Global divergences between measures: from Hausdorff distance to Optimal Transport

Jean Feydy, Alain Trouvé

Global divergences between measures: from Hausdorff distance to Optimal Transport

Jean Feydy[1,2] and Alain Trouvé[2]

[1] DMA, École Normale Supérieure, Paris, France
jean.feydy@ens.fr
[2] CMLA, ENS Paris-Saclay, Cachan, France
trouve@cmla.ens-cachan.fr

Abstract. The data fidelity term is a key component of shape registration pipelines: computed at every step, its gradient is the vector field that drives a deformed model towards its target. Unfortunately, most classical formulas are at most semi-local: their gradients saturate and stop being informative above some given distance, with appalling consequences on the robustness of shape analysis pipelines.
In this paper, we build on recent theoretical advances on *Sinkhorn entropies and divergences* [6] to present a unified view of three fidelities between measures that alleviate this problem: the Energy Distance from statistics; the (weighted) Hausdorff distance from computer graphics; the Wasserstein distance from Optimal Transport theory. The ε-Hausdorff and ε-Sinkhorn divergences are *positive* fidelities that interpolate between these three quantities, and we implement them through efficient, freely available GPU routines. They should allow the shape analyst to handle large deformations without hassle.

Keywords: shape registration · kernel · energy distance · hausdorff distance · optimal transport · GPU

1 Introduction

Shape registration as a variational problem. Given a source shape A and a target B, a key problem in medical image analysis is to register the former onto the latter. That is, to estimate a mapping φ (a change of coordinates) that maps the source A into a model $\varphi(A)$ which is "close enough" to the target.

Most classical registration algorithms strive to minimize an energy

$$\mathrm{E}(\varphi) \quad = \quad \underbrace{\mathrm{Reg}(\varphi)}_{\text{regularizer}} \quad + \quad \underbrace{\mathrm{d}\left(\varphi(A), B\right)}_{\text{fidelity}}$$

which is the sum of a regularization term – encoding a prior on acceptable mappings – and a data attachment term – or *fidelity* – that measures how far the model $\varphi(A)$ is from the target B.

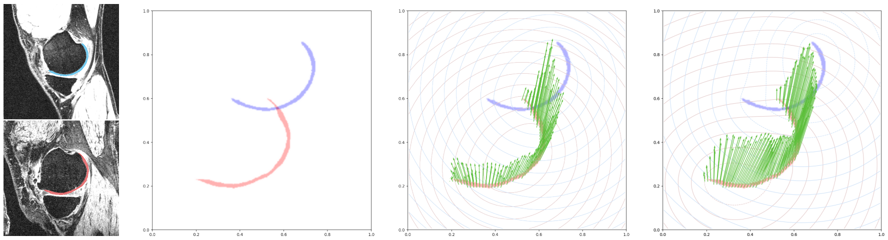
The need for robust fidelities and gradients. Unfortunately, as of today, most fidelities can at best be described as semi-local. Relying on small convolution filters or kernel functions that saturate at long range [8], they stop being informative when parts of the shapes are far away from each other. In recent years, finely crafted formulas have been proposed to alleviate this problem [9]; but they were probably too hard to implement and did not meet widespread adoption. As a result, most users today still rely on finely tuned coarse-to-fine schemes to register shape populations.

Contribution. At MICCAI 2017, we introduced the theory of Optimal Transport to the medical imaging community [5]. Leveraging the ideas and algorithms presented in [11], we showed that using *globally optimal* spring systems to drive a registration routine is tractable, and improves the robustness of pipelines to large deformations. The present paper is about taking advantage of new advances in the field [6] that let us bridge the gap between Optimal Transport and the standard shape analysis toolkit.

In section 1, we review the standard theory of measures and kernel distances (also known as blurred Sums of Squared Distances). We stress the relevance of the scale-invariant kernel $k(x) = -\|x\|$, which induces the global Energy Distance between shapes. We also notice that kernel distances rely on *linear potentials* (influence fields) generated by the shapes.

In section 2, we show how to use and compute *non-linear* potentials. We introduce a family of cheap fidelities between measures, the ε-SoftMin costs, that interpolate between the Energy Distance ($\varepsilon = +\infty$) and the weighted Hausdorff distance ($\varepsilon = 0$) borrowed from computer graphics.

Finally, in section 3, we come back to the optimal transport cost and show that it is nothing but a "Hausdorff" distance under a mass repartition constraint. We interpret the celebrated Sinkhorn algorithm as a *balancing scheme on distance fields* and put an emphasis on two fidelities: the cheap ε-Hausdorff and the high-quality ε-Sinkhorn divergence, with a guarantee of positivity for both.



(a) data (b) segmentations (c) energy distance (d) optimal transport

Fig. 1: We focus this paper on the registration of thin segmented volumes (a, from the OsteoArthritis Initiative) encoded as measures on the ambient space (b). We provide efficient GPU routines to compute long-range gradients, from cheap kernel distances (c) to high-quality optimal transport plans (d).

In practice. Most importantly, we provide efficient CUDA routines – with Matlab, numpy and pytorch bindings – that can be used to implement these new data attachment terms. As shown in section 4, our KeOps library [2] allows users to process curves, surfaces and segmentation maps with up to 100,000 actives vertices on a cheap laptop's GPU. Our code is freely available:

Please visit `github.com/jeanfeydy/global-divergences`.

These new tools fit seamlessly into the standard shape analyst's toolkit; they should help the reader to improve with little to no overhead the robustness to large deformations of its shape analysis pipeline.

1.1 Representing shapes as measures on a space of features

In this paper. We choose to focus this paper on a setting that is understood well by all researchers in medical image analysis: the registration of *normalized density maps*. Our source bitmap A (in red) and target B (in blue) will be encoded as measures

$$\alpha \;=\; \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i} \quad \text{and} \quad \beta \;=\; \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}, \quad \text{with} \quad \sum_{i=1}^{\mathrm{N}} \alpha_i \;=\; 1 \;=\; \sum_{j=1}^{\mathrm{M}} \beta_j,$$

where the x_i's (respectively y_j's) are the coordinates of the N (resp. M) nonzero pixels of A (resp. B), with positive weights α_i (resp. β_j) summing up to one.

