Global divergences between measures: from Hausdorff distance to Optimal Transport

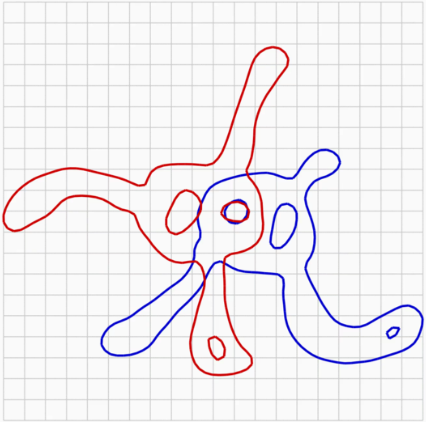
Jean Feydy Alain Trouvé

Curves and Surfaces, Arcachon – 2 juillet 2018

Écoles Normales Supérieures de Paris et Paris-Saclay

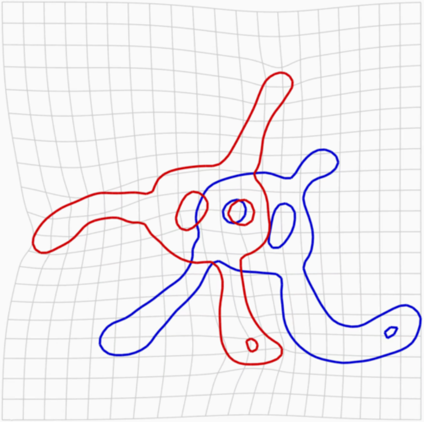
Today: focus on shape registration

Source A, target B,



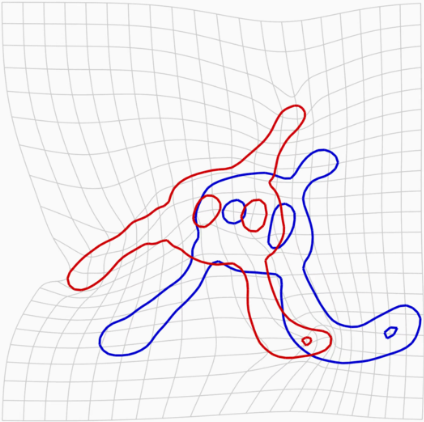
Today: focus on shape registration

Source A, target B, mapping φ



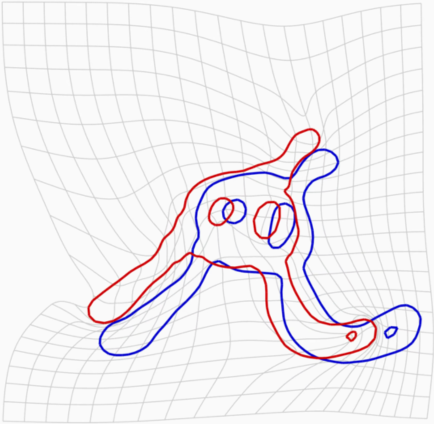
Today: focus on shape registration

Source A, target B, mapping φ



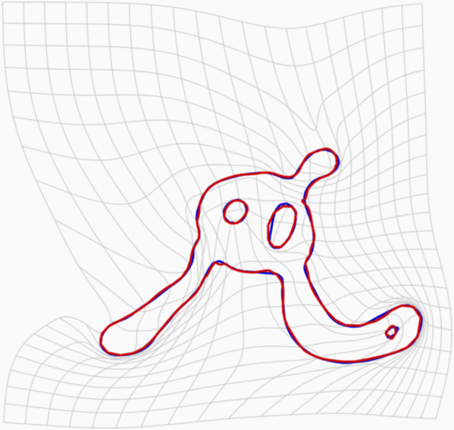
Today: focus on shape registration

Source A, target B, mapping φ



Today: focus on shape registration

Source A, target B, mapping φ



In practice: gradient descent on the deformation

$$\mathrm{Cost}(\varphi) \;=\; \underbrace{\mathrm{Reg}(\varphi)}_{\text{regularization}} \;+\; \underbrace{\mathrm{d}(\varphi(A)\,,B)}_{\text{fidelity}}$$

In practice: gradient descent on the deformation

$$\text{Cost}(\varphi) \;=\; \underbrace{\text{Reg}(\varphi)}_{\text{regularization}} \;+\; \underbrace{\text{d}(\varphi(A)\,,B)}_{\text{fidelity}}$$

If A and B are labeled vectors of $\mathbb{R}^{N\times D}$, you can use

Affine registration: $\text{Cost}(\varphi) \;=\; 1_{\text{affine}}(\varphi) + \|\varphi(A) \;-\; B\|_2^2$

Thin Plate Splines: $\text{Cost}(\varphi) \;=\; \lambda\,\|\Delta\varphi\|_2^2 + \|\varphi(A) \;-\; B\|_2^2$

In practice: gradient descent on the deformation

$$\text{Cost}(\varphi) \;=\; \underbrace{\text{Reg}(\varphi)}_{\text{regularization}} \;+\; \underbrace{\text{d}(\varphi(A)\,,B)}_{\text{fidelity}}$$

Iterative Matching Algorithm

```
$\varphi \leftarrow \text{Id}$
while updates $>$ tol do
    "$\varphi \leftarrow \varphi \,-\, \alpha \cdot \big(\nabla_\varphi \text{Reg}(\varphi) + \nabla_\varphi\, [\text{d}(\varphi(A), B)]\,\big)$"
return $\varphi$
Output : matching transformation $\varphi$.
```

In practice: gradient descent on the deformation

$$\text{Cost}(\varphi) = \underbrace{\text{Reg}(\varphi)}_{\text{regularization}} + \underbrace{\text{d}(\varphi(A), B)}_{\text{fidelity}}$$

Iterative Matching Algorithm

```
$\varphi \leftarrow \text{Id}$
while updates $>$ tol do
    "$\varphi \leftarrow \varphi - \alpha \cdot \big(\nabla_\varphi \text{Reg}(\varphi) + \nabla_\varphi\, [\text{d}(\varphi(A), B)]\,\big)$"
return $\varphi$
Output : matching transformation $\varphi$.
```

⇒ The fidelity's gradient **drives** the registration

Encoding unlabeled shapes as measures

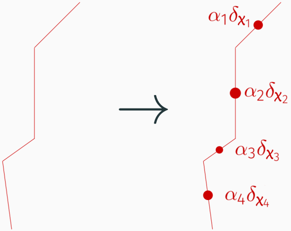
Let's enforce sampling invariance:

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

Encoding unlabeled shapes as measures

Let's enforce sampling invariance:

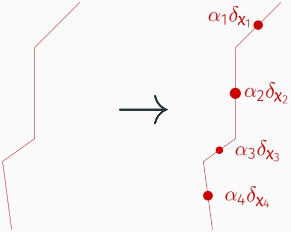
$$A \longrightarrow \alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \qquad B \longrightarrow \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}.$$

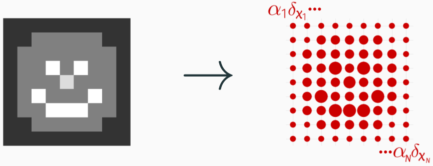


Encoding unlabeled shapes as measures

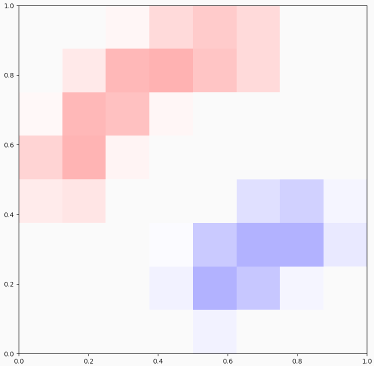
Let's enforce sampling invariance:

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

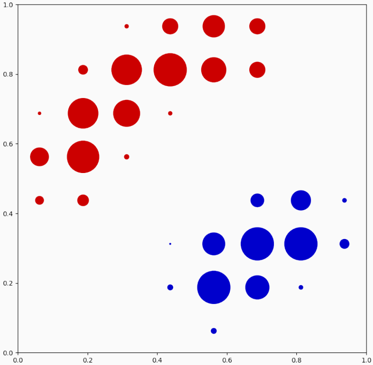




A baseline setting: density registration

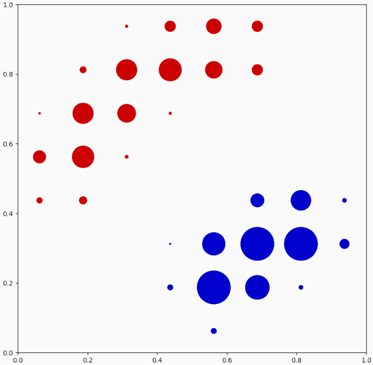


A baseline setting: density registration



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

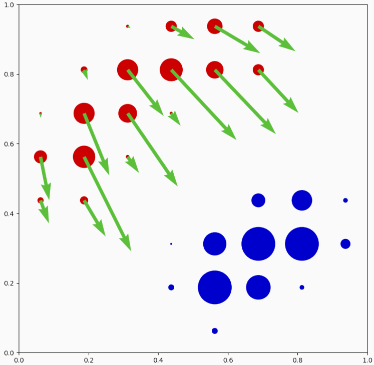
A baseline setting: density registration



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

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

A baseline setting: density registration

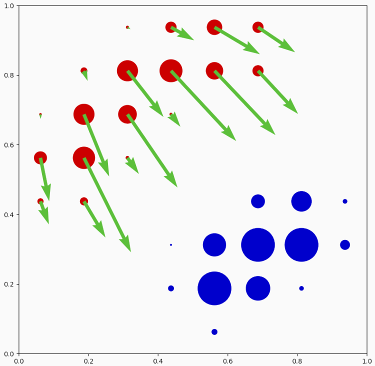


$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

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

Display $\nu \;=\; -\nabla_{x_i} \mathrm{d}(\alpha, \beta)$.

