Optimal Transport and Theano for diffeomorphic registration

A presentation to the Asclepios Inria team.

Jean Feydy

June 27, 2017

Écoles Normales Supérieures de Paris et Paris-Saclay

Some information

Jean Feydy (sept. 2016 - aug. 2019) :

- PhD student under the supervision of Alain Trouvé.
- Caïman at the ENS.

Some information

Jean Feydy (sept. 2016 - aug. 2019) :

- PhD student under the supervision of Alain Trouvé.
- Caïman at the ENS.

Two main points today :

- Optimal Transport as a data attachment term.
- `theano` as a development tool.

Supplementary material

Further references available online :

`www.math.ens.fr/~feydy/`

Research and Teaching tabs, look for :

- *Optimal Transport for Diffeomorphic Matching*,
  MICCAI 2017, J. Feydy, B. Charlier, F.-X. Vialard and G. Peyré.
- *Culture Mathématique*, chap. 9-10.
- *Introduction à la Géométrie Riemannienne par l'Étude des Espaces de Formes.*

Table of contents

Procustes Analysis

Position, Scale and Orientation

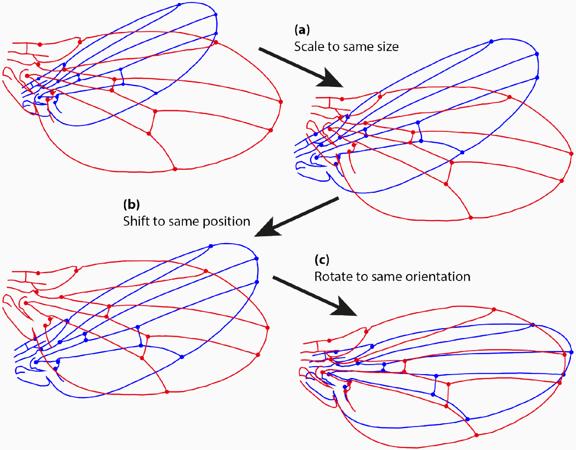


Figure 1: Matching the blue wing on the red one. (Wikipedia)

From images to labeled point clouds

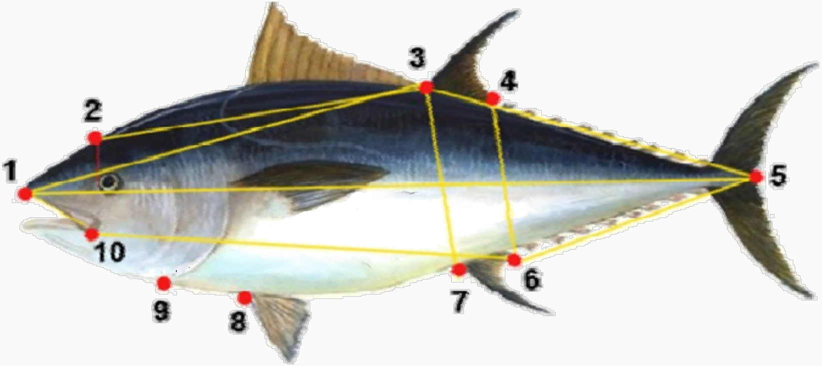


Figure 2: Anatomical landmarks on a tuna fish.
From *A morphometric approach for the analysis of body shape in bluefin tuna: preliminary results*, Addis and al.

Mathematical formulation

Let $X, Y \in \mathbb{R}^{M \times D}$ be two labeled point clouds.
Let $S_{\tau,\upsilon}$ denote the **rigid**-body transformation of parameters τ (translation) and υ (rotation + scaling).
Then, try to find

$$\begin{aligned} \tau_0, \upsilon_0 &= \arg\min_{\tau,\upsilon} \quad \| S_{\tau,\upsilon}(X) - Y \|_2^2 && (1) \\ &= \arg\min_{\tau,\upsilon} \sum_{m=1}^{M} | \upsilon \cdot x^m + \tau - y^m |^2 \,. && (2) \end{aligned}$$

Typical run on polygons

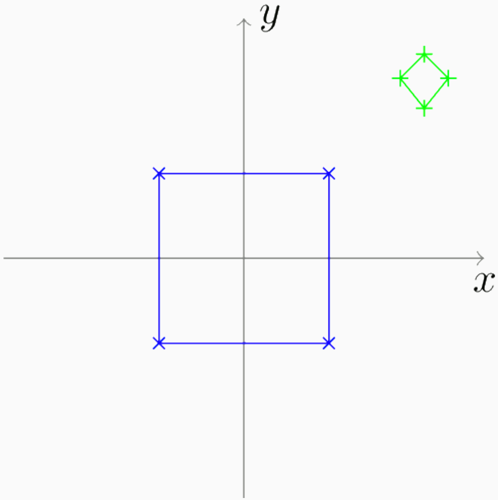


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

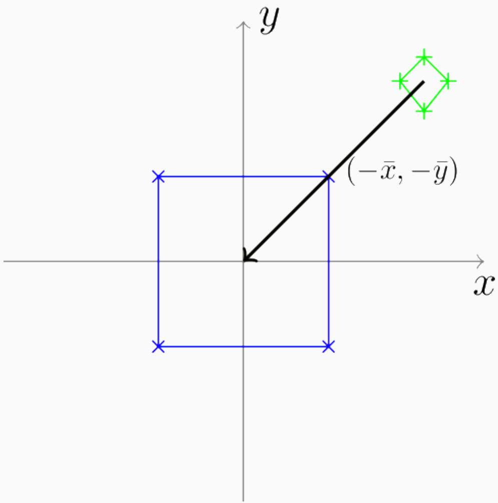


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

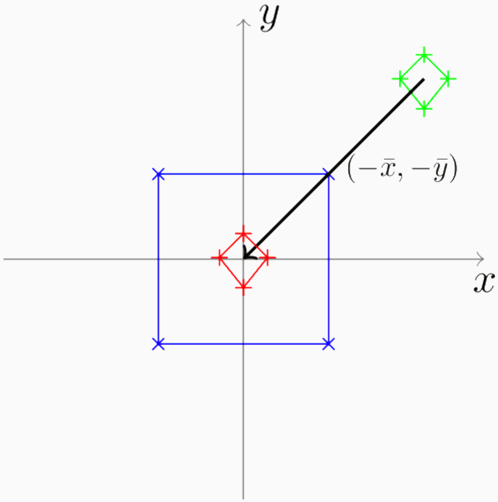


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

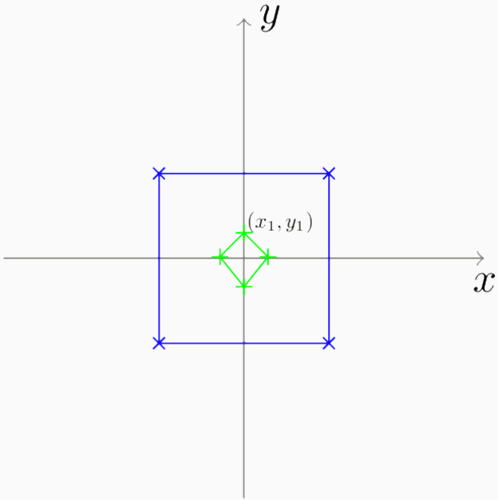


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

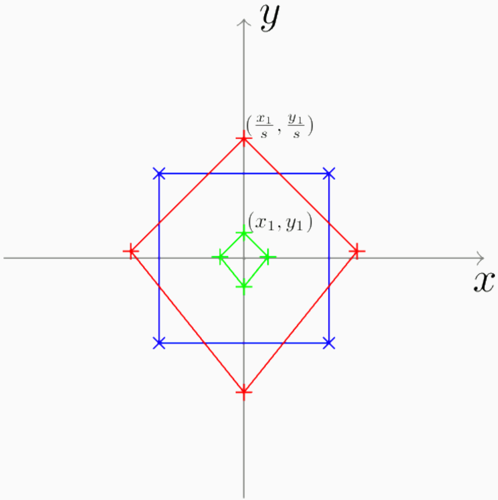


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

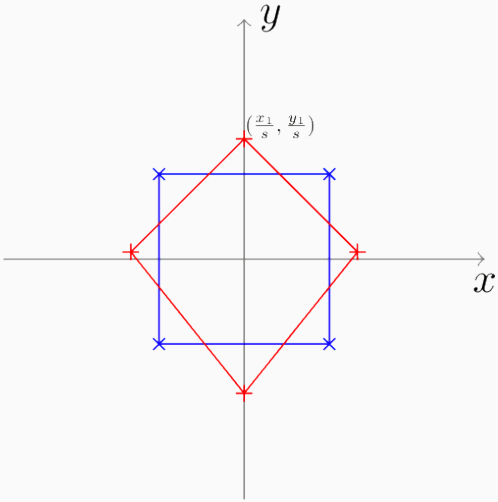


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

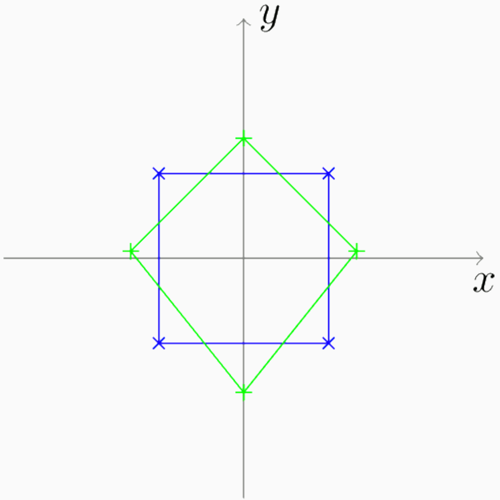


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

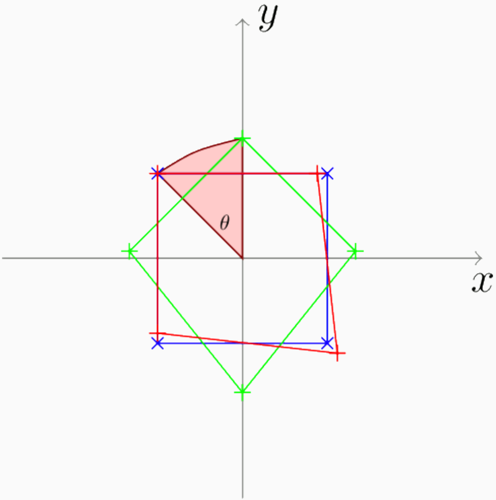


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

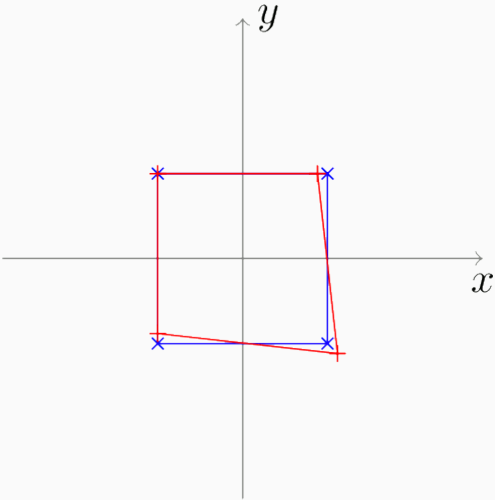


Figure 3: Matching a kitesurf on a square. (Wikipedia, Linschn)

Pros and cons of Procustes analysis

Pros :

- Simple and robust
- Parameters make sense
- Miracle results for populations of *triangles* (Kendall, 1984)

Pros and cons of Procustes analysis

Pros :

- Simple and robust
- Parameters make sense
- Miracle results for populations of *triangles* (Kendall, 1984)

Cons :

- Max. number of $2 \cdot D$ explicative parameters
- Unable to capture subtle shape deformations

Pros and cons of Procustes analysis

Pros :

- Simple and robust
- Parameters make sense
- Miracle results for populations of *triangles* (Kendall, 1984)

Cons :

- Max. number of $2 \cdot D$ explicative parameters
- Unable to capture subtle shape deformations

This model is a standard pre-processing tool.
However, it is too limited to allow in-detail analysis.

Optimal Transport

Image matching as a mass-carrying problem

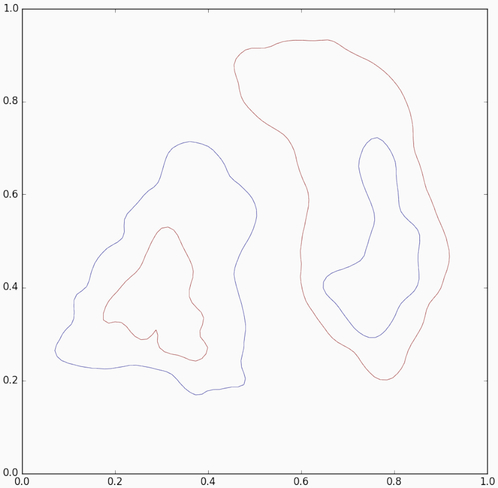


Figure 4: Optimal transport between two curves seen as mass distributions : from a déblai to a remblai.

Image matching as a mass-carrying problem

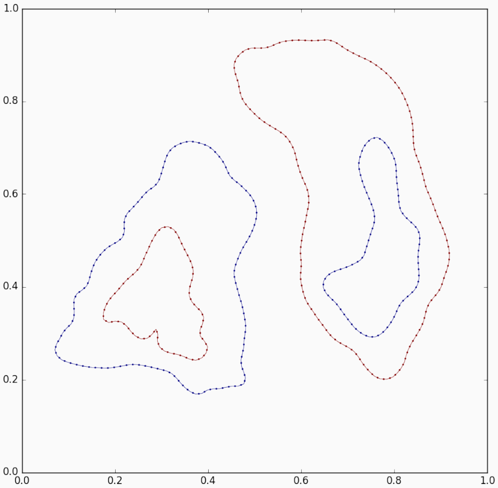


Figure 4: Optimal transport between two curves seen as mass distributions : from a déblai to a remblai.

Image matching as a mass-carrying problem

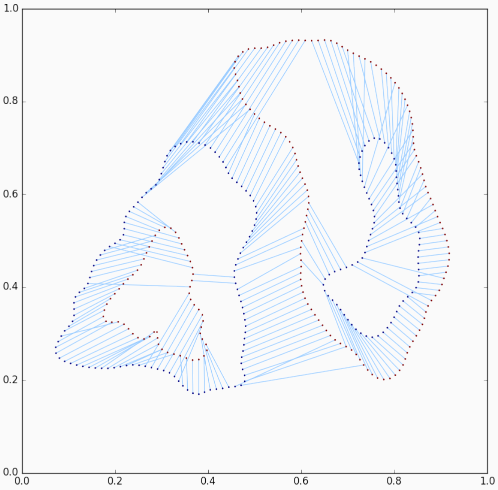


Figure 4: Optimal transport between two curves seen as mass distributions : from a déblai to a remblai.

Dynamic formulation

Let : $(x^1, \dots, x^I)$ and $(y^1, \dots, y^J)$ be two point clouds and $(\mu_1, \dots, \mu_I)$, $(\nu_1, \dots, \nu_J)$ the associated (integer) weights, such that $\sum \mu_i = M = \sum \nu_j$.

Dynamic formulation

Let : $(x^1, \dots, x^I)$ and $(y^1, \dots, y^J)$ be two point clouds and $(\mu_1, \dots, \mu_I)$, $(\nu_1, \dots, \nu_J)$ the associated (integer) weights, such that $\sum \mu_i = M = \sum \nu_j$.

Then, find a collection of paths $\gamma^m : t \in [0, 1] \mapsto \gamma_t^m$ minimizing

$$\ell^2(\gamma) \;=\; \sum_{m=1}^{M} \int_{t=0}^{1} \|\dot{\gamma}_t^m\|^2 \,\mathrm{d}t, \tag{3}$$

under the constraint that for all indices i and j,

$$\#\Big\{ m \in [\![1, M]\!]\;,\; \gamma_0^m = x^i \Big\} \;=\; \mu_i, \tag{4}$$

$$\#\Big\{ m \in [\![1, M]\!]\;,\; \gamma_1^m = y^j \Big\} \;=\; \nu_j. \tag{5}$$

Dynamic formulation

Let : $(x^1, \dots, x^I)$ and $(y^1, \dots, y^J)$ be two point clouds and $(\mu_1, \dots, \mu_I)$, $(\nu_1, \dots, \nu_J)$ the associated (integer) weights, such that $\sum \mu_i = M = \sum \nu_j$.

Then, find a collection of paths $\gamma^m : t \in [0,1] \mapsto \gamma^m_t$ minimizing

$$\ell^2(\gamma) \;=\; \sum_{m=1}^{M} \int_{t=0}^{1} \|\dot{\gamma}^m_t\|^2 \,\mathrm{d}t, \tag{3}$$

under the constraint that for all indices i and j,

$$\#\Big\{ m \in [\![1, M]\!] \;,\; \gamma^m_0 = x^i \Big\} \;=\; \mu_i, \tag{4}$$

$$\#\Big\{ m \in [\![1, M]\!] \;,\; \gamma^m_1 = y^j \Big\} \;=\; \nu_j. \tag{5}$$

γ is the optimal transport path between the two measures

$$\sum_{i=1}^{I} \mu_i \delta_{x^i} \;=\; \mu \;\xrightarrow{\;\gamma\;}\; \nu \;=\; \sum_{j=1}^{J} \nu_j \delta_{y^j}. \tag{6}$$

Static formulation : permutation

If we relabel the unit masses $(x^1, \ldots, x^M)$ and $(y^1, \ldots, y^M)$, find a **permutation** $\sigma : [\![1, M]\!] \to [\![1, M]\!]$ minimizing

$$C^{X,Y}(\sigma) \; = \; \sum_{m=1}^{M} \left\| x^m - y^{\sigma(m)} \right\|^2 . \tag{7}$$

σ is an optimal labeling.

Static formulation : transport plan

Independent particles should always go in **straight lines** :
If we denote $c_{i,j} = \left\| x^i - y^j \right\|^2$, find an optimal transport plan $\Gamma = (\gamma_{i,j})_{(i,j) \in [\![1,I]\!] \times [\![1,J]\!]}$ minimizing

$$C^{X,Y}(\Gamma) \;=\; \sum_{i,j} \gamma_{i,j}\, c_{i,j} \tag{8}$$

under the constraints :

$$\forall\, i,j,\;\; \gamma_{i,j} \geqslant 0, \qquad \forall\, i,\;\; \sum_{j} \gamma_{i,j} = \mu_i, \qquad \forall\, j,\;\; \sum_{i} \gamma_{i,j} = \nu_j. \tag{9}$$

Static formulation : transport plan

Independent particles should always go in **straight lines** :
If we denote $c_{i,j} = \left\|x^i - y^j\right\|^2$, find an optimal transport plan $\Gamma = (\gamma_{i,j})_{(i,j)\in[\![1,I]\!]\times[\![1,J]\!]}$ minimizing

$$C^{X,Y}(\Gamma) \;=\; \sum_{i,j} \gamma_{i,j}\, c_{i,j} \tag{8}$$

under the constraints :

$$\forall\; i,j,\;\; \gamma_{i,j} \geqslant 0, \qquad \forall\; i,\;\; \sum_j \gamma_{i,j} = \mu_i, \qquad \forall\; j,\;\; \sum_i \gamma_{i,j} = \nu_j. \tag{9}$$

This is textbook linear programming.

Entropic regularization

Under marginal constraints $\Gamma \mathbf{1} = \mu$, $\mathbf{1}^\top \Gamma = \nu^\top$, minimize

$$C_\varepsilon^{X,Y}(\Gamma) = \sum_{i,j} \gamma_{i,j}\, c_{i,j} - \varepsilon \cdot H(\Gamma) \tag{10}$$

with entropy $H(\Gamma) = -\sum_{i,j} \gamma_{i,j}(\log(\gamma_{i,j}) - 1)$.

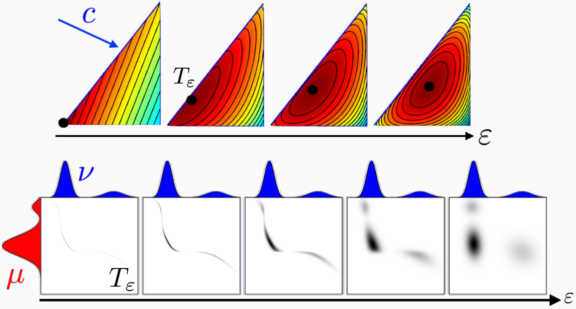


Figure 5: Image borrowed to Gabriel Peyré.

The regularized transport problem

Schrödinger problem :

How much do ε-Brownian bridges get mixed together ?

Equations satisfied by the optimal transport plan

Entropic transport is a scaling problem

The optimal transport plan can be written

$$\Gamma \;=\; \mathrm{diag}(a)\cdot K\cdot \mathrm{diag}(b) \;=\; (a_i b_j k_{i,j}), \tag{11}$$

with

$$k_{i,j} \;=\; e^{-c_{i,j}/\varepsilon}, \qquad\qquad a \geqslant 0, \qquad\qquad b \geqslant 0. \tag{12}$$

Equations satisfied by the optimal transport plan

Entropic transport is a scaling problem

The optimal transport plan can be written

$$\Gamma = \mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b) = (a_i b_j k_{i,j}), \tag{11}$$

with

$$k_{i,j} = e^{-c_{i,j}/\varepsilon}, \qquad a \geqslant 0, \qquad b \geqslant 0. \tag{12}$$

Sinkhorn theorem $\implies$ this scaling problem is tractable.

The Sinkhorn algorithm

We want :

$$\mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b) \cdot \mathbf{1} = \mu \qquad \text{and} \quad \nu^\intercal = \mathbf{1}^\intercal \cdot \mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b),$$

We want :

$$\mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b) \cdot \mathbf{1} = \mu \quad \text{and} \quad \nu^\mathsf{T} = \mathbf{1}^\mathsf{T} \cdot \mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b),$$

$$\text{i.e.} \quad \mathrm{diag}(a) \cdot Kb = \mu \quad \text{and} \quad \nu = \mathrm{diag}(b) \cdot K^\mathsf{T} a,$$

The Sinkhorn algorithm

We want :

$$\begin{aligned} \mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b) \cdot \mathbf{1} = \mu \quad &\text{and} \quad \nu^\intercal = \mathbf{1}^\intercal \cdot \mathrm{diag}(a) \cdot K \cdot \mathrm{diag}(b), \\ \text{i.e.} \quad \mathrm{diag}(a) \cdot Kb = \mu \quad &\text{and} \quad \nu = \mathrm{diag}(b) \cdot K^\intercal a, \\ \text{i.e.} \quad Kb = \frac{\mu}{a} \quad &\text{and} \quad \frac{\nu}{b} = K^\intercal a, \end{aligned}$$

The Sinkhorn algorithm

We want :

$$
\begin{aligned}
\operatorname{diag}(a) \cdot K \cdot \operatorname{diag}(b) \cdot \mathbf{1} = \mu \quad &\text{and} \quad \nu^\intercal = \mathbf{1}^\intercal \cdot \operatorname{diag}(a) \cdot K \cdot \operatorname{diag}(b),\\
\text{i.e.} \qquad \operatorname{diag}(a) \cdot Kb = \mu \quad &\text{and} \quad \nu = \operatorname{diag}(b) \cdot K^\intercal a,\\
\text{i.e.} \qquad Kb = \frac{\mu}{a} \quad &\text{and} \quad \frac{\nu}{b} = K^\intercal a,\\
\text{i.e.} \qquad a = \frac{\mu}{Kb} \quad &\text{and} \quad b = \frac{\nu}{K^\intercal a}.
\end{aligned}
$$

We want :

$$\begin{aligned}
\operatorname{diag}(a)\cdot K\cdot\operatorname{diag}(b)\cdot\mathbf{1}=\mu \quad &\text{and}\quad \nu^{\mathsf{T}}=\mathbf{1}^{\mathsf{T}}\cdot\operatorname{diag}(a)\cdot K\cdot\operatorname{diag}(b),\\
\text{i.e.}\qquad \operatorname{diag}(a)\cdot Kb=\mu \quad &\text{and}\quad \nu=\operatorname{diag}(b)\cdot K^{\mathsf{T}}a,\\
\text{i.e.}\qquad Kb=\frac{\mu}{a} \quad &\text{and}\quad \frac{\nu}{b}=K^{\mathsf{T}}a,\\
\text{i.e.}\qquad a=\frac{\mu}{Kb} \quad &\text{and}\quad b=\frac{\nu}{K^{\mathsf{T}}a}.
\end{aligned}$$

The Sinkhorn algorithm

We want :

$$\begin{aligned}
\mathrm{diag}(a)\cdot K\cdot \mathrm{diag}(b)\cdot \mathbf{1} = \mu \quad &\text{and}\quad \nu^\top = \mathbf{1}^\top\cdot \mathrm{diag}(a)\cdot K\cdot \mathrm{diag}(b),\\
\text{i.e.}\quad \mathrm{diag}(a)\cdot Kb = \mu \quad &\text{and}\quad \nu = \mathrm{diag}(b)\cdot K^\top a,\\
\text{i.e.}\quad Kb = \frac{\mu}{a} \quad &\text{and}\quad \frac{\nu}{b} = K^\top a,\\
\text{i.e.}\quad a = \frac{\mu}{Kb} \quad &\text{and}\quad b = \frac{\nu}{K^\top a}.
\end{aligned}$$

Sinkhorn algorithm :

1. start with $a = \mathbf{1}_I$, $b = \mathbf{1}_J$.

We want :

$$\begin{aligned}
\mathrm{diag}(a)\cdot K\cdot \mathrm{diag}(b)\cdot \mathbf{1} = \mu \quad &\text{and} \quad \nu^\top = \mathbf{1}^\top\cdot \mathrm{diag}(a)\cdot K\cdot \mathrm{diag}(b),\\
\text{i.e.} \quad \mathrm{diag}(a)\cdot Kb = \mu \quad &\text{and} \quad \nu = \mathrm{diag}(b)\cdot K^\top a,\\
\text{i.e.} \quad Kb = \frac{\mu}{a} \quad &\text{and} \quad \frac{\nu}{b} = K^\top a,\\
\text{i.e.} \quad a = \frac{\mu}{Kb} \quad &\text{and} \quad b = \frac{\nu}{K^\top a}.
\end{aligned}$$

Sinkhorn algorithm :

1. start with $a = \mathbf{1}_I$, $b = \mathbf{1}_J$.
2. Apply repeatedly

$$a \leftarrow \frac{\mu}{Kb}, \qquad b \leftarrow \frac{\nu}{K^\top a}. \tag{13}$$

Implementation details

We use

$$a \leftarrow \frac{\mu}{Kb}, \qquad\qquad b \leftarrow \frac{\nu}{K^\intercal a}. \tag{14}$$

- Very efficient scheme for squared distances on a grid.
- Otherwise, we work in the log-domain :

$$u = \varepsilon \log(a) \qquad \text{and} \qquad v = \varepsilon \log(b) \tag{15}$$

Implementation details

We use

$$a \leftarrow \frac{\mu}{Kb}, \qquad\qquad b \leftarrow \frac{\nu}{K^{\top}a}. \tag{14}$$

- Very efficient scheme for squared distances on a grid.
- Otherwise, we work in the log-domain :

$$u = \varepsilon \log(a) \qquad \text{and} \qquad v = \varepsilon \log(b) \tag{15}$$

so that the iterations read

$$u \leftarrow u + \varepsilon \log(\mu) - \varepsilon \log\left(\sum_j \exp\left(\frac{u_i + v_j - c_{i,j}}{\varepsilon}\right)\right) \tag{16}$$

$$v \leftarrow v + \varepsilon \log(\nu) - \varepsilon \log\left(\sum_i \exp\left(\frac{u_i + v_j - c_{i,j}}{\varepsilon}\right)\right). \tag{17}$$

The Sinkhorn algorithm : an efficient iterative solver

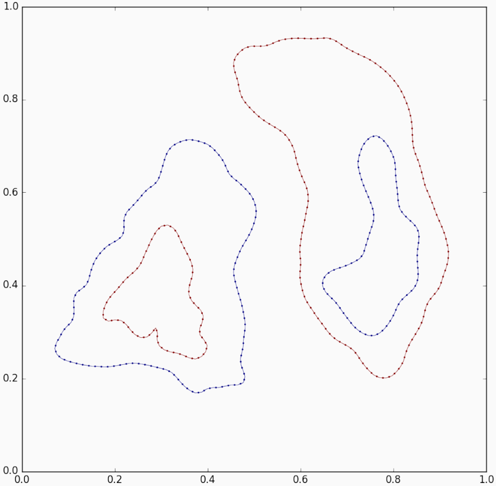


Figure 6: Measures to match.

The Sinkhorn algorithm : an efficient iterative solver

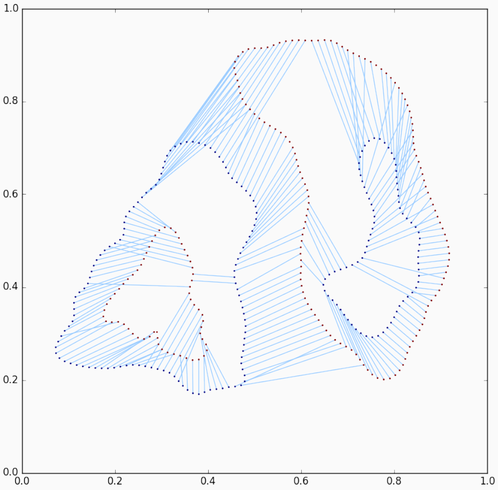


Figure 6: Monge transport, $\sqrt{\varepsilon} = 0$.

The Sinkhorn algorithm : an efficient iterative solver

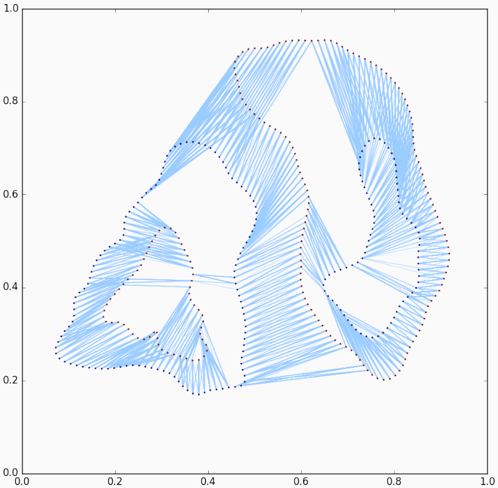


Figure 6: Diffuse transport, $\sqrt{\varepsilon} = .01$.

The Sinkhorn algorithm : an efficient iterative solver

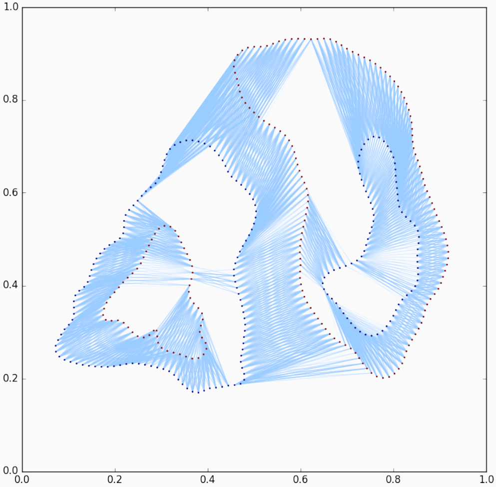


Figure 6: Diffuse transport, $\sqrt{\varepsilon} = .03$.

Pros and cons of Optimal Transport

Pros :

- Well-posed, convex problem
- Global and precise matchings
- Light-speed numerical solvers at hand (Cuturi, 2013)

Pros and cons of Optimal Transport

Pros :

- Well-posed, convex problem
- Global and precise matchings
- Light-speed numerical solvers at hand (Cuturi, 2013)

Cons :

- Discards topology : **tears** shapes apart

Pros and cons of Optimal Transport

Pros :

- Well-posed, convex problem
- Global and precise matchings
- Light-speed numerical solvers at hand (Cuturi, 2013)

Cons :

- Discards topology : **tears** shapes apart

This model is mathematically and numerically appealing. However, it does not provide any smoothness guarantee.

Can we build a rich and practical model for smooth deformations ?

The diffeomorphic framework

Spoiler alert : yes indeed, but it won't be *convex* anymore

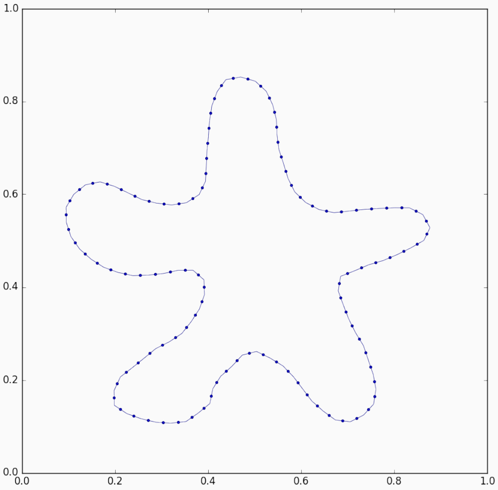


Figure 7: Source.

Spoiler alert : yes indeed, but it won't be *convex* anymore

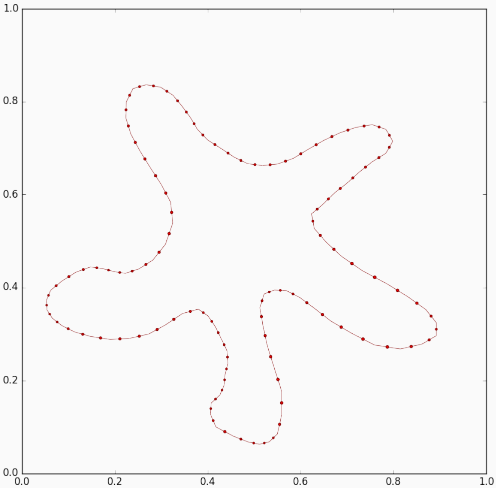


Figure 7: Target.

Spoiler alert : yes indeed, but it won't be *convex* anymore

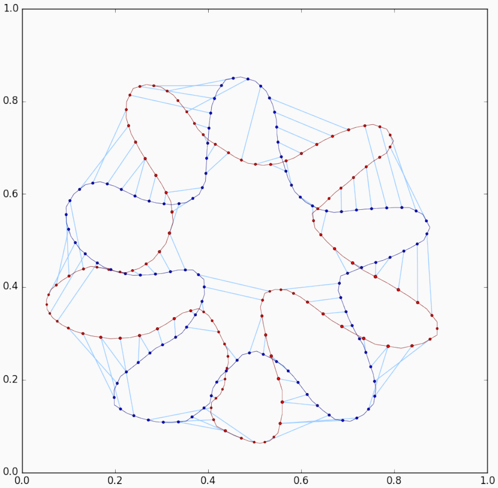


Figure 7: OT matching.