In most figures, we will display the gradient $\nabla_{x_i} \mathrm{d}(\alpha, \beta)$ of a fidelity "d" as a green vector field supported by the x_i's. This descent direction is meant to be used by registration algorithms and is thus the primary information to look at in our pictures. In the background, depending on the section, we also display the level lines of the linear potential "$k \star (\alpha - \beta)$" (in blue) or of the influence fields "a" (in red) and "b" (in blue) – more about that later.

Extensions. The results presented in this paper can be extended to other use cases fairly easily. First, we may wish to use an image-based registration of segmentation maps instead of the mass preserving "Jacobian-free" action. To do so, we should simply compute the gradient $\nabla_{\alpha_i} \mathrm{d}(\alpha, \beta)$ of fidelities with respect to the weights of the atomic dirac masses; the presence of long-range interactions is equally important to the robustness of the registration algorithm, with mass contraction (i.e. deletion) replacing the spreading out phenomenon observed in Figure 3.(a-b).

Most of our results still hold when the source and the target don't have the same mass – the only noticeable changes would be located in section 3, and we recommend [11] as an introduction to the theory of *unbalanced* optimal transport. Going further, these new tools and GPU routines can also be used to handle fiber tracks, curves and surfaces through the *varifold* framework presented in [8].

Notations. In order to let our results be useful to researchers working with curves and surfaces – which are best represented as measures on a product space (position,orientation,curvature) – we will refer to the ambient space $\mathbb{R}^2$ or $\mathbb{R}^3$ as to an abstract *feature space* $\mathcal{X}$. The letters x, y and z will denote points in the feature space, while α, β or μ stand for finitely supported positive measures; finally, a, b and m denote real-valued functions on $\mathcal{X}$ understood as **influence fields generated by their respective measures**.

If $(z_i)_{i\in[\![1,\mathrm{N}]\!]}$ is a collection of N points in $\mathcal{X}$ and if $m : \mathcal{X} \rightarrow \mathbb{R}$ is a function on the feature space, we will also write "m_{z_i}" to denote the length-N vector $(m(z_i))_{i\in[\![1,\mathrm{N}]\!]}$ of values of m sampled on the point cloud z_i.

Finally, if $\mu = \sum_{i=1}^{\mathrm{N}} \mu_i \delta_{z_i}$ is a finitely supported measure and if $m : \mathcal{X} \rightarrow \mathbb{R}$ is a function on the feature space, we will write

$$\langle\, \mu \,,\, m \,\rangle ~=~ (\, \mu_i \mid m_{z_i} \,) ~=~ \sum_{i=1}^{\mathrm{N}} \mu_i \; m(z_i).$$

Here, $(\mu_i)_{i\in[\![1,\mathrm{N}]\!]}$ and $(m_{z_i})_{i\in[\![1,\mathrm{N}]\!]}$ are two vectors of $\mathbb{R}^{\mathrm{N}}$: the measure-function duality bracket $\langle\, \cdot \,,\, \cdot \,\rangle$ is thus understood as a simple scalar product $(\,\cdot\,|\,\cdot\,)$ in $\mathbb{R}^{\mathrm{N}}$.

1.2 Kernel distances

If α and β represent two shapes in the feature space $\mathcal{X}$, using standard information-theoretic fidelities such as the symmetrised Kullback-Leibler divergence

$$\mathrm{KL}_{\mathrm{sym}}(\alpha,\beta) ~=~ \tfrac{1}{2}\mathrm{KL}(\alpha,\beta) + \tfrac{1}{2}\mathrm{KL}(\beta,\alpha) ~=~ \tfrac{1}{2}\Big\langle\, \alpha-\beta \,,\, \log\Big(\tfrac{\mathrm{d}\alpha}{\mathrm{d}\beta}\Big) \Big\rangle ~~\geqslant~ 0 \quad (1)$$

is not recommended: shape analysis routines should take into account the *geometry* of the feature space.

Kernel norms. A common way of doing so is to endow the feature space $\mathcal{X}$ with a symmetric *kernel function* $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ and to use

$$\begin{aligned} \mathrm{d}_k(\alpha,\beta) ~&=~ \tfrac{1}{2}\langle\, \alpha-\beta \,,\, k \star (\alpha-\beta) \,\rangle ~=~ \tfrac{1}{2}\langle\, \alpha-\beta \,,\, b^k - a^k \,\rangle, && (2)\\ \text{where} \quad a^k(z) ~&=~ -\,(k\star\alpha)(z) ~=~ -\sum\nolimits_{i=1}^{\mathrm{N}} \alpha_i \, k(x_i,z) && (3)\\ \text{and} \quad b^k(z) ~&=~ -\,(k\star\beta)(z) ~=~ -\sum\nolimits_{j=1}^{\mathrm{M}} \beta_j \, k(y_j,z). && (4) \end{aligned}$$

In practice, these summations can be implemented as matrix-vector products or, as advocated in Figure 11, by using the online map-reduce routines of the KeOps library [2]. Popular choices include the Gaussian and Laplacian kernels:

$$\begin{aligned} \mathrm{Gaussian}_\sigma(x-y) ~&=~ \exp(-\|x-y\|^2/\sigma^2) \\ \text{and} \quad \mathrm{Laplacian}_\sigma(x-y) ~&=~ \exp(-\|x-y\|/\sigma). \end{aligned}$$