A baseline setting: density registration



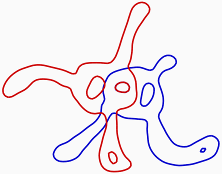
$$\alpha = \sum_{i=1}^{N} \alpha_i \delta_{x_i}\,, \quad \beta = \sum_{j=1}^{M} \beta_j \delta_{y_j}\,.$$

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

Display $\nu \;=\; -\nabla_{x_i} \mathrm{d}(\alpha, \beta)$.

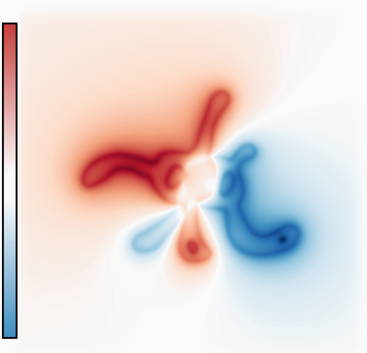
→ seamless extensions to $\sum_i \alpha_i \neq \sum_j \beta_j$ [CPSV18],
curves and surfaces [KCC17].

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$



Raw signal ($\alpha - \beta$).

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$

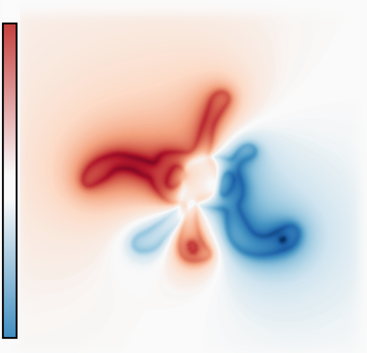


Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\mathrm{d}_k(\alpha, \beta) \;= \| \, g \star \alpha - g \star \beta \, \|^2_{L^2}$$

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$

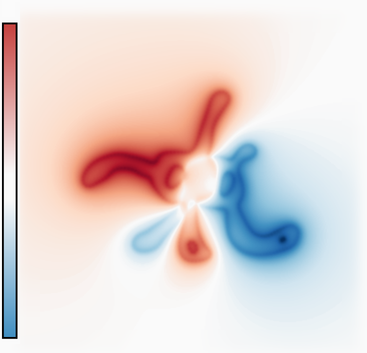


Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\begin{aligned} \mathrm{d}_k(\alpha, \beta) &= \| g \star \alpha - g \star \beta \|_{L^2}^2 \\ &= \langle \alpha - \beta \, | \, k \star (\alpha - \beta) \rangle \end{aligned}$$

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$

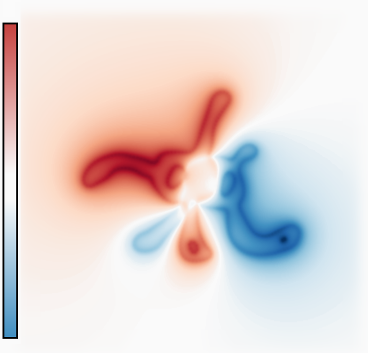


Blurred signal $g \star (\alpha - \beta)$.

Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\begin{aligned} \mathrm{d}_k(\alpha, \beta) &= \| g \star \alpha - g \star \beta \|^2_{L^2} \\ &= \langle \alpha - \beta \,|\, k \star (\alpha - \beta) \rangle \\ &= -2 \sum_{i,j} k(x_i, y_j)\, \alpha_i \, \beta_j + \cdots \end{aligned}$$

Kernel fidelities: the simplest formula for $d(\alpha, \beta)$



Blurred signal $g \star (\alpha - \beta)$.

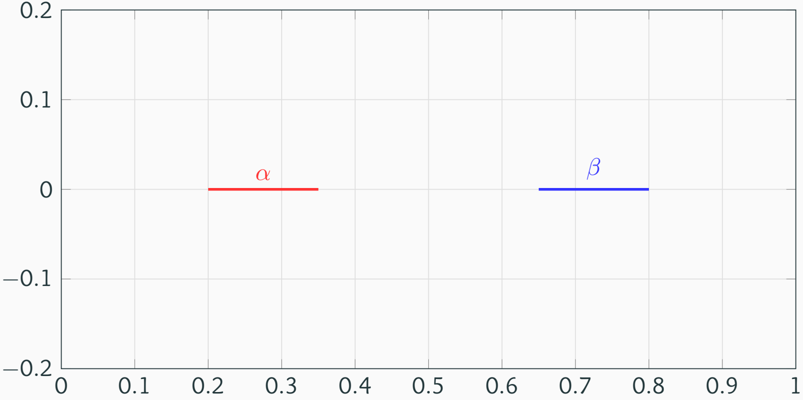
Choose a symmetric blurring function g, a **kernel** $k = g \star g$:

$$\begin{aligned} \mathrm{d}_k(\alpha, \beta) &= \| g \star \alpha - g \star \beta \|_{L^2}^2 \\ &= \langle \alpha - \beta \,|\, k \star (\alpha - \beta) \rangle \\ &= -2 \sum_{i,j} k(x_i, y_j)\, \alpha_i \, \beta_j + \cdots \\ &= \langle \alpha - \beta \,|\, b^k - a^k \rangle \end{aligned}$$

with $a^k = -k \star \alpha$, $b^k = -k \star \beta$.

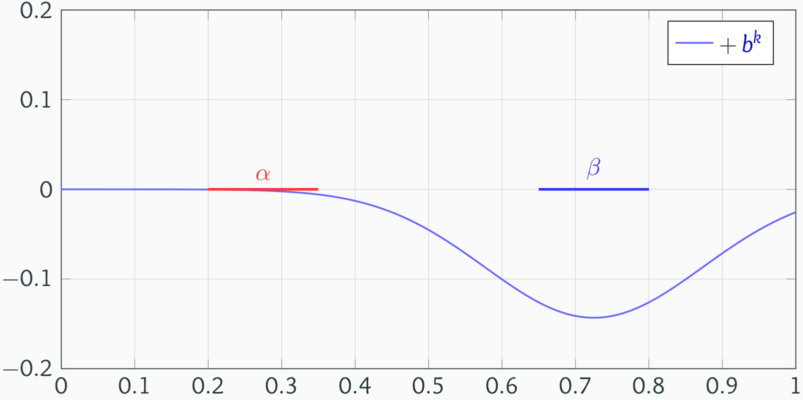
The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; \exp(-\|x-y\|^2/.2^2)$$



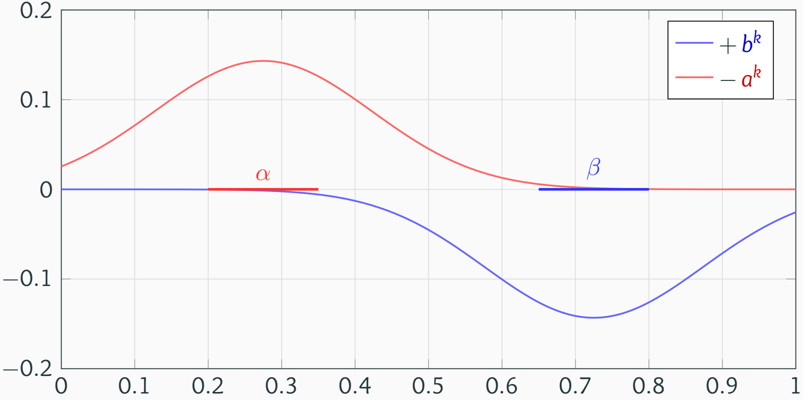
The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; \exp(-\|x-y\|^2/.2^2)$$



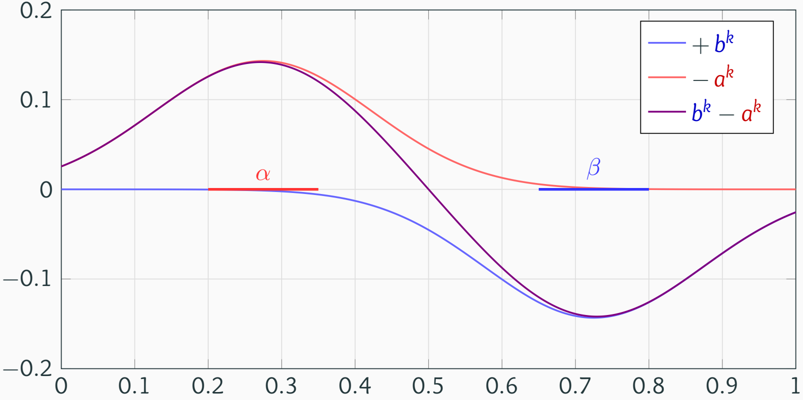
The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; \exp(-\|x-y\|^2/.2^2)$$



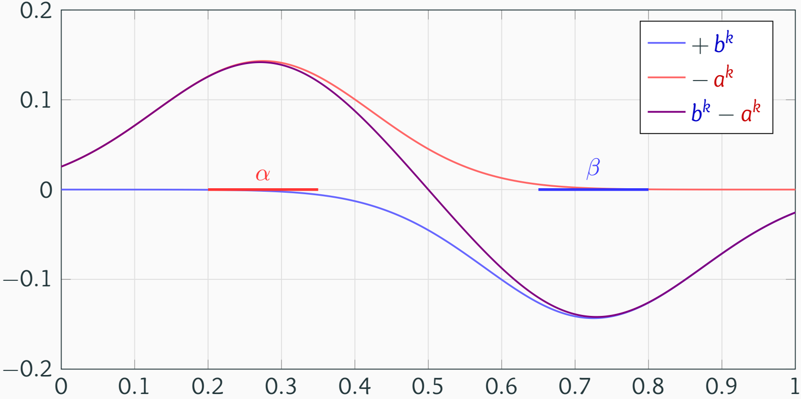
The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; \exp(-\|x-y\|^2/\,.2^2)$$



