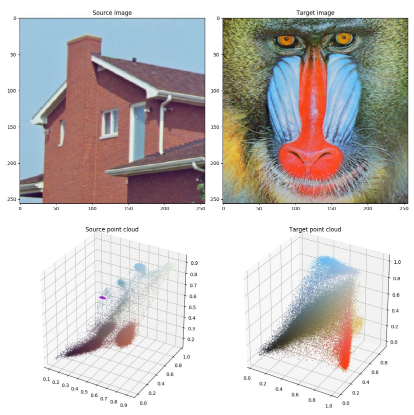
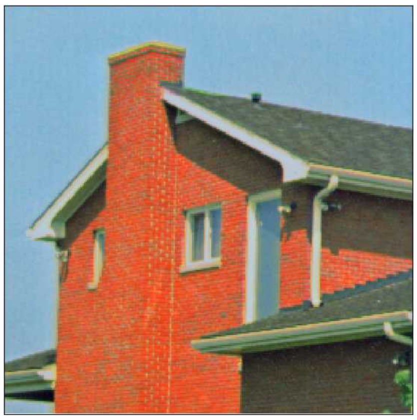

(c) No preservation of topology.

Figure 3.32: The `GeomLoss` routines are ideally suited to the processing of segmentation masks, even with little to no overlap. Since (Brenier, 1991), we know that Wasserstein-2 OT retrieves the unique gradient of a convex function that maps a measure onto another. In practice, this implies that the Sinkhorn divergence S_ε is robust to small deformations (a), translations and dilations (b). Keep in mind, however, that OT also has **clear limitations**. First of all, as evidenced Figure 3.31.h, it is *not* robust to rotations. (c) Going further, unlike spectral loss functions, OT does not preserve the topology of the input data; we illustrate this major weakness with pseudo-heart slices taken from the *spectral log-demons* paper (Lombaert et al., 2014).

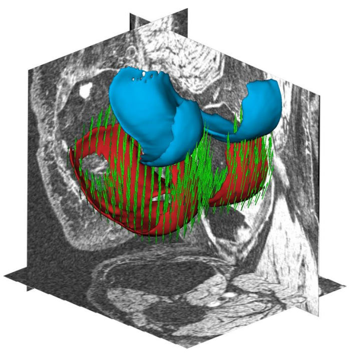


(a) Images A and B.

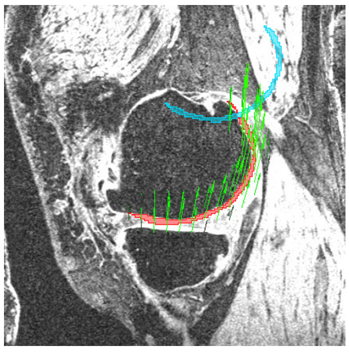


(b) Matching the distribution β onto α.

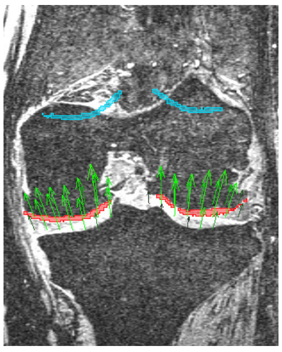
Figure 3.33: Color transfer is a nice illustraton of optimal transport in 3D. As discussed for instance in (Rabin et al., 2014), OT can be used to "equalize histograms" efficiently in a 3D color space. (a) The simplest way of doing so is to encode source and target images as point clouds α and β in the RGB unit cube $\mathcal{X} = [0, 1]^3$. Every pixel is represented as a weighted Dirac mass $\alpha_i \delta_{x_i}$ or $\beta_j \delta_{y_j}$, with $\alpha_i = 1/\mathrm{N}$ and $\beta_j = 1/\mathrm{M}$ by default. (b) Each pixel x_i in the source image then gets re-painted with a new color $x_i - \frac{1}{\alpha_i} \nabla \mathrm{S}_\varepsilon(\alpha, \beta)$ that roughly corresponds to a target in the histogram β. In practice, the regularized gradient of the Sinkhorn divergence S_ε allows users to transfer color palettes without overfitting on the precise distribution of the target, as illustrated here with a blurring scale of $\sqrt{\varepsilon} = \texttt{0.1}$. This is in line with the theoretical Lispschitz model of (Paty et al., 2019). Going further, better results could be obtained in spaces of patches or with the addition of relevant spatial features: see e.g. the impressive applications to texture synthesis of (Galerne et al., 2018; Leclaire and Rabin, 2019).



(a) 3D view.



(b) Coronal view.



(c) Sagittal view.

Figure 3.34: A typical use case: segmented knee caps kindly provided by Zhenlin Xu and Marc Niethammer from UNC Chapel Hill, from raw volumes of the OsteoArthritis Initiative dataset (Eckstein et al., 2012). The source α (in red) and target β (in blue) are respectively made up of 52,319 and 34,966 voxels – out of a pair of 192-192-160 volumes. The green vector field is the gradient of the Energy Distance of Eq. (3.154), an excellent baseline implemented alongside Sinkhorn divergences by the `GeomLoss` package. Rendering done with `3DSlicer` (Fedorov et al., 2012). As discussed in the literature since (Gold et al., 1998; Chui and Rangarajan, 2000), for real-life matching problems, OT is best used in conjunction with a generative model that enforces a prior on the structure of admissible deformations. The `GeomLoss` routines can be plugged in any shape analysis pipeline, as a straightforward replacement for the baseline chamfer and Gaussian kernel losses that are ubiquitous in the literature. In fractions of a second, our global loss functions can now provide **high-quality gradients** between segmentation masks or 3D point clouds which have **little to no overlap**.

Conclusion. After decades of research, the OT community is finally closing in on optimal algorithmic structures for the Wasserstein-2 "sorting" problem. Depending on the dimension of the input data and the level of precision required, state-of-the-art algorithms now compute optimal dual vectors with log-linear or quadratic runtimes, and always keep a linear memory footprint. Crucially, in both discrete (`GeomLoss`) and semi-discrete (`SD-OT`, `Geogram`, ...) settings, optimized CPU and GPU solvers are being packaged properly and made available through `Python` interfaces.

OT on geometric domains, graphs or with non-convex cost functions remain challenging problems... But as far as users are concerned, from 2020 onwards, standard Wasserstein-1 and Wasserstein-2 costs on $\mathbb{R}^\mathrm{D}$ should be as easy to use as generic chamfer and kernel distances.

Cleaning up the theory. As discussed throughout this chapter, the computational theory of entropic OT has made great strides over the last few years. Nevertheless, in our opinion, two major theoretical questions are still left to be answered:

1. Can we link the Sinkhorn algorithm with the blurred Wasserstein distances of page 116?
2. Can we prove the convergence of the scaling of Algorithm 5 in the general case?

We believe that these two problems are deeply linked with each other, and are working on the subject with François-Xavier Vialard and Bernhard Schmitzer. The entropic OT solvers packaged in the `GeomLoss` library are probably **good enough** for most practical purposes... But getting a **rigorous understanding** of the multiscale, wavelet-like behaviour of our algorithms as we add details through a decay of the blurring scale $\sqrt{\varepsilon}$ would be truly insightful. In some sense, couldn't we prove a Plancherel-like theorem for the Wasserstein distance? That's the dream!

How far can we go with OT? The next generation of OT solvers will probably meet the fate of standard linear algebra packages, buried in the foundations of higher-level projects. As the fundamental problem of OT computation is progressively getting solved in all practical settings, we expect the community to gradually switch its focus onto higher-level problems.

In Chapter 4, we discuss applications of OT theory to 3D shape analysis: the high-quality gradients of Sinkhorn divergences can greatly improve the reliability of registration pipelines in medical imaging. Likewise, the relevance of the Wasserstein metric for fluid mechanics and crowd modelling is now a well established fact (Santambrogio, 2015).

But is OT suited to statistics and machine learning research? Out-of-the-box, unfortunately, this seems very unlikely. The statistical properties of the Wasserstein distance have been studied extensively over the years (Dudley, 1969; Dobrić and Yukich, 1995; Weed et al., 2019), with catastrophic results in high-dimensional feature spaces: asymptotically, the number of samples needed to approximate a Wasserstein distance between two continuous distributions – its *sample complexity* – increases exponentially with the dimension of the measures' supports.

Facing the so-called *curse of dimensionality*, authors in the stats-ML literature have mostly used OT as a source of *inspiration* for the design of robust divergences. Researchers look for formulas that retain some of the appealing geometric features of the Wasserstein metric, but keep reasonable statistical properties in high-dimensional settings.

In statistics. A first strategy is to *regularize* and *constrain* the OT problem to make its solution robust to statistical fluctuations. This has been one of the major motivations behind the recent revival of entropic regularization (Cuturi, 2013) and the introduction of online OT solvers (Genevay et al., 2016), low-rank transport plans (Forrow et al., 2019) or Lipschitz-constrained Monge maps (Paty et al., 2019) in the literature.

In general, "robustified" solvers are far from being as efficient and versatile as the fast geometric algorithms discussed throughout this chapter… But in contexts where data is encoded with noisy *batches* of at most a few thousand samples at a time, statistical consistency trumps scalability. Both lines of work serve different purposes and can hardly be compared with each other. We recommend interested readers to check the "`Python Optimal Transport`" (`POT`) library (Flamary and Courty, 2017), which implements a good collection of ML-related algorithms with a unified user interface.

Generative adversarial networks. Beyond classical machine learning, a recent wave of papers on "GANs" (Goodfellow et al., 2014) and "Wasserstein-GANs" (Arjovsky et al., 2017) has sparked a strong interest for the applications of geometric measure theory to generative modelling. To understand these works, we should first mention one of the fundamental issues in image processing: the absence of mathematical formulas that can quantify the "perceptual" discrepancy between pairs of natural images. For instance, defined on the high-dimensional space of bitmaps $\mathcal{X} = \mathbb{R}^{\text{width} \times \text{height}}$, the standard Euclidean metric:

$$\|x - y\|^2_{L^2(\mathcal{X})} \;=\; \sum_{i,j} |\, x_{i,j} - y_{i,j} \,|^2 \tag{3.237}$$

does not capture – at all – the notions of *texture* and *shape* which are central to human vision. Deprived of any explicit criterion to optimize, how could we hope to generate new batches of *convincing* synthetic images?

To bypass this difficulty, (Goodfellow et al., 2014) made a fundamental remark: even though specifying an *explicit* perceptual distance on a space of images is probably out of reach, designing a parametric collection of dual test functions that models a "perceptual decision process" should be do-able. In practice, this insight leads researchers to compare distributions of images using dual losses in the mould of:

$$\text{Loss}_{\text{GAN}}(\alpha, \beta) \;\simeq\; \max_{\psi \in \mathbb{R}^{\text{P}}} \langle\, \alpha - \beta,\, g_\psi \,\rangle\;, \tag{3.238}$$

where g_ψ is a convolutional neural network parameterized by a vector of weights ψ. This choice is a sensible one for the processing of natural images, and can be justified in many different ways (LeCun et al., 1995; Mallat, 2016; Goodfellow et al., 2016).

Unfortunately, unlike the structured maximization problems of Eqs. (3.115,3.128) – *kernel norms* – or Eqs. (3.170,3.181) – *Wasserstein distance* – these generic optimization problems cannot be solved efficiently. Researchers thus compute these losses by using a gradient ascent scheme on the vector of weights ψ that parameterizes the *discriminative* network g_ψ.

As discussed page 66, generative modelling of images can then be cast as an explicit optimization problem. If $\alpha_\theta = \alpha \circ f_\theta^{-1}$ is a parametric distribution of generated images, pushforward of a reference (Gaussian) measure α under the action of a *generative network* f_θ and if β is an empirical distribution of images, the measure-fitting "GAN" problem reads, up to additional regularization terms:

$$\min_{\theta \in \mathbb{R}^Q} \text{Loss}_{\text{GAN}}(\alpha_\theta, \beta) \;\simeq\; \min_{\theta \in \mathbb{R}^Q} \max_{\psi \in \mathbb{R}^P} \langle\, \alpha \circ f_\theta^{-1} - \beta,\, g_\psi \,\rangle\,. \tag{3.239}$$

Wasserstein-GANs. In practice, such min-max problems are notoriously hard to solve: naive optimization schemes quickly diverge or cycle around saddle points without ever converging to an acceptable optimum.

In an attempt to ease the training of generative-adversarial pairs of networks (f_θ, g_ψ), (Arjovsky et al., 2017) proposed to restrict the optimization in the dual objective of Eq. (3.238) to Lipschitz discriminators g_ψ. This is essentially equivalent to promoting the use of the restricted Loss function:

$$\text{Loss}_{\text{W-GAN}}(\alpha, \beta) \;\simeq\; \max_{\psi \in \mathbb{R}^P} \langle\, \alpha - \beta,\, g_\psi \,\rangle \qquad \text{s.t.} \qquad g_\psi \text{ is 1-Lipschitz.} \tag{3.240}$$

In practice, this constraint is typically enforced (loosely) by clipping to $[-1, +1]$ the weights of all linear operators involved in the expression of g_ψ, i.e. by constraining the vector of *neural weights* ψ to stay in the unit ball of $\mathbb{R}^P$ for the L^∞ norm.

This heuristic makes sense: as discussed throughout this chapter – say, page 86 – constraining the set of *adversarial* test functions is usually a good way of making sure that the associated dual norms have smoother geometric properties, and "nicer" gradients. By analogy with the Kantorovitch-Rubinstein formulation of the Wasserstein-1 distance as a dual norm with respect to the full set of 1-Lipschitz test functions, Eq. (3.181), the authors of (Arjovsky et al., 2017) decided to call the adversarial Loss of Eq. (3.240) the "Wassertein-GAN" objective.

This unexpected connection between image processing and geometric measure theory appealed to a large number of theorists and practitioners. However, we deem important to stress that this dual loss function **is not actually related to optimal transport theory**. Defined as a dual norm over a set of **perceptual** test functions, generated by a given convolutional architecture $(\psi \mapsto g_\psi)$, the W-GAN loss is likely to be way more suited to image processing than the "genuine" Wasserstein-1 distance on $(\mathcal{X}, \|\cdot\|_{L^2})$, which relies on the *irrelevant* Euclidean norm of Eq. (3.237). The poor statistical properties of the Wasserstein-1 distance in high dimensional feature spaces come from the size of the set of 1-Lipschitz test functions, which is too large for its own good. Restricting ourselves to domain-specific test functions is therefore a sensible choice, but **has a major influence on the associated dual norm**.

Unfortunately, this essential point is often (always?) ignored in the specialized literature. Neglecting the essential prior encoded within the convolutional architectures of their imaging pipelines (Ulyanov et al., 2018), authors in the field often assume that the discriminator g_ψ – a convolutional neural network with a finely tuned architecture – somehow converges towards the "genuine" – and in this setting, completely irrelevant – optimal test function of the dual Kantorovitch problem of Eq. (3.181). This confusion is unfortunate, and at the source of a great deal of misunderstandings between authors in the "optimal transport" and "generative modelling" communities. Measure-theoretical works may provide insights and ideas to researchers working in a wide range of applied settings; but without strong empirical proofs, we should not *assume* that the geometric structure of the Wasserstein distance is ever preserved or relevant to the study of convolutional GANs.

Loss functions and models. Throughout Section 3.2, we showed that in comparison with Hausdorff and kernel losses, Wasserstein distances provide high-quality geometric gradients. In the remainder of this thesis, we explain how this new tool can improve the robustness of registration pipelines in medical imaging – where deformation models are typically flexible, loosely constrained and therefore sensitive to spurious local minimas. Keep in mind, however, that the switch from simple baselines to refined Sinkhorn divergences is **not always worth the effort**. As discussed page 82, whenever the generative or deformation model to estimate is heavily regularized, cheaper alternatives may do a fine job at a fraction of the price. We refer for instance to the use of sliced OT (Bonneel et al., 2015) for rigid pose estimation in (Bonneel and Coeurjolly, 2019).

In-between these two extreme settings, the trade-offs associated to the use of OT, Hausdorff or kernel Loss function are now pretty well understood for geometric applications in 3D. But what about machine learning problems? As discussed throughout this conclusion, the subject is currently a hot topic in the stats-ML literature, with statistical properties often taking precedence over geometric considerations. The picture is still pretty unclear as of 2020; but in years to come, we hope to see a comprehensive theory for measure-fitting problems emerge in high-dimensional feature spaces.

Final warning. The next chapter of this thesis is dedicated to the use of OT theory for shape analysis, both as a global loss function and as a cheap baseline metric for population study. As discussed in the last few pages, our algorithmic and theoretical progresses on entropic OT provide us with solid foundations to enter the field of computational anatomy. Nevertheless, before going any further, we wish to re-iterate some healthy criticism of optimal transport theory as a tool for data sciences: in spite of a flamboyant name and a flock of elegant results, this framework seldom provides ready-made answers to real-life problems.

First of all, as solutions of generalized sorting problems, OT plans are fundamentally reliant on the ground cost functions $\mathbf{C} : (x, y) \mapsto \mathbf{C}(x, y)$ defined on the feature space $\mathcal{X}$. Computing OT matchings with a poorly engineered ground metric is no more relevant than performing a *random* assignment between two lists of samples. Second, as illustrated page 133, the good algorithmic properties of OT come at the cost of the *preservation of topology*. As discussed in Chapter 5, defining relevant, affordable *and* topology-aware metrics on spaces of measures is still a major open problem in medical imaging.

Chapter 4

Encoding shapes as measures

in collaboration with Pierre Roussillon and Pietro Gori (Télécom ParisTech).

Key points – measure theory is a convenient abstraction for shape analysis:

1. **Invariance to re-meshing** is a property that should be enforced by shape analysis pipelines. In the last twenty years, this constraint has induced a flourishing literature in computer graphics and medical imaging: the focus has been put on instrinsic Laplacians, projection algorithms and geometric measure theory.
2. **Encoding shapes as weighted subsets** of a relevant space of features – e.g. (position, orientation, curvatures) triplets – is a convenient way of ensuring parameterization invariance. Researchers use weights that are proportional to the lengths of their segments or the areas of their triangles to **balance-out variations of the local sampling density**.
3. In this framework, shape elements are compared using an **extrinsic** distance on the ambient feature space, which replaces the **intrinsic** metric of the mesh. This method guarantees some robustness to topological noise and paves the way for efficient GPU implementations. On the flip side, it may also be understood as a fundamental blind spot of the theory.

Contributions – robust and scalable routines for shape analysis:

4. Leveraging our work on smooth optimal transport, we propose robust and scalable **geometric loss functions** for density maps, meshes and curves. Our `GeomLoss` package is freely available on the `PyPi` repository (`pip install geomloss`), with tutorials and documentation handy on our reference website:
   `www.kernel-operations.io/geomloss`
5. We discuss a particle-based Lagrangian scheme for the computation of Wasserstein barycenters: this algorithm provides a simple way of interpolating between neighboring distributions. It is efficient yet easy to implement and raises interesting theoretical questions.
6. With routines that can now scale up to large (500,000-1,000,000) collections of curves, we propose simple transport-based algorithms for the processing of **brain tractograms**. This work is still very much in progress, as we are yet to assess quantitatively the choice of **anatomical features** that should be used to encode white matter fibers.

Chapter 4 – Encoding shapes as measures:

4.1 A robust encoding for shapes

Throughout the last two chapters, we have explained how to speed-up a wide variety of computations: the `KeOps` and `GeomLoss` libraries bring a performance boost in physics, image processing and data sciences. Deep down, however, our main motivation has always been to work in the field of **computational anatomy**: the processing of shape data in a medical setting. As discussed Section 1.2, this is a challenging topic: the absence of uniform coordinate systems or clean mesh structures prevents us from representing shapes in a canonical vector space. Surfaces may also come with holes or sampling artifacts that must be handled with care.

Working with weighted sets: measures. A good way of mitigating these issues is to work with implicit surfaces (Alexa et al., 2001; Amenta and Kil, 2004) or intrinsic operators such as the mesh Laplacian (Zhou et al., 2005; Bronstein et al., 2017). Alternatively, as detailed in Chapter 3, we can represent shapes as discrete measures: **weighted** point clouds on $\mathbb{R}^\mathrm{D}$ or some other feature space $\mathcal{X}$:

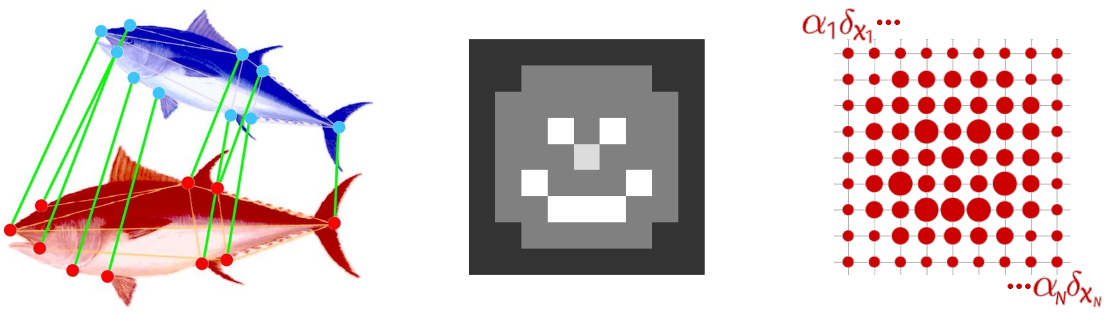
$$\alpha ~=~ \textstyle\sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}~, \qquad \text{with} \qquad (\alpha_i) \in \mathbb{R}_{\geqslant 0}^{\mathrm{N}}~,~(x_i) \in \mathcal{X}^{\mathrm{N}}~. \tag{4.1}$$

A built-in invariance to re-meshings. We discuss choices for the weights α_i and features x_i over the next few pages. An appealing trait of measure theory is that it lets us identify with each other objects whose supports do not overlap. For instance, as detailed page 68, we can say that for small values of ε:

$$\delta_0 ~=~ \tfrac{1}{2}\,\delta_0 + \tfrac{1}{2}\,\delta_0 ~\simeq~ \tfrac{1}{2}\,\delta_{-\varepsilon} + \tfrac{1}{2}\,\delta_{+\varepsilon} \tag{4.2}$$

for the weak-$\star$ topology associated to the convergence in law. Going further, just as in the classic definition of the Riemann integral, discrete measures converge to continuous objects if they are properly weighted by arc-length.

In this manuscript, as discussed Section 3.2, we restrict ourselves to computing quantities that are well-defined with respect to our measure-theoretic encodings and **continuous with respect to the convergence in law**: this ensures that our algorithms are numerically stable and robust to small deformations or re-meshings.



(a) Labeled data. (b) Density map. (c) As a measure.

Figure 4.1: Illustrating the challenge of working with unlabeled data, using tunas adapted from (Addis et al., 2010). (a) In favorable settings, well-defined *landmarks* can be identified on input images. As discussed Section 1.2.2, we can then encode shapes as large vectors: *labeled* point clouds that we process coordinate-wise. (b) Unfortunately, most problems in medical imaging cannot be dealt with so easily. As illustrated Figure 1.1, 1.9, 1.11, 3.32, 3.34 or 4.5, getting access to a segmentation "heatmap" for every type of tissue is often the best that we can hope for. (c) Introduced in Chapter 3, the theory of *measures* allows us to handle these weighted sets in a way that is principled, efficient and robust to re-parameterizations. As detailed Section 3.1.2, segmentation maps can be understood as sums of weighted Dirac masses located on pixel centers, with a weight that is proportional to the local density.

4.1.1 Density maps

Segmentation maps: bitmaps vs. weighted point clouds. For the sake of simplicity, we first understand shapes as distributions of mass on the ambient space: 2D or 3D segmentation maps that may be represented as binary masks or soft heat maps. As discussed Section 3.1.2, we can encode these objects in two different ways, with complementary strengths and weaknesses:

1. **Bitmaps.** Segmentation algorithms generally output a probability heatmap as a 2D image or 3D volume. This representation is well supported by modern libraries that provide efficient routines for convolutions, subsampling and Fourier transforms. On the flip side, *voxellization* is relatively ill-suited to the fine processing of 3D deformations, as mid- to high-resolution volumes quickly stop fitting contiguously in memory.

2. **Weighted point clouds.** Alternatively, we can use an explicit point cloud $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ with a vector of weights $(\alpha_i) \in \mathbb{R}^{\mathrm{N}}_{\geqslant 0}$: each weighted Dirac mass $\alpha_i\delta_{x_i}$ stands for a non-empty voxel localized around its center x_i with mass α_i. This encoding is ideally suited to represent thin structures with a small geometric support: the `KeOps` library lets us process clouds of $\mathrm{N} =$ `100k` to `1M` points in fractions of a second.

Working with images. With the advent of reliable segmentation networks, illustrated Figure 1.9, the study of positive density maps is becoming increasingly relevant. We must stress, however, that research in computational anatomy has historically been more concerned with *images* than density maps: grids or volumes filled with *intensities* that cannot be interpreted as weights. We will look into generalizing our methods to these signals in the future, but have so far prioritized the study of *meshes* and *curves*: as illustrated Figure 1.11, these representations are ideally suited to the study of deformations. They allow us to represent shapes and their variations with algorithms that are both memory efficient and GPU-friendly.

4.1.2 Curves

Discrete curves. In practice, we identify a curve in the ambient space $\mathbb{R}^{\mathrm{D}}$ with a finite collection of P segments encoded as a wireframe mesh:

$$A^* = (x_i^*,\, \Lambda_k)\,, \qquad \text{where} \quad \underbrace{(x_i^*) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}}_{\text{vertices}} \quad \text{and} \quad \underbrace{(\Lambda_k) \in [\![1,\mathrm{N}]\!]^{\mathrm{P}\times 2}}_{\text{edges}}\,. \tag{4.3}$$

The integer array $(\Lambda_k)_{k\in[\![1,\mathrm{P}]\!]}$ is the list of the P pairs of indices $(s,t) \in [\![1,\mathrm{N}]\!]^2$ for the end-points of the segments $[x_s^*, x_t^*]$ that make up the curve A^*.

Distributions of mass. To encode a discrete curve as a measure, we can identify its segments with weighted Dirac masses in $\mathbb{R}^{\mathrm{D}}$, as illustrated Figure 4.2 (Glaunes et al., 2004). Formally, we turn a curve A^* into a discrete measure $\alpha = \sum_{k=1}^{\mathrm{P}} \alpha_k \delta_{x_k}$ with the function:

$$\text{Measure} \,:\, A^* = (x_i^*,\, \Lambda_k) \mapsto (\alpha_k, x_k) \in \mathbb{R}^{\mathrm{P}}_{\geqslant 0} \times \mathbb{R}^{\mathrm{P}\times\mathrm{D}}\,, \tag{4.4}$$

where for all pair of indices $\Lambda_k = (s,t) \in [\![1,\mathrm{N}]\!]^2$, $\alpha_k\delta_{x_k}$ represents $[x_s^*, x_t^*]$ with:

$$\alpha_k \overset{\text{def.}}{=} \|x_s^* - x_t^*\| \qquad \text{and} \qquad x_k \overset{\text{def.}}{=} \tfrac{1}{2}(x_s^* + x_t^*)\,. \tag{4.5}$$

The weights α_k are proportional to arc length and make this representation robust to re-sampling: as detailed page 68, two discretizations of the same curve are close to each other for the weak-$\star$ topology. When all segments become finer, the Wasserstein distance between two measure encodings of the same shape tends to zero.

Orientation-aware features. Going further, we can work with higher-order features to take into account the orientation of our segments. We encode a curve A^* as a discrete measure $\alpha = \sum_{k=1}^{\mathrm{P}} \alpha_k \delta_{(x_k, \vec{n}_k)}$ on $\mathcal{X} = \mathbb{R}^{\mathrm{D}} \times \mathbb{S}^{\mathrm{D}-1}$ with the function:

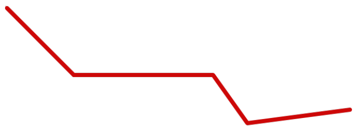
$$\text{Measure} \,:\, A^* = (x_i^*,\, \Lambda_k) \mapsto (\alpha_k, x_k, \vec{n}_k) \in \mathbb{R}^{\mathrm{P}}_{\geqslant 0} \times \mathbb{R}^{\mathrm{P}\times\mathrm{D}} \times \mathbb{R}^{\mathrm{P}\times\mathrm{D}}\,, \tag{4.6}$$

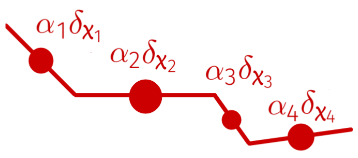
where for all pair of indices $\Lambda_k = (s,t) \in [\![1,\mathrm{N}]\!]^2$,

$$\alpha_k \overset{\text{def.}}{=} \|x_s^* - x_t^*\|\,, \qquad x_k \overset{\text{def.}}{=} \tfrac{1}{2}(x_s^* + x_t^*) \qquad \text{and} \qquad \vec{n}_k \overset{\text{def.}}{=} \tfrac{1}{\alpha_k}(x_s^* - x_t^*)\,. \tag{4.7}$$

The representation of curves and surfaces as measures has a long history in geometry (Federer, 1969). In computational anatomy, the encoding above is known under two different names: if all subsequent processings are *linear* with respect to the oriented directions $\vec{n}_k$, we say that the curve is represented as a **current** (Vaillant and Glaunès, 2005); alternatively, if our formulas are *invariant* to the orientations of the $\vec{n}_k$'s, we identify α with a measure on $\mathcal{X} = \mathbb{R}^{\mathrm{D}} \times \mathbb{S}^{\mathrm{D}-1}_{\pm}$ and say that it is a **varifold** (Charon and Trouvé, 2013).

Curvature. Going further, we can design higher-order embeddings to handle curvature and end-points in a consistent way: a first option is to process curvature as an additional coordinate in a lifted space of features (Charlier et al., 2017a). Alternatively, we can rely on the theory of normal cycles to define general order-2 encodings: we refer to (Roussillon, 2017) for an in-depth tutorial.


(a) Curve.



(b) As a measure.

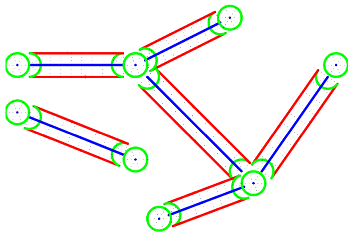
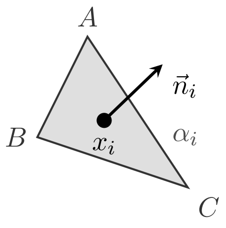

(c) Normal cycle of a collection of 6 segments.

Figure 4.2: Encoding a curve as a measure, with (c) from (Roussillon, 2017). (a) Working with curves, a simple way of designing parameterization-invariant methods is to consider the distribution of mass that a line defines on the ambient space $\mathcal{X} = \mathbb{R}^2$ or $\mathbb{R}^3$. (b) If a curve is given as a collection of segments, we can approximate the associated integrals with discrete sums. As detailed Eq. (4.1.2), we put one Dirac mass $\alpha_i \delta_{x_i}$ at the center x_i of each segment, with a weight α_i that is proportional to arc length. (c) Going further, we may consider higher-order features that take into account the local orientation and curvature. Since measure theory is additive, we only need to define the embedding of atomic shape elements – segments, vertices. Our curves end up being represented as weighted point clouds in a product space of dimension 2 to 9: we compare them with each other using the tools presented in Chapters 3 and 5.



$$
\begin{aligned}
ABC &\simeq \alpha_i \, \delta_{(x_i, \vec{n}_i)} \\
\alpha_i &= \tfrac{1}{2} \| \overrightarrow{AB} \otimes \overrightarrow{AC} \| \\
x_i &= \tfrac{1}{3}(A + B + C) \\
\vec{n}_i &= \pm \tfrac{1}{2\alpha_i} \overrightarrow{AB} \otimes \overrightarrow{AC}
\end{aligned}
$$

(a) Triangle, encoded as a Dirac measure on $\mathbb{R}^3 \times \mathbb{S}^2_{\pm}$.

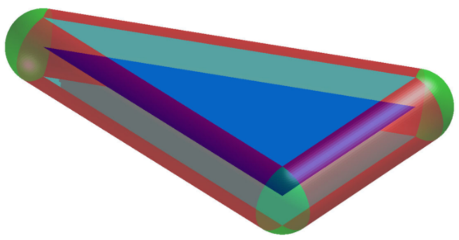
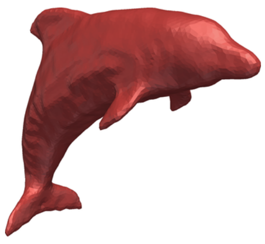
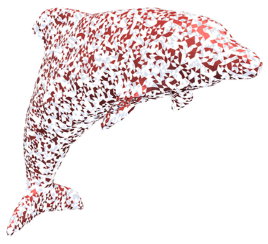
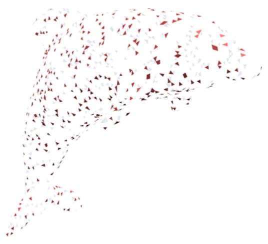

(b) Normal cycle of a triangle.

Figure 4.3: Encoding a triangle mesh as a measure, with (b) from (Roussillon, 2017). (a) Just like the curves of Fig. 4.2, surface meshes may be encoded as sums of weighted Dirac atoms, each of whom stands for a triangle element. We pick weights that are proportional to surface areas, and retain as features the centers of mass and unit un-oriented normals of our triangles. (b) Going further, defining curvature-aware features that are fully invariant to re-meshings can be tricky. Fortunately, the geometric framework of *normal cycles* (Federer, 1969) provides a satisfying answer to this problem: we refer to (Roussillon, 2017; Roussillon and Glaunès, 2019) for a kernel-centric overview.


(a) Full mesh.


(b) 30% of triangles.


(c) 5% of triangles.

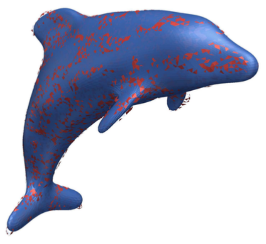

(d) Registered surface.

Figure 4.4: Measure encodings are robust to topological noise, from (Kaltenmark et al., 2017). A key feature of measure theory is that it is intrinsically robust to re-meshings and partial acquisitions. (b-c) In this example, degraded versions of a clean surface mesh are used to define rough approximations of the surface distribution associated to (a). In spite of glaring holes in the subsampled structures, the three objects remain close from a measure-theoretic perspective, as measured e.g. by the Wasserstein distance if we normalize total masses. (d) We use a geometric loss function from Chapter 3 to drive the registration of an "elastic" sphere onto a subsampled measure and retrieve a clean approximation of the original mesh. All relevant information has thus been preserved in the sparse, noisy encoding.

4.1.3 Meshes

Surfaces. We handle surfaces in a similar way to curves. Let us consider a triangle mesh:

$$A^* = (x_i^*,\, \Delta_k)\,, \qquad \text{where} \qquad \underbrace{(x_i^*) \in \mathbb{R}^{\mathrm{N}\times 3}}_{\text{vertices}} \text{ and } \underbrace{(\Delta_k) \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 3}}_{\text{faces}}\,. \tag{4.8}$$

The array (Δ_k) is the list of P triplets of indices $(s, t, u) \in [\![1, \mathrm{N}]\!]^3$ for the triangles $[x_s^*, x_t^*, x_u^*]$.

Oriented faces. Following our discussion on curves, we encode the mesh A^* as a **varifold**, i.e. as a discrete measure $\alpha = \sum_{k=1}^{\mathrm{P}} \alpha_k \delta_{(x_k, \vec{n}_k)}$ on $\mathcal{X} = \mathbb{R}^3 \times \mathbb{S}^2_{\pm}$ with:

$$\text{Measure} : A^* = (x_i^*,\, \Delta_k) \mapsto (\alpha_k, x_k, \vec{n}_k) \in \mathbb{R}^{\mathrm{P}}_{\geqslant 0} \times \mathbb{R}^{\mathrm{P}\times 3} \times \mathbb{R}^{\mathrm{P}\times 3}\,, \tag{4.9}$$

where for all triplet of indices $\Delta_k = (s, t, u) \in [\![1, \mathrm{N}]\!]^3$:

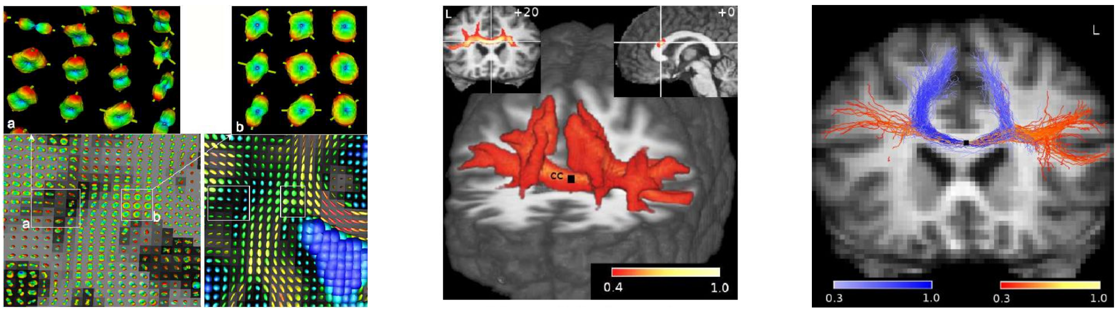
$$\begin{aligned} \alpha_k &= \text{Area}(x_s^*, x_t^*, x_u^*) &&= \tfrac{1}{2}\,\|(x_t^* - x_s^*) \otimes (x_u^* - x_s^*)\|\,, \\ x_k &= \text{Center}(x_s^*, x_t^*, x_u^*) &&= \tfrac{1}{3}(x_s^* + x_t^* + x_u^*)\,, \\ \vec{n}_k &= \text{Normal}(x_s^*, x_t^*, x_u^*) &&= \pm \frac{(x_t^* - x_s^*) \otimes (x_u^* - x_s^*)}{\|(x_t^* - x_s^*) \otimes (x_u^* - x_s^*)\|}\,. \end{aligned} \tag{4.10}$$

This encoding is illustrated Figure 4.3 and is robust to re-meshings – with an important caveat: convergence results only hold if mesh triangles do not become too thin. As detailed for instance in (Roussillon, 2017, Section 3.6), we can build ill-conditioned counter-examples – Schwarz lanterns – whose areas and normals are not faithful representations of the underlying continuous objects. Fortunately, these concerns are now addressed by standard remeshing algorithms and do not impact practical results.

