Presenting the LDDMM framework

A Riemannian setting, linking Procustes to Monge.

Jean Feydy

June 1, 2017

Écoles Normales Supérieures de Paris et Paris-Saclay

Table of contents

1. Introduction
2. Procustes Analysis
3. Optimal Transport
4. The LDDMM framework
    - Regularized transport : a Riemannian problem
    - Geodesic shooting on a Riemannian manifold
    - Kernel cometrics and Diffeomorphic trajectories
    - An iterative matching algorithm
    - Let's read some code
    - Results
5. Conclusion

Some information

Jean Feydy (2016-2019) :

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

Some information

Jean Feydy (2016-2019) :

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

Notes for this talk available online (in French) :

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

- *Culture Mathématique*, chap. 9-10.
- *Introduction à la Géométrie Riemannienne par l'Étude des Espaces de Formes.*

Introduction

How do we decompose variability ?

Research in **Image Processing** :

- Signal analysis : compression, denoising, etc.
- Classification : Google image, etc.
- Population Analysis : clinical studies, etc.

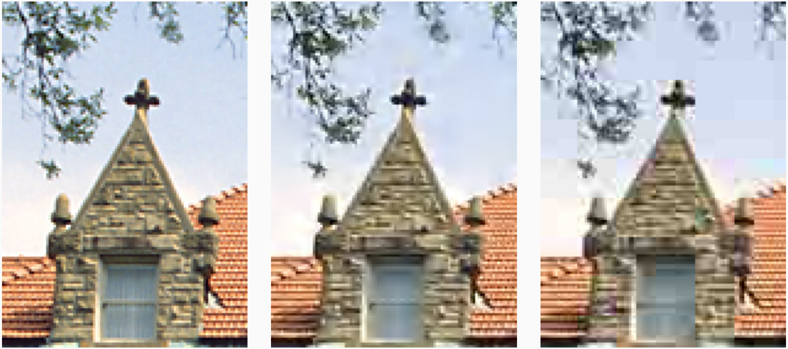
How do we decompose variability ?

Research in **Image Processing** :

- Signal analysis : compression, denoising, etc.
- Classification : Google image, etc.
- Population Analysis : clinical studies, etc.

We need appropriate representations.

JPEG2000, JPEG : Wavelets, Blockwise (high + low) frequencies



(a) Original image. (b) JPEG2000, 20 : 1. (c) JPEG, 20 : 1.

Figure 1: Taken from `www.photozone.de`.

Convolutional Neural Networks : Texture + Structure

Figure 2: Reference image.

Convolutional Neural Networks : Texture + Structure

Figure 2: With a transferred texture component. [3]

Convolutional Neural Networks : Texture + Structure

Figure 2: With a transferred texture component. [3]

Convolutional Neural Networks : Texture + Structure

Figure 2: With a transferred texture component. [3]

Figure 2: With a transferred texture component. [3]

Convolutional Neural Networks : Texture + Structure

Figure 2: With a transferred texture component. [3]

How do we handle intra-class variability ?

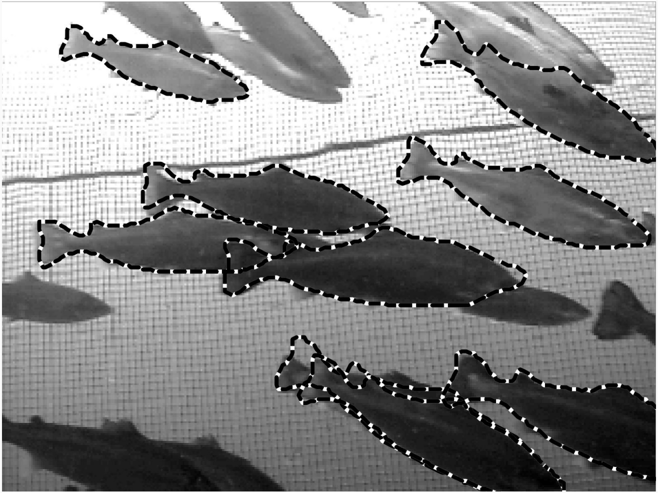


Figure 3: Silhouettes segmented from a fishing net. [2]

Procustes Analysis

Position, Scale and Orientation

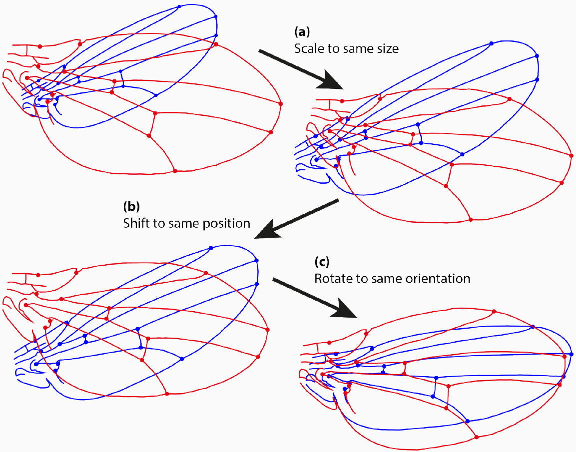


Figure 4: Matching the blue wing on the red one. (Wikipedia)

From images to labeled point clouds

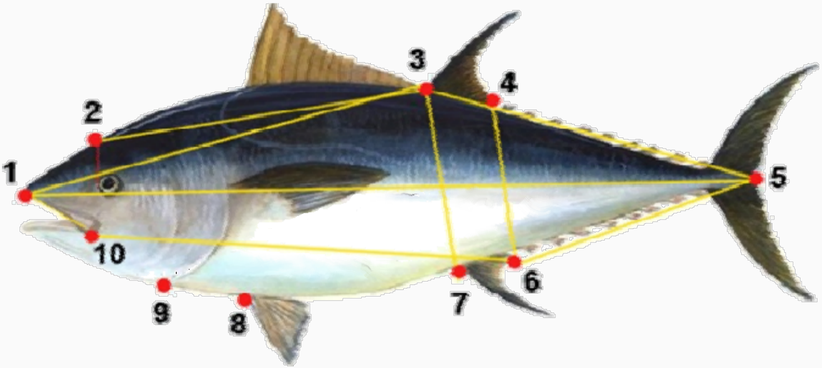


Figure 5: Anatomical landmarks on a tuna fish. [1]

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

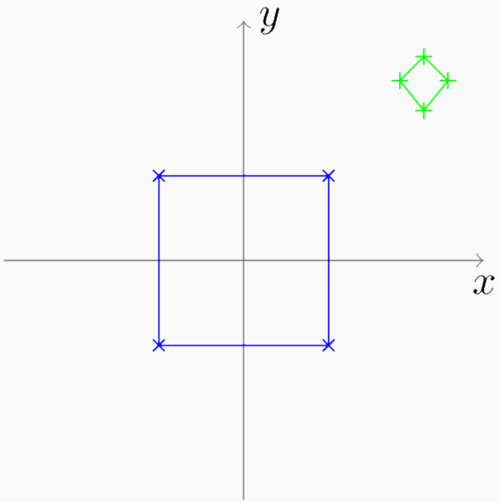


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

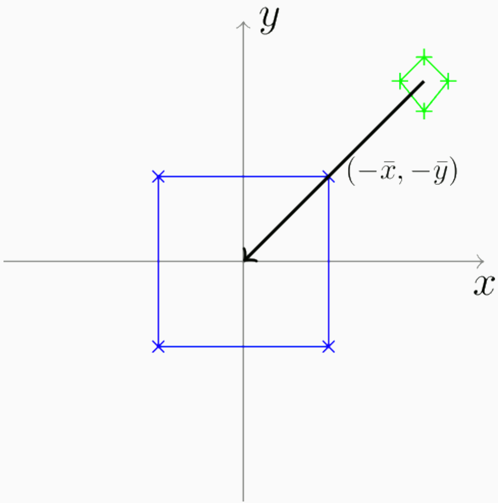


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

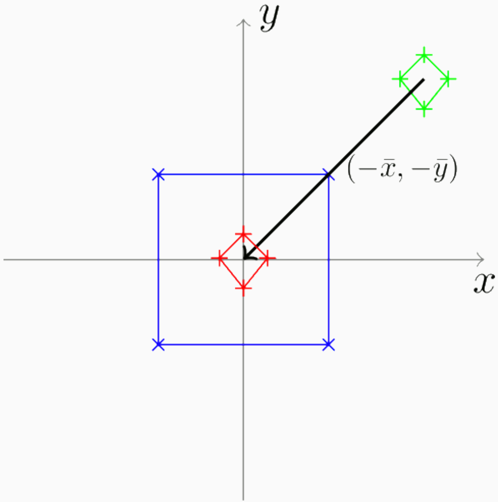


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

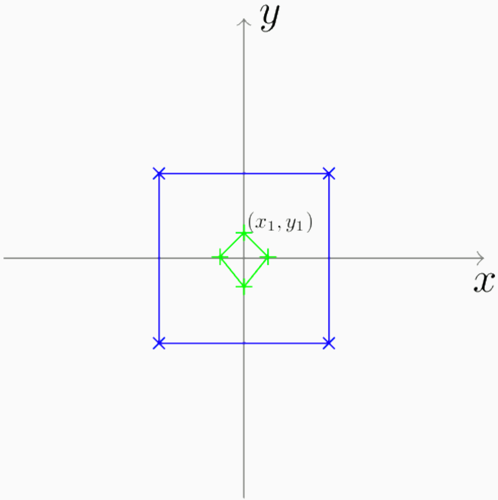


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

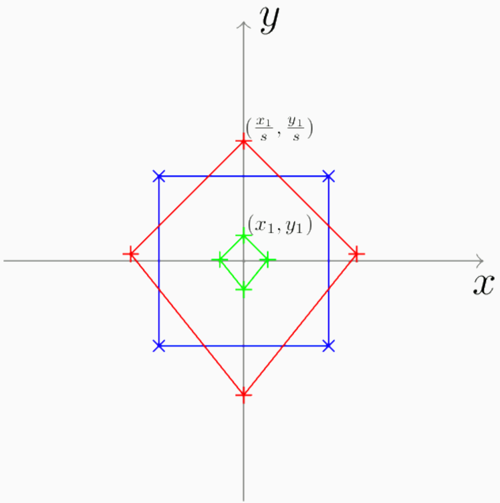


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

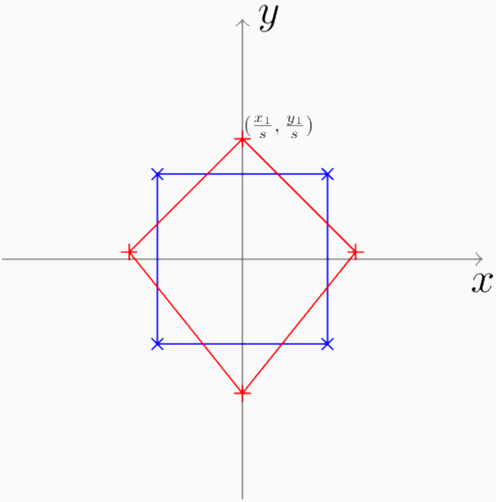


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

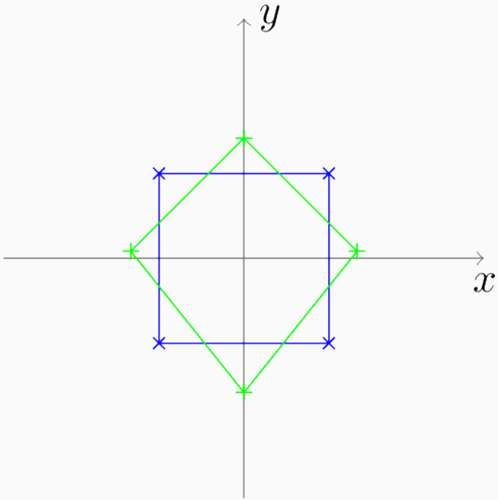


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

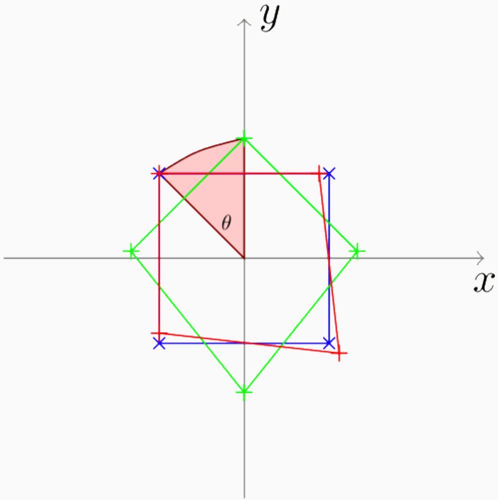


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

Typical run on polygons

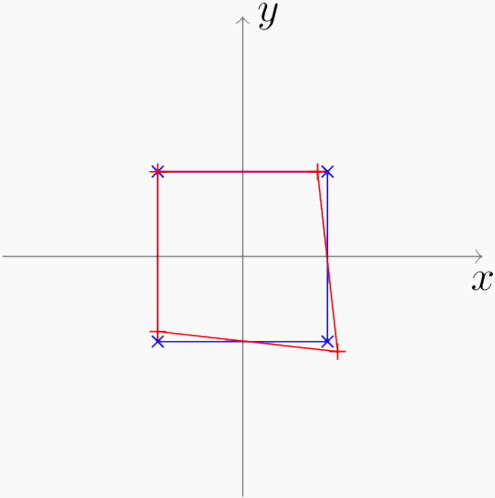


Figure 6: Matching a kitesurf on a square. (Wikipedia, Linschn)

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

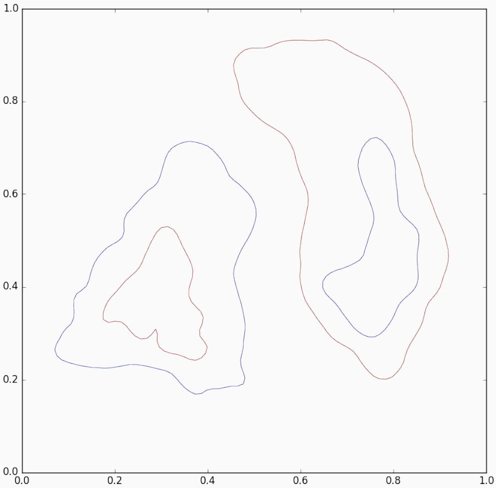


Figure 7: Optimal transport between two curves seen as mass distributions : from a déblai to a remblai.

Image matching as a mass-carrying problem

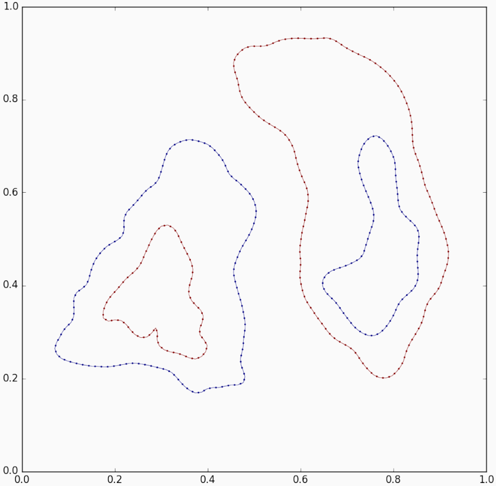


Figure 7: Optimal transport between two curves seen as mass distributions : from a déblai to a remblai.

Image matching as a mass-carrying problem

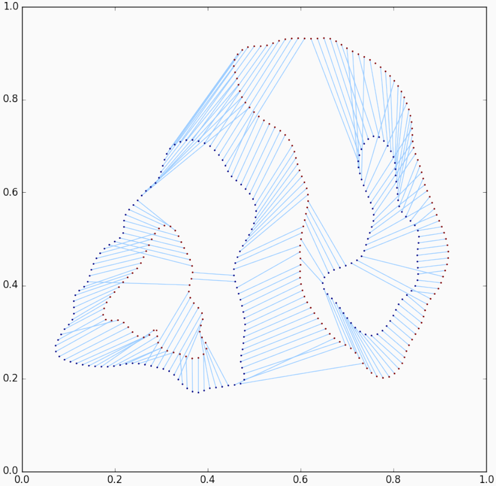


Figure 7: Optimal transport between two curves seen as mass distributions : from a déblai to a remblai.

Dynamic formulation

Let : $(x^1, \ldots, x^I)$ and $(y^1, \ldots, y^J)$ be two point clouds
and $(\mu_1, \ldots, \mu_I)$, $(\nu_1, \ldots, \nu_J)$ 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 $\gamma^m : t \in [0,1] \mapsto \gamma_t^m$ 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,

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

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

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 : 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

$$C^{X,Y}(\Gamma) \ = \ \sum_{i,j} \gamma_{i,j} \, c_{i,j} \tag{7}$$

under the constraints :

$$\forall\, i,j,\ \ \gamma_{i,j} \geqslant 0, \qquad \forall\, i,\ \ \sum_{j} \gamma_{i,j} = \mu_i, \qquad \forall\, j,\ \ \sum_{i} \gamma_{i,j} = \nu_j. \tag{8}$$

Static formulation : permutation

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

$$C^{X,Y}(\sigma) \;=\; \sum_{m=1}^{M} \left\| x^m - y^{\sigma(m)} \right\|^2 . \tag{9}$$

σ is an optimal labeling.

The Sinkhorn algorithm : an efficient iterative solver

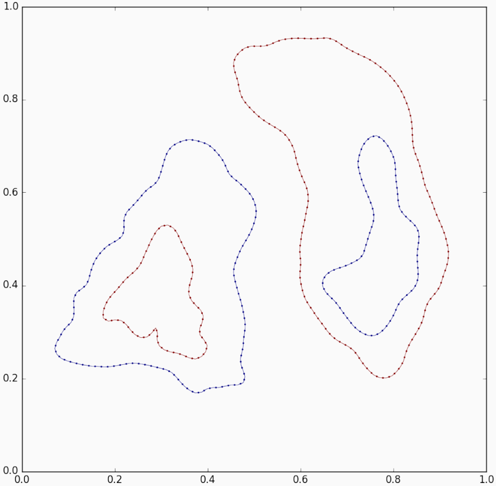


Figure 8: Measures to match.

The Sinkhorn algorithm : an efficient iterative solver

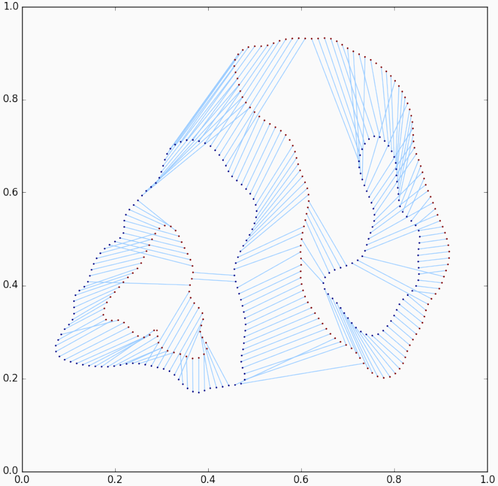


Figure 8: Monge transport, $\sqrt{\varepsilon} = 0$.

The Sinkhorn algorithm : an efficient iterative solver

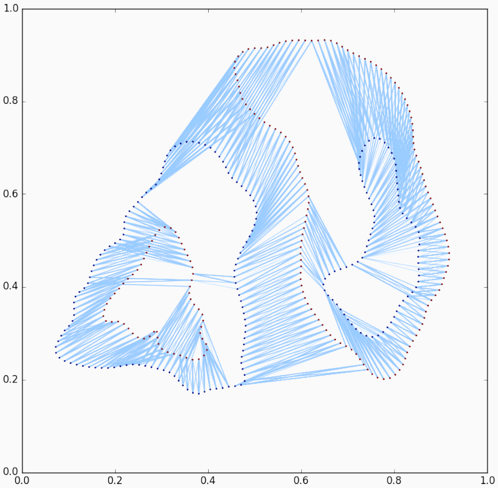


Figure 8: Diffuse transport, $\sqrt{\varepsilon} = .01$.

The Sinkhorn algorithm : an efficient iterative solver

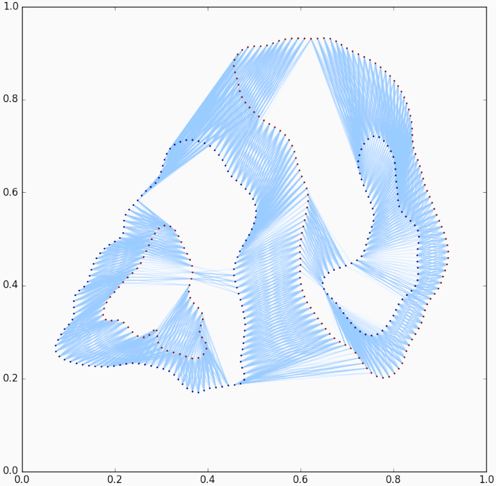


Figure 8: Diffuse transport, $\sqrt{\varepsilon} = .03$.

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 LDDMM framework

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

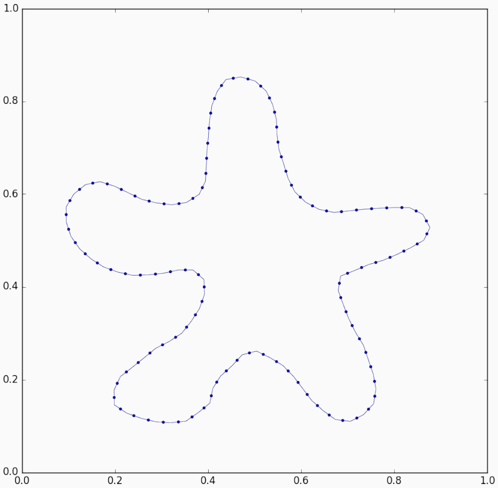


Figure 9: Source.

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

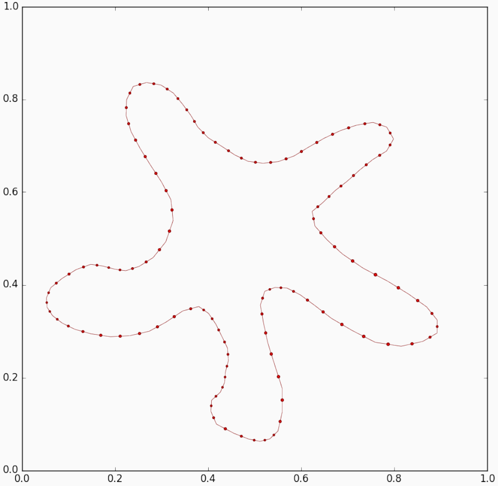


Figure 9: Target.

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

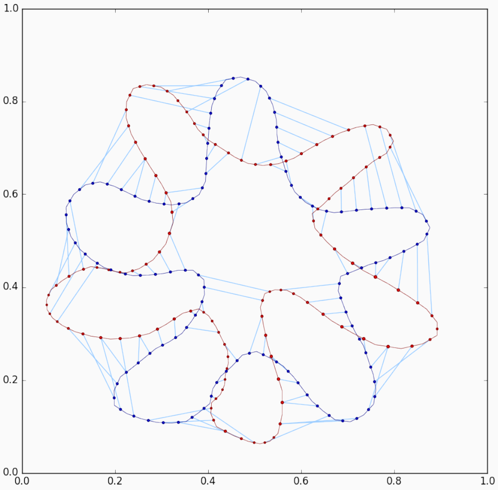


Figure 9: OT matching.