The registration flows along the gradient of $b^k - a^k$

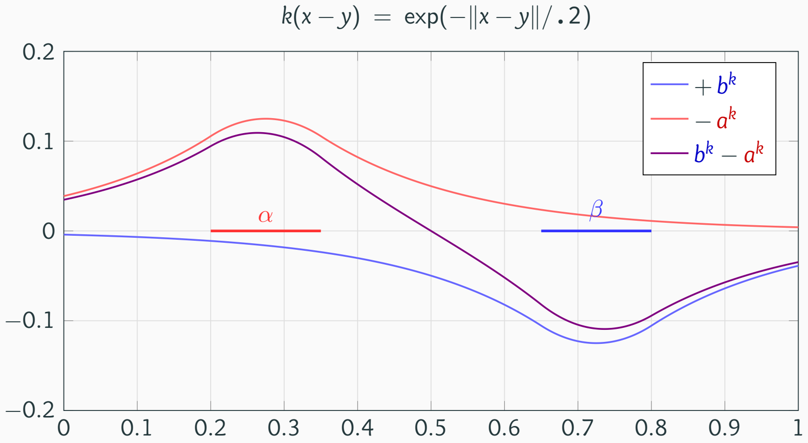
$$k(x-y) \;=\; \exp(-\|x-y\|^2/.2^2)$$



$$\mathrm{d}_k(\alpha,\beta) \;=\; \big\langle\, \alpha-\beta \,|\, k\star(\alpha-\beta)\,\big\rangle$$

$$\tfrac{1}{2}\nabla_{x_i}\mathrm{d}_k(\alpha,\beta) \;=\; \nabla\big[k\star(\alpha-\beta)\big](x_i) \;=\; \nabla b^k(x_i) - \nabla a^k(x_i)$$

The registration flows along the gradient of $b^k - a^k$

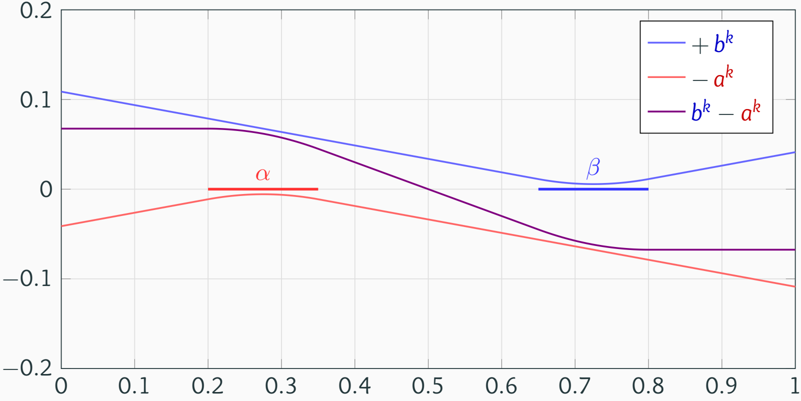


$$\mathrm{d}_k(\alpha, \beta) = \big\langle\, \alpha - \beta \,|\, k \star (\alpha - \beta) \,\big\rangle$$

$$\tfrac{1}{2}\nabla_{x_i}\mathrm{d}_k(\alpha, \beta) = \nabla\big[k \star (\alpha - \beta)\big](x_i) = \nabla b^k(x_i) - \nabla a^k(x_i)$$

The registration flows along the gradient of $b^k - a^k$

$$k(x-y) \;=\; -\|x-y\|$$

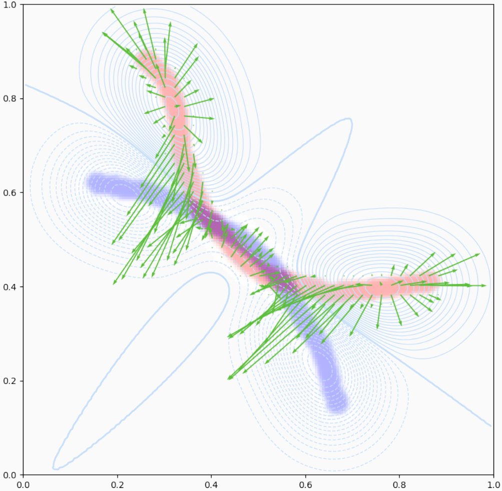


$$\mathrm{d}_k(\alpha,\beta) \;=\; \big\langle\, \alpha-\beta \;|\; k\star(\alpha-\beta)\,\big\rangle$$

$$\tfrac{1}{2}\nabla_{x_i}\mathrm{d}_k(\alpha,\beta) \;=\; \nabla\big[k\star(\alpha-\beta)\big](x_i) \;=\; \nabla b^k(x_i) - \nabla a^k(x_i)$$

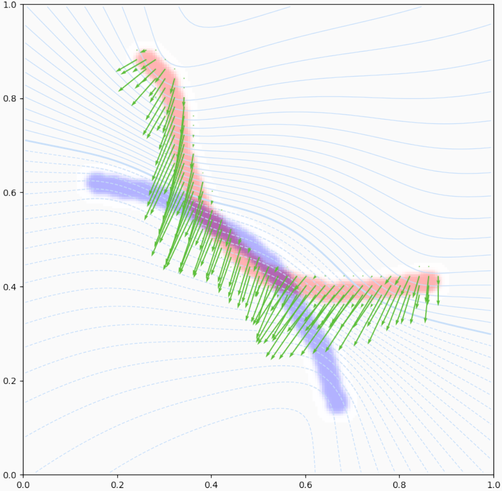
The Energy Distance is scale-invariant, robust

$$k(x - y) \;=\; \exp(-\|x - y\|^2 / .1^2)$$



The Energy Distance is scale-invariant, robust

$$k(x-y) \;=\; -\|x-y\|$$



Can we go further?

$$\begin{array}{cc} & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \begin{pmatrix} \|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\ \|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\ \vdots & \vdots & \ddots & \vdots \\ \|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\| \end{pmatrix} \end{array}$$

Can we go further?

$$\begin{array}{cc} & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \begin{pmatrix} \|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\ \|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\ \vdots & \vdots & \ddots & \vdots \\ \|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\| \end{pmatrix} \end{array}$$

Can we go further?

$$\begin{array}{cc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} &
\begin{pmatrix}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{pmatrix}
\end{array}
\begin{array}{c} \\ \to \\ \to \\ \vdots \\ \to \end{array}
\begin{array}{c} \\ \sum_j \beta_j \|x_1 - y_j\| \\ \sum_j \beta_j \|x_2 - y_j\| \\ \\ \sum_j \beta_j \|x_N - y_j\| \end{array}$$

Energy Distance : $\sum_j \beta_j \|x_i - y_j\| = b^k(x_i)$

Can we go further?

$$\begin{array}{ccl}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} & \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \begin{pmatrix} \|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\ \|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\ \vdots & \vdots & \ddots & \vdots \\ \|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\| \end{pmatrix} & \begin{array}{l} \to \quad \min_j \|x_1 - y_j\| \\ \to \quad \min_j \|x_2 - y_j\| \\ \vdots \\ \to \quad \min_j \|x_N - y_j\| \end{array}
\end{array}$$

Energy Distance : $\sum_j \beta_j \|x_i - y_j\| = b^k(x_i)$

Hausdorff Distance : $\min_j \|x_i - y_j\| = \mathrm{d}(x_i, \mathrm{supp}(\beta))$

Can we go further?

$$\begin{array}{ccc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} & \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} &
\begin{pmatrix}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{pmatrix} &
\begin{array}{ll}
\to & \text{Smin}_{\varepsilon, y\sim\beta} \|x_1 - y\| \\
\to & \text{Smin}_{\varepsilon, y\sim\beta} \|x_2 - y\| \\
\vdots & \\
\to & \text{Smin}_{\varepsilon, y\sim\beta} \|x_N - y\|
\end{array}
\end{array}$$

Energy Distance : $\sum_j \beta_j \, \|x_i - y_j\| = b^k(x_i)$

ε-SoftMin : $\text{Smin}_{\varepsilon, y\sim\beta} \, \|x_i - y\| = b^\varepsilon(x_i)$

Hausdorff Distance : $\min_j \|x_i - y_j\| = \mathrm{d}(x_i, \text{supp}(\beta))$

The log-sum-exp trick

$$\log(e^{c_1} + e^{c_2}) = c_{\max} + \log\big(\underbrace{e^{c_1 - c_{\max}} + e^{c_2 - c_{\max}}}_{\in [1,2]}\big)$$

The log-sum-exp trick

$$\log(e^{c_1} + e^{c_2}) = c_{\text{max}} + \log \big(\underbrace{e^{c_1 - c_{\text{max}}} + e^{c_2 - c_{\text{max}}}}_{\in [1,2]} \big)$$

Building on this, we define

$$b^\varepsilon(x) = \text{Smin}_{\varepsilon, y \sim \beta} \, \|x - y\|$$

The log-sum-exp trick

$$\log(\, e^{c_1} \,+\, e^{c_2} \,) \;=\; c_{\max} \,+\, \log\left(\underbrace{e^{c_1 - c_{\max}} \,+\, e^{c_2 - c_{\max}}}_{\in [1,2]}\right)$$