Curvature. The unit normals $\vec{n}_k$ of the varifold encoding allow us to take into account the orientations of our triangles. In practice though, most salient geometric features are characterized by higher-order descriptors. Assuming that we have access to reasonable triangulations of our surfaces, a key problem is therefore to define a notion of curvature that is affordable, robust to topological noise and fully invariant to re-meshings.

From the perspective of geometric measure theory, a first option is to study the variations of a varifold encoding (Buet et al., 2015, 2017). Going further, as illustrated in Figures 4.2 and 4.3, we can introduce the **normal cycle** of a mesh to compute curvatures in a principled way (Federer, 1969): all relevant information is contained in an object that is defined in a consistent way for both continuous surfaces and discrete meshes.

We refer to (Morvan, 2008) for background on the topic, with applications to mesh processing (Chazal et al., 2009) and shape registration (Roussillon, 2017; Roussillon and Glaunès, 2019). In practice, this framework turns a triangle mesh into an expanded varifold representation: additional features are associated to edges and vertices to act as generalized mean and Gaussian curvatures. We will detail the associated computations in future works.



(a) Diffusion Tensor Imaging. (b) Fiber probability map. (c) Tractogram.

Figure 4.5: Illustrating tractography with images from (Descoteaux and Deriche, 2008).
(a) As illustrated Figure 3.5, modern MRI scans are remarkably versatile (Basser et al., 1994; Le Bihan et al., 2001). Using specific pulse sequences, doctors can now visualize the diffusivity of water inside a patient's body: at every voxel location of a DTI volume, a high-order tensor encodes the local anisotropy of the tissue – represented here as a 3D glyph. We can then use these diffusivity maps to segment a brain into meaningful *fiber tracks* that inform us on the brain's connectivity patterns. These structures may be encoded as soft segmentation maps (b) or as collections of 3D curves (c).

4.1.4 Brain tractograms

Tractography. Segmentation maps, meshes and curves are the most common representations of geometric data – but other types of structure are also worth studying. In neuro-anatomy for instance, the advances made on MRI sequences now let us retrieve 3D maps of the anisotropy patterns present in a patient's tissues. As illustrated Figure 4.5, this signal can then be processed to construct *fiber tracks* that model the anatomy of white matter bundles in the brain. Understanding the shape and structure of this data has important applications for neurology and neurosurgical planning: how should we encode it on our computers?

Probability maps. A first way of doing so is illustrated Fig. 4.5.b: we segment white matter into a collection of 3D segmentation maps, each of whom stands for a structure that is anatomically meaningful. Standard atlases in the field define up to a thousand such bundles: a single brain ends up being represented as a concatenation of 50 to 1,000 segmented volumes (Zhang et al., 2018; Wasserthal et al., 2018).

Encoding a tractogram. Alternatively, we can represent each bundle as a collection of N curves sampled with Q points in $\mathbb{R}^3$. Each *fiber* is encoded as a vector $(x^i_j)_{j\in[\![1,\mathrm{Q}]\!]} \in \mathbb{R}^{\mathrm{Q}\times 3}$ for $i \in [\![1,\mathrm{N}]\!]$ and must be processed in a way that is fully invariant to the flip operator:

$$\mathrm{Flip} \;:\; (x_1,\dots,x_\mathrm{Q}) \in \mathbb{R}^{\mathrm{Q}\times 3} \;\mapsto\; (x_\mathrm{Q},\dots,x_1) \in \mathbb{R}^{\mathrm{Q}\times 3}\,. \tag{4.11}$$

In practice, as illustrated Fig. 4.9, a full brain *tractogram* can be made up of $\mathrm{N} \simeq 10^6$ curves sampled with $\mathrm{M} \simeq 20$ points each. We encode such datasets as discrete measures $\alpha = \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}} \delta_{x^i}$ on a high-dimensional space of curves $\mathcal{X} = \mathbb{R}^{\mathrm{Q}\times 3}$, endowed with a flip-invariant quotient distance:

$$\mathrm{d}(x,y) \overset{\text{def.}}{=} \min\big(\, \|x-y\|_{\mathbb{R}^{\mathrm{Q}\times 3}},\; \|x-\mathrm{Flip}(y)\|_{\mathbb{R}^{\mathrm{Q}\times 3}}\,\big)\,. \tag{4.12}$$

Going further, we may add weight to the end points of the fibers or consider a domain-specific feature space $\mathcal{X}$: we refer to (Bertò et al., 2020) for an introduction.

4.2 Computing distances and gradients

Encoding meshes or curves as measures is a first step towards robust shape analysis: as detailed in the previous chapter, we can define loss functions between measures that satisfy the geometric axioms of pages 72 and 118 while remaining affordable. But can we translate these abstract results to shape analysis? Before going any further, we now briefly summarize the main take-aways of Chapter 3 for computational anatomy.

4.2.1 Images vs. Measures: choosing appropriate weights

Which gradient should we use? Computing the gradient of a loss function with respect to the vertices of a shape is more delicate than it seems. Looking back on the curve and mesh encodings of Eqs. (4.4,4.6,4.9), we turn a list of vertices $(x_i^\star) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ into a collection of weights $(\alpha_k) \in \mathbb{R}^{\mathrm{P}}_{\geqslant 0}$ and features $(x_k) \in \mathbb{R}^{\mathrm{P}\times\mathrm{D}}$ through the use of a "Measure" operator.

This change of variables has one major consequence: if the weights α_k are updated on-the-fly with the x_k's, the gradient field $\nabla_{x_i^\star}\text{Loss}(\sum_k \alpha_k \delta_{x_k})$ of a weakly continuous loss function with respect to the vertices $x_i^\star$ of a mesh is necessarily **orthogonal to the underlying shape**. This is a consequence of the invariance properties of our encodings: since sliding tangential movements of the $x_i^\star$'s on a shape do not affect the measure $\sum_k \alpha_k \delta_{x_k} = \text{Measure}(x_i^\star)$ up to discretization effects, they are discarded by the adjoint operator $\mathrm{d}_{x_i^\star}^\top\text{Measure}$ in the chain rule of Eq. (2.17).

To retrieve the appealing geometric gradients of Chapter 3, a sensible choice is therefore to work with **fixed weights** per shape element, as detailed in the second line of Algorithm 4.2. This alleviates all problems related to the orthogonality of the gradient and lets us retrieve a vector field $\nabla_{x_i^\star}\text{Loss}$ that has the same geometric properties as the Lagrangian velocity $\nabla_{x_k}\text{Loss}$.

Constant density vs. constant mass. From a mathematical perspective, the dichotomy between variable and fixed weights is understood as the distinction between the *varifold* and *measure* transport actions of deformations on shapes: we refer to (Charlier et al., 2017a, Section 6) for a detailed discussion. Similarly, in the medical imaging literature, authors tend to stress the difference between morphing images as *intensities* or as *densities*: in the first case, pixel values are preserved by local dilations and contractions; in the second, we rescale pixel values by the inverse of the Jacobian determinant of the deformation to preserve the total mass of the distribution.

Both approaches are sensible and correspond to different priors on the deformations to study: in practice, as detailed in Section 3.2.2 and Chapter 5, the interaction between raw gradient fields and structured deformation models is a central element of shape analysis pipelines. Nevertheless, as a default baseline for practitioners, we recommend to keep the weights (α_k) fixed – possibly normalized to sum up to 1 – and only modify them using an external growth model. This choice ensures the preservation of surface areas: we dissuade deformation models from turning small patches into massive bubbles and vice versa. This acts as a sensible regularization prior in most applications.

4.2.2 Hausdorff, Kernel and Wasserstein fidelities

ICP algorithm, kernel methods and optimal transport. Assuming that our shapes are properly encoded as measures, the main lessons of Chapter 3 hold and are effective: the theories of Hausdorff distances, kernel norms and optimal transport all induce algorithms that can be scaled up to high-resolution meshes. We discuss their properties in Section 3.2, with a focus on the computation of Wasserstein distances in Section 3.3. As illustrated in Figures 3.12, 3.27 and 3.34, we can now compute an optimal transport map between two collections of `100k+` triangles in fractions of a second.

Our results hold in all relevant settings. The theoretical analysis of Chapter 3 translates well to high-order measure encodings for shapes. For instance, on the product space $\mathcal{X} = \mathbb{R}^3 \times \mathbb{S}^2_\pm$ associated to the varifold encoding of Eq. (4.9), we can use the cost function:

$$\mathrm{C}(\,(x,\vec{n}),\,(y,\vec{m})\,) \;=\; \tfrac{1}{2}\|x-y\|^2 \;+\; \tfrac{\lambda}{2}(1-\langle \vec{n},\vec{m}\rangle^2) \tag{4.13}$$

for non-negative values of the angular sensitivity $\lambda \geqslant 0$. It is associated to the Gibbs kernel:

$$k_\varepsilon(\,(x,\vec{n}),\,(y,\vec{m})\,) \;=\; \exp\big[-\tfrac{1}{2\varepsilon}\|x-y\|^2\big] \,\cdot\, \exp\big[\tfrac{\lambda}{2\varepsilon}(\langle \vec{n},\vec{m}\rangle^2 - 1)\big]\,, \tag{4.14}$$

which is positive definite on $\mathcal{X} = \mathbb{R}^3 \times \mathbb{S}^2_\pm$ (Charon and Trouvé, 2013). Assuming that our shapes have bounded support, theoretical results such as Theorem 3.1 thus hold.

4.2.3 The `GeomLoss` library

Reminder on entropic OT. The methods that we present in the remainder of this chapter are all related to the entropic regularization of optimal transport, discussed Section 3.3.1. As detailed page 108, the multiscale solvers of Algorithms 3.5 and 3.6 return dual vectors $(f_i) \in \mathbb{R}^{\mathrm{N}}$ and $(g_j) \in \mathbb{R}^{\mathrm{M}}$ that are uniquely defined up to an additive constant, Eq. (3.172), and describe a fuzzy N-by-M transport plan:

$$\pi_{i,j} \;=\; \alpha_i \beta_j \,\cdot\, \exp \tfrac{1}{\varepsilon}\big[f_i + g_j - \mathrm{C}(x_i, y_j)\big] \tag{4.15}$$

between any two discrete measures $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$.

In the classical transport setting, when the strength ρ of the marginal contraints is set to $+\infty$, we assume that α and β have the same total mass. The transport plan π associated to the *optimal* dual vectors (f_i) and (g_j) then satisfies the two constraints $\pi\mathbf{1} = \alpha$ and $\pi^\top\mathbf{1} = \beta$. The distribution α is *fully* transported onto β as for all indices i and j:

$$\textstyle\sum_{j=1}^{\mathrm{M}} \beta_j \,\exp \tfrac{1}{\varepsilon}\big[f_i + g_j - \mathrm{C}(x_i, y_j)\big] \;=\; 1 \;=\; \sum_{i=1}^{\mathrm{N}} \alpha_i \,\exp \tfrac{1}{\varepsilon}\big[f_i + g_j - \mathrm{C}(x_i, y_j)\big]\,. \tag{4.16}$$

Alternatively, in the *unbalanced* generalization of optimal transport, the constraints are softened following Eq. (3.214). This corresponds to the use of *lazy* workers, whose willingness to transport mass between the source and target points roughly decays in $\exp[-\mathrm{C}(x,y)/\rho]$.

Working with Sinkhorn divergences. Assuming that de-biasing is enabled, as detailed Eq. (3.209), our solvers let us compute efficiently the value and gradients of the (de-biased) Sinkhorn divergence $\mathrm{S}_\varepsilon(\alpha,\beta)$. The de-biased potentials $(F_i) = (F(x_i)) \in \mathbb{R}^{\mathrm{N}}$ and $(G_j) = (G(y_j)) \in \mathbb{R}^{\mathrm{M}}$ do *not* encode a transport plan π. However, when the cost function $\mathrm{C}(x,y) \simeq \frac{1}{2}\|x-y\|^2$ can be interpreted as a halved squared distance, they induce a **Brenier map**:

$$x_i \;\mapsto\; x_i + v_i\,, \qquad \text{where} \qquad v_i \;=\; v(x_i) \;=\; -\nabla F(x_i) \;=\; -\tfrac{1}{\alpha_i}\nabla_{x_i}\mathrm{S}_\varepsilon(\alpha,\beta) \tag{4.17}$$

is encoded as an array $(v_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$. We understand this vector field as an optimal low-frequency mapping from α onto β and refer to Eq. (3.227) for a detailed computation. Its behaviour for varying values of ε is illustrated in Figures 3.25 and 3.26.

Coming soon: support for orientation and curvature. As of March 2020, the `GeomLoss` package provides support for the processing of point clouds in a vector space $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$. We now plan to package our research prototypes into well-documented routines to support:

1. The processing of **meshes and curves**, with a reference implementation of the varifold and normal cycle encodings.
2. An FFT-based implementation of Algorithms 3.5-3.6 on grids, for optimal performances on **density maps and volumes**.
3. A simple interface for **arbitrary cost functions** encoded as `KeOps` `LazyTensors` to let users work on spheres, hyperbolic spaces and others.

4.3 Applications

In practice, all types of shapes can be encoded as measures and processed using the tools of Chapters 2 and 3. We now present applications of our optimal transport methods in three different settings: shape registration, segmentation and interpolation.

4.3.1 Shape registration

Morphable models, deformable templates. Matching two shapes with each other is a fundamental yet surprisingly difficult problem. In practice, as discussed in Section 1.2 and Chapter 5, most researchers map a source shape A^* onto a target B by minimizing a cost function:

$$\text{Cost}(\theta) \;=\; \underbrace{\text{Reg}(\theta, A^*)}_{\text{regularization}} \;+\; \underbrace{\text{Loss}(\,\text{Morph}(\theta, A^*),\ \text{Measure}(B)\,)}_{\text{data fidelity}} \tag{4.18}$$

with respect to a vector θ that parameterizes the deformation of the source A^* into a model $A = \text{Morph}(\theta, A^*)$ that is close to the target B.

The regularization (Reg) and deformation (Morph) routines encode a domain-specific prior on admissible transformations. Depending on the context, we solve the related optimization problem using gradient descent, quasi-Newton schemes (Liu and Nocedal, 1989), Expectation-Maximization (Dempster et al., 1977; Myronenko and Song, 2010) or explicit linear solvers (Bookstein, 1991; Chui and Rangarajan, 2000).

An ideal run of Algorithm 4.1 is illustrated Figure 4.6. Going further, more refined methods tend to enforce symmetry with respect to A^* and B (Avants et al., 2008; Lorenzi et al., 2013) or use a neural network to bypass optimization loops and estimate quickly the optimal set of parameters θ (Yang et al., 2017; Krebs et al., 2017, 2019).

Reliable loss functions. Ultimately though, all methods rely on a positive loss function and its gradient to drive the model A towards the target B. As illustrated in Figures 4.6 and 4.7, we understand shape registration as a generalization of the particle flow problem of page 80.

In our experience, the main lessons of Section 3.2 still hold: compared with Hausdorff or kernel distances, transport-based loss functions induce cleaner gradients for a faster and safer convergence. As the packaging of our algorithms progressively comes to an end, we are now working on evaluating rigorously their impact for medical imaging and computer vision.

Algorithm 4.1: Model-based shape registration

Input: Source mesh, curve or segmentation map A^*, target B.
Parameters: Deformation model $\text{Morph}(\theta\,;\,A^*) \mapsto A$ (discussed Chapter 5),
Regularization and data attachment functionals: $\text{Reg}(\theta, A^*)$ and $\text{Loss}(\alpha, \beta)$.

```
1: function Model( $\theta$, $A^*$ )                       ▷ Deform the source and turn it into a measure.
2:     $(\alpha_i^*, x_i^*) \leftarrow \text{Measure}(A^*)$                   ▷ Turn the source into a measure.
3:     $(\alpha_i, x_i) \leftarrow \text{Measure}(\text{Morph}(\theta; A^*))$          ▷ Deformed measure.
4:     return $(\alpha_i^*, x_i)$ or $(\alpha_i, x_i)$              ▷ See the discussion of Section 4.2.1.

   ▷ Use gradient descent wrt. $\theta$ or any other solver:
5: $\theta_{A^*\to B} \leftarrow \arg\min_\theta\ \text{Reg}(\theta, A^*) + \text{Loss}(\text{Model}(\theta, A^*), \text{Measure}(B))$
6: return $\theta_{A^*\to B}$          ▷ Parameters for an affine deformation, spline coefficients, etc.
```

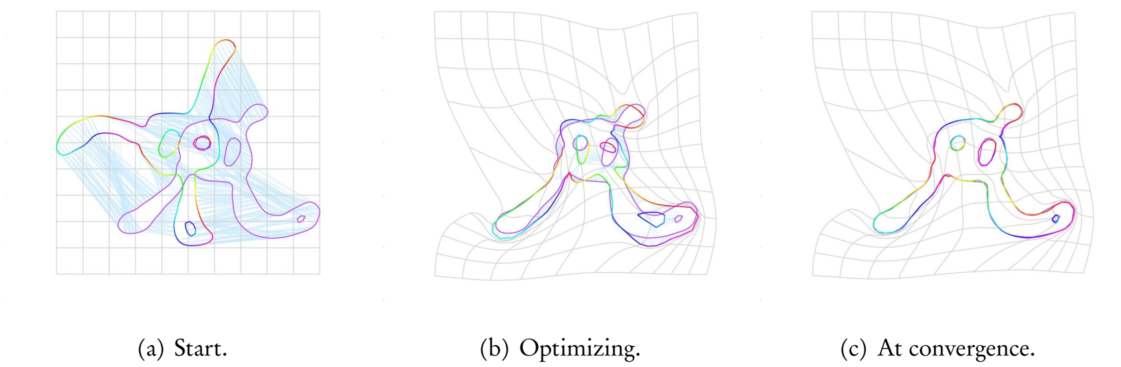

(a) Start. (b) Optimizing. (c) At convergence.

Figure 4.6: Using optimal transport to drive a diffeomorphic registration algorithm. The first motivation behind our work is the need for reliable loss functions in computational anatomy. Presented Chapter 3, Sinkhorn divergences can be interfaced with any of the deformation models discussed Chapter 5. The resulting pipelines are robust to large displacements – thanks to the global gradients provided by OT theory – and guarantee the preservation of the shapes' topologies.
(a) Let us consider the registration of a "rainbow" amoeba α onto a purple target β. The blue "spring system" between the source curve and its target represents the optimal transport plan associated to the Sinkhorn loss function $\mathrm{S}_\varepsilon(\alpha, \beta)$. It may tear apart some features of the curve, such as its bottom arm. (b) Fortunately, diffeomorphic registration models prevent our template from breaking down into pieces. As discussed page 81, they regularize the raw gradient field $v_i = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{S}_\varepsilon(\alpha, \beta)$ and project it onto a space of admissible deformations. (c) Minimizing the composite objective function of Eq. (4.18) allows us to converge towards a close overlap between the source and the target: our deformation model fully accounts for the geometric variability present in the data. Note that Theorem 3.1 ensures that (de-biased) Sinkhorn divergences define positive and definite loss functions: as discussed Section 3.3.2, we can now rely on entropic OT losses without having to worry about the entropic "shrinkage" bias.

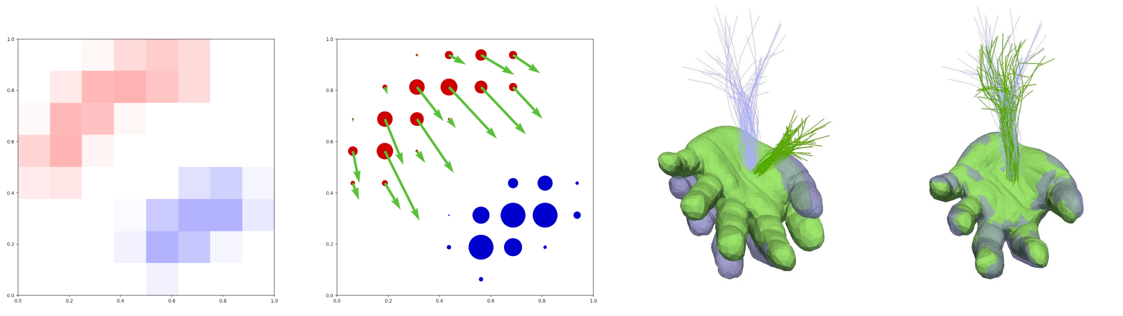

(a) Segmentation maps. (b) With gradient. (c) Start. (d) At convergence.

Figure 4.7: A versatile toolbox. (a, c) Geometric measure theory provides a coherent framework to deal with density maps, meshes and curves. (b) As detailed Eq. (3.85), the key ingredient behind most of our algorithms is the Lagrangian gradient field $v_i = -\frac{1}{\alpha_i}\partial_{x_i}\mathrm{Loss}(\alpha, \beta)$ of a geometric loss function with respect to the positions of the Dirac atoms that make up the source measure $\alpha = \sum \alpha_i \delta_{x_i}$. Following the discussion of Section 3.2.4, we generally favour an approximation of the squared Wasserstein-2 distance $\mathrm{Loss}(\alpha, \beta) = \mathrm{S}_\varepsilon(\alpha, \beta) \simeq \mathrm{OT}(\alpha, \beta)$ and display its gradient as a green velocity field. (c-d) Our loss functions are easy to interface with domain-specific pipelines. In this simple example, just as in Figure 4.6, a Sinkhorn divergence S_ε is used to drive a diffeomorphic registration pipeline. Fibers and triangles are handled simultaneously by the deformation module, but correspond to separate terms in the weighted loss function: $\mathrm{Loss}(\alpha, \beta) = \lambda_{\text{hand}}\,\mathrm{S}_\varepsilon(\alpha_{\text{hand}}, \beta_{\text{hand}}) + \lambda_{\text{fibers}}\,\mathrm{S}_\varepsilon(\alpha_{\text{fibers}}, \beta_{\text{fibers}})$.

4.3.2 Transfer of labels

Generalized assignment problems. More prosaically, we can use multiscale Sinkhorn solvers to provide solutions $(f_i) \in \mathbb{R}^{\mathrm{N}}$ and $(g_j) \in \mathbb{R}^{\mathrm{M}}$ for generalized assignment problems. As discussed page 115 and Eq. (4.15), these optimal dual vectors encode an implicit transport plan $(\pi_{i,j}) \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}_{\geqslant 0}$ that we understand as a soft matching between any two discrete measures $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ and $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$.

As illustrated Figure 4.8 and detailed in Algorithm 4.2, we can use such a transport plan π to transfer labels between distributions. At an affordable computational cost, OT-based algorithms enforce a (soft) constraint on the preservation of mass while taking into accound the geometry of the problem. They are best suited to the tracking of structured distributions over time, with successful applications to e.g. RNA-sequencing (Schiebinger et al., 2019).

Applications to tractography. In neuro-anatomy, we use this method to segment brain tractograms as follows:

1. As discussed page 145, we first encode a segmented atlas and a novel subject as discrete measures in a high-dimensional space of features $\mathcal{X}$.
2. We then use Algorithm 4.2 to transfer the source class labels (l_i) onto the subject. The blur and reach scales are anatomically meaningful: they should be of the order of the typical distances between two neighboring fibers and clusters, respectively.

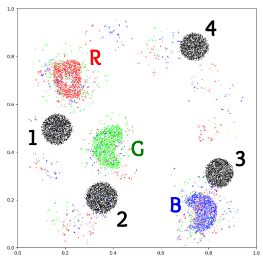
As illustrated Figure 4.9, recent progresses on multiscale OT solvers let us scale this procedure to millions of curves in minutes. This relatively affordable method could provide a simple way of dealing with tractograms, but our results are still preliminary: we are yet to assess the robustness of this method to large anatomical variations or the influence of our encoding on the quality of the results. As discussed for instance in (Bertò et al., 2020), we expect that the choice of domain-specific feature space $\mathcal{X}$ will have a sizeable impact on performances.

Algorithm 4.2: Transfer of labels with entropic, unbalanced OT

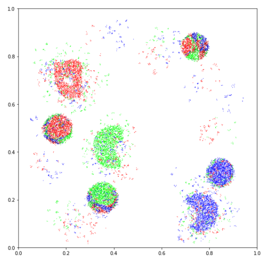
Input: Source distribution $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, labels $(l_i) \in [\![1,\mathrm{L}]\!]^{\mathrm{N}}$,
Target distribution $(y_j) \in \mathbb{R}^{\mathrm{M}\times\mathrm{D}}$,
Weights $(\alpha_i) \in \mathbb{R}^{\mathrm{N}}_{\geqslant 0}$ and $(\beta_j) \in \mathbb{R}^{\mathrm{M}}_{\geqslant 0}$ (use $\frac{1}{\mathrm{N}}$ and $\frac{1}{\mathrm{M}}$ as default).

Parameters: Cost function $\mathbf{C}(x,y)$ (use $\frac{1}{p}\|x-y\|^p$ with $p=2$ as default),
Blur scale $\sigma = \varepsilon^{1/p}$ (typically 1% to 10% of the configuration's diameter),
Maximum reach scale $r = \rho^{1/p}$ (set to $+\infty$ for balanced OT).

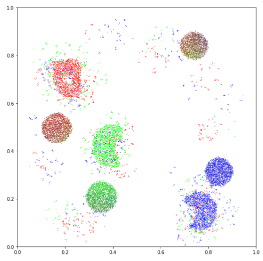
```
1: $f_i, g_j \leftarrow \mathrm{Solve}(\,\mathrm{OT}_{\varepsilon,\rho}(\sum_i \alpha_i\delta_{x_i}\,,\,\sum_j \beta_j\delta_{y_j}\,)\,)$    ▷ Use Alg. 3.5-6 without de-biasing.
2: $\pi_{i,j} \leftarrow \alpha_i\beta_j \cdot \exp \frac{1}{\varepsilon}[\,f_i + g_j - \mathbf{C}(x_i,y_j)\,]$    ▷ N-by-M transport plan, see Eq. (3.193).
3: $\ell_i \leftarrow \mathrm{OneHot}(\,l_i\,)$    ▷ $(\ell_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{L}}$, with $\ell_i[k] = 1$ if $k = l_i$, 0 otherwise.
4: $\ell_j \leftarrow \frac{1}{\beta_j}\sum_{i=1}^{\mathrm{N}} \pi_{i,j}\,\ell_i$    ▷ $(\ell_j) \in \mathbb{R}^{\mathrm{M}\times\mathrm{L}}$ – use KeOps for an efficient implementation!
5: $l_j \leftarrow \arg\max_{k\in[\![1,\mathrm{L}]\!]}\,\ell_j[\,k\,]$    ▷ Vector of labels, in $[\![1,\mathrm{L}]\!]^{\mathrm{M}}$.
6: return the labels $\ell_j \in \mathbb{R}^{\mathrm{L}}_{\geqslant 0}$ (soft) or $l_j \in [\![1,\mathrm{L}]\!]$ (hard).
```



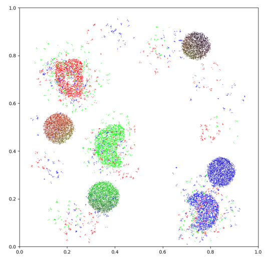
(a) Dataset.



(b) Optimal transport.

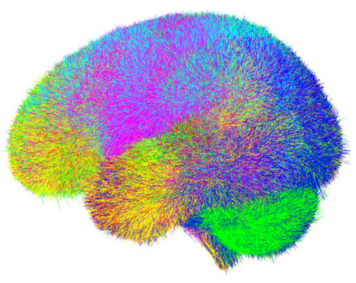


(c) Entropic transport.

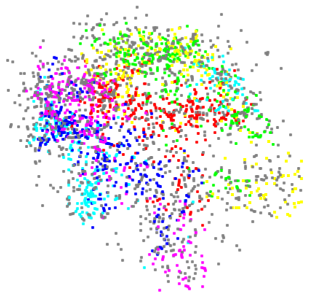


(d) Unbalanced transport.

Figure 4.8: Transfer of labels between two point clouds in the unit square.
(a) Let us consider a source point cloud (x_i) with $\{\text{red}, \text{green}, \text{blue}\}$ labels (l_i) and a target distribution of black points (y_j). We want to transfer labels from the x_i's onto the y_j's… but due to noise and outliers, nearest-neighbor projections may not produce satisfying results. What can we do? (b) A simple option is to compute the optimal transport plan $(\pi_{i,j})$ between the x_i's and the y_j's: we can then rely on the associated assignment to transfer the l_i's. In this example, we use a quadratic cost function $\mathbf{C}(x, y) = \frac{1}{2}\|x - y\|^2$ on the unit square $\mathcal{X} = [0, 1]^2$. The marginal constraints on $(\pi_{i,j})$ ensure that the assignment is one-to-one if the point clouds have the same size: they promote the non-trivial pairing (R-1, G-2) between the source and target clusters. Unfortunately, they also make the eventual labelling sensitive to noise (borders of 1, 2) and outliers (cluster 4). (c) As discussed Section 3.3.1, adding an entropic penalty to the OT problem is a simple way of smoothing the transport plan $(\pi_{i,j})$ and resulting labelling. (d) Going further, relaxing the marginal constraints is a simple way of discarding outliers. As discussed page 115, this can be done efficiently through the introduction of a dampening factor in the Sinkhorn iterations: these matchings were all computed in fractions of a second, using the multiscale Sinkhorn algorithm of page 123 without de-biasing.

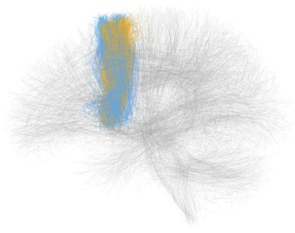


(a) Atlas.

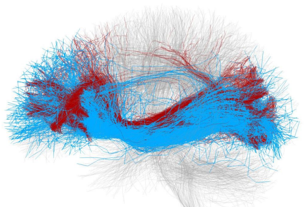


(b) As point clouds in $\mathbb{R}^{60}$.

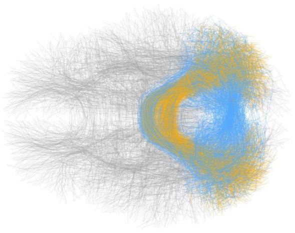
(c) Subject.



(d) Bundle 1.



(e) Bundle 2.



(f) Bundle 3.

Figure 4.9: Scaling up to brain tractograms with the `GeomLoss` routines. The OT-based method of Figure 4.8 can be applied to labeled (a) and raw (c) tractograms, encoded as large point clouds (b) in a high-dimensional feature space. Here, the source and target distributions are encoded using a flip-invariant parameterization of the fibers with 20 points per curve: $\mathrm{N} \simeq \mathrm{M} \simeq 800k$, $\mathrm{D} = 20 \times 3 = 60$. (d-f) White matter fibers are naturally clustered in anatomically meaningful bundles that link one area of the cortex to another. Thanks to this peculiar structure, which resembles that of the toy dataset of Figure 4.8, we can perform a reliable transfer of class labels from the atlas (a) to the subject (c) tractogram. Both tractograms come from the Human Connectome Project dataset (Van Essen et al., 2013) with a reference segmentation from (Zhang et al., 2018). Rendering done with `Paraview` (Ayachit, 2015).

4.3.3 Atlas construction

Wasserstein barycenters. We conclude our overview of OT theory in shape analysis with a fast scheme for geometric interpolation. If $(\beta_k)_{k\in[\![1,\mathrm{P}]\!]}$ is a collection of P probability measures and if $\lambda_1, \dots, \lambda_k \geqslant 0$ are interpolation weights that sum up to 1, the Wasserstein barycenter of the β_k's is defined as the solution of the optimization problem:

$$\text{Barycenter}(\lambda_k, \beta_k) ~=~ \arg \min_{\alpha \in \mathcal{M}_1^+(\mathcal{X})} \sum_{k=1}^{\mathrm{P}} \lambda_k \, \text{OT}(\alpha, \beta_k) ~, \tag{4.19}$$

where the OT loss is associated to a quadratic cost function $\mathbf{C}(x, y) ~=~ \frac{1}{2}\|x - y\|^2$ on the ambient space $\mathcal{X} = \mathbb{R}^{\mathrm{D}}$ (Agueh and Carlier, 2011). The existence and unicity of the barycenter derives from the strict convexity of the Wasserstein loss $\alpha \mapsto \text{OT}(\alpha, \beta_k)$.

As illustrated in Figures 3.14.b and 4.13, this problem generalizes barycentric interpolation to generic shapes and measures: in the degenerate case of atomic Dirac masses, we can show that for all weights (λ_k) and points $(x_k) \in \mathbb{R}^{\mathrm{P}\times\mathrm{D}}$, $\text{Barycenter}(\lambda_k, \delta_{x_k}) = \delta_{\sum_k \lambda_k x_k}$. Going further, as discussed in (Gerber et al., 2018) and Figure 4.14, this model can be used as a crude but affordable tool for exploratory shape analysis in e.g. neuro-anatomy.

Eulerian vs. Lagrangian schemes. Motivated by applications to shape analysis and machine learning, numerous schemes have been proposed to solve the optimization problem of Eq. (4.19). If the competitor $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$ is encoded using a vector of weights $(\alpha_i) \in \mathbb{R}_{\geqslant 0}^{\mathrm{N}}$ and a point cloud $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, possibly sampled on a grid, we can perform a Eulerian descent with respect to the α_i's (Solomon et al., 2015), optimize the x_i's with a Lagrangian method (Rabin et al., 2011) or use a mix of both strategies (Cuturi and Doucet, 2014).

The convexity of the OT loss guarantees the convergence of Eulerian schemes to the global optimum of Eq. (4.19). In practice though, as illustrated from Figure 4.10 to 4.13, these "pointwise" methods tend to be vastly outperformed by the aggressive Lagrangian iterations of Algorithm 4.3, which iterate over the linearization of the Wasserstein space discussed in (Mérigot et al., 2019) and usually converge in a handful of steps.

As illustrated Figure 4.12, this scheme may get trapped in poor local minima, but can also be combined with the `GeomLoss` routines to produces satisfying results in fractions of a second: we are currently working on bridging this gap between theory and practice.

Algorithm 4.3: Lagrangian solver for the Wasserstein barycenter problem

Input: Positive measures $(\beta_k)_{k\in[\![1,\mathrm{P}]\!]}$ on $\mathbb{R}^{\mathrm{D}}$ and weights $(\lambda_k)_{k\in[\![1,\mathrm{P}]\!]}$ that sum up to 1.

Parameters: Approximation `OT` of the squared Wasserstein-2 distance,
Reference measure α^* as initial guess (use $\sum_k \lambda_k \beta_k$ or a uniform law),
Number of discrete samples N (in the range $[10^4, 10^6]$ for typical problems).

```
1: $(\alpha_i, x_i) \leftarrow \text{Sample}(\alpha^*, \mathrm{N})$    ▷ $(\alpha_i) \in \mathbb{R}_{\geqslant 0}^{\mathrm{N}}$ and $(x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ with $\sum_i \alpha_i \delta_{x_i} \simeq \alpha^*$ .
2: for it = 1 to $n_{\text{its}}$ do    ▷ In practice, use 1 to 5 iterations.
3:     $x_i \leftarrow x_i - \frac{1}{\alpha_i} \sum_{k=1}^{\mathrm{P}} \lambda_k \, \partial_{x_i} \texttt{OT}(\sum_{j=1}^{\mathrm{N}} \alpha_j \delta_{x_j}, \beta_k)$    ▷ Wasserstein flow with $\delta t = 1$.
4: return $(\alpha_i, x_i)$    ▷ Encodes an approximation $\sum \alpha_i \delta_{x_i}$ of the barycenter.
```

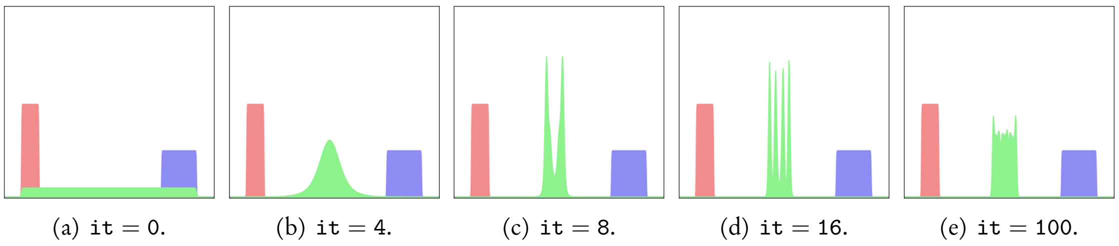

(a) $\texttt{it} = \texttt{0}$. (b) $\texttt{it} = \texttt{4}$. (c) $\texttt{it} = \texttt{8}$. (d) $\texttt{it} = \texttt{16}$. (e) $\texttt{it} = \texttt{100}$.