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

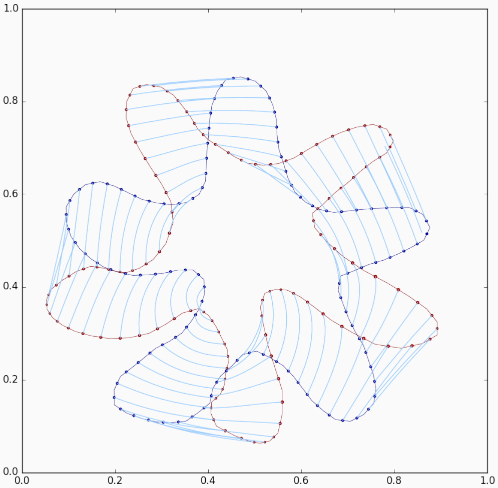


Figure 9: LDDMM matching.

The LDDMM framework

Regularized transport : a Riemannian problem

Static regularization : a first attempt

A naive way to regularize transport :
Find $\sigma : [\![1, M]\!] \to [\![1, M]\!]$ minimizing

$$C_k^{X,Y}(\sigma) = \underbrace{\sum_m \left\| x^m - y^{\sigma(m)} \right\|^2}_{\text{Displacement cost}} + \underbrace{\sum_{m,m'} k(x^m, x^{m'}) \cdot \left\| y^{\sigma(m)} - y^{\sigma(m')} \right\|^2}_{\text{Regularization cost}}, \tag{10}$$

with $k(x, y)$ a kernel neighborhood function.

An appropriate *cost* should give rise to a *distance*

If $C_k^{X,Y}(\,\cdot\,)$ is a cost on matchings, we define

$$\mathrm{d}_k(X,Y) = \min_{\sigma}\ C_k^{X,Y}(\sigma). \tag{11}$$

An appropriate *cost* should give rise to a *distance*

If $C_k^{X,Y}(\cdot)$ is a cost on matchings, we define

$$\mathrm{d}_k(X, Y) = \min_{\sigma} \; C_k^{X,Y}(\sigma). \tag{11}$$

It'd better be :

- Null iff X and Y stand for the same shape
- Symmetric
- Compatible with the triangle inequality

Static regularization : symmetry without continuity

Find a permutation $\sigma : [\![1, M]\!] \to [\![1, M]\!]$ minimizing

$$C_{k,\mathrm{sym}}^{X,Y}(\sigma) = \underbrace{\sum_{m} \left\| x^m - y^{\sigma(m)} \right\|^2}_{\text{Displacement cost}} + \underbrace{\frac{1}{2} \sum_{m,m'} k(x^m, x^{m'}) \cdot \left\| y^{\sigma(m)} - y^{\sigma(m')} \right\|^2}_{X \to Y \text{ regularization cost}}$$
$$+ \underbrace{\frac{1}{2} \sum_{m,m'} k(y^m, y^{m'}) \cdot \left\| x^{\sigma^{-1}(m)} - x^{\sigma^{-1}(m')} \right\|^2}_{Y \to X \text{ regularization cost}} .$$

This cost is symmetric, but does not handle properly the shapes between X and Y.

Going back to the kinematic transportation

Find a collection of paths γ^m from X to Y minimizing

$$C_k(\gamma) = \int_0^1 \Bigg[\underbrace{\sum_m \|\dot{\gamma}_t^m\|^2}_{\text{Displacement cost}} + \underbrace{\sum_{m,m'} k(\gamma_t^m, \gamma_t^{m'}) \cdot \left\|\dot{\gamma}_t^m - \dot{\gamma}_t^{m'}\right\|^2}_{\text{Regularization cost}} \Bigg] \mathrm{d}t.$$

Particles will move optimally if they are :

- lazy
- gregarious wrt. their k-neighbors

Geodesic path-finding on a Riemannian manifold of point clouds

With $\gamma_t = (\gamma_t^1, \ldots, \gamma_t^M) \in \mathbb{R}^{M \times D}$, we can write

$$C_k(\gamma) = \int_0^1 \dot{\gamma}_t^\top g_{\gamma_t} \dot{\gamma}_t \mathrm{d}t. \tag{12}$$

Geodesic path-finding on a Riemannian manifold of point clouds

With $\gamma_t = (\gamma_t^1, \dots, \gamma_t^M) \in \mathbb{R}^{M \times D}$, we can write

$$C_k(\gamma) = \int_0^1 \dot{\gamma}_t^\intercal g_{\gamma_t} \dot{\gamma}_t \mathrm{d}t. \tag{12}$$

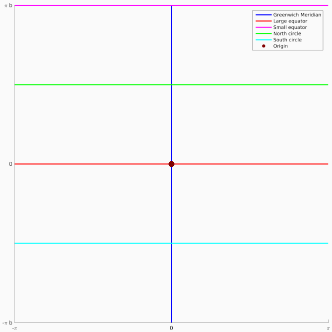
Optimal deformations are geodesics on the space of landmarks $\mathbb{R}^{M \times D}$ endowed with a Riemannian metric g_q :

$$\begin{aligned} \frac{\big(\mathrm{d}_g(q \to q + v \cdot \mathrm{d}t)\big)^2}{\mathrm{d}t} &= \sum_m \|v^m\|^2 + \sum_{m,m'} k(q^m, q^{m'}) \cdot \left\| v^m - v^{m'} \right\|^2 \\ &= \; v^\intercal g_q v \; = \; \|v\|_{g_q}^2 \end{aligned} \tag{13}$$

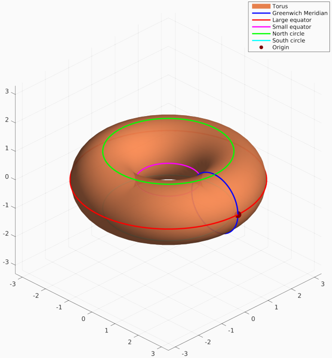
The LDDMM framework

Geodesic shooting on a Riemannian manifold

Riemann : conveniently working with arbitrary geometries



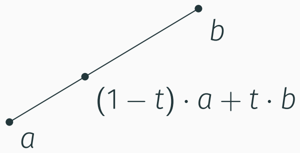
(a) As a deformed square.



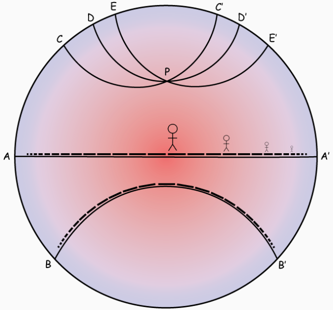
(b) Embedded in $\mathbb{R}^3$.

Figure 10: The donut-shaped torus.

Sometimes, we can compute geodesics explicitly...



(a) The Euclidean plane.



(b) The Poincaré disk.

Figure 11: Explicit geodesics on homogeneous manifolds.
(b) is adapted from `www.pitt.edu/~jdnorton/`.

But this is not the case in general

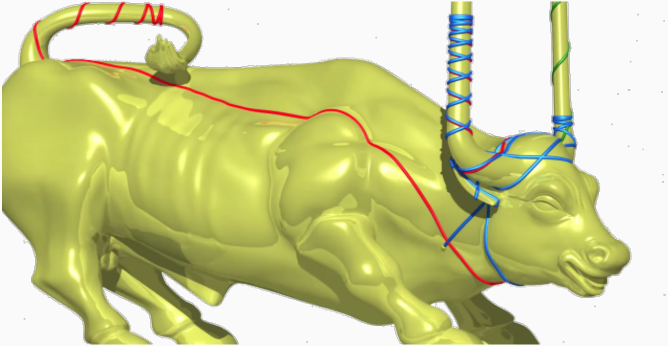


Figure 12: Geodesics on the Duhem's bull, embedded in $\mathbb{R}^3$. Taken from `www.chaos-math.org`.

A first result : the geodesic equation

Geodesic $\Longrightarrow$ locally "straight" $\Longrightarrow$ second order ODE, the **geodesic equation** satisfied by $\gamma_t = (\gamma_t^1, \ldots, \gamma_t^D)$:

$$\forall\, d \in [\![1, D]\!]\,, \quad \ddot{\gamma}_t^d = - \sum_{1 \leqslant i,j \leqslant D} \Gamma_{ij}^d(\gamma_t) \cdot \dot{\gamma}_t^i\, \dot{\gamma}_t^j, \tag{14}$$

where the Christoffel symbols $\Gamma_{ij}^d(q)$ are given by :

$$\Gamma_{ij}^d(q) = \frac{1}{2} \sum_{l=1}^{D} g^{dl}(q) \cdot \big(\partial_i g_{jl}(q) + \partial_j g_{il}(q) - \partial_l g_{ij}(q)\big)\,, \tag{15}$$

with g_{ij} the *metric* tensor and g^{dl} its inverse, the *cometric*.

From celerity to momentum

The “Christoffel” equation is an ODE in the tangent bundle :

$$(q_t, v_t) \;=\; (\gamma_t, \dot{\gamma}_t). \tag{16}$$

From celerity to momentum

The "Christoffel" equation is an ODE in the tangent bundle :

$$(q_t, v_t) \;=\; (\gamma_t, \dot{\gamma}_t). \tag{16}$$

Hamilton : one should work in the cotangent bundle :

$$(q_t, p_t) \;=\; (q_t, g_{q_t} v_t). \tag{17}$$

From celerity to momentum

The "Christoffel" equation is an ODE in the tangent bundle :

$$(q_t, v_t) \;=\; (\gamma_t, \dot{\gamma}_t). \tag{16}$$

Hamilton : one should work in the cotangent bundle :

$$(q_t, p_t) \;=\; (q_t, g_{q_t} v_t). \tag{17}$$

We denote $K_q = g_q^{-1}$ and $H(q,p) = \frac{1}{2} p^\intercal K_q p$, so that

$$\frac{1}{2} v_t^\intercal g_{q_t} v_t \;=\; \underbrace{\frac{1}{2} \|\dot{\gamma}_t\|_{\gamma_t}^2}_{\text{Kinetic energy}} \;=\; H(q_t, p_t). \tag{18}$$

Hamiltonian geodesic equations

Hamilton, 1833

γ_t is a geodesic if and only if the lifted cotangent trajectory (q_t, p_t) follows the Hamiltonian equation :

$$\left\{ \begin{array}{lllll} \dot{q}_t & = & +\frac{\partial H}{\partial p}(q_t, p_t) & = & +K_{q_t} p_t \\ \dot{p}_t & = & -\frac{\partial H}{\partial q}(q_t, p_t) & = & -\partial_q (p_t, K_q p_t)(q_t) \end{array} \right. . \tag{19}$$

Hamiltonian geodesic equations

Hamilton, 1833

γ_t is a geodesic if and only if the lifted cotangent trajectory (q_t, p_t) follows the Hamiltonian equation :

$$\left\{ \begin{array}{ccccc} \dot{q}_t & = & +\frac{\partial H}{\partial p}(q_t, p_t) & = & +K_{q_t} p_t \\ \dot{p}_t & = & -\frac{\partial H}{\partial q}(q_t, p_t) & = & -\partial_q (p_t, K_q p_t)(q_t) \end{array} \right. . \tag{19}$$

In the cotangent phase space, we flow along the symplectic gradient :

$$X(q,p) = \begin{pmatrix} +\frac{\partial H}{\partial p}(q,p) \\ -\frac{\partial H}{\partial q}(q,p) \end{pmatrix} = \text{“R}_{-90^\circ}\text{”}\big(\nabla H(q,p)\big). \tag{20}$$

Quick physical “justification”

Consider a free-falling particle of mass m :

$$\begin{aligned} q &= z, & v &= \dot{z}, \qquad (21) \\ \dot{q} &= v, & \dot{v} &= -g. \qquad (22) \end{aligned}$$

Quick physical "justification"

Consider a free-falling particle of mass m :

$$q = z, \qquad v = \dot{z}, \tag{21}$$

$$\dot{q} = v, \qquad \dot{v} = -g. \tag{22}$$

Now, we can write $\mathbf{p} = \mathbf{mv}$ so that

$$H(q,p) \;=\; \text{“}E_{\text{cin}}\text{”}(q,p) + \text{“}E_{\text{pp}}\text{”}(q,p) \;=\; \frac{1}{2}\frac{p^2}{m} + mgq. \tag{23}$$

Quick physical “justification”

Consider a free-falling particle of mass m :

$$q = z, \qquad v = \dot{z}, \tag{21}$$

$$\dot{q} = v, \qquad \dot{v} = -g. \tag{22}$$

Now, we can write $\mathbf{p} = \mathbf{mv}$ so that

$$H(q,p) \ = \ \text{“}E_{\text{cin}}\text{”}(q,p) + \text{“}E_{\text{pp}}\text{”}(q,p) \ = \ \frac{1}{2}\frac{p^2}{m} + mgq. \tag{23}$$

We find :

$$\begin{cases} \dot{q} & = & +\frac{\partial H}{\partial p} & = & +p/m \\ \dot{p} & = & -\frac{\partial H}{\partial q} & = & -mg \end{cases}. \tag{24}$$

The geodesic shooting algorithm

A geodesic path γ_t is characterized by (q_0, p_0).
To compute any geodesic starting from a source q_0, we simply need a shooting momentum p_0 and a simplistic Euler scheme :

$$\left\{ \begin{array}{lcccl} q_{t+0.1} & = & q_t & + & 0.1 \cdot K_{q_t} p_t \\ p_{t+0.1} & = & p_t & - & 0.1 \cdot \partial_q (p_t, K_q p_t)(q_t) \end{array} \right. . \tag{25}$$

The geodesic shooting algorithm

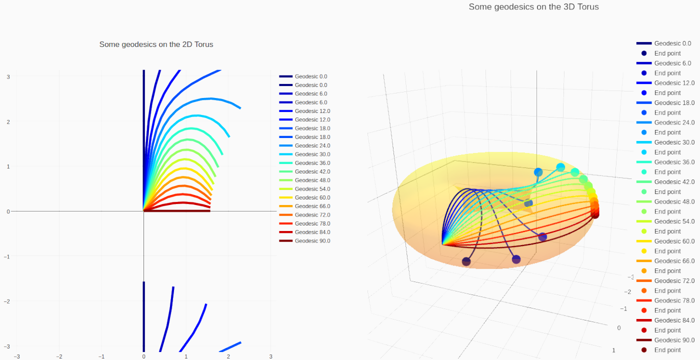
A geodesic path γ_t is characterized by (q_0, p_0).
To compute any geodesic starting from a source q_0, we simply need a shooting momentum p_0 and a simplistic Euler scheme :

$$\begin{cases} q_{t+0.1} & = & q_t & + & 0.1 \cdot K_{q_t} p_t \\ p_{t+0.1} & = & p_t & - & 0.1 \cdot \partial_q (p_t, K_q p_t)(q_t) \end{cases} . \tag{25}$$

Exponential map :

$$\mathrm{Exp}_{q_0} : p_0 \in T^\star_{q_0}\mathcal{M} \; \mapsto \; q_1 \in \mathcal{M} \tag{26}$$

It works !



(a) 2D parametrization.

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

Figure 13: Geodesics on the donut-shaped torus.

Lessons taught by the Hamiltonian theory of geodesics

We are looking for :

- Tearing-adverse metrics on the space of landmarks
- Efficient ways to compute geodesics (deformations)

Lessons taught by the Hamiltonian theory of geodesics

We are looking for :

- Tearing-adverse metrics on the space of landmarks
- Efficient ways to compute geodesics (deformations)

Hamilton has taught us that :

- Geodesics are "simple" iff the cometric $K_q = g_q^{-1}$ is simple
- The Exponential map can be computed efficiently

The LDDMM framework

Kernel cometrics and Diffeomorphic trajectories

Parallelism is the way forward

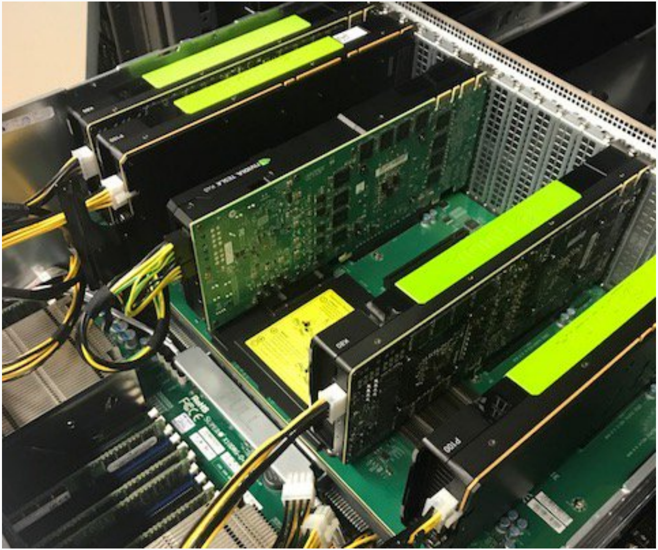


Figure 14: Highly-parallel MoKaMachine (Mokaplan Inria team).

GPUs in action

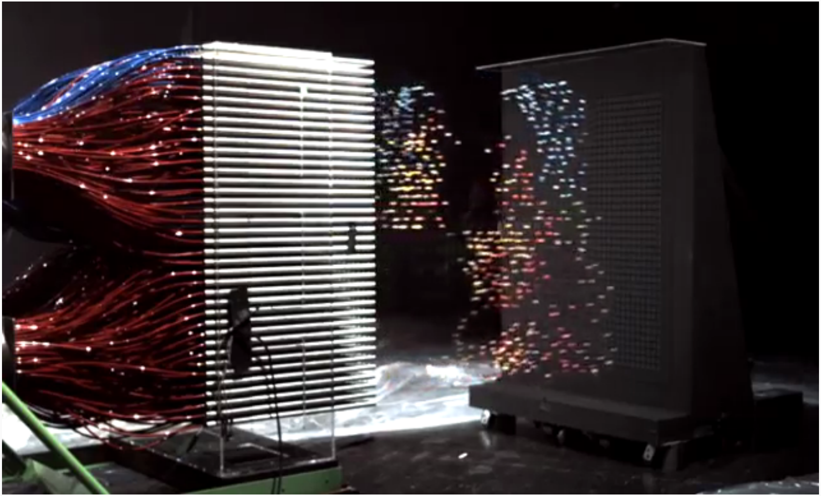


Figure 15: *Mythbusters Demo GPU versus CPU*, taken from the Nvidia YouTube channel.

Kernel cometrics, full isotropic tensor

Use a blockwise kernel matrix :

$$K_q = \left(\begin{array}{c|c|c|c} k(q^1, q^1) I_D & k(q^1, q^2) I_D & \cdots & k(q^1, q^M) I_D \\ \hline k(q^2, q^1) I_D & k(q^2, q^2) I_D & \cdots & k(q^2, q^M) I_D \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline k(q^M, q^1) I_D & k(q^M, q^2) I_D & \cdots & k(q^M, q^M) I_D \end{array}\right) \tag{27}$$

with $k(x, y)$ a kernel function (Gaussian, Cauchy, etc.).

Kernel cometrics, reduced tensor

That is, use a reduced correlation matrix

$$k_q = \begin{pmatrix} k(q^1, q^1) & k(q^1, q^2) & \cdots & k(q^1, q^M) \\ k(q^2, q^1) & k(q^2, q^2) & \cdots & k(q^2, q^M) \\ \vdots & \vdots & \ddots & \vdots \\ k(q^M, q^1) & k(q^M, q^2) & \cdots & k(q^M, q^M) \end{pmatrix} \tag{28}$$

so that

$$H(q, p) \;=\; \frac{1}{2} p^\mathsf{T} K_q p \;=\; \frac{1}{2} \sum_{i,j=1}^{M} k(q^i, q^j) \cdot (p^i)^\mathsf{T} p^j. \tag{29}$$

Translation-invariant kernels and convolution

In practice, we take

$$k(x, y) \;=\; k(\|x - y\|) \tag{30}$$

so that

$$\sum_{i,j=1}^{M} k(q^i, q^j) \cdot (p^i)^{\intercal} p^j \;=\; \sum_{i,j=1}^{M} k(q^i - q^j) \cdot \left\langle p^i, p^j \right\rangle \tag{31}$$

$$= \langle p, k \star p \rangle \tag{32}$$

with

$$p \;=\; \sum_{i=1}^{M} p^i \delta_{q^i}. \tag{33}$$

Translation-invariant kernels and convolution

In practice, we take

$$k(x,y) \;=\; k(\|x-y\|) \tag{30}$$

so that

$$\sum_{i,j=1}^{M} k(q^i,q^j)\cdot(p^i)^\top p^j \;=\; \sum_{i,j=1}^{M} k(q^i-q^j)\cdot\left\langle p^i,p^j\right\rangle \tag{31}$$

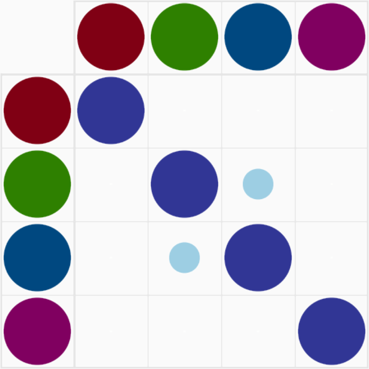
$$= \langle p, k \star p\rangle \tag{32}$$

with

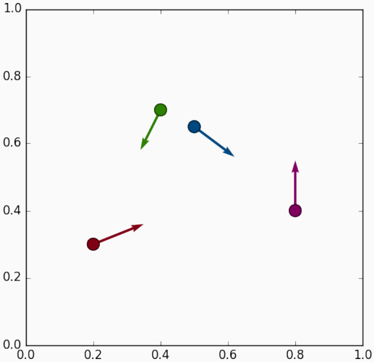
$$p \;=\; \sum_{i=1}^{M} p^i \delta_{q^i}. \tag{33}$$

In a computational sense, this is the simplest family of cometrics on the space of landmarks.

Influence of the kernel width, $\sigma = .25$



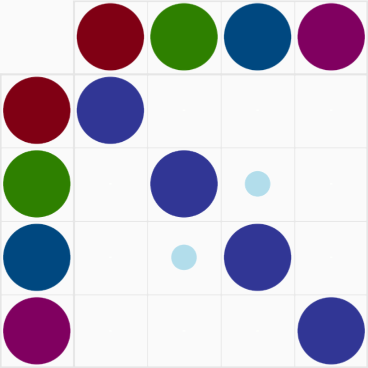
(a) Kernel matrix k_{q_t}.



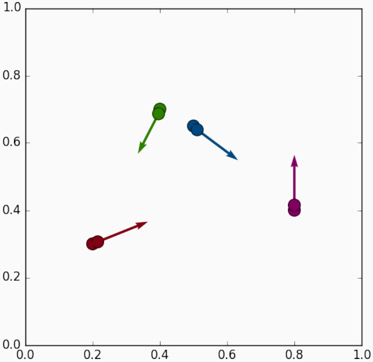
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



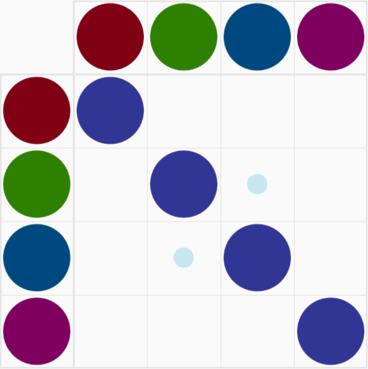
(a) Kernel matrix k_{q_t}.