Building on this, we define

$$\begin{aligned} b^{\varepsilon}(x) \;&=\; \mathrm{Smin}_{\varepsilon, y \sim \beta}\, \|x - y\| \\ &=\; -\varepsilon \,\log \sum_{j=1}^{M} \exp\left(\log(\beta_j) \,-\, \tfrac{1}{\varepsilon}\|x - y_j\|\right) \end{aligned}$$

The log-sum-exp trick

$$\log(e^{c_1} + e^{c_2}) = c_{\max} + \log \big(\underbrace{e^{c_1 - c_{\max}} + e^{c_2 - c_{\max}}}_{\in [1,2]} \big)$$

Building on this, we define

$$\begin{aligned} b^\varepsilon(x) &= \text{Smin}_{\varepsilon, y\sim\beta} \, \|x-y\| \\ &= -\varepsilon \log \sum_{j=1}^{M} \exp \big(\log(\beta_j) - \tfrac{1}{\varepsilon} \|x-y_j\| \big) \\ &\xrightarrow{\varepsilon\to+\infty} \sum_{j=1}^{M} \beta_j \, \|x-y_j\| \end{aligned}$$

The log-sum-exp trick

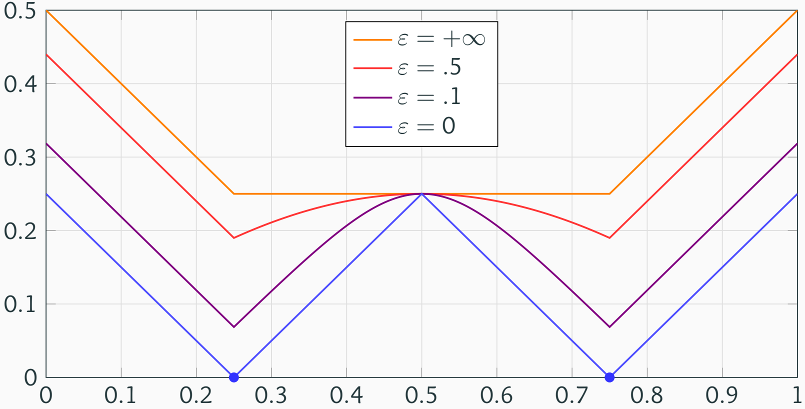
$$\log(\,e^{c_1} + e^{c_2}\,) \;=\; c_{\max} + \log\Big(\underbrace{e^{c_1 - c_{\max}} + e^{c_2 - c_{\max}}}_{\in[1,2]}\Big)$$

Building on this, we define

$$\begin{aligned}
b^\varepsilon(x) &= \operatorname{Smin}_{\varepsilon, y\sim\beta} \|x - y\| \\
&= -\varepsilon \log \sum_{j=1}^{M} \exp\Big(\log(\beta_j) - \tfrac{1}{\varepsilon}\|x - y_j\|\Big) \\
&\xrightarrow{\varepsilon\to+\infty} \sum_{j=1}^{M} \beta_j \|x - y_j\| \\
&\xrightarrow{\varepsilon\to 0} \min_j \|x - y_j\|
\end{aligned}$$

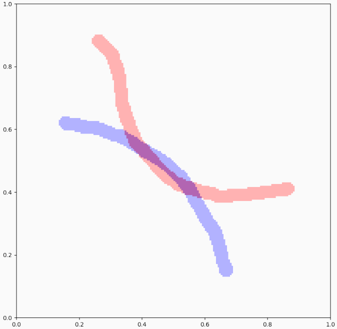
$\mathrm{Smin}_\varepsilon$ interpolates between a sum and a minimum

$x \mapsto \mathrm{Smin}_{\varepsilon, y\sim\beta} |x - y|$, with $\beta = \frac{1}{2}\delta_{.25} + \frac{1}{2}\delta_{.75}$



The ε-SoftMin fidelity interpolates between ED and Hausdorff

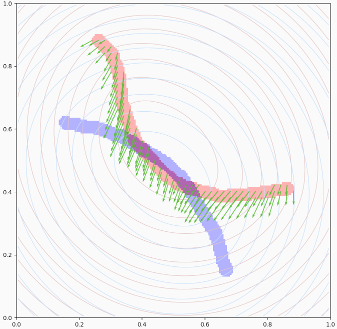
$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \qquad = \qquad \langle \alpha - \beta,\ b^\varepsilon - a^\varepsilon \rangle$$



The ε-SoftMin fidelity interpolates between ED and Hausdorff

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \quad = \quad \langle \alpha - \beta,\ b^{\varepsilon} - a^{\varepsilon} \rangle$$

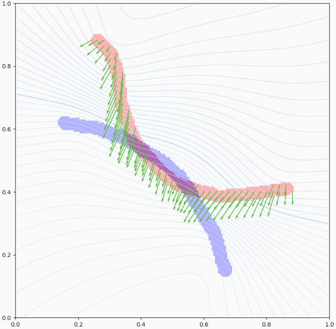
$$\xrightarrow{\varepsilon \to +\infty} \ \mathrm{d}_{\mathrm{ED}}(\alpha, \beta) \ = \ \|\alpha - \beta\|^2_{-|\cdot|}$$



The ε-SoftMin fidelity interpolates between ED and Hausdorff

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \quad = \quad \langle \alpha - \beta,\ b^{\varepsilon} - a^{\varepsilon} \rangle$$

$$\xrightarrow{\varepsilon \to +\infty} \ \mathrm{d}_{\mathrm{ED}}(\alpha, \beta) \ = \ \|\alpha - \beta\|^2_{-|\cdot|}$$

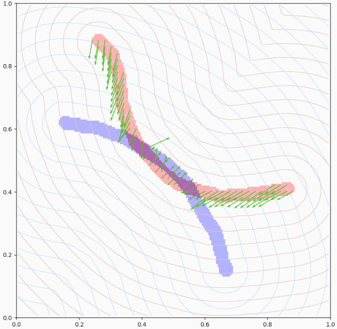


The ε-SoftMin fidelity interpolates between ED and Hausdorff

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha,\beta) \quad = \quad \langle \alpha-\beta,\ b^\varepsilon - a^\varepsilon\rangle$$

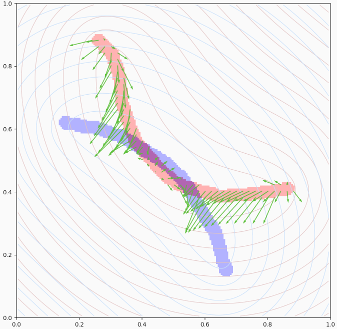
$$\xrightarrow{\varepsilon\to+\infty} \mathrm{d}_{\mathrm{ED}}(\alpha,\beta) \;=\; \|\alpha-\beta\|^2_{-|\cdot|}$$

$$\xrightarrow{\varepsilon\to 0} \sum_i \alpha_i \min_{y\sim\beta} \|x_i - y\| \;+\; \sum_j \beta_j \min_{x\sim\alpha} \|x - y_j\|$$



The ε-SoftMin fidelity interpolates between ED and Hausdorff

$$\begin{aligned}
\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha,\beta) \quad &= \quad \langle \alpha-\beta,\ b^\varepsilon - a^\varepsilon \rangle \\
&\xrightarrow{\varepsilon\to+\infty} \mathrm{d}_{\mathrm{ED}}(\alpha,\beta) \ = \ \|\alpha-\beta\|^2_{-|\cdot|} \\
&\xrightarrow{\varepsilon\to 0} \sum_i \alpha_i \min_{y\sim\beta} \|x_i - y\| \ + \ \sum_j \beta_j \min_{x\sim\alpha} \|x - y_j\|
\end{aligned}$$



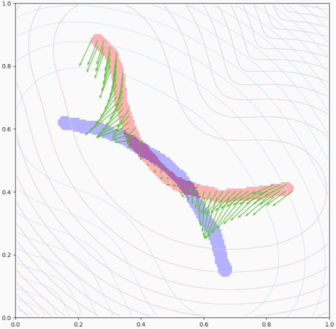
The ε-SoftMin fidelity interpolates between ED and Hausdorff

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) = \langle \alpha - \beta,\ b^\varepsilon - a^\varepsilon \rangle$$

$$\xrightarrow{\varepsilon \to +\infty} \mathrm{d}_{\mathrm{ED}}(\alpha, \beta) = \|\alpha - \beta\|^2_{-|\cdot|}$$

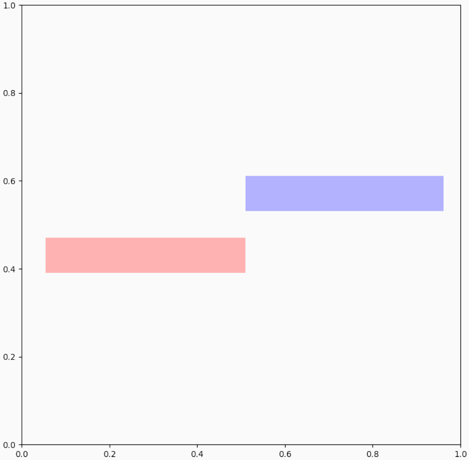
$$\xrightarrow{\varepsilon \to 0} \sum_i \alpha_i \min_{y \sim \beta} \|x_i - y\| + \sum_j \beta_j \min_{x \sim \alpha} \|x - y_j\|$$

You can also use it with $C(x, y) = \|x - y\|^2$, etc.