Spoiler alert : yes indeed, but it won't be *convex* anymore

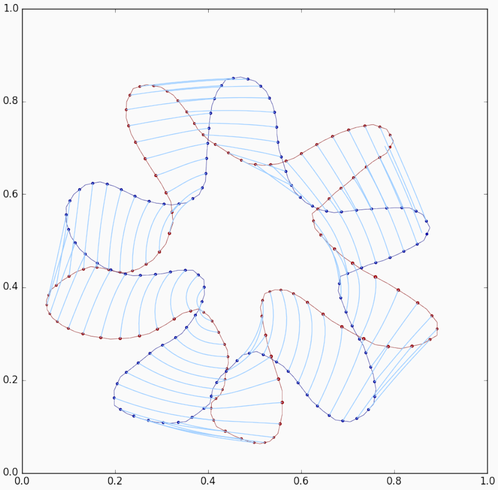
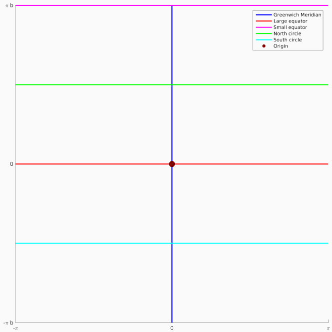


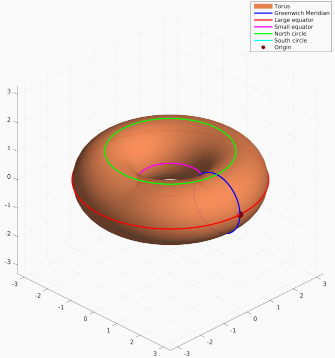
Figure 7: LDDMM matching.

The diffeomorphic framework

Shooting on spaces of diffeomorphisms



(a) As a deformed square.



(b) Embedded in $\mathbb{R}^3$.

Figure 8: The donut-shaped torus.

Natural curves on the space of diffeomorphisms

Problem : Match two shapes X and Y.

Simple solution : Try to find a sensible diffeomorphic trajectory φ_t such that

$$\varphi_0 = \mathrm{Id}_{\mathbb{R}^d} \qquad \text{and} \qquad \varphi_1 \cdot X \simeq Y. \tag{18}$$

Natural curves on the space of diffeomorphisms

Problem : Match two shapes X and Y.
Simple solution : Try to find a sensible diffeomorphic trajectory φ_t such that

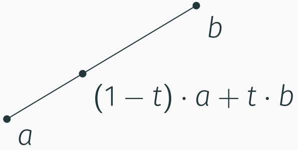
$$\varphi_0 = \mathrm{Id}_{\mathbb{R}^d} \qquad \text{and} \qquad \varphi_1 \cdot X \simeq Y. \tag{18}$$

$\dot{\varphi}_t = v_t$ is a vector field on the ambient space $\mathbb{R}^d$.
Two main models :

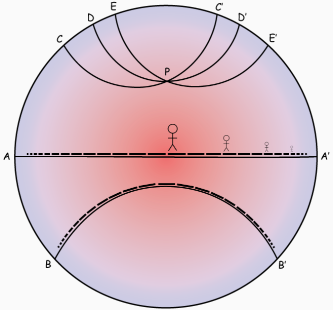
Log-demons φ_t is a one-parameter subgroup $\rightarrow v_t$ is constant.

LDDMM φ_t is a geodesic on the group of diffeomorphisms seen as a manifold endowed with a right-invariant metric given by a euclidean norm $\|v_t\|_k$
$\rightarrow (\varphi_t, v_t)$ obeys a geodesic equation.

Sometimes, we can compute geodesics explicitly...



(a) The Euclidean plane.



(b) The Poincaré disk.

Figure 9: Explicit geodesics on homogeneous manifolds.
(b) is adapted from `www.pitt.edu/~jdnorton/`.

But this is not the case in general

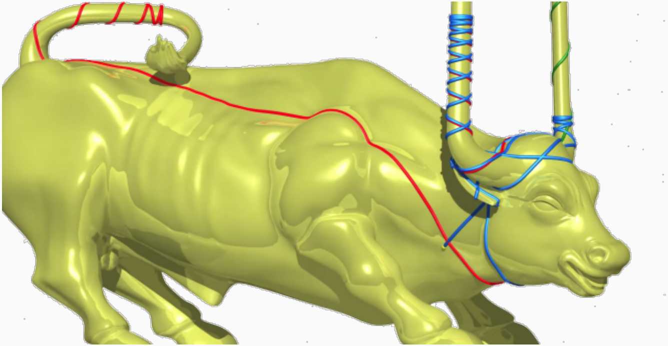


Figure 10: Geodesics on the Duhem's bull, embedded in $\mathbb{R}^3$. Taken from `www.chaos-math.org`.

In both models, we get an exponential map :

Log-demons Fast exponentiation of $(\mathrm{Id} + \frac{v}{256})^{256}$,

$$\mathrm{Exp} : v \in V \mapsto \varphi_1 \in \mathrm{Diff}(\mathbb{R}^d). \tag{19}$$

The exponential map

In both models, we get an exponential map :

Log-demons Fast exponentiation of $(\mathrm{Id} + \frac{v}{256})^{256}$,

$$\mathrm{Exp} : v \in V \mapsto \varphi_1 \in \mathrm{Diff}(\mathbb{R}^d). \tag{19}$$

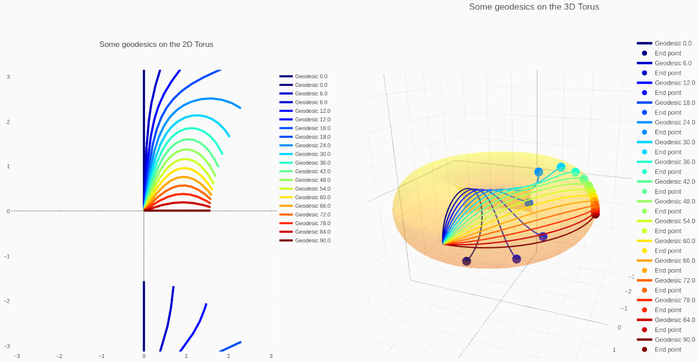
LDDMM Euler-like integration of the Hamiltonian geodesic equations :

$$\left\{ \begin{array}{rcl} q_{t+0.1} & = & q_t \;+\; 0.1 \cdot K_{q_t} p_t \\ p_{t+0.1} & = & p_t \;-\; 0.1 \cdot \partial_q (p_t, K_q p_t)(q_t) \end{array} \right. , \tag{20}$$

so that

$$\mathrm{Exp}_{q_0} : p_0 \in T^\star_{q_0}\mathcal{M} \;\mapsto\; q_1 \in \mathcal{M}. \tag{21}$$

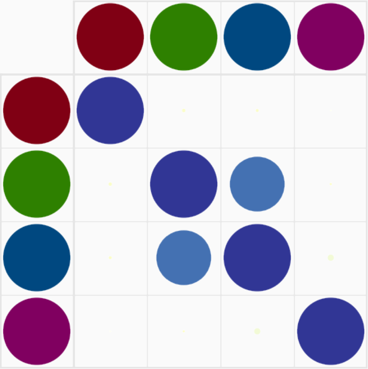
It works !



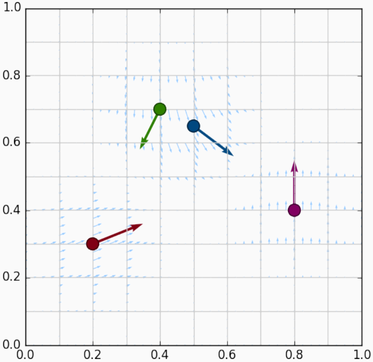
(a) 2D parametrization. (b) Embedded in $\mathbb{R}^3$.

Figure 11: Geodesics on the donut-shaped torus.

Influence of the kernel width, $\sigma = .35$



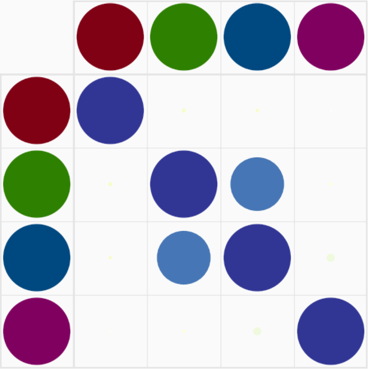
(a) Kernel matrix k_{q_t}.



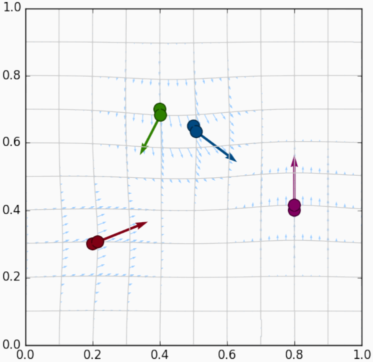
(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

Influence of the kernel width, $\sigma = .35$



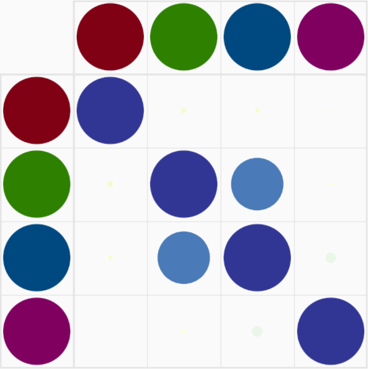
(a) Kernel matrix k_{q_t}.



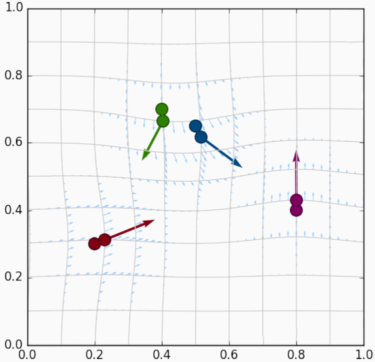
(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

Influence of the kernel width, $\sigma = .35$

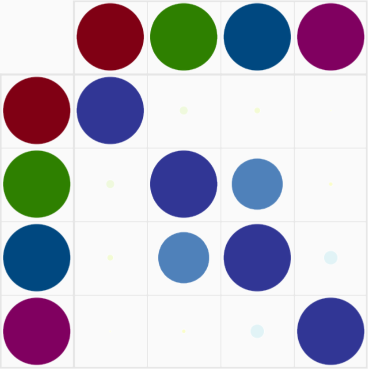


(a) Kernel matrix k_{q_t}.

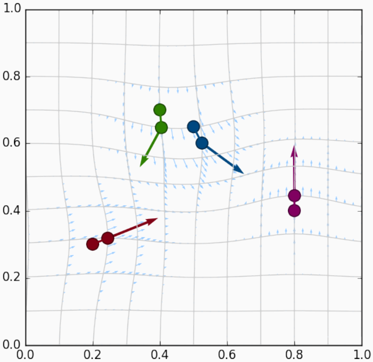


(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

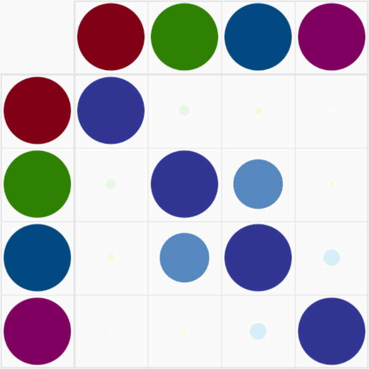


(a) Kernel matrix k_{q_t}.

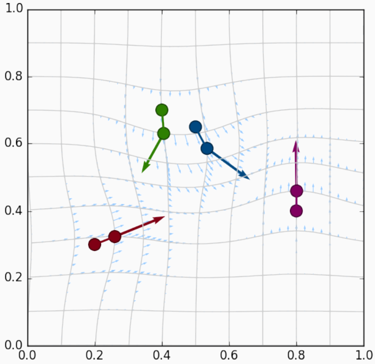


(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.



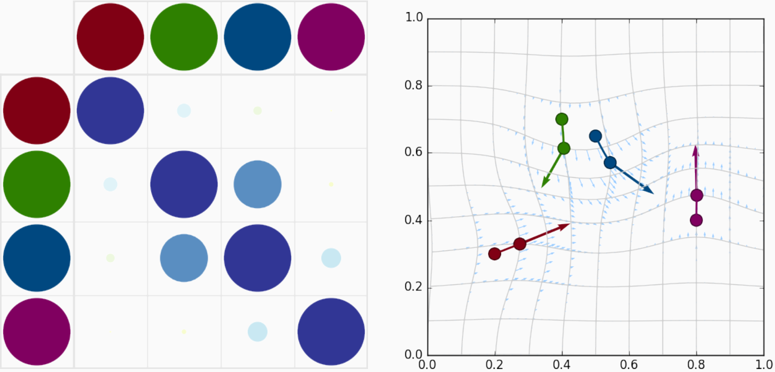
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

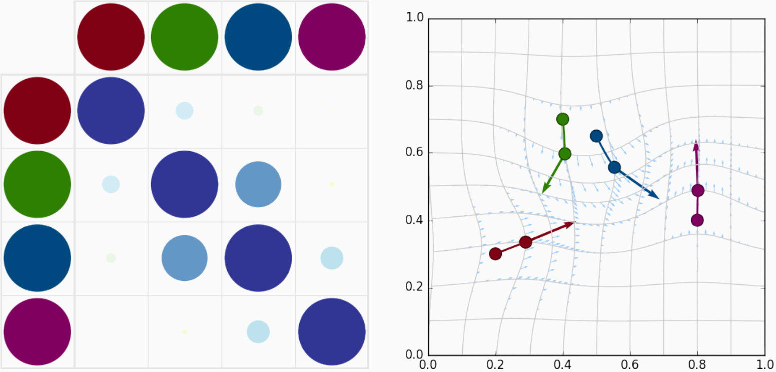
Influence of the kernel width, $\sigma = .35$



(a) Kernel matrix k_{q_t}.   (b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

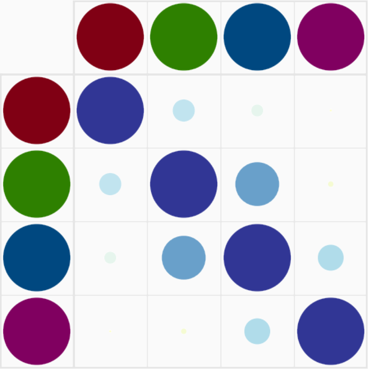
Influence of the kernel width, $\sigma = .35$



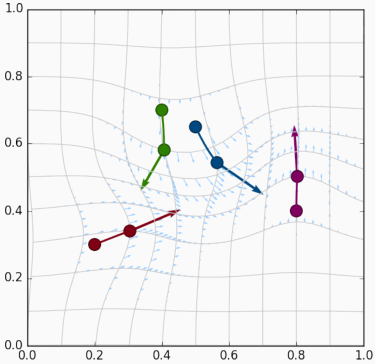
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

Influence of the kernel width, $\sigma = .35$



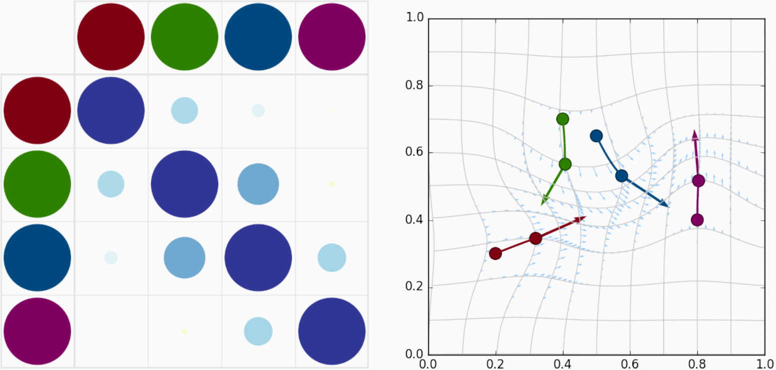
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

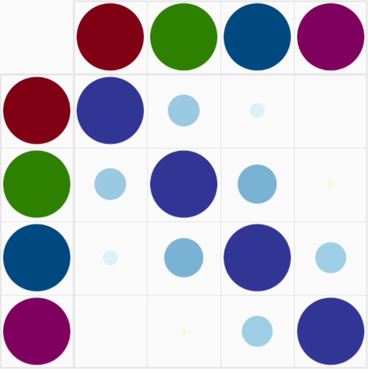
Influence of the kernel width, $\sigma = .35$



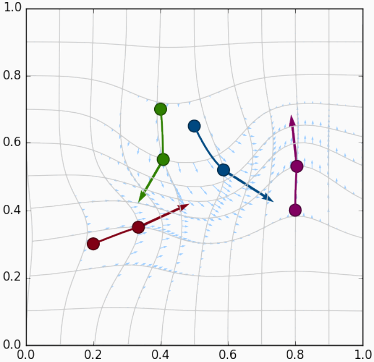
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

Influence of the kernel width, $\sigma = .35$



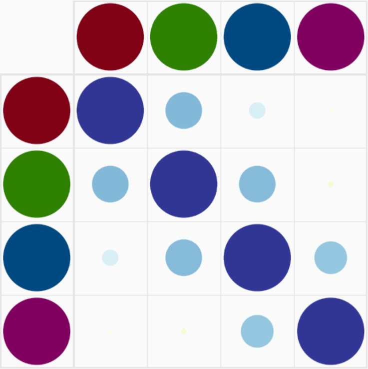
(a) Kernel matrix k_{q_t}.



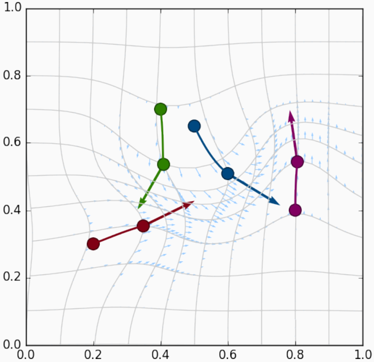
(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

Influence of the kernel width, $\sigma = .35$



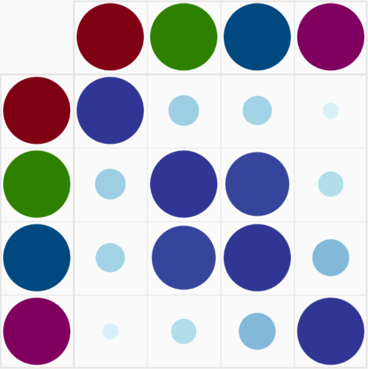
(a) Kernel matrix k_{q_t}.



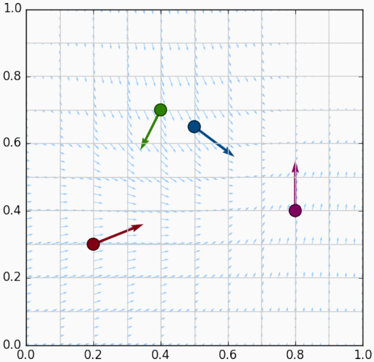
(b) Shooted cloud (q_t, p_t).

Figure 12: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

Influence of the kernel width, $\sigma = .50$



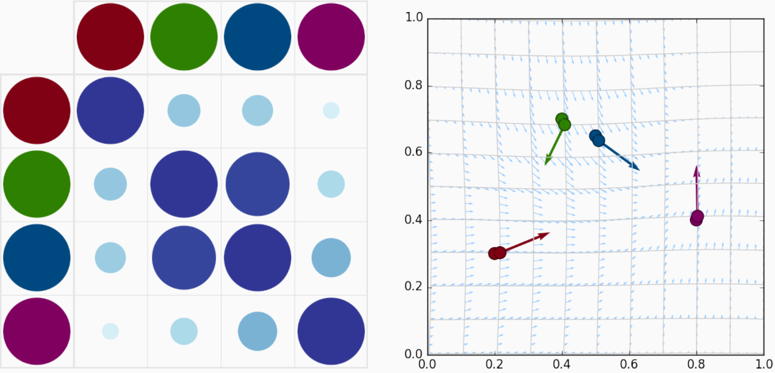
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

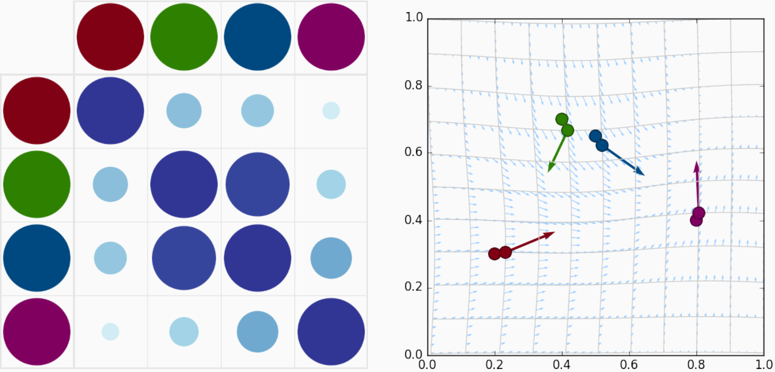
Influence of the kernel width, $\sigma = .50$



(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = .50$

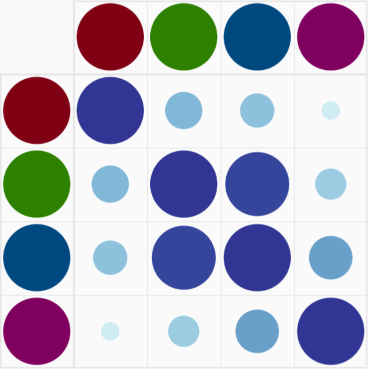


(a) Kernel matrix k_{q_t}.

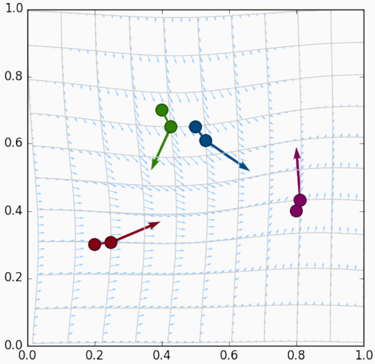
(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = .50$



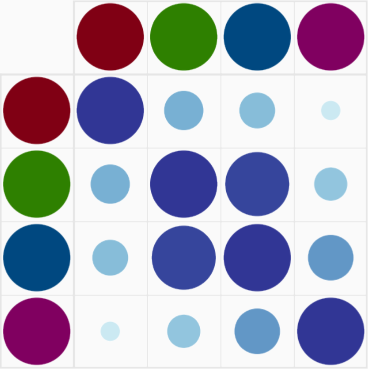
(a) Kernel matrix k_{q_t}.



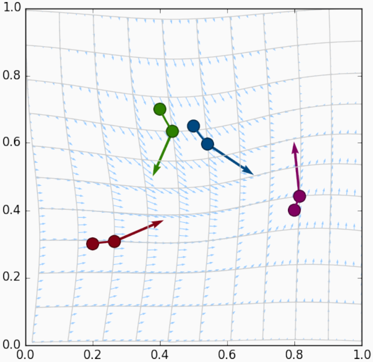
(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = \texttt{.50}$

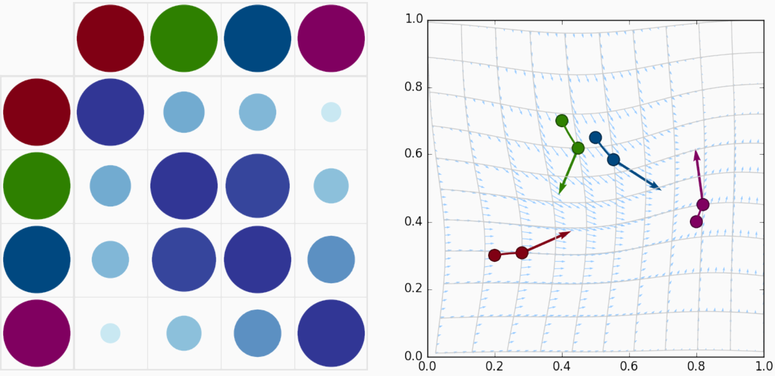


(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

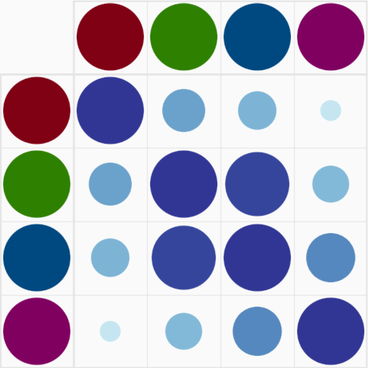
Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = \texttt{.50}$.



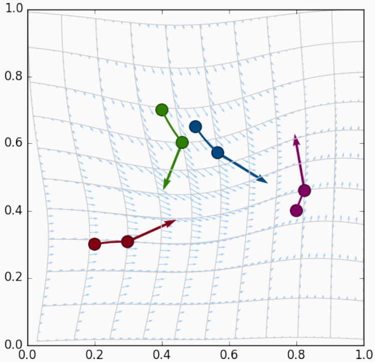
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = .50$



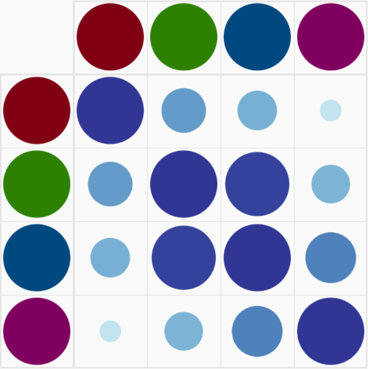
(a) Kernel matrix k_{q_t}.



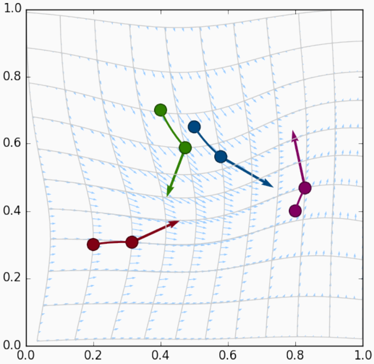
(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = .50$



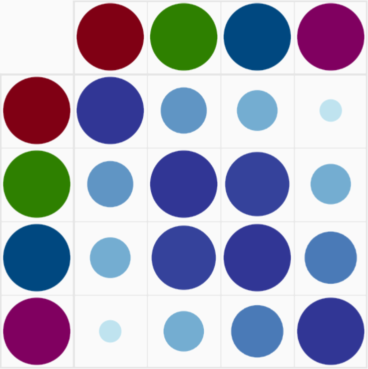
(a) Kernel matrix k_{q_t}.



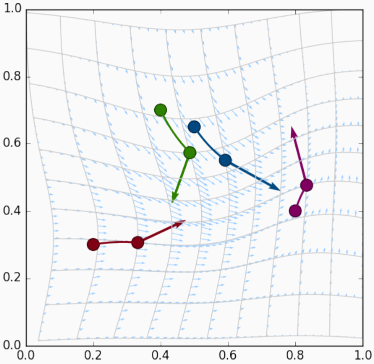
(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = .50$



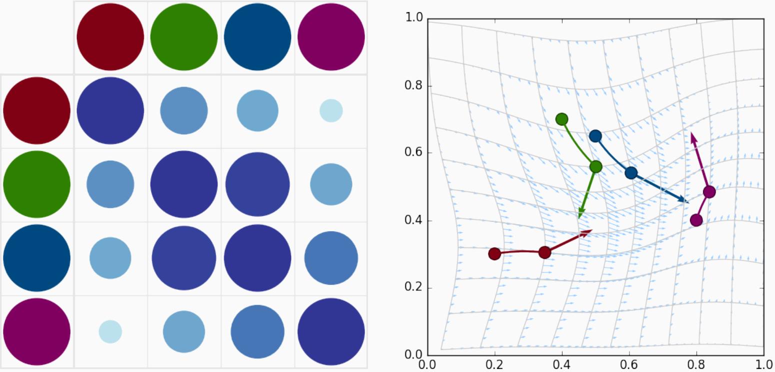
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

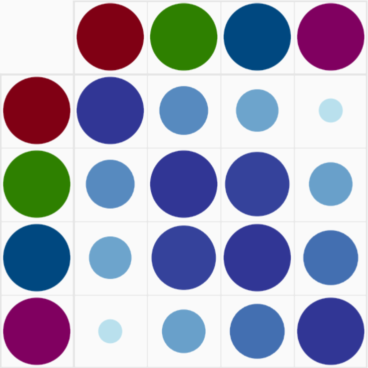
Influence of the kernel width, $\sigma = .50$



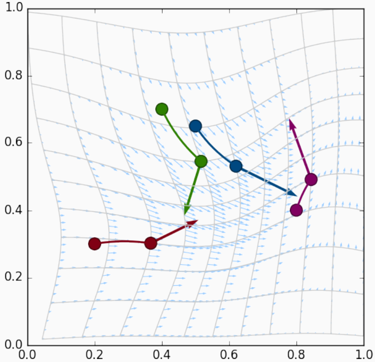
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = .50$



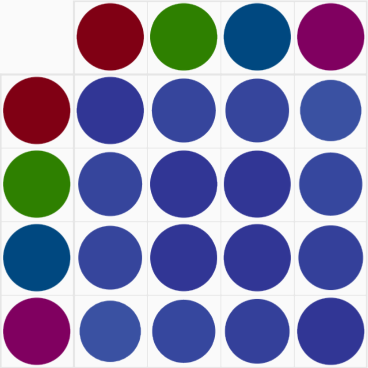
(a) Kernel matrix k_{q_t}.



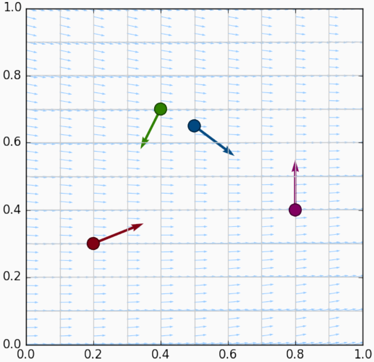
(b) Shooted cloud (q_t, p_t).

Figure 13: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

Influence of the kernel width, $\sigma = 1.$



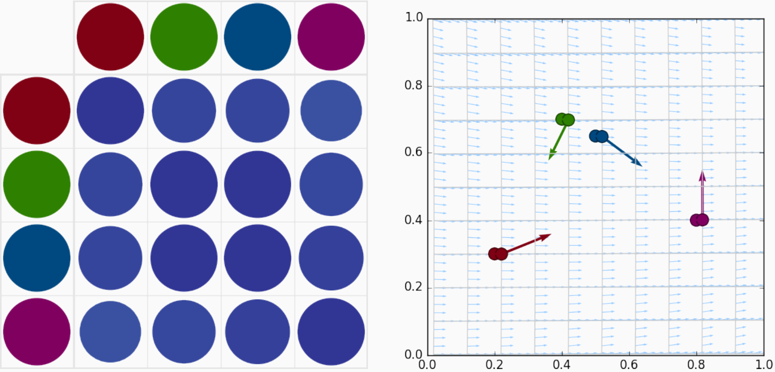
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

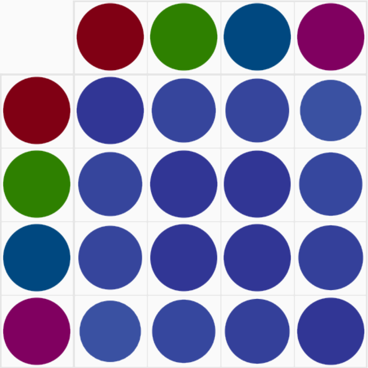
Influence of the kernel width, $\sigma = 1.$



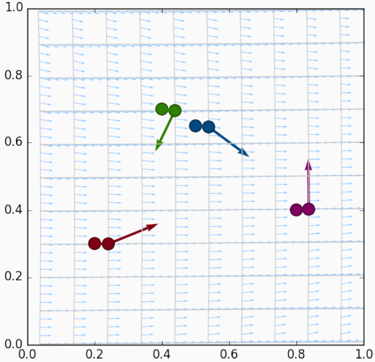
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

Influence of the kernel width, $\sigma = 1.$

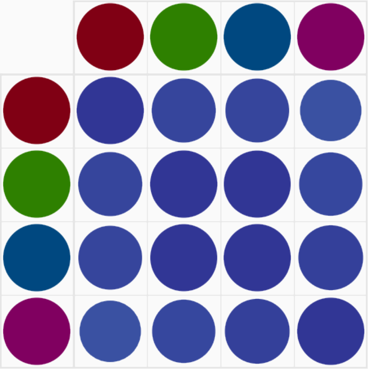


(a) Kernel matrix k_{q_t}.

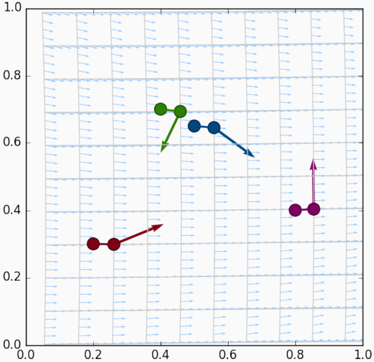


(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

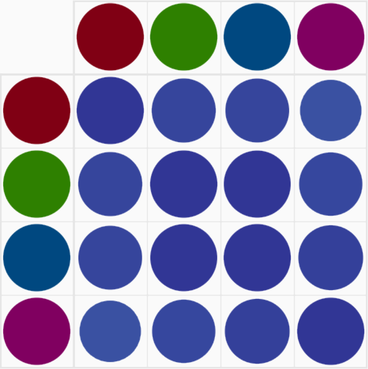


(a) Kernel matrix k_{q_t}.

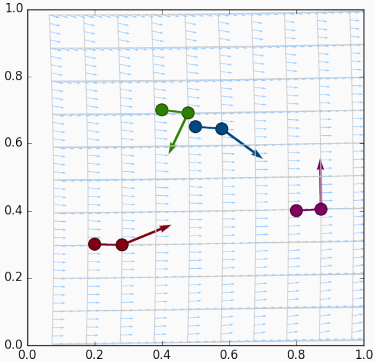


(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.



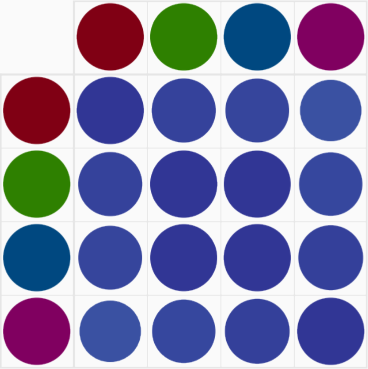
(a) Kernel matrix k_{q_t}.



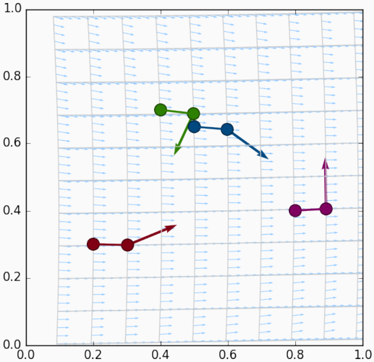
(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

Influence of the kernel width, $\sigma = 1.$



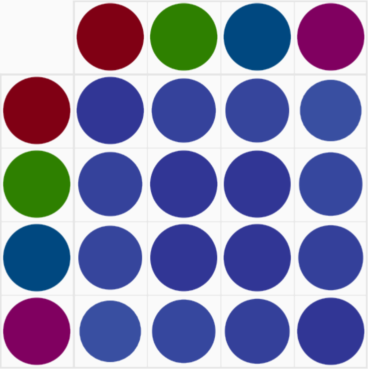
(a) Kernel matrix k_{q_t}.



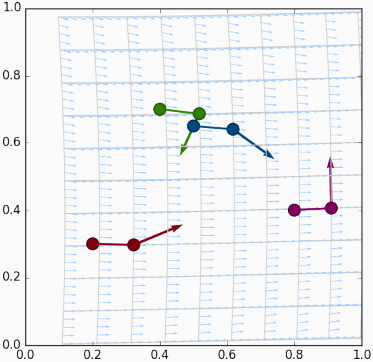
(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

Influence of the kernel width, $\sigma = 1.$



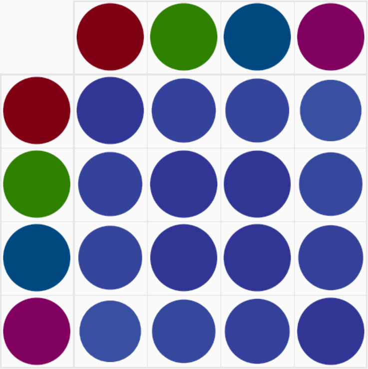
(a) Kernel matrix k_{q_t}.