Figure 4.10: Computing a Wasserstein barycenter with a Eulerian scheme, in 1D. Given uniform densities β and γ on the intervals $[\texttt{0.0}, \texttt{0.1}]$ and $[\texttt{0.8}, \texttt{1.0}]$, we iteratively update the weights of a third probability measure α to minimize the objective function $\text{Bar}(\alpha) = \frac{1}{2}\text{S}_\varepsilon(\alpha, \beta) + \frac{1}{2}\text{S}_\varepsilon(\alpha, \gamma)$. Here, we use a quadratic cost function $\mathbf{C}(x, y) = \frac{1}{2}\|x - y\|^2$ and a negligible amount of entropic regularization: $\sigma = \sqrt{\varepsilon} = \texttt{.001}$. Following the conventions of Figure 3.7, the probability measures α (green), β (red) and γ (blue) are represented by their density with respect to the uniform Lebesgue measure on the unit interval $[\texttt{0}, \texttt{1}]$.
(b-e) As discussed around Eq. (4.19), the barycenter problem is convex with respect to the α_i's. Using a gradient-based descent algorithm on their logarithms (Liu and Nocedal, 1989), we eventually converge towards the uniform measure on $[\texttt{0.4}, \texttt{0.55}]$, solution of the problem. Unfortunately, this method converges slowly and induces Gibbs-like oscillations. To the best of our knowledge, such artifacts are produced by all Eulerian solvers of the barycenter problem: researchers usually smooth them out in a post-processing step.

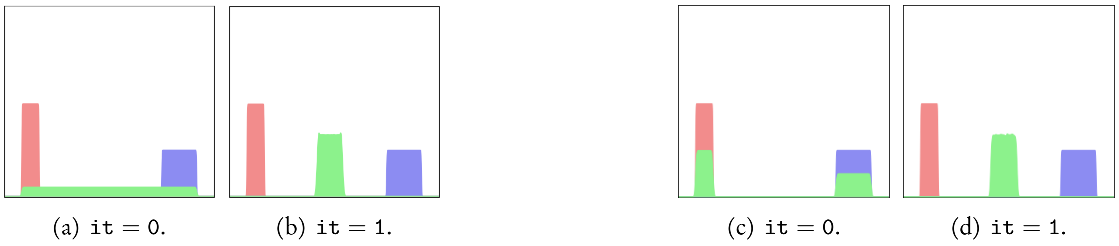
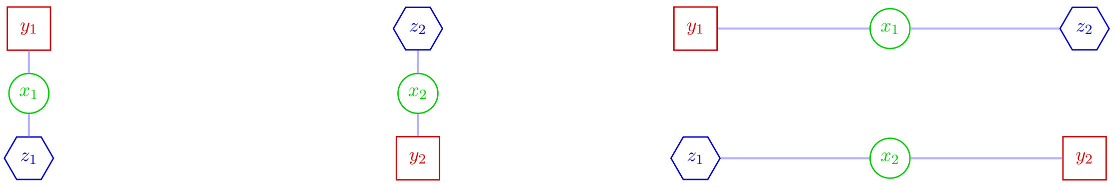
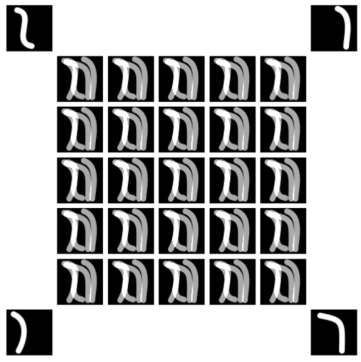

(a) $\texttt{it} = \texttt{0}$. (b) $\texttt{it} = \texttt{1}$. (c) $\texttt{it} = \texttt{0}$. (d) $\texttt{it} = \texttt{1}$.

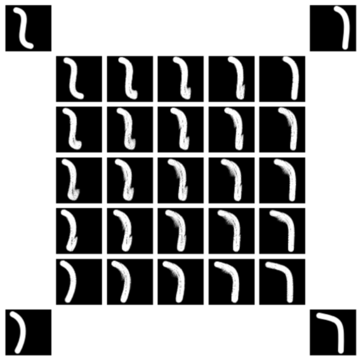
Figure 4.11: Computing a Wasserstein barycenter with a Lagrangian scheme, in 1D. The setting is the same as that of Figure 4.10. However, instead of updating the weights α_i of the competitor α, we directly move the samples x_i according to the update of Algorithm 4.3.
(a-b), (c-d) In dimension 1, optimal transport is equivalent to the equalization of histograms. The ordered structure of the real line, discussed page 100, makes straightforward the analysis of this Lagrangian scheme: it always converge to the optimum in one step, for any choice of the reference measure.



(a) Wasserstein barycenter. (b) Bad local minimum.

Figure 4.12: A counter-example for Algorithm 4.3, adapted from (Borgwardt, 2017). As illustrated Figure 4.11, the Lagrangian scheme of Algorithm 4.3 works flawlessly on the real line. In higher dimensions however, it may run into poor local minima: the Wasserstein barycenter problem is convex with respect to the weights α_i of the Dirac masses $\alpha_i\delta_{x_i}$, but *not* with respect to their positions x_i.
(a) In this 2D example, $\beta = \frac{1}{2}\delta_{y_1} + \frac{1}{2}\delta_{y_2}$ (red) and $\gamma = \frac{1}{2}\delta_{z_1} + \frac{1}{2}\delta_{z_2}$ (blue) are two probability measures on the plane, supported by the vertices of a rectangle. Their Wasserstein barycenter $\alpha = \frac{1}{2}\delta_{x_1} + \frac{1}{2}\delta_{x_2}$, with $x_i = \frac{1}{2}(y_i + z_i)$ is a stable minimum of our Lagrangian algorithm. (b) Unfortunately though, stable sub-optimal configurations can also be found as illustrated here with $x_1 = \frac{1}{2}(y_1 + z_2)$ and $x_2 = \frac{1}{2}(y_2 + z_1)$: the blue links represent the optimal transport plans $\pi_{\alpha\leftrightarrow\beta}$ and $\pi_{\alpha\leftrightarrow\gamma}$. This gridlock would hold even if the discrete Dirac masses had been replaced by continuous densities: non-convexity is an intrinsic feature of the Wasserstein barycenter problem, seen from a Lagrangian perspective.


(a) `it` = 0.


(b) `it` = 1.

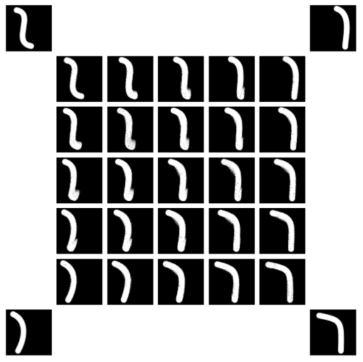
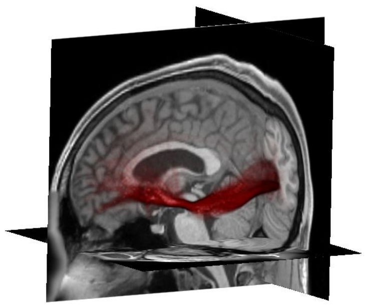
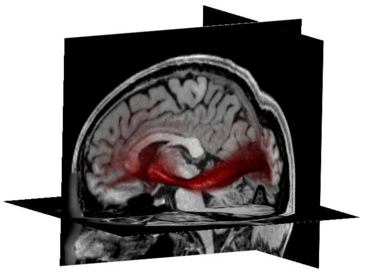
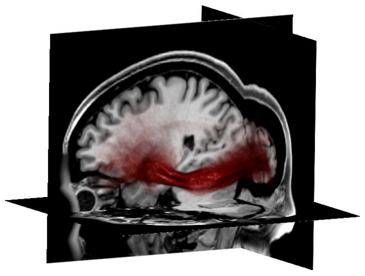
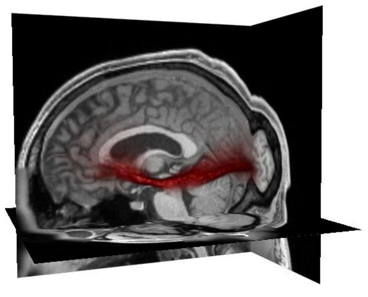

(c) `it` = 2.

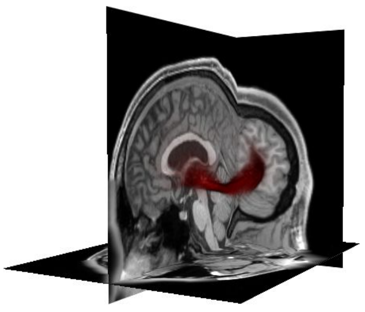
Figure 4.13: Computing Wasserstein barycenters with a Lagrangian scheme, in 2D.
(a) Starting from an average reference measure $\alpha^* = \frac{1}{4}\sum_{k=1}^{4}\beta_k$, (b-c) we represent the iterations of Algorithm 4.3 as we solve the Wasserstein barycenter problem for weights $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ that interpolate bi-linearly between $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ and $(0,0,0,1)$ at the four corners of our waffle plot. The setting is the same as that of Figure 3.14.b : we simply replaced the toy distributions A, B, C and D by worm-like shapes that are more relevant to medical applications.
Even if Algorithm 4.3 does not always converge to the solution of the barycenter problem, as highlighted Figure 4.12, it performs very well in practice: a sensible approximation of the Wasserstein barycenter between P measures can often be computed using no more than 2P calls to the efficient OT solvers presented Section 3.3.3.


(a) Subject 1.


(b) Subject 2.


(c) Subject 3.


(d) Barycenter, view 1.


(e) View 2.

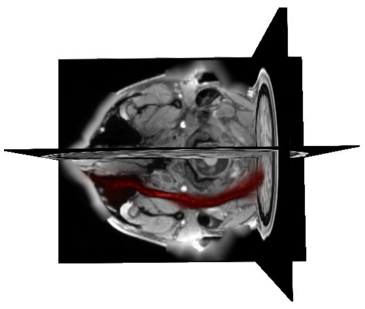

(f) View 3.

Figure 4.14: Computing Wasserstein barycenters in 3D. The efficient Lagrangian scheme of Algorithm 4.3 allows practitioners to compute geometric templates or interpolate between 3D shapes in fractions of a second – in this specific instance, segmentations of the inferior fronto-occipital fasciculus (IFOF) tract (Delmonte et al., 2019) that we process with `MRtrix3` (Tournier et al., 2019) and render with `3DSlicer` (Fedorov et al., 2012). This simple method has major limitations: as illustrated Figure 3.31.(g-h), OT matchings become unreliable when shapes differ too much from each other. Nevertheless, it could have its use for population analysis and machine learning with shapes, in the spirit of (Seguy and Cuturi, 2015; Ciliberto et al., 2016).

Chapter 5

Geometry on a space of anatomical shapes

This chapter relies on the notations of Sections 3.2.3 (kernel norms) and 4.3.1 (shape registration).

Key points – shape modelling is a hard problem:

1. Optimal transport routines do not model interactions between neighboring points and should be combined with **topology-aware** deformation models.
2. Endowing spaces of shapes with the **geometry induced by a registration algorithm** is an old idea that still defines the landscape in computer graphics and medical imaging. Affordable methods often rely on spline interpolations or stationary velocity fields. Going further, expressive Riemannian models specify small deformation costs and integrate them along geodesic trajectories.
3. Learning a metric structure on a space of shapes that is both **convenient** and **anatomically relevant** is the Holy Grail of computational anatomy. Modern approaches usually rely on convolutional neural networks or optimal control theory: they face deep structural and algorithmic challenges.

Contributions – let researchers focus on high-level questions, promote interactions:

4. First and foremost, this thesis attempts to **ease the computational burden** put on research teams that are already overwhelmed by clinical problems. With the `PyTorch+KeOps+GeomLoss` toolbox, implementing a state-of-the-art registration pipeline only requires a few lines of human-readable code. This is the path chosen by the Aramis INRIA team for the development of its reference `Deformetrica` software.
5. Enjoying the freedom provided by a compact, automatically differentiable codebase, we start to investigate methods that could **enforce meaningful axioms** on arbitrary (learned?) shape metrics. We use the symmetric Sinkhorn algorithm to ensure that **translations become geodesics** in a local and principled way: this surprising "normalization trick" allows us to fix one of the main weaknesses of the standard LDDMM framework.
6. In years to come, closer interactions with the computer graphics and deep learning communities are likely to bring relevant ideas to our field. By helping to trim down our codebases and putting together a **simple introduction to computational anatomy**, we hope to **bridge the gap** between cousin communities that have a lot to learn from each other.

Chapter 5 – Geometry on a space of anatomical shapes:

5.1 Affordable algebraic models

Going beyond optimal transport. A key problem in biomedical imaging is to extract geometric information out of our datasets. As discussed Section 1.2, this research can be understood as a quest for reliable shape metrics: we look for distances $\text{d}(A^*, B)$ between brain or heart images that are both affordable and relevant from a *medical* perspective.

Throughout Chapters 3 and 4, we used explicit metrics to compare weighted sets – or *measures* – with each other. The Hausdorff, kernel and Wasserstein distances induce geometric loss functions that are fully invariant to re-parameterizations and re-meshings. Unfortunately though, none of these formulas take into account the *topology* of the input data: as illustrated from Chapter 3 to Figure 5.1, they cannot be relied upon to handle large and complex deformations by themselves.

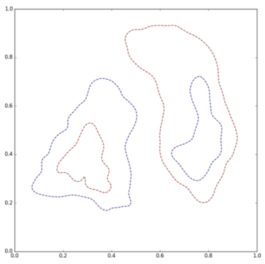
Building suitable deformation models. From the perspective of shape registration, discussed Section 4.3.1 and illustrated Figure 5.2, the Wasserstein-2 distance can be identified with a simple deformation module: the free-form model of Algorithm 5.1, with L^2 regularization of the velocity field. To overcome the limitations of optimal transport theory, a natural strategy is therefore to study structured deformation models and rely on the *stronger metrics* that they induce on spaces of shapes.

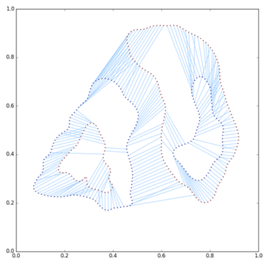
Throughout this chapter, we present deformation modules:

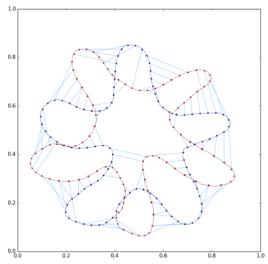
$$\text{Morph} : (\theta\,;\, A^*) \mapsto A \tag{5.1}$$

that enforce essential priors on shape variations: smoothness, preservation of the topology, etc. Computed using Algorithm 4.1 the parameters θ of our registrations can be understood as **latent codes** for the deformation $A^* \to A \simeq B$ between A^* and B.

This framework allows us to define shape metrics $\text{d}(A^*, B)$ as the sum of regularization terms $\text{Reg}(\theta, A^*)$ and data attachment penalties $\text{Loss}(A, B)$. In practice, most researchers devise *symmetric* penalties and efficient solvers for the registration problem of Eq. (4.18) (Klein et al., 2009a)... but we set these considerations aside in this high-level overview. Our goal here is to **bring geometric ideas to the fore** and open doors for cross-field interactions, without pretence to exhaustive treatment.


(a) Two measures.


(b) Optimal transport.


(c) Optimal transport.

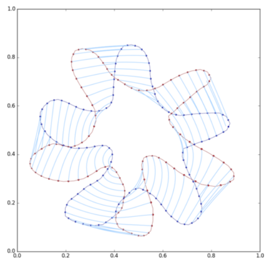
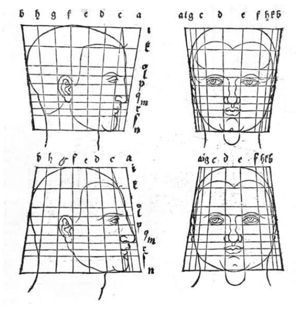
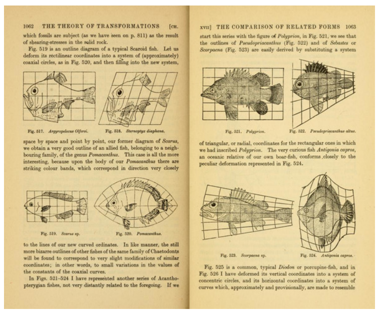
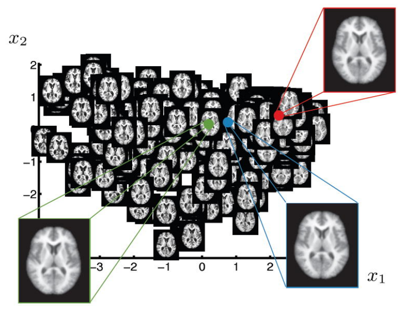

(d) Smooth registration.

Figure 5.1: Optimal transport is not the answer. (a) As discussed Chapter 4, generic shapes can be encoded as measures. (b) In order to match the red curve onto the blue one, a simple idea is to rely on optimal transport theory. Unfortunately though, as illustrated Figure 3.31, OT matchings tend to tear shapes apart. They handle curves and meshes as piles of dirt (Monge, 1781), with little regard for their topologies. (c) This limitation is most apparent in ambiguous configurations, such as the rotation of this star-shaped amoeba. (d) Chapter 5 introduces deformation models that enforce the preservation of topological features. This generally comes at the cost of the *convexity* of the related optimization problems: in order to retrieve satisfying rotations, our models must commit to one direction or another.


(a) Albrecht Dürer, 1528.


(b) D'Arcy Thompson, 1917.



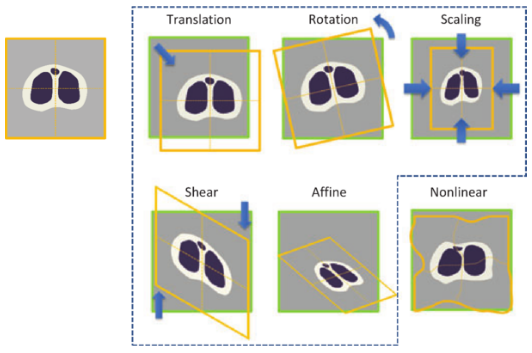
(c) Manifold of brain images.

Figure 5.2: Using deformations to study populations, from (Thompson, 1917; Gerber et al., 2010). Most anatomical shapes can be described as deformations of common templates. Up to ablations or metamorphoses that are handled separately, this generally holds for faces (a), fishes (b), brains (c) and other organs. Fortunately, relatively to raw biomedical images, deformations are easy to handle and study: fitting a deformation model to a population is a good way of analysing its modes of variation. As discussed Section 1.2, this research can be cast as a geometric problem: we look for relevant metrics on spaces of anatomical shapes, understood as sub-manifolds of a larger set of medical images.

Algorithm 5.1: Free-form deformation model

Input: Shape A^*, e.g. point cloud $(x_i^*) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and list of triangles $(\Delta_k) \in [\![1,\mathrm{N}]\!]^{\mathrm{P}\times 3}$.
Parameters: Vector field of displacements $v = (v_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.

```
1: function Morph( $\theta = (v_i)$ ; $A^* = (x_i^*, \Delta_k)$ )          ▷ Deform a shape $A^*$.
2:     $x_i \leftarrow x_i^* + v_i$                                 ▷ Free-form deformation with a vector field.
3:     return $A = (x_i, \Delta_k)$                        ▷ We move vertices but keep the same connectivity.

4: function Reg( $\theta = (v_i)$ )                               ▷ Simple $L^2$ regularization term.
5:     return $\frac{1}{2}\sum_{i=1}^{\mathrm{N}} \|v_i\|^2 = \frac{1}{2}\|v\|^2_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}}$      ▷ Squared Euclidean norm.
```



(a) Simple deformation models.

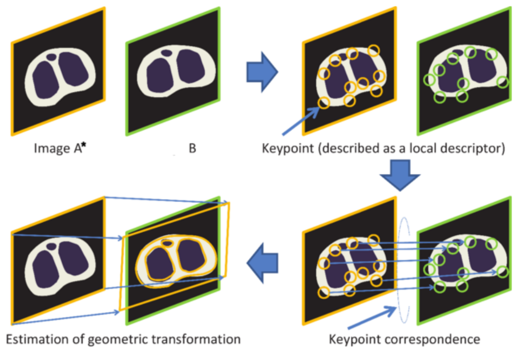


Image A*
B
Keypoint (described as a local descriptor)
Estimation of geometric transformation
Keypoint correspondence

(b) Register images with keypoints.

Figure 5.3: Affine registration models, from (Uchida, 2013).
(a) Affine deformations act on spatial coordinates to translate, rotate, rescale and distort shapes. These operations are often used to normalize acquisition parameters between different subjects.
(b) As discussed Section 1.2, this standard pre-processing is usually performed by identifying landmarks and solving a least square problem. Going further, the geometric loss functions presented in Chapter 4 allow researchers to align shapes without having to rely on pointwise correspondances.

5.1.1 Affine deformations

Parametric models. The simplest way of morphing an image is to translate, rescale and rotate its domain with a similarity of the ambient space $\mathbb{R}^{\mathrm{D}}$. Going further, the parametric model of Algorithm 5.2 allows us to apply general affine deformations. Illustrated Figure 5.3, affine registration is now a standard processing tool that puts shapes in a normalized system of coordinates.

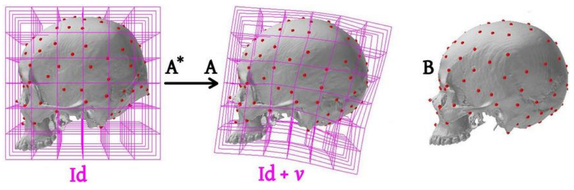
This parametric approach to shape registration can be pushed further: estimating an homography between two views of a 3D scene is a fundamental problem in computer vision (Agarwal et al., 2005), poly-affine deformations have become a staple of biomedical imaging (Arsigny et al., 2005, 2009), etc.

Explicit vs. implicit regularization. Unfortunately though *explicit* formulas can only grow so much before becoming illegible. In challenging settings, instead of designing an ad hoc parametric method, researchers thus tend to rely on expressive models with many degrees of freedom and regularize them using *implicit* constraints: smoothness, incompressibility, etc.

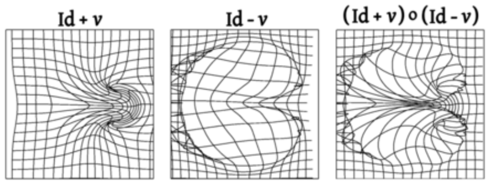
Algorithm 5.2: Affine deformation model

Input: Source shape A^*, encoded e.g. as a point cloud $(x_i^*) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and a list of triangles $(\Delta_k) \in [\![1,\mathrm{N}]\!]^{\mathrm{P}\times 3}$.
Parameters: Transformation matrix $M \in \mathbb{R}^{\mathrm{D}\times\mathrm{D}}$, offset vector $v \in \mathbb{R}^{\mathrm{D}}$.

```
1: function Morph( $\theta = (M, v)$ ; $A^* = (x_i^*, \Delta_k)$ )        ▷ Deform a shape $A^*$.
2:     $x_i \leftarrow M\, x_i^* + v$                                   ▷ Affine transform on the coordinates.
3:     return $A = (x_i, \Delta_k)$                                    ▷ Keep the same connectivity.
4: function Reg( $\theta = (M, v)$ )                                     ▷ Dummy regularization term.
5:     return 0                                                          ▷ Affine transforms are seldom penalized.
```



(a) Matching two skulls with each other.



(b) Folding patterns as we try to invert $\mathrm{Id} + v$.

Figure 5.4: Strengths and limitations of spline models, from (Ogihara et al., 2015; Ashburner, 2007). (a) A spline model finds a smooth deformation field v that brings the source shape A^* close to the target B without tearing it apart. This framework is ideally suited to the analysis of small geometric variations. (b) Unfortunately, spline deformations may also be hard or impossible to invert: large displacements can induce foldings and destroy the topological structure of the input data.

5.1.2 Spline registration

Smooth alignments. A common way of doing so it to rely on the computations of Section 3.2.3 to promote smoothness in Algorithm 5.1: we regularize the vector field $v_i = x_i - x_i^*$ from A^* to A with a **kernel metric** instead of a simple L^2 penalty.

If $k : \mathbb{R}^\mathrm{D} \times \mathbb{R}^\mathrm{D} \to \mathbb{R}$ is a positive, definite kernel that induces a generalized Sobolev norm $\|\cdot\|_k$ on the space of vector fields $v : \mathbb{R}^\mathrm{D} \to \mathbb{R}^\mathrm{D}$, we use:

$$\mathrm{Reg}(v_i) \overset{\text{def.}}{=} \min_{v:\mathbb{R}^\mathrm{D}\to\mathbb{R}^\mathrm{D}} \tfrac{1}{2}\|v\|_k^2 \qquad \text{s.t.} \quad \forall i,\ v(x_i^*) = v_i = x_i - x_i^* . \tag{5.2}$$

Working with kernels. As discussed around Eq. (3.137), the properties of kernel norms imply that there exists a set of vector coefficients $(p_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ such that the optimal field reads:

$$v = k \star \textstyle\sum_{i=1}^\mathrm{N} p_i \delta_{x_i^*} , \qquad \text{with} \qquad \|v\|_k^2 = \textstyle\sum_{i=1}^\mathrm{N}\sum_{j=1}^\mathrm{N} k(x_i^*, x_j^*)\, p_i^\top p_j . \tag{5.3}$$

We can thus parameterize our deformations by a vector field of *momenta* $(p_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and ensure the smoothness of $v_i = v(x_i^*)$ through a convolution with the kernel k. This method is detailed in Algorithm 5.3 and known under many different names, from "Gaussian Process regression" to "variational Horn-Schunck method" for the estimation of optical flow (Horn and Schunck, 1981). In practice, most researchers work with a **Thin Plate Spline** kernel to quotient out affine transforms (Bookstein, 1991), a compactly-supported **B-spline** kernel to retrieve fast $O(\mathrm{N})$ solvers (Rueckert et al., 1999) or local **Gaussian filters** to enforce coherence without long-range interactions (Myronenko and Song, 2010).

Algorithm 5.3: Spline deformation model

Input: Shape A^*, e.g. point cloud $(x_i^*) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and list of triangles $(\Delta_k) \in [\![1,\mathrm{N}]\!]^{\mathrm{P}\times 3}$.
Parameters: Vector coefficients $p = (p_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.
Hyper-parameters: Kernel function $k(x,y)$ – see Section 3.2.3.

```
1: function Morph( $\theta = (p_i)$ ; $A^* = (x_i^*, \Delta_k)$ )          ▷ Deform a shape $A^*$.
2:     $v_i \leftarrow \sum_{j=1}^\mathrm{N} k(x_i^*, x_j^*)\, p_j$          ▷ $v = k \star \sum_j p_j \delta_{x_j^*}$ ,  $(v_i) = (v(x_i^*)) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.
3:     $x_i \leftarrow x_i^* + v_i$          ▷ Free-form deformation with a $k$-smooth vector field.
4:     return $A = (x_i, \Delta_k)$          ▷ Keep the same connectivity.
5: function Reg( $\theta = (p_i)$ )          ▷ Kernel regularization term – see Eq. (3.99).
6:     return $\frac{1}{2}\sum_{i,j=1}^\mathrm{N} k(x_i^*, x_j^*) \cdot p_i^\top p_j = \frac{1}{2}\langle p\,,\, K_{x^*}\, p\rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}}$          ▷ Squared kernel norm.
```

5.1.3 Stationary velocity fields

Limitations of linear models. Alongside data-driven PCA metrics (Mahalanobis, 1936), spline models provide the reference baseline for shape analysis. Nevertheless, as discussed Section 1.2.2 and illustrated Figure 5.4, these simple methods often introduce folds and other artifacts when dealing with *large* deformations.

Diffeomorphic registration. To preserve the topology of our shapes, we should restrict ourselves to **diffeomorphic** variations and require that $x_i = \varphi(x_i^*)$ where $\varphi : \mathbb{R}^\mathrm{D} \rightarrow \mathbb{R}^\mathrm{D}$ is a smooth and invertible mapping with a smooth inverse. Diffeomorphisms φ that satisfy these axioms can be inverted and composed with each other: we say that they make up a *group* of deformations of $\mathbb{R}^\mathrm{D}$. This is most convenient from a theoretical perspective… but can we encode these applications on our machines?

Stationary velocity fields. A common way of doing so is to rely on the Picard–Lindelöf theorem for ordinary differential equations (Teschl, 2012). If $v : \mathbb{R}^\mathrm{D} \rightarrow \mathbb{R}^\mathrm{D}$ is a smooth vector field on the ambient space, we define the *Lie group exponential* $(\varphi^t)_{t\in\mathbb{R}} = (\exp(tv))_{t\in\mathbb{R}}$ through:

$$\varphi^0 = \mathrm{Id}\,, \;\text{ and }\; \tfrac{\mathrm{d}}{\mathrm{d}t}\varphi^t = v\circ\varphi^t\,, \;\text{ so that }\; \varphi^t\circ\varphi^{t'} = \varphi^{t+t'}\,. \tag{5.4}$$

In other words: (φ^t) is a one-parameter group of deformations whose derivative at time $t = 0$ is the vector field v. To evaluate the diffeomorphism $\varphi^1 = \exp(v)$ at any location x in $\mathbb{R}^\mathrm{D}$, we can let a particle flow along v and integrate the homogeneous differential equation:

$$x^{t=0} = x\,, \qquad\qquad \tfrac{\mathrm{d}}{\mathrm{d}t}x^t = v(x^t) \tag{5.5}$$

between times $t = 0$ and $t = 1$ using an Euler or Runge-Kutta scheme. Even better: we can leverage the stationarity of v through time and remark that if $\delta t > 0$ is small enough:

$$\exp(\delta t\cdot v) \simeq \mathrm{Id}+\delta t\cdot v \quad\text{ i.e. }\quad \forall\, x\in\mathbb{R}^\mathrm{D},\; \exp(\delta t\cdot v)(x) \simeq x+\delta t\cdot v(x)\,. \tag{5.6}$$

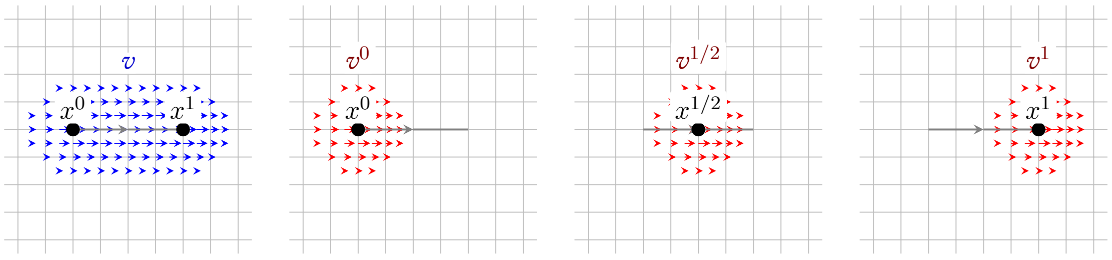
Then, relying on the scaling-and-squaring identity (Higham, 2005), we can approximate φ^1 as:

$$\exp(v) = \big(\exp(2^{-8}v)\big)^{2^8} = \big(\big(\exp(2^{-8}v)\big)^2\cdots\big)^2 \simeq \big((\mathrm{Id}+2^{-8}v)^2\cdots\big)^2\,. \tag{5.7}$$

Large diffeomorphisms can thus be evaluated by composing recursively with itself a small perturbation of the identity $(\mathrm{Id}+2^{-8}v) : x\mapsto x+2^{-8}\cdot v(x)$.

Strengths and limitations. The scaling-and-squaring trick paves the way for the fast generation of diffeomorphisms on images and volumes. We present a straightforward 2D scheme in Algorithm 5.4, and refer to (Arsigny et al., 2006; Ferraris et al., 2016) for efficient 2D and 3D implementations. Combined with the Baker-Campbell-Hausdorff (BCH) formula that allows us to approximate efficiently the gradient of the Lie group exponential (Bossa et al., 2007), these routines lie at the heart of the SVF framework that met widespread adoption for heart modelling (McLeod et al., 2011; Lombaert et al., 2014) and neuro-anatomy (Ashburner, 2007; Lorenzi et al., 2013) over the last decade.

Unfortunately though, the rigid algebraic structure of the Lie group exponential also has clear limitations. As illustrated Figure 5.5, restricting ourselves to the integration of *stationary* velocity fields prevents us from generating *all* deformations of the ambient space and extrapolating reliably beyond time $t = 1$ (Lorenzi and Pennec, 2013). To design models that handle *large* deformations well and generalize outside of training datasets, we must go further.



(a) SVF. (b) LDDMM, $t = 0$. (c) LDDMM, $t = .5$. (d) LDDMM, $t = 1.0$.

Figure 5.5: Stationary vs. time-varying velocity fields.
(a) We can use a smooth velocity field v defined on the 2D plane to move a particle between any two positions. In this example, a particle located at x^0 at time $t = 0$ flows along the blue velocity field v and arrives at $\exp(v)(x^0) = x^1$ at time $t = 1$. Assuming that v is smooth enough, the Picard–Lindelöf theorem ensures that the deformation $\varphi = \exp(v) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a diffeomorphism: it preserves all topological features and has a simple inverse, $\varphi^{-1} = \exp(-v)$. This framework is principled, robust and can be implemented efficiently. However, it also has two major drawbacks: extrapolation beyond time $t = 1$ may be meaningless, as v generally does not have support outside of a neighborhood of $[x^0, x^1]$; points' trajectories interact with each other between different time-steps and blend together in a way that is not anatomically relevant.
(b-d) In order to overcome the limitations of the Lie group exponential $v \mapsto \exp(v)$, we can build our deformations through the integration of a **time-varying** velocity field $(v^t)_{t\in[0,1]}$. This allows us to mitigate the weaknesses of the SVF framework and generate arbitrary smooth deformations, albeit at a higher computational cost. Choosing (v^t) according to a **least action principle** is then a sensible way of lowering the complexity of the method and retrieving sensible trajectories. Applications of this idea to shape registration have been studied extensively over the last two decades: we discuss the resulting "LDDMM" Riemannian framework in the remainder of this chapter.

Algorithm 5.4: SVF diffeomorphic deformation model (in 2D)

Input: Shape A^*, e.g. point cloud $(x_i^*) \in \mathbb{R}^{\mathrm{N}\times 2}$ and list of segments $(\Lambda_k) \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 2}$.
Parameters: Stationary velocity field v: a vector field over $[0, \mathrm{W}) \times [0, \mathrm{H})$, sampled at integer points and encoded as a volume $(v[\,x,\,y\,]) \in \mathbb{R}^{\mathrm{W}\times\mathrm{H}\times 2}$.

```
1:  function Square( $\varphi$ )                     ▷ Bi-linear squaring of a deformation field $\varphi \in \mathbb{R}^{W\times H\times 2}$.
2:      $X, Y \leftarrow \lfloor\varphi\rfloor$                                ▷ Integer coordinates $X, Y \in \mathbb{Z}^{W\times H}$.
3:      $s, t \leftarrow \varphi - \lfloor\varphi\rfloor$                            ▷ Residuals $s, t \in [0,1)^{W\times H}$.
4:      $\psi[x,y] \leftarrow (1-s)\cdot(1-t)\cdot\varphi[X[x,y], Y[x,y]] + s\cdot(1-t)\cdot\varphi[X[x,y]+1, Y[x,y]]$
                $+ (1-s)\cdot t\cdot\varphi[X[x,y], Y[x,y]+1] + s\cdot t\cdot\varphi[X[x,y]+1, Y[x,y]+1]$
5:      return $\psi$                   ▷ Encodes $\varphi\circ\varphi$ as a deformation field in $\mathbb{R}^{W\times H\times 2}$.

6:  function Morph( $\theta = v$; $A^* = (x_i^*, \Lambda_k)$ )                              ▷ Deform a shape $A^*$.
7:      $\varphi[x,y] \leftarrow (\mathrm{Id} + 2^{-8}\cdot v)(x,y) = (x,y) + \frac{1}{256}v[x,y]$             ▷ $\varphi \in \mathbb{R}^{W\times H\times 2}$.
8:      for n = 1 to 8 do                                                ▷ Compute $\varphi = \exp(v)$.
9:          $\varphi \leftarrow$ Square($\varphi$)                                        ▷ Scaling-and-squaring.
10:     $x_i \leftarrow$ Interpolate($\varphi, x_i^*$) $\simeq \varphi(x_i^*)$             ▷ Use e.g. the bilinear scheme of line 4.
11:     return $A = (x_i, \Lambda_k)$                                  ▷ Keep the same connectivity.

12: function Reg( $\theta = v$ )                      ▷ Use a kernel norm to enforce smoothness!
13:     return $\frac{1}{2WH}\sum_{x=0}^{W-1}\sum_{j=0}^{H-1}\|v[x,y]\|^2 = \frac{1}{2}\|v\|_{L^2}^2$              ▷ Squared $L^2$ norm.
```

5.2 Expressive Riemannian geometries

Looking for an expressive geometric framework. Classic models rely on **algebraic identities** to regularize matchings at an affordable computational cost: Algorithms 5.2 (Affine transform) to 5.4 (SVF exponential) provide reliable baselines for shape registration. They have been integrated as **transformer layers** in several deep learning libraries (Jaderberg et al., 2015; Niethammer et al., 2019b).

But crucially, in a medical setting, departing from algebraic recipes is a necessary first step to retrieve **flexibility**. In order to interpolate between two MRI volumes, our models should value medical insights over theoretical constraints: we must find ways of describing *generic* distances between shapes without losing too much on the computational front.

5.2.1 Riemannian geometry 101

A Riemannian metric is a field of local Euclidean metrics. An appealing way of doing so is to specify a Riemannian metric on a space of shapes $x = (x_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$. In other words: to use a collection of positive definite tensors $(g_x) \in \mathbb{R}^{\mathrm{ND}\times\mathrm{ND}}$ to **distort locally the Euclidean distance** on $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, and rely on the associated path length to encode a wide range of geometries with simple matrix computations.