However, as $\nabla_{x_i}\mathrm{d}_k(\alpha,\beta)$ is given by the gradient of the linear potential $(b^k - a^k)$ sampled on the x_i's and weighted by the α_i's, we argue in Figures 2-3 that a more robust baseline could be given by the Energy Distance kernel from statistics [12]:

$$\mathrm{Energy}(x-y) ~=~ -\,\|x-y\|.$$

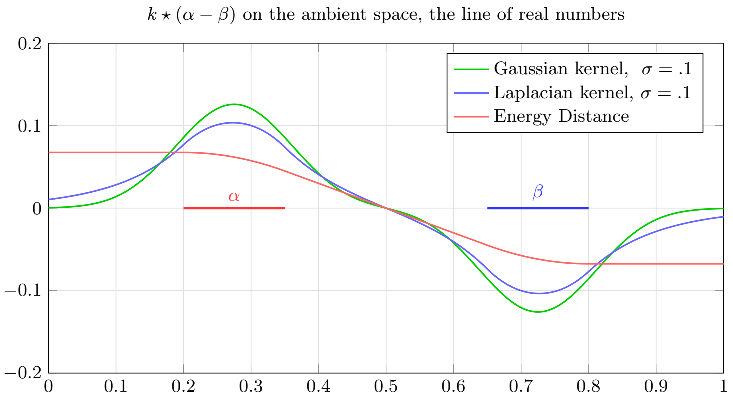
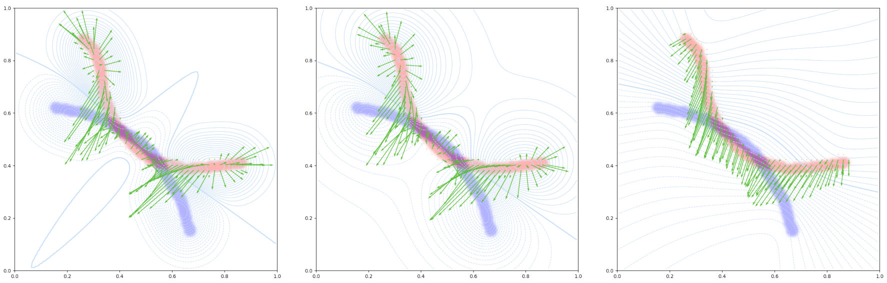
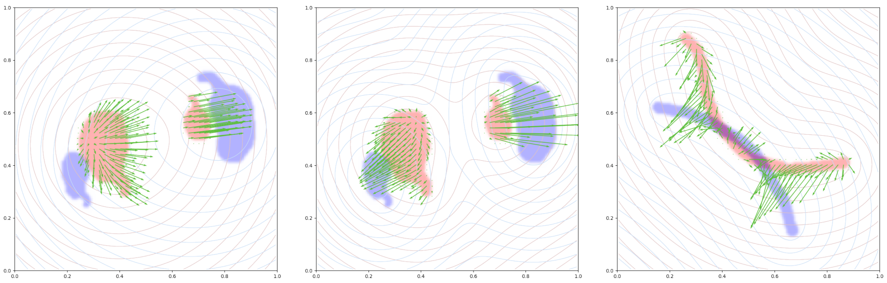


Fig. 2: **The linear potential $k \star (\alpha - \beta)$, for standard kernel functions.** Here, α and β are sampled from the standard Lebesgue measures on the segments $[.2, .35]$ and $[.65, .8]$, respectively. Out of these three curves, the third is the only one whose (minus) gradient always points from α towards β.



(a) Gaussian, $\sigma = .1$ (b) Laplacian, $\sigma = .1$ (c) Energy Distance

Fig. 3: **The Energy Distance is scale-invariant and robust to large deformations.** This is the 2D equivalent of Figure 2, with level lines of $k \star (\alpha - \beta)$ displayed in the background. Notice the spreading out effect in (a-b).



(a) Energy Distance (b) SoftMin, $\varepsilon = .05$ (c) SoftMin, $\varepsilon = .05$

Fig. 4: **Linear potentials can only take you so far.** (a) As it faces a mass imbalance, the global gradient of the Energy Distance tries to split up the largest red mass into pieces. (b-c) The SoftMin fidelity, introduced in section 2, allows us to induce a more "focused" behavior into our algorithms.

2 Computing non-linear potentials

The log-sum-exp trick. In order to build tractable algorithms, restricting ourselves to potentials a and b that depend linearly on the measures α and β seems to be a necessary evil... But we can go further. Indeed, on top of the "summation" operation of Eqs. (3-4), we can implement on the GPU another differentiable reduction operator: the log-sum-exp or SoftMax, defined through

$$\log \textstyle\sum_{i=1}^{\mathrm{N}} \exp(v_i) ~=~ V ~+~ \log \textstyle\sum_{i=1}^{\mathrm{N}} \exp(v_i - V),$$

with $V = \max_i v_i$ taken out of the expression for numerical stability. The KeOps library implements an online variant of this "max-factorization" trick, and lets us scale this operation to large values of N – see Figure 11.

Definition. Then, we propose to endow the ambient space $\mathcal{X}$ with a symmetric cost function $\mathrm{C} : \mathcal{X} \times \mathcal{X} \mapsto \mathrm{C}(x,y)$ – say, $\|x-y\|$ – a regularization strength $\varepsilon > 0$ and a kernel function $k_\varepsilon = \exp(-\tfrac{1}{\varepsilon}\mathrm{C}(\,\cdot\,,\,\cdot\,))$ to define

$$\begin{aligned} \underset{x\sim\alpha}{\min\nolimits_\varepsilon}\ \mathrm{C}(x,z) ~&=~ -\,\varepsilon\, \log(k_\varepsilon \star \alpha)(z) \\ &=~ -\,\varepsilon\, \log \textstyle\sum_{i=1}^{\mathrm{N}} \exp\big(\log(\alpha_i) - \tfrac{1}{\varepsilon}\mathrm{C}(x_i,z)\big)\,. \end{aligned}$$