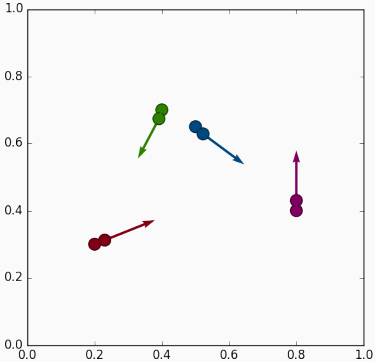
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



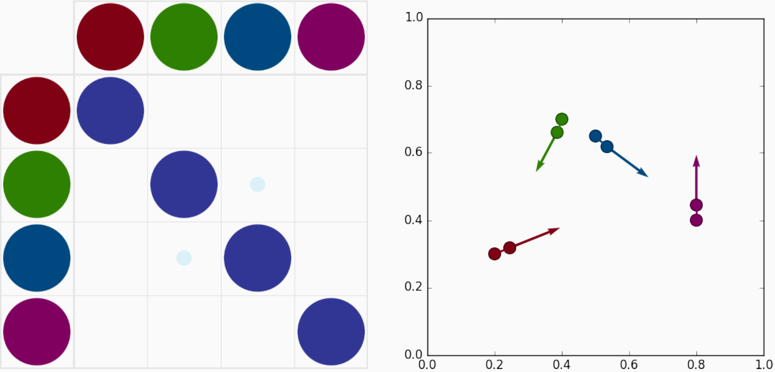
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .25$.

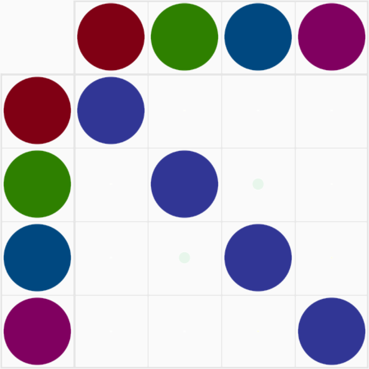
Influence of the kernel width, $\sigma = .25$



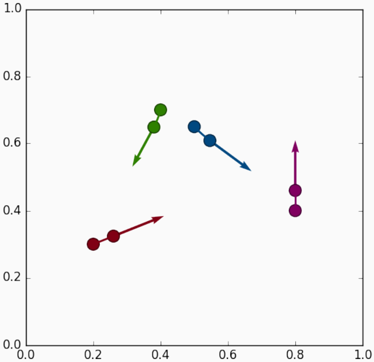
(a) Kernel matrix k_{q_t}. **(b)** Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



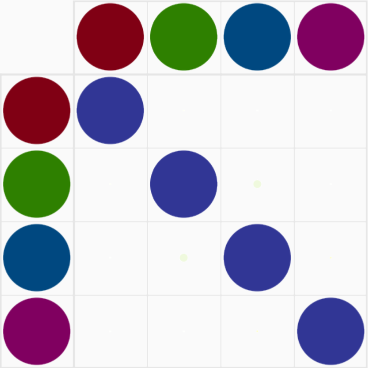
(a) Kernel matrix k_{q_t}.



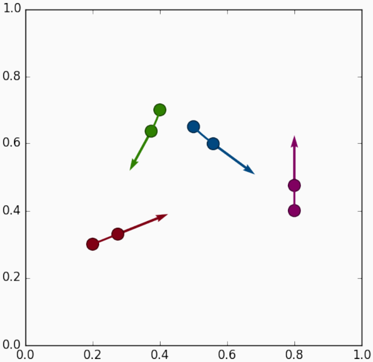
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x - y) = \exp(-\|x - y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



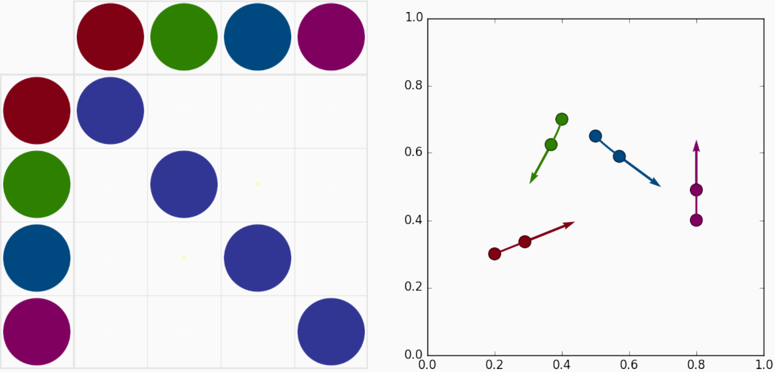
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

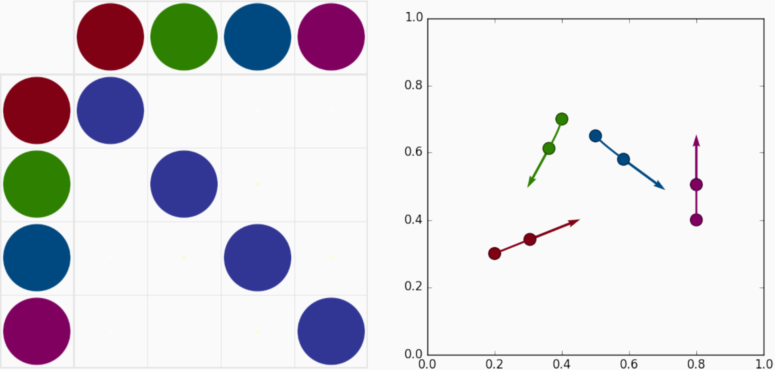
Influence of the kernel width, $\sigma = .25$



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

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$

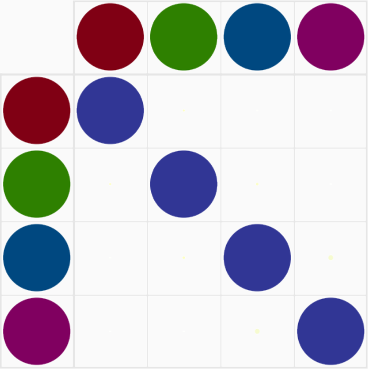


(a) Kernel matrix k_{q_t}.

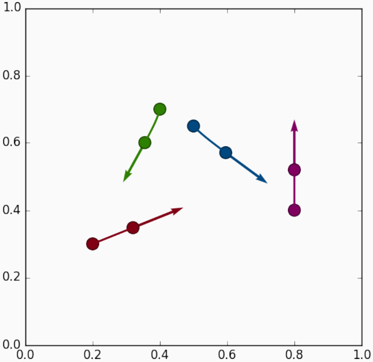
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



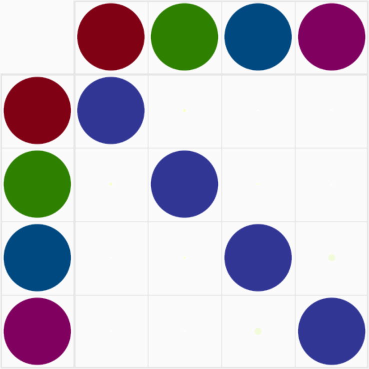
(a) Kernel matrix k_{q_t}.



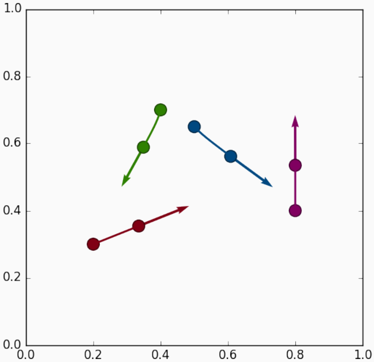
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



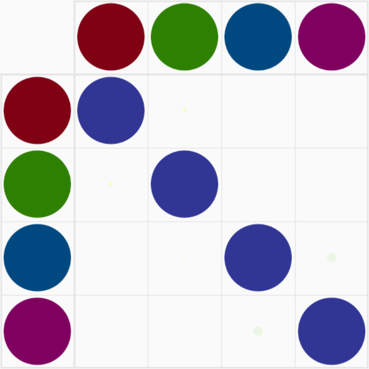
(a) Kernel matrix k_{q_t}.



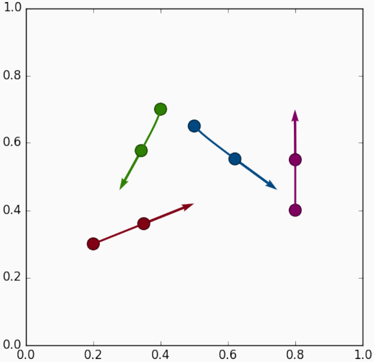
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



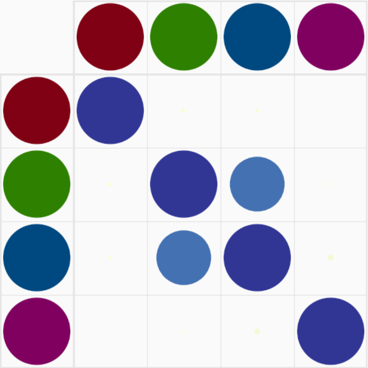
(a) Kernel matrix k_{q_t}.



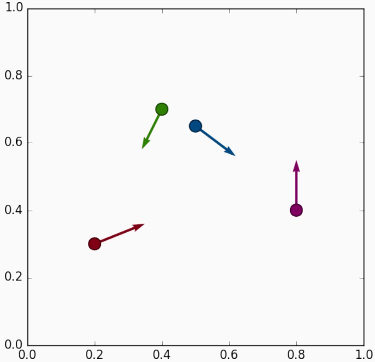
(b) Shooted cloud (q_t, p_t).

Figure 16: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

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

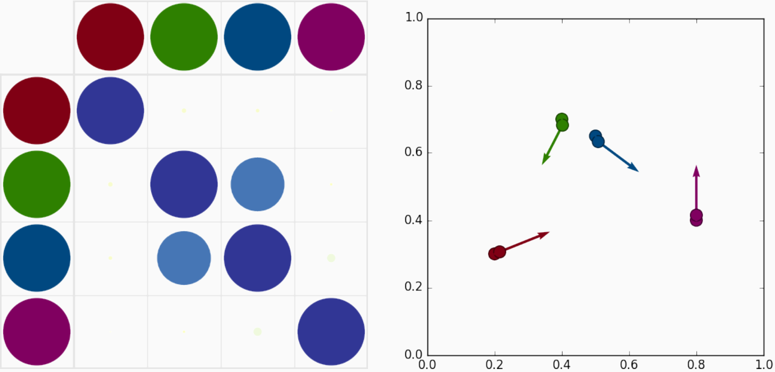


(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

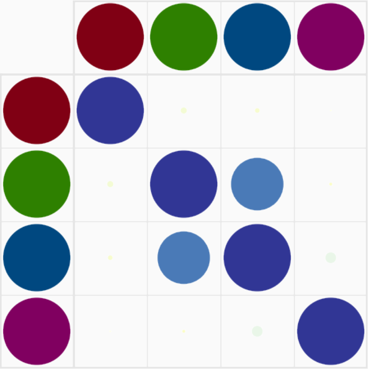


(a) Kernel matrix k_{q_t}.

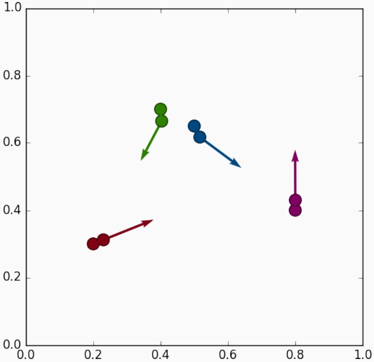
(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

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



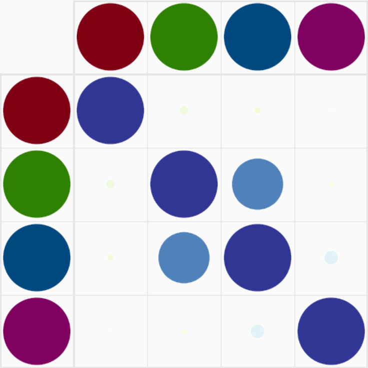
(a) Kernel matrix k_{q_t}.



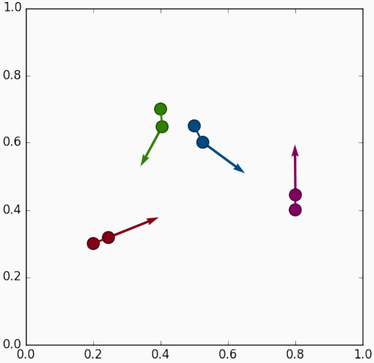
(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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

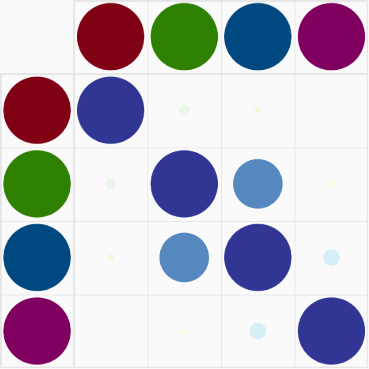


(a) Kernel matrix k_{q_t}.

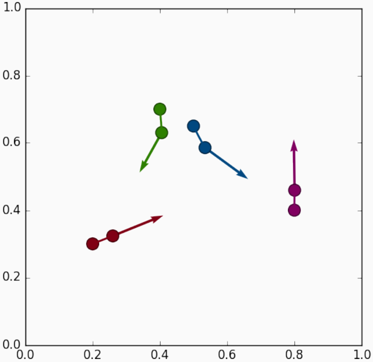


(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.



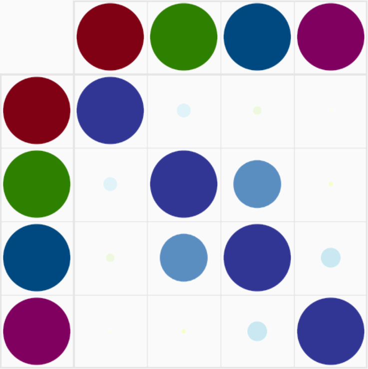
(a) Kernel matrix k_{q_t}.



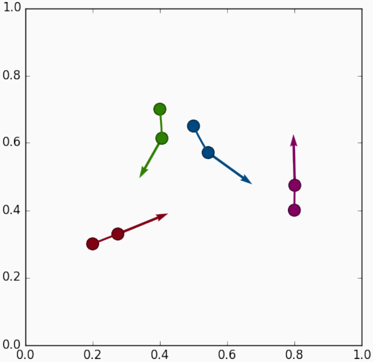
(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

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



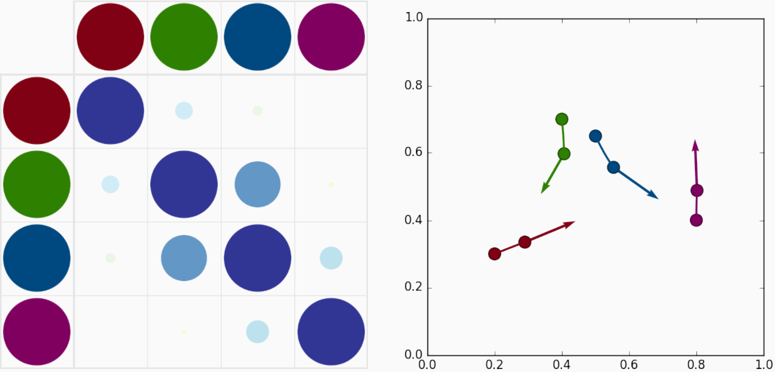
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

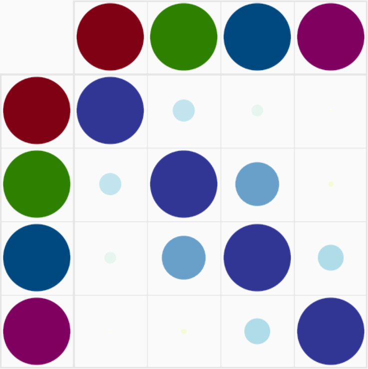
Influence of the kernel width, $\sigma = .35$



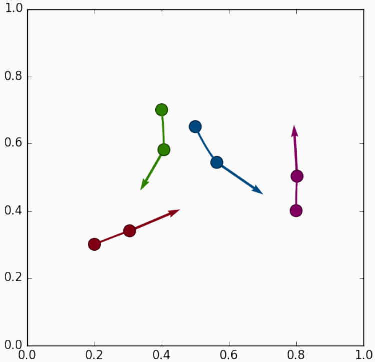
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

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



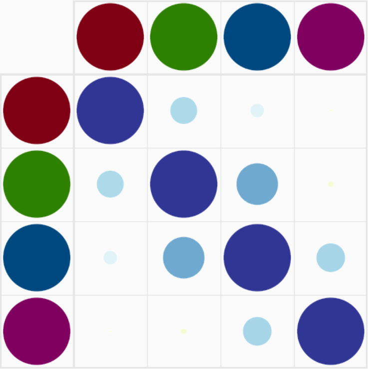
(a) Kernel matrix k_{q_t}.



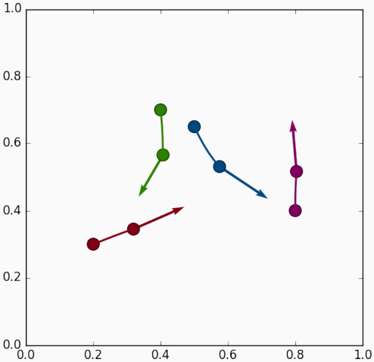
(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

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



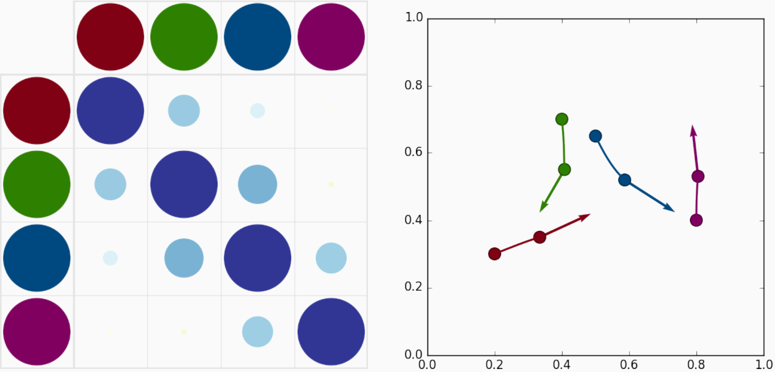
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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

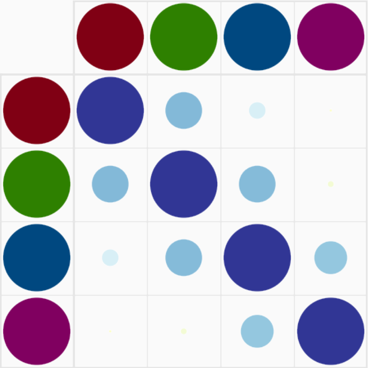


(a) Kernel matrix k_{q_t}.

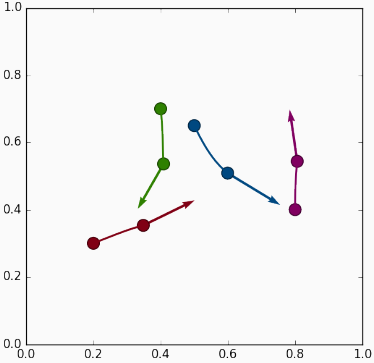
(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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



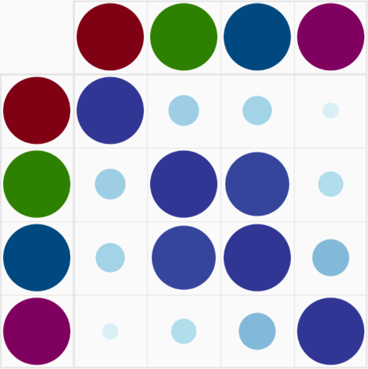
(a) Kernel matrix k_{q_t}.



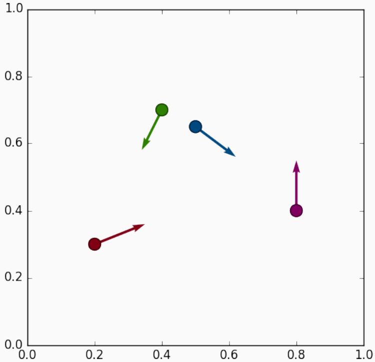
(b) Shooted cloud (q_t, p_t).

Figure 17: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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



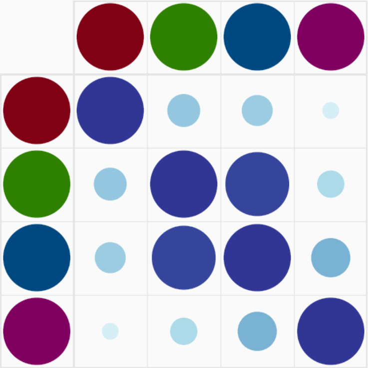
(a) Kernel matrix k_{q_t}.



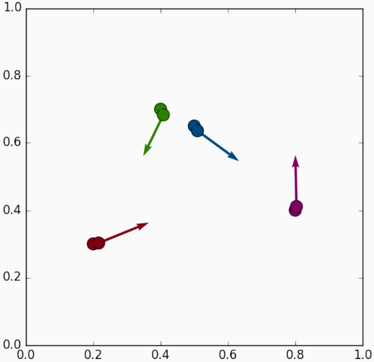
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



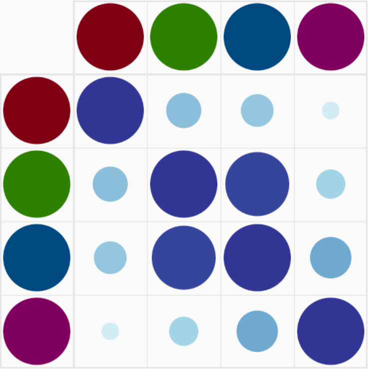
(a) Kernel matrix k_{q_t}.



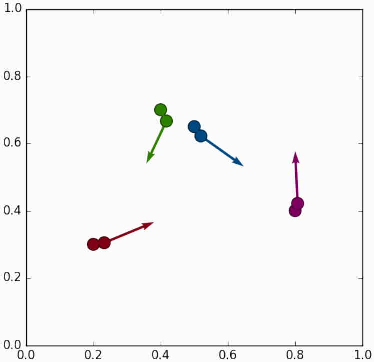
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



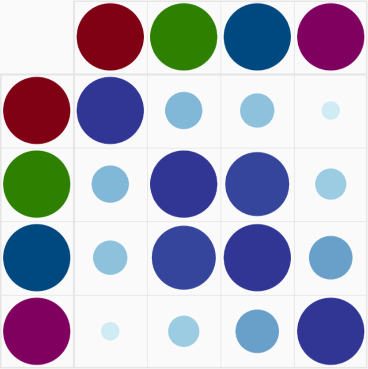
(a) Kernel matrix k_{q_t}.