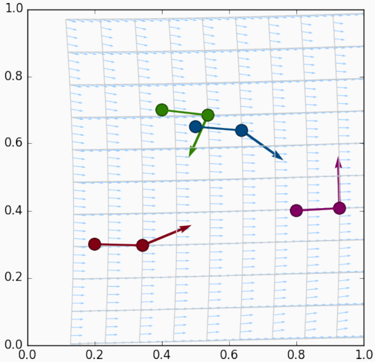
(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

Influence of the kernel width, $\sigma = 1.$



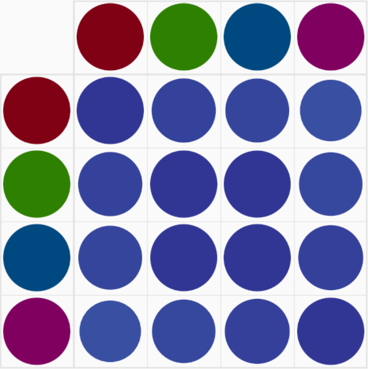
(a) Kernel matrix k_{q_t}.



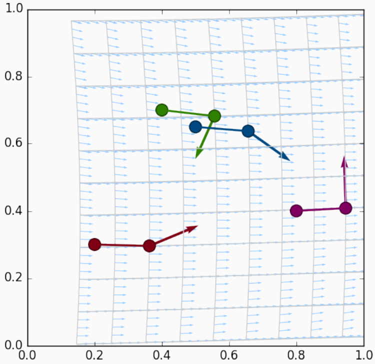
(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

Influence of the kernel width, $\sigma = 1.$



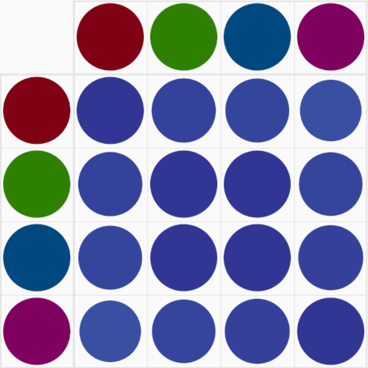
(a) Kernel matrix k_{q_t}.



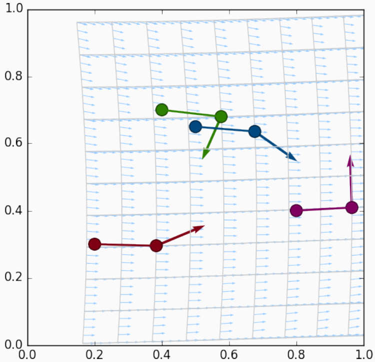
(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

Influence of the kernel width, $\sigma = 1.$



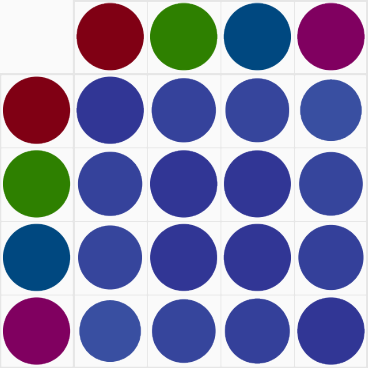
(a) Kernel matrix k_{q_t}.



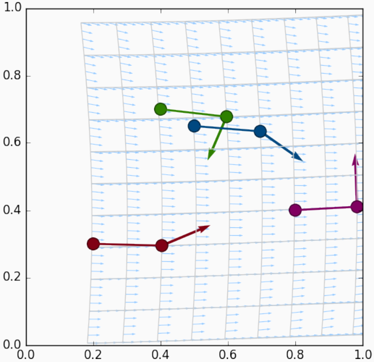
(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

Influence of the kernel width, $\sigma = 1$.



(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 14: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

Conclusion

We have now presented the *Large Deformation Diffeomorphic Metric Mapping*, or LDDMM setting :

- OT $\quad (\sigma = 0) \xrightarrow{\sigma++} G_k \xrightarrow{\sigma++} (\sigma = +\infty)$ Translations
- Deformations computed through geodesic shooting

Conclusion

We have now presented the *Large Deformation Diffeomorphic Metric Mapping*, or LDDMM setting :

- OT $(\sigma = 0) \xrightarrow{\sigma++} G_k \xrightarrow{\sigma++} (\sigma = +\infty)$ Translations
- Deformations computed through geodesic shooting

The (basic) framework relies on three pillars :

- Hamilton's theorem $(g_q \longrightarrow K_q)$
- The current availability of GPUs (parallelism)
- The Reduction Principle $((q_t, p_t) \longleftrightarrow \varphi_t)$

The diffeomorphic framework

An iterative matching algorithm

Variability decomposition

Let X and Y be two shapes, we are looking for a k-deformation $\varphi \in G_k$ such that :

$$X \xrightarrow{\varphi} \varphi(X) \leftrightarrows Y \quad \text{with minimal dissimilarity} \quad \text{“}\|\varphi(X) - Y\|^2\text{”}.$$

Variability decomposition

Let X and Y be two shapes, we are looking for a k-deformation $\varphi \in G_k$ such that :

$$X \xrightarrow{\varphi} \varphi(X) \leftrightarrows Y \quad \text{with minimal dissimilarity} \quad \text{“}\|\varphi(X) - Y\|^2\text{”}.$$

As dissimilarity, one can use generic kernel or wasserstein distances between measures, such as :

$$\|\varphi(X) - Y\|_s^2 \; = \; \|\mu - \nu\|_s^2 \; = \; \|B_s \star (\mu - \nu)\|_{L^2(\mathbb{R}^D)}^2 \,. \tag{22}$$

Variability decomposition

Let X and Y be two shapes, we are looking for a k-deformation $\varphi \in G_k$ such that :

$$X \xrightarrow{\varphi} \varphi(X) \leftrightarrows Y \quad \text{with minimal dissimilarity} \quad \text{“}\|\varphi(X) - Y\|^2\text{”}.$$

As dissimilarity, one can use generic kernel or wasserstein distances between measures, such as :

$$\|\varphi(X) - Y\|_s^2 \; = \; \|\mu - \nu\|_s^2 \; = \; \|B_s \star (\mu - \nu)\|_{L^2(\mathbb{R}^D)}^2 \, . \tag{22}$$

Ideally, we are looking for

$$p_s^\perp(Y \to G_k \cdot X) \; = \; \arg\min_{\varphi \in G_k} \|\varphi(X) - Y\|_s^2. \tag{23}$$

Regularized matching problem

However, in practice :

- G_k is not well understood
- We want $d_k(X, \varphi(X)) = d_{G_k}(\mathrm{Id}_{\mathbb{R}^D}, \varphi) \leqslant C < +\infty$

Regularized matching problem

However, in practice :

- G_k is not well understood
- We want $d_k(X, \varphi(X)) = d_{G_k}(\mathrm{Id}_{\mathbb{R}^D}, \varphi) \leqslant C < +\infty$

We settle for the minimization over the **deformation** φ of :

$$\mathrm{Cost}(\varphi) = \gamma_{\text{reg}} \cdot d_k^2(X, \varphi(X)) + \gamma_{\text{att}} \cdot \|\varphi(X) - Y\|_S^2. \tag{24}$$

Regularized matching problem

However, in practice :

- G_k is not well understood
- We want $d_k(X, \varphi(X)) = d_{G_k}(\mathrm{Id}_{\mathbb{R}^D}, \varphi) \leqslant C < +\infty$

We settle for the minimization over the **deformation** φ of :

$$\mathrm{Cost}(\varphi) = \gamma_{\text{reg}} \cdot d_k^2(X, \varphi(X)) + \gamma_{\text{att}} \cdot \|\varphi(X) - Y\|_S^2. \tag{24}$$

That is, minimize over the **shooting momentum** p_0 :

$$\mathrm{Cost}(\mathbf{p_0}) = \gamma_{\text{reg}} \cdot p_0^{\intercal} K_{q_0} p_0 + \gamma_{\text{att}} \cdot \|\mathbf{q_1} - Y\|_S^2. \tag{25}$$

Regularized matching problem

However, in practice :

- G_k is not well understood
- We want $d_k(X, \varphi(X)) = d_{G_k}(\mathrm{Id}_{\mathbb{R}^D}, \varphi) \leqslant C < +\infty$

We settle for the minimization over the **deformation** φ of :

$$\mathrm{Cost}(\varphi) = \gamma_{\mathrm{reg}} \cdot d_k^2(X, \varphi(X)) + \gamma_{\mathrm{att}} \cdot \|\varphi(X) - Y\|_S^2. \tag{24}$$

That is, minimize over the **shooting momentum** p_0 :

$$\mathrm{Cost}(\mathbf{p_0}) = \gamma_{\mathrm{reg}} \cdot p_0^\intercal K_{q_0} p_0 + \gamma_{\mathrm{att}} \cdot \|\mathbf{q_1} - Y\|_S^2. \tag{25}$$

If $\gamma_{\mathrm{reg}} << \gamma_{\mathrm{att}}$, q_1 should be good enough.

Gradient descent on finite-dimensional manifolds

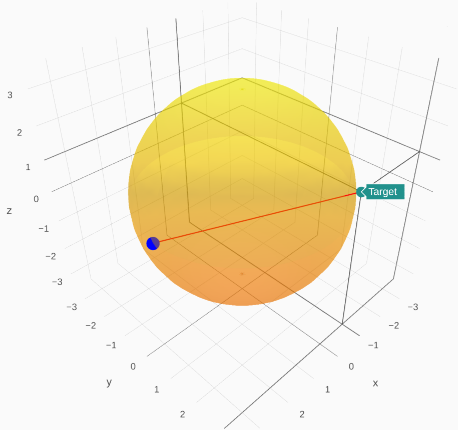


Figure 15: Matching from the source X to the target Y, constrained to the golden sphere $G_k \cdot X$.

Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, \varphi(X))$ is much less constrained than the dissimilarity $\|\varphi(X) - Y\|_S^2$.

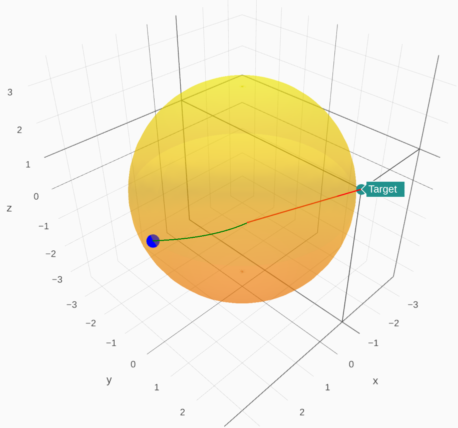


Figure 15: Matching from the source X to the target Y, constrained to the golden sphere $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, \varphi(X))$ is much less constrained than the dissimilarity $\|\varphi(X) - Y\|_s^2$.

Gradient descent on finite-dimensional manifolds

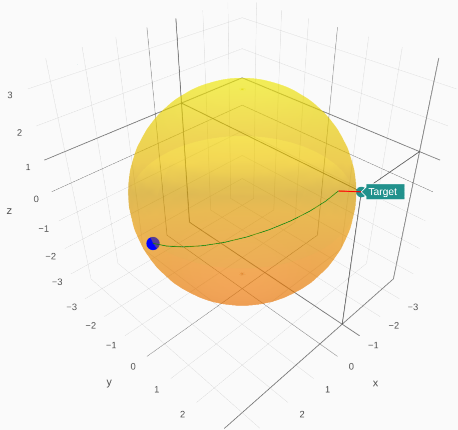


Figure 15: Matching from the source X to the target Y, constrained to the golden sphere $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, \varphi(X))$ is much less constrained than the dissimilarity $\|\varphi(X) - Y\|_S^2$.

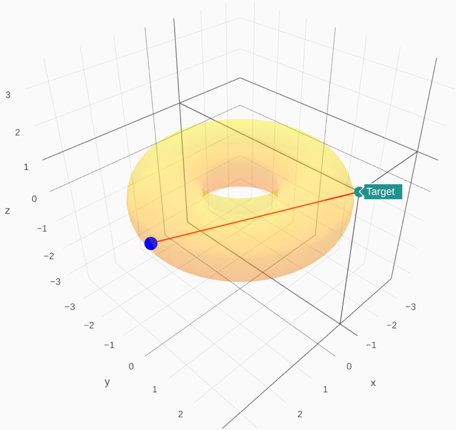


Figure 16: Matching from the source X to the target Y, constrained to the golden torus $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, \varphi(X))$ is much less constrained than the dissimilarity $\|\varphi(X) - Y\|_S^2$.

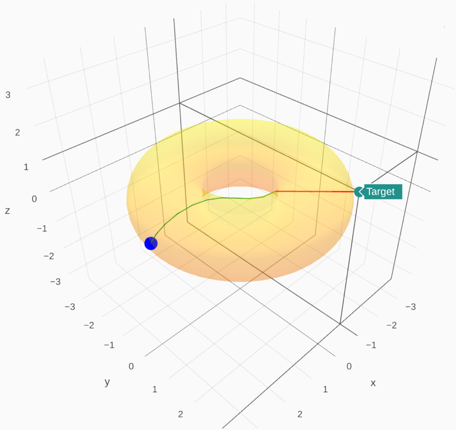


Figure 16: Matching from the source X to the target Y, constrained to the golden torus $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, \varphi(X))$ is much less constrained than the dissimilarity $\|\varphi(X) - Y\|_s^2$.

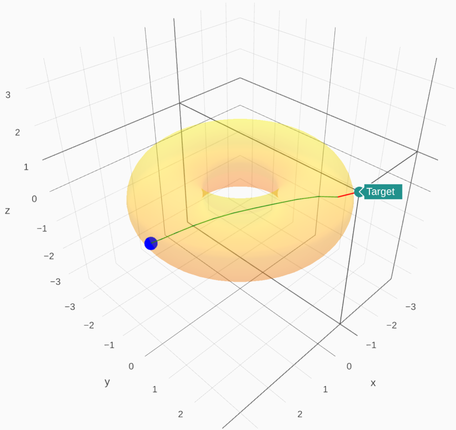


Figure 16: Matching from the source X to the target Y, constrained to the golden torus $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, \varphi(X))$ is much less constrained than the dissimilarity $\|\varphi(X) - Y\|_S^2$.

The diffeomorphic framework

Let’s read some code

The `theano` library

```python
# Import the relevant tools
import time                  # to measure performance
import numpy as np           # standard array library
import theano                # Autodiff & symbolic calculus library :
import theano.tensor as T    #             - mathematical tools;
from   theano import config, printing #   - printing of the Sinkhorn error.
```

`theano` :

- Is a `python` library
- Symbolic computations $\Longrightarrow$ efficient CPU/GPU binaries
- Auto-differentiates expressions

The `theano` library

```python
# Import the relevant tools
import time                  # to measure performance
import numpy as np           # standard array library
import theano                # Autodiff & symbolic calculus library :
import theano.tensor as T    #              - mathematical tools;
from   theano import config, printing #   - printing of the Sinkhorn error.
```

`theano` :

- Is a `python` library
- Symbolic computations $\Longrightarrow$ efficient CPU/GPU binaries
- Auto-differentiates expressions
- It changed my life... Let's see why.

The Hamiltonian

```python
# Part 1 : kinetic energy on the phase space (Hamiltonian) =========================


def _squared_distances(x, y) :
    "Returns the matrix of |x_i-y_j|^2."
    x_col = x.dimshuffle(0, 'x', 1)
    y_lin = y.dimshuffle('x', 0, 1)
    return T.sum( (x_col - y_lin)**2 , 2 )

def _k(x, y, s) :
    "Returns the matrix of k(x_i,y_j)= 1/(1+|x_i-y_j|^2)^{1/4}, with a heavy tail."
    sq = _squared_distances(x, y) / (s**2)
    return T.pow( 1. / ( 1. + sq ), .25 )

def _cross_kernels(q, x, s) :
    "Returns the full k-correlation matrices between two point clouds q and x."
    K_qq = _k(q, q, s)
    K_qx = _k(q, x, s)
    K_xx = _k(x, x, s)
    return (K_qq, K_qx, K_xx)

def _Hqp(q, p, sigma) :
    "The hamiltonian, or kinetic energy of the shape q with momenta p."
    pKqp =  _k(q, q, sigma) * (p.dot(p.T))# Use a simple isotropic kernel
    return .5 * T.sum(pKqp)               # H(q,p) = 1/2 · sum_{i,j} k(x_i,x_j)p_i.p_j
```

Geodesic shooting

```
# Part 2 : Geodesic shooting ======================================================


# The partial derivatives of the Hamiltonian are automatically computed !
def _dq_Hqp(q,p,sigma) :
    return T.grad(_Hqp(q,p,sigma), q)
def _dp_Hqp(q,p,sigma) :
    return T.grad(_Hqp(q,p,sigma), p)

def _hamiltonian_step(q,p, sigma) :
    "Simplistic euler scheme step with dt = .1."
    return [q + .1 * _dp_Hqp(q,p,sigma) ,
            p - .1 * _dq_Hqp(q,p,sigma) ]

def _HamiltonianShooting(q, p, sigma) :
    "Shoots to time 1 a k-geodesic starting (at time 0) from q with momentum p."
    # We use the "scan" theano routine, which  can be understood as a "for" loop
    result, updates = theano.scan(fn            = _hamiltonian_step,
                                  outputs_info  = [q,p],
                                  non_sequences = sigma,
                                  n_steps       = 10    )  # hardcode the "dt = .1"
    # We do not store the intermediate results,
    # and only return the final state + momentum :
    final_result = [result[0][-1], result[1][-1]]
    return final_result
```

OT fidelity, part 1

```
# Part 3 : Data attachment ============================================================

def _ot_matching(q1_x, q1_mu, xt_x, xt_mu, radius) :
    """
    Given two measures q1 and xt represented by locations/weights arrays,
    outputs an optimal transport fidelity term and the transport plan.
    """
    # The Sinkhorn algorithm takes as input three Theano variables :
    c = _squared_distances(q1_x, xt_x) # Wasserstein cost function
    mu = q1_mu ; nu = xt_mu

    # Parameters of the Sinkhorn algorithm.
    epsilon           = (.02)**2          # regularization parameter
    rho               = (.5) **2          # unbalanced transport (Lenaic Chizat)
    niter             = 10000             # max niter in the sinkhorn loop
    tau               = -.8               # Nesterov-like acceleration
    lam = rho / (rho + epsilon)           # Update exponent
    # Elementary operations ..........................................................
    def ave(u,u1) :
        "Barycenter subroutine, used by kinetic acceleration through extrapolation."
        return tau * u + (1-tau) * u1
    def M(u,v)  :
        "M_{ij} = (-c_{ij} + u_i + v_j) / \epsilon"
        return (-c + u.dimshuffle(0,'x') + v.dimshuffle('x',0)) / epsilon
    lse = lambda A    : T.log(T.sum( T.exp(A), axis=1 ) + 1e-6) # prevents NaN
```

OT fidelity, part 2

```
# Actual Sinkhorn loop ..........................................................
# Iteration step :
def sinkhorn_step(u, v, foo) :
    u1=u # useful to check the update
    u = ave( u, lam * ( epsilon * ( T.log(mu) - lse(M(u,v))   ) + u ) )
    v = ave( v, lam * ( epsilon * ( T.log(nu) - lse(M(u,v).T) ) + v ) )
    err = T.sum(abs(u - u1))
    # "break" the loop if error < tol
    return (u,v,err), theano.scan_module.until(err < 1e-4)

# Scan = "For loop" :
err0 = np.arange(1, dtype=config.floatX)[0]
result, updates = theano.scan( fn            = sinkhorn_step, # Iterated routine
                               outputs_info  = [(0.*mu), (0.*nu), err0], # Start
                               n_steps       = niter          # Number of iters
                             )
U, V = result[0][-1], result[1][-1] # We only keep the final dual variables
Gamma = T.exp( M(U,V) )             # Transport plan g = diag(a)*K*diag(b)
cost  = T.sum( Gamma * c )          # Simplistic cost, chosen for readability
if True : # Shameful hack to prevent the pruning of the error-printing node...
    print_err_shape = printing.Print('error  : ', attrs=['shape'])
    errors          = print_err_shape(result[2])
    print_err  = printing.Print('error  : ') ; err_fin  = print_err(errors[-1])
    cost += .00000001 * err_fin
return [cost, Gamma]
```

Kernel fidelity, Data attachment term

```python
def _kernel_matching(q1_x, q1_mu, xt_x, xt_mu, radius) :
    """
    Given two measures q1 and xt represented by locations/weights arrays,
    outputs a kernel-fidelity term and an empty 'info' array.
    """
    K_qq, K_qx, K_xx = _cross_kernels(q1_x, xt_x, radius)
    q1_mu = q1_mu.dimshuffle(0,'x')  # column
    xt_mu = xt_mu.dimshuffle(0,'x')  # column
    cost = .5 * (   T.sum(K_qq * q1_mu.dot(q1_mu.T)) \
                 +  T.sum(K_xx * xt_mu.dot(xt_mu.T)) \
                 -2*T.sum(K_qx * q1_mu.dot(xt_mu.T))  )

    [...]  # error-tracking stuff
    return [cost , ... ]

def _data_attachment(q1_measure, xt_measure, radius) :
    "Given two measures and a radius, returns a cost (Theano symbolic variable)."
    if radius == 0 : # Convenient way to allow the choice of a method
        return _ot_matching(q1_measure[0], q1_measure[1],
                            xt_measure[0], xt_measure[1],
                            radius)
    else :
        return _kernel_matching(q1_measure[0], q1_measure[1],
                                xt_measure[0], xt_measure[1],
                                radius)
```

Actual cost function

```
# Part 4 : Cost function and derivatives ============================================


def _cost( q,p, xt_measure, connec, params ) :
    """
    Returns a total cost, sum of a small regularization term and the data attachment.
    .. math ::

        C(q_0, p_0) = .1 * H(q0,p0) + 1 * A(q_1, x_t)

    Needless to say, the weights can be tuned according to the signal-to-noise ratio.
    """
    s,r  = params                        # Deformation scale, Attachment scale
    q1 = _HamiltonianShooting(q,p,s)[0]  # Geodesic shooting from q0 to q1
    # Convert the set of vertices 'q1' into a measure.
    q1_measure  = Curve._vertices_to_measure( q1, connec )
    attach_info = _data_attachment( q1_measure,  xt_measure,  r )
    return [ .1* _Hqp(q, p, s) + 1.* attach_info[0] , attach_info[1] ] # [cost, info]


# The discrete backward scheme is automatically computed :
def _dcost_p( q,p, xt_measure, connec, params ) :
    "The gradients of C wrt. p_0 is automatically computed."
    return T.grad( _cost(q,p, xt_measure, connec, params)[0] , p)

```

Minimization script, part 1

```
def perform_matching( Q0, Xt, params, scale_momentum = 1, scale_attach = 1) :
    """ Performs a matching from the source Q0 to the target Xt,
        returns the optimal momentum P0.                            """
    (Xt_x, Xt_mu) = Xt.to_measure()          # Transform the target into a measure
    q0 = Q0.points ; p0 = np.zeros(q0.shape)  # Null initialization for the momentum

    # Compilation ---------------------------------------------------------------------
    print('Compiling the energy functional.')
    time1 = time.time()
    # Cost is a function of 6 parameters :
    # The source 'q',                    the starting momentum 'p',
    # the target points 'xt_x',          the target weights 'xt_mu',
    # the deformation scale 'sigma_def', the attachment scale 'sigma_att'.
    q, p, xt_x = T.matrices('q', 'p', 'xt_x') ; xt_mu = T.vector('xt_mu') # types

    # Compilation. Depending on settings specified in the ~/.theanorc file or
    # given at execution time, this will produce CPU or GPU code under the hood.
    Cost  = theano.function([q,p, xt_x,xt_mu ],
                        [   _cost( q,p, (xt_x,xt_mu), Q0.connectivity, params )[0],
                         _dcost_p( q,p, (xt_x,xt_mu), Q0.connectivity, params )   ,
                            _cost( q,p, (xt_x,xt_mu), Q0.connectivity, params )[1] ],
                         allow_input_downcast=True)
    time2 = time.time()
    print('Compiled in : ', '{0:.2f}'.format(time2 - time1), 's')

```

Minimization script, part 2

```python
    # Display pre-computing ---------------------------------------------------------
    connec   = Q0.connectivity ; q0 = Q0.points ;
    g0,cgrid = GridData()      ; G0 = Curve(g0, cgrid )
    # Given q0, p0 and grid points grid0 , outputs (q1,p1,grid1) after the flow
    # of the geodesic equations from t=0 to t=1 :
    ShootingVisualization = VisualizationRoutine(q0, params)
    # L-BFGS minimization -----------------------------------------------------------
    from scipy.optimize import minimize
    def matching_problem(p0_vec) :
        "Energy minimized in the variable 'p0'."
        p0 = p0_vec.reshape(q0.shape)
        [c, dp_c, info] = Cost(q0, p0, Xt_x, Xt_mu)
        matching_problem.Info = info
        if (matching_problem.it % 1 == 0) and (c < matching_problem.bestc) :
            matching_problem.bestc = c
            q1,p1,g1 = ShootingVisualization(q0, p0, np.array(g0))
            Q1 = Curve(q1, connec) ; G1 = Curve(g1, cgrid )
            DisplayShoot( Q0, G0, p0, Q1, G1, Xt, info,
                              matching_problem.it, scale_momentum, scale_attach)
        print('Iteration : ',matching_problem.it,', cost : ',c,' info : ',info.shape)
        matching_problem.it += 1
        # The fortran routines used by scipy.optimize expect float64 vectors
        # instead of gpu-friendly float32 matrices: we need a slight conversion
        return (c, dp_c.ravel().astype('float64'))
    matching_problem.bestc=np.inf ; matching_problem.it=0 ; matching_problem.Info=None
```

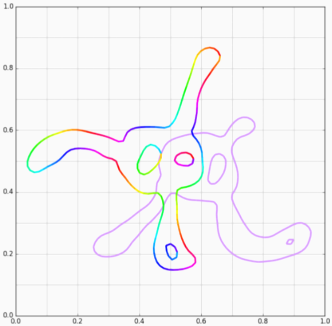
Minimization script, part 3

```
    time1 = time.time()
    res = minimize( matching_problem,     # function to minimize
                    p0.ravel(),           # starting estimate
                    method = 'L-BFGS-B',  # an order 2 method
                    jac = True,           # matching_problems returns the gradient
                    options = dict(
                        maxiter = 1000,   # max number of iterations
                        ftol    = .000001,# Don't bother fitting to float precision
                        maxcor  = 10      # Prev. grads. used to approx. the Hessian
                    ))
    time2 = time.time()

    p0 = res.x.reshape(q0.shape)
    print('Convergence success  : ', res.success, ', status = ', res.status)
    print('Optimization message : ', res.message.decode('UTF-8'))
    print('Final cost   after ', res.nit, ' iterations : ', res.fun)
    print('Elapsed time after ', res.nit, ' iterations : ',
                                '{0:.2f}'.format(time2 - time1), 's')
    return p0, matching_problem.Info

def matching_demo(source_file, target_file, params, scale_mom = 1, scale_att = 1) :
    Q0 = Curve.from_file(source_file) # Load source...
    Xt = Curve.from_file(target_file) # and target.
    # Compute the optimal shooting momentum :
    p0, info = perform_matching( Q0, Xt, params, scale_mom, scale_att)
```

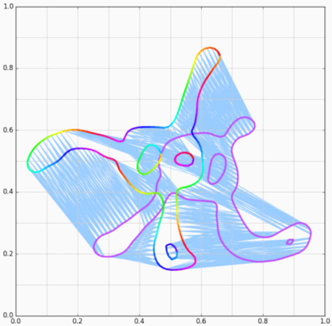
The diffeomorphic framework

Results

Typical run with OT fidelity



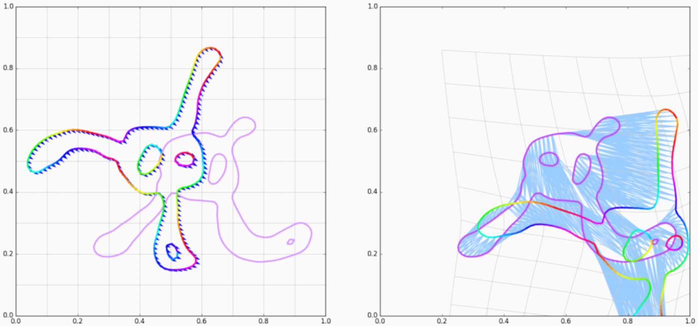
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 0.

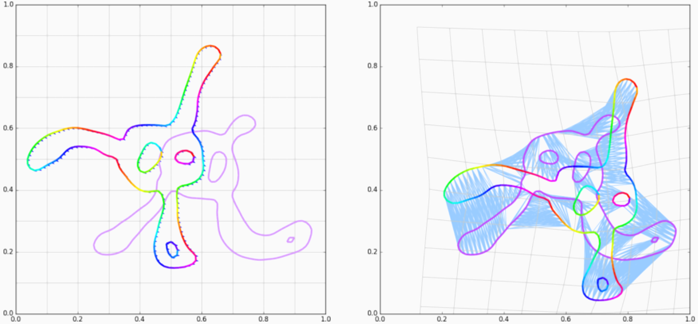
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 3.

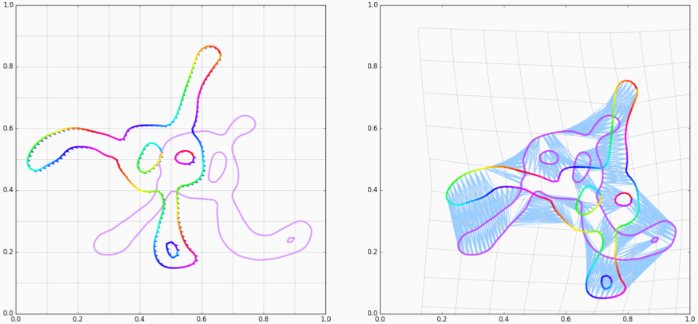
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 4.

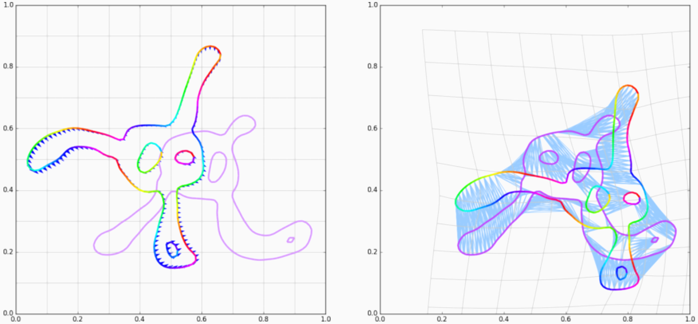
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

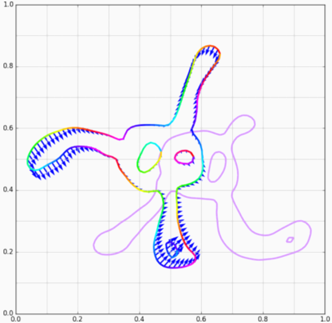
Figure 17: Iteration 5.

Typical run with OT fidelity

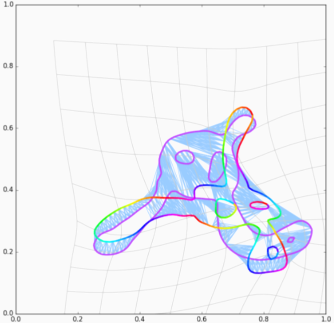


(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 6.



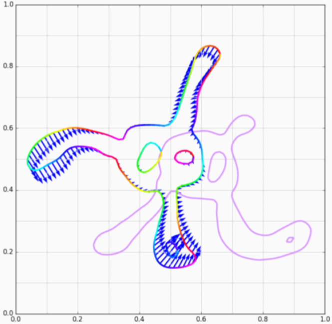
(a) Momentum p_0.



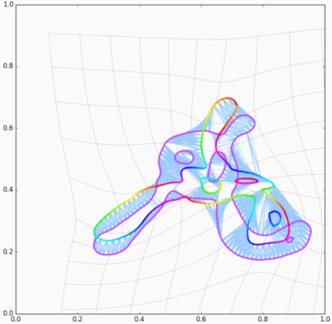
(b) Shooted model q_1.

Figure 17: Iteration 7.

Typical run with OT fidelity



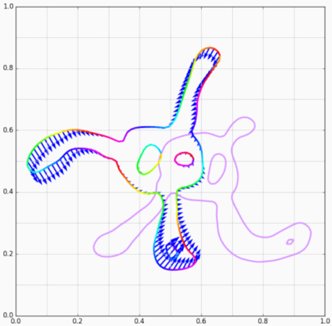
(a) Momentum p_0.



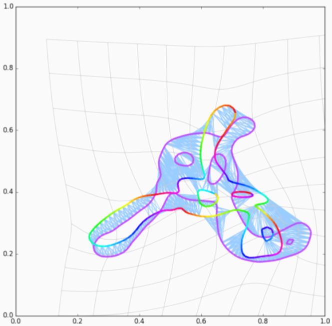
(b) Shooted model q_1.

Figure 17: Iteration 8.

Typical run with OT fidelity

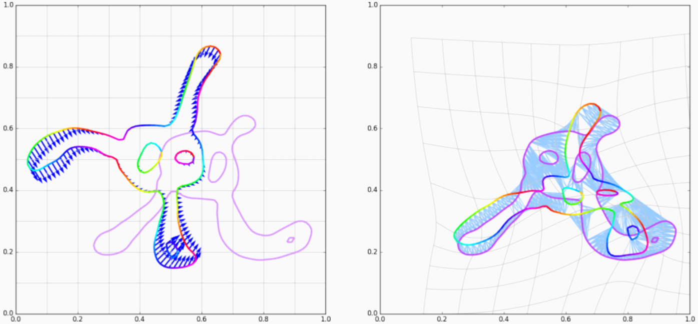


(a) Momentum p_0.



(b) Shooted model q_1.

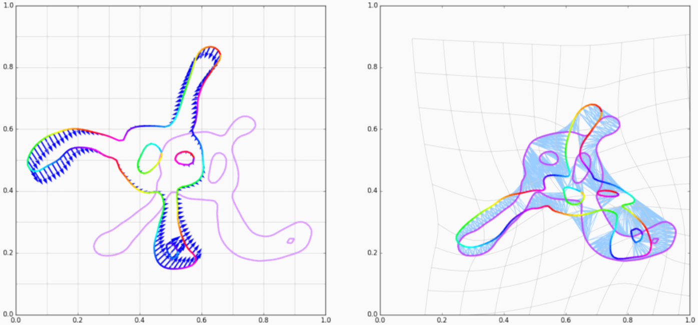
Figure 17: Iteration 9.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 10.

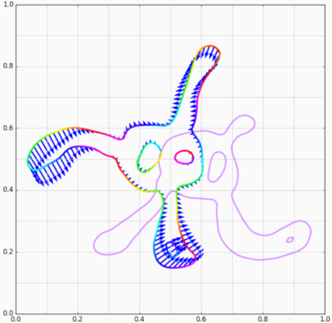
Typical run with OT fidelity



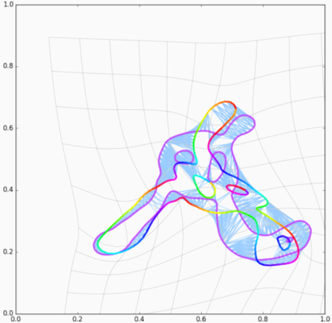
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 11.