Mimicking Eq. (2), we propose to see the SoftMin functions as non-linear influence fields, analogous to the linear potentials a^k and b^k. Hence, we introduce the ε-SoftMin cost through

$$\begin{aligned} \mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha,\beta) ~&=~ \tfrac{1}{2}\langle\, \alpha - \beta\,,\, b^\varepsilon - a^\varepsilon\,\rangle \\ &=~ \tfrac{1}{2}\big(\,\alpha_i \mid b^\varepsilon_{x_i} - a^\varepsilon_{x_i}\,\big) ~+~ \tfrac{1}{2}\big(\,\beta_j \mid a^\varepsilon_{y_j} - b^\varepsilon_{y_j}\,\big), \\ \text{where}\quad a^\varepsilon(z) ~=~ \underset{x\sim\alpha}{\min\nolimits_\varepsilon}\ \mathrm{C}(x,z) ~&\text{ and }~ b^\varepsilon(z) ~=~ \underset{y\sim\beta}{\min\nolimits_\varepsilon}\ \mathrm{C}(y,z). \end{aligned}$$

Interpretation. Simple calculations show that if $\mathrm{C}(x,y) = \|x-y\|$, the ε-SoftMin cost converges towards the Energy distance as ε goes to infinity. At the other end of the spectrum, if $\mathrm{C}(x,y) \geqslant 0$ with equality if $x = y$,

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha,\beta) ~\xrightarrow{\varepsilon\rightarrow 0}~ \tfrac{1}{2}\sum_{i=1}^{\mathrm{N}} \alpha_i\ \min_j \mathrm{C}(x_i,y_j) ~+~ \tfrac{1}{2}\sum_{j=1}^{\mathrm{M}} \beta_j\ \min_i \mathrm{C}(x_i,y_j)$$

As shown in Figures 5-6, the SoftMin operators is thus allowing us to interpolate between statistics and computer graphics.

Positivity. Unfortunately, one cannot guarantee the positivity of the ε-SoftMin fidelity: linearizing the cost, we find pairs of measures such that $\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha + \delta\alpha, \alpha) < 0$. However, if λ is a reference measure on the feature space $\mathcal{X}$ (say, the Lebesgue measure on $\mathbb{R}^{\mathrm{D}}$), then

$$\begin{aligned} \varepsilon\,\mathrm{KL}_{\mathrm{sym}}((k_\varepsilon \star \alpha)\cdot\lambda, (k_\varepsilon \star \beta)\cdot\lambda) ~&=~ \tfrac{1}{2}\Big\langle\ \lambda\,\cdot\,k_\varepsilon \star (\alpha - \beta)\,,\, \varepsilon\,\log \frac{k_\varepsilon \star \alpha}{k_\varepsilon \star \beta}\ \Big\rangle \\ &=~ \tfrac{1}{2}\langle\,\lambda\,\cdot\,k_\varepsilon \star (\alpha - \beta)\,,\, b^\varepsilon - a^\varepsilon\,\rangle ~~\geqslant~~ 0. \end{aligned}$$

In practice, if $(\alpha - \beta)$ is close enough to its ε-blurred image $\lambda\,\cdot\,k_\varepsilon \star (\alpha - \beta)$, $\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha,\beta)$ is thus positive too.

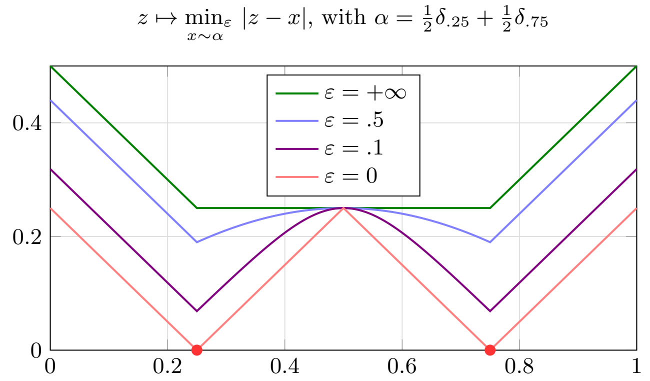
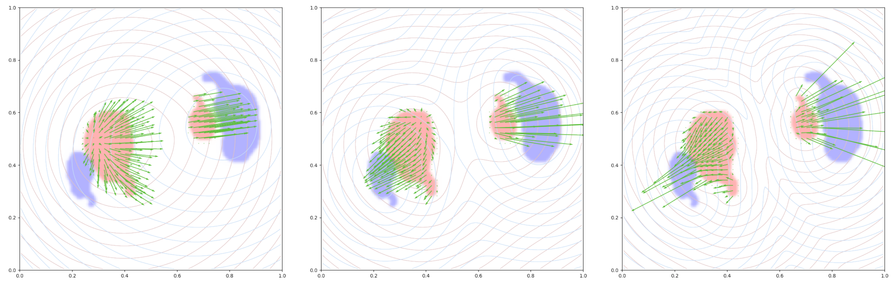
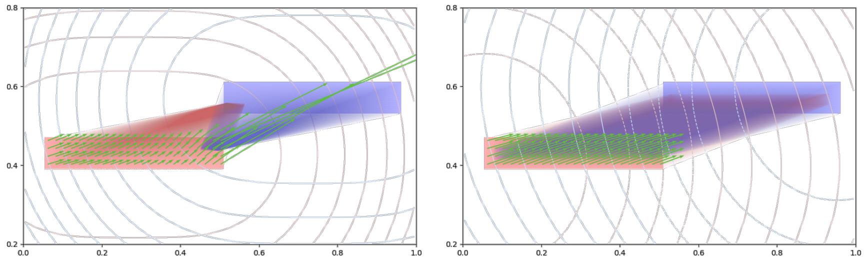


Fig. 5: **The SoftMin operator.** The log-sum-exp trick allows us to interpolate between two kinds of distance fields to a measure: the “Energy potential” $|\cdot|\star\alpha$ – for $\varepsilon = +\infty$ – and the distance field to the support $\{.25, .75\}$ of α – for $\varepsilon = 0$.



(a) SoftMin, $\varepsilon = +\infty$ (b) SoftMin, $\varepsilon = .05$ (c) SoftMin, $\varepsilon = .01$

Fig. 6: **The ε-SoftMin fidelity interpolates between the Energy Distance and a weighted Hausdorff distance between the supports [1].** Here, we use the simple Euclidean cost $C(x,y) = \|x-y\|$.