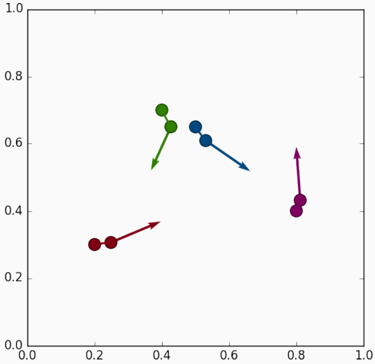
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

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



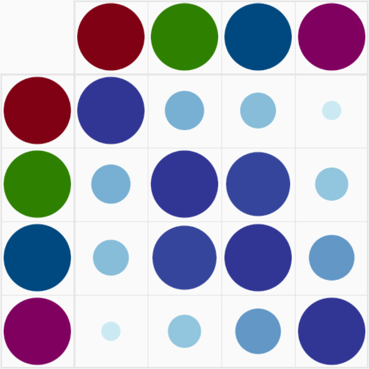
(a) Kernel matrix k_{q_t}.



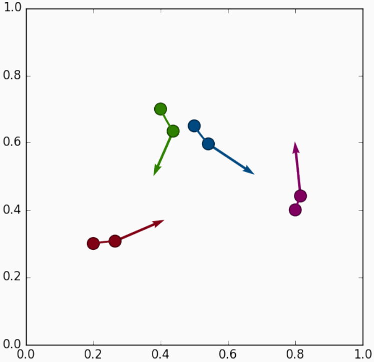
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

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



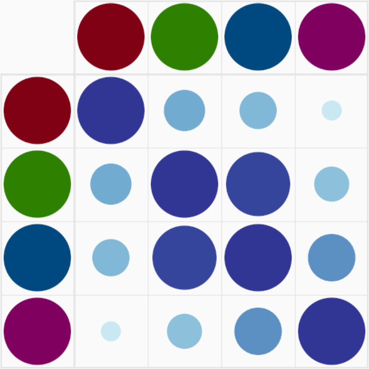
(a) Kernel matrix k_{q_t}.



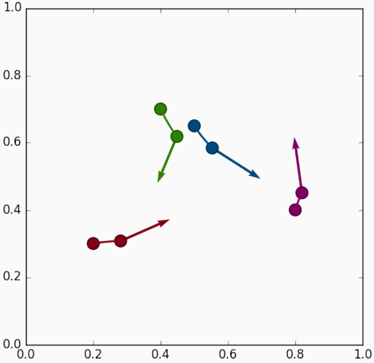
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = \texttt{.50}$.

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



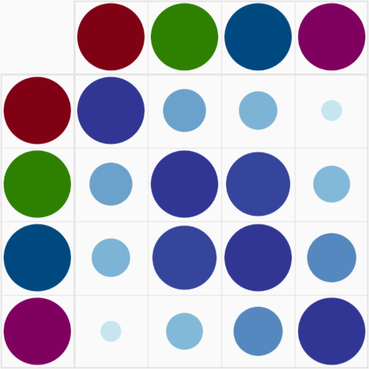
(a) Kernel matrix k_{q_t}.



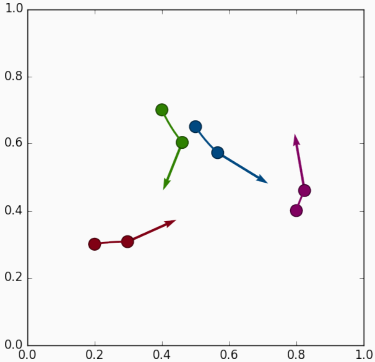
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



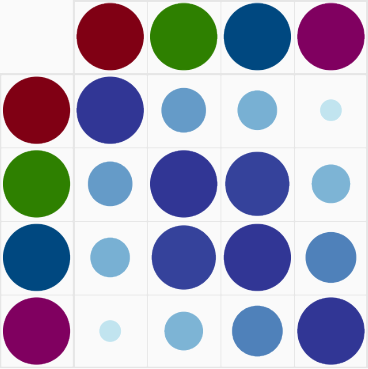
(a) Kernel matrix k_{q_t}.



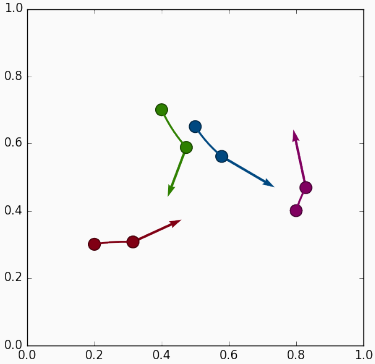
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



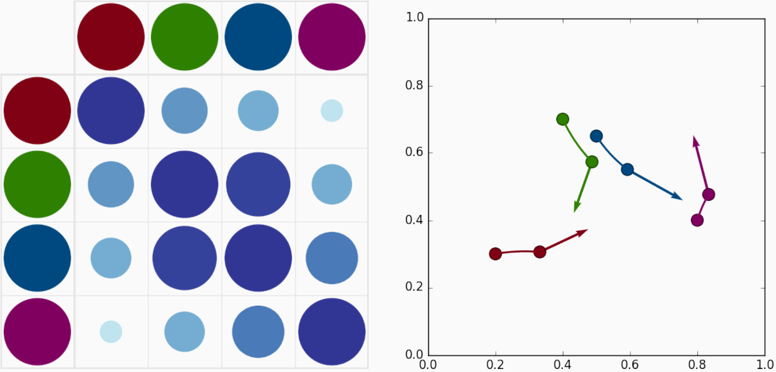
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

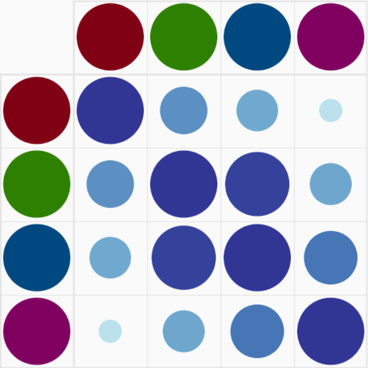
Influence of the kernel width, $\sigma = .50$



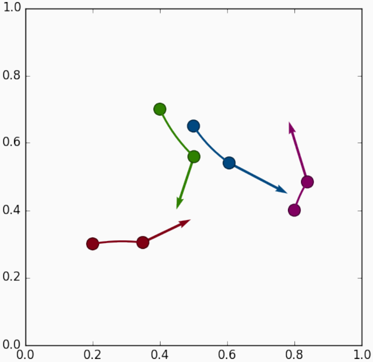
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



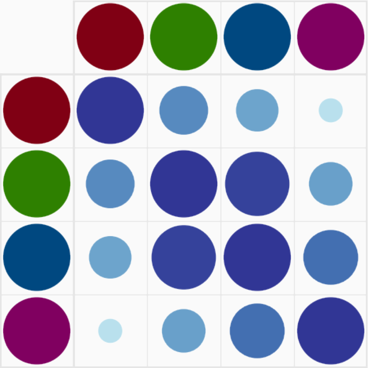
(a) Kernel matrix k_{q_t}.



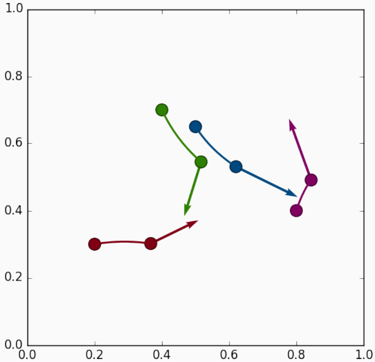
(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

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



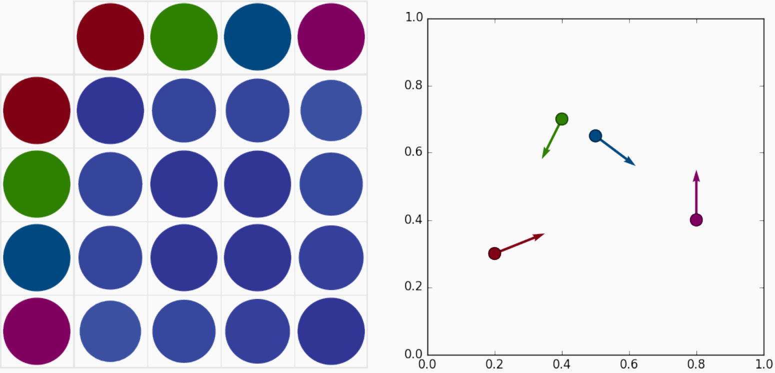
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 18: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

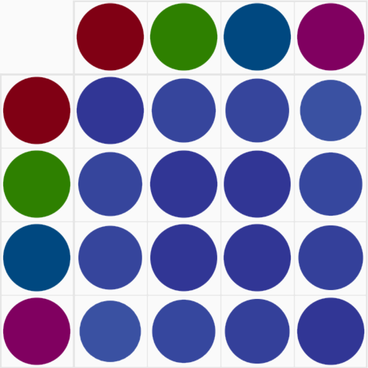
Influence of the kernel width, $\sigma = 1.$



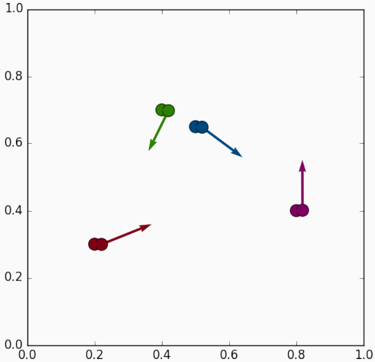
(a) Kernel matrix k_{q_t}. **(b)** Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

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



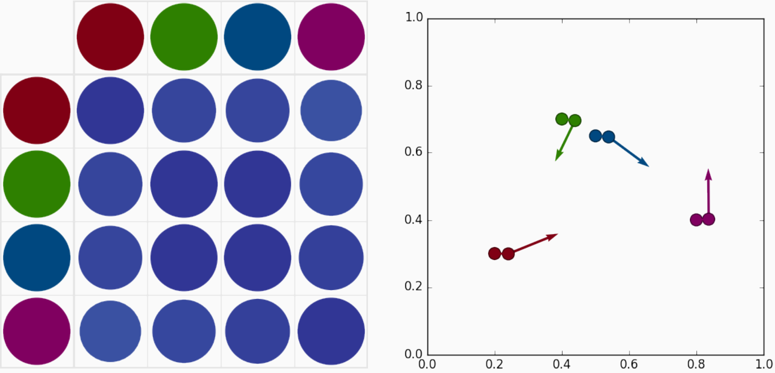
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1..$

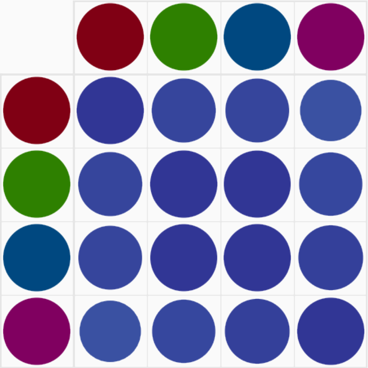
Influence of the kernel width, $\sigma = 1$.



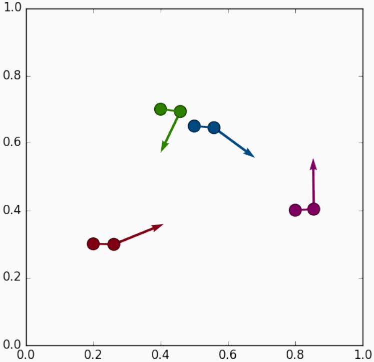
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

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



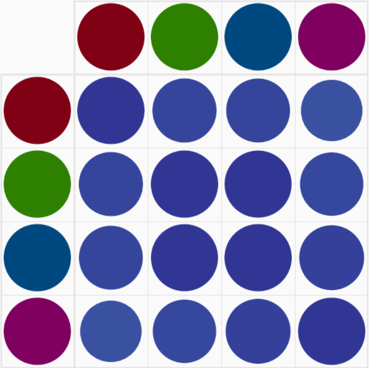
(a) Kernel matrix k_{q_t}.



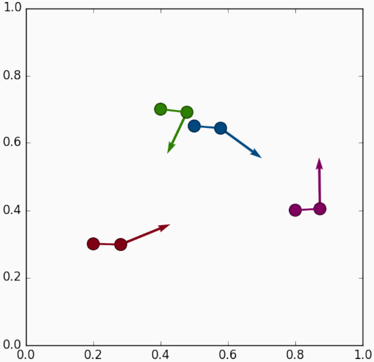
(b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1$..

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



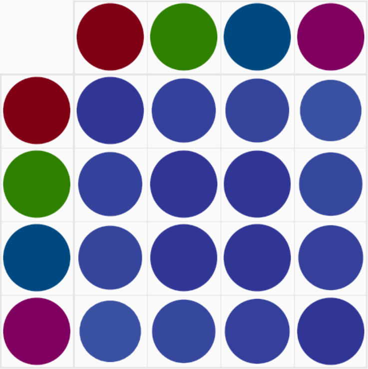
(a) Kernel matrix k_{q_t}.



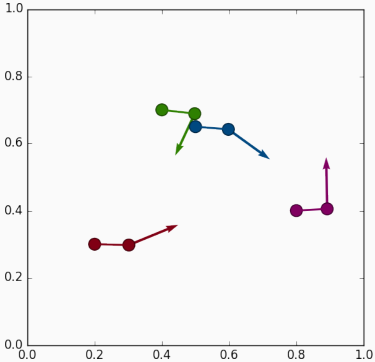
(b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

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



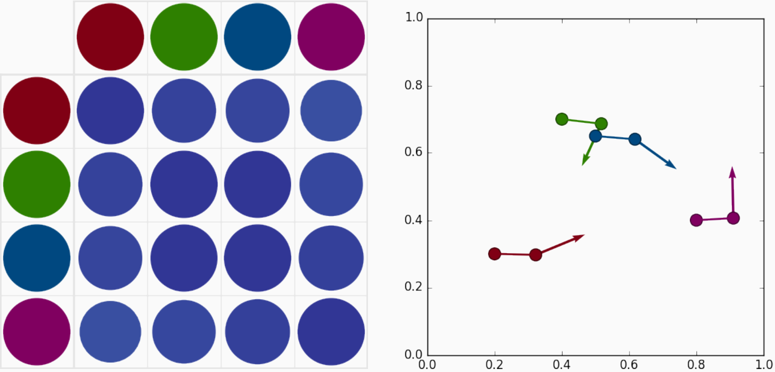
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

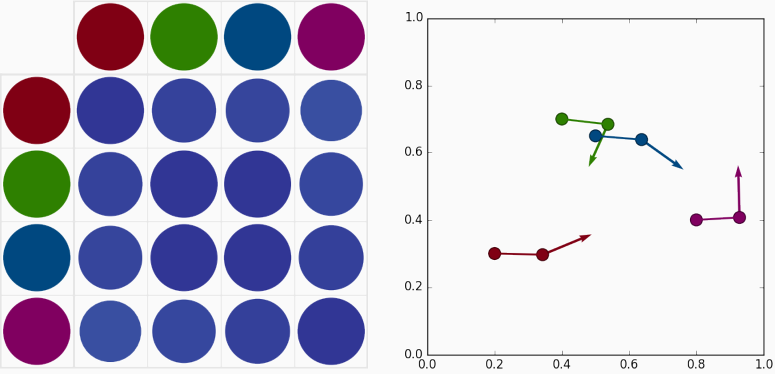
Influence of the kernel width, $\sigma = 1.$



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

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

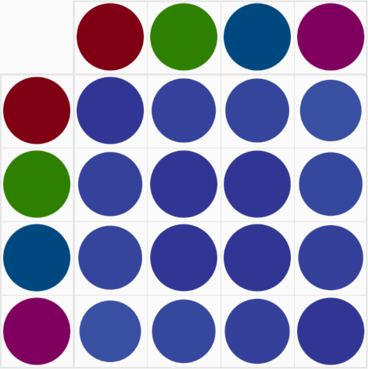
Influence of the kernel width, $\sigma = 1.$



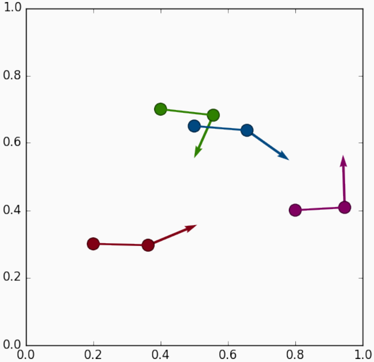
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

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



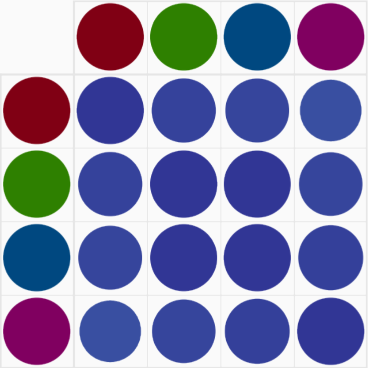
(a) Kernel matrix k_{q_t}.



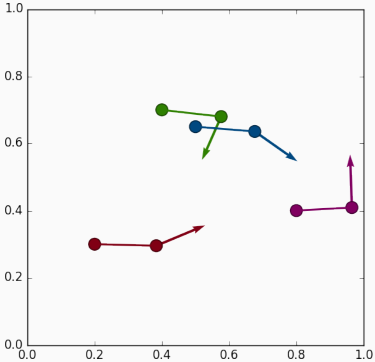
(b) Shooted cloud (q_t, p_t).

Figure 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

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



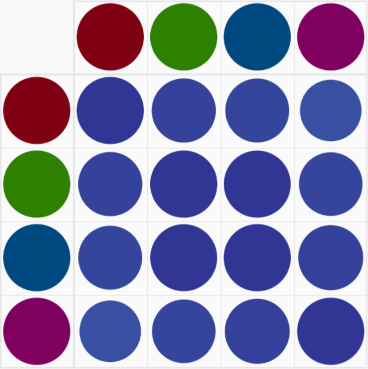
(a) Kernel matrix k_{q_t}.



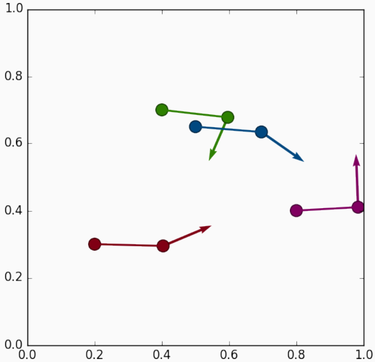
(b) Shooted cloud (q_t, p_t).

Figure 19: 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 19: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

RKHS norms on velocity vector fields

Let k be a smooth enough kernel function, with $\widehat{k}(\omega) \in \mathbb{R}_+^\star$.
If $v : \mathbb{R}^D \to \mathbb{R}^D$ is a vector field on the ambient space, define

$$\|v\|_k^2 \;=\; \int_{\omega \in \mathbb{R}^D} \frac{1}{\widehat{k}(\omega)}\, |\widehat{v}(\omega)|^2 \, \mathrm{d}\omega. \tag{34}$$

RKHS norms on velocity vector fields

Let k be a smooth enough kernel function, with $\widehat{k}(\omega) \in \mathbb{R}^\star_+$.
If $v : \mathbb{R}^D \to \mathbb{R}^D$ is a vector field on the ambient space, define

$$\|v\|_k^2 \;=\; \int_{\omega \in \mathbb{R}^D} \frac{1}{\widehat{k}(\omega)} \, |\widehat{v}(\omega)|^2 \, \mathrm{d}\omega. \tag{34}$$

- $V_k = \{v \mid \|v\|_k < \infty\}$ is a Hilbert space of *k-smooth* vector fields
- We assume k is smooth enough, so that $\delta_x : v \mapsto v(x)$ belongs to the dual space $(V_k)^\star$: we link with the theory of **Reproducing Kernel Hilbert Spaces**.

Integration of k-smooth vector flows

Assume that (v_t) is a time-varying vector field such that

$$\ell_k(v)^2 \;=\; \int_0^1 \|v_t\|_k^2 \,\mathrm{d}t \;<\; \infty. \tag{35}$$

Integration of k-smooth vector flows

Assume that (v_t) is a time-varying vector field such that

$$\ell_k(v)^2 \;=\; \int_0^1 \|v_t\|_k^2 \,\mathrm{d}t \;<\; \infty. \tag{35}$$

According to Picard-Lindelöf theorem, we can integrate the flow, find a unique trajectory ϕ_t of diffeomorphisms such that for every point $x \in \mathbb{R}^D$ and time $t \in [0,1]$:

$$\phi_0(x) = x \qquad \text{and} \qquad \frac{\mathrm{d}}{\mathrm{d}t}\left[\phi_t(x)\right] = v_t \circ \phi_t(x),$$

$$\text{i.e.} \qquad \phi_0 \;=\; \mathrm{Id}_{\mathbb{R}^D} \qquad \text{and} \qquad \phi_t \;=\; \int_{s=0}^{t} v_s \circ \phi_s \,\mathrm{d}s.$$

An infinite-dimensional matching problem

We define $G_k = \{\phi_1 \mid \cdots\}$ the set of diffeomorphisms obtained by integrating *finite-cost* vector flows $(v_t) \in L^2(V_k)$.

An infinite-dimensional matching problem

We define $G_k = \{\phi_1 \mid \cdots\}$ the set of diffeomorphisms obtained by integrating *finite-cost* vector flows $(v_t) \in L^2(V_k)$.

G_k is an infinite-dimensional Riemannian manifold modeled on V_k.

An infinite-dimensional matching problem

We define $G_k = \{\phi_1 \mid \cdots\}$ the set of diffeomorphisms obtained by integrating *finite-cost* vector flows $(v_t) \in L^2(V_k)$.