Typical run with OT fidelity



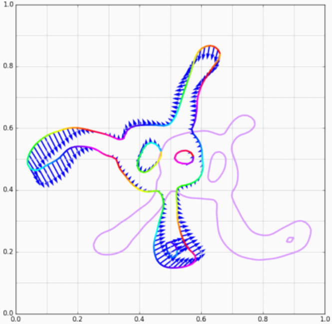
(a) Momentum p_0.



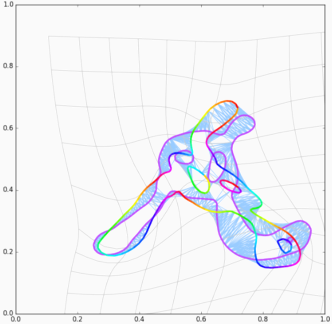
(b) Shooted model q_1.

Figure 17: Iteration 12.

Typical run with OT fidelity



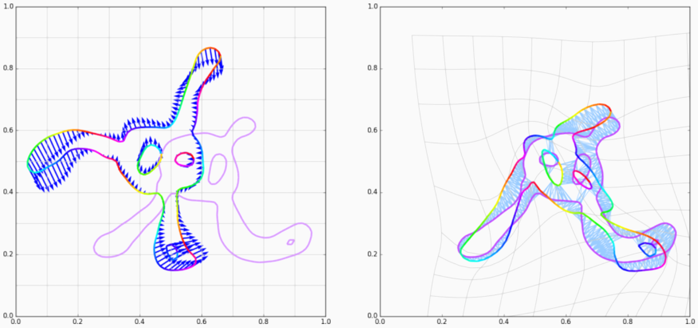
(a) Momentum p_0.



(b) Shooted model q_1.

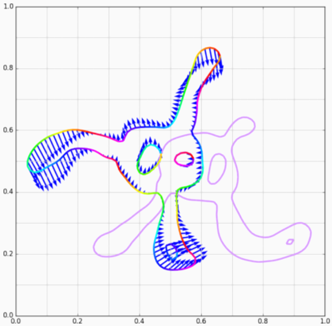
Figure 17: Iteration 13.

Typical run with OT fidelity

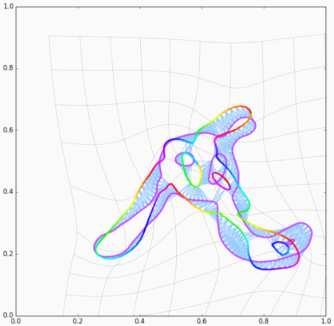


(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 14.

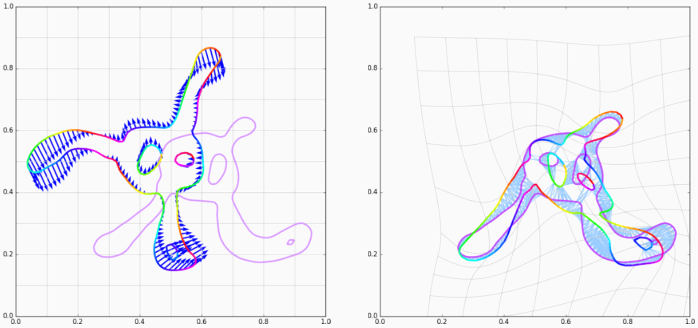


(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 15.

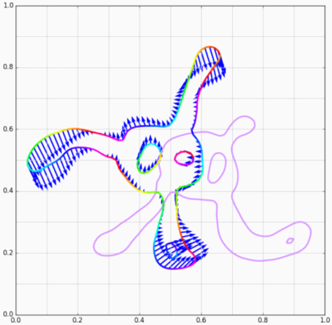


(a) Momentum p_0.

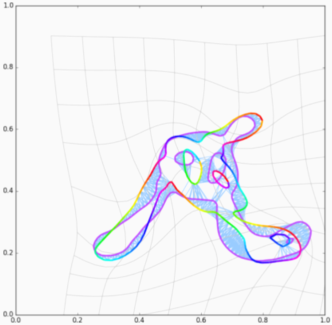
(b) Shooted model q_1.

Figure 17: Iteration 16.

Typical run with OT fidelity



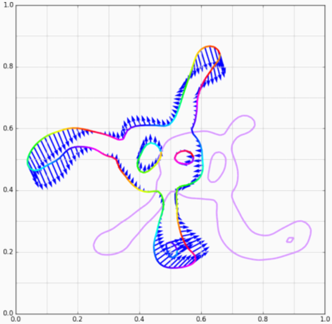
(a) Momentum p_0.



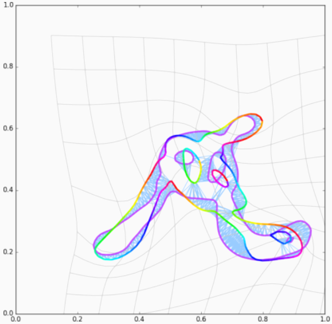
(b) Shooted model q_1.

Figure 17: Iteration 17.

Typical run with OT fidelity



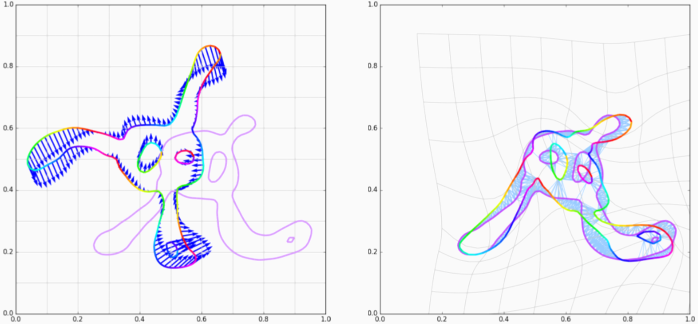
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 18.

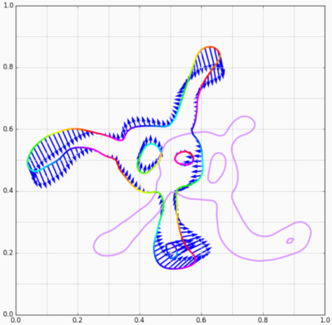
Typical run with OT fidelity



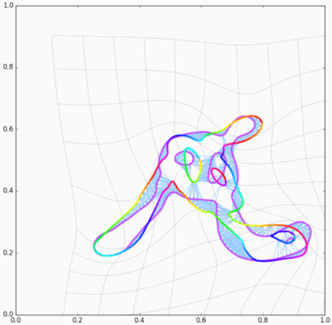
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 19.

Typical run with OT fidelity

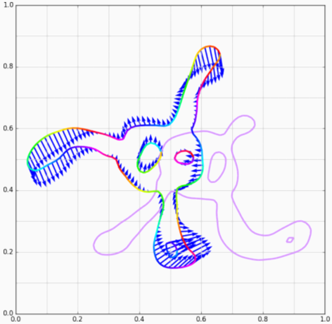


(a) Momentum p_0.

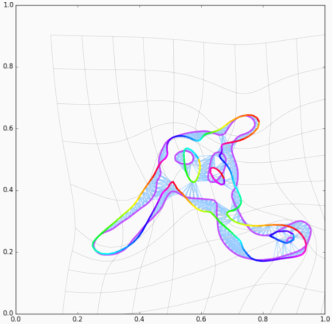


(b) Shooted model q_1.

Figure 17: Iteration 20.



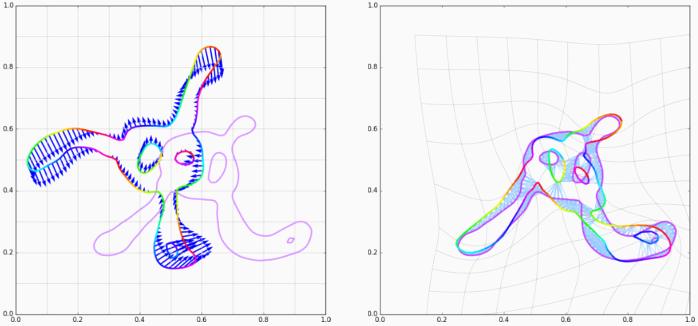
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 21.

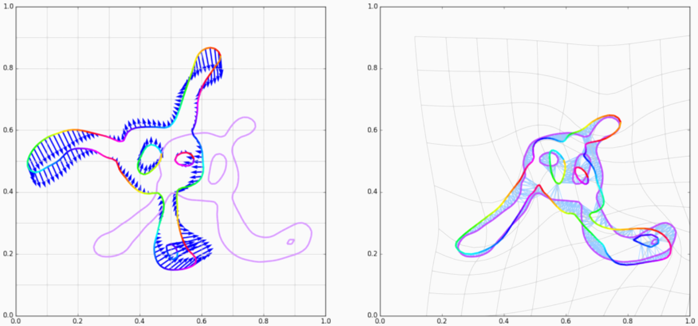
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 22.

Typical run with OT fidelity

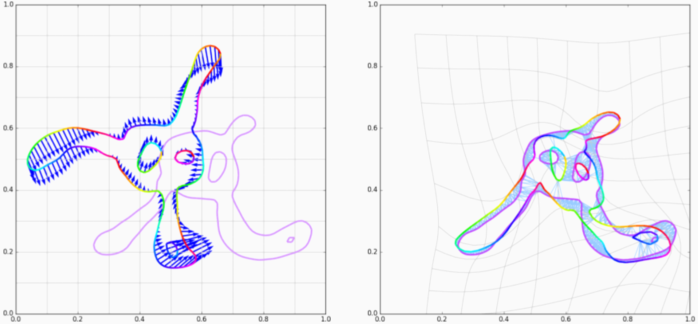


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 17: Iteration 23.

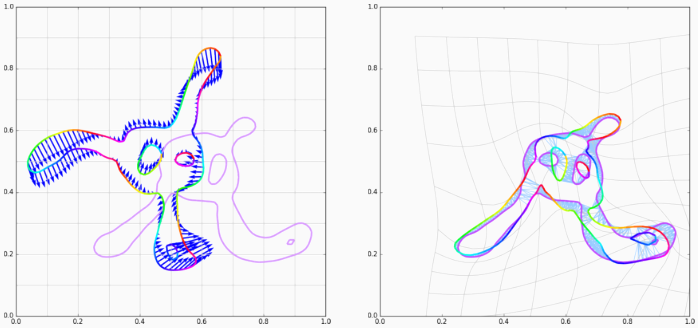
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 24.

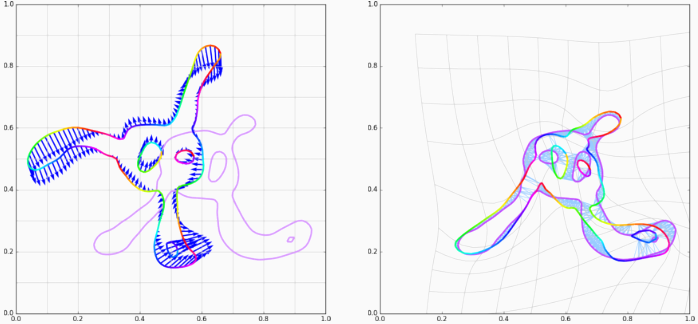
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 25.

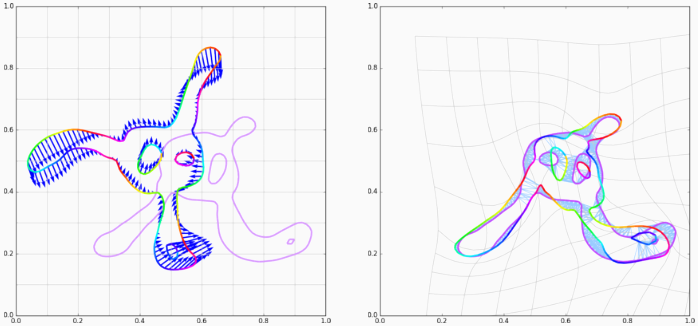
Typical run with OT fidelity



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 17: Iteration 26.

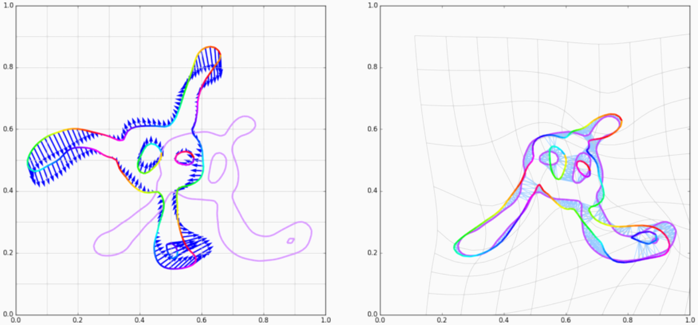
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 27.

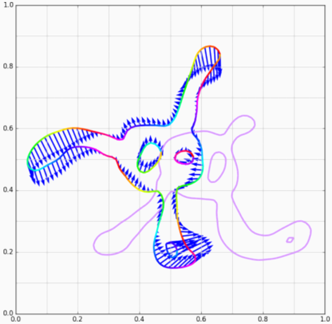
Typical run with OT fidelity



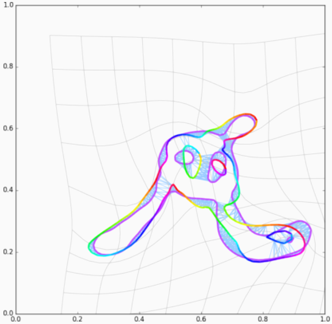
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 28.

Typical run with OT fidelity



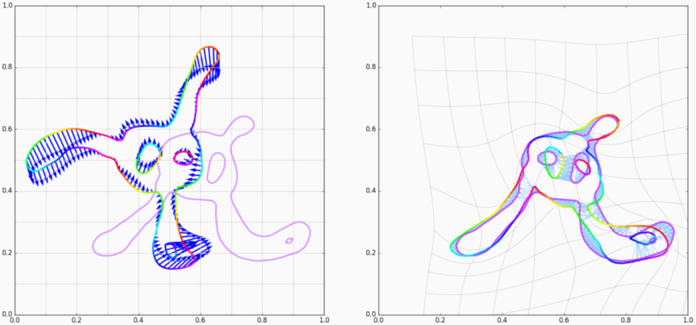
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 29.

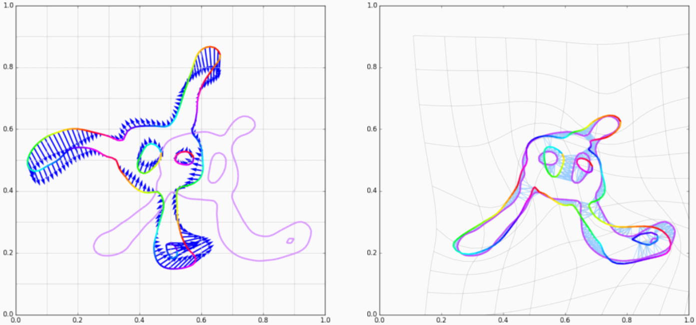
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

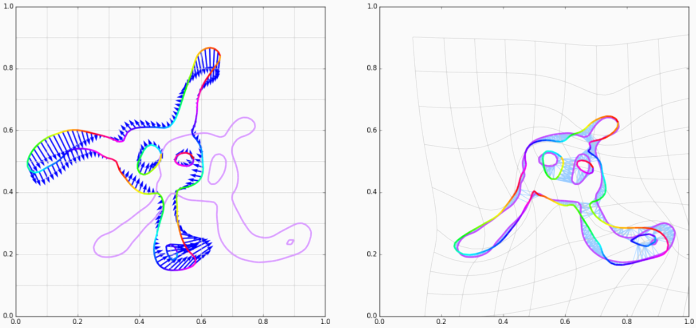
Figure 17: Iteration 30.

Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

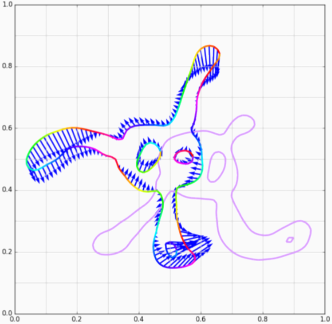
Figure 17: Iteration 31.



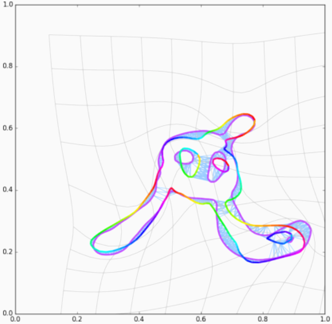
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 32.

Typical run with OT fidelity



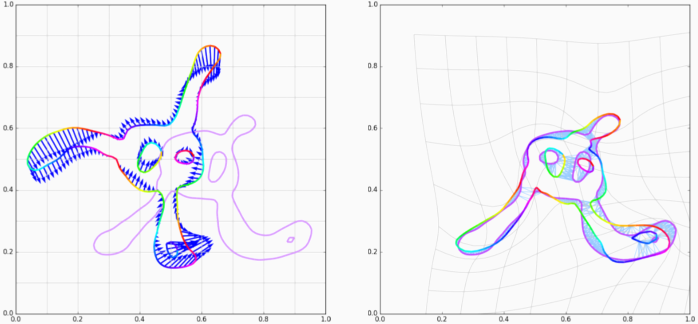
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 33.

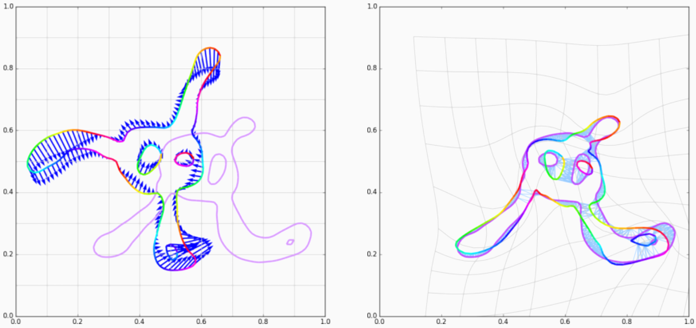
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

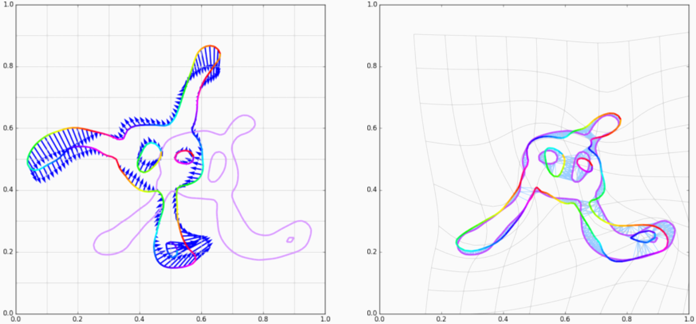
Figure 17: Iteration 34.

Typical run with OT fidelity



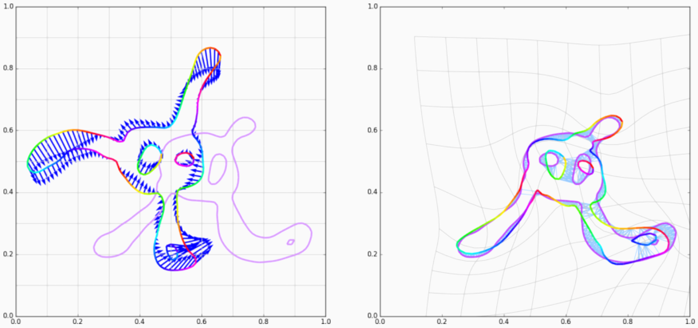
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 35.



(a) Momentum p_0. (b) Shooted model q_1.

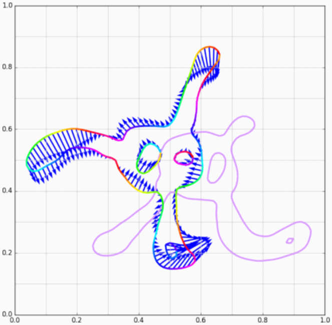
Figure 17: Iteration 36.



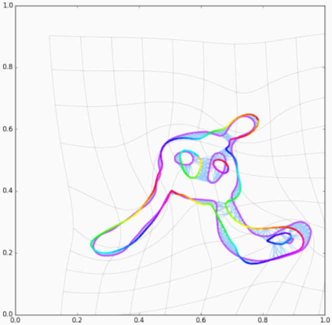
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 37.

Typical run with OT fidelity



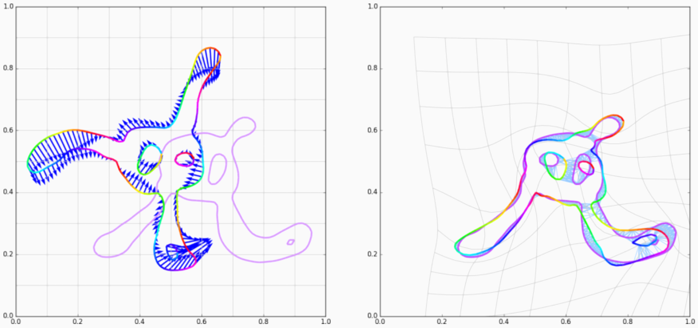
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 38.

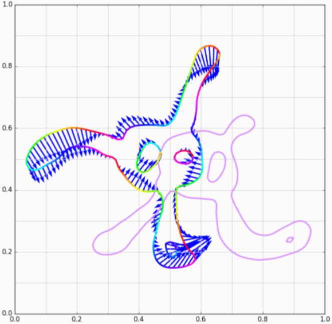
Typical run with OT fidelity



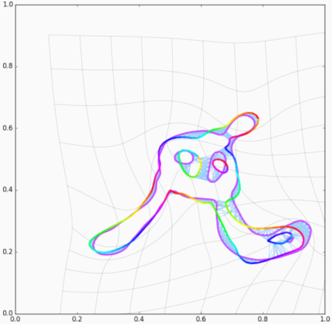
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 39.

Typical run with OT fidelity



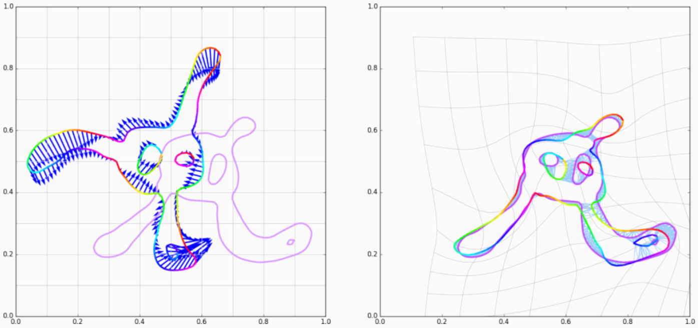
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 41.

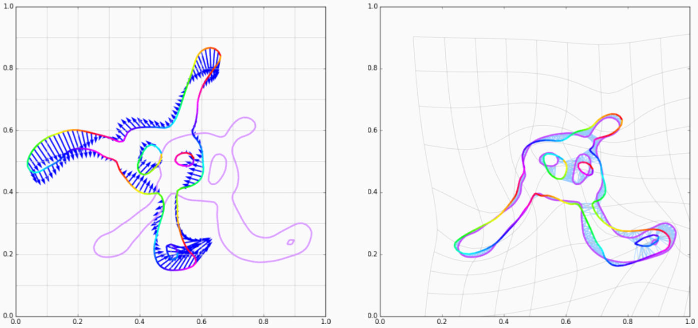
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 42.

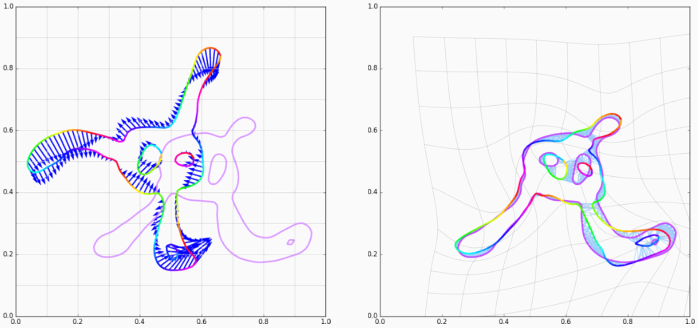
Typical run with OT fidelity



(a) Momentum p_0.　(b) Shooted model q_1.

Figure 17: Iteration 43.

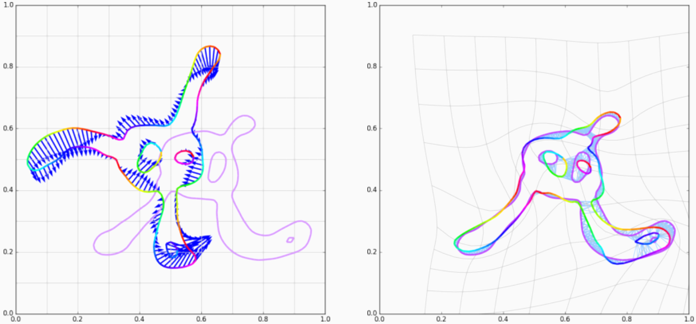
Typical run with OT fidelity



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 17: Iteration 44.

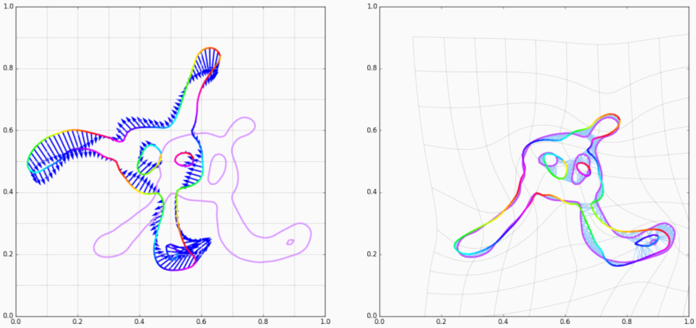
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 46.

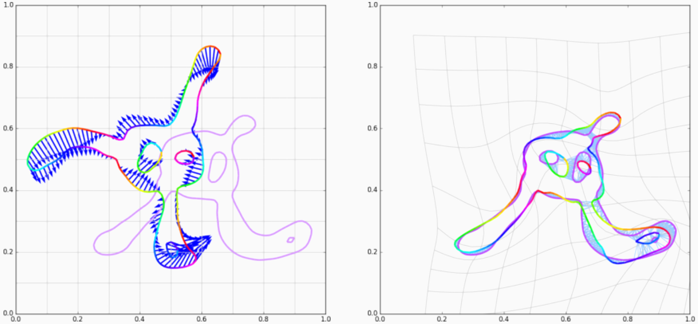
Typical run with OT fidelity



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 17: Iteration 47.

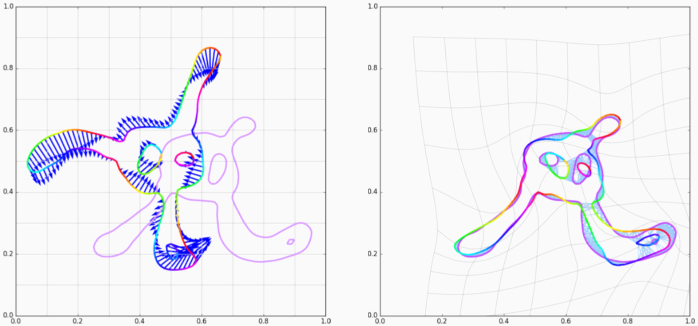
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 48.

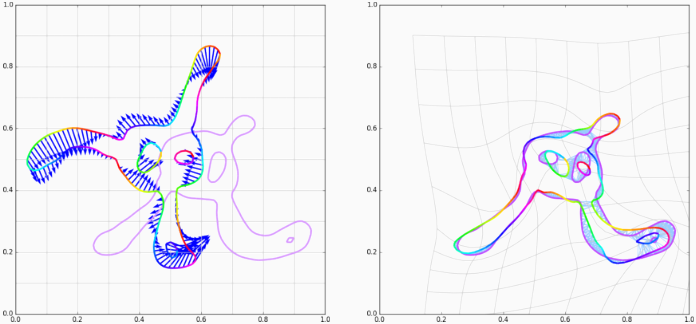
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 49.

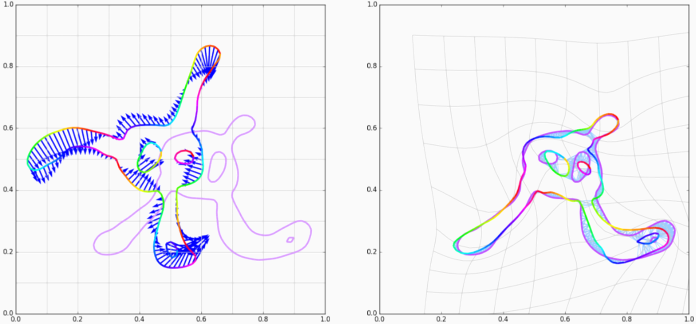
Typical run with OT fidelity



(a) Momentum p_0.　**(b)** Shooted model q_1.

Figure 17: Iteration 50.

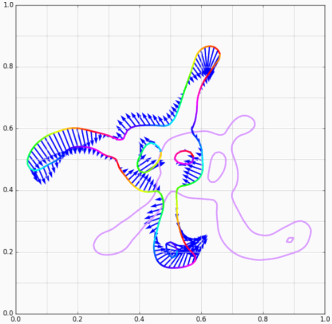
Typical run with OT fidelity



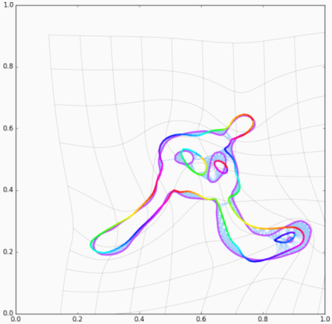
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 52.

Typical run with OT fidelity



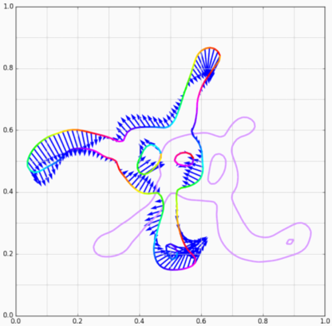
(a) Momentum p_0.



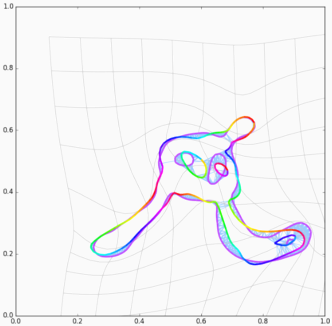
(b) Shooted model q_1.

Figure 17: Iteration 53.

Typical run with OT fidelity



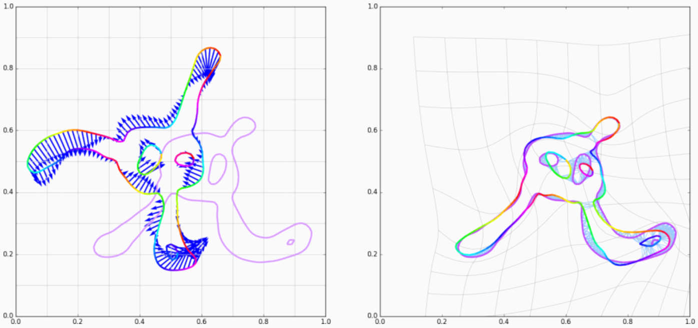
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 54.

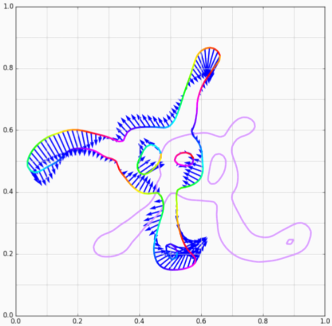
Typical run with OT fidelity



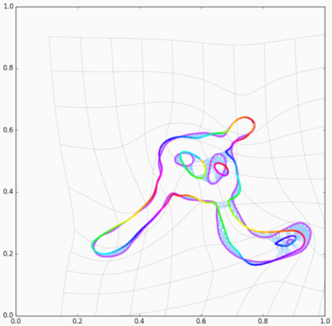
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 55.

Typical run with OT fidelity



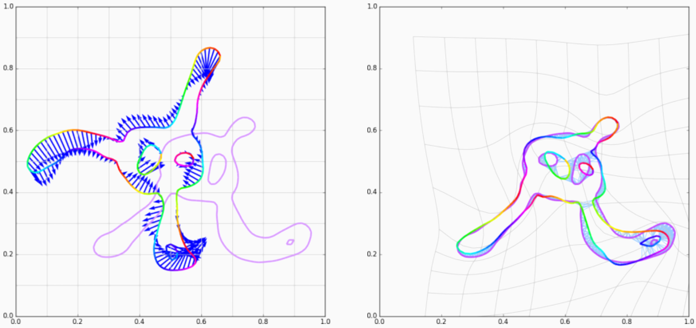
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 56.

Typical run with OT fidelity

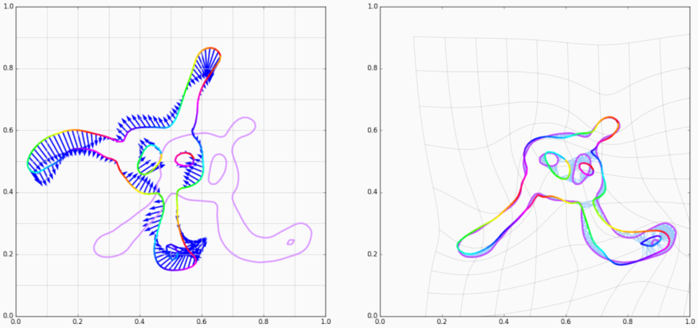


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 17: Iteration 57.

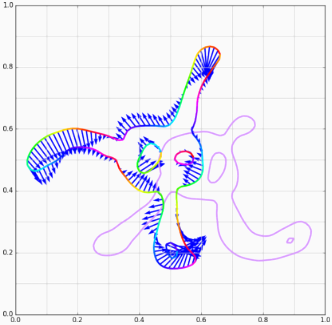
Typical run with OT fidelity



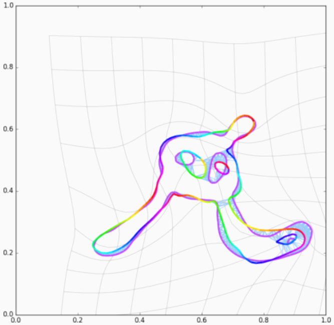
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 58.

Typical run with OT fidelity



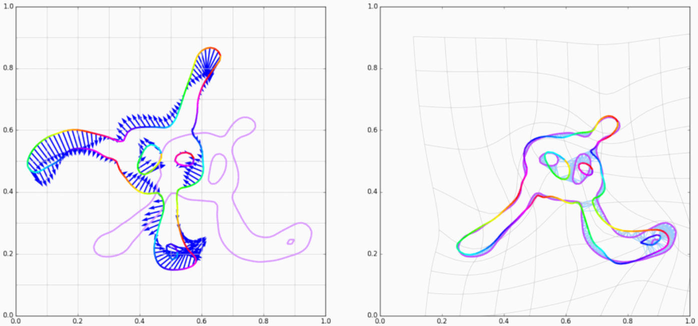
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 59.