(a) SoftMin, $\sqrt{\varepsilon} = .1$ (b) Sinkhorn, $\sqrt{\varepsilon} = .1$

Fig. 7: **Naive projection isn’t the panacea.** Using an ε-SoftMin cost is equivalent to encoding our shapes through their distance images – the so-called *influence fields*, displayed in the background. (a) Unfortunately, such a cost is prone to giving a disproportionate importance to the extremities of both shapes, as points are only influenced by their *nearest neighbors*. (b) Presented in section 3, the Sinkhorn loop lets us introduce a **mass distribution constraint** to alleviate this problem: we shift the influence fields a (in red) and b (in blue) to retrieve *balanced* gradient fields. Figures computed with $\mathrm{C}(x,y) = \|x-y\|^2$.

3 Balancing distance fields: the Sinkhorn algorithm

As seen in Figure 7, adding a **mass distribution constraint** to SoftMin distances can improve the quality of our descent directions. Thankfully, this is now possible thanks to the theory of Optimal Transport, which generalizes the Wasserstein distance – we recommend the recent handbook [11] for reference.

Primal "Monge" problem. As illustrated in Figure 8, Optimal Transport is about solving a *convex registration problem*: for $\varepsilon > 0$, we strive to minimize a primal cost OT_ε defined through

$$\mathrm{OT}_\varepsilon(\alpha,\beta) = \min_{\pi_{x_i,y_j}\in\mathbb{R}^{\mathrm{N}\times\mathrm{M}}_{\geqslant 0}} \underbrace{\sum_{i,j}\pi_{x_i,y_j}\,\mathrm{C}(x_i,y_j)}_{\text{transport cost } \langle\pi,\mathrm{C}\rangle} + \underbrace{\varepsilon\sum_{i,j}\pi_{x_i,y_j}\,\log\frac{\pi_{x_i,y_j}}{\alpha_i\,\beta_j} - \pi_{x_i,y_j} + \alpha_i\beta_j}_{\text{entropic regularization, } \varepsilon\,\mathrm{KL}(\,\pi,\,\alpha\otimes\beta)},$$

under a linear constraint – α should be fully transported onto β:

$$\forall\, i \in [\![1,\mathrm{N}]\!]\,,\ \alpha_i \ = \ \textstyle\sum_{j=1}^{\mathrm{M}}\pi_{x_i,y_j} \qquad \text{and} \qquad \forall\, j \in [\![1,\mathrm{M}]\!]\,,\ \beta_j \ = \ \textstyle\sum_{i=1}^{\mathrm{N}}\pi_{x_i,y_j}\,.$$

The Sinkhorn algorithm. Strong duality holds on OT_ε. The major contribution from [3] was to show that the *dual* problem can be solved efficiently on the GPU. In a nutshell, we run the following algorithm:

```
Algorithm 1   Sinkhorn Iterative Algorithm: Sink($\alpha_i$, $x_i$, $\beta_j$, $y_j$)
Parameters : symmetric cost $\mathrm{C} : (x,y) \mapsto \mathrm{C}(x,y)$, regularization $\varepsilon > 0$
Input      : source $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$, target $\beta = \sum_{j=1}^{\mathrm{M}} \beta_j \delta_{y_j}$
Output     : influence fields $a^{\alpha\to\beta}$ and $b^{\beta\to\alpha}$, sampled on the $y_j$'s and $x_i$'s respectively

$a_{y_j} \leftarrow$ zeros(M) ; $b_{x_i} \leftarrow$ zeros(N)          ▷ Vectors of size M and N, respectively
for it = 1 to $n_{\text{its}}$ do                                  ▷ In practice, $n_{\text{its}}$ = 10 to 30 is enough
    $a_{y_j} \leftarrow \min_{\varepsilon, x\sim\alpha}[\mathrm{C}(x,y_j) - b(x)] = -\varepsilon \log \sum_{i=1}^{\mathrm{N}} \exp\big[\log(\alpha_i) - \frac{1}{\varepsilon}(\mathrm{C}(x_i,y_j) - b_{x_i})\big]$
    $b_{x_i} \leftarrow \min_{\varepsilon, y\sim\beta}[\mathrm{C}(x_i,y) - a(y)] = -\varepsilon \log \sum_{j=1}^{\mathrm{M}} \exp\big[\log(\beta_j) - \frac{1}{\varepsilon}(\mathrm{C}(x_i,y_j) - a_{y_j})\big]$
return $a_{y_j}$, $b_{x_i}$                                          ▷ Vectors of size M and N, respectively
```

And at convergence, with $(a^{\alpha\to\beta}_{y_j}, b^{\beta\to\alpha}_{x_i}) = \mathrm{Sink}(\alpha_i, x_i, \beta_j, y_j)$, we get

$$\mathrm{OT}_\varepsilon(\alpha,\beta) \ = \ \big\langle\, \alpha\,,\, b^{\beta\to\alpha} \,\big\rangle \ + \ \big\langle\, \beta\,,\, a^{\alpha\to\beta} \,\big\rangle \ = \ \big(\, \alpha_i \mid b^{\beta\to\alpha}_{x_i} \,\big) \ + \ \big(\, \beta_j \,|\, a^{\alpha\to\beta}_{y_j} \,\big).$$

Interpretation. The Sinkhorn algorithm is a block-coordinate ascent on the *dual* variables. Mathematically speaking, these are Lipschitz functions defined on the ambient space $\mathcal{X}$ and sampled on the measures' supports. As illustrated in Fig. 9 and detailed in [6], we propose to understand them as *influence fields* $a^{\alpha\to\beta}$ and $b^{\beta\to\alpha}$ that encode an implicit transport plan $\pi = \exp\big[\frac{1}{\varepsilon}(b \oplus a - \mathrm{C})\big] \cdot \alpha \otimes \beta$.