G_k is an infinite-dimensional Riemannian manifold modeled on V_k. As diffeomorphisms carry around images and measures, we try to minimize

$$C^2(\phi_1) \;=\; \ell_k(v)^2 \;=\; \int_0^1 \|v_t\|_k^2 \,\mathrm{d}t \;<\; \infty \tag{36}$$

under the constraint that

$$X \xrightarrow{\phi_1} Y. \tag{37}$$

The kernel and diffeomorphic geodesics coincide

Reduction Principle

Let q_t be a time-dependent point cloud, k a kernel function. Then, the two propositions below are equivalent :

i) q_t is a geodesic for the kernel cometric K_q, with momentum p_t associated to the Hamiltonian

$$H(q, p) \;=\; \frac{1}{2} p^\intercal K_q p. \tag{38}$$

ii) q_t is carried around by a locally optimal diffeomorphic trajectory ϕ_t, and we have

$$v_t \;=\; k \star p_t \qquad \text{i.e.} \qquad v_t(x) \;=\; \sum_{m=1}^{M} k(q_t^m, x)\, p_t^m. \tag{39}$$

Hand-waving proof of the reduction principle, part 1

At any time t,

$$v_t \;=\; \arg\min\{\, \|v\|_k \mid \forall\, m, v(q_t^m) = v_t(q_t^m) \,\}\,. \tag{40}$$

Hand-waving proof of the reduction principle, part 1

At any time t,

$$v_t \;=\; \arg\min\{\, \|v\|_k \;|\; \forall\, m, v(q_t^m) = v_t(q_t^m)\,\}\,. \tag{40}$$

Hence, as v_t does not have any superfluous component,

$$v_t \in \{\, v \mid \forall\, m, v(q_t^m) = 0\,\}^{\perp_k} \tag{41}$$

$$\text{i.e.} \quad v_t \in \left(\bigcap_{m=1}^{M} \left\{\, v \mid \left\langle \delta_{q_t^m}, v \right\rangle = 0 \,\right\} \right)^{\perp_k}. \tag{42}$$

Hand-waving proof of the reduction principle, part 1

At any time t,

$$v_t = \arg\min\{ \|v\|_k \mid \forall\, m, v(q_t^m) = v_t(q_t^m) \}. \tag{40}$$

Hence, as v_t does not have any superfluous component,

$$v_t \in \{ v \mid \forall\, m, v(q_t^m) = 0 \}^{\perp_k} \tag{41}$$

$$\text{i.e.} \quad v_t \in \left(\bigcap_{m=1}^{M} \left\{ v \mid \left\langle \delta_{q_t^m}, v \right\rangle = 0 \right\} \right)^{\perp_k}. \tag{42}$$

But we also know that :

$$\left\langle k \star \delta_{q_t^m}, v \right\rangle_k = \int_{\omega \in \mathbb{R}^D} \frac{1}{\widehat{k}(\omega)} \overline{\widehat{k \star \delta_{q_t^m}}(\omega)} \cdot \widehat{v}(\omega) \;\mathrm{d}\omega \tag{43}$$

$$= \int_{\omega \in \mathbb{R}^D} \overline{\widehat{\delta_{q_t^m}}(\omega)} \cdot \widehat{v}(\omega) \;\mathrm{d}\omega \tag{44}$$

$$= \left\langle \delta_{q_t^m}, v \right\rangle = v(q_t^m). \tag{45}$$

Hand-waving proof of the reduction principle, part 2

Hence why, at any time t,

$$v_t \in \left(\bigcap_{m=1}^{M} \left\{ v \mid \left\langle k \star \delta_{q_t^m}, v \right\rangle_k = 0 \right\} \right)^{\perp_k} \tag{46}$$

$$= \bigcup_{m=1}^{M} \left(k \star \delta_{q_t^m} \right)^{\perp_k \perp_k} \tag{47}$$

$$= \mathrm{Vect}\left(k \star \delta_{q_t^m}, \; m \in [\![1, M]\!] \right). \tag{48}$$

Hand-waving proof of the reduction principle, part 2

Hence why, at any time t,

$$v_t \in \left(\bigcap_{m=1}^{M} \left\{ v \mid \left\langle k \star \delta_{q_t^m}, v \right\rangle_k = 0 \right\} \right)^{\perp_k} \tag{46}$$

$$= \bigcup_{m=1}^{M} \left(k \star \delta_{q_t^m} \right)^{\perp_k \perp_k} \tag{47}$$

$$= \mathrm{Vect}\left(k \star \delta_{q_t^m},\ m \in [\![1, M]\!] \right). \tag{48}$$

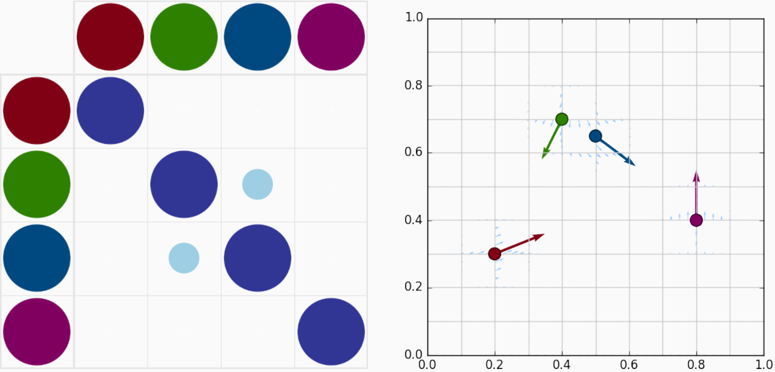
So, one can write

$$v_t = k \star \left(\sum_{m=1}^{M} p_t^m \, \delta_{q_t^m} \right) = k \star p_t, \tag{49}$$

and

$$\|v_t\|_k^2 = \left\langle k \star p_t, k^{(-1)} \star k \star p_t \right\rangle = \langle k \star p_t, p_t \rangle = p_t^\intercal K_{q_t} p_t. \tag{50}$$

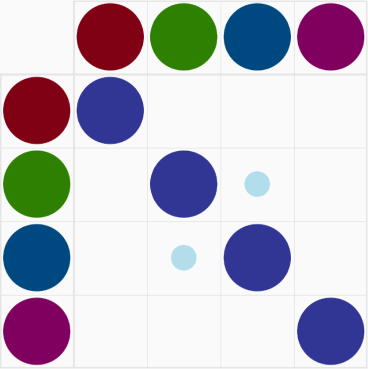
Influence of the kernel width, $\sigma = .25$



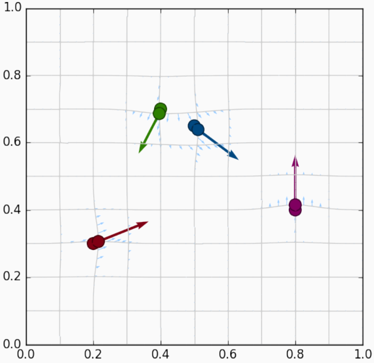
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



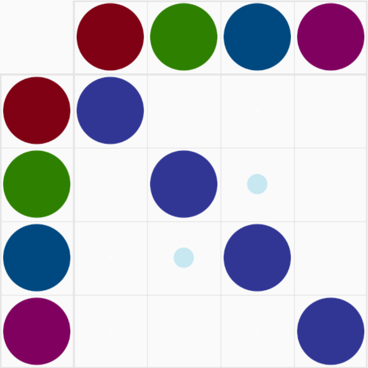
(a) Kernel matrix k_{q_t}.



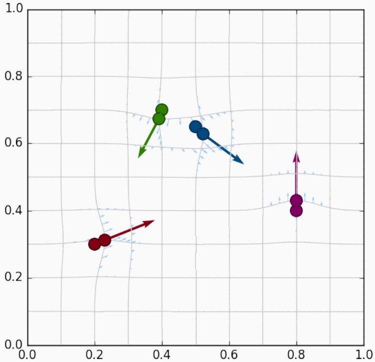
(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



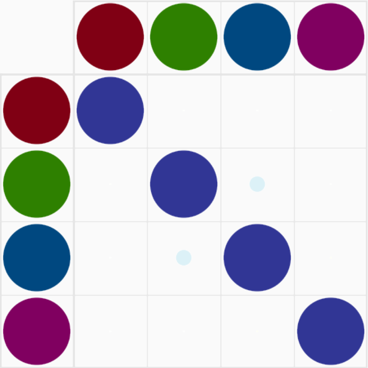
(a) Kernel matrix k_{q_t}.



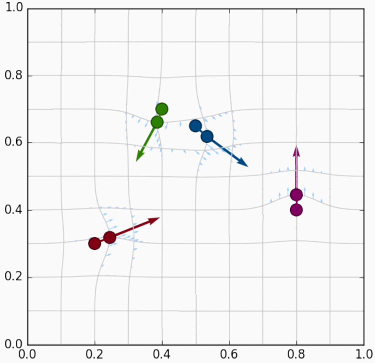
(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



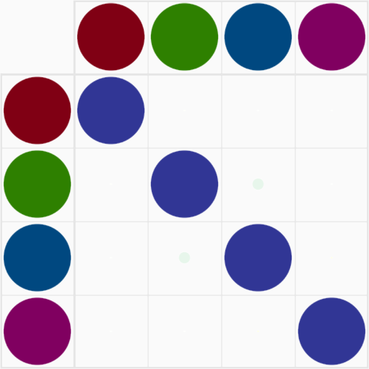
(a) Kernel matrix k_{q_t}.



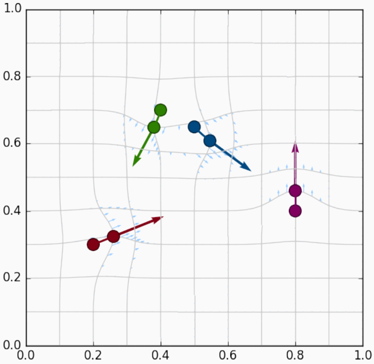
(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x - y) = \exp(-\|x - y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



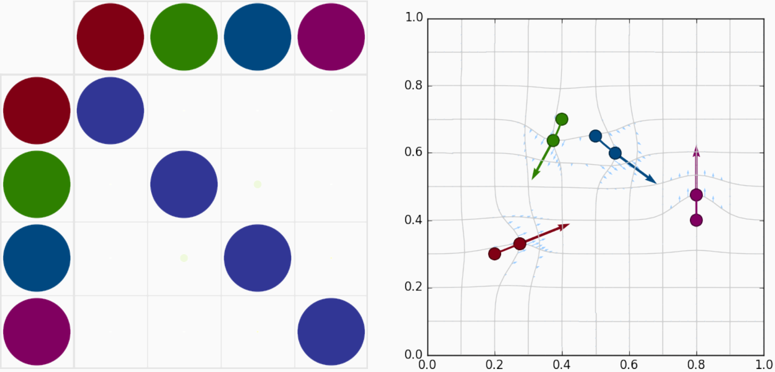
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

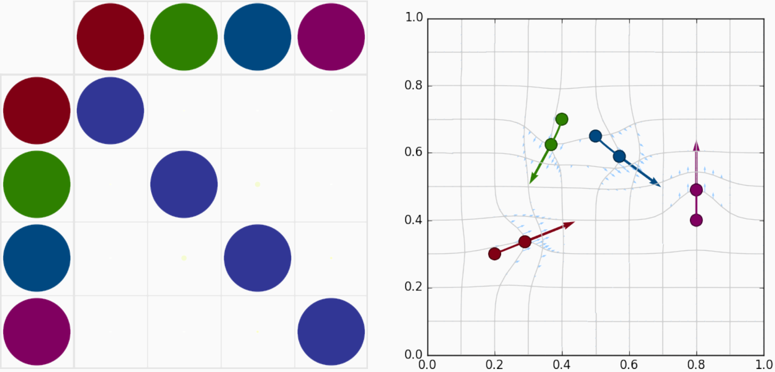
Influence of the kernel width, $\sigma = .25$



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

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$

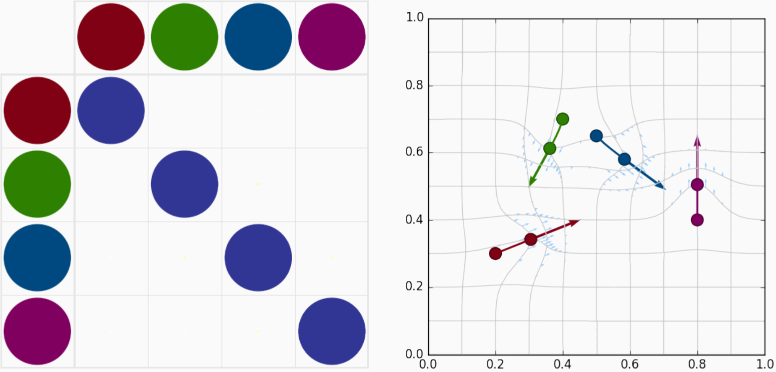


(a) Kernel matrix k_{q_t}.

(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .25$.

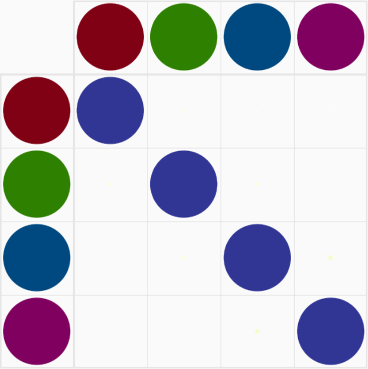
Influence of the kernel width, $\sigma = .25$



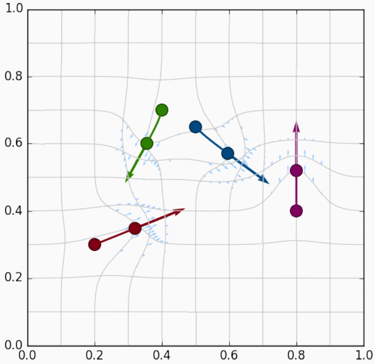
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$

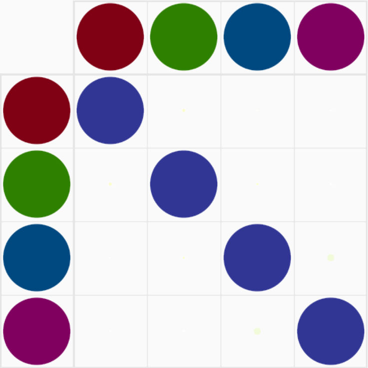


(a) Kernel matrix k_{q_t}.

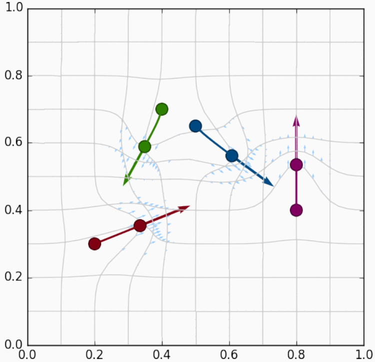


(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.



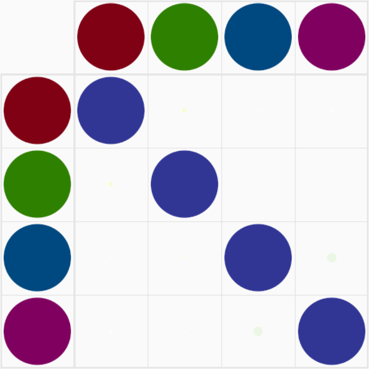
(a) Kernel matrix k_{q_t}.



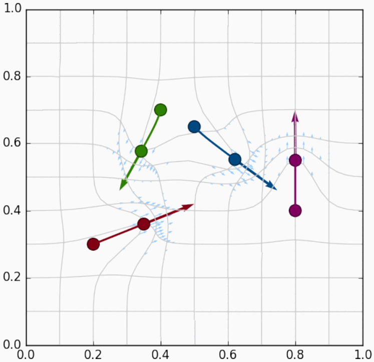
(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .25$.

Influence of the kernel width, $\sigma = .25$



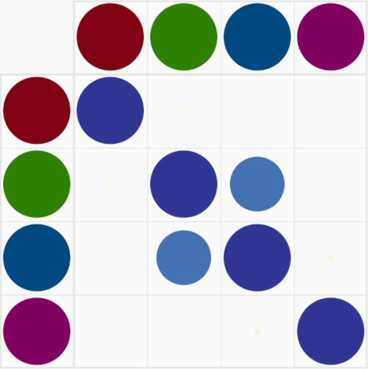
(a) Kernel matrix k_{q_t}.



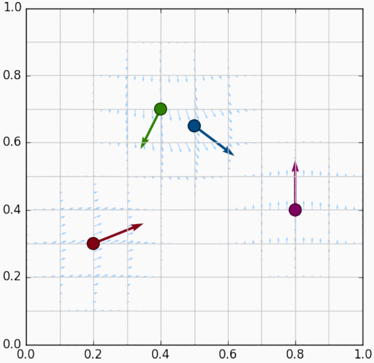
(b) Shooted cloud (q_t, p_t).

Figure 20: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .25$.

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



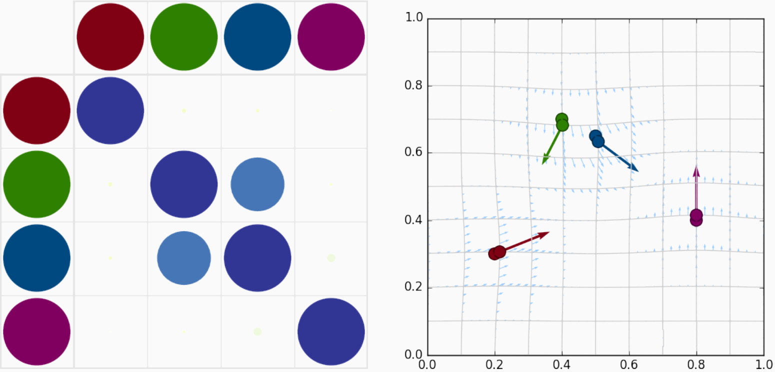
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

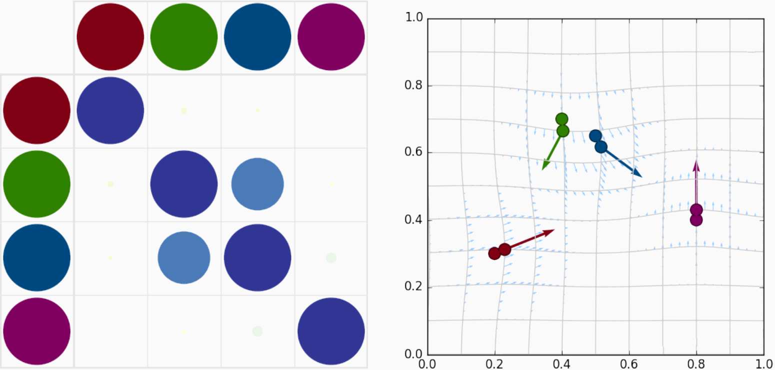
Influence of the kernel width, $\sigma = .35$



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

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

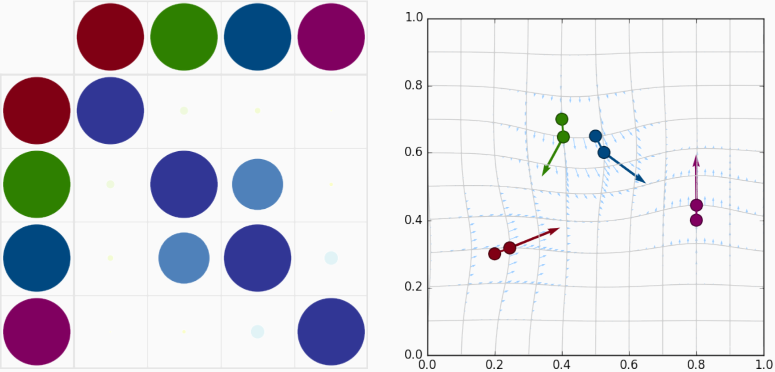
Influence of the kernel width, $\sigma = .35$



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

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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

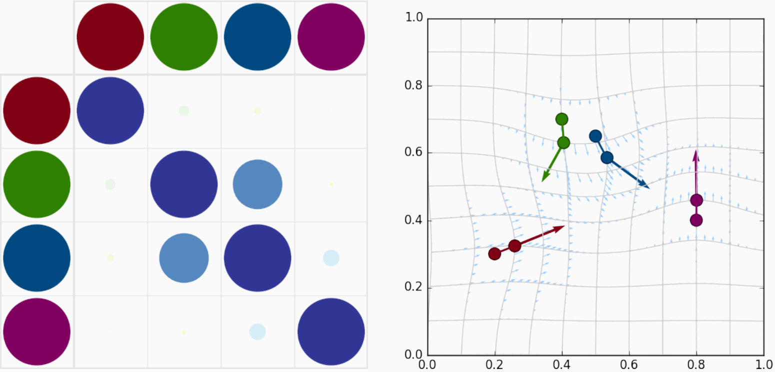


(a) Kernel matrix k_{q_t}.

(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

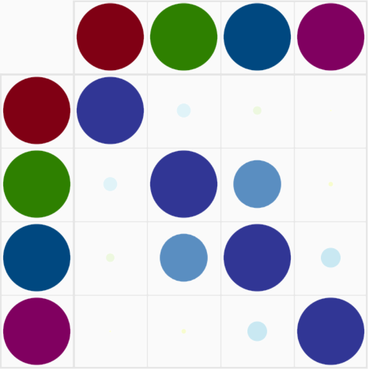
Influence of the kernel width, $\sigma = .35$



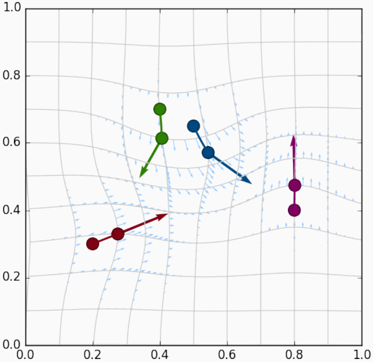
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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



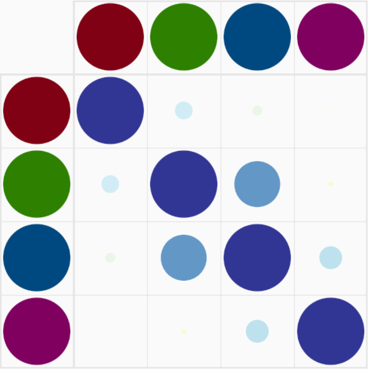
(a) Kernel matrix k_{q_t}.



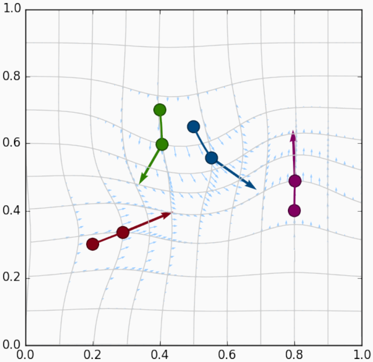
(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

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



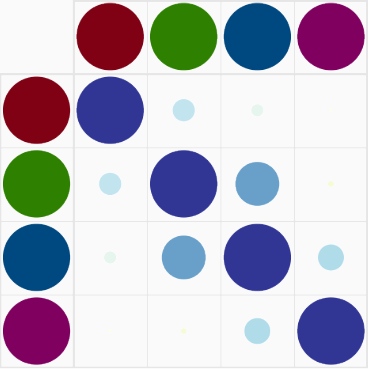
(a) Kernel matrix k_{q_t}.



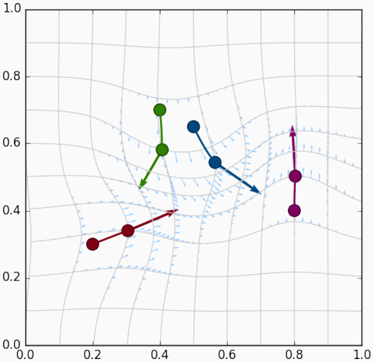
(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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



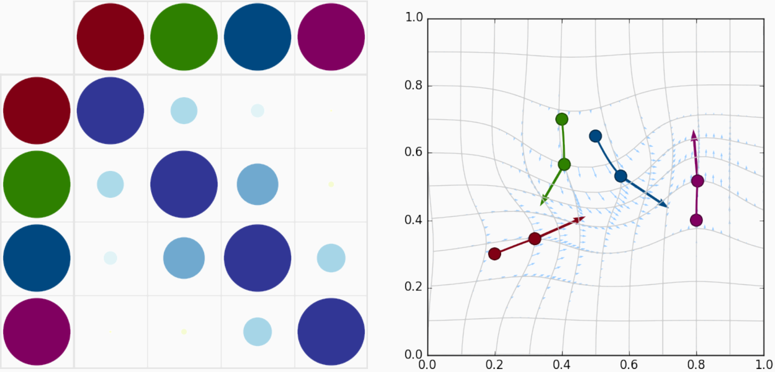
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

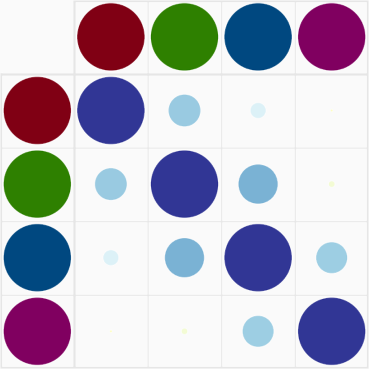
Influence of the kernel width, $\sigma = .35$



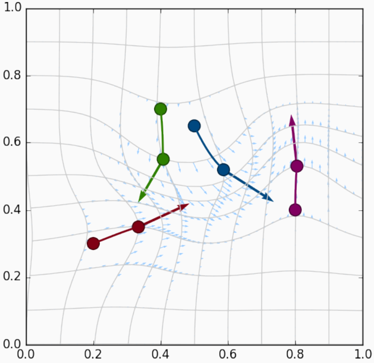
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

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



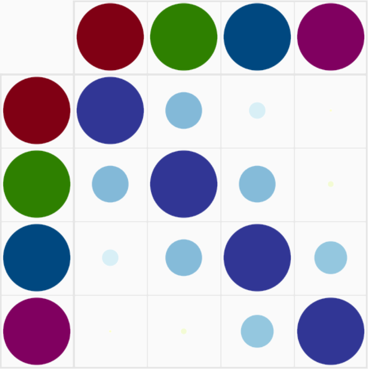
(a) Kernel matrix k_{q_t}.