Formally, a Riemannian metric on a shape space $\mathcal{S} \simeq \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is defined as an application:

$$g \;:\; (x, v) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \times \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \;\mapsto\; g_x v \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \tag{5.8}$$

that is smooth with respect to the shape x and linear with respect to the tangent velocity v. For all $x \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, we assume that the ND-by-ND matrix g_x is symmetric, positive and definite. The local **metric tensor** g_x induces, around every shape $x \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, a Euclidean norm:

$$\|v\|_x \;\overset{\text{def.}}{=}\; \sqrt{\langle\, v \,,\, g_x\, v \,\rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}}}\;. \tag{5.9}$$

The Riemannian length of any smooth curve $x : t \in [0,1] \mapsto x^t \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is then given by:

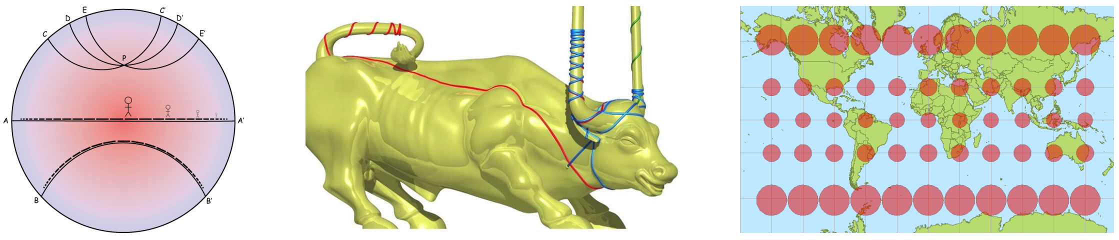
$$\text{Length}(x) \;\overset{\text{def.}}{=}\; \int_0^1 \|\dot{x}^t\|_{x^t}\,\mathrm{d}t \;\overset{\text{def.}}{=}\; \int_0^1 \sqrt{\langle\, \tfrac{\mathrm{d}}{\mathrm{d}t}x^t \,,\, g_{x^t}\, \tfrac{\mathrm{d}}{\mathrm{d}t}x^t \,\rangle}\,\mathrm{d}t\;, \tag{5.10}$$

and we define the Riemannian distance between any two shapes A^* and A in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ as the minimal path length associated to a **geodesic interpolation**:

$$\mathrm{d}(A^*, A) \;=\; \min_{(x^t)_{t\in[0,1]}} \;\text{Length}(x) \qquad \text{s.t.} \qquad x^0 \;=\; A^* \quad \text{and} \quad x^1 \;=\; A\,. \tag{5.11}$$

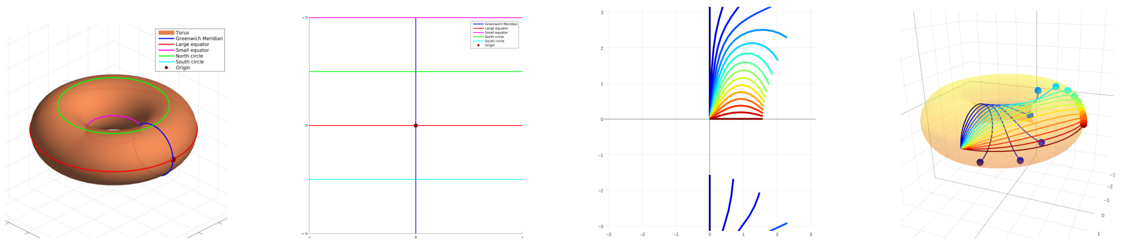
A convenient and versatile framework. As illustrated from Figure 5.6 to 5.8, Riemannian geometry encompasses all kinds of curvy models within a unified framework. Beyond the simple linear model of Eq. (1.6), local metrics (g_x) could allow us to represent "medical distances" in a space of brains or bones without having to rely on a reference template x^*.

But first of all, in order to work with Riemannian models, we must be able to solve the **geodesic interpolation problem** of Eq. (5.11) between any two shapes A^* and A. This is a challenging task: optimizing a non-convex functional on continuous paths is generally a hard problem. Fortunately though, Riemannian metrics retain enough structure to make this process tractable: we now present the theory behind the two most common algorithms for geodesic interpolation, the **mean curvature flow** and **Hamiltonian geodesic shooting**.



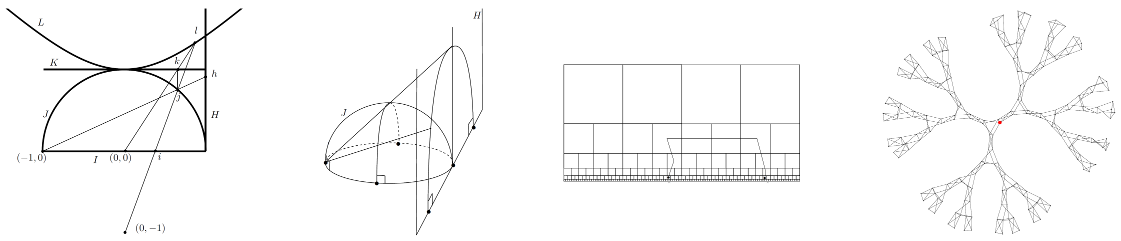
(a) The Poincaré disk. (b) Charging Bull. (c) Mercator projection.

Figure 5.6: Riemannian geometry, from (Norton, 2013; Leys et al., 2013; Kühn, 2019). (a) Riemannian geometry distorts the ambient space $\mathbb{R}^{\mathrm{D}}$ with a local operator $K_x = g_x^{-1}$ that is analogous to a "temperature": when it is high, objects expand and make large steps; when it is low, objects shrink and endure longer distances (Poincaré, 1902). (b) This idea is at the heart of the theory of relativity. It allows us to describe generic metric spaces in a unified framework, from 3D surfaces to high-dimensional spaces of anatomical shapes. (c) We can represent local variations of the metric tensor g_x by displaying unit balls for the local Euclidean norm $\|\cdot\|_x$. These ellipsoids are known as Tissot's indicatrices (Tissot, 1878): in cartography, they allow us to visualize the geometry induced by the spherical metric of $\mathbb{S}^2 \subset \mathbb{R}^3$ onto the 2D plane through the Mercator projection.



(a) Donut. (b) Flat torus. (c) Geodesics... (d) On a donut.

Figure 5.7: Riemannian geometry is a convenient way of dealing with curvature. (a) Encoding a torus as a 2D level set in a 3D space is cumbersome: projections are required to prevent points from leaving the donut-shaped surface. (b) The homogeneous encoding of the torus as a Cartesian product $\mathbb{S}^1 \times \mathbb{S}^1 \simeq \mathbb{R}^2/\mathbb{Z}^2$ with two angular coordinates is more convenient: two parameters are used to describe a two-dimensional surface. Unfortunately, this parameterization also discards the specific "donut" geometry of the 3D torus: it assigns an equal length to all tropics and equators. (c) Riemannian geometry allows us to get the best of both worlds: we can define a metric g_x on the periodic 2D model $\mathbb{R}^2/\mathbb{Z}^2$ (b) that endows it with the curved intrinsic geometry of (a) (Irons, 2005; Jantzen, 2012). Standard computations, detailed Eq. (5.22), then allow us to draw geodesic paths in the curved 2D space... which correspond exactly to the usual "straight lines" of the 3D model (d).



(a) Hyperbolic models. (b) Isometry. (c) Discrete model. (d) Hyperbolic graph.

Figure 5.8: A convenient model for arbitrary geometries, from (Cannon et al., 1997). Riemannian geometry allows us to study metric structures in a unified continuous framework. (a-b) The well-known Poincaré disk I can be identified with other models such as the Poincaré half-plane H, the hemisphere J or the hyperboloid L. Going further, it can be put in correspondence with discrete graphs (c) and groups (d) (Gromov, 1987): we display here a Cayley graph of the hyperbolic group $\mathrm{SL}_2(\mathbb{Z})$ in a neighborhood of the identity, rendered with `GraphViz` (Gansner and North, 2000). These sketches are to the hyperbolic plane what the discrete grid $\mathbb{Z}^2$ is to the Euclidean plane $\mathbb{R}^2$: continuous and discrete geometries interact seamlessly with each other.

Geodesic curves. Formally, a curve $(x^t)_{t\in[0,1]}$ is a **geodesic** between two shapes $x^0 = A^*$ and $x^1 = A$ in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ if it goes at constant speed and if it minimizes length locally. Otherwise said, if for all time $t \in [0,1]$, there exists a margin $\varepsilon > 0$ such that $(x^s)_{s\in[t-\varepsilon,t+\varepsilon]}$ is the shortest continuous path between $x^{t-\varepsilon}$ and $x^{t+\varepsilon}$ and

$$\|\dot{x}^t\|_{x^t} = \mathrm{Length}(x) \ . \tag{5.12}$$

As discussed Section 1.2.3, geodesic curves are **generalized straight lines**. Due to curvature and non-uniqueness, we cannot guarantee that all geodesics are shortest paths between their end points... But up to arc-length re-parameterization, all solutions of Eq. (5.11) are geodesics. Using the Cauchy-Schwarz inequality, we can show that geodesics are critical points of the energy functional:

$$\mathrm{Energy}(x) \ = \ \frac{1}{2}\int_0^1 \|\dot{x}^t\|^2_{x^t}\,\mathrm{d}t \ = \ \frac{1}{2}\int_0^1 \langle\, \dot{x}^t \,,\, g_{x^t}\dot{x}^t \,\rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}}\,\mathrm{d}t \ , \tag{5.13}$$

defined on the set of smooth curves of $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ that take the correct values A^* and A at times $t = 0$ and $t = 1$. We refer to (Lee, 2006, Chapter 6) for a detailed proof of this result, which is known as the **least action principle** in mechanics.

Theorem 5.1 (Euler–Lagrange equation)**.** *Building upon the characterization of Eq. (5.13), and under suitable regularity assumptions, we then show that geodesic curves are solutions of a second-order differential equation. If $v^t = \dot{x}^t = \frac{\mathrm{d}}{\mathrm{d}t}x^t \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ denotes its velocity at time t, a path (x^t) is a geodesic if and only if:*

$$\forall\, t \in [0,1], \ \ \tfrac{\mathrm{d}}{\mathrm{d}t}[g_{x^t}v^t] \ = \ +\tfrac{\partial}{\partial x}[E](x^t,v^t) \qquad \textit{with} \qquad E(x,v) \overset{\text{def.}}{=} \tfrac{1}{2}\langle v, g_x v\rangle \ . \tag{5.14}$$

Sketch of proof. Let $x : t \in [0,1] \mapsto x^t \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ be a geodesic curve between A^* and A. If $\delta x : t \in [0,1] \mapsto \delta x^t \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is a (small) perturbation of x with $\delta x^0 = \delta x^1 = 0$, we can expand the energy of the alternative path $x + \delta x$ between A^* and A as:

$$\mathrm{Energy}(x+\delta x) \ = \ \frac{1}{2}\int_0^1 \langle\, v^t + \delta v^t \,,\, g_{x^t+\delta x^t}(v^t+\delta v^t) \,\rangle\,\mathrm{d}t \tag{5.15}$$

$$\simeq \ \underbrace{\frac{1}{2}\int_0^1 \langle v^t, g_{x^t}v^t\rangle\,\mathrm{d}t}_{\mathrm{Energy}(x)} \ + \ \underbrace{\int_0^1 \langle \delta x^t, \tfrac{\partial}{\partial x}[E](x^t,v^t)\rangle\,\mathrm{d}t \ + \ \int_0^1 \langle \delta v^t, g_{x^t}v^t\rangle\,\mathrm{d}t}_{\mathrm{d\,Energy}(x)\cdot\delta x} \tag{5.16}$$

at order 1 with respect to δx. Since x is geodesic, it is a critical point of the energy: the linear term dEnergy$(x)\cdot\delta x$ must vanish. Otherwise said:

$$\int_0^1 \langle\, \delta x^t \,,\, \tfrac{\partial}{\partial x}[E](x^t,v^t) \,\rangle\,\mathrm{d}t \ + \ \int_0^1 \langle\, \delta\dot{x}^t \,,\, g_{x^t}v^t \,\rangle\,\mathrm{d}t \ = \ 0 \ . \tag{5.17}$$

As we know that $\delta x^0 = \delta x^1 = 0$, we then integrate by parts the second term and retrieve:

$$\int_0^1 \langle\, \delta x^t \,,\, \tfrac{\partial}{\partial x}[E](x^t,v^t) \ - \ \tfrac{\mathrm{d}}{\mathrm{d}t}[g_{x^t}v^t] \,\rangle\,\mathrm{d}t \ = \ 0 \ . \tag{5.18}$$

Since this holds for any admissible variation δx^t, we conclude with Eq. (5.14). □

Examples. On a flat Euclidean space endowed with a constant metric $g_x = \mathrm{Id}_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}}$, the Euler–Lagrange equation reads:

$$\ddot{x}^t \;=\; \dot{v}^t \;=\; \tfrac{\mathrm{d}}{\mathrm{d}t}[g_{x^t} v^t] \;=\; \tfrac{\partial}{\partial x}\big[\tfrac{1}{2}\|v^t\|^2\big] \;=\; 0\,. \tag{5.19}$$

We retrieve the well-known fact that the shortest path between A^* and A is the segment $x^t : t \in [0,1] \mapsto (1-t)A^* + tA$. Going further, explicit computations let us derive expressions for great circles on a sphere or for geodesics on a torus.

Hamilton's change of variables. In the general case, however, Eq. (5.14) is hard to deal with effectively. Expanding the left-hand term could allow us to write the Euler–Lagrange equation in coordinates $(x^t, \dot{x}^t, \ddot{x}^t)$, but computing the associated coefficients – the *Christoffel symbols* – leads to cumbersome and unstable derivations.

Fortunately though, just as in Section 5.1.2, a clever change of variables allows us to save the day. Instead of describing a trajectory with its position x^t and velocity $v^t = \dot{x}^t$, we rely on the **position-momentum** coordinates (q^t, p^t) in the phase space $(\mathbb{R}^{\mathrm{N}\times\mathrm{D}})^2$, defined as:

$$q \overset{\text{def.}}{=} x \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \quad \text{and} \quad p \overset{\text{def.}}{=} g_q v \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}, \quad \text{i.e.} \quad v \;=\; K_q p \quad \text{with} \quad K_q \overset{\text{def.}}{=} g_q^{-1}\,. \tag{5.20}$$

This parameterization was introduced by Hamilton in a successful bid to simplify the rules of Newtonian and Lagrangian mechanics (Hamilton, 1835). The fundamental operator K_q, inverse of the metric tensor $g_{x=q}$, is known as the **cometric**. Defined over the space of position-momentum pairs, the **Hamiltonian** H can be identified with the kinetic energy E and reads:

$$H(q,p) \overset{\text{def.}}{=} \tfrac{1}{2}\langle p, K_q p\rangle \;=\; \tfrac{1}{2}\langle g_q v, v\rangle \;=\; \tfrac{1}{2}\langle v, g_q v\rangle \;=\; E(q, K_q p)\,. \tag{5.21}$$

Theorem 5.2 (Geodesic equation)**.** *Using these notations, we show that a smooth curve* (x^t) *on* $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ *is geodesic if and only if the associated phase* $(q^t, p^t) = (x^t, g_{x^t}\dot{x}^t)$ *satisfies the coupled geodesic equation:*

$$\begin{cases} \tfrac{\mathrm{d}}{\mathrm{d}t} q^t \;=\; +\tfrac{\partial H}{\partial p}(q^t, p^t) & \textit{(follow the velocity } v^t = K_{q^t} p^t\textit{)} \\ \tfrac{\mathrm{d}}{\mathrm{d}t} p^t \;=\; -\tfrac{\partial H}{\partial q}(q^t, p^t) & \textit{(steer the momentum to stay on a geodesic path).} \end{cases} \tag{5.22}$$

Sketch of proof. According to the definitions of Eq. (5.20), we know that:

$$\tfrac{\mathrm{d}}{\mathrm{d}t} q^t \;=\; v^t \;=\; K_{q^t} p^t \;=\; \tfrac{\partial}{\partial p}[H](q^t, p^t)\,. \tag{5.23}$$

On the other hand, if we differentiate Eq. (5.21) with respect to q, we see that:

$$\tfrac{\partial}{\partial q} H(q,p) \;=\; \tfrac{\partial}{\partial x} E(q, K_q p) + \tfrac{\partial}{\partial q}[\langle p, K_q p\rangle](q,p) \;=\; \tfrac{\partial}{\partial x} E(q, K_q p) + 2\,\tfrac{\partial}{\partial q} H(q,p)\,. \tag{5.24}$$

If (x^t) is a geodesic curve, the Euler–Lagrange equation satisfied by (q^t, p^t) thus reads:

$$\tfrac{\mathrm{d}}{\mathrm{d}t} p^t \;=\; \tfrac{\mathrm{d}}{\mathrm{d}t}[g_{q^t} K_{q^t} p_t] \;=\; +\tfrac{\partial}{\partial x}[E](q^t, K_{q^t} p^t) \;=\; -\tfrac{\partial}{\partial q}[H](q^t, p^t)\,, \tag{5.25}$$

which allows us to conclude. □

Working in a Riemannian shape space. The clean, homogeneous structure of the geodesic Equation (5.22) in Hamiltonian coordinates is at the heart of modern mechanics. The configuration (q^t, p^t) flows along a smooth vector field, the *symplectic gradient* $\Lambda H = (+\partial_p H, -\partial_q H)$ that is identified with the gradient of the Hamiltonian $\nabla H = (\partial_q H, \partial_p H)$ up to a "rotation" by $90°$ in phase space.

Since ΛH is orthogonal to ∇H for any value of q and p, the Hamiltonian is preserved by the geodesic flow: if $(x^t) = (q^t)$ is a geodesic curve,

$$\forall t \in [0,1],\;\; H(q^t, p^t) \;=\; \tfrac{1}{2}\langle\, p^t \,,\, K_{q^t} p^t \,\rangle \;=\; \tfrac{1}{2}\langle\, v^t \,,\, g_{q^t} v^t \,\rangle \;=\; H(q^0, p^0)\,. \qquad (5.26)$$

This is coherent with Eq. (5.12) and allows us to compute the length of any geodesic curve as:

$$\text{Length}(q^t) \;=\; \sqrt{\langle\, p^0 \,,\, K_{q^0} p^0 \,\rangle}\,. \qquad (5.27)$$

The exp and log maps. By analogy with the complex exponential $\exp_1 : i\theta \in i\mathbb{R} \mapsto e^{i\theta} \in \mathbb{S}^1$ that wraps the real line onto the unit circle, we define the **Riemannian exponential map** at any point q^0 through:

$$\exp_{q^0} \;:\; p^0 \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \;\mapsto\; q^1 \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}\,, \qquad \text{where } (q^t, p^t)_{t\in[0,1]} \text{ follows Eq. (5.22).} \qquad (5.28)$$

This **interpretable decoder** turns any momentum $p^0 \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ into a shape q^1 by flowing along the geodesic equation. Unlike the Lie group exponential introduced Section 5.1.3, this operator relies on the (model-dependent) metric g_q instead of the (fixed) composition rule "$\circ$" to generate trajectories in shape space. It is as-faithful-as-possible to the metric g_q around the template q^0 and thus fairly reliable for statistical studies on small to medium deformations.

Remarkably, the exponential map at any location q^0 is invertible in a neighborhood of 0 (Lee, 2006, Lemma 5.10). If q is close enough to q^0, this allows us to define the **Riemannian logarithm** as the geometric encoder:

$$\log_{q^0} \;:\; q \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \;\mapsto\; p^0 \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \qquad \text{such that } \exp_{q^0}(p^0) = q\,. \qquad (5.29)$$

Effective algorithms. Given any two shapes A^* and A in $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, geodesic interpolation can thus be cast as a non-linear inverse problem: we look for a fast and reliable inverse $\log_{A^*} : A \mapsto p^0$ to the Riemannian exponential $\exp_{A^*} : p^0 \mapsto A$. Can we solve this problem efficiently? The characterization of geodesic paths as local minimizers of the energy – Eq. (5.13) – and as solutions of the geodesic equation – Eq. (5.22) – suggests two strategies:

1. **Mean curvature flow – using the metric g_x.** For some time-step $\delta t = 1/\mathrm{T} > 0$, we can work with a sampled curve $(x_i^t) = (x_i^0, x_i^{\delta t}, \dots, x_i^1) \in \mathbb{R}^{(\mathrm{T}+1)\times\mathrm{N}\times\mathrm{D}}$ and minimize:

$$\text{Energy}(x) \;\simeq\; \frac{1}{2}\sum_{t=0}^{1-\delta t} \delta t \,\cdot\, \|\tfrac{1}{\delta t}(x^{t+\delta t} - x^t)\|_{x^t}^2 \;=\; \frac{1}{2\mathrm{T}}\sum_{t=0}^{1-\delta t} \langle v^t, g_{x^t} v^t\rangle \qquad (5.30)$$

under the constraint that $x^0 = A^*$ and $x^1 = A \simeq B$, with $v^t \overset{\text{def.}}{=} \frac{1}{\delta t}(x^{t+\delta t} - x^t)$. The associated deformation model is detailed in Algorithm 5.5: descent schemes on the path energy $\text{Reg}(\theta\,; A^*)$ are often understood as curve-shortening **geometric flows** that straighten the path (x^t) with respect to the Riemannian metric (g_x) (Brandt et al., 2016).

2. **Geodesic shooting – using the cometric K_q.** Alternatively, we can optimize the *shooting momentum* p^0 so that the solution (q^t, p^t) of the geodesic equation with initial condition $(q^0 = A^*, p^0)$ gets as close as possible to the target at time $t = 1$: $q^1 = A \simeq B$.

   This approach is summarized by the Hamiltonian deformation model of Algorithm 5.6 and often studied from the perspective of optimal control theory (Arguillere et al., 2015). We understand it as a continuous-time generalization of the linear spline model of Algorithm 5.3, that corresponds to the case where $\delta t = 1$ and K_q is a kernel matrix.

Algorithm 5.5: Time-varying deformation model

Input: Shape A^*, e.g. point cloud $(x_i^*) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and list of triangles $(\Delta_k) \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 3}$.
Parameters: Velocities $(v_i^t) = ((v_i^0), (v_i^{\delta t}), (v_i^{2\delta t}), \dots, (v_i^{1-\delta t})) \in \mathbb{R}^{\mathrm{T}\times\mathrm{N}\times\mathrm{D}}$.
Hyper-parameters: Time step $\delta t = 1/\mathrm{T} > 0$ ($\delta t =$ `0.1` as default),
Shape metric $g : (x, v) \in (\mathbb{R}^{\mathrm{N}\times\mathrm{D}})^2 \mapsto g_x v \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, smooth wrt. x, linear wrt. v.

```
1: function Morph( $\theta = (v_i^t)$ ; $A^* = (x_i^*, \Delta_k)$ )          ▷ Deform a shape $A^*$.
2:     $x_i^0 \leftarrow x_i^*$                                              ▷ Starting configuration.
3:     for $t$ in $\{0, \delta t, \dots, 1 - \delta t\}$ do                 ▷ Simple Euler integrator, T timesteps.
4:         $x_i^{t+\delta t} \leftarrow x_i^t + \delta t \cdot v_i^t$        ▷ Trajectory, with $\dot{x}_i(t) = v^t(x_i(t))$.
5:     return $A = (x_i^1, \Delta_k)$                                         ▷ Keep the same connectivity as $A^*$.
6: function Reg( $\theta = (v_i^t)$ ; $A^* = (x_i^*, \Delta_k)$ )            ▷ Path energy, see Eq. (5.30).
7:     $(x_i^t) \leftarrow$ Trajectory( $x_i^*$ , $v_i^t$ )                  ▷ As above: $(x_i^t) \in \mathbb{R}^{\mathrm{T}\times\mathrm{N}\times\mathrm{D}}$.
8:     return $\frac{1}{2\mathrm{T}} \sum_t \langle v^t , g(x^t, v^t) \rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}} \simeq \frac{1}{2} \int_0^1 \langle v^t , g_{x^t} v^t \rangle \mathrm{d}t$
```

Algorithm 5.6: Hamiltonian deformation model

Input: Shape A^*, e.g. point cloud $(x_i^*) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and list of triangles $(\Delta_k) \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 3}$.
Parameters: Shooting momentum $(p_i^0) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.
Hyper-parameters: Time step $\delta t = 1/\mathrm{T} > 0$ ($\delta t =$ `0.1` as default),
Cometric $K : (q, p) \in (\mathbb{R}^{\mathrm{N}\times\mathrm{D}})^2 \mapsto K_q p \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, smooth wrt. q, linear wrt. p.

```
1: function Morph( $\theta = p^0$ ; $A^* = (x^*, \Delta_k)$ )                  ▷ Deform a shape $A^*$.
2:     $q, p \leftarrow x^*, p^0$                                 ▷ Starting configuration in phase space: $q, p \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.
3:     for $t$ in $\{0, \delta t, \dots, 1 - \delta t\}$ do                 ▷ Simple Euler integrator, T timesteps.
4:         $\dot{q} \leftarrow +\frac{\partial H}{\partial p}(q, p) = K_q p$          ▷ Velocity $v = \dot{q} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.
5:         $\dot{p} \leftarrow -\frac{\partial H}{\partial q}(q, p) = -\frac{1}{2}\frac{\partial}{\partial q}[\langle p, K_q p \rangle]$   ▷ Steer the momentum, $\dot{p} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$.
6:         $q, p \leftarrow q + \delta t \cdot \dot{q}, p + \delta t \cdot \dot{p}$     ▷ Geodesic trajectory, as in Eq. (5.22).
7:     return $A = (q, \Delta_k)$                                             ▷ Keep the same connectivity as $A^*$.
8: function Reg( $\theta = p^0$ ; $A^* = (x^*, \Delta_k)$ )                   ▷ Path energy, see Eq. (5.13).
9:     return $\frac{1}{2}\langle p^0 , K_{x^*} p^0 \rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}} = \frac{1}{2}\|p^0\|_{x^*}^2$   ▷ Squared length of the geodesic, Eq. (5.27).
```

5.2.2 Elastic metrics: thin shells and finite elements

The deformation routines of Algorithms 5.5 and 5.6 allow us to work with any Riemannian structure on a space of shapes $x = q \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$, encoded using either a metric (g_x) or a cometric (K_q). In practice, researchers combine these morphing models with efficient optimizers or neural networks to compute geodesic interpolations in fractions of a second (Brunn et al., 2019; Yang et al., 2017).

The underlying assumption here is that a well-chosen Riemannian metric will promote meaningful interpolations and induce relevant latent codes p^0 for statistical studies. With medical applications in mind, picking a **reliable shape metric** is therefore an important question: what are our options?

Elastic meshes. If the point cloud $x \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is the set of vertices of a wireframe mesh with connectivity $\Lambda \in [\![1, \mathrm{N}]\!]^{\mathrm{P}\times 2}$, a simple choice is to rely on the **graph Laplacian**:

$$g_x \overset{\text{def.}}{=} -\Delta_{\Lambda,w} \qquad \text{so that} \qquad \langle\, v \,,\, g_x v \,\rangle \overset{\text{def.}}{=} \textstyle\sum_{(i,j)\in\Lambda} w(\|x_i - x_j\|) \cdot \|v_i - v_j\|^2 \,, \tag{5.31}$$

where $w : \mathbb{R}_{\geqslant 0} \to \mathbb{R}_{\geqslant 0}$ is a non-negative weight function. This affordable (semi-definite) metric penalizes tearings along graph edges without putting any constraint on global translations.

Going further, more refined metrics can be derived from standard mechanical models: as detailed in (Bonet and Wood, 1997), a common option is to rely on hyper-elastic descriptions of surface or volumetric meshes. Alternatively, parameterization-invariant elastic metrics can be used as reliable baselines: we refer to (Michor and Mumford, 2006; Srivastava et al., 2010; Su et al., 2019) for an overview.

Overall, as illustrated in Figures 5.9 and 5.10, elastic metrics induce an intuitive geometry on spaces of shapes. But unfortunately, these **intrinsic** methods also rely on clean mesh structures that may be hard to specify in a clinical setting. Working with noisy images and segmentation masks, what should we do?

Morphing the ambient space. A sensible choice is to rely on **extrinsic** metrics that penalize deformations of the **ambient space** $\mathbb{R}^{\mathrm{D}}$. If k is a positive definite kernel associated to a generalized Sobolev norm $\|\cdot\|_k$, we use a metric (g_x) on $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ that is defined through:

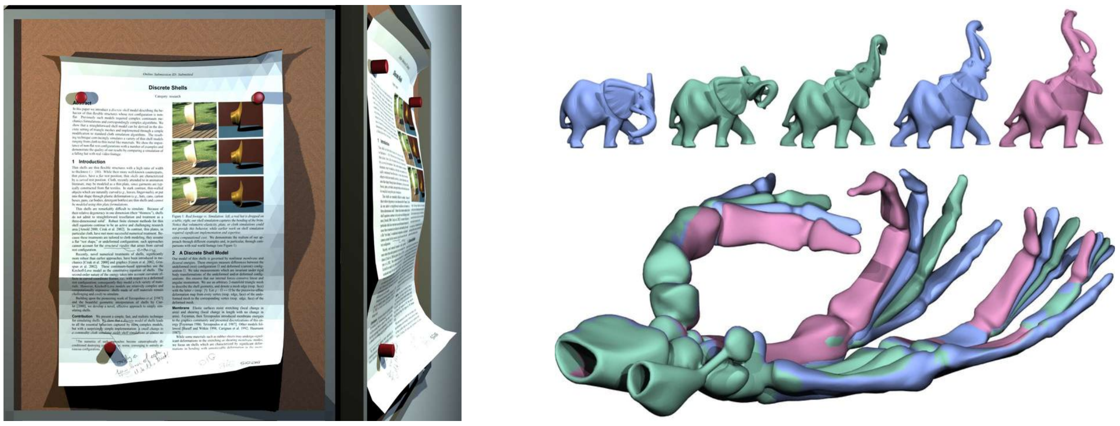
$$\forall\, v \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \quad \langle v, g_x v\rangle \overset{\text{def.}}{=} \min_{v:\mathbb{R}^{\mathrm{D}}\to\mathbb{R}^{\mathrm{D}}} \|v\|_k^2 \qquad \text{s.t.} \quad \forall i,\;\; v(x_i) = v_i \,. \tag{5.32}$$

As discussed in (Beg et al., 2005; Zhang and Fletcher, 2019; Brunn et al., 2019) and illustrated Figure 5.11, the associated interpolation problem can be solved efficiently on 2D images and 3D volumes. In practice, most implementations rely on kernel norms that are associated to the Laplacian on the ambient space $\mathbb{R}^{\mathrm{D}}$ and use a formula along the lines of:

$$\|v\|_k^2 \overset{\text{def.}}{=} \langle\, v \,,\, (\mathrm{Id} - \alpha\Delta)^2\, v \,\rangle_{L^2(\mathbb{R}^{\mathrm{D}},\mathbb{R}^{\mathrm{D}})} \tag{5.33}$$

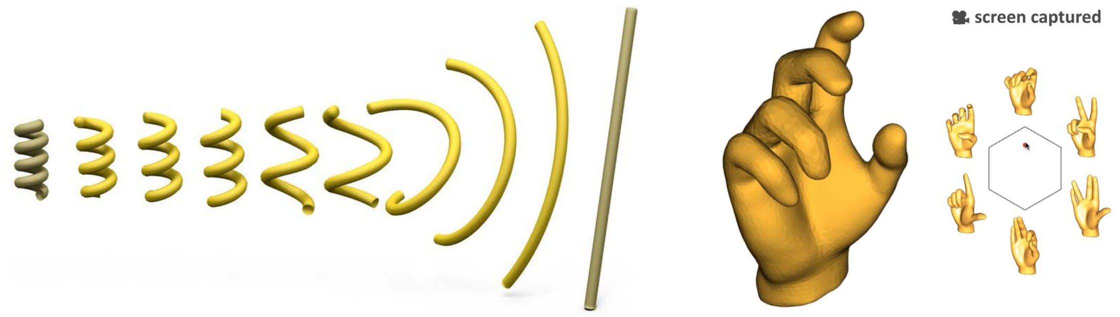
$$\overset{\text{def.}}{=} \textstyle\sum_{d=1}^{\mathrm{D}} \int_{x\in\mathbb{R}^{\mathrm{D}}} \|v_d(x)\|^2 \,+\, 2\alpha\|\nabla v_d(x)\|^2 \,+\, \alpha^2\|\Delta v_d(x)\|^2 \,\mathrm{d}x \tag{5.34}$$

where $v_1, \dots, v_{\mathrm{D}}$ denote the coordinates of the vector field $v : \mathbb{R}^{\mathrm{D}} \to \mathbb{R}^{\mathrm{D}}$ and $\alpha > 0$ is a parameter that controls the smoothness of v over $\mathbb{R}^{\mathrm{D}} = \mathbb{R}^2$ or $\mathbb{R}^3$. The associated geodesic equation can then be cast as a PDE inspired by fluid mechanics (Holm and Marsden, 2005) and solved using a mean curvature flow or a specific multiscale algorithm. Overall, this Riemannian theory allows researchers to generate large diffeomorphisms without being constrained by the properties of the Lie group exponential discussed Section 5.1.3: it is known in the literature as the Large Deformation Diffeomorphic Metric Mapping (LDDMM) framework.



(a) Discrete shells. (b) Geodesics in shape space.

Figure 5.9: Elastic shape metrics, from (Grinspun et al., 2003; Kilian et al., 2007). Over the last two decades, motivated by applications to game design and 3D graphics, affordable approximations of mechanical models have been cast as metrics on spaces of shapes: simple baselines are provided by thin shells (a) and "gummy" finite element models (b). These elastic metrics allow us to interpolate (green) and extrapolate (red) realistically between any two key frames encoded as 3D meshes (blue).



(a) Elastic geodesic. (b) Barycenter in shape space.

Figure 5.10: Optimized numerical schemes (Von-Tycowicz et al., 2015; von Radziewsky et al., 2016). Modern implementations of elastic metrics let artists solve geodesic (a) and barycentric (b) interpolation problems in real-time. Unlike OT-based algorithms, these methods guarantee the preservation of the shapes' topologies along deformations: they can handle convincingly some *extreme* non-linearities. Unfortunately though, mesh-based methods often rely on clean segmentations, pointwise correspondences and clever pre-computations to produce these impressive results. A key challenge for computational anatomy is to mitigate these constraints to combine biomechanical models with noisy clinical data.

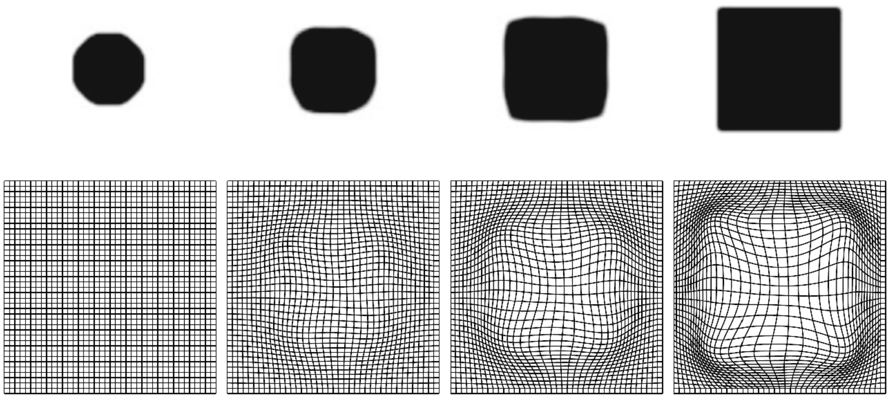


Figure 5.11: LDDMM geodesic between a disk and a square, from (Younes, 2010). When no "intrinsic" model of the data is available, shapes can still be registered onto one another by morphing the **ambient space** itself. In this example, a disk (left) is mapped onto a square (right) through the action of a diffeomorphism on the unit square (bottom row).

5.2.3 Kernel cometrics: the LDDMM framework

Kernel cometrics. Extrinsic Sobolev metrics along the lines of Eq. (5.32) are often implemented on a grid discretization of the spatial domain: we rely on convolution filters or Fourier transforms to compute metrics $\|v\|_k$ in the mould of Eq. (5.34).

Remarkably though, the Riemannian metric on point clouds defined Eq. (5.32) can also be written in closed form. Following the derivations of Eqs. (3.137-3.142), we can show that for any kernel function k:

$$\langle\, v \,,\, K_x^{-1} v \,\rangle \;=\; \min_{v:\mathbb{R}^{\mathrm{D}}\to\mathbb{R}^{\mathrm{D}}} \|\, v \,\|_k^2 \qquad \text{s.t.}\;\; \forall i,\;\; v(x_i) = v_i \;, \tag{5.35}$$

where $K_x \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is the **kernel matrix** associated to the point cloud $x \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$:

$$(K_x)_{i,j} \;\overset{\text{def.}}{=}\; k(x_i, x_j)\;, \qquad \text{i.e.} \qquad (K_x p)_i \;=\; \sum_{j=1}^{\mathrm{N}} k(x_i, x_j)\, p_j\;. \tag{5.36}$$

In other words: the metric tensor induced by an extrinsic Sobolev norm $\|\cdot\|_k$ on a point cloud (x_i) is exactly the inverse of the kernel matrix $K_x = (k(x_i, x_j))_{i,j}$. For any suitable choice of the symmetric, continuous, positive and definite kernel k, LDDMM geodesics can thus be generated with Algorithm 5.6, using the simple kernel matrix $K_q = (k(q_i, q_j))_{i,j}$ as a Riemannian cometric. Going further, anisotropic and inhomogeneous norms can also fit in this framework and induce e.g. incompressible deformations (Micheli and Glaunès, 2014).

First intuitions. Geodesic shooting with a kernel cometric is illustrated Figure 5.12: this algorithm allows us to generate large deformations without ever tearing apart our shapes. But how should we understand the LDDMM metric $g_x = K_x^{-1}$?