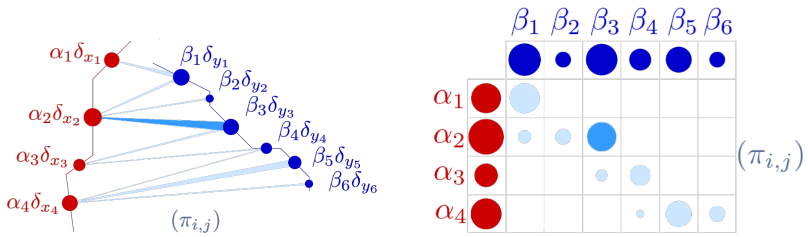
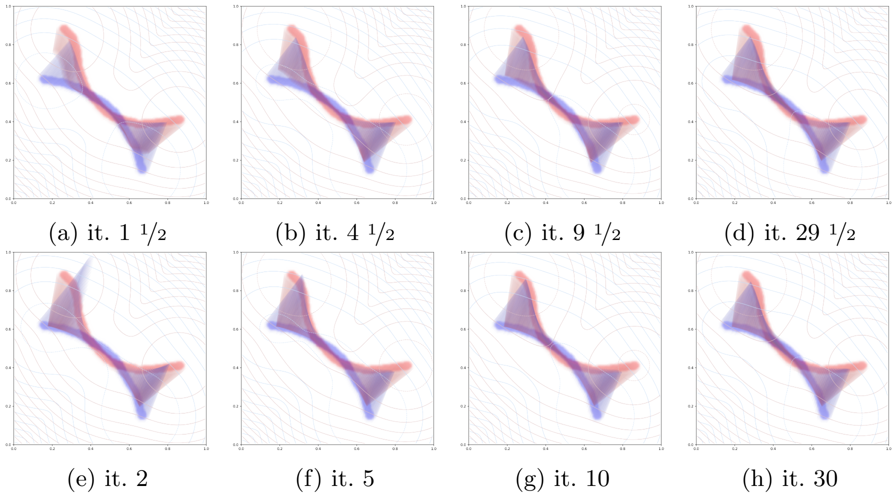
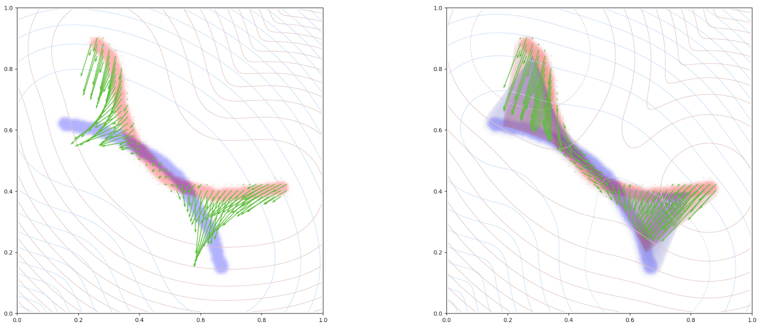


Fig. 8: **Looking for a low-cost mapping, from α to β.** As we solve the Optimal Transport problem, we find a *transport plan* π whose marginals are equal to α and β respectively. In practice, we solve a regularized problem whose dual solution is easy to compute on the GPU.


(a) it. 1 1/2 (b) it. 4 1/2 (c) it. 9 1/2 (d) it. 29 1/2

(e) it. 2 (f) it. 5 (g) it. 10 (h) it. 30

Fig. 9: **The (standard) Sinkhorn algorithm brings balance to the force.** On top of α, β, $a^{\alpha\to\beta}$ (in red) and $b^{\beta\to\alpha}$ (in blue), we display the mean “springs” linking the x_i’s to β (in red) and the y_j’s to α (in blue). Algorithm 1 is all about normalizing the blue (line 3) and red (line 4) springs until reaching equilibrium.



(a) ε-Hausdorff, $\sqrt{\varepsilon} = .1$ (b) ε-Sinkhorn, $\sqrt{\varepsilon} = .1$

Fig. 10: **Computing ε-Hausdorff and ε-Sinkhorn divergences.** On this page, we use a quadratic cost $\mathrm{C}(x, y) = \|x - y\|^2$ so that ε is homogeneous to a squared distance. As evidenced in (b), the gradient of the ε-Sinkhorn divergence is good enough for one-shot registration in simple cases.

Towards a positive Optimal Transport cost. In [5], we advocated the use of $\mathrm{OT}_\varepsilon(\alpha,\beta)$ as a data attachment term for diffeomorphic registration. Unfortunately though, just as if we used an ε-SoftMin fidelity, the minimum of the loss functional $\alpha \mapsto \mathrm{OT}_\varepsilon(\alpha,\beta)$ is *not* reached when α is exactly equal to β.

With collaborators [6], we thus decided to shift our attention towards a new *geometric* entropy:

$$\mathrm{F}_\varepsilon(\alpha) \;=\; -\tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) = \;\varepsilon \min_{\mu_i \in \mathbb{R}^{\mathrm{N}}_{\geqslant 0}} \Big\langle\, \alpha \,,\, \log \tfrac{\mathrm{d}\alpha}{\mathrm{d}\mu} \Big\rangle + \tfrac{1}{2}\langle\, \mu \,,\, k_\varepsilon \star \mu \,\rangle - \tfrac{1}{2}, \qquad (5)$$

with $\mu = \sum_{i=1}^{\mathrm{N}} \mu_i \delta_{x_i}$ – this identity stands thanks to a change of variable "$\mu_i = \exp(a_{x_i}/\varepsilon)\,\alpha_i$". We let SymSink denotes the *symmetrized* Sinkhorn algorithm:

Algorithm 2 Symmetric Sinkhorn Algorithm: SymSink(α_i, x_i, y_j)