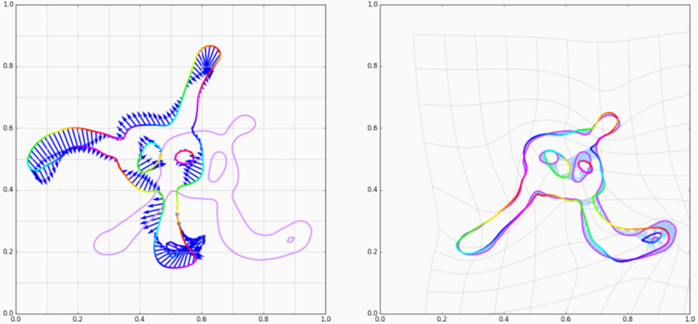
Typical run with OT fidelity



(a) Momentum p_0.

(b) Shooted model q_1.

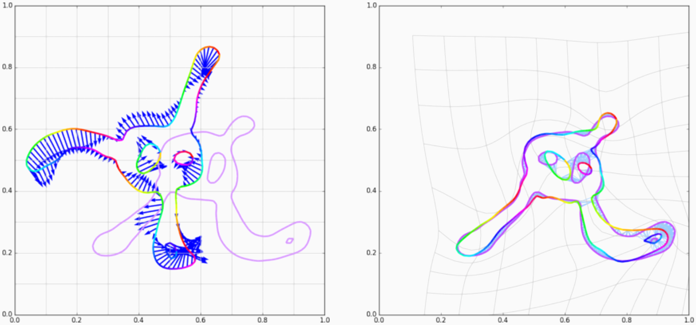
Figure 17: Iteration 60.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 61.

Typical run with OT fidelity

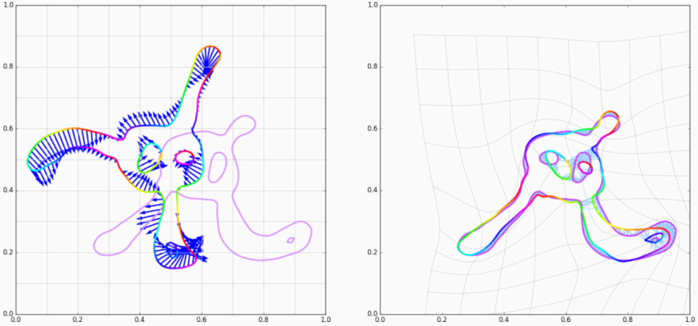


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 17: Iteration 62.

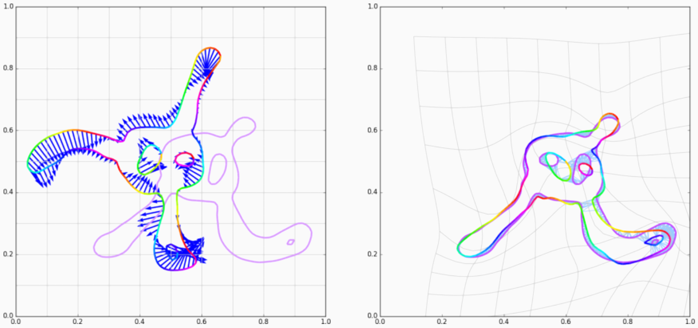
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

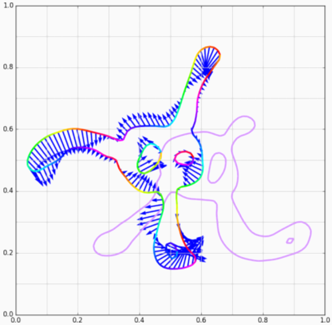
Figure 17: Iteration 64.

Typical run with OT fidelity

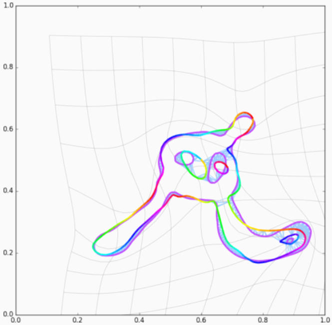


(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 65.



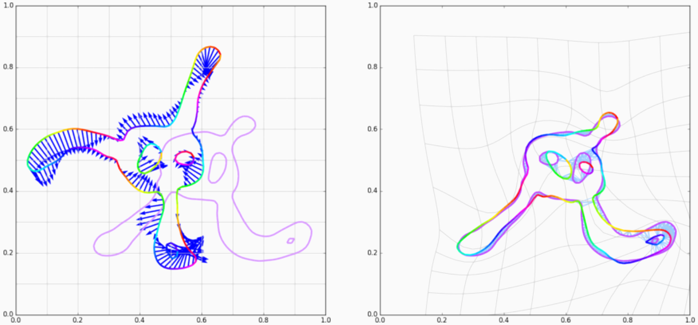
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 66.

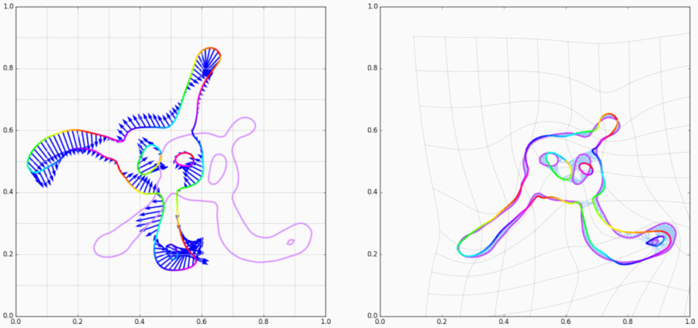
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 67.

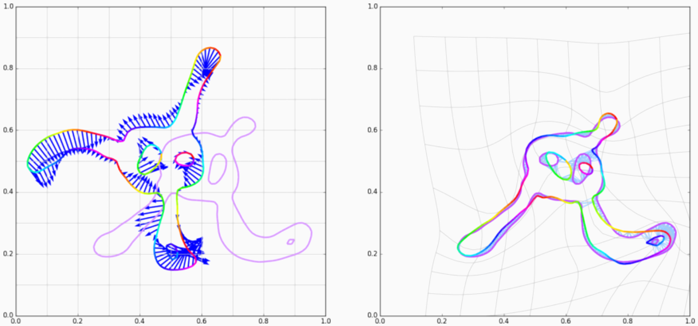
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 68.

Typical run with OT fidelity

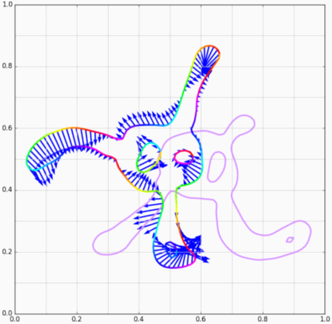


(a) Momentum p_0.

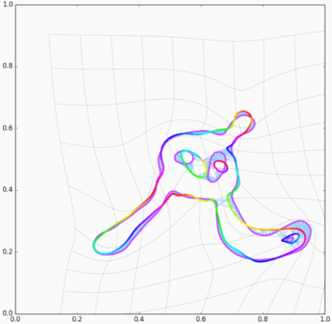
(b) Shooted model q_1.

Figure 17: Iteration 69.

Typical run with OT fidelity



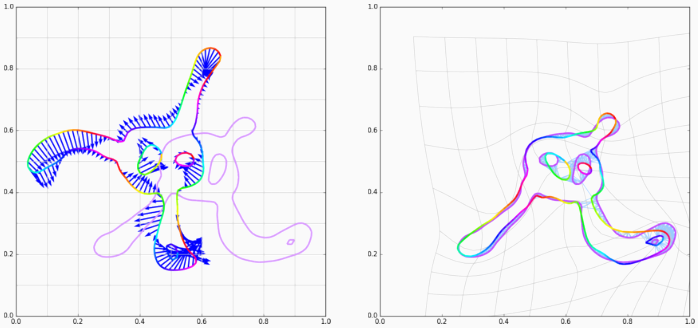
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 70.

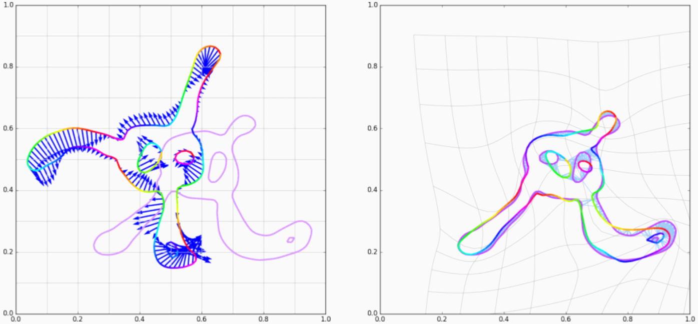
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 71.

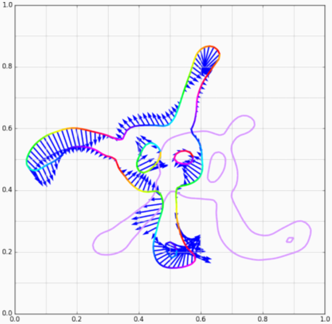
Typical run with OT fidelity



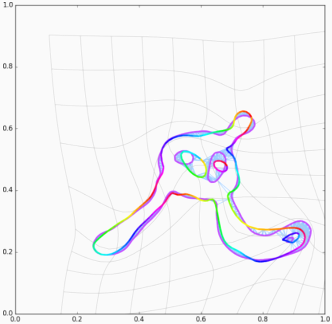
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 72.

Typical run with OT fidelity



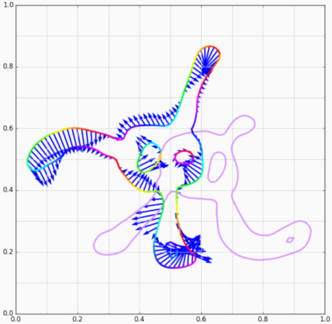
(a) Momentum p_0.



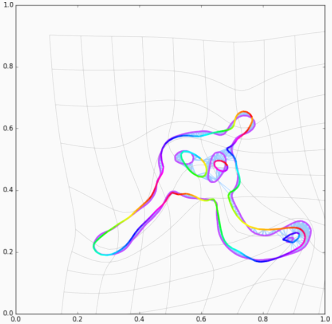
(b) Shooted model q_1.

Figure 17: Iteration 73.

Typical run with OT fidelity

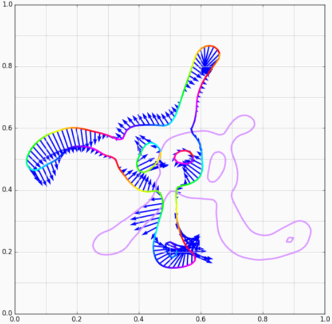


(a) Momentum p_0.

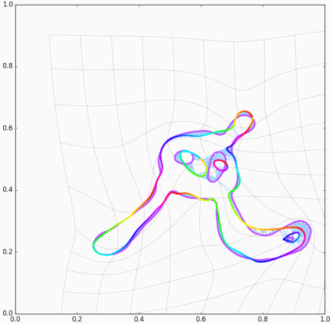


(b) Shooted model q_1.

Figure 17: Iteration 74.



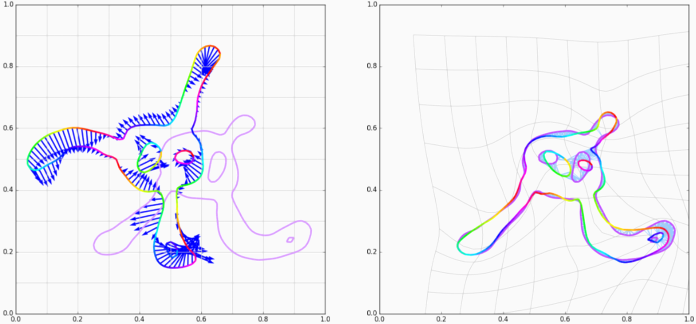
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 75.

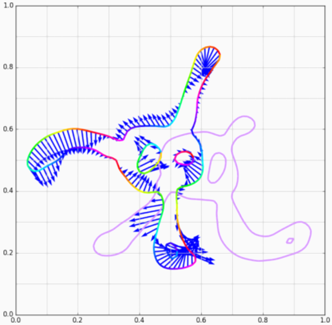
Typical run with OT fidelity



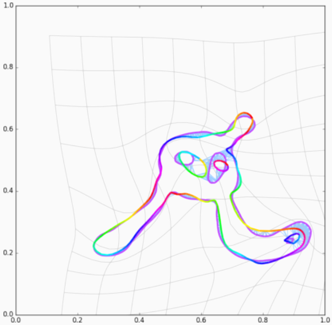
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 77.

Typical run with OT fidelity



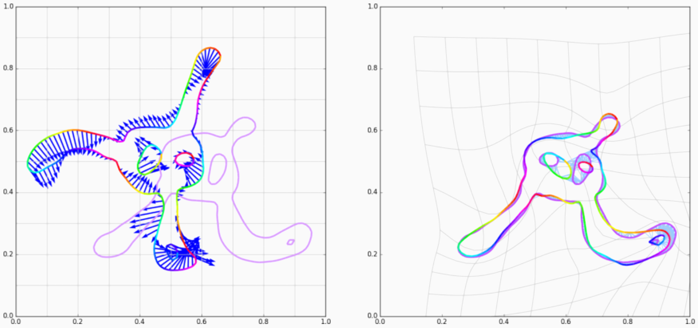
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 78.

Typical run with OT fidelity

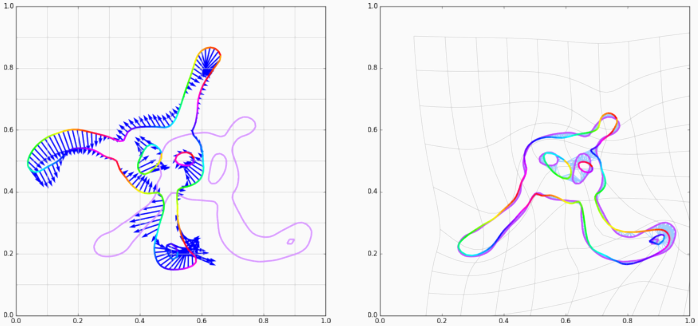


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 17: Iteration 79.

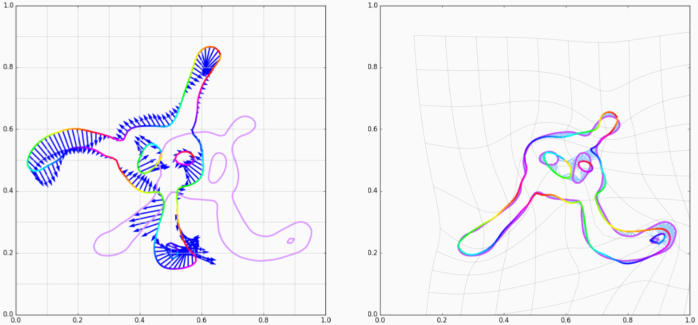
Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 80.

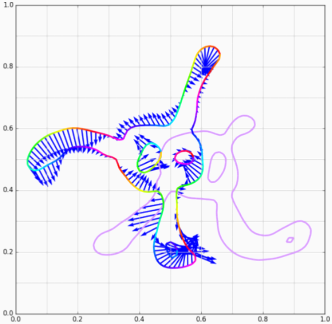
Typical run with OT fidelity



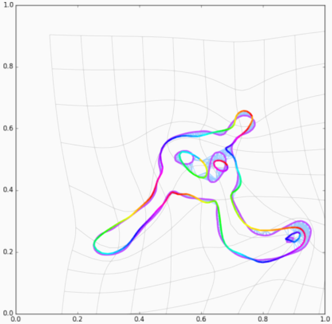
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 81.

Typical run with OT fidelity



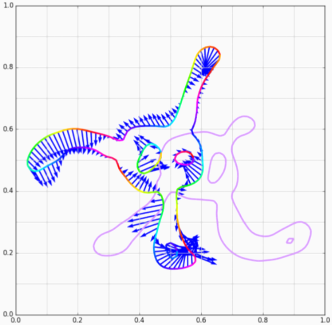
(a) Momentum p_0.



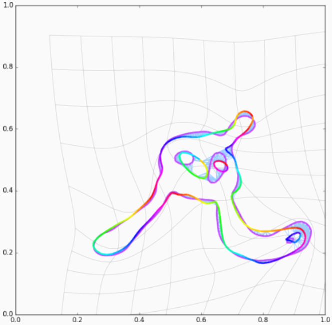
(b) Shooted model q_1.

Figure 17: Iteration 82.

Typical run with OT fidelity



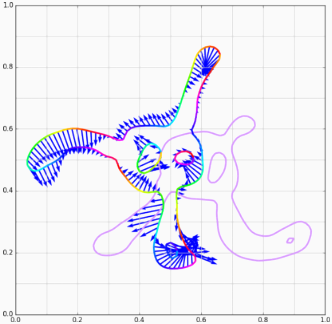
(a) Momentum p_0.



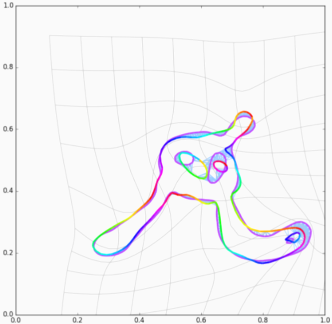
(b) Shooted model q_1.

Figure 17: Iteration 83.

Typical run with OT fidelity



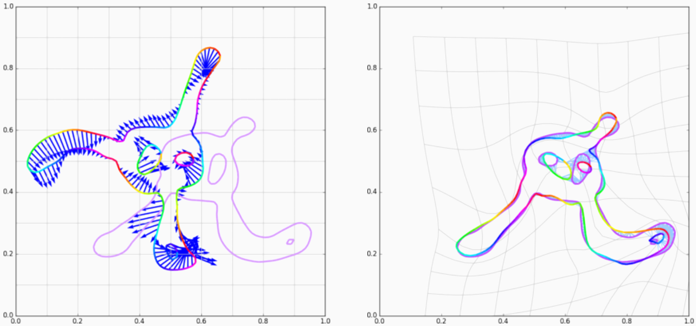
(a) Momentum p_0.



(b) Shooted model q_1.

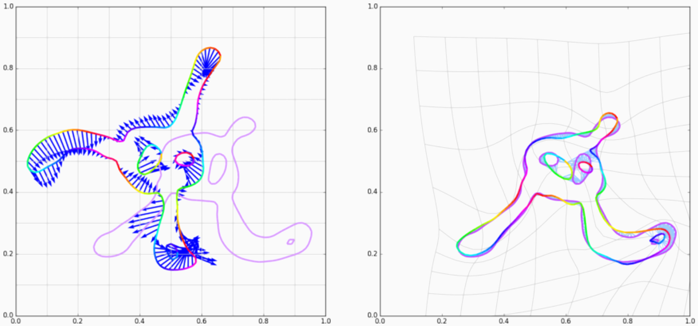
Figure 17: Iteration 85.

Typical run with OT fidelity



(a) Momentum p_0. (b) Shooted model q_1.

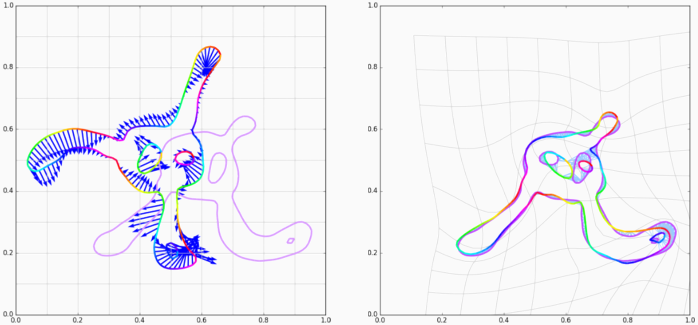
Figure 17: Iteration 86.



(a) Momentum p_0.

(b) Shooted model q_1.

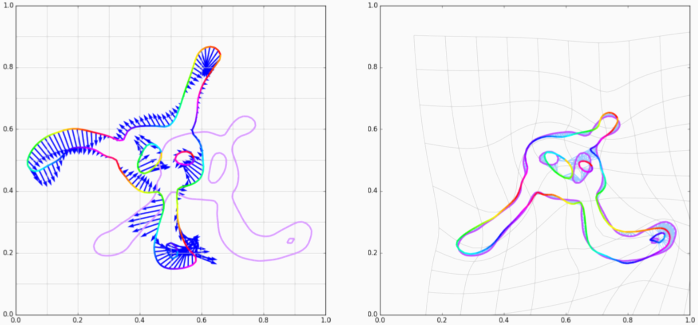
Figure 17: Iteration 87.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 88.

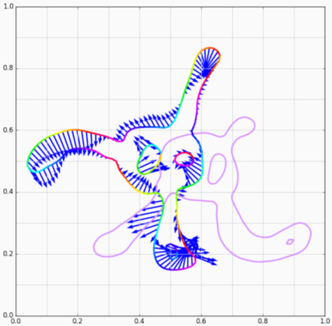
Typical run with OT fidelity



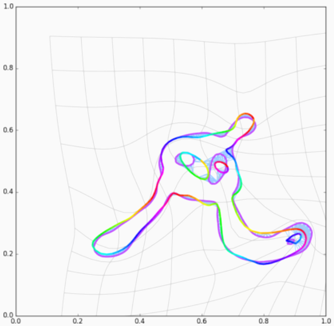
(a) Momentum p_0. (b) Shooted model q_1.

Figure 17: Iteration 89.

Typical run with OT fidelity



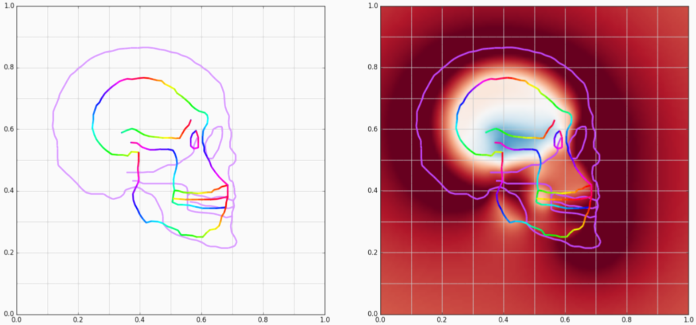
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 17: Iteration 90.

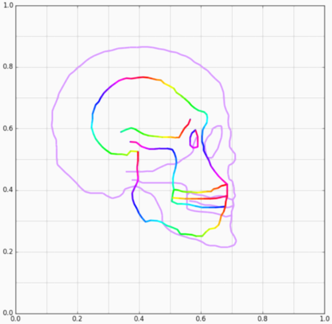
Typical run with kernel fidelity



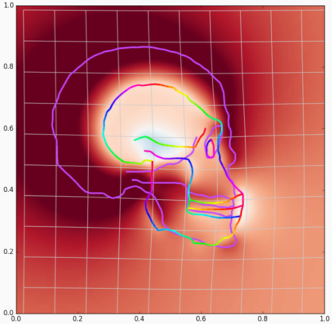
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 18: Iteration 0.

Typical run with kernel fidelity



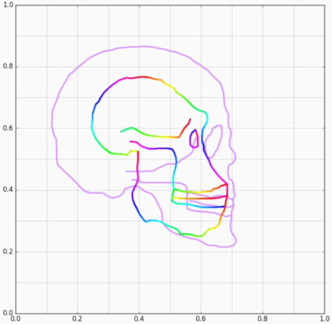
(a) Momentum p_0.



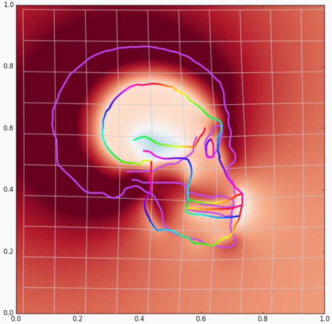
(b) Shooted model q_1.

Figure 18: Iteration 3.

Typical run with kernel fidelity



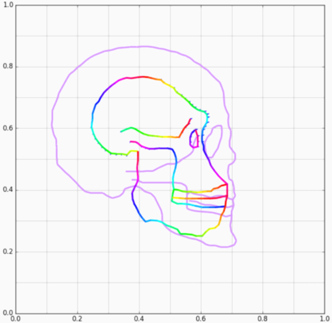
(a) Momentum p_0.



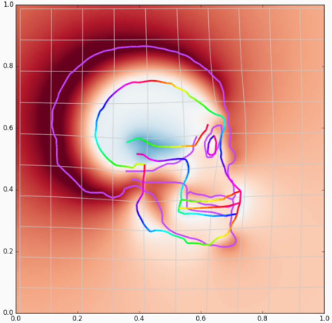
(b) Shooted model q_1.

Figure 18: Iteration 4.

Typical run with kernel fidelity



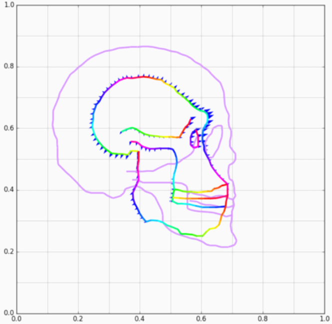
(a) Momentum p_0.



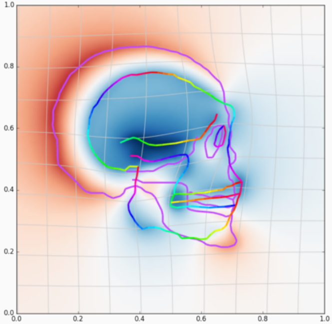
(b) Shooted model q_1.

Figure 18: Iteration 5.

Typical run with kernel fidelity



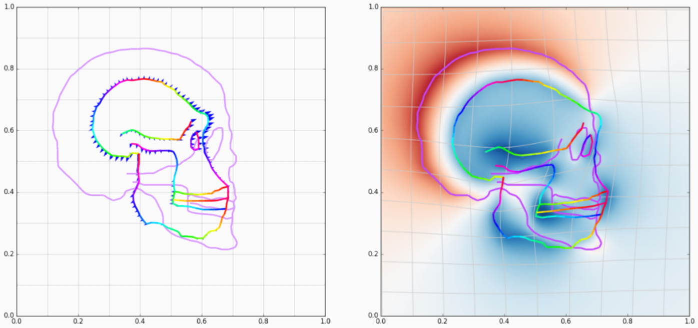
(a) Momentum p_0.



(b) Shooted model q_1.

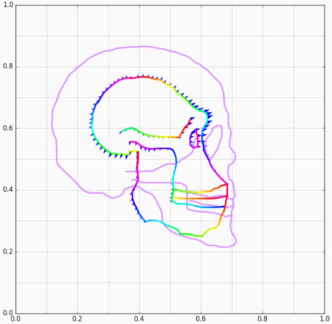
Figure 18: Iteration 6.

Typical run with kernel fidelity

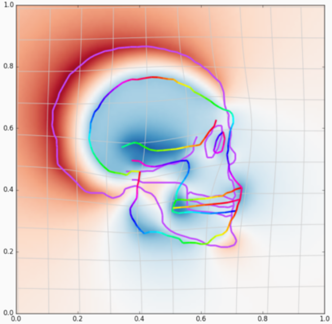


(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 7.



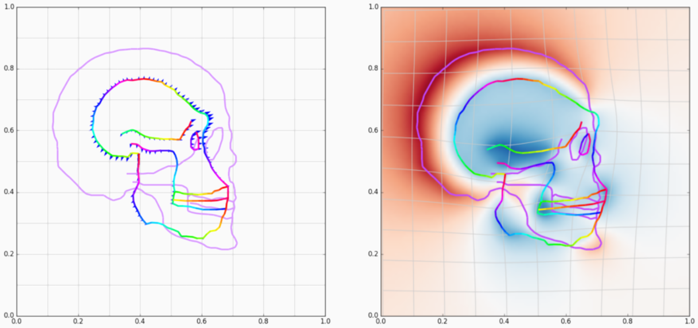
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 18: Iteration 8.

Typical run with kernel fidelity

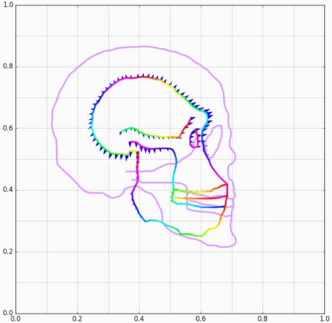


(a) Momentum p_0.

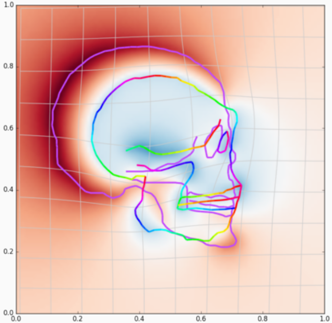
(b) Shooted model q_1.

Figure 18: Iteration 9.

Typical run with kernel fidelity



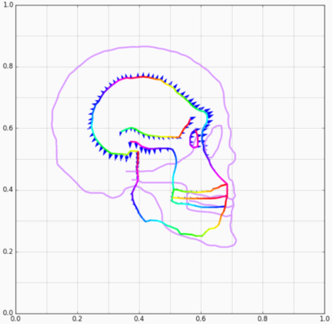
(a) Momentum p_0.



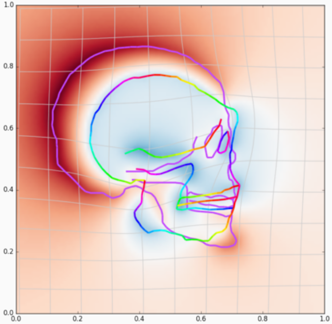
(b) Shooted model q_1.

Figure 18: Iteration 10.

Typical run with kernel fidelity

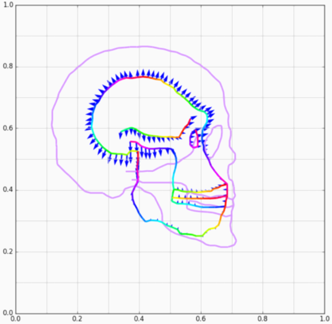


(a) Momentum p_0.

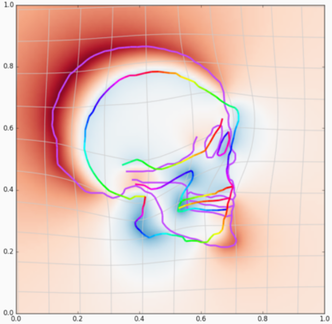


(b) Shooted model q_1.

Figure 18: Iteration 11.



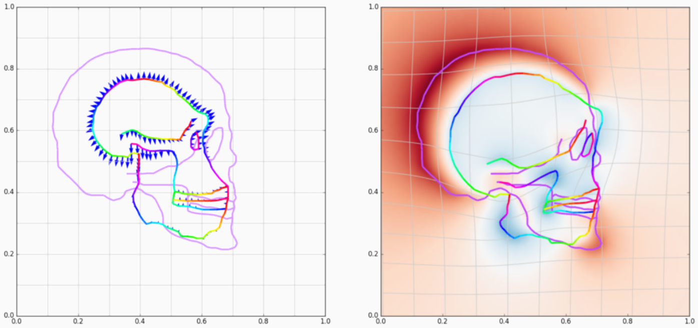
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 18: Iteration 12.

Typical run with kernel fidelity

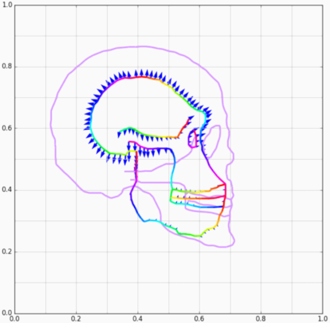


(a) Momentum p_0.

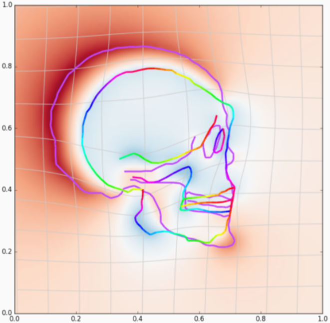
(b) Shooted model q_1.

Figure 18: Iteration 13.

Typical run with kernel fidelity



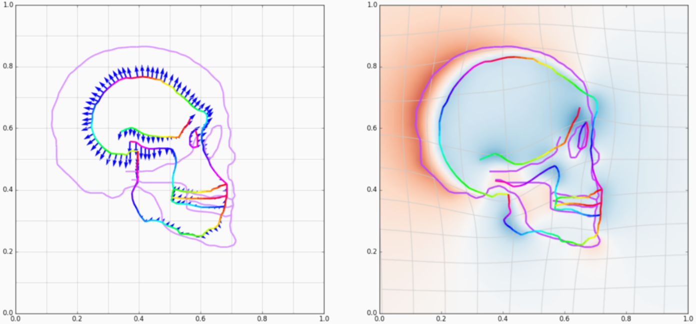
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 18: Iteration 14.

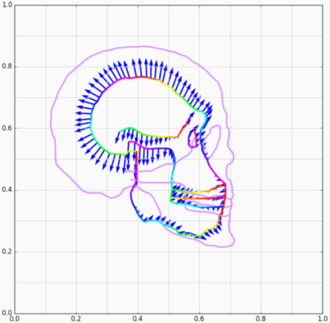
Typical run with kernel fidelity



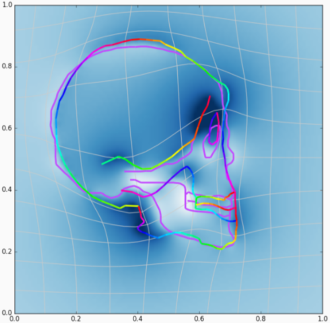
(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 15.

Typical run with kernel fidelity



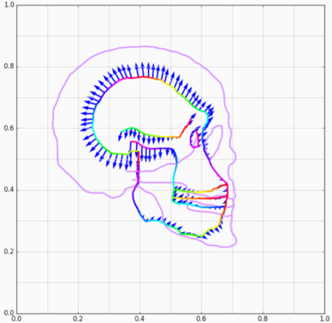
(a) Momentum p_0.



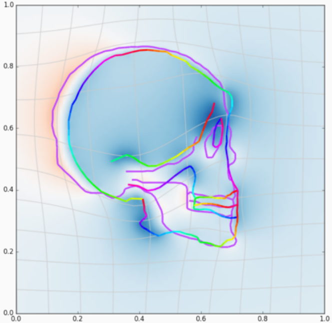
(b) Shooted model q_1.

Figure 18: Iteration 16.

Typical run with kernel fidelity

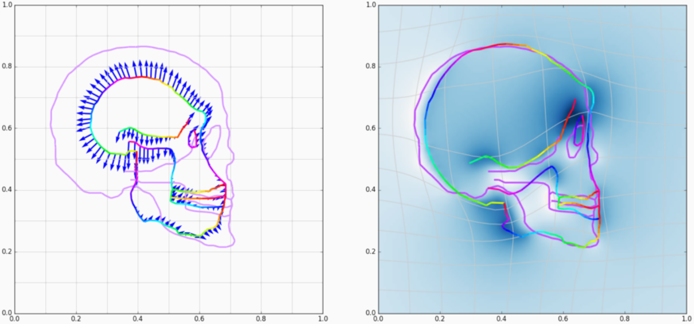


(a) Momentum p_0.



(b) Shooted model q_1.

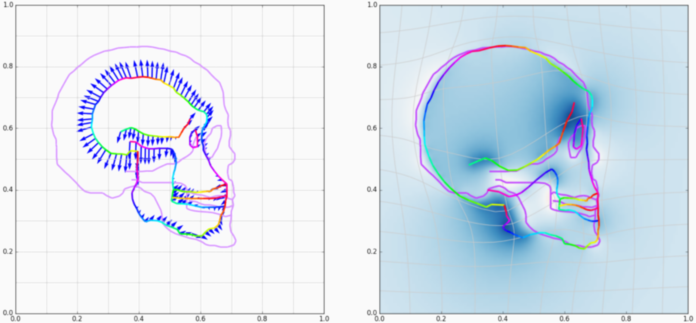
Figure 18: Iteration 17.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 19.