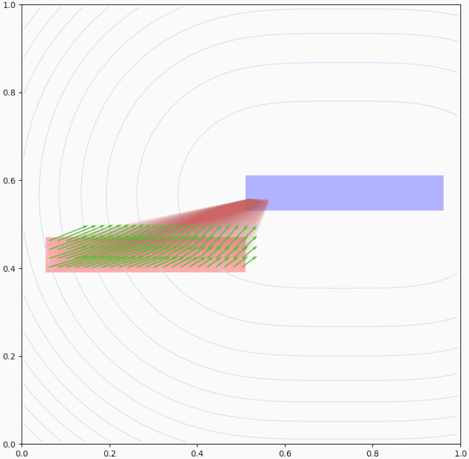
Shape registration isn't *always* about naive projections...

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \quad = \quad \langle\, \alpha,\ b^{\varepsilon} - a^{\varepsilon} \,\rangle \ + \ \langle\, \beta,\ a^{\varepsilon} - b^{\varepsilon} \,\rangle$$



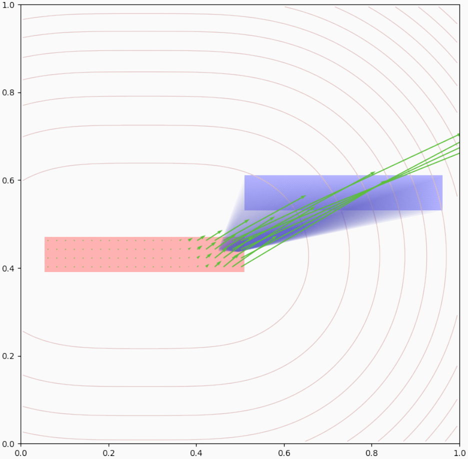
Shape registration isn't *always* about naive projections...

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \quad = \quad \langle\, \alpha,\ b^\varepsilon - a^\varepsilon \,\rangle \ + \ \langle\, \beta,\ a^\varepsilon - b^\varepsilon \,\rangle$$



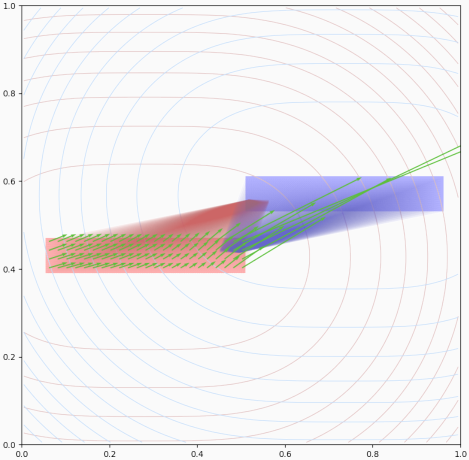
Shape registration isn't *always* about naive projections...

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \quad = \quad \langle\, \alpha,\ b^\varepsilon - a^\varepsilon \,\rangle \ + \ \langle\, \beta,\ a^\varepsilon - b^\varepsilon \,\rangle$$



Shape registration isn't *always* about naive projections...

$$\mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta) \quad = \quad \langle\, \alpha,\, b^{\varepsilon} - a^{\varepsilon} \,\rangle \;+\; \langle\, \beta,\, a^{\varepsilon} - b^{\varepsilon} \,\rangle$$



Optimal Transport = Hausdorff + mass repartition constraint

Computational Optimal Transport [Cut13, PC18]:

Enforce a **mass repartition** constraint through alternating projections onto α and β.

Optimal Transport = Hausdorff + mass repartition constraint

Computational Optimal Transport [Cut13, PC18]:

Enforce a **mass repartition** constraint through alternating projections onto α and β.

Baseline algorithm:

Use this **Sinkhorn** loop to compute $a^{\alpha\to\beta}$ and $b^{\beta\to\alpha}$.

Define $\mathcal{W}_\varepsilon(\alpha, \beta) = \langle\, \alpha \,,\, b^{\beta\to\alpha} \,\rangle + \langle\, \beta \,,\, a^{\alpha\to\beta} \,\rangle$

Optimal Transport = Hausdorff + mass repartition constraint

Computational Optimal Transport [Cut13, PC18]:

Enforce a **mass repartition** constraint through alternating projections onto α and β.

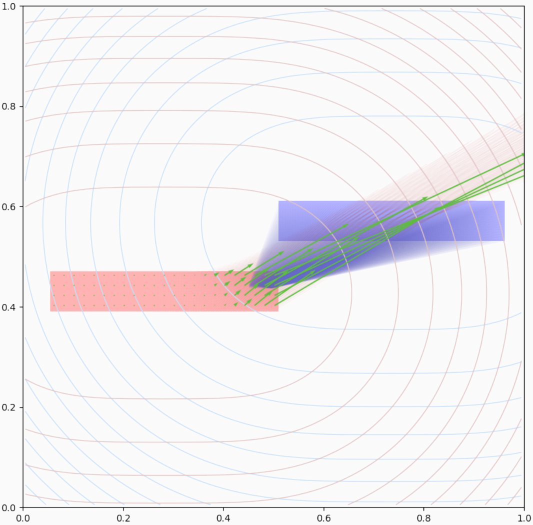
Baseline algorithm:

Use this **Sinkhorn** loop to compute $a^{\alpha\to\beta}$ and $b^{\beta\to\alpha}$.

Define $\mathcal{W}_\varepsilon(\alpha,\beta) \,=\, \langle\, \alpha\,,\, b^{\beta\to\alpha}\,\rangle \,+\, \langle\,\beta\,,\, a^{\alpha\to\beta}\,\rangle$

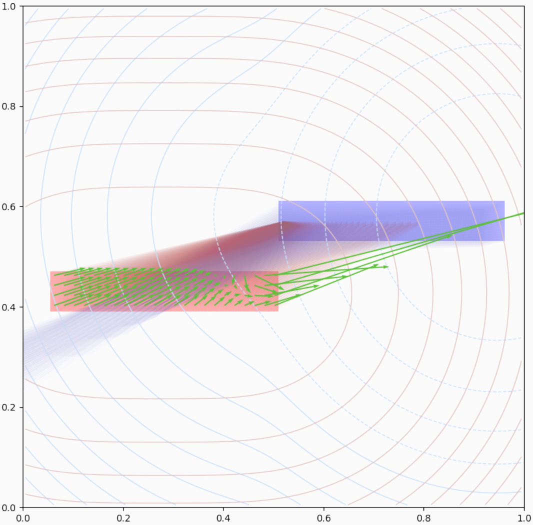
The core operation is still **Smin**$_\varepsilon$, for some $\varepsilon > 0$.

The Sinkhorn algorithm, in practice; $C(x,y) = \|x - y\|^2$, $\sqrt{\varepsilon} = .1$



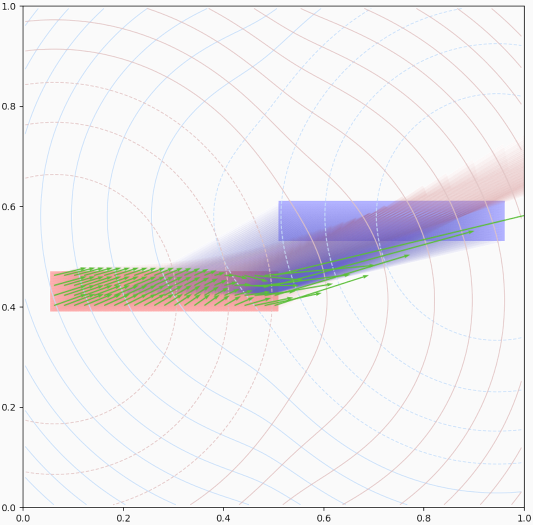
Iteration 1/2

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



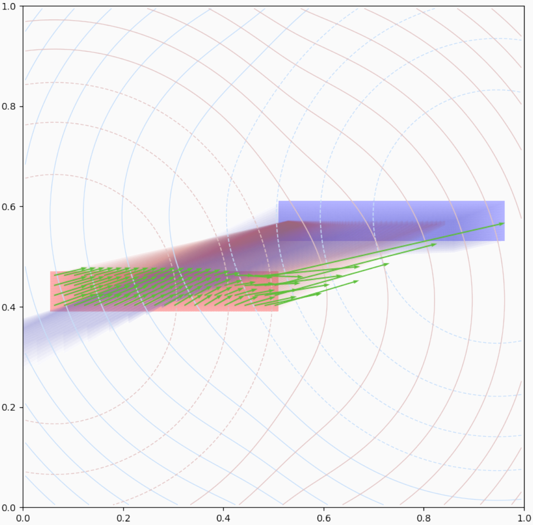
Iteration 1

The Sinkhorn algorithm, in practice; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = .1$



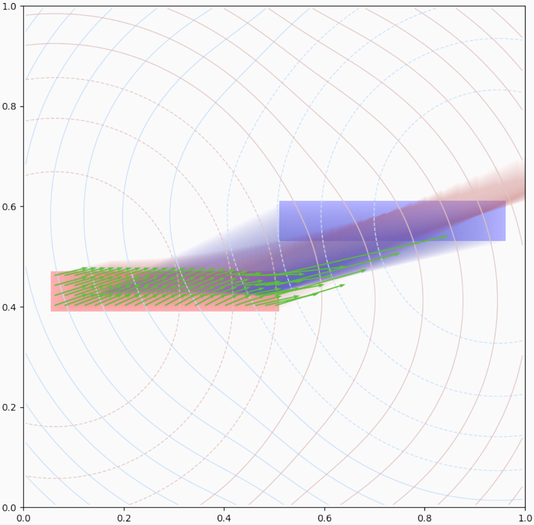
Iteration 1 ½

The Sinkhorn algorithm, in practice; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = .1$