Parameters : symmetric cost $\mathrm{C} : (x,y) \mapsto \mathrm{C}(x,y)$, regularization $\varepsilon > 0$
Input : source $\alpha = \sum_{i=1}^{\mathrm{N}} \alpha_i \delta_{x_i}$, target point cloud $(y_j)_{j \in [\![1,\mathrm{M}]\!]}$
Output : influence field $a^{\alpha\leftrightarrow\alpha}$ sampled on the x_i's and the y_j's

```
$a_{x_i} \leftarrow \mathrm{zeros}(\mathrm{N})$                                ▷ Vector of size N
for it $= 1$ to $n_{\mathrm{its}} - 1$ do                      ▷ In practice, $n_{\mathrm{its}} = 3$ is enough
    $a_{x_i} \leftarrow \frac{1}{2}(a_{x_i} \;+\; \min_{\varepsilon, x\sim\alpha}\,[\mathrm{C}(x_i,x) - a(x)])$
        $= \frac{1}{2}\Big(a_{x_i} \;-\; \varepsilon \,\log \sum_{k=1}^{\mathrm{N}} \exp\big[\log(\alpha_k) - \frac{1}{\varepsilon}(\mathrm{C}(x_i,x_k) - a_{x_k})\big]\Big)$
$a'_{x_i} \leftarrow \min_{\varepsilon, x\sim\alpha}[\mathrm{C}(x_i,x) - a(x)] = -\varepsilon\,\log \sum_{k=1}^{\mathrm{N}} \exp\big[\log(\alpha_k) - \frac{1}{\varepsilon}(\mathrm{C}(x_i,x_k) - a_{x_k})\big]$
$a''_{y_j} \leftarrow \min_{\varepsilon, x\sim\alpha}[\mathrm{C}(y_j,x) - a(x)] = -\varepsilon\,\log \sum_{k=1}^{\mathrm{N}} \exp\big[\log(\alpha_k) - \frac{1}{\varepsilon}(\mathrm{C}(y_j,x_k) - a_{x_k})\big]$

return $a''_{y_j}$, $a'_{x_i}$                      ▷ Vectors of size M and N, respectively
```

The ε-Sinkhorn divergence. Then, we can define

$$\begin{aligned}
\mathrm{d}_{\varepsilon\text{-Hausdorff}}(\alpha,\beta) \;&=\; \tfrac{1}{2}\langle\, \alpha - \beta \,,\, \nabla\mathrm{F}_\varepsilon(\alpha) - \nabla\mathrm{F}_\varepsilon(\beta)\,\rangle \\
&=\; \tfrac{1}{2}\big(\, \alpha_i \mid b^{\beta\leftrightarrow\beta}_{x_i} - a^{\alpha\leftrightarrow\alpha}_{x_i} \,\big) \;+\; \tfrac{1}{2}\big(\, \beta_j \,|\, a^{\alpha\leftrightarrow\alpha}_{y_j} - b^{\beta\leftrightarrow\beta}_{y_j} \,\big), \\
\mathrm{d}_{\varepsilon\text{-Sinkhorn}}(\alpha,\beta) \;&=\; \mathrm{OT}_\varepsilon(\alpha,\beta) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\alpha,\alpha) - \tfrac{1}{2}\mathrm{OT}_\varepsilon(\beta,\beta) \\
&=\; \big(\, \alpha_i \mid b^{\beta\rightarrow\alpha}_{x_i} - a^{\alpha\leftrightarrow\alpha}_{x_i} \,\big) \;+\; \big(\, \beta_j \,|\, a^{\alpha\rightarrow\beta}_{y_j} - b^{\beta\leftrightarrow\beta}_{y_j} \,\big),
\end{aligned}$$

with $(a^{\alpha\leftrightarrow\alpha}_{y_j}, a^{\alpha\leftrightarrow\alpha}_{x_i}) = \mathrm{SymSink}(\alpha_i, x_i, y_j)$, $(b^{\beta\leftrightarrow\beta}_{x_i}, b^{\beta\leftrightarrow\beta}_{y_j}) = \mathrm{SymSink}(\beta_j, y_j, x_i)$ and $(a^{\alpha\rightarrow\beta}_{y_j}, b^{\beta\rightarrow\alpha}_{x_i}) = \mathrm{Sink}(\alpha_i, x_i, \beta_j, y_j)$.

The ε-Hausdorff divergence is the symmetrized Bregman divergence associated to F_ε on the space of probability measures on $\mathcal{X}$, and can be shown to behave like the ε-SoftMin cost at the small and large ε limits. Meanwhile, the ε-Sinkhorn divergence is an "unbiased" Optimal Transport cost that has been recently introduced in the Machine Learning community [7].

The intuition here is that since $\mathrm{OT}_\varepsilon(\alpha,\beta)$ converges towards a kernel scalar product $\langle \alpha, \mathrm{C} \star \beta \rangle$ when ε goes to infinity, adding the self-correlation corrective terms lets us converge towards a genuine kernel squared norm $\frac{1}{2}\|\alpha - \beta\|^2_{-\mathrm{C}}$ – say, the Energy Distance if $\mathrm{C}(x,y) = \|x - y\|$. Most importantly, we are able to prove that both formulas define *positive* divergences for $\varepsilon > 0$:

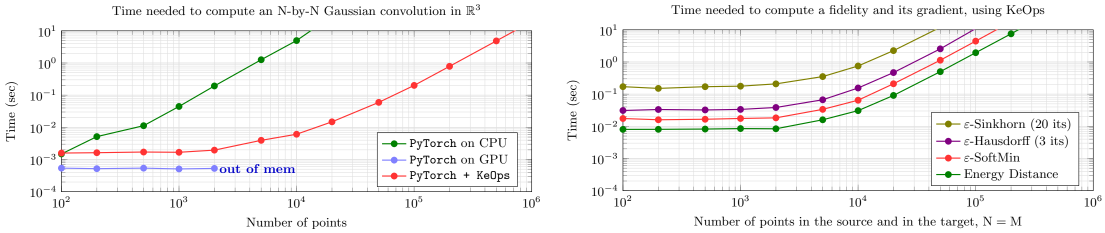
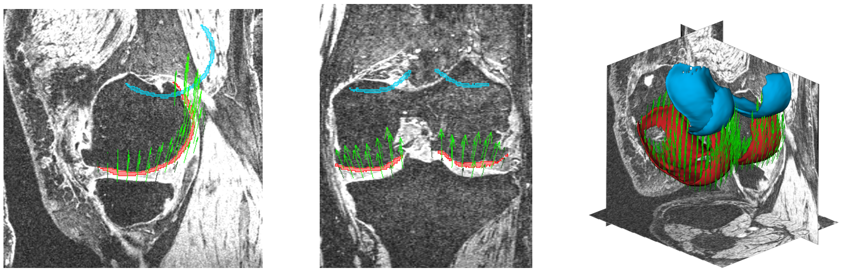