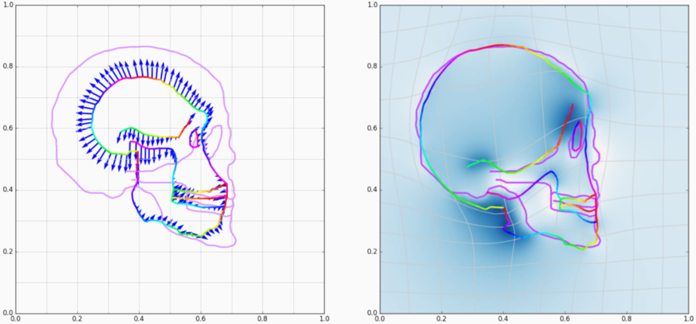
Typical run with kernel fidelity



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 18: Iteration 20.

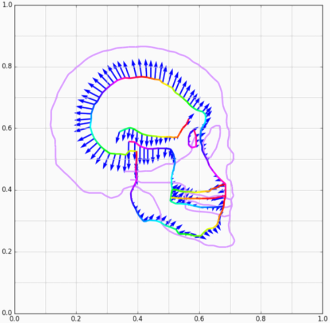
Typical run with kernel fidelity



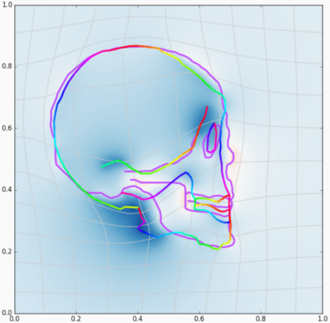
(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 21.

Typical run with kernel fidelity



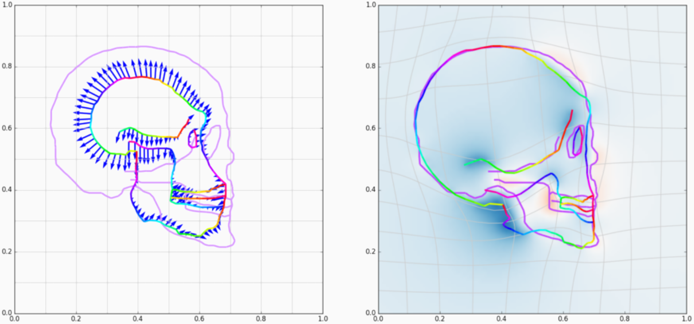
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 18: Iteration 22.

Typical run with kernel fidelity

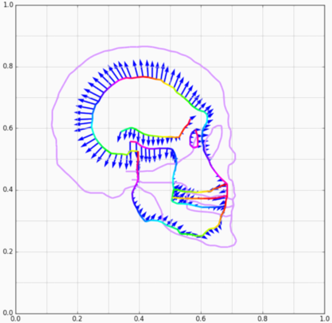


(a) Momentum p_0.

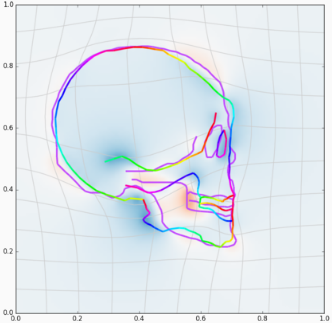
(b) Shooted model q_1.

Figure 18: Iteration 23.

Typical run with kernel fidelity



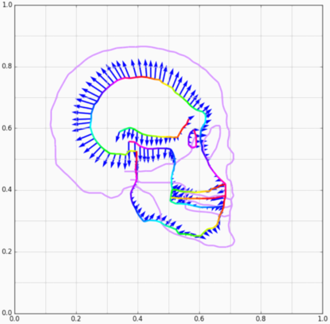
(a) Momentum p_0.



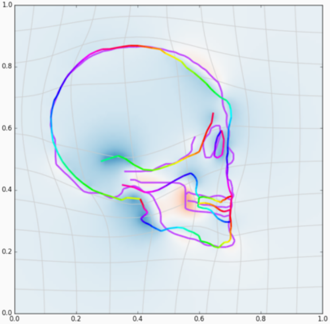
(b) Shooted model q_1.

Figure 18: Iteration 24.

Typical run with kernel fidelity



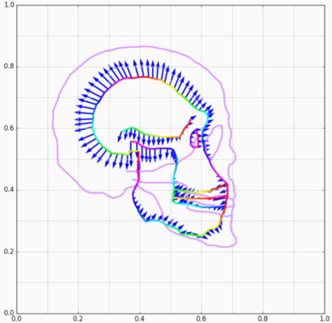
(a) Momentum p_0.



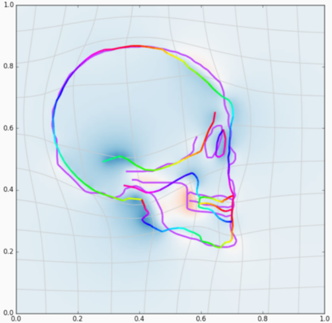
(b) Shooted model q_1.

Figure 18: Iteration 25.

Typical run with kernel fidelity



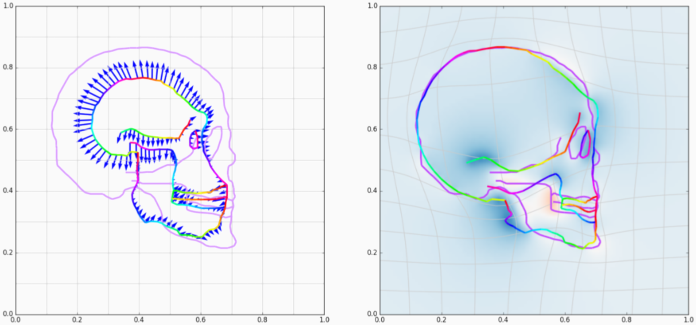
(a) Momentum p_0.



(b) Shooted model q_1.

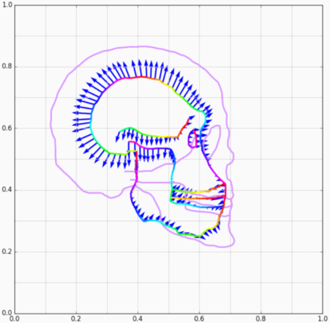
Figure 18: Iteration 26.

Typical run with kernel fidelity

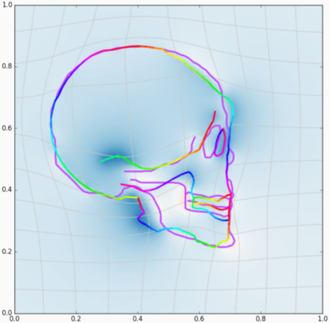


(a) Momentum p_0.   (b) Shooted model q_1.

Figure 18: Iteration 27.

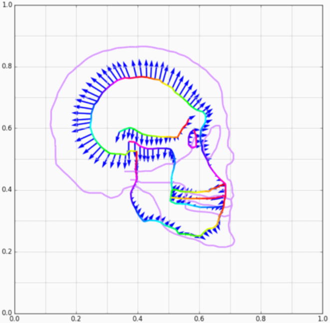


(a) Momentum p_0.

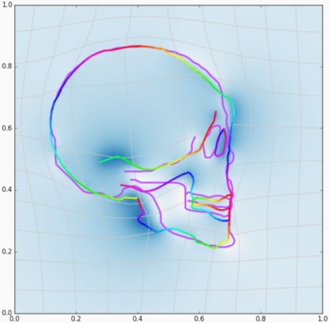


(b) Shooted model q_1.

Figure 18: Iteration 28.

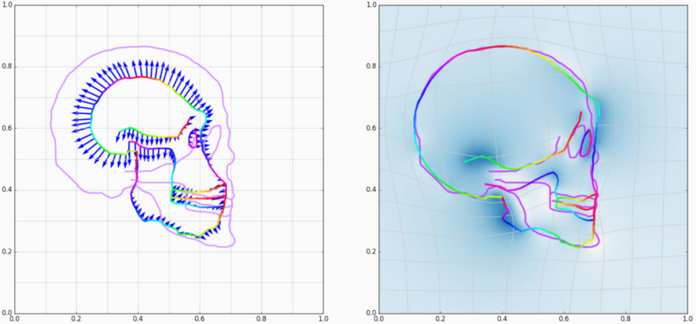


(a) Momentum p_0.



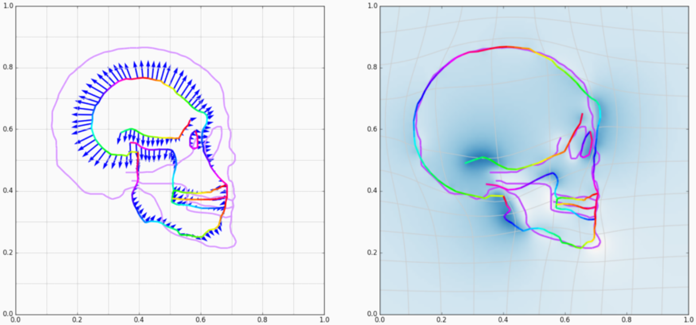
(b) Shooted model q_1.

Figure 18: Iteration 30.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 31.

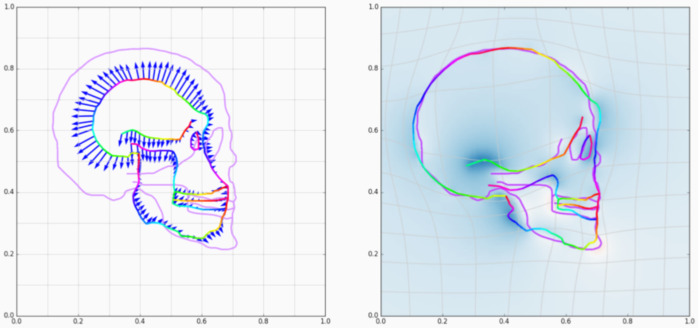


(a) Momentum p_0.

(b) Shooted model q_1.

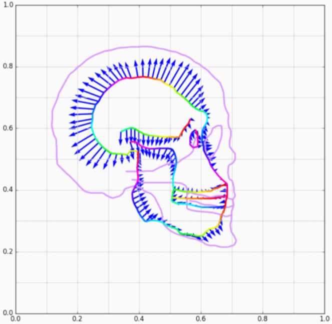
Figure 18: Iteration 32.

Typical run with kernel fidelity

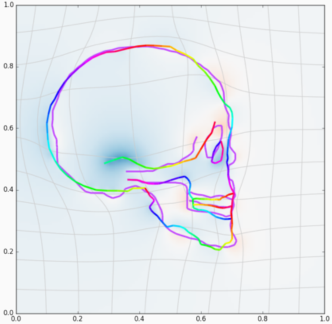


(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 33.



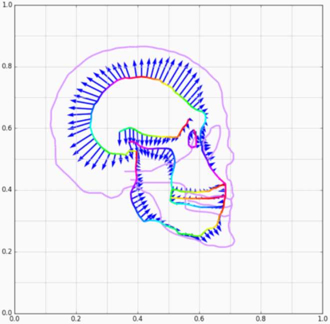
(a) Momentum p_0.



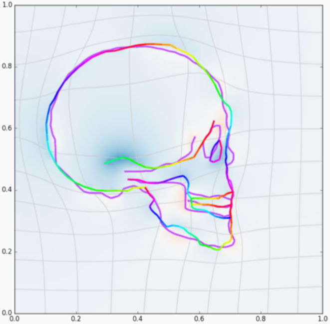
(b) Shooted model q_1.

Figure 18: Iteration 34.

Typical run with kernel fidelity



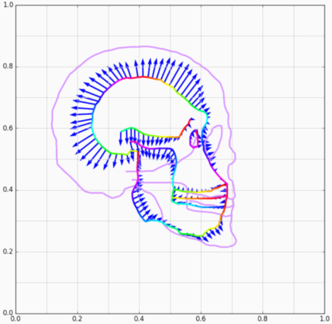
(a) Momentum p_0.



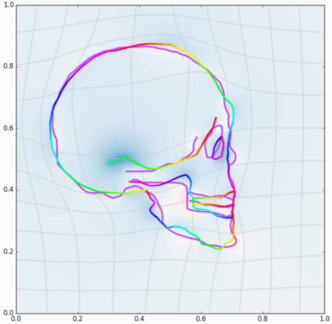
(b) Shooted model q_1.

Figure 18: Iteration 36.

Typical run with kernel fidelity

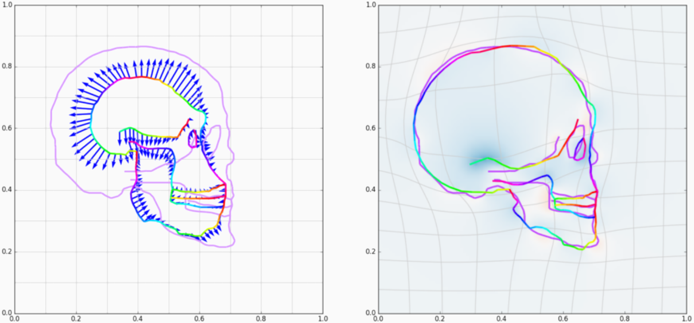


(a) Momentum p_0.



(b) Shooted model q_1.

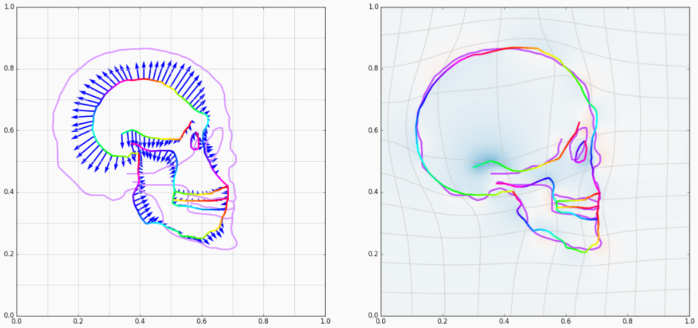
Figure 18: Iteration 37.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 38.

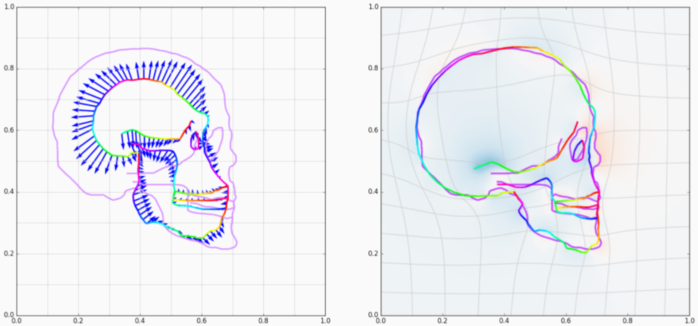
Typical run with kernel fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 39.

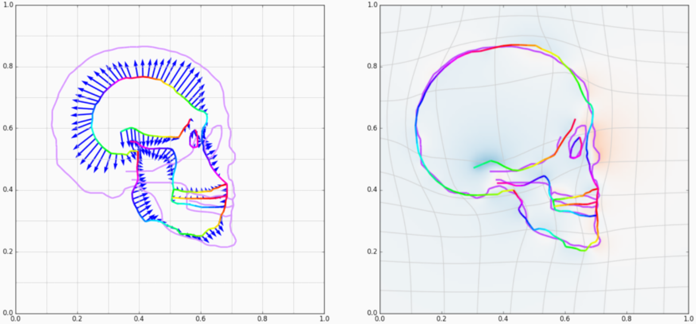
Typical run with kernel fidelity



(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 40.

Typical run with kernel fidelity

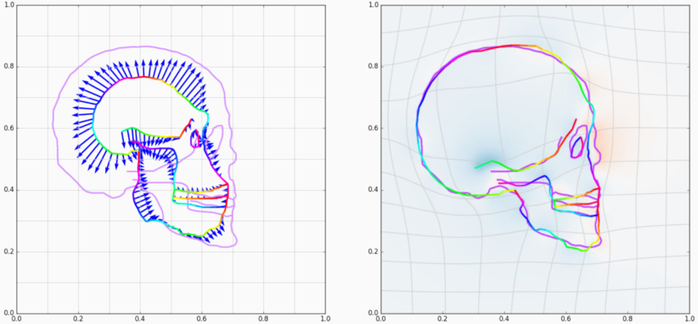


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 18: Iteration 41.

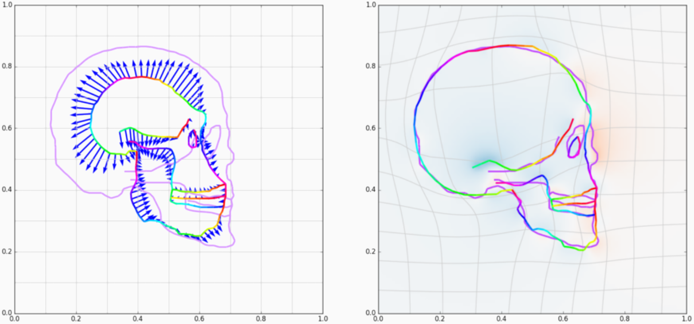
Typical run with kernel fidelity



(a) Momentum p_0. **(b)** Shooted model q_1.

Figure 18: Iteration 42.

Typical run with kernel fidelity

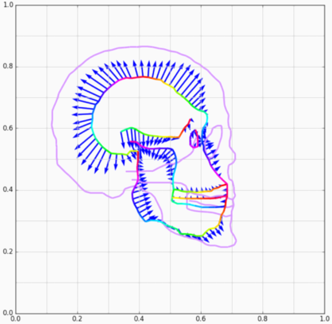


(a) Momentum p_0.

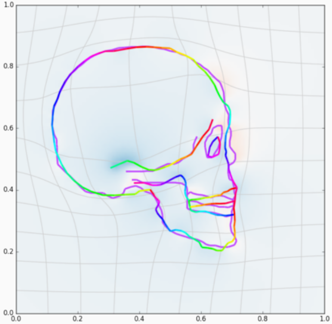
(b) Shooted model q_1.

Figure 18: Iteration 44.

Typical run with kernel fidelity



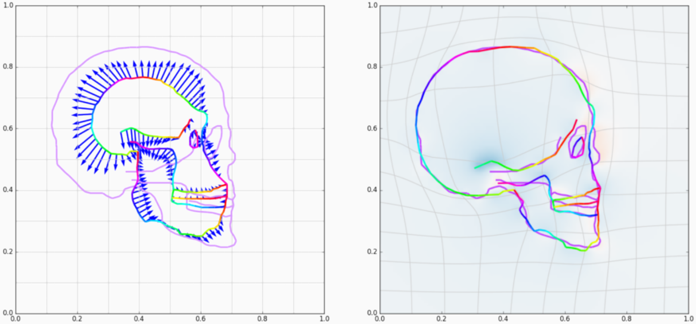
(a) Momentum p_0.



(b) Shooted model q_1.

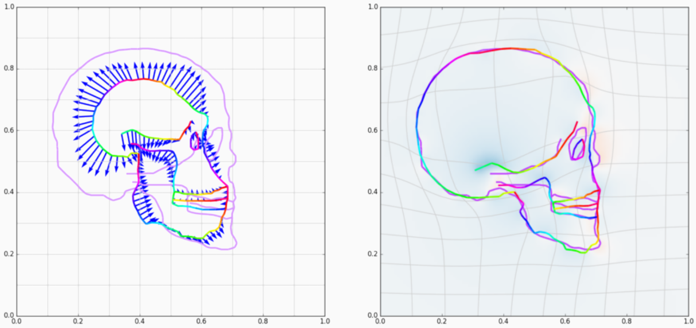
Figure 18: Iteration 45.

Typical run with kernel fidelity



(a) Momentum p_0. (b) Shooted model q_1.

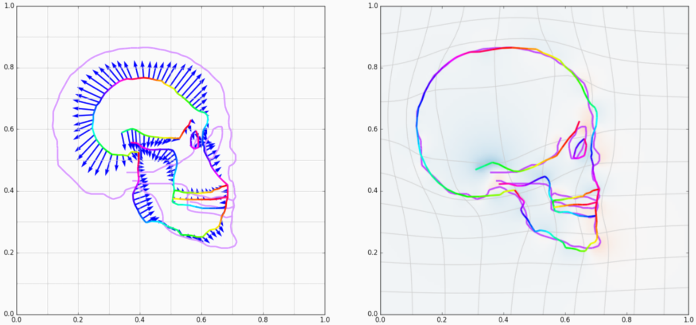
Figure 18: Iteration 46.



(a) Momentum p_0. (b) Shooted model q_1.

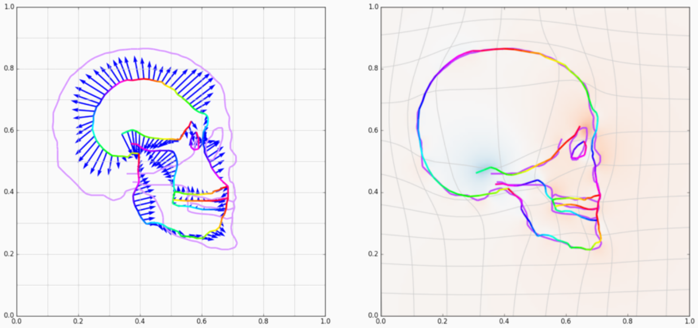
Figure 18: Iteration 47.

Typical run with kernel fidelity



(a) Momentum p_0. (b) Shooted model q_1.

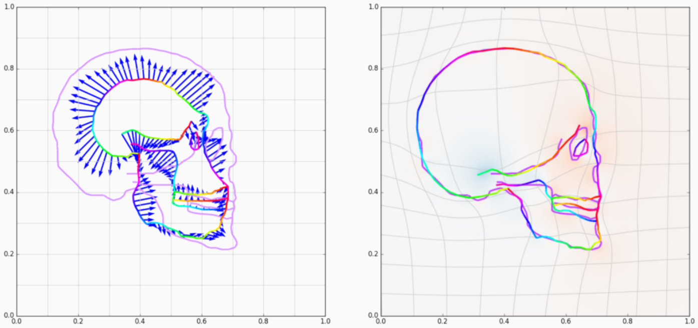
Figure 18: Iteration 50.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 70.

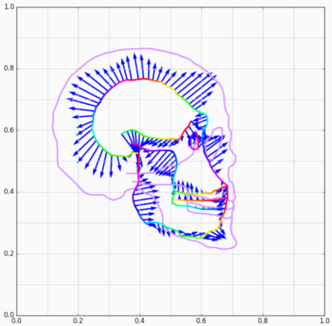
Typical run with kernel fidelity



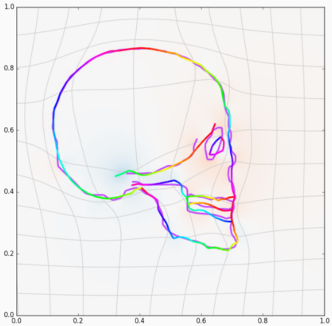
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 18: Iteration 90.

Typical run with kernel fidelity



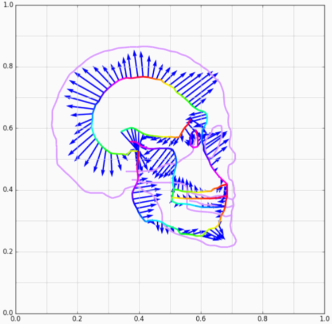
(a) Momentum p_0.



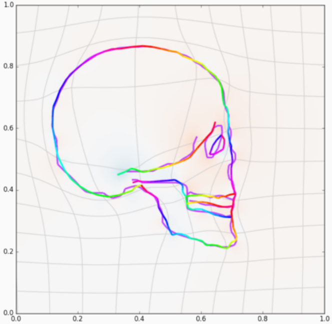
(b) Shooted model q_1.

Figure 18: Iteration 110.

Typical run with kernel fidelity



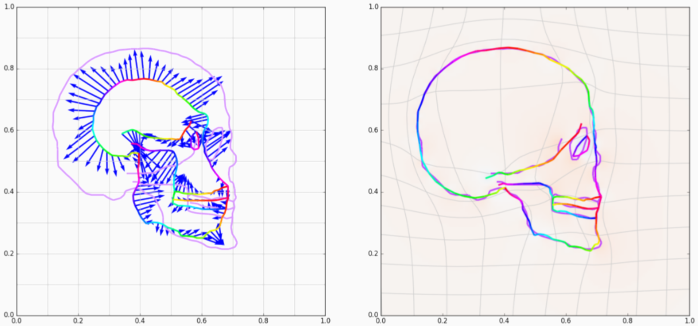
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 18: Iteration 130.

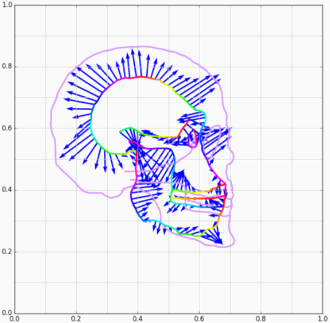
Typical run with kernel fidelity



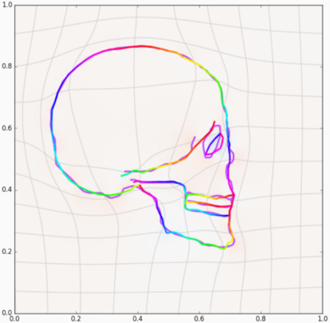
(a) Momentum p_0. (b) Shooted model q_1.

Figure 18: Iteration 150.

Typical run with kernel fidelity

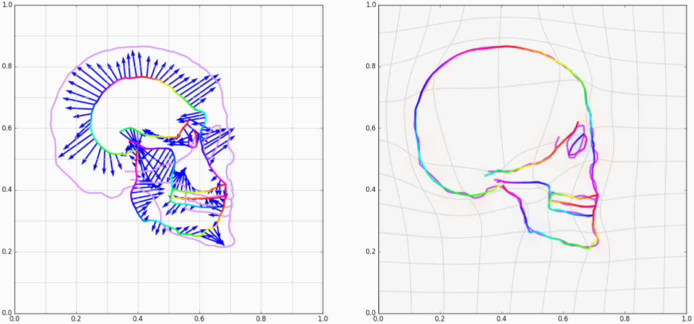


(a) Momentum p_0.



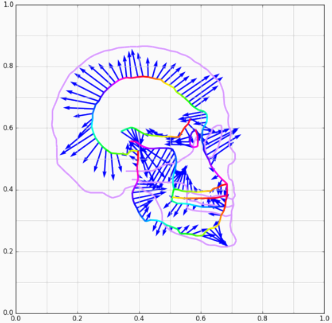
(b) Shooted model q_1.

Figure 18: Iteration 170.

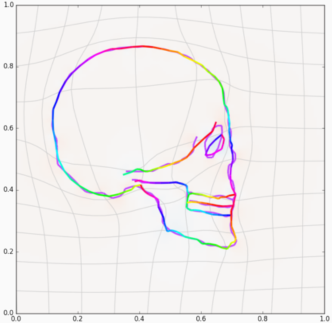


(a) Momentum p_0.   (b) Shooted model q_1.

Figure 18: Iteration 200.



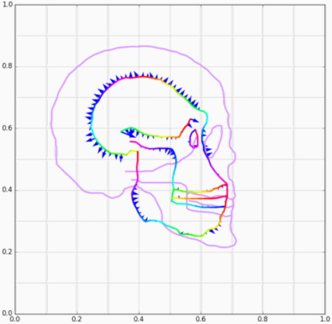
(a) Momentum p_0.



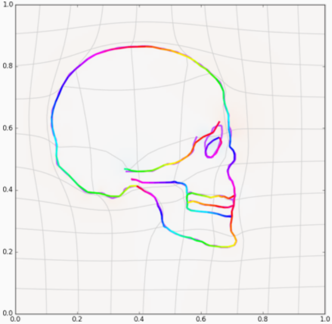
(b) Shooted model q_1.

Figure 18: Iteration 240.

Influence of the kernel width



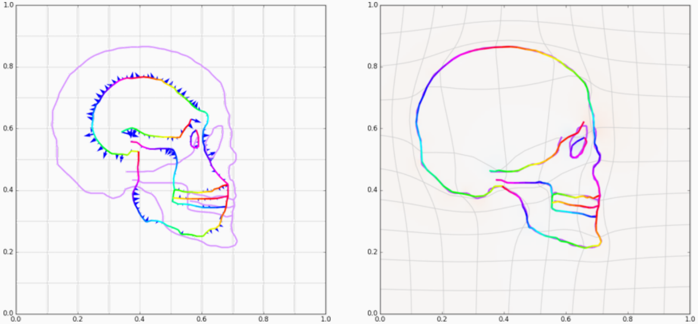
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.01}$.

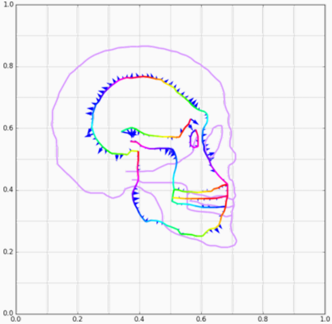
Influence of the kernel width



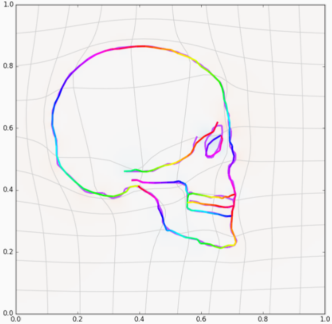
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.02}$.

Influence of the kernel width

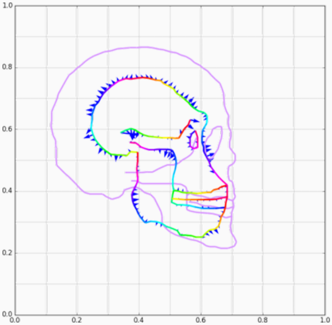


(a) Momentum p_0.

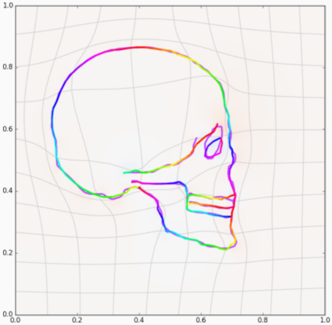


(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.03}$.



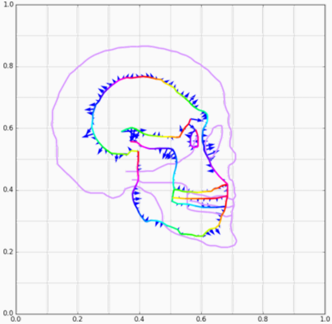
(a) Momentum p_0.



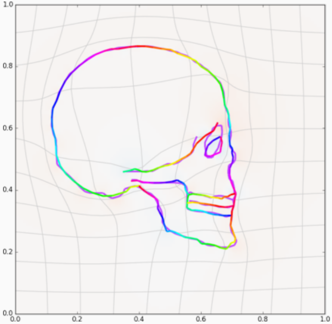
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.04}$.

Influence of the kernel width



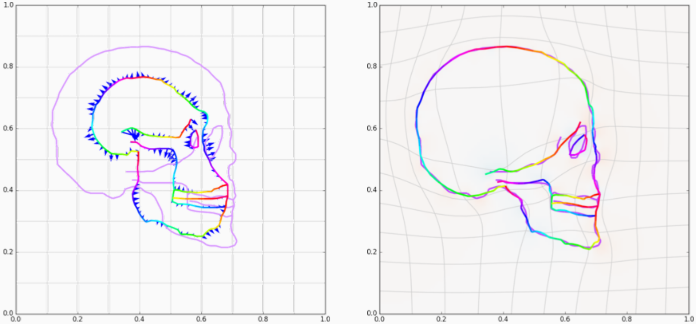
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.05}$.

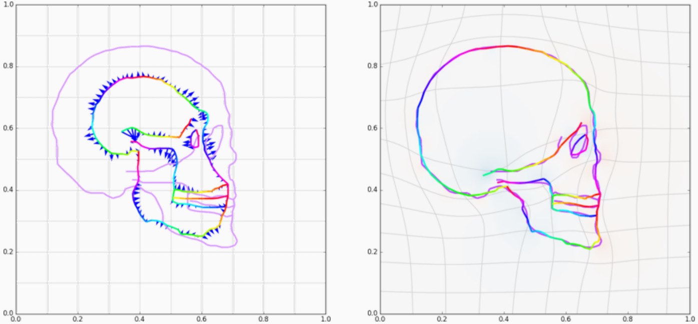
Influence of the kernel width



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.06}$.

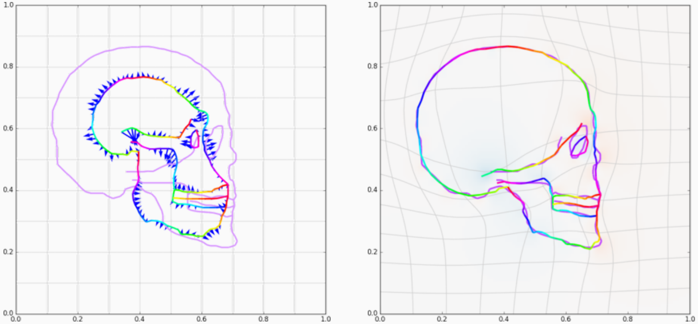
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.07}$.

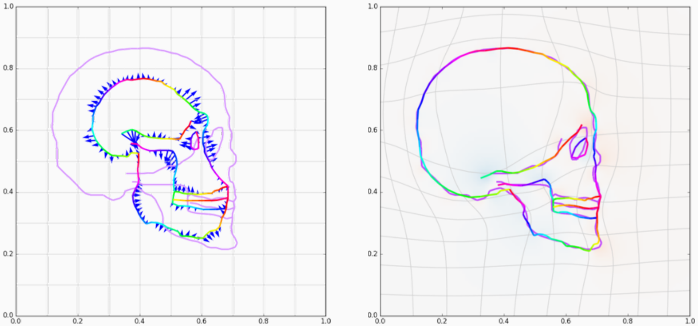
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.08}$.

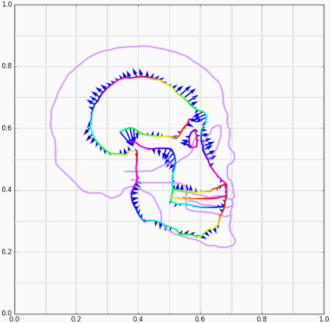
Influence of the kernel width



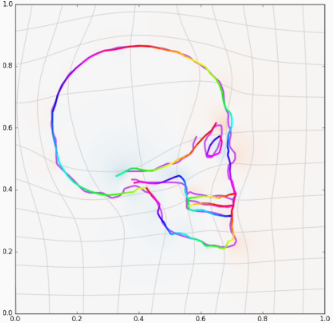
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \mathtt{.09}$.

Influence of the kernel width



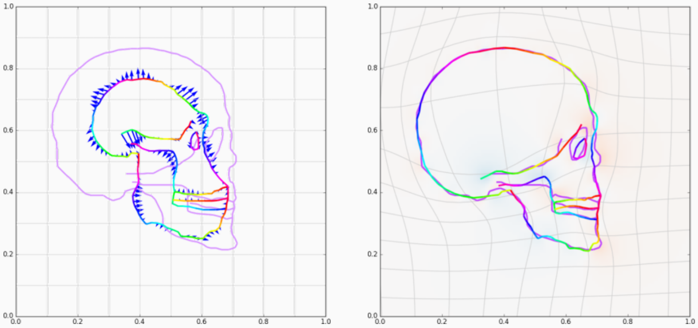
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .1$.