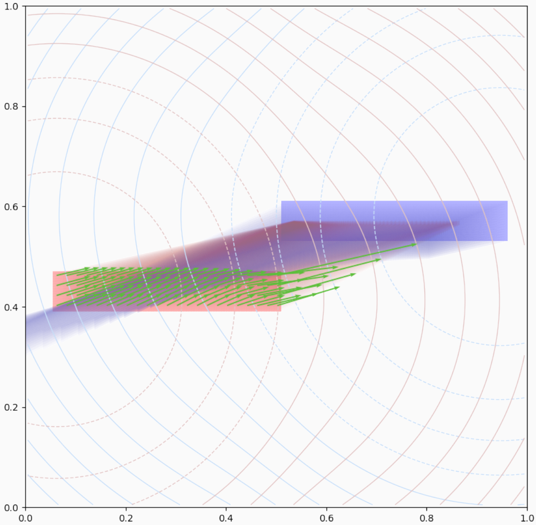
Iteration 2

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



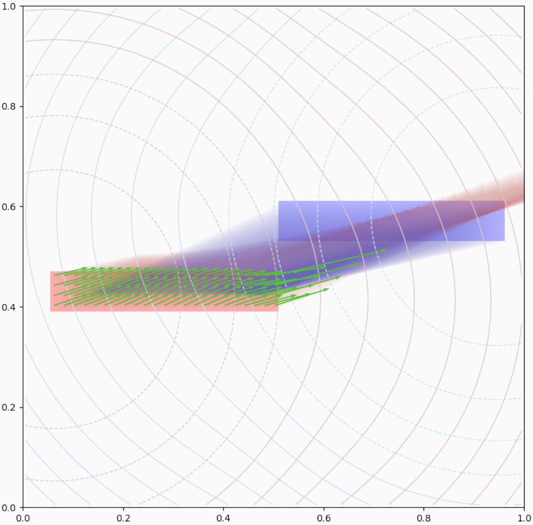
Iteration 2 ½

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



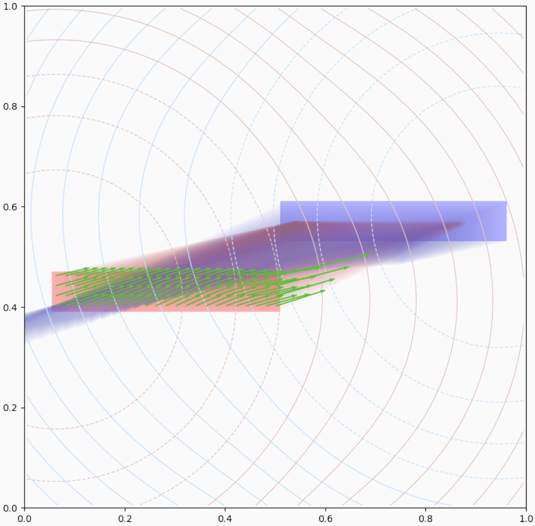
Iteration 3

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



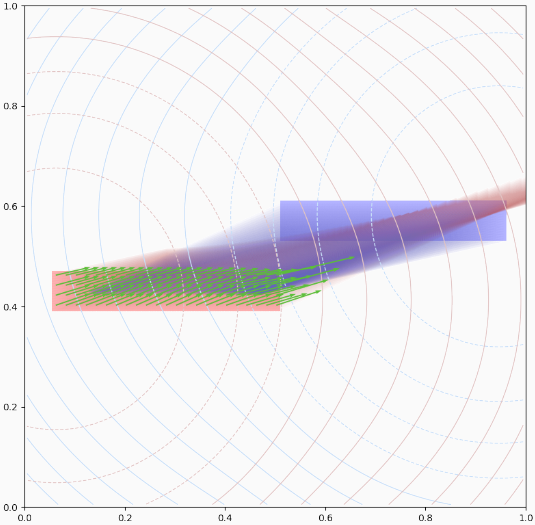
Iteration 3 ½

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



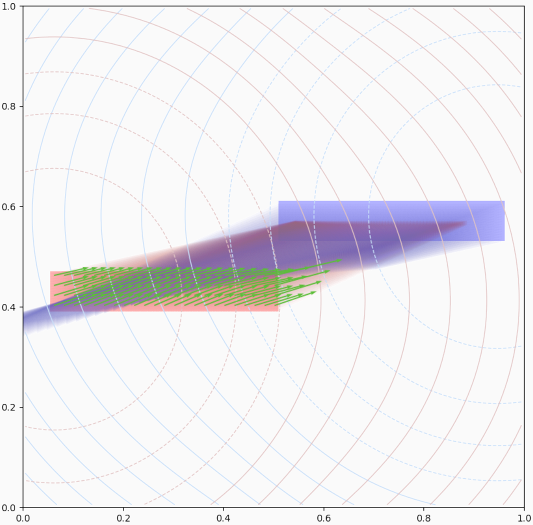
Iteration 4

The Sinkhorn algorithm, in practice; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = .1$



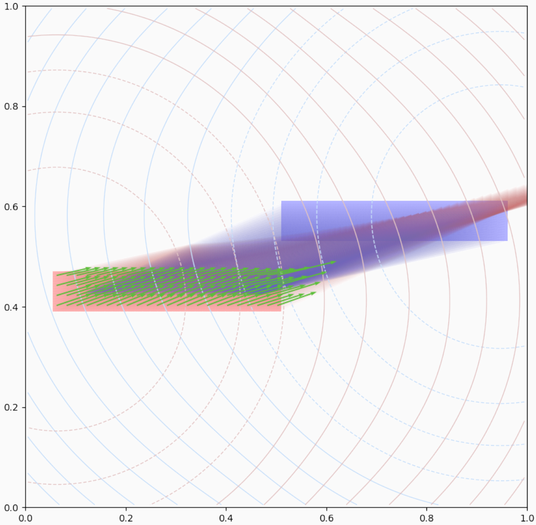
Iteration 4 ½

The Sinkhorn algorithm, in practice; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = .1$



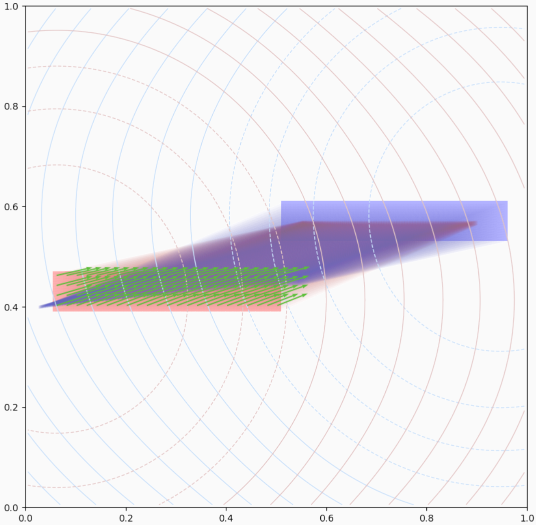
Iteration 5

The Sinkhorn algorithm, in practice; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = .1$