Fig. 11: **The KeOps library lets shape analysis routines scale up to real data.** Performances on a cheap laptop's GPU (GTX 960M). (a) As it provides CUDA routines for online map-reduce operations, our "KErnelOPerationS" library – developed with Benjamin Charlier and Joan A. Glaunès – allows Matlab, numpy and pytorch users to compute huge N-by-N convolutions without having to store large kernel matrices in the GPU memory. (b) Experiments performed on point clouds in $\mathbb{R}^3$, endowed with a Euclidean cost $C(x, y) = \|x - y\|$.



(a) sagittal (b) coronal (c) 3D Slicer view [4]

Fig. 12: **On real data.** Our routines could be used to registrate thin structures such as these knee caps from the OsteoArthritis Initiative – special thanks to Zhenlin Xu and Marc Niethammer for letting us know about this dataset. Here, the source and target volumes are respectively made up of 52,319 and 34,966 voxels – out of a 192-192-160 volume. As advertised in Figure 11, this Energy Distance's gradient was computed in half a second on the author's laptop.

Theorem 1 (Positivity). *Let α and β be two positive measures with finite support and same total mass on a feature space $\mathcal{X}$. Let us choose a smoothing scale $\varepsilon > 0$ and a cost function* C *on $\mathcal{X} \times \mathcal{X}$ such that*

$$k_\varepsilon(x, y) \;=\; \exp(-\mathrm{C}(x, y)/\varepsilon)$$

defines a ***positive kernel*** *function on $\mathcal{X}$. Then, one can show that*

$$0 \;\leqslant\; d_{\varepsilon\text{-}Hausdorff}(\alpha, \beta) \;\leqslant\; d_{\varepsilon\text{-}Sinkhorn}(\alpha, \beta),$$

with a null value if and only if $\alpha = \beta$.

Proof. The proof of this result is given in [6].
In a nutshell: the first inequality relies on the positivity of the kernel k_ε, as it ensures the convexity of the potential F_ε – Eq. (5) – and the positivity of the associated Bregman divergence. The second inequality derives from the convexity of $\mathrm{OT}_\varepsilon(\alpha, \beta)$ with respect to α and β varying independently.

4 Conclusion

Overview. All things considered, we introduced three positive divergences to the shape analysis community: the cheap and global Energy Distance; the high-quality ε-Sinkhorn cost; and, sitting in-between, a brand new ε-Hausdorff divergence inspired by computer graphics. All of them define well-posed, differentiable loss functions for registration problems.

As we linked these theories with each other in sections 2 and 3, we were able to provide important theoretical guarantees and efficient GPU routines. In practice, we advocate the use of the PyTorch + KeOps combination [10,2] that provides *automatic differentiation* and scalability to shapes with up to 100,000 active vertices.

Going further. Now, which one of these formulas should we use in practice? As seen in Figure 10, using an ε-Sinkhorn divergence is equivalent to performing a full convex registration – with no guarantee of topology preservation – every time we need a descent direction... Do we really need to go that far?

The answer to this question is highly dependent on the remainder of the registration pipeline. In months to come, we thus plan to test our new fidelities in a wide range of settings – from standard LDDMM to Deep Learning based methods – as we strive to provide our colleagues with reliable tools.

References

1. Aspert, N., Santa-Cruz, D., Ebrahimi, T.: Mesh: Measuring errors between surfaces using the hausdorff distance. In: Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on. vol. 1, pp. 705–708. IEEE (2002)
2. Charlier, B., Feydy, J., Glaunès, J.: Kernel operations on the gpu, with autodiff, without memory overflows. `www.kernel-operations.io`, accessed: 2018-08-15
3. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Advances in neural information processing systems. pp. 2292–2300 (2013)
4. Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Finet, J., et al.: 3d slicer as an image computing platform for the quantitative imaging network. Magnetic resonance imaging **30**(9), 1323–1341 (2012)
5. Feydy, J., Charlier, B., Vialard, F.X., Peyré, G.: Optimal transport for diffeomorphic registration. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 291–299. Springer (2017)
6. Feydy, J., Séjourné, T., Vialard, F.X., Amari, S.i., Trouvé, A., Peyré, G.: Sinkhorn entropies and divergences. to be published in September 2018
7. Genevay, A., Peyré, G., Cuturi, M.: Learning generative models with sinkhorn divergences. In: Storkey, A., Perez-Cruz, F. (eds.) Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 84, pp. 1608–1617. PMLR (09–11 Apr 2018)
8. Kaltenmark, I., Charlier, B., Charon, N.: A general framework for curve and surface comparison and registration with oriented varifolds. In: Computer Vision and Pattern Recognition (CVPR) (2017)
9. Lombaert, H., Grady, L., Pennec, X., Ayache, N., Cheriet, F.: Spectral log-demons: diffeomorphic image registration with very large deformations. International journal of computer vision **107**(3), 254–271 (2014)
10. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
11. Peyré, G., Cuturi, M.: Computational optimal transport. arXiv preprint arXiv:1803.00567 (2018)
12. Székely, G.J., Rizzo, M.L.: Energy statistics: A class of statistics based on distances. Journal of statistical planning and inference **143**(8), 1249–1272 (2013)