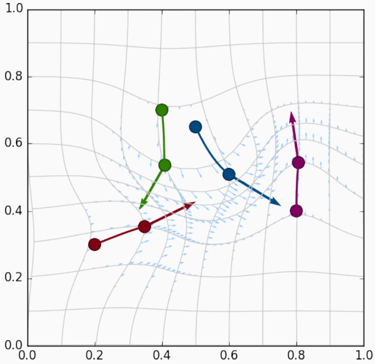
(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .35$.

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



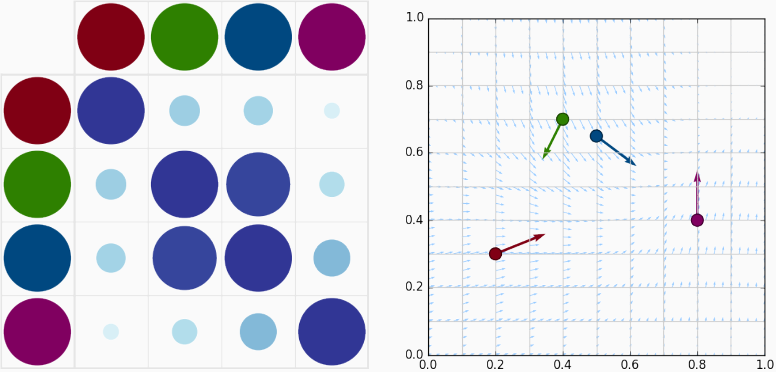
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 21: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .35$.

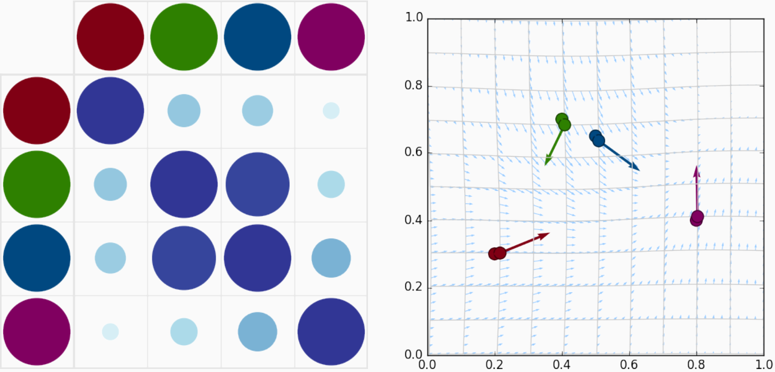
Influence of the kernel width, $\sigma = .50$



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

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

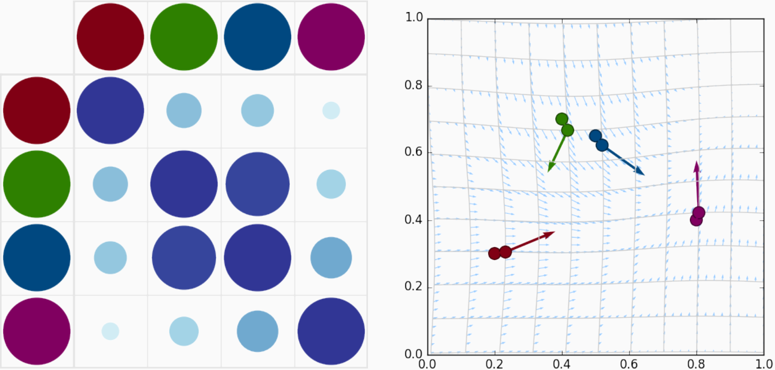
Influence of the kernel width, $\sigma = .50$



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

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

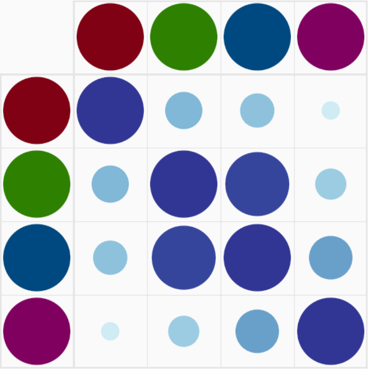
Influence of the kernel width, $\sigma = .50$



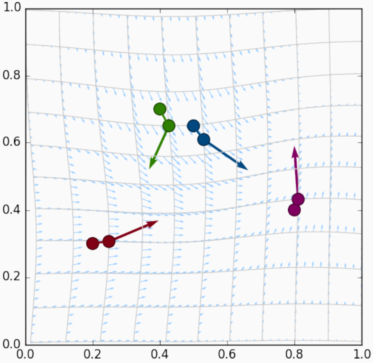
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

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



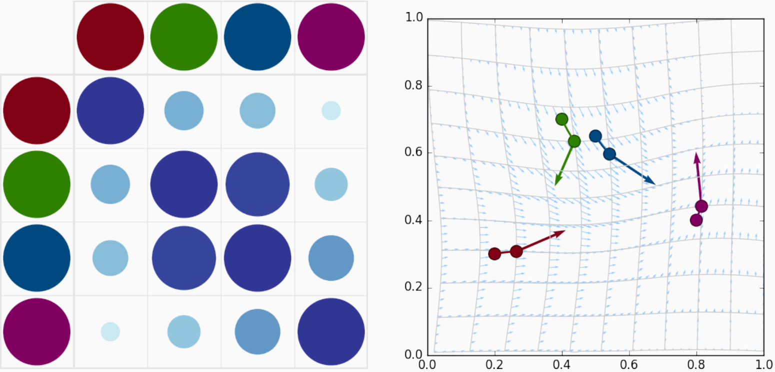
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

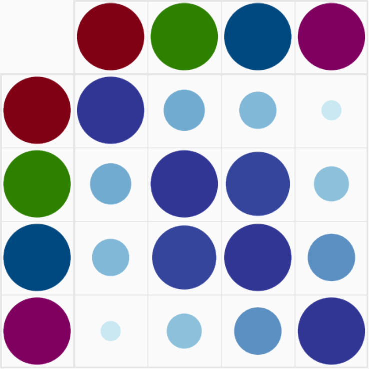
Influence of the kernel width, $\sigma = .50$



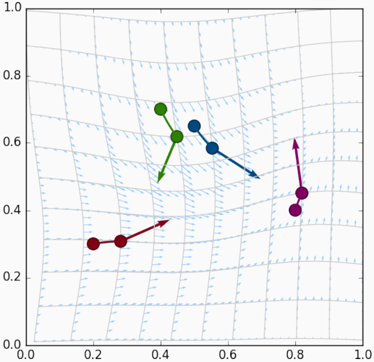
(a) Kernel matrix k_{q_t}.　(b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



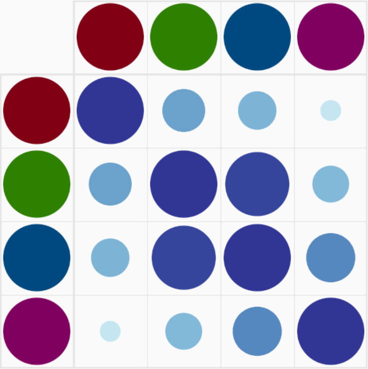
(a) Kernel matrix k_{q_t}.



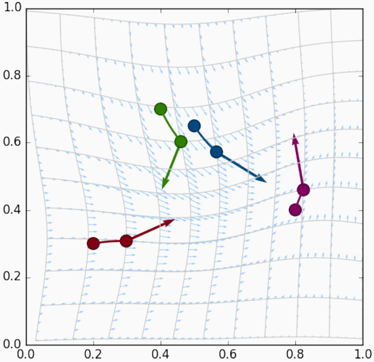
(b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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



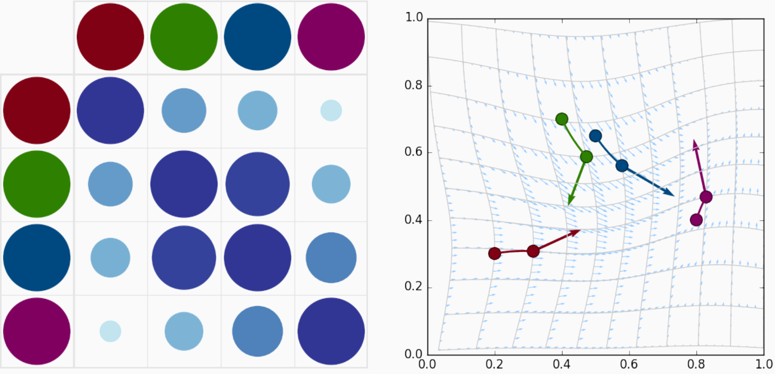
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

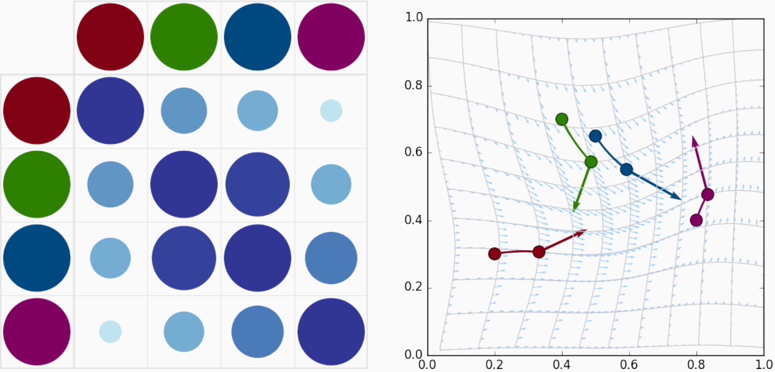
Influence of the kernel width, $\sigma = .50$



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

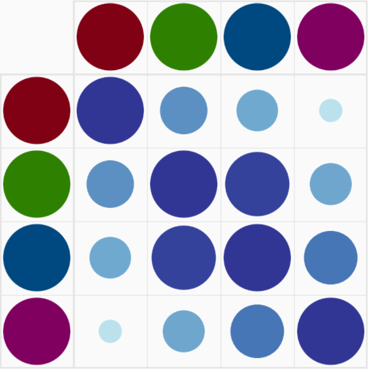
Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = .50$.

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

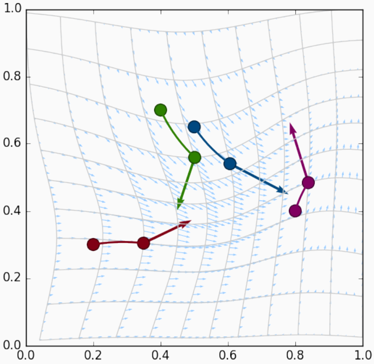


(a) Kernel matrix k_{q_t}. **(b)** Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.



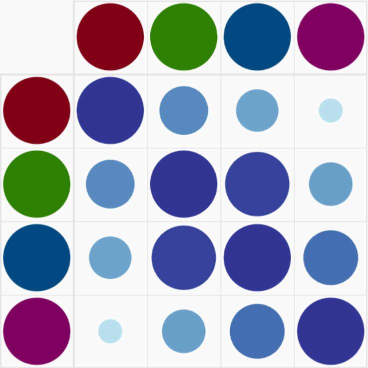
(a) Kernel matrix k_{q_t}.



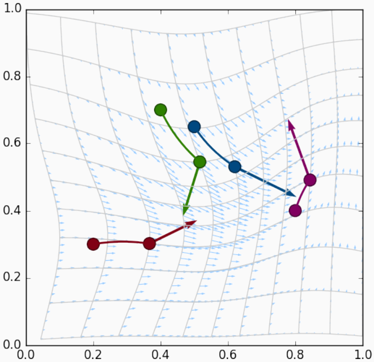
(b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

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



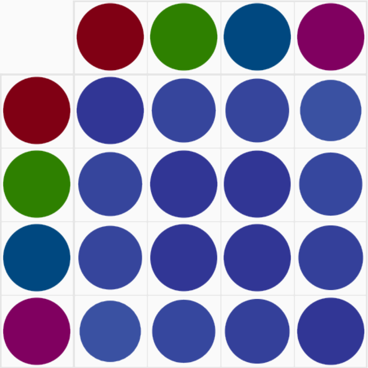
(a) Kernel matrix k_{q_t}.



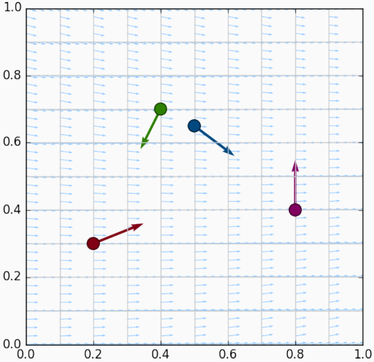
(b) Shooted cloud (q_t, p_t).

Figure 22: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = .50$.

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



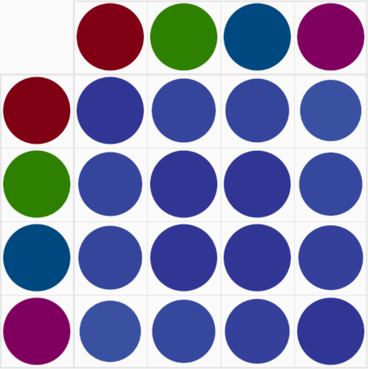
(a) Kernel matrix k_{q_t}.



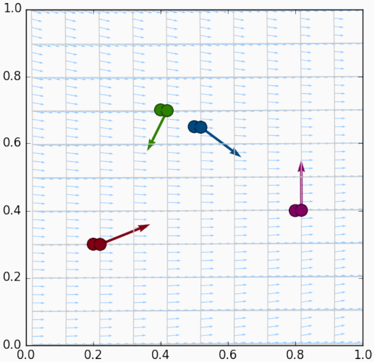
(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

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



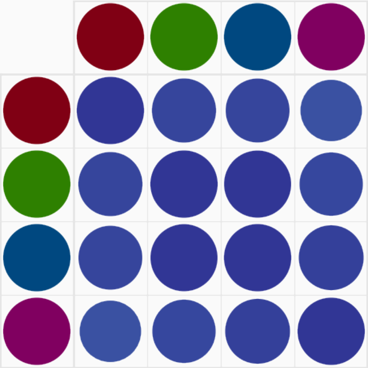
(a) Kernel matrix k_{q_t}.



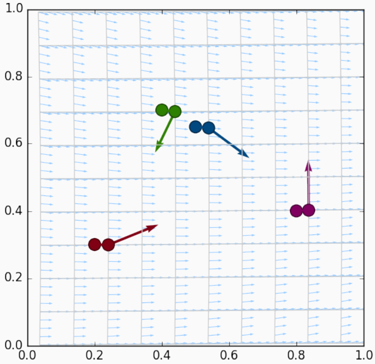
(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1..$

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



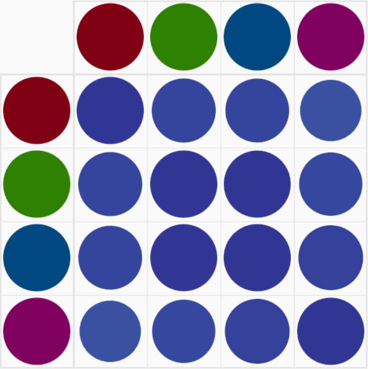
(a) Kernel matrix k_{q_t}.



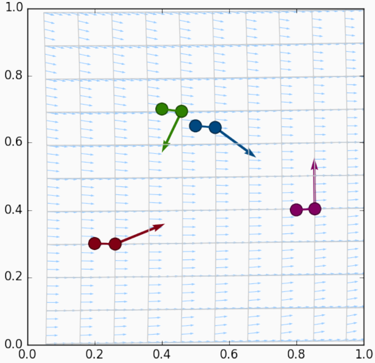
(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1..$

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



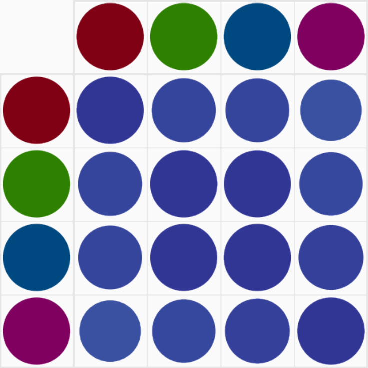
(a) Kernel matrix k_{q_t}.



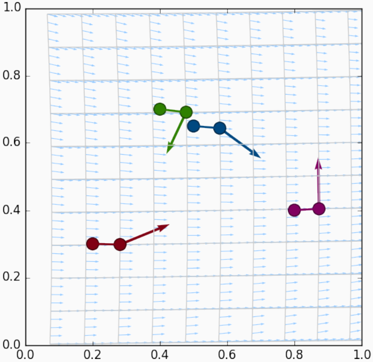
(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

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

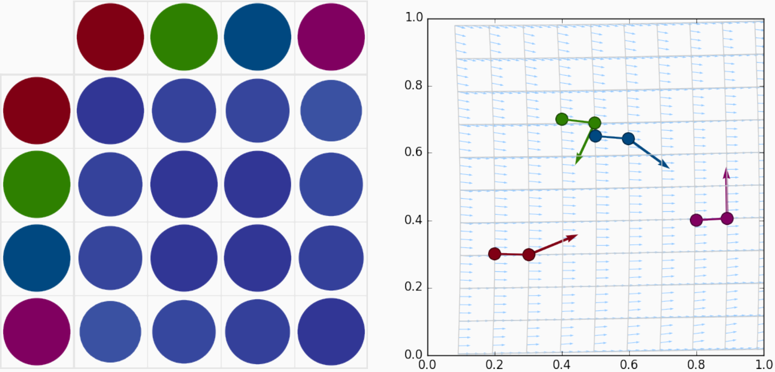


(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

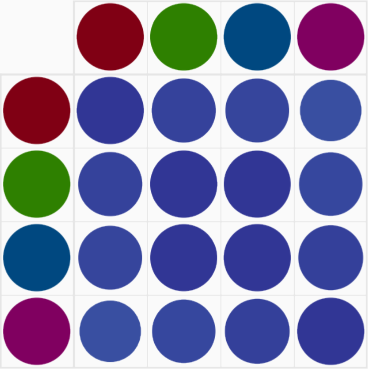
Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.



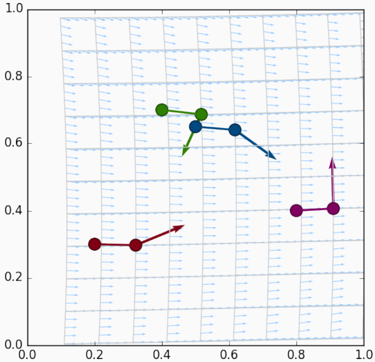
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1.$.

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



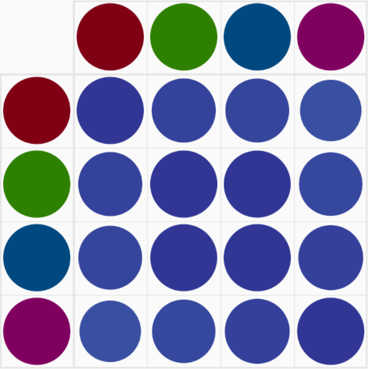
(a) Kernel matrix k_{q_t}.



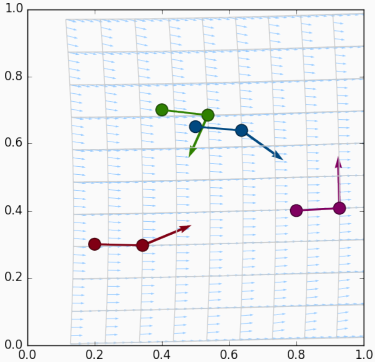
(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

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



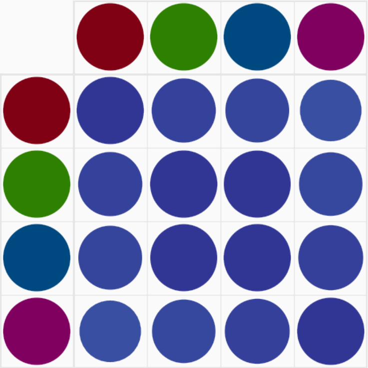
(a) Kernel matrix k_{q_t}.



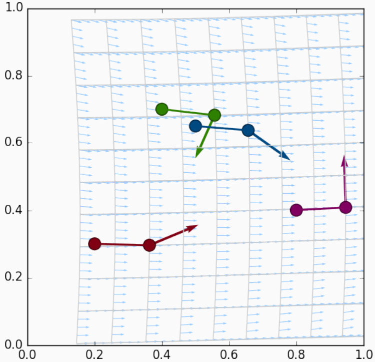
(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2 / 2\sigma^2)$, $\sigma = 1.$.

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



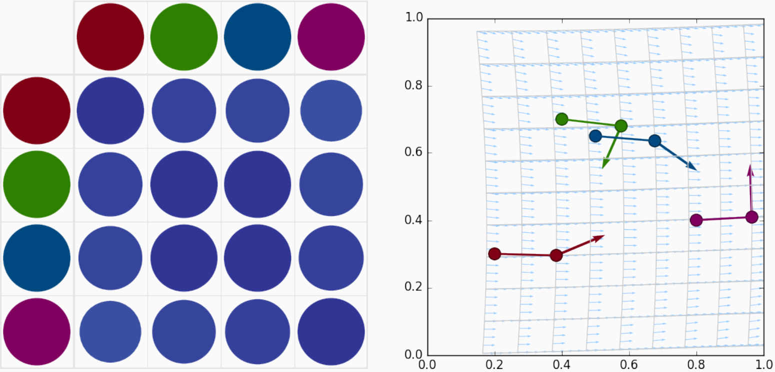
(a) Kernel matrix k_{q_t}.



(b) Shooted cloud (q_t, p_t).

Figure 23: Geodesic shooting, $k(x-y) = \exp(-\|x-y\|^2/2\sigma^2)$, $\sigma = 1..$

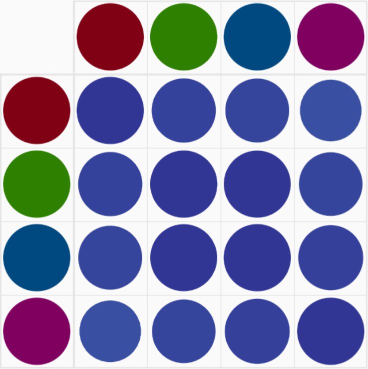
Influence of the kernel width, $\sigma = 1.$



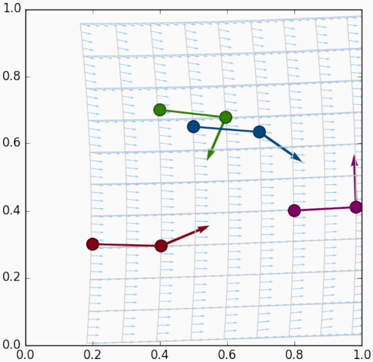
(a) Kernel matrix k_{q_t}. (b) Shooted cloud (q_t, p_t).