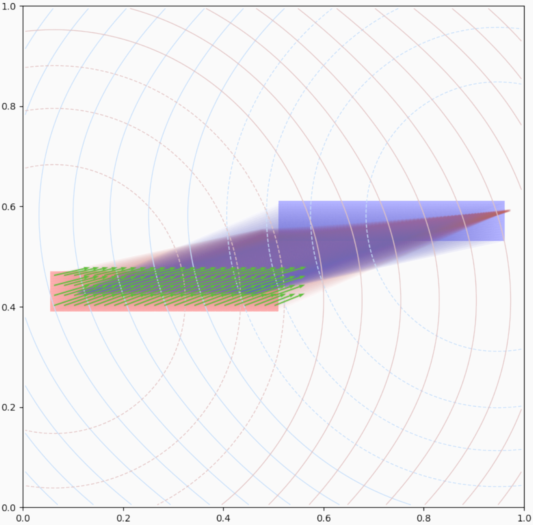
Iteration 5 ½

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



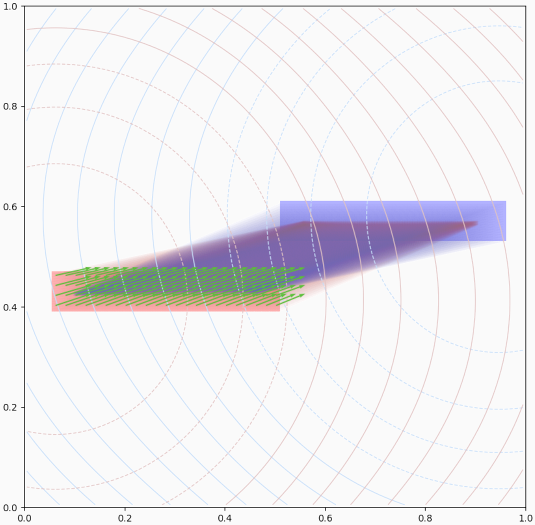
Iteration 10

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



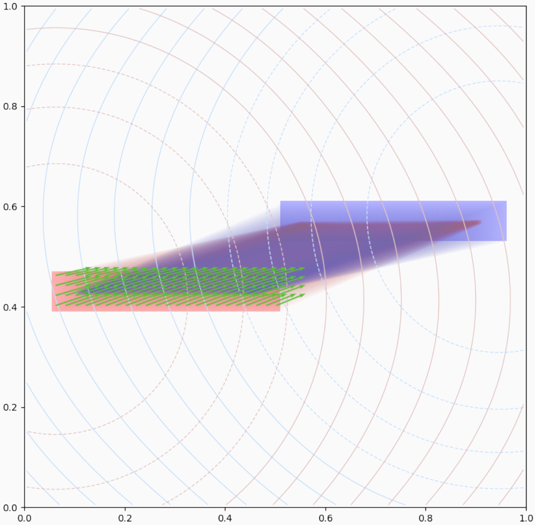
Iteration 10 ½

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



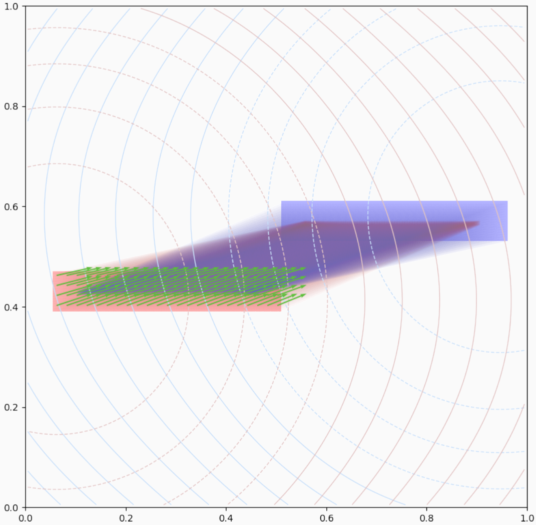
Iteration 20

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



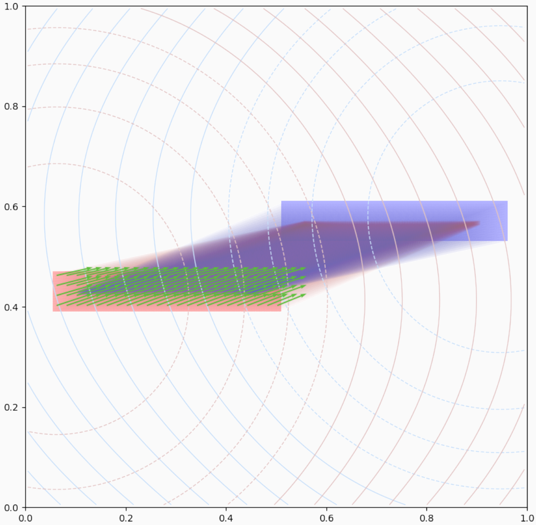
Iteration 20 ½

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



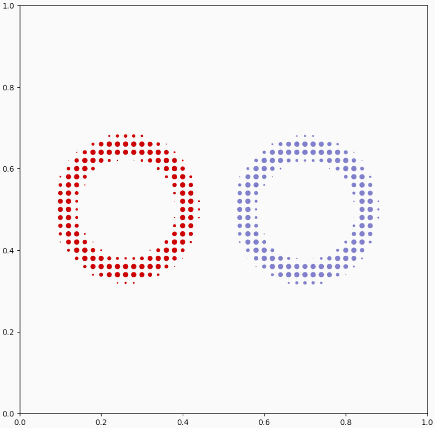
Iteration 30

The Sinkhorn algorithm, in practice; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$

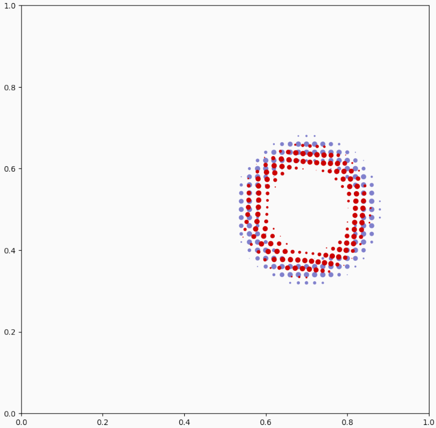


Iteration 30 ½

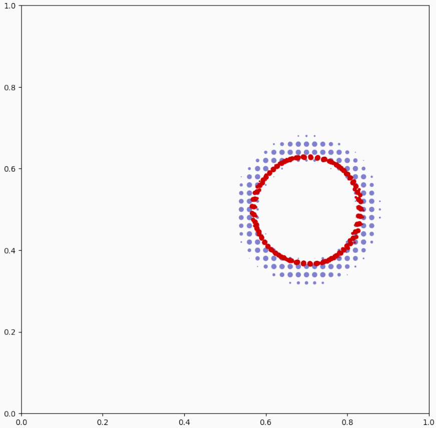
Registrating circles; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = 0.1$



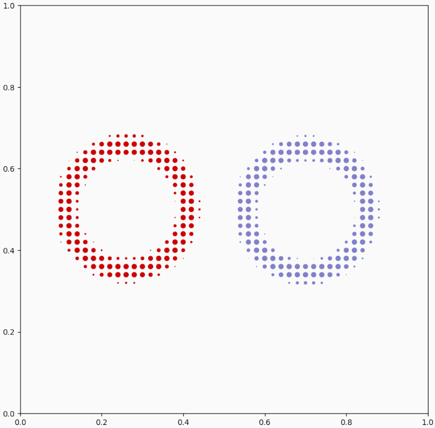
Registrating circles; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = 0.1$



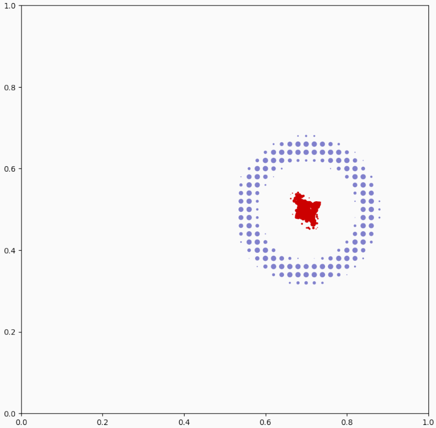
Registrating circles; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.1$



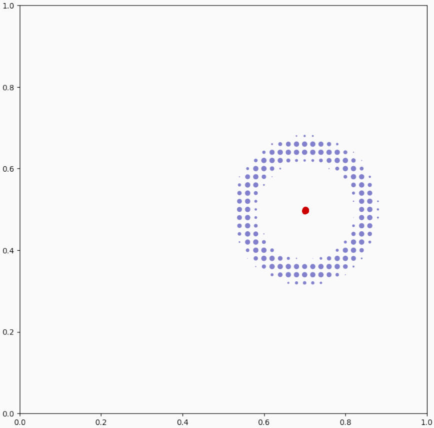
Registrating circles; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.2$



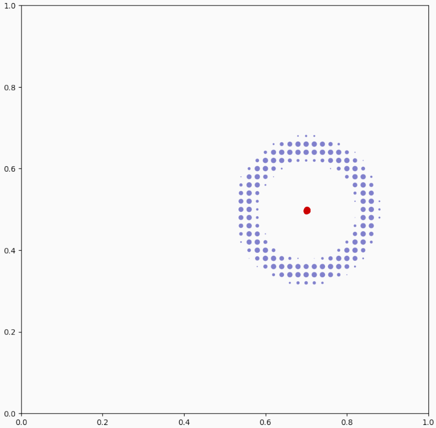
Registrating circles; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = 0.2$



Registrating circles; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = 0.2$



Registrating circles; $C(x, y) = \|x - y\|^2$, $\sqrt{\varepsilon} = 0.2$



Bad news: for $0 < \varepsilon < +\infty$, we converge towards α such that

$$\mathrm{W}_\varepsilon(\alpha, \beta) \;<\; \mathrm{W}_\varepsilon(\beta, \beta).$$

In our paper: theoretical guarantees

Solution: Use an unbiased divergence [GPC18]

$$\mathrm{d}_{\varepsilon\text{-Sinkhorn}}(\alpha,\beta) \;=\; \mathcal{W}_\varepsilon(\alpha,\beta) - \tfrac{1}{2}\mathcal{W}_\varepsilon(\alpha,\alpha) - \tfrac{1}{2}\mathcal{W}_\varepsilon(\beta,\beta).$$

In our paper: theoretical guarantees

Solution: Use an unbiased divergence [GPC18]

$$\mathrm{d}_{\varepsilon\text{-Sinkhorn}}(\alpha, \beta) \;=\; \mathrm{W}_\varepsilon(\alpha, \beta) - \tfrac{1}{2}\mathrm{W}_\varepsilon(\alpha, \alpha) - \tfrac{1}{2}\mathrm{W}_\varepsilon(\beta, \beta).$$

Theorem (Positivity ; F., Vialard)

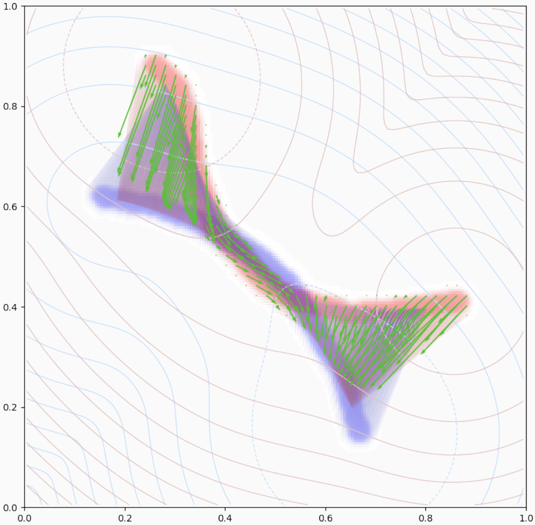
We define $\mathrm{d}_{\varepsilon\text{-Hausdorff}}(\alpha, \beta) \simeq \mathrm{d}_{\varepsilon\text{-SoftMin}}(\alpha, \beta)$. *Then, if*

$$k_\varepsilon(x, y) \;=\; \exp\big(-\tfrac{1}{\varepsilon}\mathrm{C}(x, y)\big)$$

defines a positive kernel, we have

$$0 \;\leqslant\; \mathrm{d}_{\varepsilon\text{-Hausdorff}}(\alpha, \beta) \;\leqslant\; \mathrm{d}_{\varepsilon\text{-Sinkhorn}}(\alpha, \beta).$$

The ε-Sinkhorn divergence; $C(x,y) = \|x-y\|^2$, $\sqrt{\varepsilon} = .1$



A high-quality gradient.