To get some intuition of its properties, we first work in the plane $\mathbb{R}^{\mathrm{D}} = \mathbb{R}^2$ and consider x_1, x_2 and x_3 the vertices of an equilateral triangle whose sides have length ℓ. If k is, say, a Gaussian kernel of deviation σ, the kernel matrix $K_x = (k(x_i, x_j))_{i,j}$ reads:

$$\begin{aligned} K_x \;&=\; \begin{pmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{pmatrix}\;, \qquad \text{with}\;\; k(0) = 1 \;\;\text{and}\;\; k(\,\ell\,) = e^{-\ell^2/2\sigma^2} = a > 0\,. & (5.37) \\ &=\; (1 + (3-1)a)\,\mathbf{u}\mathbf{u}^\top \;+\; (1-a)(\mathrm{Id}_{\mathbb{R}^3} - \mathbf{u}\mathbf{u}^\top) & (5.38) \end{aligned}$$

where $\mathbf{u} = \mathbf{1}/\sqrt{3}$ is a unit-norm vector, colinear to the vector $\mathbf{1} = (1,1,1)$. Taking advantage of this decomposition of K_x into orthogonal components, we invert it explicitely:

$$g_x \;=\; K_x^{-1} \;=\; \frac{1}{1+2a}\,\mathbf{u}\mathbf{u}^\top \;+\; \frac{1}{1-a}(\mathrm{Id}_{\mathbb{R}^3} - \mathbf{u}\mathbf{u}^\top)\;. \tag{5.39}$$

If $v = (v_1, v_2, v_3) \in \mathbb{R}^{3\times 2}$ is an infinitesimal deformation of the point cloud $x = (x_1, x_2, x_3)$, illustrated Figure 5.13, we can thus compute its Riemannian squared norm as:

$$\|v\|_x^2 \;=\; \langle v, g_x v\rangle \;=\; \frac{3}{1+2a}\|v^{\mathrm{mean}}\|^2 \;+\; \frac{1}{1-a}\sum_{i=1}^{3}\|v_i^{\mathrm{var}}\|^2\;, \tag{5.40}$$

where the orthogonal decomposition $v_i = v^{\mathrm{mean}} + v_i^{\mathrm{var}}$ is computed as:

$$v^{\mathrm{mean}} \;\overset{\text{def.}}{=}\; \tfrac{1}{3}(v_1 + v_2 + v_3) \in \mathbb{R}^2 \qquad \text{and} \qquad v_i^{\mathrm{var}} \;\overset{\text{def.}}{=}\; v_i - v^{\mathrm{mean}}\;. \tag{5.41}$$

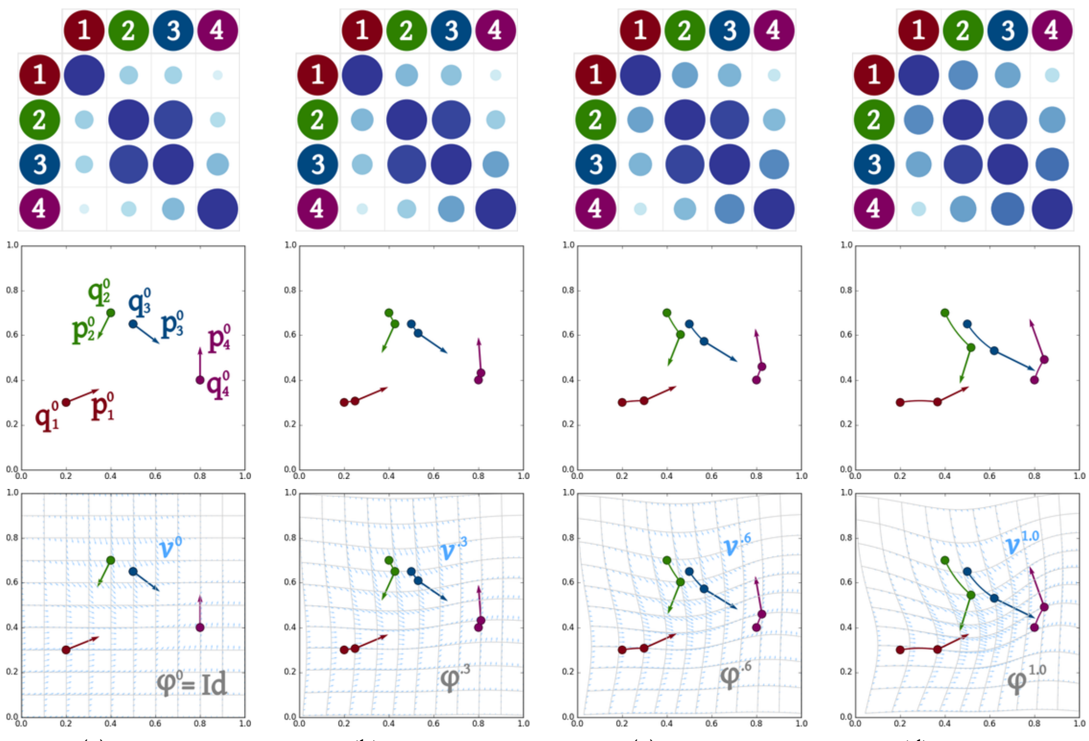

(a) $t = 0$. (b) $t = .3$. (c) $t = .6$. (d) $t = 1.0$.

Figure 5.12: Geodesic shooting with a kernel cometric – Algorithm 5.6. We work with $\mathrm{N} = 4$ points in the plane of dimension $\mathrm{D} = 2$. (a) We start from an initial point cloud $(q_i^0) \in \mathbb{R}^{4\times 2}$ and shooting momentum $(p_i^0) \in \mathbb{R}^{4\times 2}$ to display the geodesic trajectory $(q^t, p^t)_{t\in[0,1]}$ as we flow along Eq. (5.22) (b-d). (top row) Our cometric K_{q^t} is a 4-by-4 kernel matrix $(k(q_i^t, q_j^t))$ associated to a Gaussian kernel of deviation $\sigma = 0.5$. It correlates the trajectories of the points q_2 and q_3, which stay close to each other throughout the process. (middle row) The velocity $v^t = \dot{q}^t = K_{q^t} p^t$ is computed as the k-smoothing of the vector field $(p_i^t) \in \mathbb{R}^{4\times 2}$ supported by the q_i^t's. (bottom row) Theorem 5.3 allows us to see the geodesic trajectory $(q^t)_{t\in[0,1]}$ as the image of the initial point cloud (q_i^0) under the action of a time-varying velocity field $v^t : \mathbb{R}^2 \to \mathbb{R}^2$.

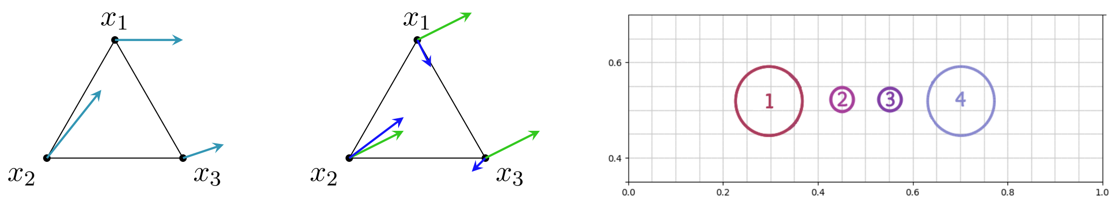

(a) Cloud x, velocity v. (b) $v = v^{\text{mean}} + v^{\text{var}}$. (c) Carpooling artifact.

Figure 5.13: Understanding kernel cometrics with toy configurations. (a) When the point cloud $(x_i) \in \mathbb{R}^{3\times 2}$ is an equilateral triangle, explicit computations allow us to invert the kernel matrix $K_x = (k(x_i, x_j))$ and make sense out of Eq. (5.35). (b) As detailed Eq. (5.40), the LDDMM metric $g_x = K_x^{-1}$ penalizes separately the group velocity v^{mean} (green) and the individual perturbations v^{var} (blue). As the three points x_1, x_2 and x_3 get closer to each other, they start behaving as a **single particle of mass 1**. (c) The **extrinsic** geometry induced by kernel cometrics has a surprising consequence: in LDDMM shape spaces, assuming that we use a typical "bell-shaped" kernel, the Riemannian geodesic that joins two translated copies of the same disk (1,4) is *not* a translation. Instead of sliding the red curve (1) sideways to match its blue target (4), the least action model saves up on "spatial deformation costs" by shrinking the shape (1→2), translating the small package (2→3) and expanding it back to its initial size (3→4).

If the side length ℓ is much larger than the kernel radius σ, then $k(\ell) = a \to 0$ and the kernel metric behaves as a simple L^2 norm:

$$\|v\|^2_{(x_1,x_2,x_3)} \simeq \|v_1\|^2 + \|v_1\|^2 + \|v_3\|^2 \,. \tag{5.42}$$

On the other hand, if ℓ is smaller than the kernel radius σ, then $k(\ell) = a \to 1$ and the kernel norm starts penalizing individual variations. At the limit, when $k(\ell) = a \simeq 1$, our three particles are only allowed to move as a single body with a uniform group velocity.

Rigorous results: the LDDMM framework. This behaviour is coherent with Eq. (5.32): LDDMM metrics penalize deformations of the point cloud x as the effort made by a virtual agent that distorts the ambient space $\mathbb{R}^\mathrm{D}$. The local density of points has no influence on the final cost, which only depends of the smoothness and magnitude of the displacement field:

$$v = k \star \sum_{i=1}^{\mathrm{N}} p_i \delta_{x_i} \,:\, x \in \mathbb{R}^\mathrm{D} \mapsto v(x) = \sum_{i=1}^{\mathrm{N}} k(x, x_i)\, p_i \in \mathbb{R}^\mathrm{D} \,, \tag{5.43}$$

solution of the kernel interpolation problem of Eq. (5.32). More generally, all discrete trajectories fit in a unified continuous framework with the following result, whose proof is detailed in (Beg et al., 2005; Arguillere et al., 2015):

Theorem 5.3 (Structure of the LDDMM geodesics)**.** *Let $k : \mathbb{R}^\mathrm{D} \times \mathbb{R}^\mathrm{D} \to \mathbb{R}$ be a smooth, symmetric, positive and definite kernel that is associated to a generalized Sobolev norm $\|\cdot\|_k$. For any number of points* $\mathrm{N} > 0$*, we consider the discrete space of point clouds* $\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ *endowed with the Riemannian metric:*

$$g_x = K_x^{-1} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}} \qquad \textit{where} \qquad (K_x)_{i,j} = k(x_i, x_j) \,. \tag{5.44}$$

On the other hand, we consider the continuous group of diffeomorphisms:

$$G_k \overset{\text{def.}}{=} \{\ \varphi = \varphi^1 : \mathbb{R}^\mathrm{D} \to \mathbb{R}^\mathrm{D} \ \textit{solution for a trajectory } (v^t)_{t\in[0,1]} \textit{ of the ODE} \tag{5.45}$$
$$\varphi^0(x) = x \,,\ \tfrac{\mathrm{d}}{\mathrm{d}t}\varphi^t(x) = v^t \circ \varphi^t(x) \,,\ \textit{subject to} \int_0^1 \|\, v^t \,\|_k^2 \,\mathrm{d}t < +\infty \ \}$$

endowed with the infinite-dimensional Riemannian metric:

$$\mathrm{d}^2(\varphi, \varphi + \delta\varphi) \simeq \langle\, \delta\varphi \,,\, g_\varphi\, \delta\varphi \,\rangle \overset{\text{def.}}{=} \|\, \delta\varphi \circ \varphi^{-1} \|_k^2 \simeq \|\, v \,\|_k^2 \tag{5.46}$$

that penalizes infinitesimal deformation using the extrinsic Sobolev norm $\|\cdot\|_k$. We can show that (G_k, g_φ) is a complete, geodesic Riemannian manifold of diffeomorphisms that are generated through the integration of a k-smooth time-varying velocity field v^t.

Then, most remarkably, ***all discrete geodesics in*** $(\mathbb{R}^{\mathrm{N}\times\mathrm{D}}, g_x)$ ***correspond to geodesic trajectories in the group of diffeomorphisms*** (G_k, g_φ)*. Any geodesic path (q^t, p^t) in phase space $(\mathbb{R}^{\mathrm{N}\times\mathrm{D}})^2$ can be lifted as a geodesic curve φ^t in G_k, with:*

$$\varphi^0 \overset{\text{def.}}{=} \mathrm{Id}_{\mathbb{R}^\mathrm{D}} \quad \textit{and} \quad \tfrac{\mathrm{d}}{\mathrm{d}t}\varphi^t = v^t \overset{\text{def.}}{=} k \star \sum_{i=1}^{\mathrm{N}} p_i^t\, \delta_{q_i^t} \,, \quad \textit{as} \quad \varphi^t(q_i^0) = q_i^t \ \textit{for all time } t. \tag{5.47}$$

An affordable and stable way of generating diffeomorphisms. All trajectories generated by Algorithms 5.5 and 5.6 with an LDDMM metric can be understood as deformations of the ambient space: they map the source point cloud $A^* = q^0$ onto the target $q^1 = A \simeq B$ as efficiently as possible, with a smoothness of trajectories that is controlled by the norm $\| \cdot \|_k$.

Loosely speaking, Theorem 5.3 describes a nested hierarchy of discrete shape spaces that progressively fill up the group of deformations G_k:

$$(\mathbb{R}^{1\times \mathrm{D}},\, g_x) \,\subset\, (\mathbb{R}^{2\times \mathrm{D}},\, g_x) \,\subset\, \cdots \,\subset\, (\mathbb{R}^{\mathrm{N}\times \mathrm{D}},\, g_x) \,\subset\, \cdots \,\subset\, (G_k,\, g_\varphi)\,. \tag{5.48}$$

As far as practitionners are concerned, this result ensures that LDDMM algorithms are *stable*: if we re-mesh our shapes and improve sampling rates, the discrete geodesics illustrated Figure 5.12 progressively converge towards a smooth interpolation in the group of diffeomorphisms G_k. Numerical "explosions" never occur.

Over the last fifteen years, this fundamental guarantee has led the LDDMM framework to become a standard method in biomedical imaging. Understood as a non-linear generalization of the theory of optical flows, this Riemannian toolbox allows researchers to overcome the limitations of the SVF exponential at a reasonable computational cost. It has been widely implemented in registration toolkits and powers reference software in neuro-anatomy (Klein et al., 2009a; Avants et al., 2009; Ashburner and Friston, 2011; Bône et al., 2018).

Historical ties with fluid mechanics. On the theoretical front, Theorem 5.3 is a right fit for the geometric theory of fluid mechanics pioneered by (Arnold, 1966), which cast the incompressible Euler equation as a geodesic flow in the space of volume-preserving diffeomorphisms. As detailed in (Holm et al., 2004), LDDMM trajectories are linked to the study of soliton waves: we refer to (Holm and Marsden, 2005; Sommer et al., 2013b; Bruveris and Holm, 2015) for an overview of the relevant Camassa–Holm and EPDiff equations.

The carpooling artifact. Thanks to the **extrinsic** definition of the metric $g_x = K_x^{-1}$ in Eqs. (5.32-5.35), LDDMM geodesics are fully invariant to re-parameterizations. If $\mathrm{d}(x, y)$ is the geodesic distance induced by an LDDMM metric between any two shapes x and y in $\mathbb{R}^{\mathrm{N}\times \mathrm{D}}$, then for all permutation $\sigma : [\![1, \mathrm{N}]\!] \rightarrow [\![1, \mathrm{N}]\!]$:

$$\mathrm{d}(x, y) \;=\; d(x \circ \sigma, y \circ \sigma)\;, \tag{5.49}$$

where $(x \circ \sigma)_i = (x_{\sigma(i)})$ is the cloud of points $x = (x_i)$ shuffled by σ. Going further, in the group of diffeomorphisms (G_k, g_φ) endowed with the geodesic distance d, we can show that:

$$\forall\, \varphi, \psi, \xi \in G_k,\;\; \mathrm{d}(\,\varphi\,,\,\psi\,) \;=\; \mathrm{d}(\,\varphi \circ \xi\,,\, \psi \circ \xi\,) \;=\; \mathrm{d}(\,\mathrm{Id}_{\mathbb{R}^{\mathrm{D}}}\,,\, \psi \circ \varphi^{-1}\,)\;. \tag{5.50}$$

Geometers say that LDDMM metrics are **righ-invariant** and leverage this property to prove strong existence theorems (Misiołek, 1998; Kouranbaeva, 1999; Constantin and Kolev, 2003; Bauer et al., 2013).

In practice, the extrinsic nature of LDDMM metrics has one major consequence: assuming that we use a typical "bell-shaped" kernel $k(x, y)$, geodesic interpolations are biased towards a **contraction-dilation dynamics** that we illustrate in Figures 5.13.c, 5.14.c and 5.15.a-c. The phenomenon is reminiscent of branched optimal transport (Bernot et al., 2008) and was studied in depth by (Micheli et al., 2012). As far as shape analysis is concerned, unfortunately, this **carpooling artifact** prevents LDDMM algorithms from producing reliable extrapolations: geodesics tend to blow up instead of simulating plausible evolutions in the footsteps of Figure 5.9.b.

5.3 Working towards anatomical relevance

Overview of Riemannian shape models. From Section 1.2 to the last few pages, we have seen that the design of shape metrics is a central question in medical imaging. In the context of Algorithms 5.5 and 5.6, we look for relevant functions:

$$g : (x, v) \mapsto g_x v \qquad \text{(metric)} \qquad (5.51)$$

$$\text{or} \quad K : (q, p) \mapsto K_q p \qquad \text{(cometric)} \qquad (5.52)$$

that are smooth with respect to the shape x or q and linear, symmetric, positive and definite with respect to the velocity v or momentum p.

In most settings, we need to preserve the topology of our shapes along geodesic trajectories. To control the smoothness of the infinitesimal deformation v, researchers thus tend to penalize derivatives with a **high-pass metric** $\|v\|_x^2 = \langle v, g_x v\rangle$ or generate velocities as the filtering $v = K_q p$ of the momentum p with a **low-pass cometric** K_q. The two most iconic Riemannian frameworks for shape analysis fit this template, with complementary strengths and weaknesses:

1. **Elastic models** rely on mesh structures to define a Laplacian-like metric g_x that is encoded as a sparse matrix. They produce sensible trajectories in shape space but are notoriously hard to interface with clinical data: building a clean biomechanical mesh out of a noisy 3D scan is a challenging problem.

2. **LDDMM models** rely on grid convolutions or kernel dot products to define extrinsic metrics $g_x \simeq (\mathrm{Id} - \alpha\Delta)^2$ and cometrics $K_q = (k(q_i, q_j))_{i,j}$. They are versatile and robust to topological noise but do not extrapolate well "beyond time $t = 1$".

Working towards data-driven heterogeneity. Standard implementations of the models above rely on a uniform description of the elastic material or on a translation-invariant kernel $k(x-y)$. The simplistic assumption of homogeneity is convenient, but hardly relevant in a medical setting: an organ is everything but an isotropic piece of gum. Going forward, researchers should leverage expert knowledge and data-driven insights to define **adaptive** shape metrics... But this is easier said than done.

5.3.1 Learning the metric

Anatomical data is expensive: every single medical scan is processed by a team of skilled technicians and radiologists. Compared with the millions of annotated images that have been made available in computer vision, **high-quality data is thus relatively scarce in computational anatomy**: datasets for population studies now document up to 100k patients, but specific syndroms are seldom present in more than 100 to 1,000 images at a time.

Combined with the high dimensionality of shape data, this situation prevents standard machine learning techniques from bringing a ready-made solution to our problems: to generalize outside of small datasets, we must combine statistical tools with **geometric architectures**.

Finding the right balance between built-in priors and data-driven insights has always been a challenging problem for the shape analysis community. But fortunately, from 2017 onwards, versatile deep learning libraries have acted as a catalyst for research in the field (Kühnel and Sommer, 2017): as discussed throughout Chapter 2, the powerful mix of automatic differentiation and GPU computing provided by `Theano`, `PyTorch` and others has allowed researchers to divide by 10 (!) the size of their codebases over the last three years.

The `KeOps` and `GeomLoss` libraries fit within this global progression towards modular development. They allow us to focus on the design of data-driven shape models, where three main strategies prevail:

1. **Deformable models and autoencoders.** In line with (Blanz et al., 1999; Wu et al., 2016; Litany et al., 2018), we can fit a parametric generative model:

$$\Phi_\theta \;:\; c \in \mathbb{R}^{\mathrm{M}} \;\mapsto\; x \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \tag{5.53}$$

   to our datasets, where θ is a vector of neural weights to optimize. Shapes x are then compared with each other in the latent space of codes $c = \Phi^{-1}(x) \in \mathbb{R}^{\mathrm{M}}$, endowed with the standard Euclidean distance. This is equivalent to endowing the shape space $\Phi_\theta(\mathbb{R}^{\mathrm{M}}) \subset \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ with the Riemannian metric:

$$\mathrm{d}^2(x, x+\delta x) = \|\, \mathrm{d}\Phi_\theta^{-1}(x)\cdot \delta x\,\|^2_{\mathbb{R}^{\mathrm{M}}} \qquad \text{i.e.} \qquad g_x = [\mathrm{d}\Phi_\theta^{-1}(x)]^\top\, \mathrm{d}\Phi_\theta^{-1}(x) \tag{5.54}$$

$$\text{or}\quad K_x = \mathrm{d}\Phi_\theta(c)\,\mathrm{d}\Phi_\theta^\top(c)\,, \qquad \text{with} \qquad c = \Phi_\theta^{-1}(x)\,, \tag{5.55}$$

   making it isometric to the flat Euclidean space $\mathbb{R}^{\mathrm{M}}$ – a strong but convenient assumption. Going further, standard machine learning techniques for metric learning can be applied to the distribution of latent codes (Lin and Zha, 2008): we refer to (Krebs et al., 2019) for an example of medical application.

2. **Hierarchical and implicit models.** Alternatively, we can put an emphasis on interpretability and build our operators g_x and K_q as hierarchical compositions of well-known deformation modules (Sommer et al., 2011, 2013a). This approach is usually studied through the lens of optimal control theory (Arguillere, 2014; Arguillere et al., 2015) and is well-suited to the formulation of semi-parametric models in biology (Kaltenmark, 2016; Gris et al., 2018; Gris, 2019).

3. **Adaptive metrics.** Finally, as discussed for instance in (Vialard and Risser, 2014), we can rely on feature detectors to alter locally the parameters of a generalized elastic or LDDMM model. Modern deep learning libraries now allow us to scale this approach to realistic datasets, with promising results (Niethammer et al., 2019a; Shen et al., 2019).

5.3.2 Enforcing axioms on the target geometry

Opening the Pandora box. Relieved from the burden of low-level development, researchers in the field are now more creative than ever. But can we design data-driven models while retaining the guarantees that made the SVF and LDDMM frameworks popular in the first place?

Modern research in the field is concerned with the development of advanced (black-box?) models g_x or K_q that encode the geometry of a population. These may for instance be encoded as forward-backward products:

$$K_q p \overset{\text{def.}}{=} L_q^\top L_q p\,, \qquad \text{where} \qquad L_q : \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \to \mathbb{R}^{\mathrm{M}} \qquad \text{is an arbitrary feature map} \tag{5.56}$$

to guarantee the symmetry and positivity of a Hamiltonian function that reads:

$$H(q,p) \overset{\text{def.}}{=} \tfrac{1}{2}\langle p\,,\, K_q p\rangle \;=\; \tfrac{1}{2}\|\, L_q p\,\|^2_{L^2(\mathbb{R}^{\mathrm{M}})}\,. \tag{5.57}$$

In this very open context, an interesting research topic is to look for ways of enforcing **meaningful axioms** on generic shape metrics, without making strong assumptions on the inner structure of the operators (g_x) and (K_q). How should we proceed?

Keeping metrics under control with an adaptive scaling. If $K_q \in \mathbb{R}^{\mathrm{ND}\times\mathrm{ND}}$ is a black-box Riemannian cometric, a simple way of controlling its properties is to compose it with a local **scaling** matrix $S_q \in \mathbb{R}^{\mathrm{ND}\times\mathrm{ND}}$ and consider the operator:

$$\widetilde{K} \,:\, (q,p) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \times \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \mapsto \widetilde{K}_q p \overset{\text{def.}}{=} S_q^\top K_q S_q p \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}} \,. \tag{5.58}$$

After all, the adaptive change of coordinates induced by the matrix S_q on the momentum $p \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ could allow us to enforce some structure on the Riemannian cometric $\widetilde{K} = S^\top K S$ while retaining guarantees of symmetry and positivity.

As an illustration, let us assume that our base cometric K_q is encoded as a large N-by-N matrix with *positive* coefficients, associated to a Hamiltonian:

$$H(q,p) \,=\, \tfrac{1}{2}\langle\, p\,,\, K_q p \,\rangle_{\mathbb{R}^{\mathrm{N}\times\mathrm{D}}} \,=\, \sum_{i=1}^{\mathrm{N}}\sum_{j=1}^{\mathrm{N}} (K_q)_{i,j}\, \langle p_i, p_j\rangle_{\mathbb{R}^{\mathrm{D}}} \,\geqslant\, 0\,. \tag{5.59}$$

This is typically the case with LDDMM metrics, where $K_q = (k(q_i,q_j))_{i,j} \in \mathbb{R}^{\mathrm{N}\times\mathrm{N}}$ is induced by a positive kernel function $k(x,y) > 0$. Let us also assume that our shape q is encoded as a positive measure on the ambient space: non-negative **importance weights** $m_i \in \mathbb{R}_{\geqslant 0}$ are associated to the points $q_i \in \mathbb{R}^{\mathrm{D}}$, defining a measure $\mu = \sum_{i=1}^{\mathrm{N}} m_i \delta_{q_i}$ with a positive total mass.

Then, we propose to rescale adaptively the cometric $K_q \in \mathbb{R}_{>0}^{\mathrm{N}\times\mathrm{N}}$ using a diagonal matrix $S_q = \mathrm{Diag}(\, s_{q_i}\,, i \in [\![1,\mathrm{N}]\!]\,)$, where the positive vector $s_q = (s_{q_i}) \in \mathbb{R}_{>0}^{\mathrm{N}}$ is chosen so that:

$$s_q \cdot K_q(s_q \cdot m) \,=\, \widetilde{K}_q m \,=\, \mathbf{1}_{\mathbb{R}^{\mathrm{N}}}\,, \quad \text{with "$\cdot$" denoting the pointwise product in } \mathbb{R}^{\mathrm{N}}. \tag{5.60}$$

First properties. The existence and unicity of the scaling vector s_q is guaranteed by our results on entropic optimal transport. For any choice of the positive definite matrix K_q and vector of weights $m = (m_i) \in \mathbb{R}_{>0}^{\mathrm{N}}$, using the notations of page 112, we can indeed remark that $f^{\mu\leftrightarrow\mu} = \log(s_q)$ is the unique solution of the symmetric problem $\mathrm{OT}_{\varepsilon=1}(\mu,\mu)$ with a cost matrix $\mathrm{C}(q_i,q_j) = -\log(K_q)_{i,j}$. A detailed proof is given page 194. In practice, following the discussion of page 119, we compute the values of s_q on-the-fly with the symmetric Sinkhorn loop of Algorithm 5.7 and rely on automatic differentiation whenever required by Algorithm 5.6.

An LDDMM cometric $K_q = (k(q_i - q_j))_{i,j}$ acts as a convolution with the kernel k. On the other hand, after our proposed normalization with the scaling vector s_q, the cometric $\widetilde{K}$ behaves as an **interpolation operator**. Thanks to Eq. (5.60), the velocity field $v \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ induced by a momentum $p = (p_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ reads:

$$v \,=\, \widetilde{K}_q p \,=\, \frac{K_q(s_q\cdot p)}{K_q(s_q\cdot m)} \qquad \text{i.e.} \qquad v_i \,=\, v(q_i) \,=\, \frac{\sum_{j=1}^{\mathrm{N}} k(q_i-q_j)\, s_{q_j}\, p_j}{\sum_{j=1}^{\mathrm{N}} k(q_i-q_j)\, s_{q_j}\, m_j}\,. \tag{5.61}$$

The velocity v_i of a point q_i is therefore a **barycentric combination** of the anchor velocities:

$$\tilde{v}_j \overset{\text{def.}}{=} p_j/m_j \in \mathbb{R}^{\mathrm{D}} \qquad \text{with weights} \qquad w_{i,j} \,\propto\, k(q_i-q_j)\, s_{q_j}\, m_j \,\geqslant\, 0 \tag{5.62}$$

and is associated to a total energy that reads:

$$H(q,p) \,=\, \tfrac{1}{2}\langle p, v\rangle \,=\, \frac{1}{2}\sum_{i=1}^{\mathrm{N}} m_i \langle \widetilde{v}_i, v_i\rangle_{\mathbb{R}^{\mathrm{D}}} \,=\, \tfrac{1}{2}\langle v, \widetilde{K}_q^{-1} v\rangle \,=\, E(q,v)\,. \tag{5.63}$$

This new Riemannian structure on weighted point clouds behaves as a smooth generalization of the Wasserstein-2 metric $\frac{1}{2}\langle v, g_q v\rangle = \frac{1}{2}\sum_i m_i \|v_i\|_{\mathbb{R}^{\mathrm{D}}}^2$. We implement it on meshes and density maps at a cost that is 2 to 5 times that of a standard LDDMM metric: the loop of Algorithm 5.7 can be inserted in any implementation of Algorithm 5.6.

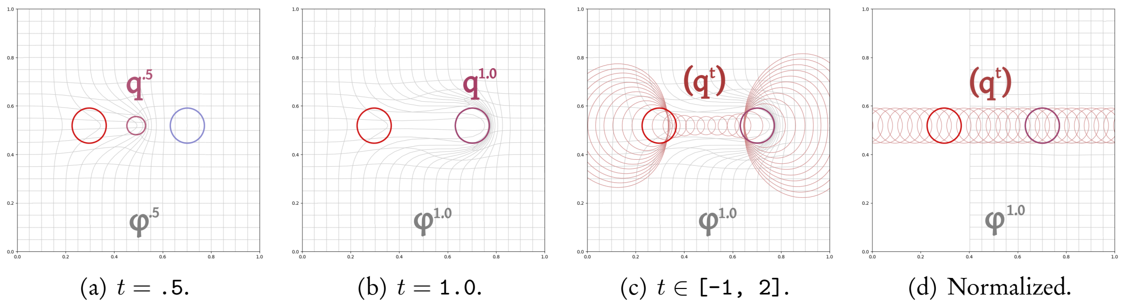


Figure 5.14: Normalizing metrics to retrieve sensible translations. We display the shape geodesic (q^t) between translated copies of a disk at times $t = 0$ and 1 using two distinct Riemannian models: (a-c) a Gaussian LDDMM cometric $(K_q)_{i,j} = \exp(-\|q_i - q_j\|^2/2\sigma^2)$ with $\sigma = .05$ on the unit square; (d) its normalization $\widetilde{K}_q$, with importance weights m_i that are proportional to arc length.
(a-c) As discussed Figure 5.13, LDDMM geodesics rely on a contraction-dilation dynamics that hampers extrapolations outside of the interval $t \in [0, 1]$. (d) Fortunately, the normalization loop of Algorithm 5.7 allows us to get rid of the carpooling artifact: we retrieve a clean translation.

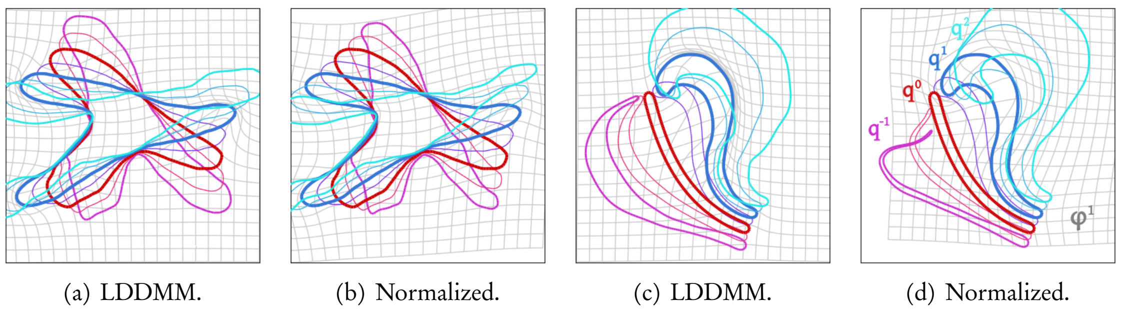


Figure 5.15: Robust extrapolations with a normalized LDDMM metric. We display side-by-side the geodesic curves between $A = q^0$ (red) and $B = q^1$ (blue) for an LDDMM metric K_q (a, c) and its normalization $\widetilde{K}_q$ (b, d). We use a Cauchy kernel $k(x, y) = 1/(1 + \|x - y\|^2/\sigma^2)$ of radius $\sigma = .05$ on the unit square, with importance weights m_i that are proportional to arc length at time $t = 0$.
As illustrated Figure 5.14, normalized cometrics $\widetilde{K}_q$ do not suffer from the carpooling artifact. The extrapolations at time $t = -1$ (purple) and $t = +2$ (cyan) preserve the topologies of our shapes with a natural fit, free of any "rebound" and extravagant growth.

Algorithm 5.7: Normalized shape cometric $\widetilde{K}_q$

Input: Weighted point cloud $q = \sum_{i=1}^{N} m_i \delta_{q_i}$ with $(q_i) \in \mathbb{R}^{N\times D}$ and $(m_i) \in \mathbb{R}^{N}_{>0}$,
Momentum $p = (p_i) \in \mathbb{R}^{N\times D}$ supported by the q_i's .

Parameters: Black-box cometric $K : (q, p) \mapsto K_q p$, with $K_q \in \mathbb{R}^{N\times N}_{>0}$.

```
1: $s_q \leftarrow \mathbf{1}_{\mathbb{R}^N}$
2: for it = 1 to $n_{\text{its}}$ do                        ▷ 2 to 5 steps are enough in practice.
3:     $s_q \leftarrow \sqrt{s_q / K(q, s_q \cdot m)}$      ▷ Symmetric Sinkhorn iterations, $s_q = \exp(f/1)$.
4: return $s_q \cdot K(q, s_q \cdot p)$                      ▷ Velocity $v = \widetilde{K}_q p \in \mathbb{R}^{N\times D}$.
```

Local translations. The Sinkhorn normalization trick turns a convolution K_q into an interpolator $\widetilde{K_q}$ that shares many properties with a poly-affine model (Arsigny et al., 2005, 2009). Remarkably, it also guarantees the preservation of **translations as geodesics in shape space**.

If $q = (q_i) \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is a point cloud with non-negative weights $m = (m_i) \in \mathbb{R}^{\mathrm{N}}_{\geqslant 0}$, we can indeed remark that the infinitesimal translation of vector $\vec{v} \in \mathbb{R}^3$ is generated by the momentum:

$$p_{\vec{v}} \overset{\text{def.}}{=} m\,\vec{v}^\top = (m_i\,\vec{v})_i \in \mathbb{R}^{\mathrm{N}\times 3}\,, \qquad \text{so that} \qquad \widetilde{K_q}\,p_{\vec{v}} \;=\; \widetilde{K_q}\,m\,\vec{v}^\top \;=\; \mathbf{1}_{\mathbb{R}^\mathrm{N}}\,\vec{v}^\top\,, \tag{5.64}$$

with a total cost $H(q, p_{\vec{v}}) = \frac{1}{2}\sum_i m_i \|\vec{v}\|^2$ that does *not* depend on the shape q. The trajectory $(q^t, p^t) = (q + t\vec{v}, p_{\vec{v}})$ is therefore a solution of the geodesic Equation (5.22).

5.3.3 Missing pieces in the puzzle – future works

Unlocking exciting research directions. As illustrated in Figures 5.14 and 5.15, normalizing an LDDMM metric allows us to straighten its geodesics and get rid of the carpooling artifact: the normalized cometric $\widetilde{K_q}$ induces sensible trajectories in shape space without relying on a clean mesh. Going further, we could work with iterated kernel products $K_q^n = K_q \cdots K_q$ and their normalizations to approximate an intrinsic mesh diffusion with point clouds: this would allow us to bridge the gap between LDDMM, optimal transport and elastic models.

These ideas will certainly appeal to geometers. But can we *prove* that they define stable trajectories in shape space? That they bring value to practitioners in a significant and reproducible way? We believe that normalized metrics can advantageously replace LDDMM or spline models as a robust baseline for the study of shape dynamics (Trouvé and Vialard, 2012; Durrleman et al., 2013; Schiratti et al., 2015; Koval et al., 2017) – but providing solid evidence to back up this claim will likely require a few more years of work.

A powerful and versatile environment. Crucially, none of this research would have been possible without the investment in foundational tools that we described from Chapters 2 to 4:

1. **Efficient geometric routines** let us scale up to real 3D shapes with human-readable scripts. Thanks to the `KeOps` library, we can study principled Riemannian models while remaining credible in the deep learning era.

2. **Global loss functions** stabilize our pipelines and make them robust to large deformations. Wasserstein-like distances are usually more reliable than kernel norms: they allow us to experiment freely with deformation models, without having to re-tune kernel parameters at every turn.

Enabling new interactions. These tools will certainly become fundamental bricks for a new generation of methods in computational anatomy. Over the last three years, the transition from monolithic C++ codebases towards modular high-level packages has been a revolution for the field. The barrier of entry to state-of-the-art methods has been greatly reduced, which has opened the door to a whole new range of interactions with related communities. Going forward, computational anatomy is bound to benefit from a closer integration with the graphics, vision and simulation literatures: we look forward to working on the topic in years to come.

Appendix A

Detailed proofs on entropic optimal transport

in collaboration with François-Xavier Vialard (Paris-Est University), Thibault Séjourné and Gabriel Peyré (École Normale Supérieure).