Influence of the kernel width

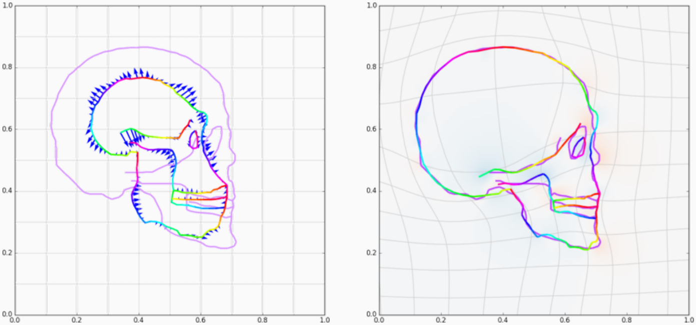


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .11$.

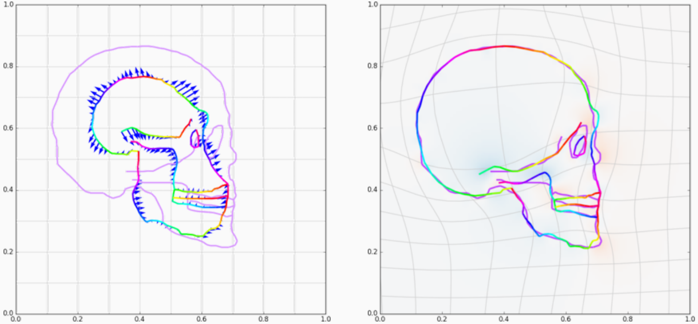
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

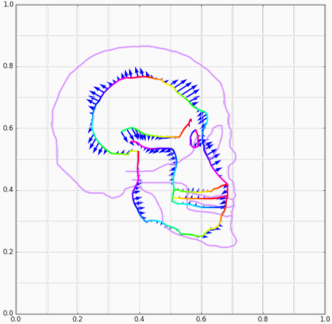
Figure 19: Final matching, $\sigma = .12$.

Influence of the kernel width

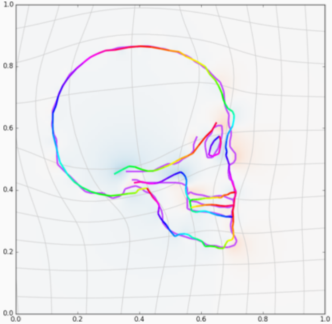


(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .13$.



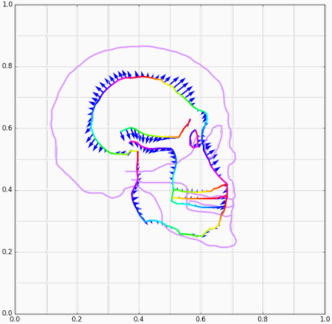
(a) Momentum p_0.



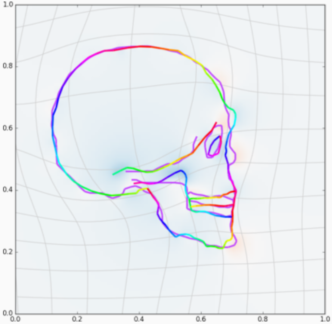
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .14$.

Influence of the kernel width

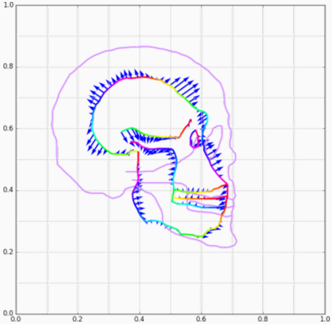


(a) Momentum p_0.

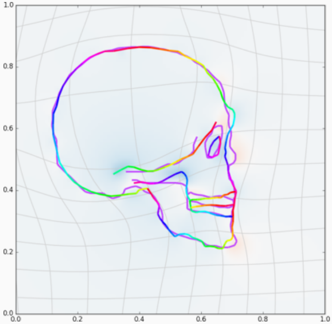


(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .15$.



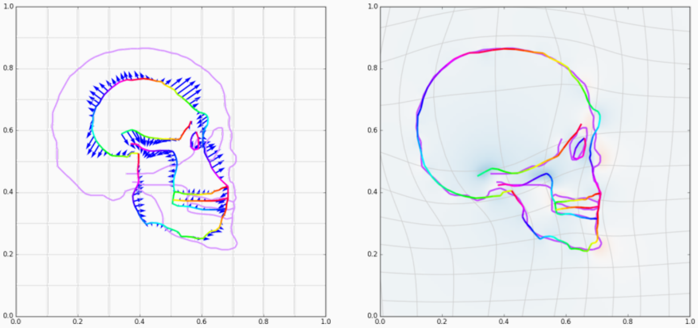
(a) Momentum p_0.



(b) Shooted model q_1.

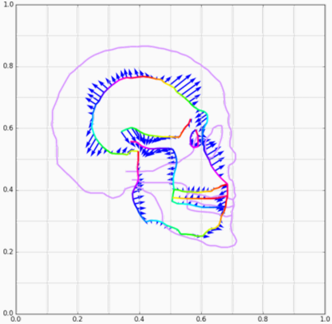
Figure 19: Final matching, $\sigma = .16$.

Influence of the kernel width

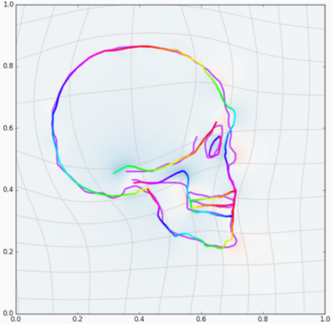


(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .17$.

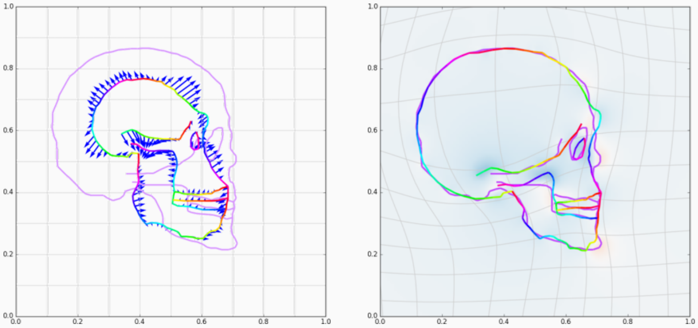


(a) Momentum p_0.



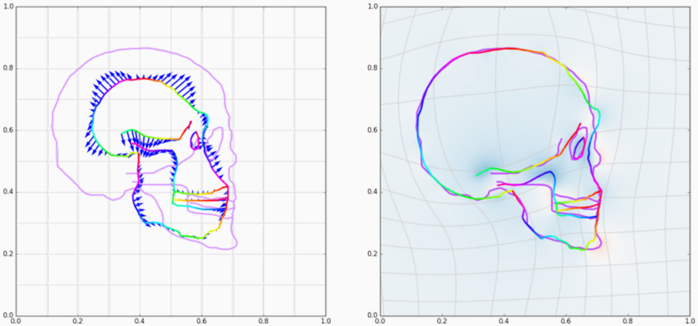
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .18$.



(a) Momentum p_0. **(b)** Shooted model q_1.

Figure 19: Final matching, $\sigma = .19$.

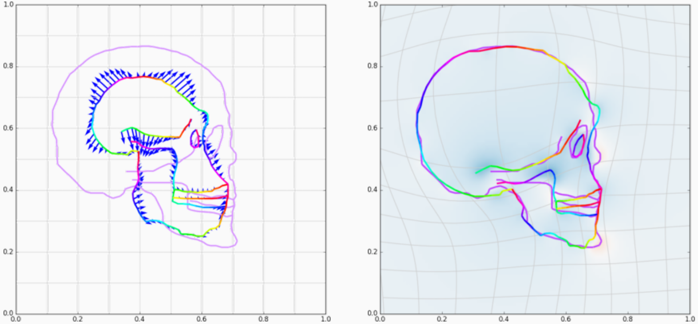


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .2$.

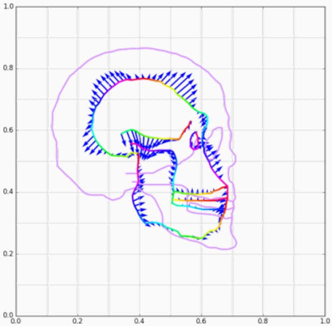
Influence of the kernel width



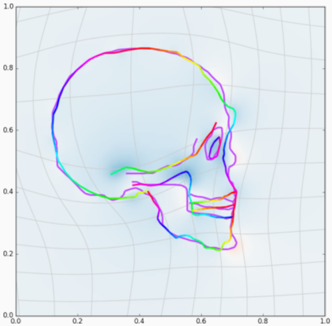
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .21$.

Influence of the kernel width



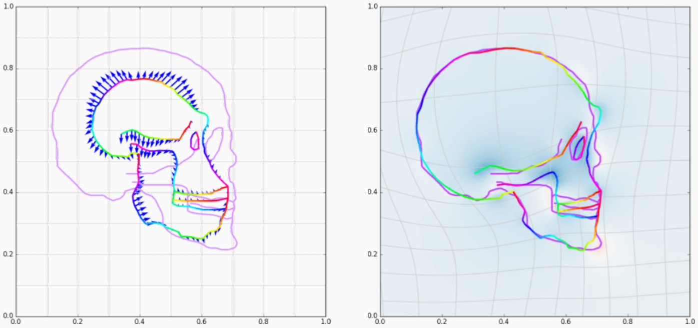
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .22$.

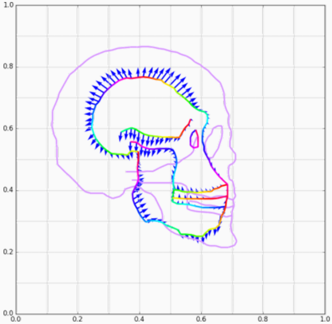
Influence of the kernel width



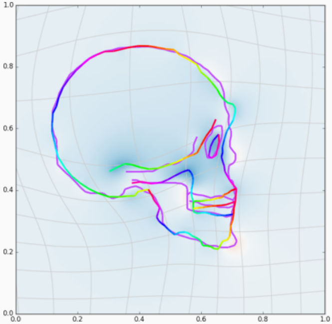
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .23$.

Influence of the kernel width



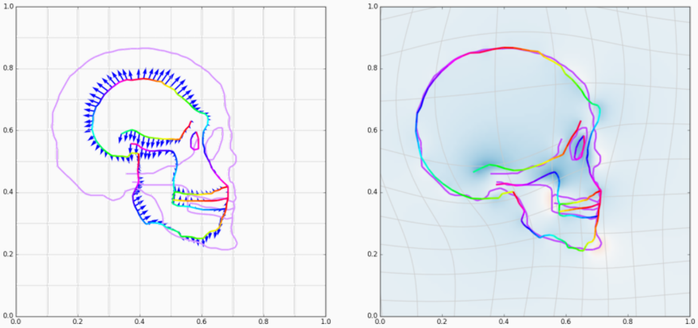
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .24$.

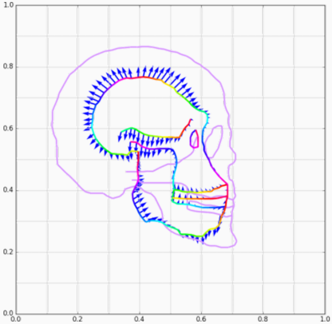
Influence of the kernel width



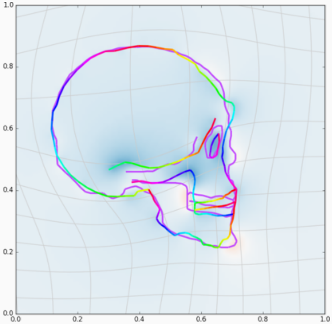
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .25$.

Influence of the kernel width

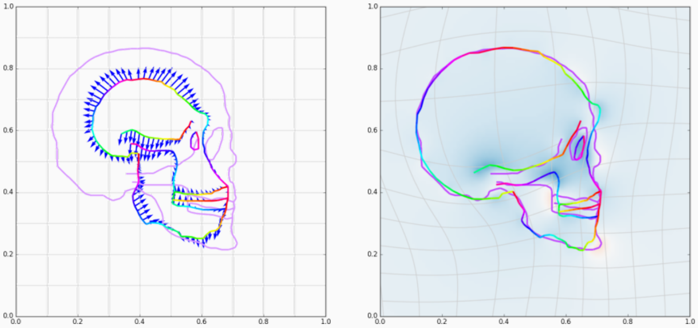


(a) Momentum p_0.



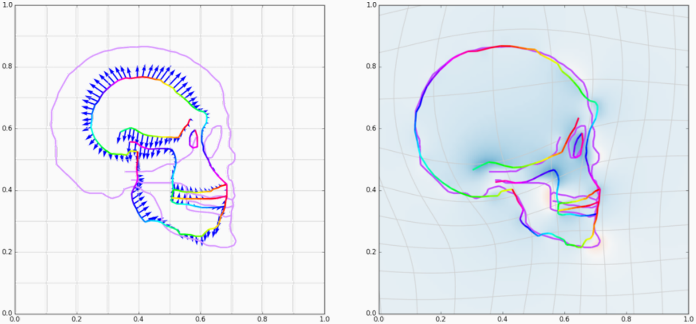
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .26$.



(a) Momentum p_0. (b) Shooted model q_1.

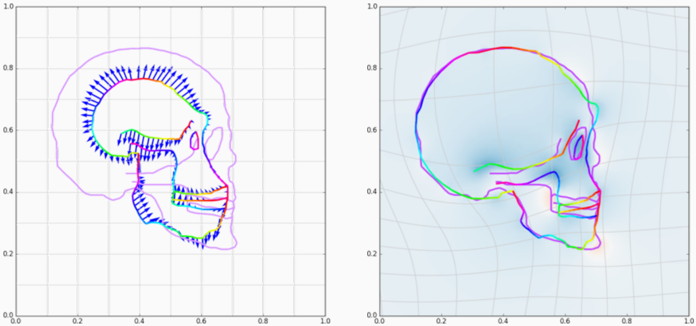
Figure 19: Final matching, $\sigma = .27$.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .28$.

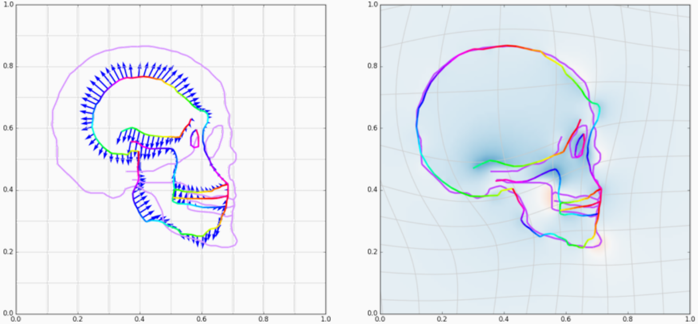
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .29$.

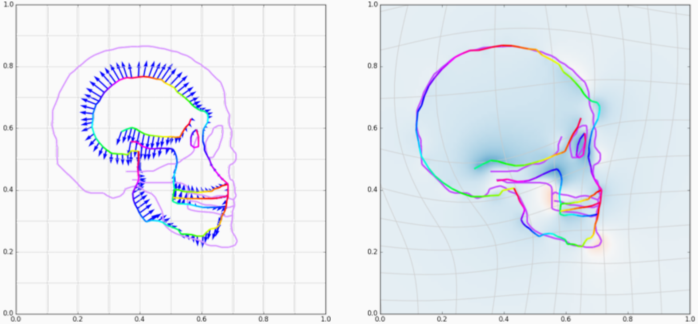
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .3$.

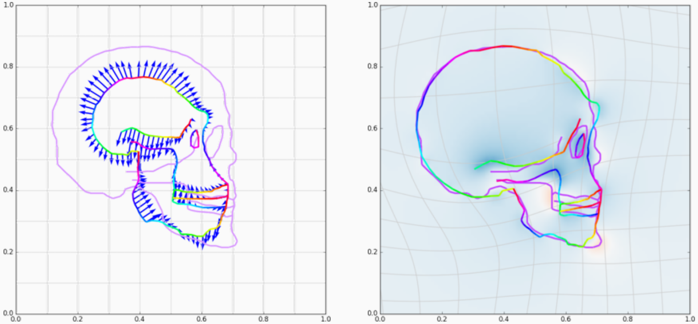
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .31$.

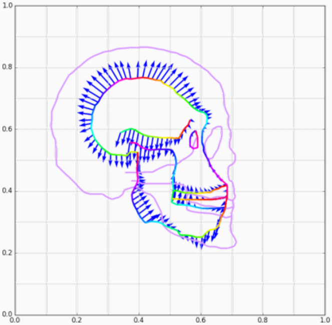
Influence of the kernel width



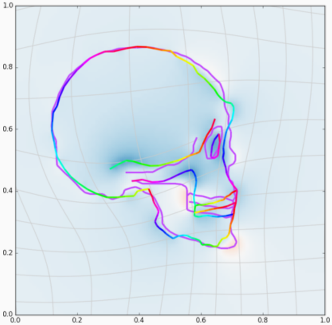
(a) Momentum p_0.　(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .32$.

Influence of the kernel width



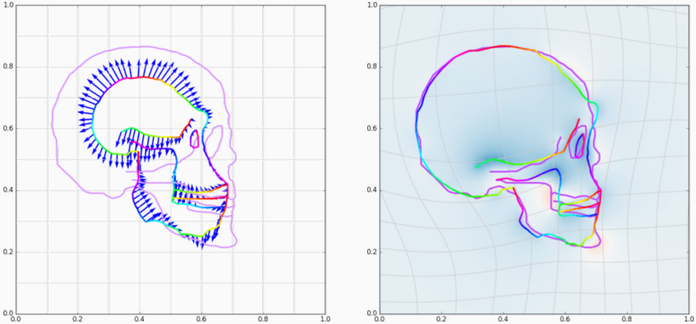
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .33$.

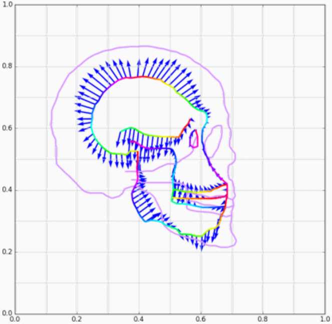
Influence of the kernel width



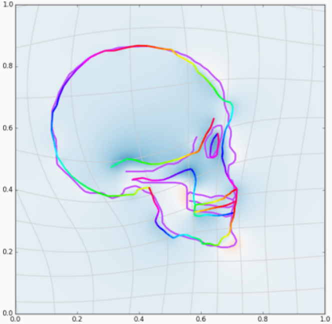
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .34$.

Influence of the kernel width



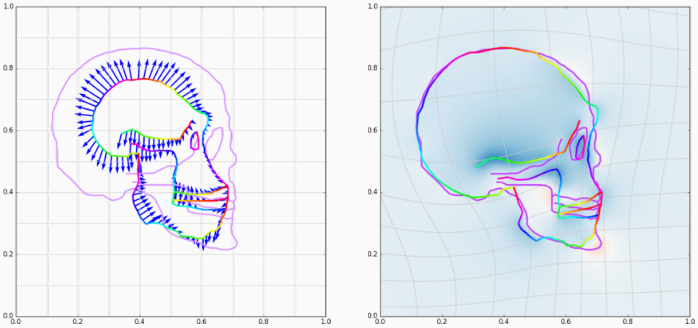
(a) Momentum p_0.



(b) Shooted model q_1.

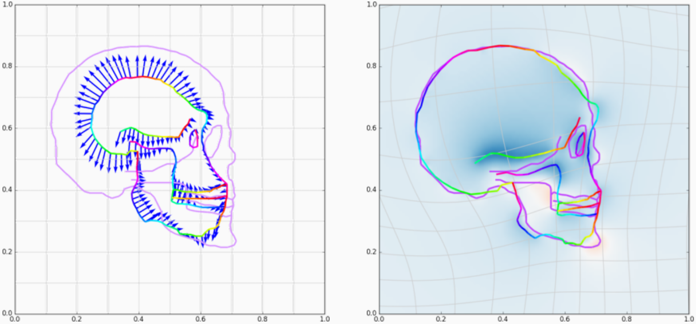
Figure 19: Final matching, $\sigma = .35$.

Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

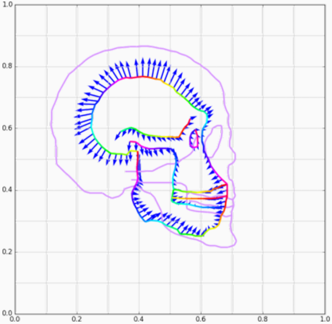
Figure 19: Final matching, $\sigma = .36$.



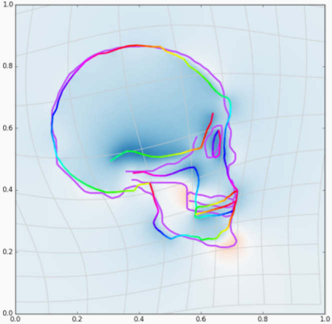
(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .37$.

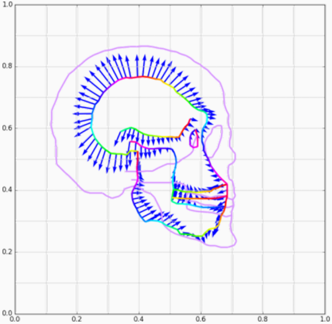


(a) Momentum p_0.

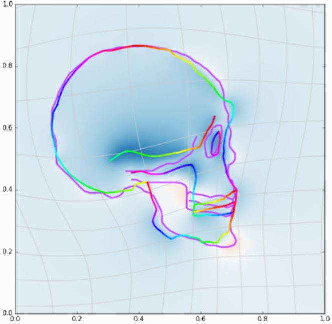


(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .38$.



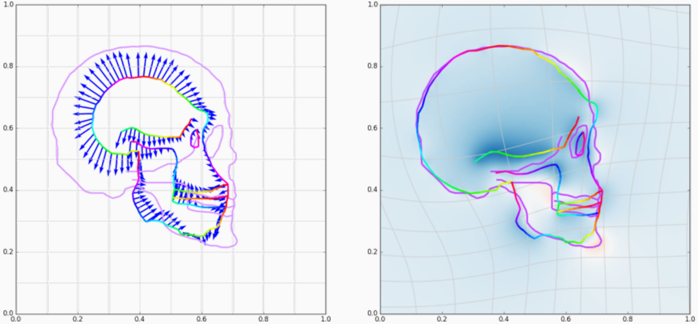
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .39$.

Influence of the kernel width

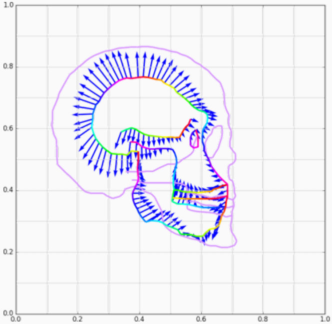


(a) Momentum p_0.

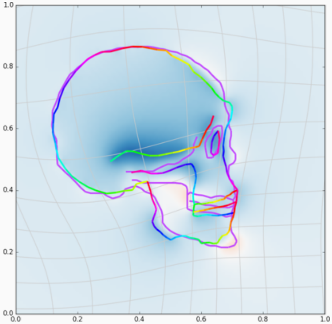
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .4$.

Influence of the kernel width



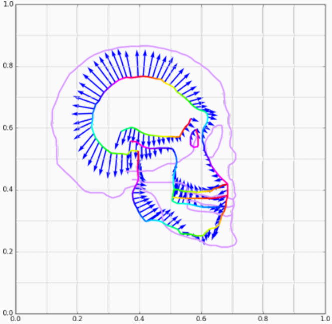
(a) Momentum p_0.



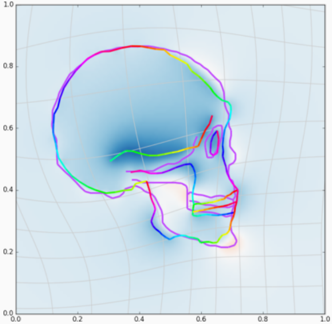
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .41$.

Influence of the kernel width

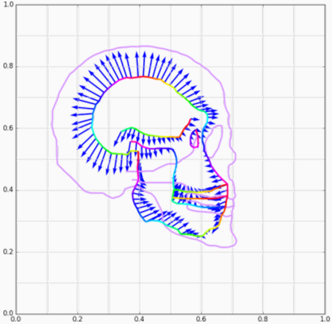


(a) Momentum p_0.

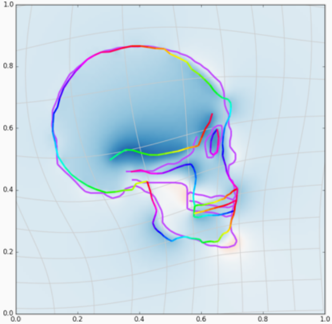


(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .42$.



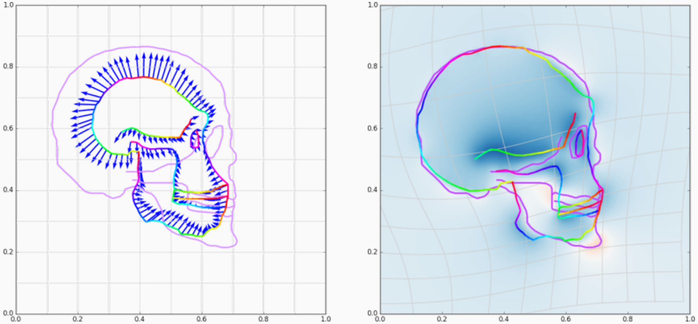
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .43$.

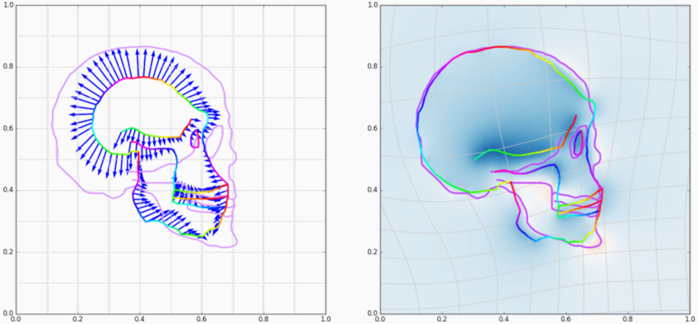
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .44$.

Influence of the kernel width

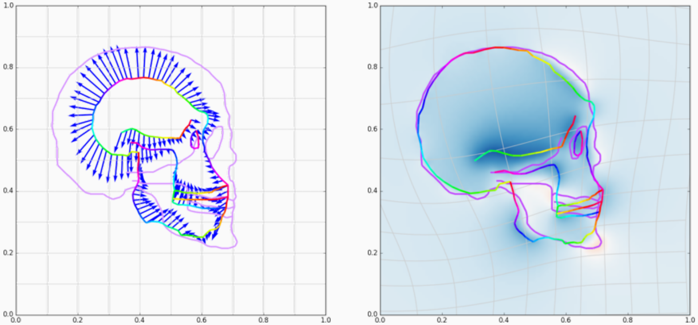


(a) Momentum p_0.

(b) Shooted model q_1.

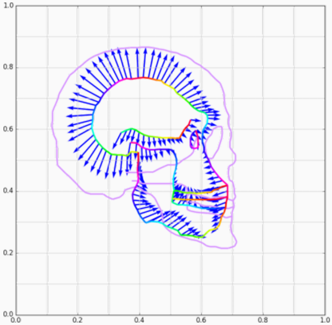
Figure 19: Final matching, $\sigma = .45$.

Influence of the kernel width

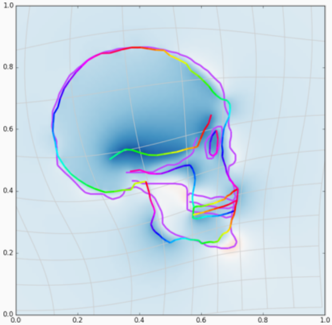


(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .46$.



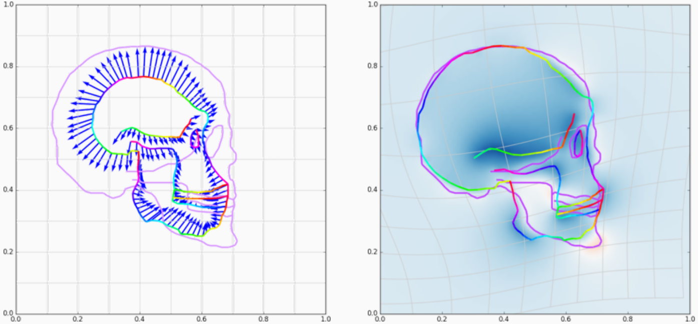
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .47$.

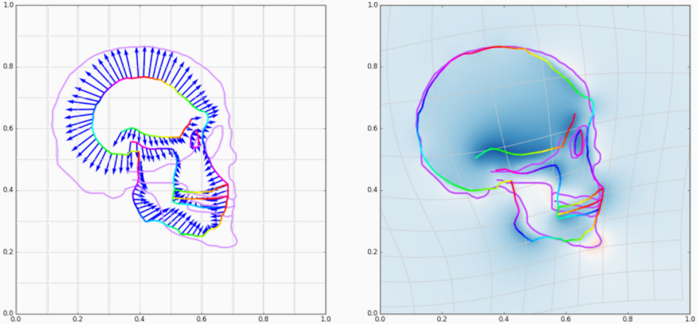
Influence of the kernel width



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .48$.

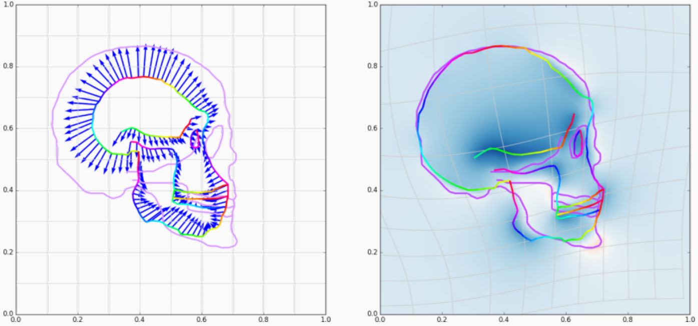
Influence of the kernel width



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .49$.

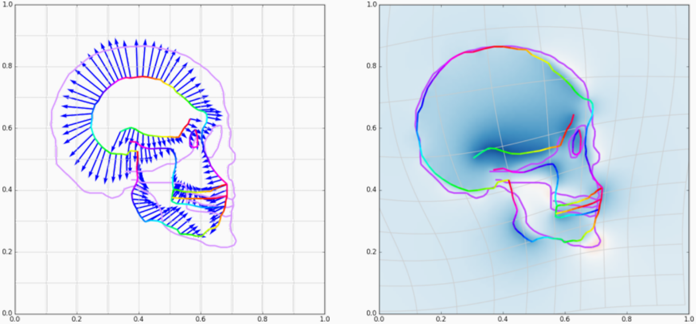
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .5$.

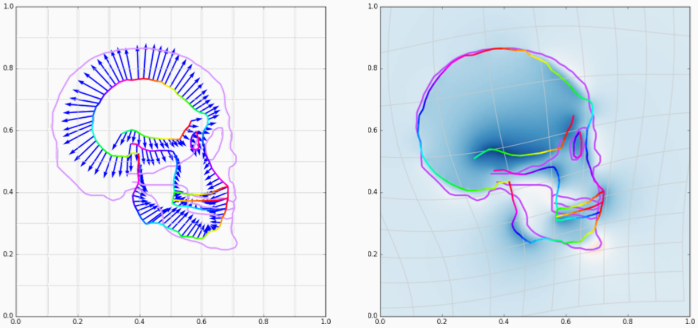
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

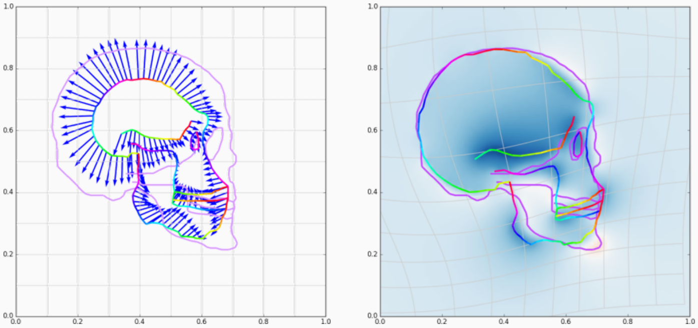
Figure 19: Final matching, $\sigma = \mathtt{.51}$.

Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

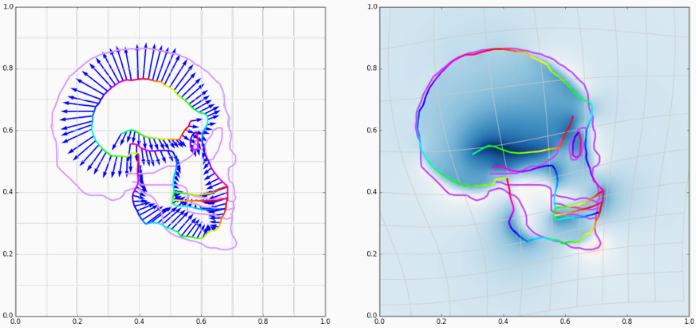
Figure 19: Final matching, $\sigma = .52$.



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .53$.

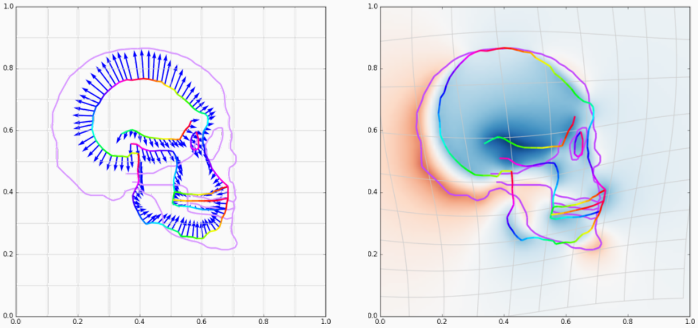
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

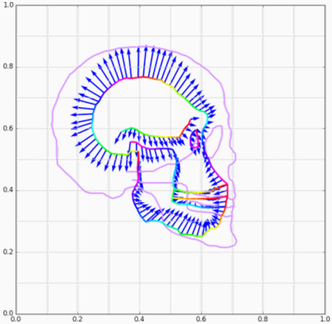
Figure 19: Final matching, $\sigma = .54$.

Influence of the kernel width

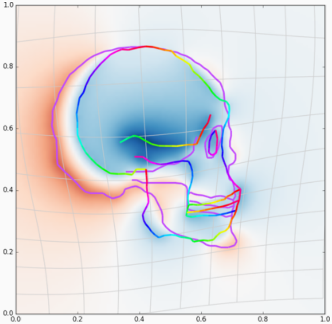


(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .55$.