How do I implement this on real shapes?

$$\begin{array}{ccc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} & \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \left(\begin{array}{cccc}
\|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\
\|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\
\vdots & \vdots & \ddots & \vdots \\
\|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\|
\end{array}\right) & \begin{array}{cl}
\rightarrow & \text{Smin}_{\varepsilon, y \sim \beta} \|x_1 - y\| \\
\rightarrow & \text{Smin}_{\varepsilon, y \sim \beta} \|x_2 - y\| \\
\vdots & \\
\rightarrow & \text{Smin}_{\varepsilon, y \sim \beta} \|x_N - y\|
\end{array}
\end{array}$$

How do I implement this on real shapes?

$$\begin{array}{cc}
 & \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_M \end{array} \\
\begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \begin{pmatrix} \|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\ \|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\ \vdots & \vdots & \ddots & \vdots \\ \|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\| \end{pmatrix} \begin{array}{c} \rightarrow \quad \text{Smin}_{\varepsilon, y\sim\beta} \|x_1 - y\| \\ \rightarrow \quad \text{Smin}_{\varepsilon, y\sim\beta} \|x_2 - y\| \\ \vdots \\ \rightarrow \quad \text{Smin}_{\varepsilon, y\sim\beta} \|x_N - y\| \end{array}
\end{array}$$

Huge 100,000-by-100,000 matrices just don't fit into GPU memories.

How do I implement this on real shapes?

$$\begin{array}{cc} & \begin{array}{cccc} \quad\beta_1\quad & \quad\beta_2\quad & \cdots & \quad\beta_M\quad \end{array} \\ \begin{array}{c} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{array} & \begin{pmatrix} \|x_1 - y_1\| & \|x_1 - y_2\| & \cdots & \|x_1 - y_M\| \\ \|x_2 - y_1\| & \|x_2 - y_2\| & \cdots & \|x_2 - y_M\| \\ \vdots & \vdots & \ddots & \vdots \\ \|x_N - y_1\| & \|x_N - y_2\| & \cdots & \|x_N - y_M\| \end{pmatrix} \end{array} \begin{array}{c} \\ \to \quad \text{Smin}_{\varepsilon, y\sim\beta} \|x_1 - y\| \\ \to \quad \text{Smin}_{\varepsilon, y\sim\beta} \|x_2 - y\| \\ \vdots \\ \to \quad \text{Smin}_{\varepsilon, y\sim\beta} \|x_N - y\| \end{array}$$

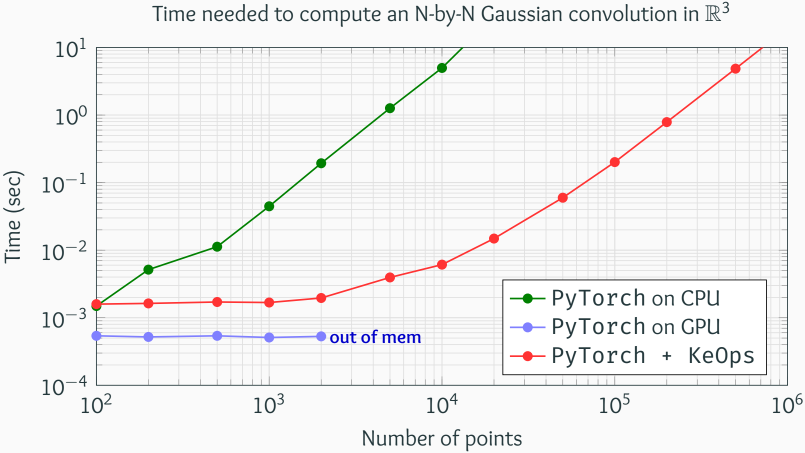
Huge 100,000-by-100,000 matrices just don’t fit into GPU memories.

We need online map-reduce routines.

KErnel OPerationS, with autodiff, without memory overflows

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

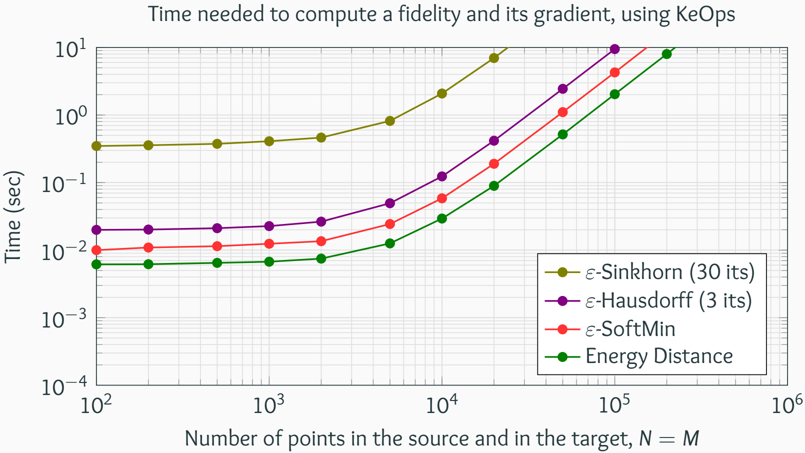
(Thanks Benjamin and Joan!)



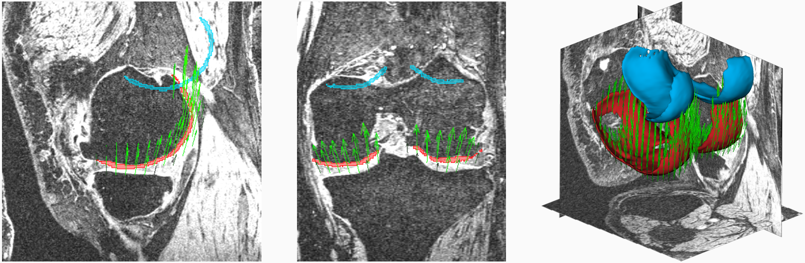
KErnel OPerationS, with autodiff, without memory overflows

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

(Thanks Benjamin and Joan!)



On real data, from the OsteoArthritris Initiative



Gradient of the Energy Distance, computed in 0.5s on my laptop.
(52,319 and 34,966 voxels – out of a 192-192-160 volume)

Conclusion

- Try using $k(x, y) \ = \ -\|x - y\|$!

Conclusion

- Try using $k(x, y) \; = \; -\|x - y\|$!

Our contributions:

- Statistics $\longleftrightarrow$ Computer Graphics $\longleftrightarrow$ Optimal Transport

Conclusion

- Try using $k(x,y) = -\|x-y\|$!

Our contributions:

- Statistics $\longleftrightarrow$ Computer Graphics $\longleftrightarrow$ Optimal Transport
- New ε-**Hausdorff** fidelity, metric entropy.

Conclusion

- Try using $k(x,y) \;=\; -\|x-y\|$!

Our contributions:

- Statistics $\longleftrightarrow$ Computer Graphics $\longleftrightarrow$ Optimal Transport
- New ε-**Hausdorff** fidelity, metric entropy.
- First proof that the ε-Sinkhorn divergence (from ML) is **positive**.

Conclusion

- Try using $k(x,y) \;=\; -\|x-y\|$!

Our contributions:

- Statistics $\longleftrightarrow$ Computer Graphics $\longleftrightarrow$ Optimal Transport
- New ε-**Hausdorff** fidelity, metric entropy.
- First proof that the ε-Sinkhorn divergence (from ML) is **positive**.
- **KeOps**: efficient online map-reduce routines

CUDA + Matlab, numpy, PyTorch

References

Our code is available:

`plmlab.math.cnrs.fr/jeanfeydy/shapes_toolbox`

References

Our code is available:

`plmlab.math.cnrs.fr/jeanfeydy/shapes_toolbox`

Our papers:

- *Global divergences between measures: from Hausdorff distance to Optimal Transport*, F., Trouvé, 2018 (uploaded yesterday on HAL)

References

Our code is available:

`plmlab.math.cnrs.fr/jeanfeydy/shapes_toolbox`

Our papers:

- *Global divergences between measures: from Hausdorff distance to Optimal Transport*, F., Trouvé, 2018 (uploaded yesterday on HAL)
- *Sinkhorn entropies and divergences*, F., Séjourné, Vialard, Amari, Trouvé, Peyré, 2018 (soon)

References

Our code is available:

`plmlab.math.cnrs.fr/jeanfeydy/shapes_toolbox`

Our papers:

- *Global divergences between measures: from Hausdorff distance to Optimal Transport*, F., Trouvé, 2018 (uploaded yesterday on HAL)
- *Sinkhorn entropies and divergences*,
  F., Séjourné, Vialard, Amari, Trouvé, Peyré, 2018 (soon)
- *Optimal Transport for diffeomorphic registration*,
  F., Charlier, Vialard, Peyré, 2017

Thank you for your attention.

Any questions ?

References i

- Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard.
  Unbalanced optimal transport: Dynamic and kantorovich formulations.
  Journal of Functional Analysis, 274(11):3090–3123, 2018.
- Marco Cuturi.
  Sinkhorn distances: Lightspeed computation of optimal transport.
  In *Advances in neural information processing systems*, pages 2292–2300, 2013.

References ii

Aude Genevay, Gabriel Peyre, and Marco Cuturi.
Learning generative models with sinkhorn divergences.
In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617. PMLR, 09–11 Apr 2018.

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

References iii

Gabriel Peyré and Marco Cuturi.
Computational optimal transport.
arXiv preprint arXiv:1803.00567, 2018.