A.1 Statement of Theorem 3.1, simple arguments

This chapter contains our original proofs on optimal transport theory, related to the Schrödinger problem $\mathrm{OT}_\varepsilon(\alpha,\beta)$ of Eq. (3.187) and to the debiased Sinkhorn divergence:

$$\mathrm{S}_\varepsilon(\alpha,\beta) \overset{\text{def.}}{=} \mathrm{OT}_\varepsilon(\alpha,\beta) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta)\,. \tag{A.1}$$

As discussed page 115, it contains two quick proofs of positivity for S_ε, followed by a full demonstration of Theorem 3.1.

A.1.1 Notations, technical hypotheses

Throughout these pages, we use the measure-theoretic notations detailed Section 3.1.3. In addition to that, if $f \in \mathcal{C}(\mathcal{X})$, we write:

$$\|f\|_\infty \overset{\text{def.}}{=} \max_{x\in\mathcal{X}} |f(x)| \tag{A.2}$$

and say that a sequence f_n of continuous functions converges uniformly towards a limit f_∞ if $\|f_n - f\|_\infty$ converges towards 0.

Lipschitz cost functions. In this chapter, we consider *ground cost* functions:

$$\mathbf{C} \;:\; (x,y)\in\mathcal{X}\times\mathcal{X} \;\mapsto\; \mathbf{C}(x,y) = \mathbf{C}(y,x) \in \mathbb{R} \tag{A.3}$$

that are symmetric and Lipschitz with respect to both variables on the bounded feature space $(\mathcal{X},\mathrm{d})$. That is, there should exist some constant $\kappa > 0$ such that:

$$\forall\, x,y,y' \in \mathcal{X},\;\; |\,\mathbf{C}(x,y) - \mathbf{C}(x,y')\,| \;\leqslant\; \kappa\cdot\mathrm{d}(y,y')\,. \tag{A.4}$$

Gibbs kernel. An arbitrary cost function defines a kernel through an exponential mapping: for any temperature $\varepsilon > 0$, we define the Gibbs kernel of $\mathbf{C}$ as:

$$k_\varepsilon ~:~ (x,y) \in \mathcal{X} \times \mathcal{X} ~\mapsto~ \exp(-\tfrac{1}{\varepsilon}\mathbf{C}(x,y)) \in \mathbb{R}~. \tag{A.5}$$

This operation is central to the theory of regularized OT, and our main results (positivity, metrization of the convergence in law by Sinkhorn and Hausdorff divergences) hold for cost functions that define *positive* kernels k_ε as discussed in Section 3.2.3.

Typical use cases. In the simplest of all settings, we compare with each other two measures α and β on a Euclidean space $\mathbb{R}^\mathrm{D}$. We suppose that both have compact support, and can thus work in a *compact*, bounded domain:

$$\mathcal{X} ~=~ \{x \in \mathbb{R}^\mathrm{D},~ \|x\| \leqslant R\} \quad \text{with} \quad \mathrm{d}(x,y) ~=~ \|x-y\|~. \tag{A.6}$$

Then, we endow $\mathcal{X}$ with an earth mover cost $\mathbf{C}(x,y) = \|x-y\|$ or a Wasserstein quadratic cost $\mathbf{C}(x,y) = \frac{1}{2}\|x-y\|^2$, which are both Lipschitz on bounded subsets of $\mathbb{R}^\mathrm{D}$ and induce exponential and Gaussian kernels respectively:

$$k_\varepsilon(x,y) ~=~ \exp(-\tfrac{1}{\varepsilon}\|x-y\|)~, \quad k_\varepsilon(x,y) ~=~ \exp(-\tfrac{1}{2\varepsilon}\|x-y\|^2)~. \tag{A.7}$$

A.1.2 The theorem

As discussed in Section 3.3.2, the theorem below legitimizes the use of the debiased Sinkhorn divergence S_ε for measure-fitting applications.

Theorem A.1. *Let $\mathcal{X}$ be a compact metric space with a Lipschitz cost function $\mathbf{C}(x,y)$ that induces, for $\varepsilon > 0$, a* positive universal *kernel $k_\varepsilon(x,y) \overset{\text{def.}}{=} \exp(-\mathbf{C}(x,y)/\varepsilon)$. Then, S_ε defines a symmetric, positive, definite and smooth loss function that is convex in each of its input variables. It also metrizes the convergence in law: for all probability Radon measures α and $\beta \in \mathcal{M}_1^+(\mathcal{X})$,*

$$0 ~= \mathrm{S}_\varepsilon(\beta,\beta) ~\leqslant~ \mathrm{S}_\varepsilon(\alpha,\beta)~, \tag{A.8}$$
$$\alpha = \beta ~\iff~ \mathrm{S}_\varepsilon(\alpha,\beta) = 0~, \tag{A.9}$$
$$\alpha_n \rightharpoonup \alpha ~\iff~ \mathrm{S}_\varepsilon(\alpha_n,\alpha) \to 0~. \tag{A.10}$$

Notably, these results also hold for measures with bounded support on a Euclidean space $\mathcal{X} = \mathbb{R}^\mathrm{D}$ endowed with ground cost functions $\mathbf{C}(x,y) = \|x-y\|$ or $\mathbf{C}(x,y) = \frac{1}{2}\|x-y\|^2$ – which induce exponential and Gaussian kernels respectively.

A.1.3 Two quick, partial proofs

We give a full demonstration of Theorem 3.1 in the next fifteen pages or so. This proof is fairly technical, especially with regards to the (surprising) concavity of OT_ε on the diagonal "$\alpha = \beta$" of the space of pairs of measures.

Measure-function change of variables. We remark, however, that a simple change of variables can be used to show that:

$$\forall\, \alpha, \beta \in \mathcal{M}_1^+(\mathcal{X}),\;\; \text{S}_\varepsilon(\alpha, \beta) \;\geqslant\; 0 \tag{A.11}$$

with a minimal amount of effort. If (f, g) is a pair of competitors for the dual problem $\text{OT}_\varepsilon(\alpha, \beta)$ of Eq. (3.189), we introduce the "scaled" measures:

$$\overline{\alpha} \;\overset{\text{def.}}{=}\; e^{f/\varepsilon}\,\alpha \qquad\qquad \text{and} \qquad\qquad \overline{\beta} \;\overset{\text{def.}}{=}\; e^{g/\varepsilon}\,\beta\,. \tag{A.12}$$

The optimality equations for $\text{OT}_\varepsilon(\alpha, \beta)$, detailed Eqs. (3.194-3.195), ensure that at the optimum:

$$\langle\, \overline{\alpha},\, \overline{\beta}\,\rangle_{k_\varepsilon} \;\overset{\text{def.}}{=}\; \langle\, \overline{\alpha},\, k_\varepsilon \star \overline{\beta}\,\rangle \;=\; \langle\, \overline{\beta},\, k_\varepsilon \star \overline{\alpha}\,\rangle \;=\; 1\,. \tag{A.13}$$

Consequently, the soft penalty "$f \oplus g \leqslant \mathbf{C}$" vanishes at the optimum. If $\overline{\alpha}$ and $\overline{\beta}$ are associated to the **optimal** pair of dual potentials (f, g):

$$\text{OT}_\varepsilon(\alpha, \beta) \;=\; \langle \alpha,\, f\rangle + \langle \beta,\, g\rangle \;=\; \varepsilon\langle \alpha,\, \log \tfrac{\mathrm{d}\overline{\alpha}}{\mathrm{d}\alpha}\rangle \;+\; \varepsilon\langle \beta,\, \log \tfrac{\mathrm{d}\overline{\beta}}{\mathrm{d}\beta}\rangle\,. \tag{A.14}$$

First proof of positivity. Now, let $(f^{\alpha\leftrightarrow\alpha}, f^{\alpha\leftrightarrow\alpha})$ and $(g^{\beta\leftrightarrow\beta}, f^{\beta\leftrightarrow\beta})$ be the unique solutions of the symmetric Schrödinger problems $\text{OT}_\varepsilon(\alpha, \alpha)$ and $\text{OT}_\varepsilon(\beta, \beta)$ on the diagonal of the space of pairs of dual potentials. According to the discussion above, we know that:

$$\tfrac{1}{2}\text{OT}_\varepsilon(\alpha, \alpha) \;=\; \langle \alpha,\, f^{\alpha\leftrightarrow\alpha}\rangle \quad\;\; \text{and} \quad\;\; \tfrac{1}{2}\text{OT}_\varepsilon(\beta, \beta) \;=\; \langle \beta,\, f^{\beta\leftrightarrow\beta}\rangle\,. \tag{A.15}$$

We also know that the scaled "symmetric" measures:

$$\overline{\alpha} \;\overset{\text{def.}}{=}\; e^{f^{\alpha\leftrightarrow\alpha}/\varepsilon}\,\alpha \qquad\qquad \text{and} \qquad\qquad \overline{\beta} \;\overset{\text{def.}}{=}\; e^{g^{\beta\leftrightarrow\beta}/\varepsilon}\,\beta \tag{A.16}$$

lie on the **unit ball** of the space of probability measures $\mathcal{M}_1^+(\mathcal{X})$, endowed with the kernel norm $\|\,\cdot\,\|_{k_\varepsilon}$:

$$\|\overline{\alpha}\|_{k_\varepsilon}^2 \;\overset{\text{def.}}{=}\; \langle \overline{\alpha},\, k_\varepsilon \star \overline{\alpha}\rangle \;=\; 1 \quad\;\; \text{and} \quad\;\; \|\overline{\beta}\|_{k_\varepsilon}^2 \;\overset{\text{def.}}{=}\; \langle \overline{\beta},\, k_\varepsilon \star \overline{\beta}\rangle \;=\; 1\,. \tag{A.17}$$

To conclude, it suffices to remark that $(f^{\alpha\leftrightarrow\alpha}, g^{\beta\leftrightarrow\beta})$ is an admissible competitor for the concave maximization problem $\mathrm{OT}_\varepsilon(\alpha,\beta)$:

$$\begin{aligned} \mathrm{OT}_\varepsilon(\alpha,\beta) \;\geqslant\; & \langle \alpha,\, f^{\alpha\leftrightarrow\alpha}\rangle \;+\; \langle \beta,\, g^{\beta\leftrightarrow\beta}\rangle \\ & +\; \varepsilon\langle \alpha\otimes\beta,\, 1-\exp\tfrac{1}{\varepsilon}[f^{\alpha\leftrightarrow\alpha}\oplus g^{\beta\leftrightarrow\beta}-\mathbf{C}]\rangle\;. \end{aligned} \tag{A.18}$$

We remind the reader that in this chapter, α and β are both assumed to be probability measures that sum-up to 1. Therefore, using our scaled measures $\overline{\alpha}$ and $\overline{\beta}$:

$$\begin{aligned} \mathrm{S}_\varepsilon(\alpha,\beta) \;&\overset{\text{def.}}{=}\; \mathrm{OT}_\varepsilon(\alpha,\beta) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta) && \text{(A.19)} \\ &=\; \mathrm{OT}_\varepsilon(\alpha,\beta) - \langle\alpha,\, f^{\alpha\leftrightarrow\alpha}\rangle - \langle\beta,\, g^{\beta\leftrightarrow\beta}\rangle && \text{(A.20)} \\ &\geqslant\; \varepsilon\langle\alpha\otimes\beta,\, 1-\exp\tfrac{1}{\varepsilon}[^{\alpha\leftrightarrow\alpha}\oplus g^{\beta\leftrightarrow\beta}-\mathbf{C}]\rangle && \text{(A.21)} \\ &=\; \varepsilon\big[\langle\alpha\otimes\beta,\, 1\rangle - \langle\overline{\alpha}\otimes\overline{\beta},\, k_\varepsilon\rangle\big] && \text{(A.22)} \\ &=\; \varepsilon\big[1-\langle\overline{\alpha},\, \overline{\beta}\rangle_{k_\varepsilon}\big] && \text{(A.23)} \\ &=\; \tfrac{\varepsilon}{2}\|\,\overline{\alpha}\;-\;\overline{\beta}\,\|^2_{k_\varepsilon}\;. && \text{(A.24)} \end{aligned}$$

Under the assumption that k_ε defines a positive kernel, this allows us to conclude: $\mathrm{S}_\varepsilon(\alpha,\beta)\geqslant 0\,.$ □

Second proof of positivity. Alternatively, we remark that the dual Schrödinger problem OT_ε can be expressed using the "elementary" formulation of Eq. (3.191) :

$$\begin{aligned} \mathrm{OT}_\varepsilon(\alpha,\beta) \;&=\; \max_{f,g\in\mathcal{C}(\mathcal{X})}\; \langle\alpha,\, f\rangle \;+\; \langle\beta,\, g\rangle \;\text{ s.t. }\; \max_{\alpha\otimes\beta}{}_{\varepsilon}\;[f\oplus g-\mathbf{C}] \;\leqslant\; 0 && \text{(A.25)} \\ &=\; \max_{\overline{\alpha},\overline{\beta}\in\mathcal{M}^+(\mathcal{X})}\; \varepsilon\langle\alpha,\, \log\tfrac{\mathrm{d}\overline{\alpha}}{\mathrm{d}\alpha}\rangle \;+\; \varepsilon\langle\beta,\, \log\tfrac{\mathrm{d}\overline{\beta}}{\mathrm{d}\beta}\rangle \;\text{ s.t. }\; \langle\overline{\alpha},\, \overline{\beta}\rangle_{k_\varepsilon}\leqslant 1\;. && \text{(A.26)} \end{aligned}$$

Meanwhile, the debiasing term can be expressed in similar fashion:

$$\begin{aligned} \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) \;+\; \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta) \;=\; & \max_{\overline{\alpha},\overline{\beta}\in\mathcal{M}^+(\mathcal{X})}\; \varepsilon\langle\alpha,\, \log\tfrac{\mathrm{d}\overline{\alpha}}{\mathrm{d}\alpha}\rangle \;+\; \varepsilon\langle\beta,\, \log\tfrac{\mathrm{d}\overline{\beta}}{\mathrm{d}\beta}\rangle \\ & \text{s.t. }\; \|\overline{\alpha}\|^2_{k_\varepsilon} \;\leqslant\; 1 \quad\text{and}\quad \|\overline{\beta}\|^2_{k_\varepsilon} \;\leqslant\; 1\;. \end{aligned} \tag{A.27}$$

This is essentially equivalent to Proposition A.3, whose rigorous proof is detailed in Section A.4.3. Since we assume that k_ε is a positive kernel, $\|\cdot\|^2_{k_\varepsilon}$ satisfies the Cauchy-Schwarz inequality:

$$\|\overline{\alpha}\|^2_{k_\varepsilon} \;\leqslant\; 1 \;\text{ and }\; \|\overline{\beta}\|^2_{k_\varepsilon} \;\leqslant\; 1 \quad\Longrightarrow\quad \langle\overline{\alpha},\, \overline{\beta}\rangle_{k_\varepsilon}\leqslant 1\;. \tag{A.28}$$

Both OT_ε and the debiasing term can be expressed as the maximum of the same functional, with the second domain included into the first one. This allows us to conclude: $\mathrm{S}_\varepsilon(\alpha,\beta)\geqslant 0\,.$ □

A.2 High-level demonstration of Theorem 3.1

We now give a full proof of Theorem 3.1. Our argument relies on a new Bregman divergence derived from a weak-$\star$ continuous entropy that we call the *Sinkhorn entropy* (see Section A.2.2). We believe this (convex) entropy function to be of independent interest: since our first publication on the subject, it was for instance used in (Mensch et al., 2019) for structured learning. Note that all this section is written under the assumptions of Theorem 3.1.

A.2.1 Properties of the OT_ε loss

First, let us recall some standard results of regularized OT theory whose proof may for instance be found in (Peyré and Cuturi, 2017). Thanks to the Fenchel-Rockafellar theorem, we can rewrite the Schrödinger problem of Eq. (3.187) as an un-constrained dual maximization:

$$\begin{aligned}\text{OT}_\varepsilon(\alpha,\beta) \overset{\text{def.}}{=} \max_{(f,g)\in\mathcal{C}(\mathcal{X})^2} &\langle\alpha,f\rangle + \langle\beta,g\rangle \\ &- \varepsilon\langle\alpha\otimes\beta, \exp\big(\tfrac{1}{\varepsilon}(f\oplus g-\mathbf{C})\big)-1\rangle\,,\end{aligned} \tag{A.29}$$

where $f\oplus g$ is the tensor sum $(x,y)\in\mathcal{X}^2\mapsto f(x)+g(y)$. The primal-dual relationship linking an optimal transport plan π solving (3.168) to an optimal dual pair (f,g) that solves (A.29) is recalled in Eqs. (3.192-3.193):

$$\pi = \exp\big(\tfrac{1}{\varepsilon}(f\oplus g-\mathbf{C})\big)\cdot(\alpha\otimes\beta)\,. \tag{A.30}$$

Crucially, the first order optimality conditions for the dual variables are equivalent to the primal's marginal constraints ($\pi_1=\alpha, \pi_2=\beta$) on (A.30). They read

$$f = T(\beta,g) \;\; \alpha\text{-a.e.} \quad \text{and} \quad g = T(\alpha,f) \;\; \beta\text{-a.e.}\,, \tag{A.31}$$

where the "Sinkhorn mapping" $T:\mathcal{M}_1^+(\mathcal{X})\times\mathcal{C}(\mathcal{X})\to\mathcal{C}(\mathcal{X})$ is defined through

$$T:(\alpha,f)\mapsto\Big(y\in\mathcal{X}\mapsto \min_{x\sim\alpha}{}_\varepsilon\,\big[\mathbf{C}(x,y)-f(x)\big]\Big)\,, \tag{A.32}$$

with a SoftMin operator of strength ε defined through

$$\min_{x\sim\alpha}{}_\varepsilon\,\varphi(x) \overset{\text{def.}}{=} -\varepsilon\log\int_{\mathcal{X}}\exp\big(-\tfrac{1}{\varepsilon}\varphi(x)\big)\,\mathrm{d}\alpha(x)\,. \tag{A.33}$$

Dual potentials. The following proposition recalls some important properties of OT_ε and the associated dual potentials. Its proof can be found in Section A.4.1.

Proposition A.1 (Properties of OT_ε)**.** *The optimal potentials (f,g) exist and are unique (α,β)-a.e. up to an additive constant, i.e. $\forall K\in\mathbb{R}$, $(f+K,g-K)$ is also optimal. At optimality, we get:*

$$\text{OT}_\varepsilon(\alpha,\beta) = \langle\alpha,f\rangle+\langle\beta,g\rangle\,. \tag{A.34}$$

Gradients. We recall that, as discussed page 73, a functional $\mathrm{F} : \mathcal{M}_1^+(\mathcal{X}) \rightarrow \mathbb{R}$ is said to be *differentiable* if there exists $\nabla\mathrm{F}(\alpha) \in \mathcal{C}(\mathcal{X})$ such that for any displacement $\xi = \beta - \beta'$ with $(\beta, \beta') \in \mathcal{M}_1^+(\mathcal{X})^2$, we have:

$$\mathrm{F}(\alpha + t\xi) = \mathrm{F}(\alpha) + t\langle\, \xi,\, \nabla\mathrm{F}(\alpha)\rangle + o(t)\ . \tag{A.35}$$

The following proposition, whose proof is detailed in Section A.4.2, shows that the dual potentials are the gradients of OT_ε.

Proposition A.2. OT_ε *is* weak-$\star$ continuous *and* differentiable. *Its gradient reads:*

$$\nabla\mathrm{OT}_\varepsilon(\alpha, \beta) = (f, g) \tag{A.36}$$

where (f, g) *satisfies* $f = T(\beta, g)$ *and* $g = T(\alpha, f)$ *on the whole domain* $\mathcal{X}$ *and* T *is the Sinkhorn mapping of Eq.* (A.32).

Let us stress that even though the solutions of the dual problem (A.29) are defined (α, β)-a.e., the gradient (A.36) is defined on the whole domain $\mathcal{X}$. Fortunately, an optimal dual pair (f_0, g_0) defined (α, β)-a.e. satisfies the optimality condition (A.31) and can be *extended* in a canonical way: to compute the "gradient" pair $(f, g) \in \mathcal{C}(\mathcal{X})^2$ associated to a pair of measures (α, β), using $f = T(\beta, g_0)$ and $g = T(\alpha, f_0)$ is enough.

A.2.2 Sinkhorn and Haussdorf divergences

Having recalled some standard properties of OT_ε, let us now state a few *original* facts about the corrective, symmetric term $-\frac{1}{2}\mathrm{OT}_\varepsilon(\alpha, \alpha)$ used to define the debiased Sinkhorn divergence S_ε. We still suppose that $(\mathcal{X}, \mathrm{d})$ is a compact set endowed with a symmetric, Lipschitz cost function $\mathbf{C}(x, y)$. For $\varepsilon > 0$, the associated *Gibbs kernel* is defined through:

$$k_\varepsilon : (x, y) \in \mathcal{X} \times \mathcal{X} \mapsto \exp\big(-\mathbf{C}(x, y)/\varepsilon\big)\ . \tag{A.37}$$

Crucially, we now assume that k_ε is a *positive universal* kernel on the space of signed Radon measures.

Definition A.1 (Sinkhorn negentropy)**.** Under the assumptions above, we define the Sinkhorn negentropy of a probability Radon measure $\alpha \in \mathcal{M}_1^+(\mathcal{X})$ through:

$$\mathrm{F}_\varepsilon(\alpha) \overset{\text{def.}}{=} -\tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha, \alpha)\ . \tag{A.38}$$

The following proposition is the cornerstone of our approach to prove the positivity of S_ε, providing an alternative expression of F_ε. Its proof relies on a change of variables $\mu = \exp(f/\varepsilon)\,\alpha$ in (A.29) that is detailed in Section A.4.3.

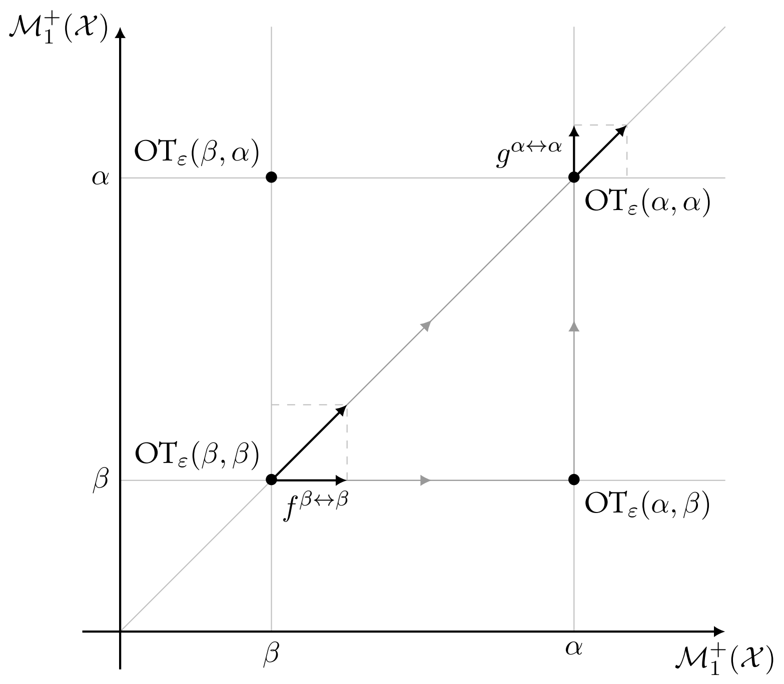

$\mathcal{M}_1^+(\mathcal{X})$
α
$\mathrm{OT}_\varepsilon(\beta,\alpha)$
$g^{\alpha\leftrightarrow\alpha}$
$\mathrm{OT}_\varepsilon(\alpha,\alpha)$
β
$\mathrm{OT}_\varepsilon(\beta,\beta)$
$f^{\beta\leftrightarrow\beta}$
$\mathrm{OT}_\varepsilon(\alpha,\beta)$
β
α
$\mathcal{M}_1^+(\mathcal{X})$

Figure A.1: Sketch for the proof of Theorem 3.1. On the product space of dual pairs $\mathcal{M}_1^+(\mathcal{X}) \times \mathcal{M}_1^+(\mathcal{X})$, the regularized cost OT_ε is *convex* with respect to each variable: this allows us to show (A.41-A.42) and eventually that $H_\varepsilon \leqslant \mathrm{S}_\varepsilon$. Crucially, we have also seen with Proposition A.4 that OT_ε is *concave* on the diagonal: we show that $0 \leqslant H_\varepsilon$ and can then state our main result: Sinkhorn divergences define positive Loss functions over the space of probability measures.

Proposition A.3. *Let* $(\mathcal{X}, \mathrm{d})$ *be a compact set endowed with a symmetric, Lipschitz cost function* $\mathrm{C}(x, y)$ *that induces a* positive *kernel* k_ε*. Then, for* $\varepsilon > 0$ *and* $\alpha \in \mathcal{M}_1^+(\mathcal{X})$*, one has:*

$$\tfrac{1}{\varepsilon}\mathrm{F}_\varepsilon(\alpha) + \tfrac{1}{2} = \min_{\mu \in \mathcal{M}^+(\mathcal{X})} \langle \alpha, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu} \rangle + \tfrac{1}{2}\|\mu\|_{k_\varepsilon}^2 \,. \tag{A.39}$$

The following proposition, whose proof can be found in Section A.4.4, leverages the alternative expression (A.39) to ensure the convexity of F_ε.

Proposition A.4. *Under the same hypotheses as Proposition A.3,* F_ε *is a* strictly convex *functional on* $\mathcal{M}_1^+(\mathcal{X})$*.*

We now define an auxiliary "Hausdorff" divergence that can be interpreted as an OT_ε loss with *decoupled* dual potentials.

Definition A.2 (Hausdorff divergence)**.** Thanks to Proposition A.2, the Sinkhorn negentropy F_ε is differentiable in the sense of (A.35). For any probability measures $\alpha, \beta \in \mathcal{M}_1^+(\mathcal{X})$ and regularization strength $\varepsilon > 0$, we can thus define:

$$H_\varepsilon(\alpha,\beta) \overset{\text{def.}}{=} \tfrac{1}{2}\langle \alpha - \beta, \nabla \mathrm{F}_\varepsilon(\alpha) - \nabla \mathrm{F}_\varepsilon(\beta)\rangle \ \geqslant\ 0\,. \tag{A.40}$$

It is the symmetric Bregman divergence induced by the strictly convex functional F_ε (Bregman, 1967) and is therefore a positive definite quantity.

A.2.3 Proof of the Theorem

We are now ready to conclude. First, remark that the dual expression (A.29) of $\mathrm{OT}_\varepsilon(\alpha,\beta)$ as a maximization of linear forms ensures that $\mathrm{OT}_\varepsilon(\alpha,\beta)$ is *convex* with respect to α and with respect to β (but *not* jointly convex if $\varepsilon > 0$). S_ε is thus convex with respect to both inputs α and β as a sum of the functions OT_ε and F_ε – see Proposition A.4.

Convexity also implies that:

$$\mathrm{OT}_\varepsilon(\alpha,\alpha) + \langle \beta - \alpha,\ \nabla_2 \mathrm{OT}_\varepsilon(\alpha,\alpha)\rangle \ \leqslant \mathrm{OT}_\varepsilon(\alpha,\beta)\ , \tag{A.41}$$
$$\mathrm{OT}_\varepsilon(\beta,\beta) + \langle \alpha - \beta,\ \nabla_1 \mathrm{OT}_\varepsilon(\beta,\beta)\rangle \ \leqslant \mathrm{OT}_\varepsilon(\alpha,\beta)\ . \tag{A.42}$$

Using (A.36) to get $\nabla_2 \mathrm{OT}_\varepsilon(\alpha,\alpha) = -\nabla \mathrm{F}_\varepsilon(\alpha)$, $\nabla_1 \mathrm{OT}_\varepsilon(\beta,\beta) = -\nabla \mathrm{F}_\varepsilon(\beta)$ and summing the above inequalities, we show that $H_\varepsilon \leqslant \mathrm{S}_\varepsilon$, which implies (A.8). To prove (A.9), note that $\mathrm{S}_\varepsilon(\alpha,\beta) = 0 \Rightarrow H_\varepsilon(\alpha,\beta) = 0$, which implies that $\alpha = \beta$ since F_ε is a *strictly* convex functional. Finally, we show that S_ε metrizes the convergence in law (A.10) in Section A.4.5. □

A.3 Lemmas: standard results

Before detailing our proofs, we first recall some well-known results regarding the Kullback-Leibler divergence and the SoftMin operator defined in (A.33).

A.3.1 The Kullback-Leibler divergence

First properties. For any pair of Radon measures $\alpha, \beta \in \mathcal{M}^+(\mathcal{X})$ on the compact metric set $(\mathcal{X}, \mathrm{d})$, the Kullback-Leibler divergence is defined through:

$$\mathrm{KL}(\alpha,\beta) \ \overset{\text{def.}}{=}\ \begin{cases} \langle \alpha,\, \log \frac{\mathrm{d}\alpha}{\mathrm{d}\beta} - 1\rangle + \langle \beta,\, 1\rangle & \text{if } \alpha \ll \beta \\ \quad +\infty & \text{otherwise.} \end{cases} \tag{A.43}$$

It can be rewritten as an f-divergence associated to:

$$\psi : x \in \mathbb{R}_{\geqslant 0} \mapsto x\,\log(x) - x + 1\ \in \mathbb{R}_{\geqslant 0}\ , \tag{A.44}$$

with $0 \cdot \log(0) = 0$, as:

$$\mathrm{KL}(\alpha, \beta) \;=\; \begin{cases} \langle \beta,\, \psi(\frac{\mathrm{d}\alpha}{\mathrm{d}\beta}) \rangle & \text{if } \alpha \ll \beta \\ \quad +\infty & \text{otherwise.} \end{cases} \tag{A.45}$$

Since ψ is a strictly convex function with a unique global minimum at $\psi(1) = 0$, we thus get that $\mathrm{KL}(\alpha, \beta) \geqslant 0$ with equality iff $\alpha = \beta$.

Dual formulation. The convex conjugate of ψ is defined for $u \in \mathbb{R}$ by:

$$\psi^*(u) \;\overset{\text{def.}}{=}\; \sup_{x>0} \left(xu - \psi(x)\right) \;=\; e^u - 1, \tag{A.46}$$

$$\text{and we have} \quad \psi(x) + \psi^*(u) \;\geqslant\; xu \tag{A.47}$$

for all $(x, u) \in \mathbb{R}_{\geqslant 0} \times \mathbb{R}$, with equality if $x > 0$ and $u = \log(x)$. This allows us to rewrite the Kullback-Leibler divergence as the solution of a dual concave problem:

Proposition A.5 (Dual formulation of KL). *Under the assumptions above,*

$$\mathrm{KL}(\alpha, \beta) \;=\; \sup_{h \in \mathcal{F}_b(\mathcal{X}, \mathbb{R})} \langle \alpha,\, h \rangle - \langle \beta,\, e^h - 1 \rangle \tag{A.48}$$

where $\mathcal{F}_b(\mathcal{X}, \mathbb{R})$ is the space of bounded measurable functions from $\mathcal{X}$ to $\mathbb{R}$.

Proof. Lower bound on the sup. If α is not absolutely continuous with respect to β, there exists a Borel set A such that $\alpha(A) > 0$ and $\beta(A) = 0$. Consequently, for $h = \lambda\, \mathbf{1}_A$,

$$\langle \alpha,\, h \rangle - \langle \beta,\, e^h - 1 \rangle \;=\; \lambda\, \alpha(A) \;\xrightarrow{\lambda \to +\infty}\; +\infty\;. \tag{A.49}$$

Otherwise, if $\alpha \ll \beta$, we define $h_* = \log \frac{\mathrm{d}\alpha}{\mathrm{d}\beta}$ and see that:

$$\langle \alpha,\, h_* \rangle - \langle \beta,\, e^{h_*} - 1 \rangle \;=\; \mathrm{KL}(\alpha, \beta)\;. \tag{A.50}$$

If $h_n = \log(\frac{\mathrm{d}\alpha}{\mathrm{d}\beta})\, \mathbf{1}_{1/n \leqslant \mathrm{d}\alpha/\mathrm{d}\beta \leqslant n} \in \mathcal{F}_b(\mathcal{X}, \mathbb{R})$, the monotone and dominated convergence theorems then allow us to show that:

$$\langle \alpha,\, h_n \rangle - \langle \beta,\, e^{h_n} - 1 \rangle \;\xrightarrow{n \to +\infty}\; \mathrm{KL}(\alpha, \beta)\;. \tag{A.51}$$

Upper bound on the sup. If $h \in \mathcal{F}_b(\mathcal{X}, \mathbb{R})$ and $\alpha \ll \beta$, combining (A.45) and (A.47) allow us to show that:

$$\mathrm{KL}(\alpha, \beta) - \langle \alpha,\, h \rangle + \langle \beta,\, e^h - 1 \rangle \;=\; \langle \beta,\, \psi(\tfrac{\mathrm{d}\alpha}{\mathrm{d}\beta}) + \psi^*(h) - h\tfrac{\mathrm{d}\alpha}{\mathrm{d}\beta} \rangle \;\geqslant\; 0\;. \tag{A.52}$$

The optimal value of $\langle \alpha,\, h \rangle - \langle \beta,\, e^h - 1 \rangle$ is bounded above and below by $\mathrm{KL}(\alpha, \beta)$: we get (A.48). □

Convexity. Since $\langle \alpha, h \rangle - \langle \beta, e^h - 1 \rangle$ is a convex function of (α, β), taking the supremum over test functions $h \in \mathcal{F}_b(\mathcal{X}, \mathbb{R})$ defines a *convex* divergence:

Proposition A.6. *The* KL *divergence is a (jointly) convex function on $\mathcal{M}^+(\mathcal{X}) \times \mathcal{M}^+(\mathcal{X})$.*

Going further, the density of continuous functions in the space of bounded measurable functions allows us to restrict the optimization domain:

Proposition A.7. *Under the same assumptions,*

$$\mathrm{KL}(\alpha, \beta) = \sup_{h \in \mathcal{C}(\mathcal{X}, \mathbb{R})} \langle \alpha, h \rangle - \langle \beta, e^h - 1 \rangle \tag{A.53}$$

where $\mathcal{C}(\mathcal{X}, \mathbb{R})$ is the space of (bounded) continuous functions on the compact set $\mathcal{X}$.

Proof. Let $h = \sum_{i \in I} h_i \, \mathbf{1}_{A_i}$ be a simple Borel function on $\mathcal{X}$, and let us choose some error margin $\delta > 0$. Since α and β are Radon measures, for any i in the finite set of indices I, there exists a compact set K_i and an open set V_i such that $K_i \subset A_i \subset V_i$ and:

$$\sum_{i \in I} \max[\, \alpha(V_i \backslash K_i) \, , \, \beta(V_i \backslash K_i) \,] \leqslant \delta \, . \tag{A.54}$$

Moreover, for any $i \in I$, there exists a continuous function φ_i such that $\mathbf{1}_{K_i} \leqslant \varphi_i \leqslant \mathbf{1}_{V_i}$. The continuous function $g = \sum_{i \in I} h_i \varphi_i$ is then such that:

$$|\langle \alpha, g - h \rangle| \leqslant \|h\|_\infty \, \delta \qquad \text{and} \qquad |\langle \beta, e^g - e^h \rangle| \leqslant \|e^h\|_\infty \, \delta \tag{A.55}$$

so that:

$$|\, (\langle \alpha, h \rangle - \langle \beta, e^h - 1 \rangle) \, - \, (\langle \alpha, g \rangle - \langle \beta, e^g - 1 \rangle) \,| \tag{A.56}$$

$$\leqslant (\|h\|_\infty + \|e^h\|_\infty) \, \delta \, . \tag{A.57}$$

As we let our simple function approach any measurable function in $\mathcal{F}_b(\mathcal{X}, \mathbb{R})$, choosing δ arbitrarily small, we then get (A.53) through (A.48). □

Convergence in law. We can then show that the Kullback-Leibler divergence is weakly lower semi-continuous:

Proposition A.8. *If $\alpha_n \rightharpoonup \alpha$ and $\beta_n \rightharpoonup \beta$ are weakly converging sequences in $\mathcal{M}^+(\mathcal{X})$, we get:*

$$\liminf_{n \to +\infty} \mathrm{KL}(\alpha_n, \beta_n) \geqslant \mathrm{KL}(\alpha, \beta) \, . \tag{A.58}$$

Proof. According to (A.53), the KL divergence is defined as a pointwise supremum of weakly continuous applications:

$$\varphi_h \, : \, (\alpha, \beta) \mapsto \langle \alpha, h \rangle - \langle \beta, e^h - 1 \rangle \, , \tag{A.59}$$

for $h \in \mathcal{C}(\mathcal{X}, \mathbb{R})$. It is thus lower semi-continuous for the convergence in law. □

A.3.2 SoftMin operator

Proposition A.9 (The SoftMin interpolates between a minimum and a sum). *Under the assumptions of the definition* (A.33), *notably the fact that α is a probability measure, we get that:*

$$\underset{x\sim\alpha}{\min_\varepsilon}\ \varphi(x) \xrightarrow{\varepsilon\to 0} \min_{x\in\mathrm{Supp}(\alpha)} \varphi(x) \tag{A.60}$$

$$\xrightarrow{\varepsilon\to+\infty} \quad \langle\alpha,\varphi\rangle\ . \tag{A.61}$$

If φ and ψ are two continuous functions in $\mathcal{C}(\mathcal{X})$ such that $\varphi \leqslant \psi$, then:

$$\underset{x\sim\alpha}{\min_\varepsilon}\ \varphi(x) \leqslant \underset{x\sim\alpha}{\min_\varepsilon}\ \psi(x)\ . \tag{A.62}$$

Finally, if $K\in\mathbb{R}$ is constant with respect to x, we have that:

$$\underset{x\sim\alpha}{\min_\varepsilon}\ [K+\varphi(x)]\ = K + \underset{x\sim\alpha}{\min_\varepsilon}\ [\varphi(x)]\ . \tag{A.63}$$

Proposition A.10 (The SoftMin operator is continuous). *Let (α_n) be a sequence of probability measures converging* weakly *towards α, and (φ_n) be a sequence of continuous functions that converges* uniformly *towards φ. Then, for $\varepsilon > 0$, the SoftMin of the values of φ_n on α_n converges towards the SoftMin of the values of φ on α, i.e.*

$$(\alpha_n \rightharpoonup \alpha,\ \varphi_n \xrightarrow{\|\cdot\|_\infty} \varphi) \implies \underset{x\sim\alpha_n}{\min_\varepsilon}\ \varphi_n(x) \to \underset{x\sim\alpha}{\min_\varepsilon}\ \varphi(x)\ . \tag{A.64}$$

A.4 Lemmas: original proofs

A.4.1 Dual potentials

Proof of Proposition A.1. We first state some important properties of solutions (f,g) to the dual problem (A.29). Please note that these results hold under the assumption that $(\mathcal{X},\mathrm{d})$ is a compact metric space, endowed with a *ground cost* function $\mathrm{C}:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ that is κ-Lipschitz with respect to both of its input variables.