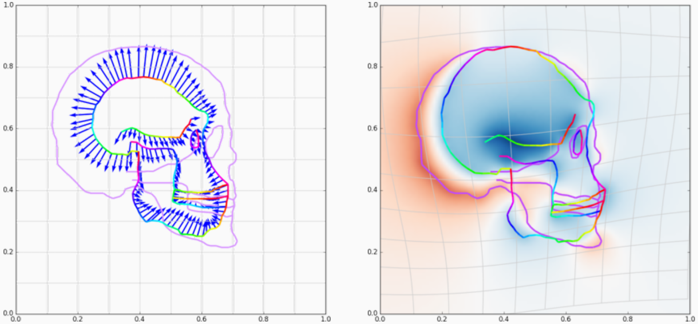
(a) Momentum p_0.



(b) Shooted model q_1.

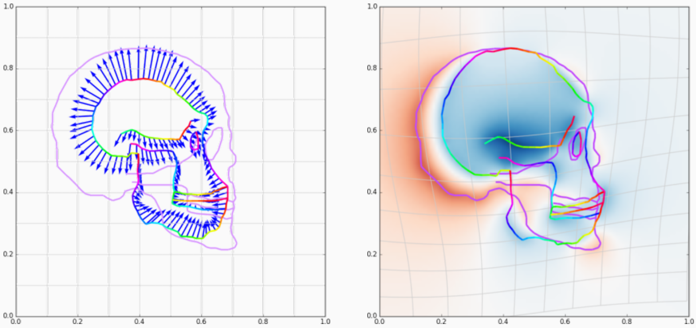
Figure 19: Final matching, $\sigma = .56$.

Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .57$.

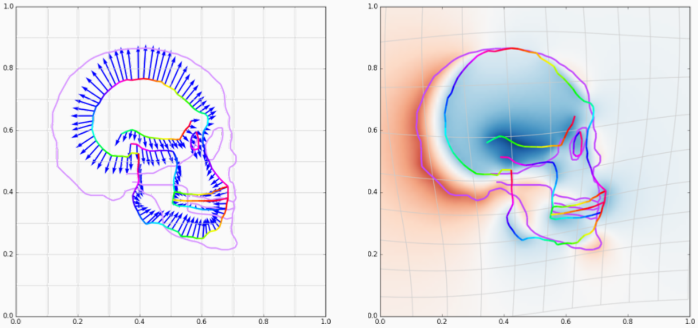


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .58$.

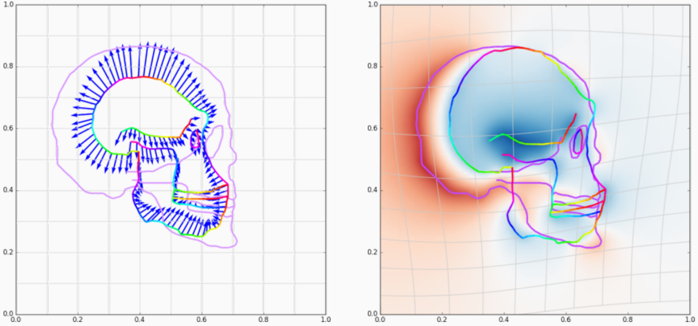
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

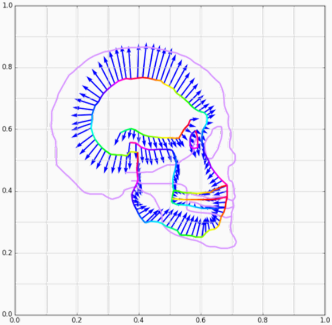
Figure 19: Final matching, $\sigma = .59$.

Influence of the kernel width

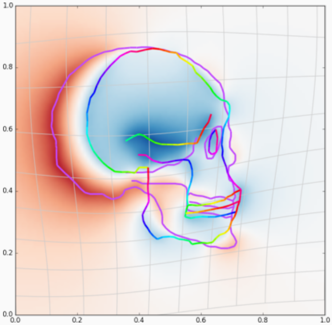


(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .6$.



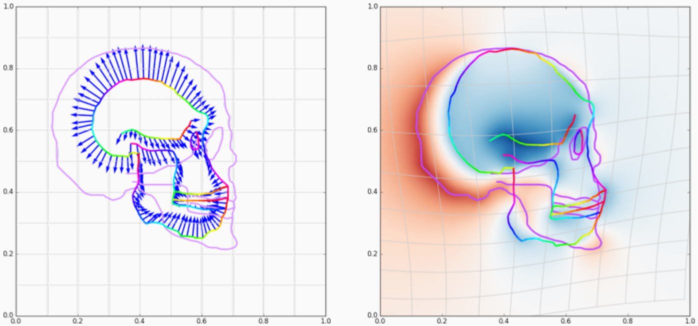
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .61$.

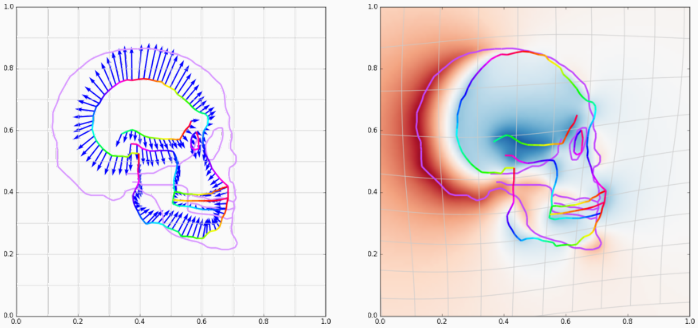
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .62$.

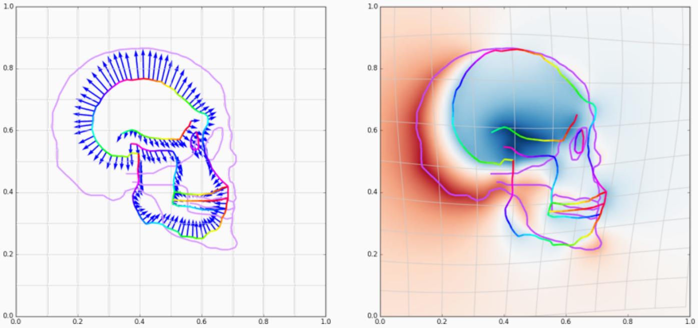
Influence of the kernel width



(a) Momentum p_0.   (b) Shooted model q_1.

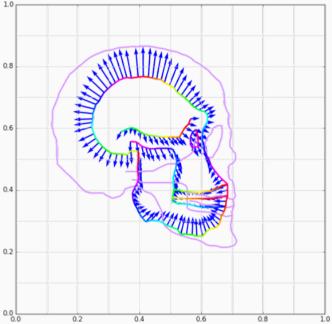
Figure 19: Final matching, $\sigma = .63$.

Influence of the kernel width

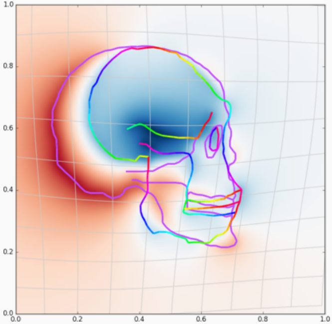


(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .64$.



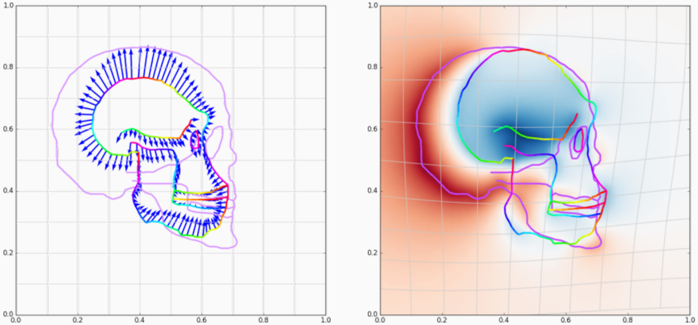
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .65$.

Influence of the kernel width

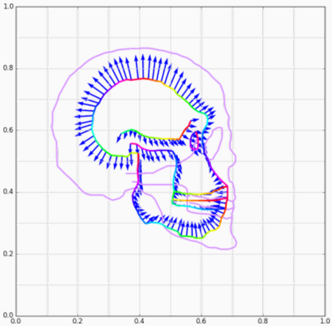


(a) Momentum p_0.

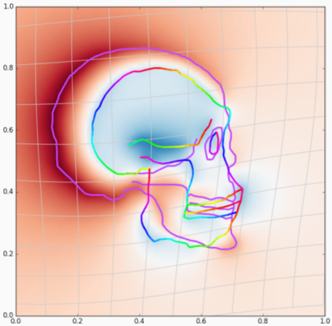
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .66$.

Influence of the kernel width



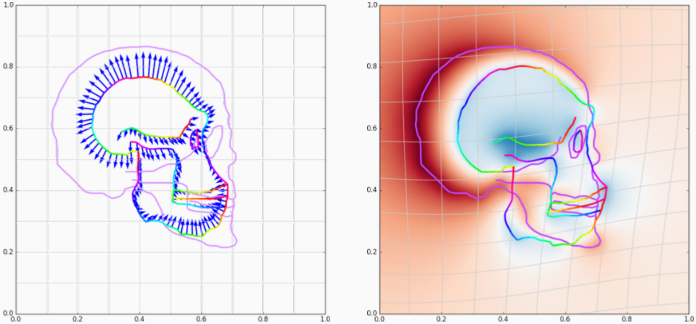
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .67$.

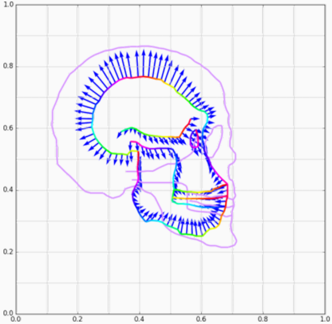
Influence of the kernel width



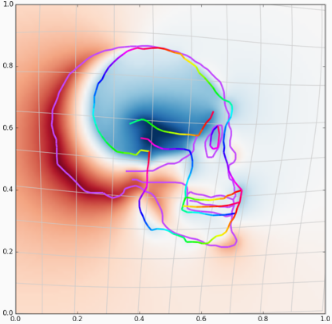
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .68$.

Influence of the kernel width



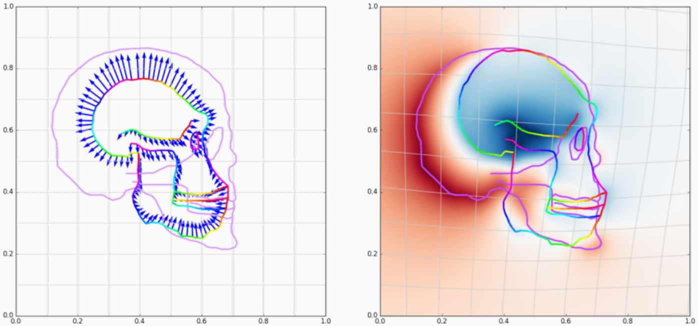
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .69$.

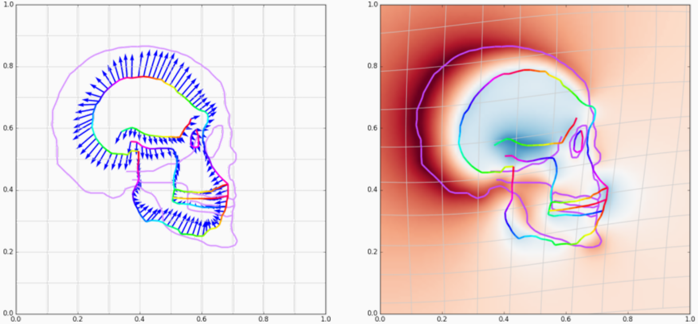
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{.70}$.

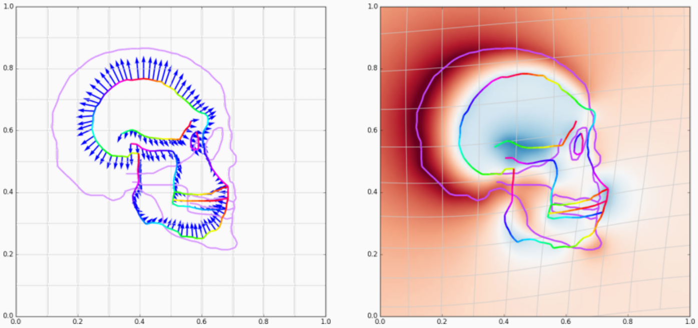
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .71$.

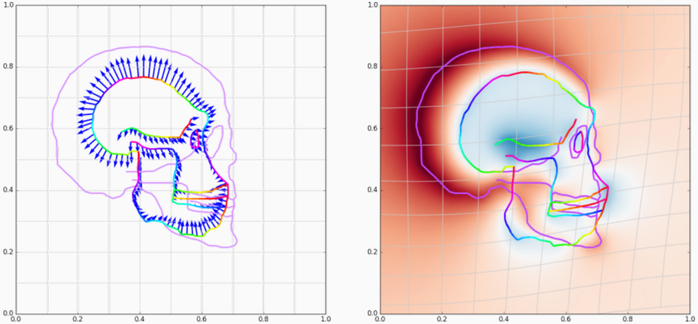
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .72$.

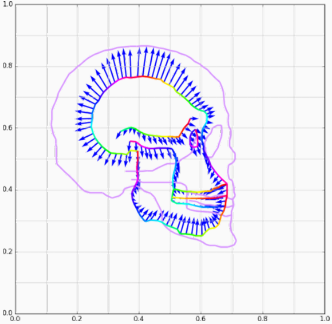
Influence of the kernel width



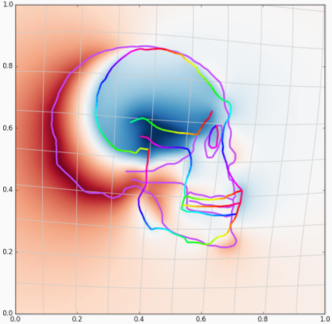
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .73$.

Influence of the kernel width



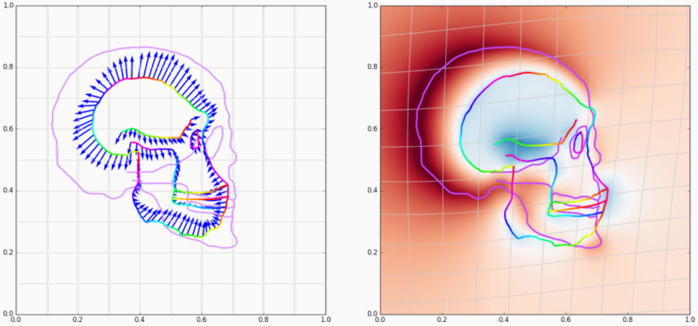
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .74$.

Influence of the kernel width

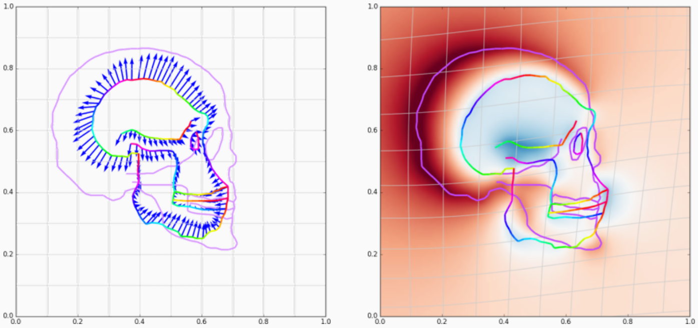


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .75$.

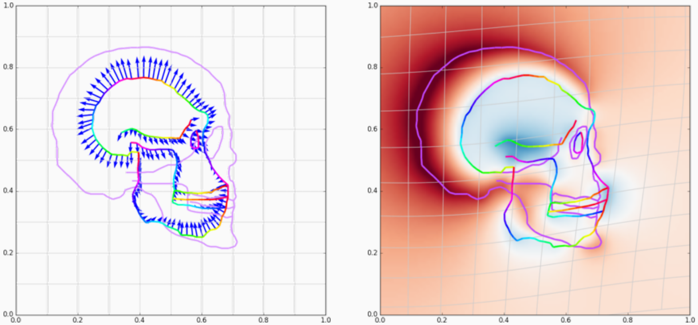
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .76$.

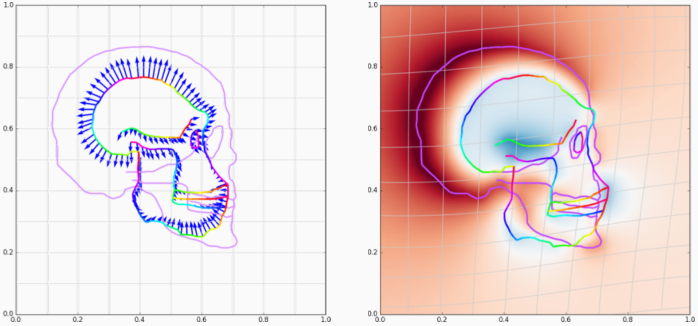
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .77$.

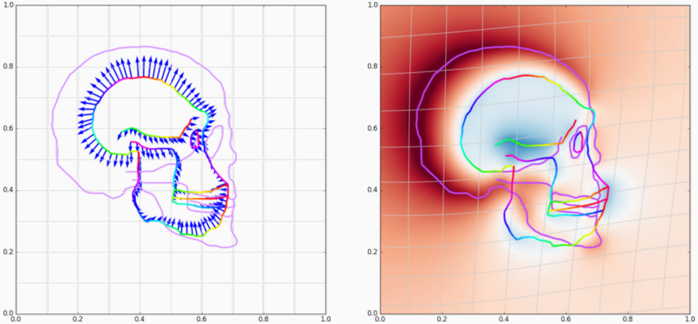
Influence of the kernel width



(a) Momentum p_0.　(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .78$.

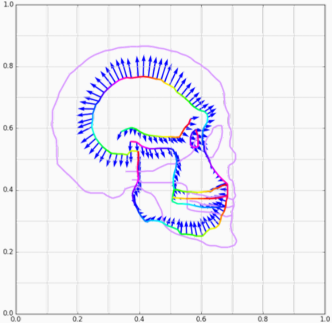
Influence of the kernel width



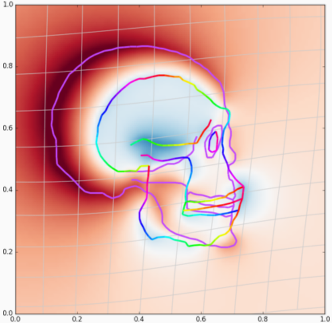
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .79$.

Influence of the kernel width



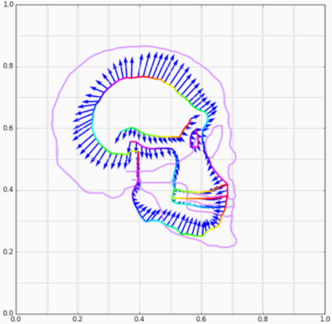
(a) Momentum p_0.



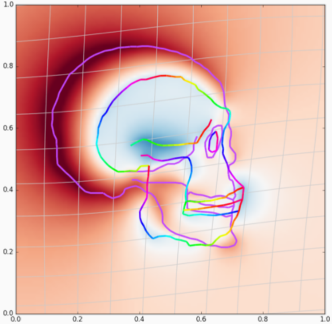
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .8$.

Influence of the kernel width

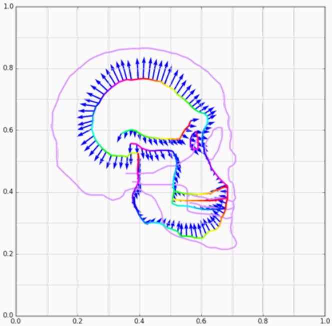


(a) Momentum p_0.

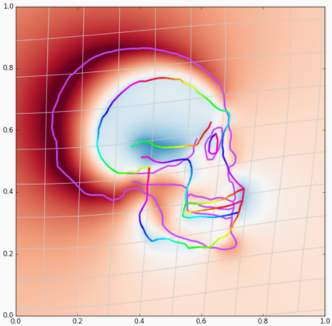


(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .81$.



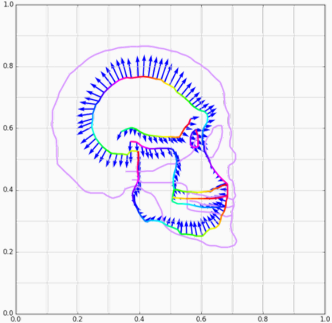
(a) Momentum p_0.



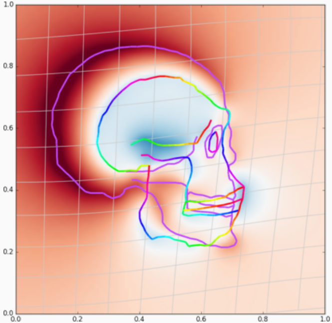
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .82$.

Influence of the kernel width

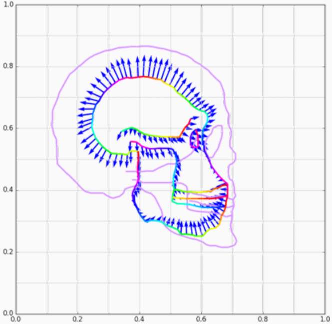


(a) Momentum p_0.

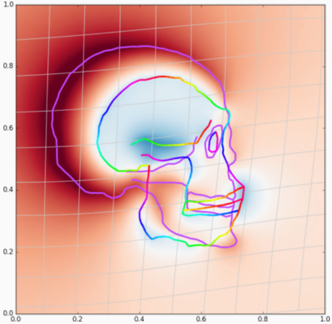


(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .83$.



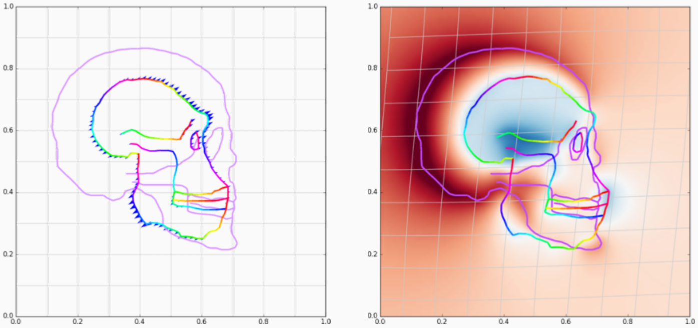
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .84$.

Influence of the kernel width

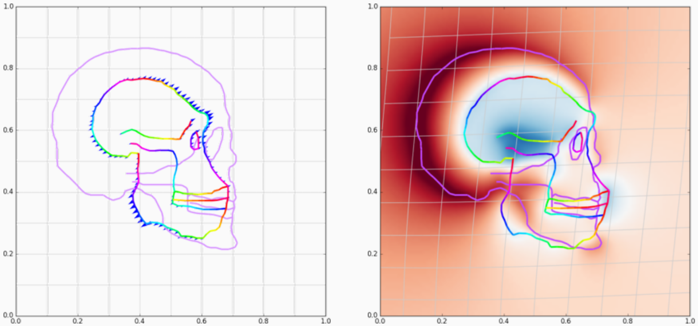


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .85$.

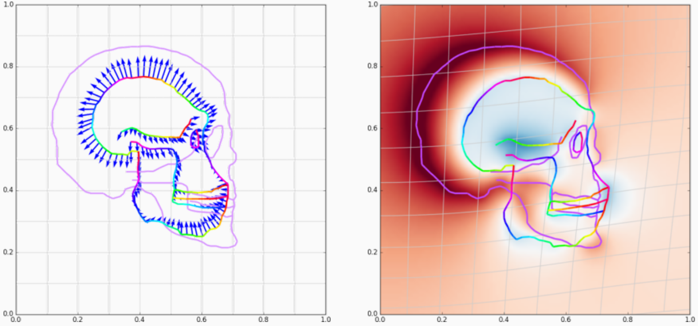
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .86$.

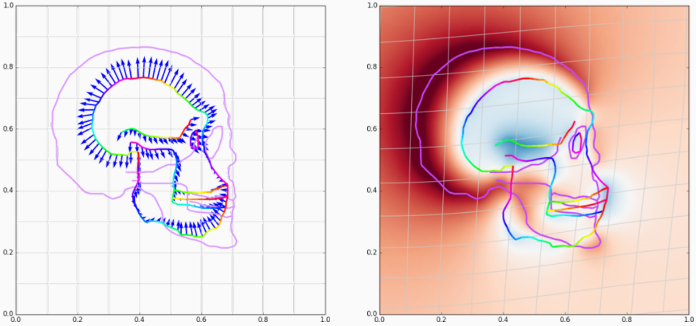
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .87$.

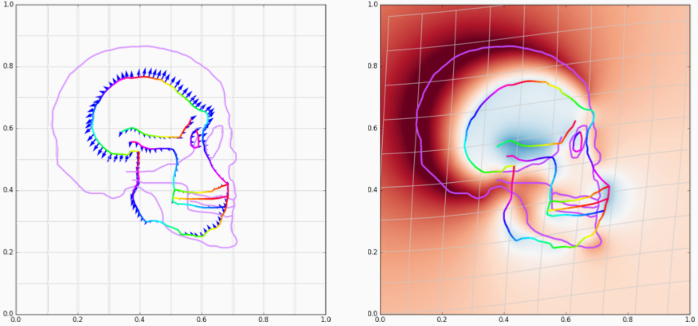
Influence of the kernel width



(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .88$.

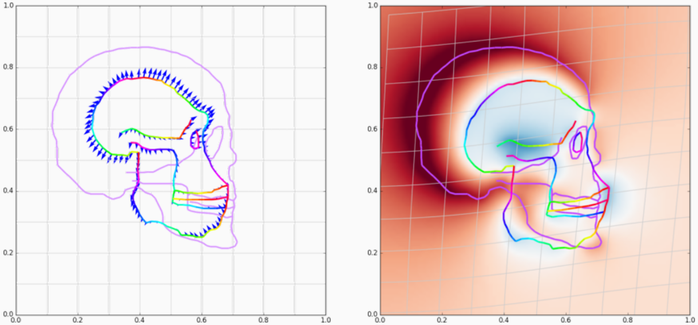
Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

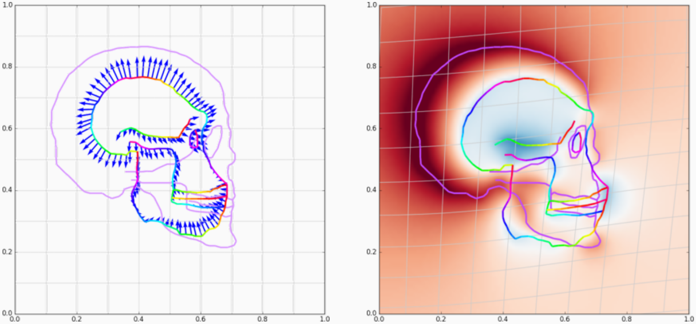
Figure 19: Final matching, $\sigma = .89$.

Influence of the kernel width



(a) Momentum p_0. (b) Shooted model q_1.

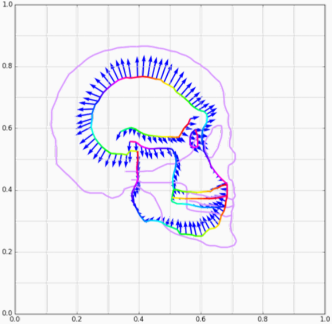
Figure 19: Final matching, $\sigma = .9$.



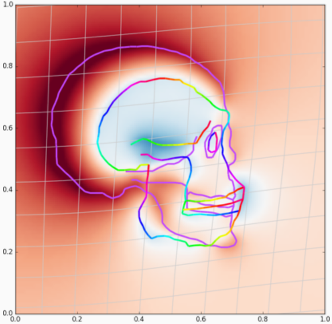
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .91$.

Influence of the kernel width



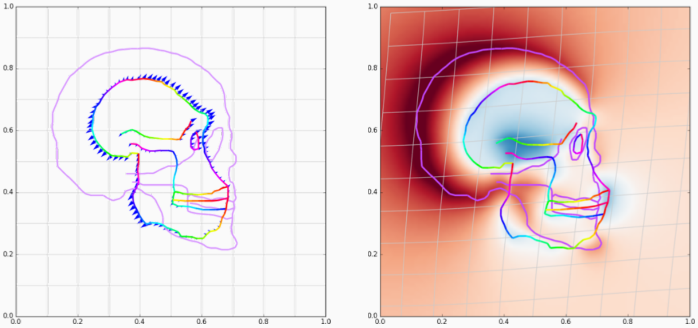
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .92$.

Influence of the kernel width

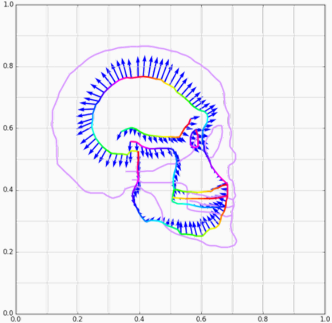


(a) Momentum p_0.

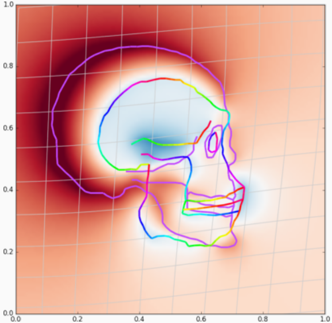
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .93$.

Influence of the kernel width

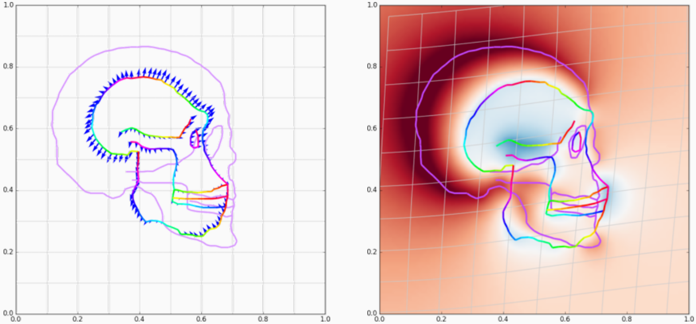


(a) Momentum p_0.



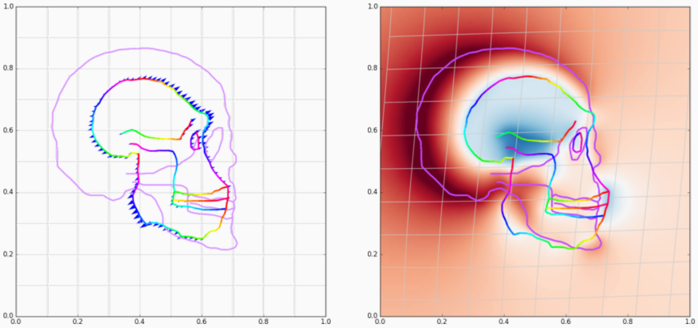
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .94$.



(a) Momentum p_0. (b) Shooted model q_1.

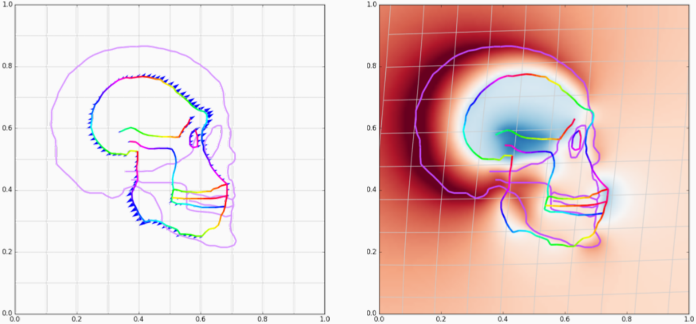
Figure 19: Final matching, $\sigma = .95$.



(a) Momentum p_0.

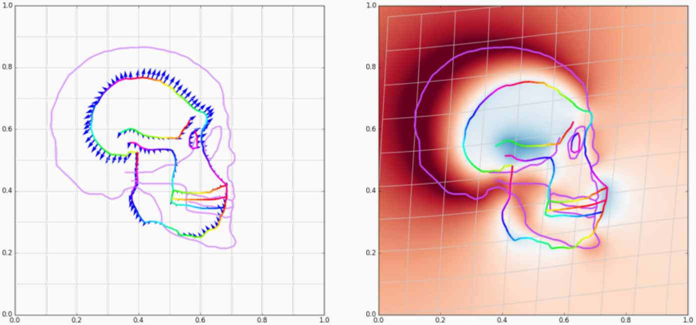
(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .96$.



(a) Momentum p_0. (b) Shooted model q_1.

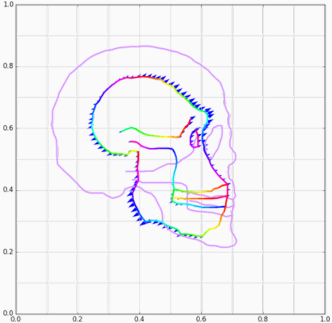
Figure 19: Final matching, $\sigma = .97$.



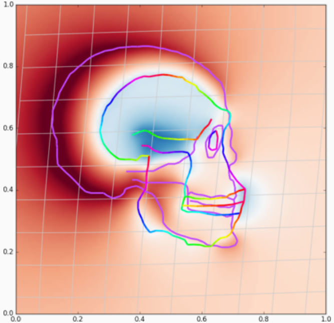
(a) Momentum p_0. (b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .98$.

Influence of the kernel width



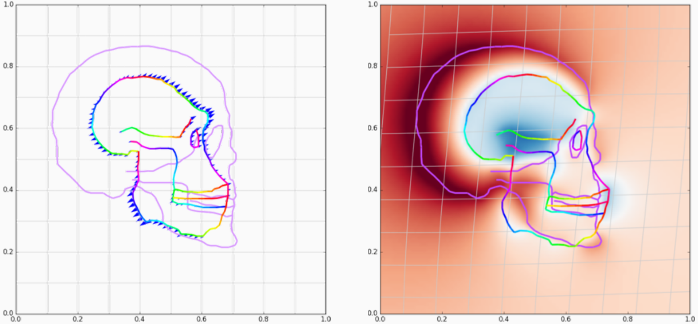
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = .99$.

Influence of the kernel width



(a) Momentum p_0.

(b) Shooted model q_1.

Figure 19: Final matching, $\sigma = \texttt{1.0}$.

Conclusion

OT as a fidelity term

Pros :

- Principled globalization trick.
- Versatile : any distance on any feature space will do.

OT as a fidelity term

Pros :

- Principled globalization trick.
- Versatile : any distance on any feature space will do.

Cons :

- Only affordable for large ε diffusion values.
- Can still be tricked in symmetric situations.

OT as a fidelity term

Pros :

- Principled globalization trick.
- Versatile : any distance on any feature space will do.

Cons :

- Only affordable for large ε diffusion values.
- Can still be tricked in symmetric situations.

Coming soon (say, end of 2017) :

- Implementation on 3D dense images.
- Investigate the continuum "RKHS $\rightarrow$ OT".

Pros :

- Incredibly versatile and math-friendly.
- Unleash your GPU without getting stuck in CUDA.
- Exact derivative : safer to use with BFGS and line searchs.

`theano` for image registration

Pros :

- Incredibly versatile and math-friendly.
- Unleash your GPU without getting stuck in CUDA.
- Exact derivative : safer to use with BFGS and line searchs.

Cons :

- Current bottleneck : memory overflows.
- Using BCH formula will require some hack (`OpFromGraph`...).

`theano` for image registration

Pros :

- Incredibly versatile and math-friendly.
- Unleash your GPU without getting stuck in CUDA.
- Exact derivative : safer to use with BFGS and line searchs.

Cons :

- Current bottleneck : memory overflows.
- Using BCH formula will require some hack (`OpFromGraph`...).

Stay tuned :

- RAM-GPU memory links coming soon ?
- Libraries are moving fast : check `TensorFlow`, etc.

Questions?