Figure 23: 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 23: 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 $((q_t, p_t) \longleftrightarrow \phi_t)$

The LDDMM framework

An iterative matching algorithm

Variability decomposition

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

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

Variability decomposition

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

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

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

$$\|f(X) - Y\|_s^2 = \|\mu - \nu\|_s^2 = \|B_s \star (\mu - \nu)\|_{L^2(\mathbb{R}^D)}^2 . \tag{51}$$

Variability decomposition

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

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

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

$$\|f(X) - Y\|_s^2 \;=\; \|\mu - \nu\|_s^2 \;=\; \|B_s \star (\mu - \nu)\|_{L^2(\mathbb{R}^D)}^2 \,. \tag{51}$$

Ideally, we are looking for

$$p_s^\perp(Y \to G_k \cdot X) \;=\; \arg\min_{f \in G_k} \|f(X) - Y\|_s^2. \tag{52}$$

Regularized matching problem

However, in practice :

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

Regularized matching problem

However, in practice :

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

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

$$\mathrm{Cost}(f) = \gamma_{\mathrm{reg}} \cdot d_k^2(X, f(X)) + \gamma_{\mathrm{att}} \cdot \|f(X) - Y\|_s^2. \tag{53}$$

Regularized matching problem

However, in practice :

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

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

$$\mathrm{Cost}(f) = \gamma_{\mathrm{reg}} \cdot d_k^2(X, f(X)) + \gamma_{\mathrm{att}} \cdot \|f(X) - Y\|_S^2. \tag{53}$$

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

$$\mathrm{Cost}(\mathbf{p_0}) = \gamma_{\mathrm{reg}} \cdot p_0^\top K_{q_0} p_0 + \gamma_{\mathrm{att}} \cdot \|\mathbf{q_1} - Y\|_S^2. \tag{54}$$

Regularized matching problem

However, in practice :

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

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

$$\mathrm{Cost}(f) = \gamma_{\mathrm{reg}} \cdot d_k^2(X, f(X)) + \gamma_{\mathrm{att}} \cdot \|f(X) - Y\|_S^2. \tag{53}$$

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

$$\mathrm{Cost}(\mathbf{p_0}) = \gamma_{\mathrm{reg}} \cdot p_0^\intercal K_{q_0} p_0 + \gamma_{\mathrm{att}} \cdot \|\mathbf{q_1} - Y\|_S^2. \tag{54}$$

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

Gradient descent on finite-dimensional manifolds

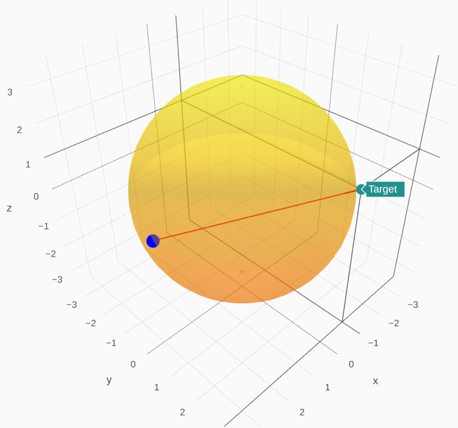


Figure 24: Matching from the source X to the target Y, constrained to the golden sphere $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, f(X))$ is much less constrained than the dissimilarity $\|f(X) - Y\|_S^2$.

Gradient descent on finite-dimensional manifolds

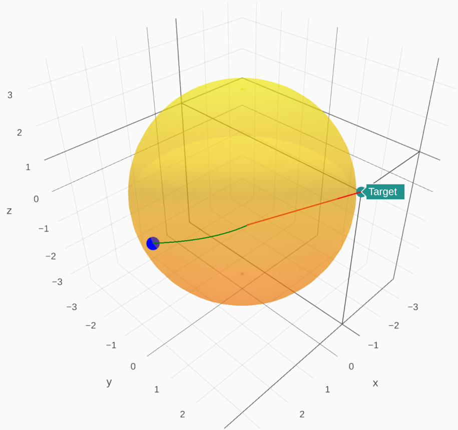


Figure 24: Matching from the source X to the target Y, constrained to the golden sphere $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, f(X))$ is much less constrained than the dissimilarity $\|f(X) - Y\|_S^2$.

Gradient descent on finite-dimensional manifolds

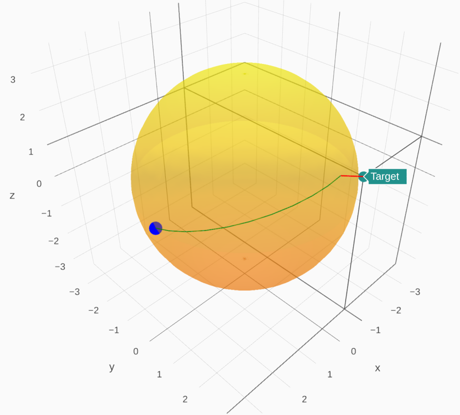


Figure 24: Matching from the source X to the target Y, constrained to the golden sphere $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, f(X))$ is much less constrained than the dissimilarity $\|f(X) - Y\|_S^2$.

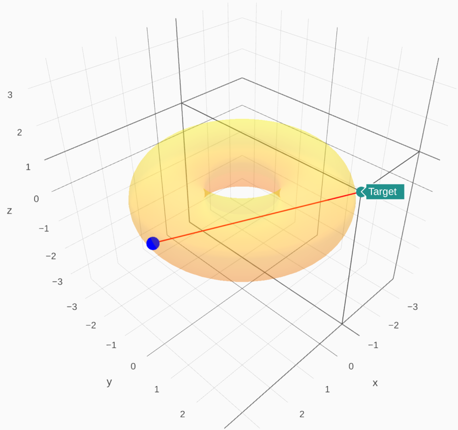


Figure 25: Matching from the source X to the target Y, constrained to the golden torus $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, f(X))$ is much less constrained than the dissimilarity $\|f(X) - Y\|_s^2$.

Gradient descent on finite-dimensional manifolds

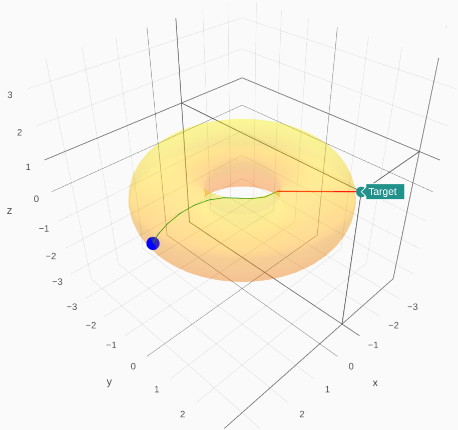


Figure 25: Matching from the source X to the target Y, constrained to the golden torus $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, f(X))$ is much less constrained than the dissimilarity $\|f(X) - Y\|_S^2$.

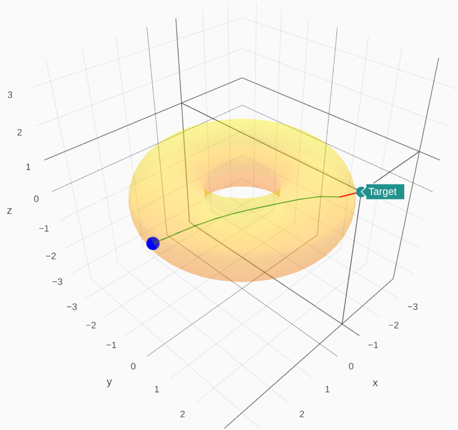


Figure 25: Matching from the source X to the target Y, constrained to the golden torus $G_k \cdot X$.
Here, $\gamma_{\text{reg}} << \gamma_{\text{att}}$: the geodesic length $d_k^2(X, f(X))$ is much less constrained than the dissimilarity $\|f(X) - Y\|_S^2$.

The LDDMM 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)                # $H(q,p) = \frac{1}{2} \cdot \sum_{i,j} k(x_i,x_j) p_i . p_j$
```

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

```python
# 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

```
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

```python
# 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) = .01 * 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
```

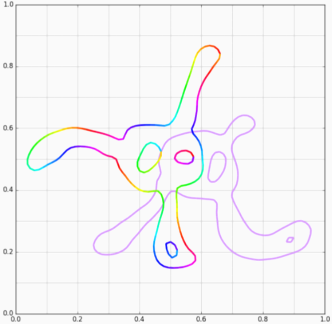
Minimization script, part 3