The existence of an optimal pair (f,g) of potentials that reaches the maximal value of the dual objective is a standard result for entropic OT. It can be proved using the contractance of the Sinkhorn map T, defined in (A.32), for the Hilbert projective metric (Franklin and Lorenz, 1989).

While optimal potentials are only defined (α,β)-a.e., as highlighted in Proposition A.1, they are extended to the whole domain $\mathcal{X}$ by imposing, similarly to the classical theory of OT (Santambrogio, 2015, Remark 1.13), that they satisfy:

$$f = T(\beta,g) \quad\text{and}\quad g = T(\alpha,f)\ , \tag{A.65}$$

with T defined in (A.32). We thus assume in the following that this condition holds. The propositions below study the uniqueness and the smoothness (with respect to the spacial position and with respect to the input measures) of these functions (f, g) defined on the whole space.

Proposition A.11 (Uniqueness of the dual potentials up to an additive constant)**.** *Let* (f_0, g_0) *and* (f_1, g_1) *be two optimal pairs of dual potentials for a problem* $\mathrm{OT}_\varepsilon(\alpha,\beta)$ *that satisfy* (A.65)*. Then, there exists a constant* $K \in \mathbb{R}$ *such that:*

$$f_0 = f_1 + K \quad \textit{and} \quad g_0 = g_1 - K. \tag{A.66}$$

Proof. For $t \in [0,1]$, let us define $f_t = f_0 + t(f_1 - f_0)$, $g_t = g_0 + t(g_1 - g_0)$ and:

$$\varphi(t) \;= \langle \alpha, f_t\rangle + \langle \beta, g_t\rangle - \varepsilon\langle \alpha\otimes\beta, \exp\big(\tfrac{1}{\varepsilon}(f_t \oplus g_t - \mathbf{C})\big) - 1\rangle\ , \tag{A.67}$$

the value of the dual objective between the two optimal pairs. As φ is a concave function bounded above by $\varphi(0) = \varphi(1) = \mathrm{OT}_\varepsilon(\alpha,\beta)$, it is *constant* with respect to t. Hence, for all t in $[0,1]$,

$$\begin{aligned} 0 &= \varphi''(t) && \text{(A.68)}\\ &= -\tfrac{1}{\varepsilon}\langle \alpha\otimes\beta, e^{(f_t\oplus g_t - \mathbf{C})/\varepsilon}((f_1-f_0)\oplus(g_1-g_0))^2\rangle\ . && \text{(A.69)}\end{aligned}$$

This is only possible if, $\alpha\otimes\beta$-a.e. in (x,y),

$$\big(f_1(x) - f_0(x) + g_1(y) - g_0(y)\big)^2 = 0\ , \tag{A.70}$$

i.e. there exists a constant $K \in \mathbb{R}$ such that:

$$\begin{aligned} f_1(x) - f_0(x) &= +K \quad \alpha\text{-a.e.} && \text{(A.71)}\\ g_1(y) - g_0(y) &= -K \quad \beta\text{-a.e.} && \text{(A.72)}\end{aligned}$$

As we extend the potentials through (A.65), the SoftMin operator commutes with the addition of K (A.63) and lets our result hold on the whole feature space. □

Proposition A.12 (Lipschitz property)**.** *The optimal potentials* (f, g) *of the dual problem* (A.29) *are both* κ*-Lipschitz functions on the feature space* $(\mathcal{X}, \mathrm{d})$*, where* κ *is the Lipschitz constant of* $\mathbf{C}$*.*

Proof. According to (A.65), f is a SoftMin combination of κ-Lipschitz functions of the variable x; using the algebraic properties of the SoftMin operator detailed in (A.62-A.63), one can thus show that f is a κ-Lipschitz function on the feature space. The same argument holds for g. □

Proposition A.13 (The dual potentials vary continuously with the input measures). *Let $\alpha_n \rightharpoonup \alpha$ and $\beta_n \rightharpoonup \beta$ be weakly converging sequences of measures in $\mathcal{M}_1^+(\mathcal{X})$. Given some arbitrary anchor point $x_o \in \mathcal{X}$, let us denote by (f_n, g_n) the (unique) sequence of optimal potentials for $\mathrm{OT}_\varepsilon(\alpha_n, \beta_n)$ such that $f_n(x_o) = 0$.*

Then, f_n and g_n converge uniformly towards the unique pair of optimal potentials (f, g) for $\mathrm{OT}_\varepsilon(\alpha, \beta)$ such that $f(x_o) = 0$. Up to the value at the anchor point x_o, we thus have that:

$$(\alpha_n \rightharpoonup \alpha,\ \beta_n \rightharpoonup \beta) \implies (f_n \xrightarrow{\|\cdot\|_\infty} f,\ g_n \xrightarrow{\|\cdot\|_\infty} g)\ . \tag{A.73}$$

Proof. For all n in $\mathbb{N}$, the potentials f_n and g_n are κ-Lipschitz functions on the compact, bounded set $\mathcal{X}$. As $f_n(x_o)$ is set to zero, we can bound $|f_n|$ on $\mathcal{X}$ by κ times the diameter of $\mathcal{X}$; combining this with (A.65), we can then produce a uniform bound on both f_n and g_n: there exists a constant $M \in \mathbb{R}$ such that:

$$\forall\, n \in \mathbb{N}, \forall\, x \in \mathcal{X}, -M \leqslant f_n(x), g_n(x) \leqslant +M\ . \tag{A.74}$$

Being equicontinuous and uniformly bounded on the compact set $\mathcal{X}$, the sequence $(f_n, g_n)_n$ satisfies the hypotheses of the Ascoli-Arzela theorem: there exists a subsequence $(f_{n_k}, g_{n_k})_k$ that converges uniformly towards a pair (f, g) of continuous functions. As k tends to infinity, we see that $f(x_o) = 0$ and, using the continuity of the SoftMin operator (Proposition A.10) on the optimality equations (A.31), we show that (f, g) is an optimal pair for $\mathrm{OT}_\varepsilon(\alpha, \beta)$.

Now, according to Proposition A.11, such a limit pair of optimal potentials (f, g) is *unique*. $(f_n, g_n)_n$ is thus a *compact* sequence with a *single* possible adherence value: it has to converge, uniformly, towards (f, g). □

A.4.2 Differentiability of the OT_ε loss

Proof of Proposition A.2. This demonstration is inspired by (Santambrogio, 2015, Proposition 7.17). Let us consider some measures α, β, variations $\delta\alpha$, $\delta\beta$ and times t in a neighborhood of 0, as in the statement above. We define $\alpha_t = \alpha + t\delta\alpha$, $\beta_t = \beta + t\delta\beta$ and denote by Δ_t the variation ratio given by:

$$\Delta_t \overset{\text{def.}}{=} \frac{\mathrm{OT}_\varepsilon(\alpha_t, \beta_t) - \mathrm{OT}_\varepsilon(\alpha, \beta)}{t}\ . \tag{A.75}$$

Using the very definition of OT_ε and the continuity property of Proposition A.13, we now provide lower and upper bounds on Δ_t as t goes to 0.

Weak-$\star$ continuity. As written in (A.34), $\mathrm{OT}_\varepsilon(\alpha, \beta)$ can be computed through a straightforward, *continuous* expression that does not depend on the value of the optimal dual potentials (f, g) at the anchor point x_o:

$$\mathrm{OT}_\varepsilon(\alpha, \beta) = \langle \alpha, f \rangle + \langle \beta, g \rangle\ . \tag{A.76}$$

Combining this equation with Proposition A.13 (that guarantees the *uniform* convergence of potentials for weakly converging sequences of probability measures) allows us to show that OT_ε is continuous with respect to the convergence in law.

Lower bound. First, let us remark that (f, g) is a *suboptimal* pair of dual potentials for $\mathrm{OT}_\varepsilon(\alpha_t, \beta_t)$. Hence,

$$\mathrm{OT}_\varepsilon(\alpha_t, \beta_t) \geqslant \langle \alpha_t, f\rangle + \langle \beta_t, g\rangle - \varepsilon\langle \alpha_t \otimes \beta_t, \exp\left(\tfrac{1}{\varepsilon}(f \oplus g - \mathbf{C})\right) - 1\rangle \quad \text{(A.77)}$$

and thus, since:

$$\mathrm{OT}_\varepsilon(\alpha, \beta) = \langle \alpha, f\rangle + \langle \beta, g\rangle - \varepsilon\langle \alpha \otimes \beta, \exp(\tfrac{1}{\varepsilon}(f \oplus g - \mathbf{C})) - 1\rangle\ , \quad \text{(A.78)}$$

one has:

$$\begin{aligned} \Delta_t &\geqslant \langle \delta\alpha, f\rangle + \langle \delta\beta, g\rangle & \text{(A.79)} \\ &\quad - \varepsilon\langle \delta\alpha \otimes \beta + \alpha \otimes \delta\beta, \exp(\tfrac{1}{\varepsilon}(f \oplus g - \mathbf{C}))\rangle + o(1) \\ &\geqslant \langle \delta\alpha, f - \varepsilon\rangle + \langle \delta\beta, g - \varepsilon\rangle + o(1)\ , & \text{(A.80)} \end{aligned}$$

since g and f satisfy the optimality equations (A.31).

Upper bound. Conversely, let us denote by (g_t, f_t) the optimal pair of potentials for $\mathrm{OT}_\varepsilon(\alpha_t, \beta_t)$ satisfying $g_t(x_o) = 0$ for some arbitrary anchor point $x_o \in \mathcal{X}$. As (f_t, g_t) are suboptimal potentials for $\mathrm{OT}_\varepsilon(\alpha, \beta)$, we get that:

$$\mathrm{OT}_\varepsilon(\alpha, \beta) \geqslant \langle \alpha, f_t\rangle + \langle \beta, g_t\rangle - \varepsilon\langle \alpha \otimes \beta, \exp\left(\tfrac{1}{\varepsilon}(f_t \oplus g_t - \mathbf{C})\right) - 1\rangle \quad \text{(A.81)}$$

and thus, since:

$$\mathrm{OT}_\varepsilon(\alpha_t, \beta_t) = \langle \alpha_t, f_t\rangle + \langle \beta_t, g_t\rangle - \varepsilon\langle \alpha_t \otimes \beta_t, \exp(\tfrac{1}{\varepsilon}(f_t \oplus g_t - \mathbf{C})) - 1\rangle, \quad \text{(A.82)}$$

we can show that:

$$\begin{aligned} \Delta_t &\leqslant \langle \delta\alpha, f_t\rangle + \langle \delta\beta, g_t\rangle & \text{(A.83)} \\ &\quad - \varepsilon\langle \delta\alpha \otimes \beta_t + \alpha_t \otimes \delta\beta, \exp(\tfrac{1}{\varepsilon}(f_t \oplus g_t - \mathbf{C}))\rangle + o(1) \\ &\leqslant \langle \delta\alpha, f_t - \varepsilon\rangle + \langle \delta\beta, g_t - \varepsilon\rangle + o(1)\ . & \text{(A.84)} \end{aligned}$$

Conclusion. Now, let us remark that as t goes to 0 :

$$\alpha + t\delta\alpha \rightharpoonup \alpha \qquad \text{and} \qquad \beta + t\delta\beta \rightharpoonup \beta\ . \quad \text{(A.85)}$$

Thanks to Proposition A.13, we thus know that f_t and g_t converge uniformly towards f and g. Combining the lower and upper bound, we get:

$$\Delta_t \xrightarrow{t \to 0} \langle \delta\alpha, f - \varepsilon\rangle + \langle \delta\beta, g - \varepsilon\rangle = \langle \delta\alpha, f\rangle + \langle \delta\beta, g\rangle\ , \quad \text{(A.86)}$$

since $\delta\alpha$ and $\delta\beta$ both have an overall mass that sums up to zero. Otherwise said, f and g are the gradients of $\mathrm{OT}_\varepsilon(\alpha, \beta)$ with respect to α and β. □

A.4.3 Function-measure change of variable

Proof of Proposition A.3. The definition of $\mathrm{OT}_\varepsilon(\alpha,\alpha)$ is that:

$$\mathrm{OT}_\varepsilon(\alpha,\alpha) = \max_{(f,g)\in\mathcal{C}(\mathcal{X})^2} \langle \alpha, f+g\rangle - \varepsilon\langle \alpha\otimes\alpha, e^{(f\oplus g-\mathrm{C})/\varepsilon} - 1\rangle\,. \tag{A.87}$$

Reduction of the problem. Thanks to the symmetry of this concave problem with respect to the variables f and g, we know that there exists a pair $(f, g=f)$ of optimal potentials on the diagonal, and:

$$\mathrm{OT}_\varepsilon(\alpha,\alpha) = \max_{f\in\mathcal{C}(\mathcal{X})} 2\langle \alpha, f\rangle - \varepsilon\langle \alpha\otimes\alpha, e^{(f\oplus f-\mathrm{C})/\varepsilon} - 1\rangle\,. \tag{A.88}$$

Thanks to the density of continuous functions in the set of simple measurable functions, just as in the proof of Proposition A.7, we show that this maximization can be done in the full set of measurable functions $\mathcal{F}_b(\mathcal{X},\mathbb{R})$:

$$\begin{aligned}\mathrm{OT}_\varepsilon(\alpha,\alpha) &= \max_{f\in\mathcal{F}_b(\mathcal{X},\mathbb{R})} 2\langle \alpha, f\rangle - \varepsilon\langle \alpha\otimes\alpha, e^{(f\oplus f-\mathrm{C})/\varepsilon} - 1\rangle & \text{(A.89)}\\ &= \max_{f\in\mathcal{F}_b(\mathcal{X},\mathbb{R})} 2\langle \alpha, f\rangle - \varepsilon\langle \exp(f/\varepsilon)\alpha, k_\varepsilon\star\exp(f/\varepsilon)\alpha\rangle + \varepsilon\,, & \text{(A.90)}\end{aligned}$$

where $\star$ denotes the smoothing (convolution) operator defined through:

$$[k\star\mu](x) \;=\; \int_{\mathcal{X}} k(x,y)\,\mathrm{d}\mu(y) \tag{A.91}$$

for $k\in\mathcal{C}(\mathcal{X}\times\mathcal{X})$ and $\mu\in\mathcal{M}^+(\mathcal{X})$, as discussed in detail in Section 3.1.3.

Optimizing on measures. Through a change of variables:

$$\mu = \exp(f/\varepsilon)\,\alpha \qquad\qquad \text{i.e.} \qquad\qquad f=\varepsilon\log\tfrac{\mathrm{d}\mu}{\mathrm{d}\alpha}\,, \tag{A.92}$$

keeping in mind that α is a probability measure, we then get that:

$$\begin{aligned}\mathrm{OT}_\varepsilon(\alpha,\alpha) &= \varepsilon \max_{\mu\in\mathcal{M}^+(\mathcal{X}),\alpha\ll\mu\ll\alpha} 2\langle \alpha, \log\tfrac{\mathrm{d}\mu}{\mathrm{d}\alpha}\rangle - \langle \mu, k_\varepsilon\star\mu\rangle + 1 & \text{(A.93)}\\ -\tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) &= \varepsilon \min_{\mu\in\mathcal{M}^+(\mathcal{X}),\alpha\ll\mu\ll\alpha} \langle \alpha, \log\tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu}\rangle + \tfrac{1}{2}\langle \mu, k_\varepsilon\star\mu\rangle - \tfrac{1}{2}\,, & \text{(A.94)}\end{aligned}$$

where we optimize on positive measures $\mu\in\mathcal{M}^+(\mathcal{X})$ such that $\alpha\ll\mu$ and $\mu\ll\alpha$.

Expansion of the problem. As $k_\varepsilon(x,y) = \exp(-\mathbf{C}(x,y)/\varepsilon)$ is positive for all x and y in $\mathcal{X}$, we can remove the $\mu \ll \alpha$ constraint from the optimization problem:

$$-\tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) = \varepsilon \min_{\mu\in\mathcal{M}^+(\mathcal{X}),\alpha\ll\mu} \langle \alpha, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu}\rangle + \tfrac{1}{2}\langle \mu, k_\varepsilon \star \mu\rangle - \tfrac{1}{2}\ . \tag{A.95}$$

Indeed, restricting a positive measure μ to the support of α lowers the right-hand term $\langle \mu, k_\varepsilon \star \mu\rangle$ without having any influence on the density of α with respect to μ. Finally, let us remark that the $\alpha \ll \mu$ constraint is already encoded in the $\log \frac{\mathrm{d}\alpha}{\mathrm{d}\mu}$ operator, which blows up to infinity if α has no density with respect to μ; all in all, we thus have:

$$\begin{aligned} \mathrm{F}_\varepsilon(\alpha) &= -\tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) && \text{(A.96)}\\ &= \varepsilon \min_{\mu\in\mathcal{M}^+(\mathcal{X})} \langle \alpha, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu}\rangle + \tfrac{1}{2}\langle \mu, k_\varepsilon \star \mu\rangle - \tfrac{1}{2}\ , && \text{(A.97)} \end{aligned}$$

which is the desired result.

Existence of the optimal measure μ. In the expression above, the existence of an optimal μ is given as a consequence of the well-known fact from OT theory that optimal dual potentials f and g *exist*, so that the dual OT problem (A.29) is a max and not a mere supremum. Nevertheless, since this property of F_ε is key to the metrization of the convergence in law by Sinkhorn divergences, let us endow it with a **direct, alternative proof**:

Proposition A.14. *For any $\alpha \in \mathcal{M}_1^+(\mathcal{X})$, assuming that $\mathcal{X}$ is compact, there exists a unique $\mu_\alpha \in \mathcal{M}^+(\mathcal{X})$ such that:*

$$\mathrm{F}_\varepsilon(\alpha) \;=\; \varepsilon\left[\, \langle \alpha, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu_\alpha}\rangle + \tfrac{1}{2}\langle \mu_\alpha, k_\varepsilon \star \mu_\alpha\rangle - \tfrac{1}{2}\,\right]. \tag{A.98}$$

Moreover, $\alpha \ll \mu_\alpha \ll \alpha$.

Proof. Notice that for $(\alpha,\mu) \in \mathcal{M}_1^+(\mathcal{X}) \times \mathcal{M}^+(\mathcal{X})$,

$$\begin{aligned} \mathrm{E}_\varepsilon(\alpha,\mu) \;&\overset{\text{def.}}{=}\; \langle \alpha, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu}\rangle + \tfrac{1}{2}\langle \mu, k_\varepsilon \star \mu\rangle && \text{(A.99)}\\ &=\; \mathrm{KL}(\alpha,\mu) \;+\; \langle \alpha-\mu,\, 1\rangle \;+\; \tfrac{1}{2}\|\mu\|_{k_\varepsilon}^2 - \tfrac{1}{2}\ . && \text{(A.100)} \end{aligned}$$

Since $\mathbf{C}$ is bounded on the compact set $\mathcal{X}\times\mathcal{X}$ and α is a probability measure, we can already say that:

$$\tfrac{1}{\varepsilon}\mathrm{F}_\varepsilon(\alpha) \leqslant \mathrm{E}_\varepsilon(\alpha,\alpha) - \tfrac{1}{2} = \tfrac{1}{2}\langle \alpha\otimes\alpha,\, e^{-\mathbf{C}/\varepsilon}\rangle - \tfrac{1}{2} < +\infty\ . \tag{A.101}$$

Upper bound on the mass of μ. Since $\mathcal{X} \times \mathcal{X}$ is compact and $k_\varepsilon(x, y) > 0$, there exists $\eta > 0$ such that $k(x, y) > \eta$ for all x and y in $\mathcal{X}$. We thus get:

$$\|\mu\|^2_{k_\varepsilon} \;\geqslant\; \langle \mu,\, 1\rangle^2\, \eta \tag{A.102}$$

and show that:

$$\begin{aligned} \mathrm{E}_\varepsilon(\alpha, \mu) \;&\geqslant \langle \alpha - \mu,\, 1\rangle \;+\; \tfrac{1}{2}\|\mu\|^2_{k_\varepsilon} - \tfrac{1}{2} && \text{(A.103)} \\ &\geqslant \langle \mu,\, 1\rangle\, (\langle \mu,\, 1\rangle\, \eta \;-\; 1) - \tfrac{1}{2}\,. && \text{(A.104)} \end{aligned}$$

As we build a minimizing sequence (μ_n) for $\mathrm{F}_\varepsilon(\alpha)$, we can thus assume that $\langle \mu_n,\, 1\rangle$ is uniformly bounded by some constant $M > 0\,$.

Weak continuity. Crucially, the Banach-Alaoglu theorem asserts that:

$$\{\, \mu \in \mathcal{M}^+(\mathcal{X}) \mid \langle \mu,\, 1\rangle \leqslant M \,\} \tag{A.105}$$

is weakly compact; we can thus extract a weakly converging subsequence $\mu_{n_k} \rightharpoonup \mu_\infty$ from the minimizing sequence (μ_n). Using Proposition A.8 and the fact that k_ε is continuous on $\mathcal{X} \times \mathcal{X}$, we show that $\mu \mapsto \mathrm{E}_\varepsilon(\alpha, \mu)$ is a weakly lower semi-continuous function: $\mu_\infty = \mu_\alpha$ realizes the minimum of E_ε and we get our existence result.

Uniqueness. We assumed that our kernel k_ε is *positive universal*. The squared norm $\mu \mapsto \|\mu\|^2_{k_\varepsilon}$ is thus a strictly convex functional and using Proposition A.6, we can show that $\mu \mapsto \mathrm{E}_\varepsilon(\alpha, \mu)$ is *strictly* convex. This ensures that μ_α is uniquely defined. □

A.4.4 Concavity of $\mathrm{OT}_\varepsilon(\alpha, \alpha)$ on the diagonal

Proof of Proposition A.4. Let us take a pair of measures $\alpha_0 \neq \alpha_1$ in $\mathcal{M}_1^+(\mathcal{X})$, and $t \in (0, 1)$; according to Proposition A.14, there exists a pair of measures μ_0, μ_1 in $\mathcal{M}^+(\mathcal{X})$ such that

$$\begin{aligned} (1-t)\,\mathrm{F}_\varepsilon(\alpha_0) + t\,\mathrm{F}_\varepsilon(\alpha_1) &= \varepsilon\,(1-t)\,\mathrm{E}_\varepsilon(\alpha_0, \mu_0) + \varepsilon\, t\,\mathrm{E}_\varepsilon(\alpha_1, \mu_1) && \text{(A.106)} \\ &> \varepsilon\,\mathrm{E}_\varepsilon((1-t)\,\alpha_0 + t\,\alpha_1, (1-t)\,\mu_0 + t\,\mu_1) && \text{(A.107)} \\ &\geqslant \mathrm{F}_\varepsilon((1-t)\,\alpha_0 + t\,\alpha_1)\;, && \text{(A.108)} \end{aligned}$$

which is enough to conclude. To show the strict inequality, let us remark that:

$$\begin{aligned} &(1-t)\,\mathrm{E}_\varepsilon(\alpha_0, \mu_0) + t\,\mathrm{E}_\varepsilon(\alpha_1, \mu_1) && \text{(A.109)} \\ =\; &\mathrm{E}_\varepsilon((1-t)\,\alpha_0 + t\,\alpha_1, (1-t)\,\mu_0 + t\,\mu_1) \end{aligned}$$

would imply that $\mu_0 = \mu_1$, since $\mu \mapsto \|\mu\|_{k_\varepsilon}^2$ is strictly convex. As $\alpha \mapsto \mathrm{KL}(\alpha, \beta)$ is strictly convex on the set of measures α that are absolutely continuous with respect to β, we would then have $\alpha_0 = \alpha_1$ and a contradiction with our first hypothesis. □

A.4.5 Metrization of the convergence in law

The regularized OT cost is weakly continuous, and the uniform convergence for dual potentials ensures that H_ε and S_ε are both continuous too. Paired with (A.9), this property guarantees the convergence towards 0 of the Hausdorff and Sinkhorn divergences, as soon as $\alpha_n \rightharpoonup \alpha$.

Conversely, let us assume that $\mathrm{S}_\varepsilon(\alpha_n, \alpha) \rightarrow 0$ (resp. $H_\varepsilon(\alpha_n, \alpha)$). Any weak limit α_{n_∞} of a subsequence $(\alpha_{n_k})_k$ is equal to α: since our divergence is weakly continuous, we have $\mathrm{S}_\varepsilon(\alpha_{n_\infty}, \alpha) = 0$ (resp. $H_\varepsilon(\alpha_{n_\infty}, \alpha)$), and positive definiteness holds through (A.9).

In the meantime, since $\mathcal{X}$ is compact, the set of probability Radon measures $\mathcal{M}_1^+(\mathcal{X})$ is sequentially compact for the weak-$\star$ topology. α_n is thus a compact sequence with a unique adherence value: it converges, towards α. □

List of publications

1. Charlier, B., Feydy, J., Jacobs, D. W., and Trouvé, A. (2017b). Distortion minimizing geodesic subspaces in shape spaces and computational anatomy. In *European Congress on Computational Methods in Applied Sciences and Engineering*, pages 1135–1144. Springer

2. Feydy, J., Charlier, B., Vialard, F.-X., and Peyré, G. (2017). Optimal transport for diffeomorphic registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 291–299. Springer

3. Feydy, J. and Trouvé, A. (2018). Global divergences between measures: from Hausdorff distance to optimal transport. In *International Workshop on Shape in Medical Imaging*, pages 102–115. Springer

4. Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. (2019b). Interpolating between optimal transport and MMD using Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690

5. Feydy, J., Roussillon, P., Trouvé, A., and Gori, P. (2019a). Fast and scalable optimal transport for brain tractograms. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 636–644. Springer

6. Séjourné, T., Feydy, J., Vialard, F.-X., Trouvé, A., and Peyré, G. (2019). Sinkorn divergences for unbalanced Optimal Transport. *arXiv preprint arXiv:1910.12958*

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Addis, P., Melis, P., Cannas, R., Secci, M., Tinti, F., Piccinetti, C., and Cau, A. (2010). A morphometric approach for the analysis of body shape in bluefin tuna: preliminary results. *Collect. Vol. Sci. Pap. ICCAT*, 65(3):982–987.

Agarwal, A., Jawahar, C., and Narayanan, P. (2005). A survey of planar homography estimation techniques. *Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12*.

Agueh, M. and Carlier, G. (2011). Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1988). *Network flows*. Alfred P. Sloan School of Management, Massachusetts.

Aiger, D., Mitra, N. J., and Cohen-Or, D. (2008). 4-points congruent sets for robust pairwise surface registration. *ACM transactions on graphics (TOG)*, 27(3):85.

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Bleecher Snyder, J., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brébisson, A., Breuleux, O., Carrier, P.-L., Cho, K., Chorowski, J., Christiano, P., Cooijmans, T., Côté, M.-A., Côté, M., Courville, A., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Ebrahimi Kahou, S., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I., Graham, M., Gulcehre, C., Hamel, P., Harlouchet, I., Heng, J.-P., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrancois, S., Lemieux, S., Léonard, N., Lin, Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P.-A., Mastropietro, O., McGibbon, R. T., Memisevic, R., van Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., Simon, E., Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. (2001). Point set surfaces. In *Proceedings of the Conference on Visualization'01*, pages 21–28. IEEE Computer Society.

Ali-Hamadi, D., Liu, T., Gilles, B., Kavan, L., Faure, F., Palombi, O., and Cani, M.-P. (2013). Anatomy transfer. *ACM Transactions on Graphics (TOG)*, 32(6):188.

Altschuler, J., Bach, F., Rudi, A., and Weed, J. (2018a). Approximating the quadratic transportation metric in near-linear time. *arXiv preprint arXiv:1810.10046*.

Altschuler, J., Bach, F., Rudi, A., and Weed, J. (2018b). Massively scalable Sinkhorn distances via the Nyström method. *arXiv preprint arXiv:1812.05189*.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Amari, S.-i. and Nagaoka, H. (2007). *Methods of information geometry*, volume 191. American Mathematical Soc.

Amenta, N. and Kil, Y. J. (2004). Defining point-set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 23(3), pages 264–270. ACM.

Arguillere, S. (2014). *Géométrie sous-Riemannienne en dimension infinie et applications à l'analyse mathématique des formes*. PhD thesis, Université Paris VI – Pierre et Marie Curie.

Arguillere, S., Trélat, E., Trouvé, A., and Younes, L. (2015). Shape deformation analysis from the optimal control viewpoint. *Journal de mathématiques pures et appliquées*, 104(1):139–178.

Arguillere, S., Trélat, E., Trouvé, A., and Younes, L. (2016). Registration of multiple shapes using constrained optimal control. *SIAM Journal on Imaging Sciences*, 9(1):344–385.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223.

Arnold, V. (1966). Sur la géométrie différentielle des groupes de lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. In *Annales de l'institut Fourier*, volume 16, pages 319–361.

Arsigny, V., Commowick, O., Ayache, N., and Pennec, X. (2009). A fast and log-euclidean polyaffine framework for locally linear registration. *Journal of Mathematical Imaging and Vision*, 33(2):222–238.

Arsigny, V., Commowick, O., Pennec, X., and Ayache, N. (2006). A log-euclidean framework for statistics on diffeomorphisms. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 924–931. Springer.

Arsigny, V., Pennec, X., and Ayache, N. (2005). Polyrigid and polyaffine transformations: a novel geometrical tool to deal with non-rigid deformations–application to the registration of histological slices. *Medical image analysis*, 9(6):507–523.

Ashburner, J. (2007). A fast diffeomorphic image registration algorithm. *Neuroimage*, 38(1):95–113.

Ashburner, J. and Friston, K. J. (2011). Diffeomorphic registration using geodesic shooting and gauss–newton optimisation. *NeuroImage*, 55(3):954–967.

Aussal, M. and Bakry, M. (2019). The Fast and Free Memory method for the efficient computation of convolution kernels. *arXiv preprint arXiv:1909.05600*.

Avants, B. B., Epstein, C. L., Grossman, M., and Gee, J. C. (2008). Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain. *Medical image analysis*, 12(1):26–41.

Avants, B. B., Tustison, N., and Song, G. (2009). Advanced normalization tools (ANTS). *Insight journal*, 2:1–35.

Ayachit, U. (2015). *The ParaView guide: a parallel visualization application*. Kitware, Inc.

Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J., and Dalca, A. V. (2019). VoxelMorph: a learning framework for deformable medical image registration. *IEEE transactions on medical imaging*.

Baradat, A. (2019). *Incompressible optimal transport: dependence to the data and entropic regularization*. PhD thesis, Université Paris-Saclay.

Baradat, A. and Monsaingeon, L. (2018). Small noise limit and convexity for generalized incompressible flows, schrödinger problems, and optimal transport. *arXiv preprint arXiv:1810.12036*.

Barnes, J. and Hut, P. (1986). A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(6096):446.

Basser, P. J., Mattiello, J., and LeBihan, D. (1994). MR diffusion tensor spectroscopy and imaging. *Biophysical journal*, 66(1):259–267.

Bauer, M., Bruveris, M., Harms, P., and Michor, P. W. (2013). Geodesic distance for right invariant Sobolev metrics of fractional order on the diffeomorphism group. *Annals of Global Analysis and Geometry*, 44(1):5–21.

Bauer, M., Bruveris, M., and Michor, P. W. (2016). Uniqueness of the Fisher–Rao metric on the space of smooth densities. *Bulletin of the London Mathematical Society*, 48(3):499–506.

Bauer, M., Joshi, S., and Modin, K. (2015). Diffeomorphic density matching by optimal information transport. *SIAM Journal on Imaging Sciences*, 8(3):1718–1751.

Bauer, M., Joshi, S., and Modin, K. (2017). Diffeomorphic random sampling using optimal information transport. In *International Conference on Geometric Science of Information*, pages 135–142. Springer.

Bauer, M., Joshi, S., and Modin, K. (2018). Diffeomorphic density registration. *arXiv preprint arXiv:1807.05505*.

Beatson, R. and Greengard, L. (1997). A short course on fast multipole methods. *Wavelets, multilevel methods and elliptic PDEs*, 1:1–37.

Bebendorf, M. (2000). Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589.

Beg, M. F., Miller, M. I., Trouvé, A., and Younes, L. (2005). Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International journal of computer vision*, 61(2):139–157.

Ben Tanfous, A., Drira, H., and Ben Amor, B. (2018). Coding Kendall's shape trajectories for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2840–2849.

Benamou, J.-D. and Brenier, Y. (2000). A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393.

Benamou, J.-D., Carlier, G., Cuturi, M., Nenna, L., and Peyré, G. (2015). Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138.

Benamou, J.-D., Froese, B. D., and Oberman, A. M. (2014). Numerical solution of the optimal transportation problem using the Monge–Ampère equation. *Journal of Computational Physics*, 260:107–126.

Berman, R. J. (2017). The Sinkhorn algorithm, parabolic optimal transport and geometric Monge-Ampère equations. *arXiv preprint arXiv:1712.03082*.

Bernot, M., Caselles, V., and Morel, J.-M. (2008). *Optimal transportation networks: models and theory*. Springer.

Bertò, G., Bullock, D., Astolfi, P., Hayashi, S., Zigiotto, L., Annicchiarico, L., Corsini, F., De Benedictis, A., Sarubbo, S., Pestilli, F., et al. (2020). Classifyber, a robust streamline-based linear classifier for white matter bundle segmentation. *BioRxiv*.

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

Bertsekas, D. P. (1992). Auction algorithms for network flow problems: A tutorial introduction. *Computational optimization and applications*, 1(1):7–66.

Bertsekas, D. P. (2009). Auction algorithms. *Encyclopedia of optimization*, pages 128–132.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics.

Blanz, V., Vetter, T., et al. (1999). A morphable model for the synthesis of 3d faces. In *"ACM Transactions on Graphics (Proceedings of SIGGRAPH)"*, volume 99, pages 187–194.

Blinn, J. F. (1982). A generalization of algebraic surface drawing. *ACM transactions on graphics (TOG)*, 1(3):235–256.

Bône, A., Louis, M., Martin, B., and Durrleman, S. (2018). Deformetrica 4: an open-source software for statistical shape analysis. In *International Workshop on Shape in Medical Imaging*, pages 3–13. Springer.

Bonet, J. and Wood, R. D. (1997). *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press.

Bonneel, N. and Coeurjolly, D. (2019). Sliced partial optimal transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 38(4).

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. (2015). Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45.

Bookstein, F. L. (1991). *Morphometric tools for landmark data: geometry and biology*. Cambridge University Press.

Borgefors, G. (1984). Distance transformations in arbitrary dimensions. *Computer vision, graphics, and image processing*, 27(3):321–345.

Borgwardt, S. (2017). An LP-based, strongly polynomial 2-approximation algorithm for sparse Wasserstein barycenters. *arXiv preprint arXiv:1704.05491*.

Bossa, M., Hernandez, M., and Olmos, S. (2007). Contributions to 3D diffeomorphic atlas estimation: application to brain images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 667–674. Springer.

Bouaziz, S., Tagliasacchi, A., Li, H., and Pauly, M. (2016). Modern techniques and applications for real-time non-rigid registration. In *SIGGRAPH ASIA 2016 Courses*, page 11. ACM.

Bouaziz, S., Tagliasacchi, A., and Pauly, M. (2013). Sparse iterative closest point. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, pages 113–123. Eurographics Association.

Brandt, C., von Tycowicz, C., and Hildebrandt, K. (2016). Geometric flows of curves in shape space for processing motion of deformable objects. In *Computer Graphics Forum*, volume 35(2), pages 295–305. Wiley Online Library.

Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217.

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

Brenier, Y. (1999). Minimal geodesics on groups of volume-preserving maps and generalized solutions of the euler equations. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 52(4):411–452.

Broadbent, S. (1980). Simulating the ley hunter. *Journal of the Royal Statistical Society: Series A (General)*, 143(2):109–126.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Brunn, M., Himthani, N., Biros, G., Mehl, M., and Mang, A. (2019). Fast 3D diffeomorphic image registration on GPUs. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis*.

Bruveris, M. and Holm, D. D. (2015). Geometry of image registration: The diffeomorphism group and momentum maps. In *Geometry, Mechanics, and Dynamics*, pages 19–56. Springer.

Buet, B., Leonardi, G. P., and Masnou, S. (2015). Discrete varifolds: A unified framework for discrete approximations of surfaces and mean curvature. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 513–524. Springer.

Buet, B., Leonardi, G. P., and Masnou, S. (2017). A varifold approach to surface approximation. *Archive for Rational Mechanics and Analysis*, 226(2):639–694.

Burt, P. and Adelson, E. (1983). The Laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540.

Buttazzo, G., De Pascale, L., and Gori-Giorgi, P. (2012). Optimal-transport formulation of electronic density-functional theory. *Physical Review A*, 85(6):062502.

Cambier, L. and Darve, E. (2019). Fast low-rank kernel matrix factorization through skeletonized interpolation. *SIAM Journal on Scientific Computing*, 41(3):A1652–A1680.

Cannon, J. W., Floyd, W. J., Kenyon, R., Parry, W. R., et al. (1997). Hyperbolic geometry. *Flavors of geometry*, 31:59–115.

Charlier, B. (2011). *Étude des propriétés statistiques des moyennes de Fréchet dans des modèles de déformations pour l'analyse de courbes et d'images en grande dimension*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

Charlier, B., Charon, N., and Trouvé, A. (2017a). The fshape framework for the variability analysis of functional shapes. *Foundations of Computational Mathematics*, 17(2):287–357.

Charlier, B., Feydy, J., Jacobs, D. W., and Trouvé, A. (2017b). Distortion minimizing geodesic subspaces in shape spaces and computational anatomy. In *European Congress on Computational Methods in Applied Sciences and Engineering*, pages 1135–1144. Springer.

Charon, N. and Trouvé, A. (2013). The varifold representation of nonoriented shapes for diffeomorphic registration. *SIAM Journal on Imaging Sciences*, 6(4):2547–2580.

Charpentier, É., Ghys, E., and Lesne, A. (2010). *The scientific legacy of Poincaré*, volume 36. American Mathematical Soc.

Chazal, F., Cohen-Steiner, D., Lieutier, A., and Thibert, B. (2009). Stability of curvature measures. In *Computer Graphics Forum*, volume 28(5), pages 1485–1496. Wiley Online Library.

Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., et al. (2018). TVM: An automated end-to-end optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 578–594.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. (2018a). Scaling algorithms for unbalanced transport problems. *to appear in Mathematics of Computation*.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. (2018b). Unbalanced optimal transport: Dynamic and Kantorovich formulations. *Journal of Functional Analysis*, 274(11):3090–3123.

Chnafa, C., Mendez, S., and Nicoud, F. (2014). Image-based large-eddy simulation in a realistic left heart. *Computers & Fluids*, 94:173–187.

Chui, H. and Rangarajan, A. (2000). 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.

Ciliberto, C., Rosasco, L., and Rudi, A. (2016). A consistent regularization approach for structured prediction. In *Advances in neural information processing systems*, pages 4412–4420.

Conner-Simons, A. and Gordon, R. (2019). Using AI to predict breast cancer and personalize care. `http://news.mit.edu/2019/using-ai-predict-breast-cancer-and-personalize-care-0507`. MIT CSAIL.

Constantin, A. and Kolev, B. (2003). Geodesic flow on the diffeomorphism group of the circle. *Commentarii Mathematici Helvetici*, 78(4):787–804.

Cotar, C., Friesecke, G., and Klüppelberg, C. (2013). Density functional theory and optimal transportation with Coulomb cost. *Communications on Pure and Applied Mathematics*, 66(4):548–599.

Csiszár, I., Shields, P. C., et al. (2004). Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528.

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, pages 2292–2300.

Cuturi, M. and Doucet, A. (2014). Fast computation of wasserstein barycenters. In *International Conference on Machine Learning*, pages 685–693.

Delmonte, A., Mercier, C., Pallud, J., Bloch, I., and Gori, P. (2019). White matter multi-resolution segmentation using fuzzy set theory. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 459–462. IEEE.

Delon, J., Salomon, J., and Sobolevski, A. (2012). Local matching indicators for transport problems with concave costs. *SIAM Journal on Discrete Mathematics*, 26(2):801–827.

Deming, W. E. and Stephan, F. F. (1940). On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11(4):427–444.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Descoteaux, M. and Deriche, R. (2008). Mapping neuronal fiber crossings in the human brain. *SPIE Newsroom (August 2008)*, 2.

Dobrić, V. and Yukich, J. E. (1995). Asymptotics for transportation cost in high dimensions. *Journal of Theoretical Probability*, 8(1):97–118.

Dominitz, A. and Tannenbaum, A. (2009). Texture mapping via optimal mass transport. *IEEE transactions on visualization and computer graphics*, 16(3):419–433.

Dryden, I. L. and Mardia, K. V. (1998). *Statistical Shape Analysis: With Applications in R*, volume 995. John Wiley & Sons.

Dudley, R. M. (1969). The speed of mean Glivenko-Cantelli convergence. *The Annals of Mathematical Statistics*, 40(1):40–50.

Durrleman, S., Pennec, X., Trouvé, A., Braga, J., Gerig, G., and Ayache, N. (2013). Toward a comprehensive framework for the spatiotemporal statistical analysis of longitudinal shape data. *International journal of computer vision*, 103(1):22–59.

Durrleman, S., Prastawa, M., Charon, N., Korenberg, J. R., Joshi, S., Gerig, G., and Trouvé, A. (2014). Morphometry of anatomical shape complexes with dense deformations and sparse parameters. *NeuroImage*, 101:35–49.

Dutt, A. and Rokhlin, V. (1993). Fast Fourier transforms for nonequispaced data. *SIAM Journal on Scientific computing*, 14(6):1368–1393.

Ecabert, O., Peters, J., Walker, M. J., Ivanc, T., Lorenz, C., von Berg, J., Lessick, J., Vembar, M., and Weese, J. (2011). Segmentation of the heart and great vessels in CT images using a model-based adaptation framework. *Medical image analysis*, 15(6):863–876.

Eckstein, F., Wirth, W., and Nevitt, M. C. (2012). Recent advances in osteoarthritis imaging—the osteoarthritis initiative. *Nature Reviews Rheumatology*, 8(10):622.

Erlander, S. (1980). *Optimal spatial interaction and the gravity model*, volume 173. Springer Science & Business Media.

Federer, H. (1969). *Geometric measure theory*. Springer.

Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Finet, J., Fillion-Robin, J.-C., Pujol, S., Bauer, C., Jennings, D., Fennessy, F., Sonka, M., et al. (2012). 3D Slicer as an image computing platform for the quantitative imaging network. *Magnetic resonance imaging*, 30(9):1323–1341.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

Fernando, R. et al. (2004). *GPU gems: programming techniques, tips, and tricks for real-time graphics*, volume 590. Addison-Wesley Reading.

Ferraris, S., Lorenzi, M., Daga, P., Modat, M., and Vercauteren, T. (2016). Accurate small deformation exponential approximant to integrate large velocity fields: Application to image registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 17–24.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Feydy, J., Charlier, B., Vialard, F.-X., and Peyré, G. (2017). Optimal transport for diffeomorphic registration. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 291–299. Springer.

Feydy, J., Roussillon, P., Trouvé, A., and Gori, P. (2019a). Fast and scalable optimal transport for brain tractograms. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 636–644. Springer.

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. (2019b). Interpolating between optimal transport and MMD using Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690.

Feydy, J. and Trouvé, A. (2018). Global divergences between measures: from Hausdorff distance to optimal transport. In *International Workshop on Shape in Medical Imaging*, pages 102–115. Springer.

Fienberg, S. E. et al. (1970). An iterative procedure for estimation in contingency tables. *The Annals of Mathematical Statistics*, 41(3):907–917.

Fischl, B. (2012). Freesurfer. *Neuroimage*, 62(2):774–781.

Flamary, R. and Courty, N. (2017). POT python optimal transport library.

Fletcher, P. T., Lu, C., Pizer, S. M., and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE transactions on medical imaging*, 23(8):995–1005.

Fletcher, P. T., Venkatasubramanian, S., and Joshi, S. (2008). Robust statistics on Riemannian manifolds via the geometric median. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Fletcher, T. (2011). Geodesic regression on Riemannian manifolds. *Mathematical Foundations of Computational Anatomy*.

Forrow, A., Hütter, J.-C., Nitzan, M., Rigollet, P., Schiebinger, G., and Weed, J. (2019). Statistical optimal transport via factored couplings. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2454–2465.

Franklin, J. and Lorenz, J. (1989). On the scaling of multidimensional matrices. *Linear Algebra and its applications*, 114:717–735.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

Galerne, B., Leclaire, A., and Rabin, J. (2018). A texture synthesis model based on semi-discrete optimal transport in patch space. *SIAM Journal on Imaging Sciences*, 11(4):2456–2493.

Galichon, A. and Salanié, B. (2010). Matching with trade-offs: Revealed preferences over competing characteristics. *CEPR Discussion Paper No. DP7858*.

Gallouët, T. and Mérigot, Q. (2016). A Lagrangian scheme for the incompressible Euler equation using optimal transport. *arXiv:1605.00568*.

Gansner, E. R. and North, S. C. (2000). An open graph visualization system and its applications to software engineering. *Software: practice and experience*, 30(11):1203–1233.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). GPytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423.

Gauss, C. F. (1809). *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*, volume 7. Perthes et Besser.

Genevay, A., Chizat, L., Bach, F., Cuturi, M., and Peyré, G. (2019). Sample complexity of Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1574–1583.

Genevay, A., Cuturi, M., Peyré, G., and Bach, F. (2016). Stochastic optimization for large-scale optimal transport. In *Advances in Neural Information Processing Systems*, pages 3440–3448.

Genevay, A., Peyré, G., and Cuturi, M. (2018). Learning generative models with Sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617.

Gerber, S. and Maggioni, M. (2017). Multiscale strategies for computing optimal transport. *The Journal of Machine Learning Research*, 18(1):2440–2471.

Gerber, S., Niethammer, M., Styner, M., and Aylward, S. (2018). Exploratory population analysis with unbalanced optimal transport. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 464–472. Springer.

Gerber, S., Tasdizen, T., Fletcher, P. T., Joshi, S., Whitaker, R., Initiative, A. D. N., et al. (2010). Manifold modeling for brain population analysis. *Medical image analysis*, 14(5):643–653.

Glaunes, J. (2005). Transport par difféomorphismes de points, de mesures et de courants pour la comparaison de formes et l'anatomie numérique. *These de sciences, Université Paris*, 13.

Glaunes, J., Trouvé, A., and Younes, L. (2004). Diffeomorphic matching of distributions: A new approach for unlabelled point-sets and sub-manifolds matching. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. Citeseer.

Gold, S., Rangarajan, A., Lu, C.-P., Pappu, S., and Mjolsness, E. (1998). New algorithms for 2d and 3d point matching: Pose estimation and correspondence. *Pattern recognition*, 31(8):1019–1031.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gori-Giorgi, P., Seidl, M., and Vignale, G. (2009). Density-functional theory for strongly interacting electrons. *Physical review letters*, 103(16):166402.

Greengard, L. (1988). *The rapid evaluation of potential fields in particle systems*. MIT press.

Greengard, L. and Lee, J.-Y. (2004). Accelerating the nonuniform fast Fourier transform. *SIAM review*, 46(3):443–454.

Grenander, U. and Miller, M. I. (1998). Computational anatomy: An emerging discipline. *Quarterly of applied mathematics*, 56(4):617–694.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P. (2003). Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67. Eurographics Association.

Gris, B. (2019). Incorporation of a deformation prior in image reconstruction. *Journal of Mathematical Imaging and Vision*, 61(5):691–709.

Gris, B., Durrleman, S., and Trouvé, A. (2018). A sub-Riemannian modular framework for diffeomorphism-based analysis of shape ensembles. *SIAM Journal on Imaging Sciences*, 11(1):802–833.

Gromov, M. (1987). Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer.

Haber, E., Rehman, T., and Tannenbaum, A. (2010). An efficient numerical method for the solution of the L2 optimal mass transfer problem. *SIAM Journal on Scientific Computing*, 32(1):197–211.

Hackbusch, W. (2015). *Hierarchical matrices: algorithms and analysis*, volume 49. Springer.

Haker, S., Zhu, L., Tannenbaum, A., and Angenent, S. (2004). Optimal mass transport for registration and warping. *International Journal of computer vision*, 60(3):225–240.

Hamilton, S. W. R. (1835). *Second essay on a general method in dynamics*. Philosophical Transactions of the Royal Society.

Hascoët, L. and Pascual, V. (2013). The Tapenade Automatic Differentiation tool: Principles, Model, and Specification. *ACM Transactions On Mathematical Software*, 39(3).

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, page 282. Citeseer.

Higham, N. J. (2005). The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193.

Holm, D. D. and Marsden, J. E. (2005). Momentum maps and measure-valued solutions (peakons, filaments, and sheets) for the EPDiff equation. In *The breadth of symplectic and Poisson geometry*, pages 203–235. Springer.

Holm, D. D., Ratnanather, J. T., Trouvé, A., and Younes, L. (2004). Soliton dynamics in computational anatomy. *NeuroImage*, 23:S170–S178.

Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial intelligence*, 17(1-3):185–203.

Huang, D., Swanson, E. A., Lin, C. P., Schuman, J. S., Stinson, W. G., Chang, W., Hee, M. R., Flotte, T., Gregory, K., Puliafito, C. A., et al. (1991). Optical coherence tomography. *Science*, 254(5035):1178–1181.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90.

Inc., P. T. (2015). Collaborative data science. *Montreal, QC.*

Irons, M. L. (2005). The curvature and geodesics of the torus.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.

Jakob, W., Rhinelander, J., and Moldovan, D. (2017). PyBind11 – seamless operability between C++11 and python. https://github.com/pybind/pybind11.

Jantzen, R. T. (2012). Geodesics on the torus and other surfaces of revolution clarified using undergraduate physics tricks with bonus: nonrelativistic and relativistic kepler problems. *arXiv preprint arXiv:1212.6206.*

Jayasumana, S., Salzmann, M., Li, H., and Harandi, M. (2013). A framework for shape analysis via Hilbert space embedding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1249–1256.

Jones, E., Oliphant, T., Peterson, P., et al. (2001). SciPy: Open source scientific tools for Python. [Online; accessed 17th November 2019].

Jordan, R., Kinderlehrer, D., and Otto, F. (1998). The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17.

Kahan, W. (1965). Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40.

Kaltenmark, I. (2016). *Geometrical Growth Models for Computational Anatomy*. PhD thesis, Université Paris-Saclay.

Kaltenmark, I., Charlier, B., and Charon, N. (2017). A general framework for curve and surface comparison and registration with oriented varifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3346–3355.

Kantorovich, L. V. (1942). On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201.

Kendall, D. G. (1977). The diffusion of shape. *Advances in applied probability*, 9(3):428–430.

Kendall, D. G. (1984). Shape manifolds, Procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121.

Kendall, D. G. (1989). A survey of the statistical theory of shape. *Statistical Science*, pages 87–99.

Kendall, D. G. and Kendall, W. S. (1980). Alignments in two-dimensional random sets of points. *Advances in Applied probability*, 12(2):380–424.

Kilian, M., Mitra, N. J., and Pottmann, H. (2007). Geometric modeling in shape space. In *ACM Transactions on Graphics (TOG)*, volume 26, page 64. ACM.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirk, D. B. and Wen-Mei, W. H. (2010). *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann.

Klein, A., Andersson, J., Ardekani, B. A., Ashburner, J., Avants, B., Chiang, M.-C., Christensen, G. E., Collins, D. L., Gee, J., Hellier, P., et al. (2009a). Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration. *Neuroimage*, 46(3):786–802.

Klein, S., Staring, M., Murphy, K., Viergever, M. A., and Pluim, J. P. (2009b). Elastix: a toolbox for intensity-based medical image registration. *IEEE transactions on medical imaging*, 29(1):196–205.

Klingenberg, C. P. (2015). Analyzing fluctuating asymmetry with geometric morphometrics: concepts, methods, and applications. *Symmetry*, 7(2):843–934.

Knight, P. A., Ruiz, D., and Uçar, B. (2014). A symmetry preserving algorithm for matrix scaling. *SIAM journal on Matrix Analysis and Applications*, 35(3):931–955.

Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76.

Kosowsky, J. J. and Yuille, A. L. (1994). The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural networks*, 7(3):477–490.

Kouranbaeva, S. (1999). The Camassa–Holm equation as a geodesic flow on the diffeomorphism group. *Journal of Mathematical Physics*, 40(2):857–868.

Koval, I., Schiratti, J.-B., Routier, A., Bacci, M., Colliot, O., Allassonnière, S., Durrleman, S., Initiative, A. D. N., et al. (2017). Statistical learning of spatiotemporal patterns from longitudinal manifold-valued networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 451–459. Springer.

Krebs, J., e Delingette, H., Mailhé, B., Ayache, N., and Mansi, T. (2019). Learning a probabilistic model for diffeomorphic registration. *IEEE transactions on medical imaging*.

Krebs, J., Mansi, T., Delingette, H., Zhang, L., Ghesu, F. C., Miao, S., Maier, A. K., Ayache, N., Liao, R., and Kamen, A. (2017). Robust non-rigid registration through agent-based action learning. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 344–352. Springer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Kühnel, L. and Sommer, S. (2017). Computational anatomy in Theano. In *Graphs in Biomedical Image Analysis, Computational Anatomy and Imaging Genetics*, pages 164–176. Springer.

Kullback, S. (1997). *Information theory and statistics*. Courier Corporation.

Kühn, S. (2019). Tissot's indicatrix. `https://upload.wikimedia.org/wikipedia/commons/8/87/Tissot_mercator.png`. CC BY-SA 3.0.

Lavenant, H. (2019a). *Courbes et applications optimales à valeurs dans l'espace de Wasserstein*. PhD thesis, Université Paris-Saclay.

Lavenant, H. (2019b). Harmonic mappings valued in the Wasserstein space. *Journal of Functional Analysis*, 277(3):688–785.

Lavenant, H. (2019c). Unconditional convergence for discretizations of dynamical optimal transport. *arXiv preprint arXiv:1909.08790*.

Lavenant, H., Claici, S., Chien, E., and Solomon, J. (2018). Dynamical optimal transport on discrete surfaces. In *SIGGRAPH Asia 2018 Technical Papers*, page 250. ACM.

Le Bihan, D., Mangin, J.-F., Poupon, C., Clark, C. A., Pappata, S., Molko, N., and Chabriat, H. (2001). Diffusion tensor imaging: concepts and applications. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 13(4):534–546.

Leary, C. and Wang, T. (2017). XLA: TensorFlow, compiled. *TensorFlow Dev Summit*.

Leclaire, A. and Rabin, J. (2019). A fast multi-layer approximation to semi-discrete optimal transport. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 341–353. Springer.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y. and Cortes, C. (1998). The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Ledig, C., Schuh, A., Guerrero, R., Heckemann, R. A., and Rueckert, D. (2018). Structural brain imaging in Alzheimer's disease and mild cognitive impairment: biomarker analysis and shared morphometry database. *Scientific reports*, 8(1):11258.

Lee, J. M. (2006). *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media.

Lee, S., Charon, N., Charlier, B., Popuri, K., Lebed, E., Sarunic, M. V., Trouvé, A., and Beg, M. F. (2017). Atlas-based shape analysis and classification of retinal optical coherence tomography images using the functional shape (fshape) framework. *Medical image analysis*, 35:570–581.

Legendre, A. M. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot.

Léger, F. (2020). A gradient descent perspective on Sinkhorn. *arXiv preprint arXiv:2002.03758*.

Léonard, C. (2012). From the Schrödinger problem to the Monge–Kantorovich problem. *Journal of Functional Analysis*, 262(4):1879–1920.

Lévy, B. (2015). A numerical algorithm for l2 semi-discrete optimal transport in 3d. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(6):1693–1715.

Leys, J., Ghys, E., and Alvarez, A. (2013). Chaos, une aventure mathématique.

Li, W., Ryu, E. K., Osher, S., Yin, W., and Gangbo, W. (2018). A parallel method for earth mover's distance. *Journal of Scientific Computing*, 75(1):182–197.

Liero, M., Mielke, A., and Savaré, G. (2015). Optimal entropy-transport problems and a new hellinger–kantorovich distance between positive measures. *Inventiones mathematicae*, pages 1–149.

Lin, T. and Zha, H. (2008). Riemannian manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):796–809.

Litany, O., Bronstein, A., Bronstein, M., and Makadia, A. (2018). Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1886–1895.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Lombaert, H., Grady, L., Pennec, X., Ayache, N., and Cheriet, F. (2014). Spectral log-demons: diffeomorphic image registration with very large deformations. *International journal of computer vision*, 107(3):254–271.

Lorenzi, M., Ayache, N., Frisoni, G. B., Pennec, X., (ADNI, A. D. N. I., et al. (2013). LCC-Demons: a robust and accurate symmetric diffeomorphic registration algorithm. *NeuroImage*, 81:470–483.

Lorenzi, M. and Pennec, X. (2013). Geodesics, parallel transport & one-parameter subgroups for diffeomorphic image registration. *International journal of computer vision*, 105(2):111–127.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *IEEE international conference on computer vision*, volume 99, pages 1150–1157.

Ma, J., Zhao, J., Tian, J., Yuille, A. L., and Tu, Z. (2014). Robust point matching via vector field consensus. *IEEE Transactions on Image Processing*, 23(4):1706–1721.

Mahalanobis, P. C. (1925). Analysis of race-mixture in Bengal. *Journal of the Asiatic Society of Bengal*.

Mahalanobis, P. C. (1936). On the generalised distance in statistics. In *Proceedings of the National Institute of Science of India*, volume 12, pages 49–55.

Mairal, J., Bach, F., and Ponce, J. (2014). Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283.

Mallat, S. (1999). *A wavelet tour of signal processing*. Elsevier.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 7:674–693.

Mansi, T., Voigt, I., Leonardi, B., Pennec, X., Durrleman, S., Sermesant, M., Delingette, H., Taylor, A. M., Boudjemline, Y., Pongiglione, G., et al. (2011). A statistical model for quantification and prediction of cardiac remodelling: Application to tetralogy of Fallot. *IEEE transactions on medical imaging*, 30(9):1605–1616.

Mastroianni, G. and Occorsio, D. (2001). Optimal systems of nodes for Lagrange interpolation on bounded intervals. a survey. *Journal of computational and applied mathematics*, 134(1-2):325–341.

McLeod, K., Prakosa, A., Mansi, T., Sermesant, M., and Pennec, X. (2011). An incompressible log-domain demons algorithm for tracking heart tissue. In *International Workshop on Statistical Atlases and Computational Models of the Heart*, pages 55–67. Springer.

Mena, G. and Weed, J. (2019). Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. *arXiv preprint arXiv:1905.11882*.

Mensch, A., Blondel, M., and Peyré, G. (2019). Geometric losses for distributional learning. In *International Conference on Machine Learning*, pages 4516–4525.

Mérigot, Q. (2011). A multiscale approach to optimal transport. In *Computer Graphics Forum*, volume 30, pages 1583–1592. Wiley Online Library.

Mérigot, Q. and Mirebeau, J.-M. (2016). Minimal geodesics along volume-preserving maps, through semidiscrete optimal transport. *SIAM Journal on Numerical Analysis*, 54(6):3465–3492.

Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667.

Micheli, M. and Glaunès, J. A. (2014). Matrix-valued kernels for shape deformation analysis. *Geometry, Imaging and Computing*, 1(1):57–139.

Micheli, M., Michor, P. W., and Mumford, D. (2012). Sectional curvature in terms of the cometric, with applications to the Riemannian manifolds of landmarks. *SIAM Journal on Imaging Sciences*, 5(1):394–433.

Michor, P. W. and Mumford, D. B. (2006). Riemannian geometries on spaces of plane curves. *Journal of the European Mathematical Society*.

Misiołek, G. (1998). A shallow water equation as a geodesic flow on the bott-virasoro group. *Journal of Geometry and Physics*, 24(3):203–208.

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences*, pages 666–704.

Montagnat, J., Delingette, H., and Ayache, N. (2001). A review of deformable surfaces: topology, geometry and deformation. *Image and vision computing*, 19(14):1023–1040.

Morvan, J.-M. (2008). *Generalized curvatures*. Springer.

Myronenko, A. and Song, X. (2010). Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275.

Mérigot, Q., Delalande, A., and Chazal, F. (2019). Quantitative stability of optimal transport maps and linearization of the 2-Wasserstein space. *arXiv preprint arXiv:1910.05954*.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

Niethammer, M., Kwitt, R., and Vialard, F.-X. (2019a). Metric learning for image registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8463–8472.

Niethammer, M., Shen, Z., and Kwitt, R. (2019b). Mermaid: Image registration via automatic differentiation. `https://mermaid.readthedocs.io/`.

Nikulin, Y. and Novak, R. (2016). Exploring the neural algorithm of artistic style. *arXiv preprint arXiv:1602.07188*.

Norton, J. (2013). Philosophical significance of the general theory of relativity.

Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *European conference on computer vision*, pages 490–503. Springer.

Ogihara, N., Amano, H., Kikuchi, T., Morita, Y., Hasegawa, K., Kochiyama, T., and Tanabe, H. C. (2015). Towards digital reconstruction of fossil crania and brain morphology. *Anthropological Science*, page 141109.

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill*. https://distill.pub/2017/feature-visualization.

Ollier, W., Sprosen, T., and Peakman, T. (2005). UK Biobank: from concept to reality. *Pharmacogenomics*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Paty, F.-P., d'Aspremont, A., and Cuturi, M. (2019). Regularity as regularization: Smooth and strongly convex Brenier potentials in optimal transport. *arXiv preprint arXiv:1905.10812*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Peemen, M., Mesman, B., and Corporaal, H. (2011). Speed sign detection and recognition by convolutional neural networks. In *Proceedings of the 8th international automotive congress*, pages 162–170. sn.

Pennec, X. (2008). Statistical computing on manifolds: from Riemannian geometry to computational anatomy. In *LIX Fall Colloquium on Emerging Trends in Visual Computing*, pages 347–386. Springer.

Pennec, X., Fillard, P., and Ayache, N. (2006). A Riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66.

Pennec, X., Sommer, S., and Fletcher, T. (2019). *Riemannian Geometric Statistics in Medical Image Analysis*. Academic Press.

Peyré, G. and Cuturi, M. (2017). Computational optimal transport. *arXiv:1610.06519*.

Poincaré, H. (1902). *La science et l'hypothèse*. Flammarion.

Portilla, J. and Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70.

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

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.

Rabin, J., Ferradans, S., and Papadakis, N. (2014). Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4852–4856. IEEE.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. (2011). Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 435–446. Springer.

Ragan-Kelley, J., Barnes, C., Adams, A., Paris, S., Durand, F., and Amarasinghe, S. (2013). Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In *Acm Sigplan Notices*, volume 48, pages 519–530. ACM.

Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

Ramdas, A., Trillos, N. G., and Cuturi, M. (2017). On Wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2).

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

Ray, N., Sokolov, D., Lefebvre, S., and Lévy, B. (2018). Meshless Voronoi on the GPU. In *SIGGRAPH Asia 2018 Technical Papers*, page 265. ACM.

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

Roussillon, P. (2017). *Modèles de cycles normaux pour l'analyse des déformations*. PhD thesis, Sorbonne Paris Cité.

Roussillon, P. and Glaunès, J. A. (2019). Representation of surfaces with normal cycles and application to surface registration. *Journal of Mathematical Imaging and Vision*, pages 1–27.

Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L., Leach, M. O., and Hawkes, D. J. (1999). Nonrigid registration using free-form deformations: application to breast MR images. *IEEE transactions on medical imaging*, 18(8):712–721.

Ruschendorf, L. et al. (1995). Convergence of the iterative proportional fitting procedure. *The Annals of Statistics*, 23(4):1160–1174.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Ryu, E. K., Li, W., Yin, P., and Osher, S. (2018). Unbalanced and partial L1 Monge–Kantorovich problem: A scalable parallel first-order method. *Journal of Scientific Computing*, 75(3):1596–1613.

Salimans, T., Zhang, H., Radford, A., and Metaxas, D. (2018). Improving GANs using optimal transport. *arXiv preprint arXiv:1803.05573*.

Sanjabi, M., Ba, J., Razaviyayn, M., and Lee, J. D. (2018). On the convergence and robustness of training GANs with regularized optimal transport. *arXiv preprint arXiv:1802.08249*.

Santambrogio, F. (2015). *Optimal Transport for applied mathematicians*, volume 87 of *Progress in Nonlinear Differential Equations and their applications*. Springer.

Santambrogio, F. (2017). {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154.

Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., Gould, J., Liu, S., Lin, S., Berube, P., et al. (2019). Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943.

Schiratti, J.-B., Allassonniere, S., Colliot, O., and Durrleman, S. (2015). Learning spatiotemporal trajectories from manifold-valued longitudinal data. In *Advances in Neural Information Processing Systems*, pages 2404–2412.

Schmaltz, C., Gwosdek, P., Bruhn, A., and Weickert, J. (2010). Electrostatic halftoning. In *Computer Graphics Forum*, volume 29, pages 2313–2327. Wiley Online Library.

Schmitzer, B. (2014). *Isometry Invariant Shape Priors for Variational Image Segmentation*. PhD thesis, Ruperto-Carola University of Heidelberg.

Schmitzer, B. (2016). A sparse multiscale algorithm for dense optimal transport. *Journal of Mathematical Imaging and Vision*, 56(2):238–259.

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

Schrijver, A. (2003). *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media.

Schrödinger, E. (1932). Sur la théorie relativiste de l'électron et l'interprétation de la mécanique quantique. In *Annales de l'institut Henri Poincaré*, volume 2, pages 269–310.

Seguy, V. and Cuturi, M. (2015). Principal geodesic analysis for probability measures under the optimal transport metric. In *Advances in Neural Information Processing Systems*, pages 3312–3320.

Séjourné, T., Feydy, J., Vialard, F.-X., Trouvé, A., and Peyré, G. (2019). Sinkorn divergences for unbalanced Optimal Transport. *arXiv preprint arXiv:1910.12958*.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.

Sharify, M., Gaubert, S., and Grigori, L. (2011). Solution of the optimal assignment problem by diagonal scaling algorithms. *arXiv preprint arXiv:1104.3830*.

Shawe-Taylor, J., Cristianini, N., et al. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

Shen, Z., Vialard, F.-X., and Niethammer, M. (2019). Region-specific diffeomorphic metric mapping. In *Advances in Neural Information Processing Systems*.

Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statist.*, 35:876–879.

Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the Ninth IEEE International Conference on Computer Vision-Volume 2*, page 1470. IEEE Computer Society.

Skodras, A., Christopoulos, C., and Ebrahimi, T. (2001). The JPEG 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58.

Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. (2015). Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66.

Sommer, S., Lauze, F., Hauberg, S., and Nielsen, M. (2010). Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations. In *European conference on computer vision*, pages 43–56. Springer.

Sommer, S., Lauze, F., Nielsen, M., and Pennec, X. (2013a). Sparse multi-scale diffeomorphic registration: the kernel bundle framework. *Journal of mathematical imaging and vision*, 46(3):292–308.

Sommer, S., Nielsen, M., Darkner, S., and Pennec, X. (2013b). Higher-order momentum distributions and locally affine LDDMM registration. *SIAM Journal on Imaging Sciences*, 6(1):341–367.

Sommer, S., Nielsen, M., Lauze, F., and Pennec, X. (2011). A multi-scale kernel bundle for LDDMM: towards sparse deformation description across space and scales. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 624–635. Springer.

Song, A., Uhlmann, V., Fageot, J., and Unser, M. (2019). Dictionary learning for 2d Kendall shapes. *arXiv preprint arXiv:1903.11356*.

Srivastava, A., Klassen, E., Joshi, S. H., and Jermyn, I. H. (2010). Shape analysis of elastic curves in euclidean spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1415–1428.

Srivastava, A. and Klassen, E. P. (2016). *Functional and shape data analysis*. Springer.

Staniforth, A. and Côté, J. (1991). Semi-Lagrangian integration schemes for atmospheric models – a review. *Monthly weather review*, 119(9):2206–2223.

Su, Z., Bauer, M., Preston, S. C., Laga, H., and Klassen, E. (2019). Shape analysis of surfaces using general elastic metrics. *arXiv preprint arXiv:1910.02045*.

Surace, S. C., Pfister, J.-P., Gerstner, W., and Brea, J. (2018). On the choice of metric in gradient-based theories of brain function. *arXiv preprint arXiv:1805.11851*.

Szekely, G. J. and Rizzo, M. L. (2005). Hierarchical clustering via joint between-within distances: Extending Ward's minimum variance method. *Journal of classification*, 22(2):151–183.

Takeda, H., Farsiu, S., and Milanfar, P. (2007). Kernel regression for image processing and reconstruction. *IEEE Transactions on image processing*, 16(2):349–366.

Teschl, G. (2012). *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc.

Thibault, A., Chizat, L., Dossal, C., and Papadakis, N. (2017). Overrelaxed Sinkhorn-Knopp algorithm for regularized optimal transport. *arXiv preprint arXiv:1711.01851*.

Thompson, D. W. (1917). *On growth and form*. Cambridge University Press.

Tissot, A. (1878). Mémoire sur la représentation des surfaces et les projections des cartes géographiques. *Nouvelles annales de mathématiques: journal des candidats aux écoles polytechnique et normale*, 17:145–163.

Tournier, J.-D., Smith, R., Raffelt, D., Tabbara, R., Dhollander, T., Pietsch, M., Christiaens, D., Jeurissen, B., Yeh, C.-H., and Connelly, A. (2019). MRtrix3: A fast, flexible and open software framework for medical image processing and visualisation. *NeuroImage*, page 116137.

Trouvé, A. and Vialard, F.-X. (2012). Shape splines and stochastic shape evolutions: A second order point of view. *Quarterly of Applied Mathematics*, pages 219–251.

Uchida, S. (2013). Image processing and recognition for biological images. *Development, growth & differentiation*, 55(4):523–549.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2018). Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454.

Vaillant, M. and Glaunès, J. (2005). Surface matching via currents. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 381–392. Springer.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22.

Van Essen, D. C., Smith, S. M., Barch, D. M., Behrens, T. E., Yacoub, E., Ugurbil, K., Consortium, W.-M. H., et al. (2013). The WU–Minn human connectome project: an overview. *Neuroimage*, 80:62–79.

Vasilache, N., Zinenko, O., Theodoridis, T., Goyal, P., DeVito, Z., Moses, W. S., Verdoolaege, S., Adams, A., and Cohen, A. (2018). Tensor comprehensions: Framework-agnostic high-performance machine learning abstractions. *arXiv preprint arXiv:1802.04730*.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S-PLUS*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Vialard, F.-X. (2019). An elementary introduction to entropic regularization and proximal methods for numerical optimal transport. *HAL preprint, hal-02303456*.

Vialard, F.-X. and Risser, L. (2014). Spatially-varying metric learning for diffeomorphic image registration: A variational framework. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 227–234. Springer.

Villani, C. (2003). *Topics in optimal transportation*. Graduate studies in Mathematics. American Mathematical Society.

Villani, C. (2008). *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

von Radziewsky, P., Eisemann, E., Seidel, H.-P., and Hildebrandt, K. (2016). Optimized subspaces for deformation-based modeling and shape interpolation. *Computers & Graphics*, 58:128–138.

Von-Tycowicz, C., Schulz, C., Seidel, H.-P., and Hildebrandt, K. (2015). Real-time nonlinear shape interpolation. *ACM Transactions on Graphics (TOG)*, 34(3):34.

Walker, H. F. and Ni, P. (2011). Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735.

Wallace, G. K. (1992). The JPEG still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv.

Wang, K. A., Pleiss, G., Gardner, J. R., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact Gaussian processes on a million data points. *arXiv preprint arXiv:1903.08114*.

Wasserthal, J., Neher, P., and Maier-Hein, K. H. (2018). Tractseg – Fast and accurate white matter tract segmentation. *NeuroImage*, 183:239–253.

Weed, J., Bach, F., et al. (2019). Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance. *Bernoulli*, 25(4A):2620–2648.

Wei, D., Zhou, B., Torralba, A., and Freeman, W. T. (2017). mNeuron: A Matlab plugin to visualize neurons from deep models. *Massachusetts Institute of Technology*.

Wilson, A. G. (1969). The use of entropy maximising models, in the theory of trip distribution, mode split and route split. *Journal of Transport Economics and Policy*, pages 108–126.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90.

Yang, C., Duraiswami, R., Gumerov, N. A., and Davis, L. (2003). Improved fast Gauss transform and efficient kernel density estimation. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, page 464. IEEE Computer Society.

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012). Nyström method vs random Fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484.

Yang, X., Kwitt, R., Styner, M., and Niethammer, M. (2017). Quicksilver: Fast predictive image registration – a deep learning approach. *NeuroImage*, 158:378–396.

Yoo, T. S., Ackerman, M. J., Lorensen, W. E., Schroeder, W., Chalana, V., Aylward, S., Metaxas, D., and Whitaker, R. (2002). Engineering and algorithm design for an image processing API: a technical report on ITK-the insight toolkit. *Studies in health technology and informatics*, pages 586–592.

Younes, L. (2010). *Shapes and diffeomorphisms*, volume 171. Springer.

Yule, G. U. (1912). On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society*, 75(6):579–652.

Zhang, F., Wu, Y., Norton, I., Rigolo, L., Rathi, Y., Makris, N., and O'Donnell, L. J. (2018). An anatomically curated fiber clustering white matter atlas for consistent white matter tract parcellation across the lifespan. *NeuroImage*, 179:429–447.

Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved Nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM.

Zhang, M. and Fletcher, P. T. (2019). Fast diffeomorphic image registration via Fourier-approximated Lie algebras. *International Journal of Computer Vision*, 127(1):61–73.

Zhao, K., Vouvakis, M. N., and Lee, J.-F. (2005). The adaptive cross approximation algorithm for accelerated method of moments computations of emc problems. *IEEE transactions on electromagnetic compatibility*, 47(4):763–773.

Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., and Shum, H.-Y. (2005). Large mesh deformation using the volumetric graph Laplacian. In *"ACM Transactions on Graphics (Proceedings of SIGGRAPH)"*, pages 496–503.

Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536.