```
    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)
```

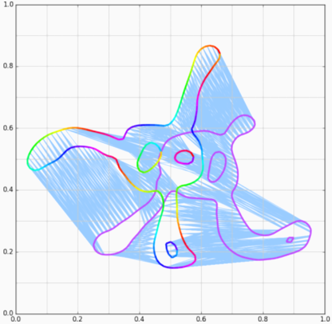
The LDDMM framework

Results

Typical run with OT fidelity

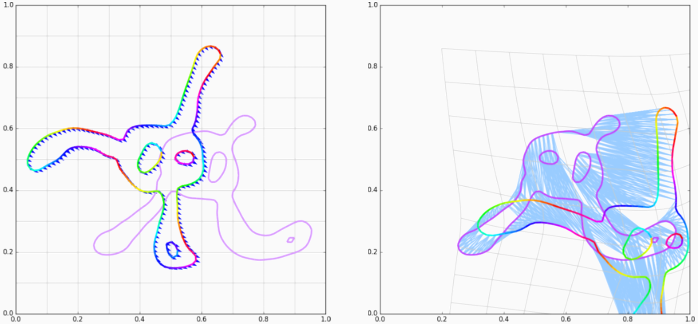


(a) Momentum p_0.



(b) Shooted model q_1.

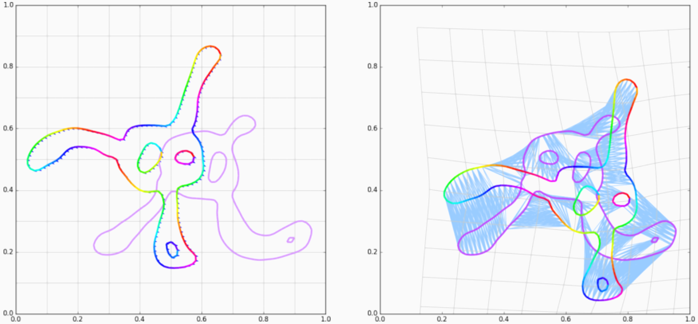
Figure 26: Iteration 0.



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

Figure 26: Iteration 3.

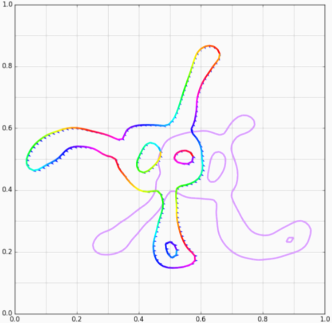
Typical run with OT fidelity



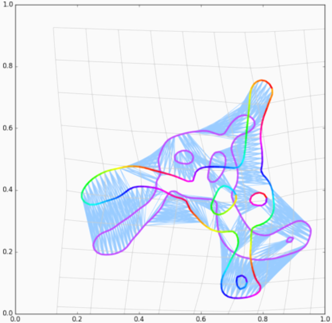
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 26: Iteration 4.

Typical run with OT fidelity



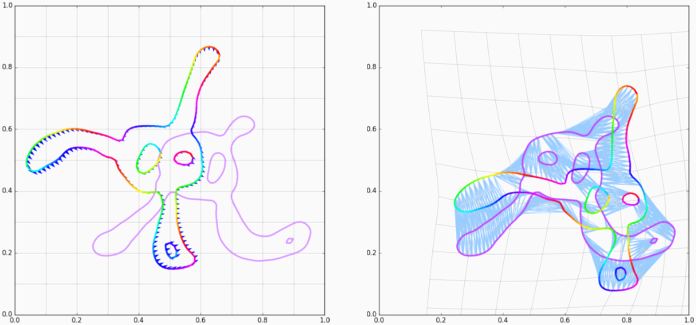
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 5.

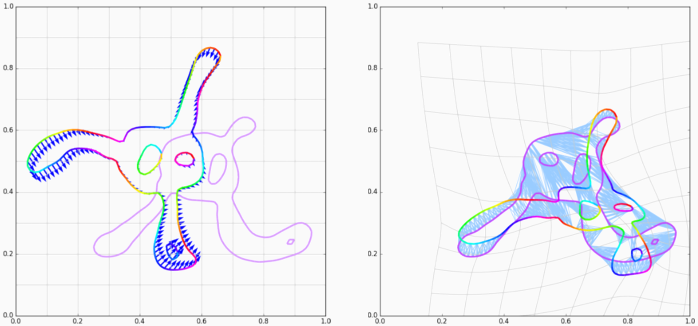
Typical run with OT fidelity



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

Figure 26: Iteration 6.

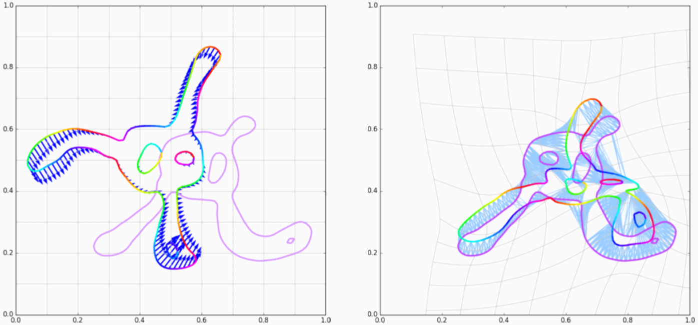
Typical run with OT fidelity



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

Figure 26: Iteration 7.

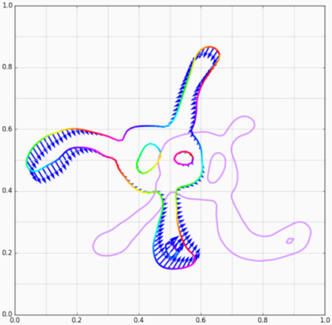
Typical run with OT fidelity



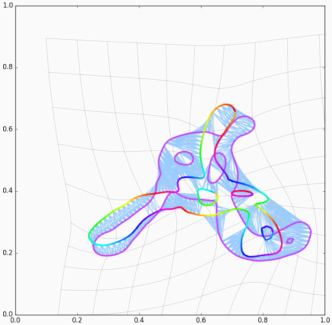
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 26: Iteration 8.

Typical run with OT fidelity



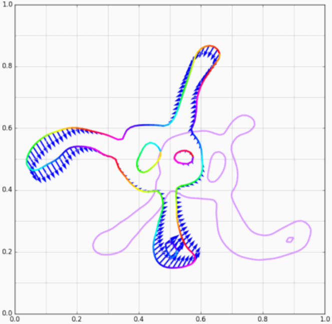
(a) Momentum p_0.



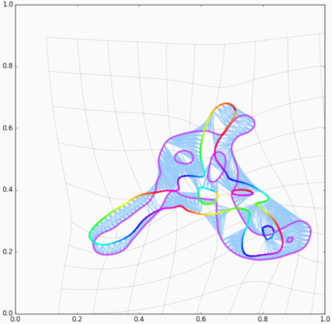
(b) Shooted model q_1.

Figure 26: Iteration 9.

Typical run with OT fidelity



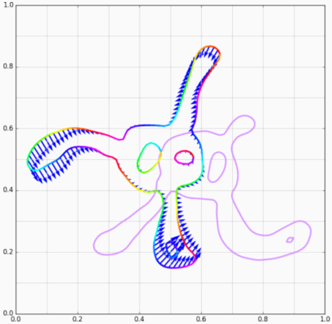
(a) Momentum p_0.



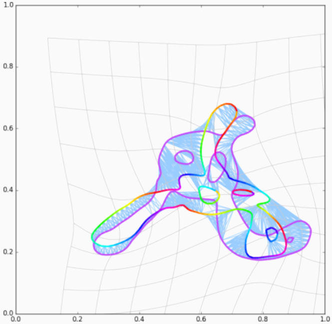
(b) Shooted model q_1.

Figure 26: Iteration 10.

Typical run with OT fidelity

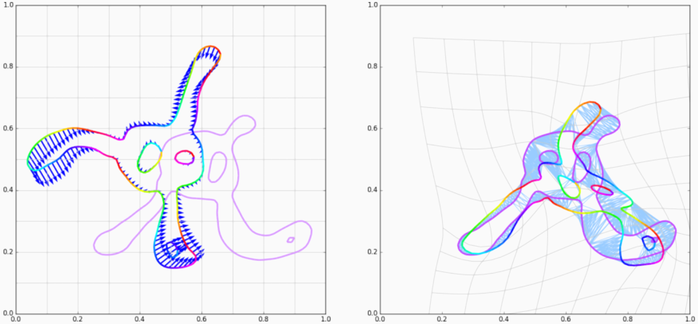


(a) Momentum p_0.



(b) Shooted model q_1.

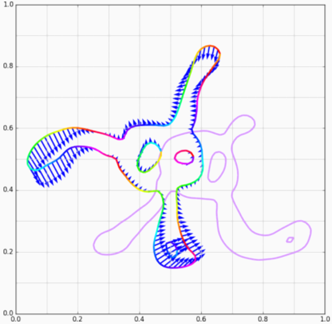
Figure 26: Iteration 11.



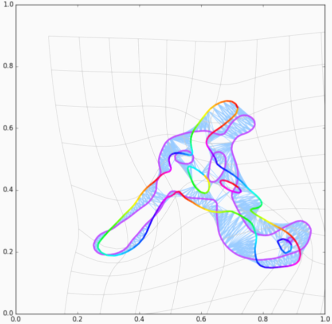
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 12.

Typical run with OT fidelity



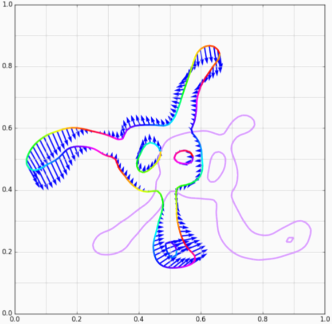
(a) Momentum p_0.



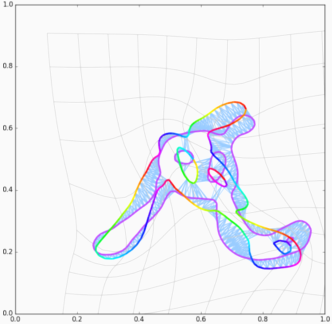
(b) Shooted model q_1.

Figure 26: Iteration 13.

Typical run with OT fidelity



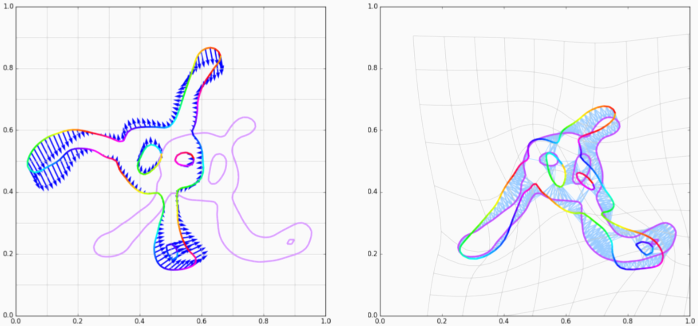
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 14.

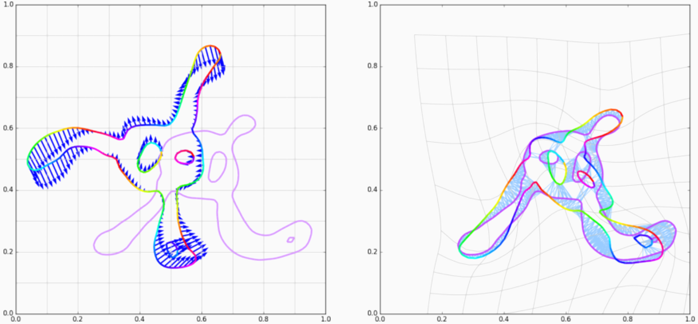
Typical run with OT fidelity



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

Figure 26: Iteration 15.

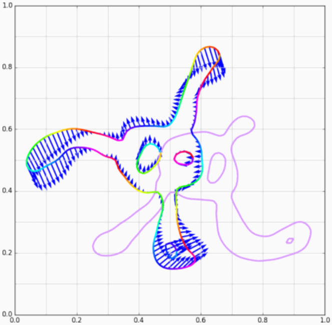
Typical run with OT fidelity



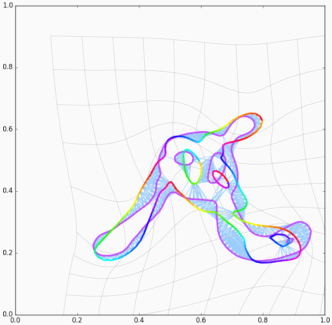
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 16.

Typical run with OT fidelity



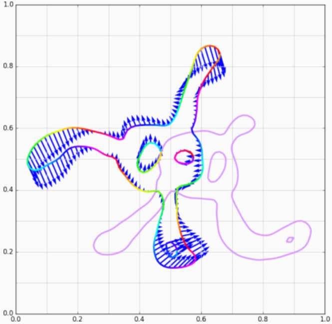
(a) Momentum p_0.



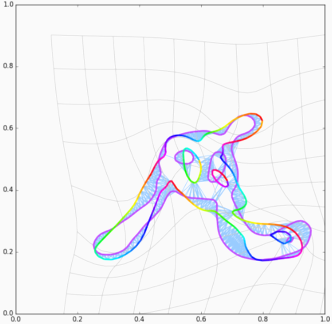
(b) Shooted model q_1.

Figure 26: Iteration 17.

Typical run with OT fidelity



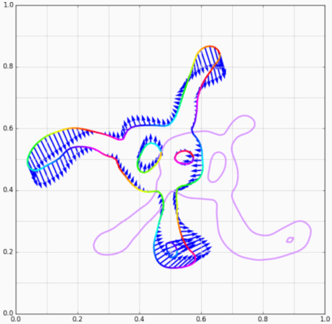
(a) Momentum p_0.



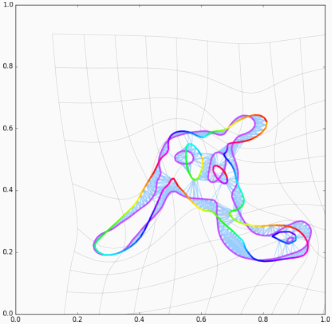
(b) Shooted model q_1.

Figure 26: Iteration 18.

Typical run with OT fidelity



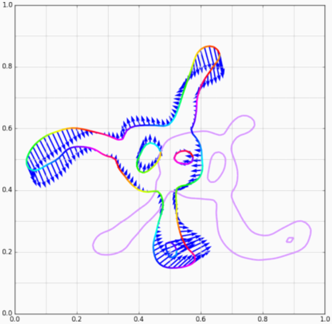
(a) Momentum p_0.



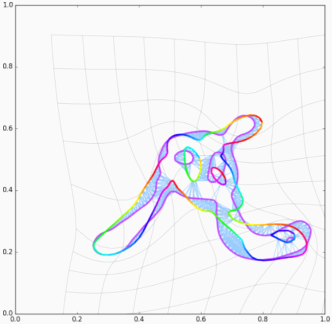
(b) Shooted model q_1.

Figure 26: Iteration 19.

Typical run with OT fidelity



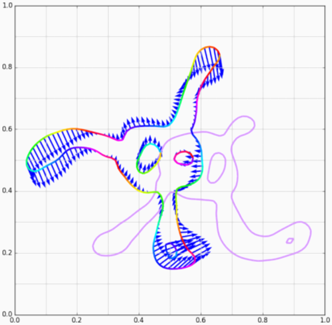
(a) Momentum p_0.



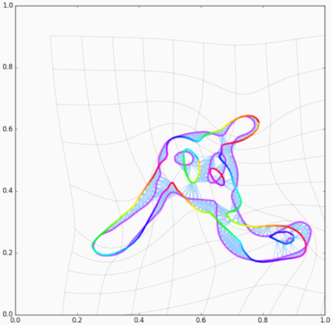
(b) Shooted model q_1.

Figure 26: Iteration 20.

Typical run with OT fidelity



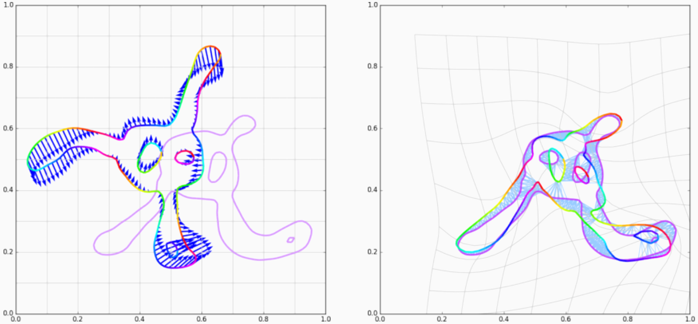
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 21.

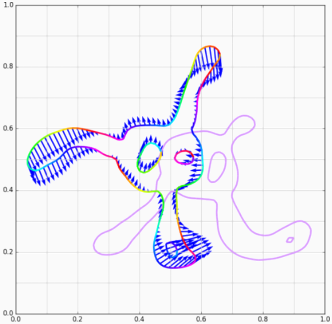
Typical run with OT fidelity



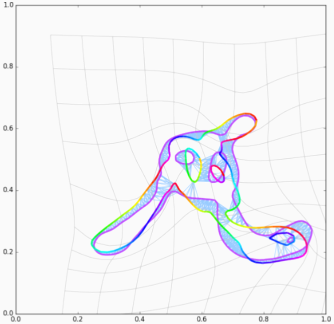
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 26: Iteration 22.

Typical run with OT fidelity



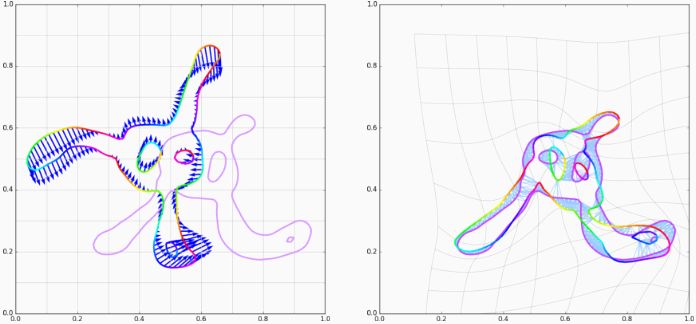
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 23.

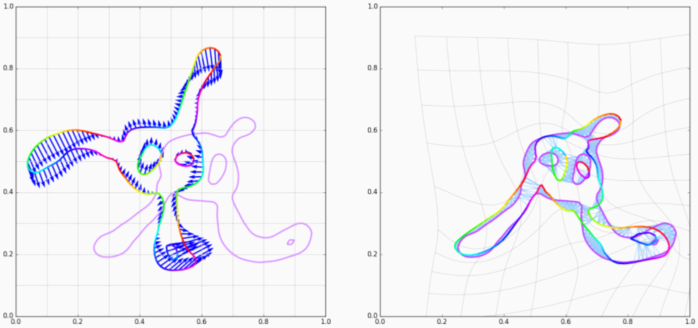
Typical run with OT fidelity



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

Figure 26: Iteration 24.

Typical run with OT fidelity

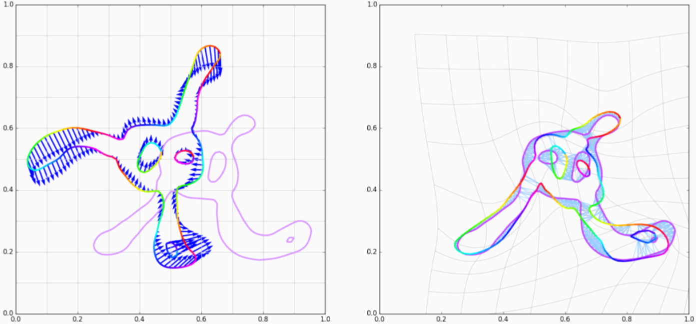


(a) Momentum p_0.

(b) Shooted model q_1.

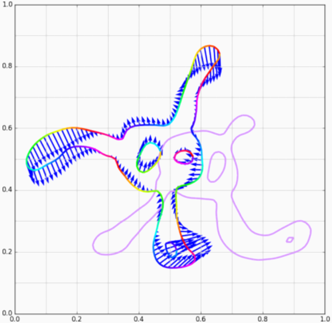
Figure 26: Iteration 25.

Typical run with OT fidelity

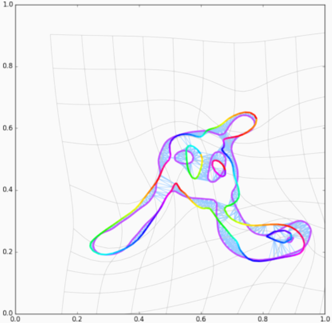


(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 26.



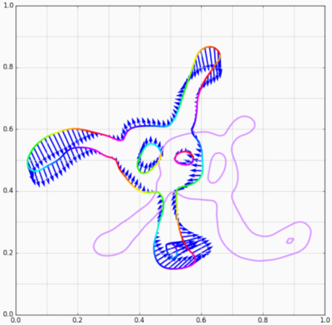
(a) Momentum p_0.



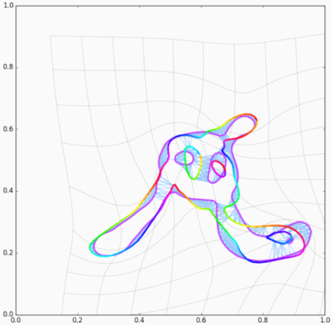
(b) Shooted model q_1.

Figure 26: Iteration 27.

Typical run with OT fidelity



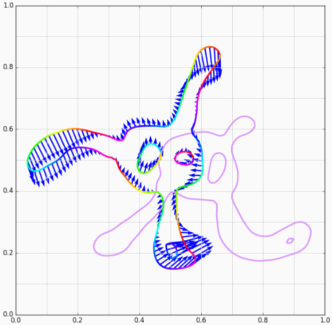
(a) Momentum p_0.



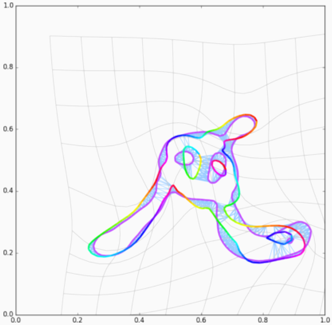
(b) Shooted model q_1.

Figure 26: Iteration 28.

Typical run with OT fidelity



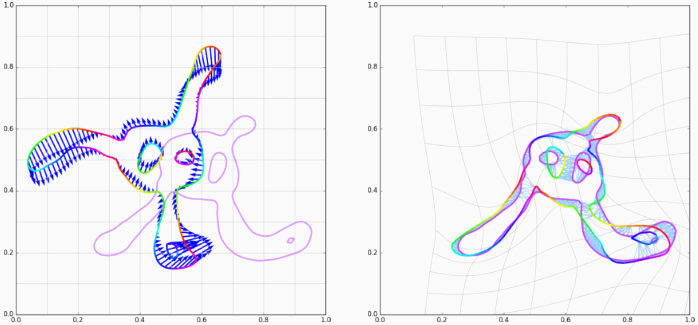
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 29.

Typical run with OT fidelity

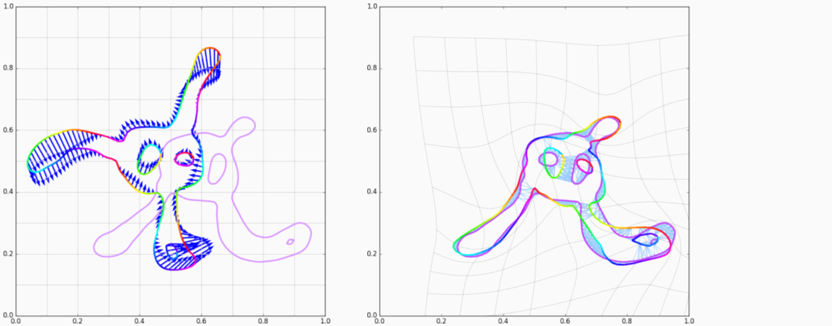


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 26: Iteration 30.

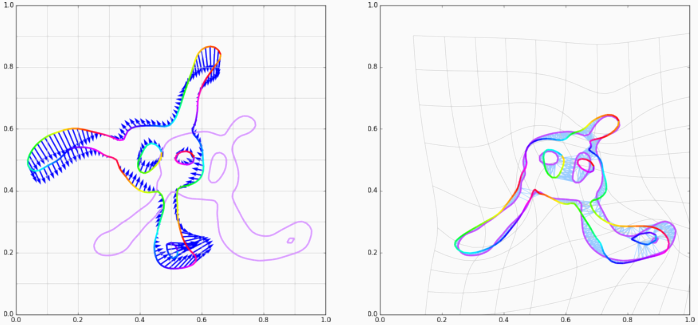
Typical run with OT fidelity



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

Figure 26: Iteration 31.

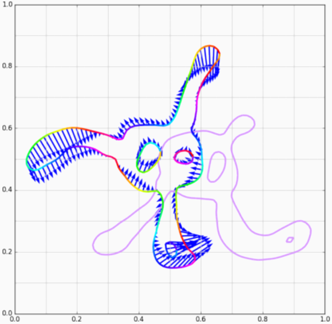
Typical run with OT fidelity



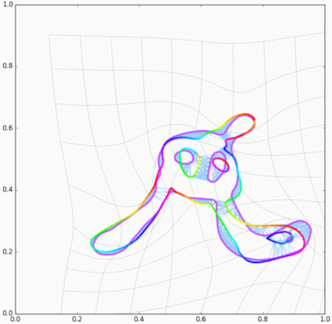
(a) Momentum p_0.　(b) Shooted model q_1.

Figure 26: Iteration 32.

Typical run with OT fidelity



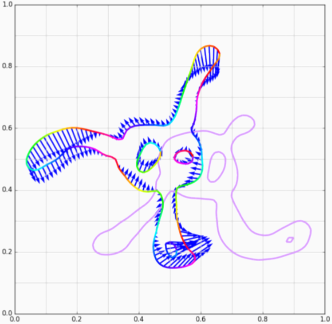
(a) Momentum p_0.



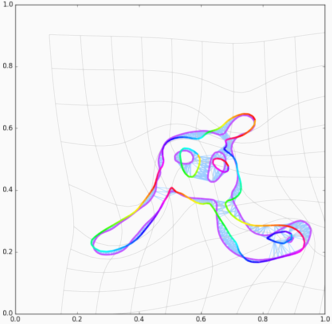
(b) Shooted model q_1.

Figure 26: Iteration 33.

Typical run with OT fidelity



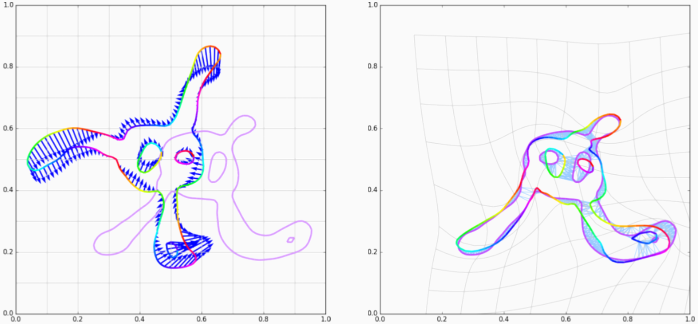
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 34.

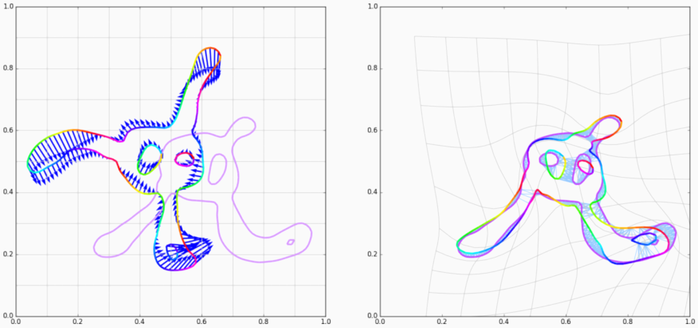
Typical run with OT fidelity



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

Figure 26: Iteration 35.

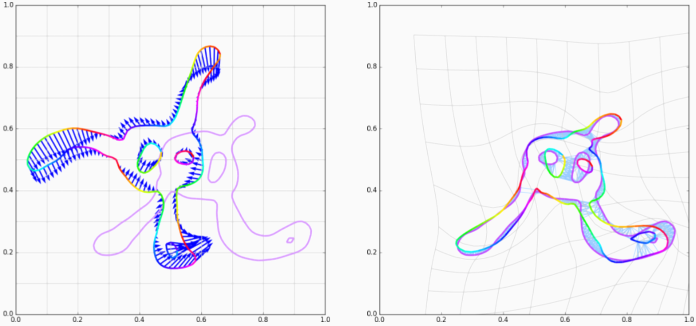
Typical run with OT fidelity



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

Figure 26: Iteration 36.

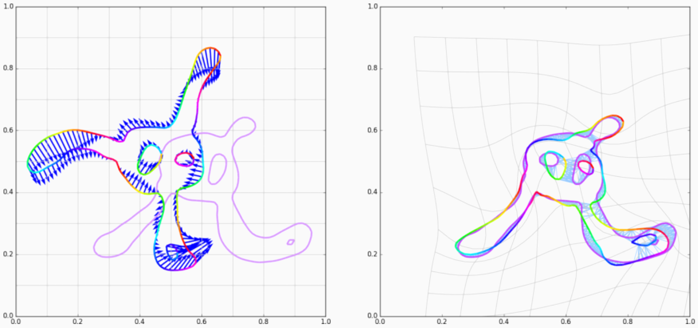
Typical run with OT fidelity



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

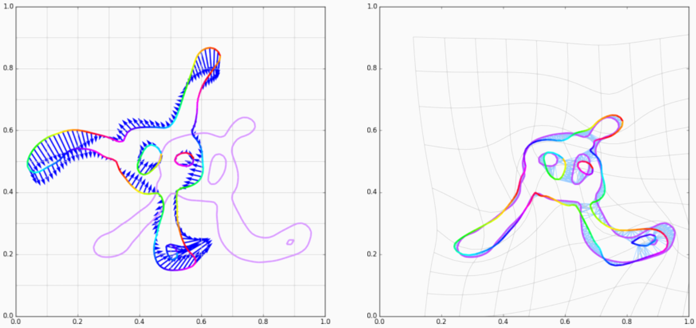
Figure 26: Iteration 37.

Typical run with OT fidelity



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

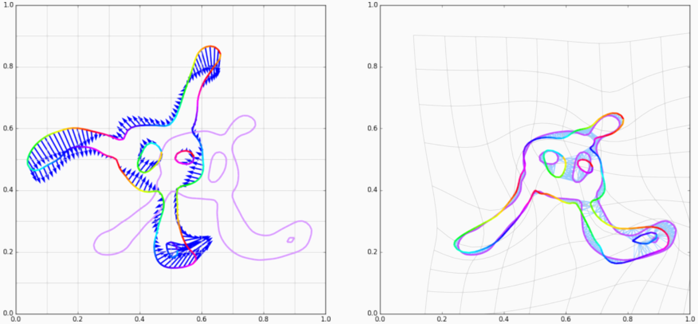
Figure 26: Iteration 38.



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

Figure 26: Iteration 39.

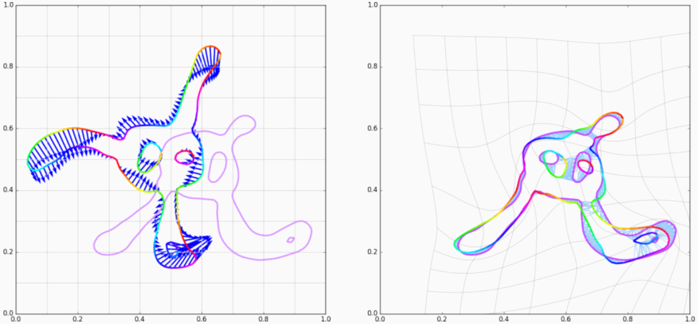
Typical run with OT fidelity



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

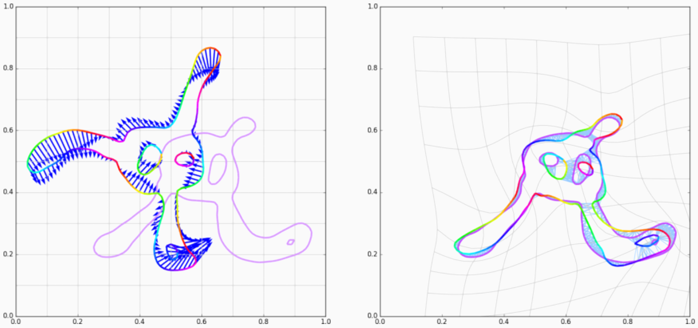
Figure 26: Iteration 41.

Typical run with OT fidelity



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

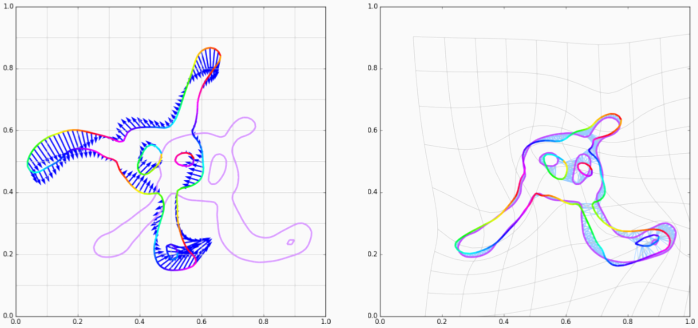
Figure 26: Iteration 42.



(a) Momentum p_0.

(b) Shooted model q_1.

Figure 26: Iteration 43.

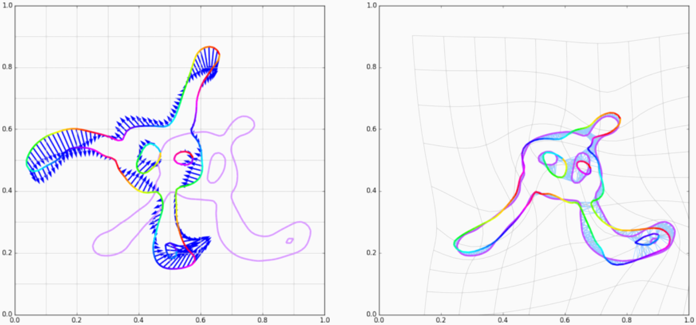


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 26: Iteration 44.

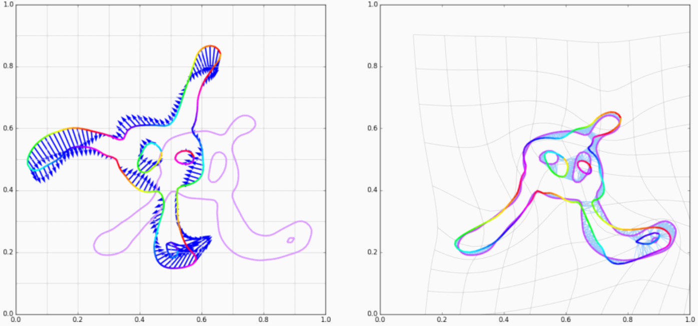
Typical run with OT fidelity



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

Figure 26: Iteration 46.

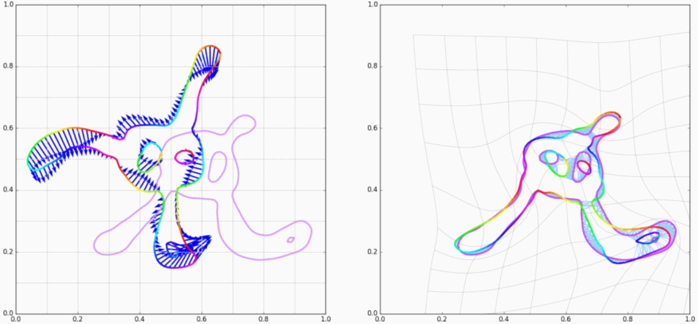
Typical run with OT fidelity



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

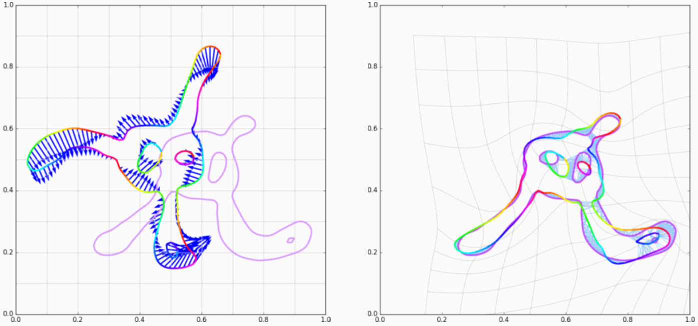
Figure 26: Iteration 47.

Typical run with OT fidelity



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

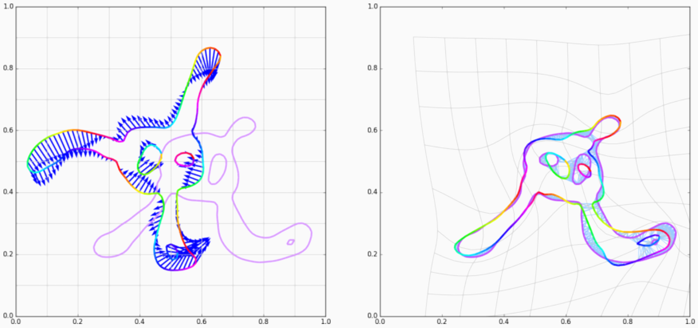
Figure 26: Iteration 48.



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

Figure 26: Iteration 49.

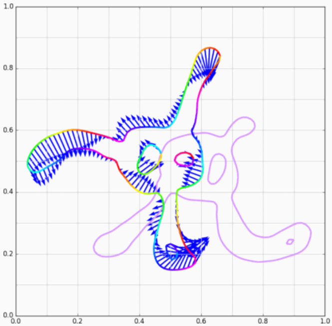
Typical run with OT fidelity



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

Figure 26: Iteration 50.

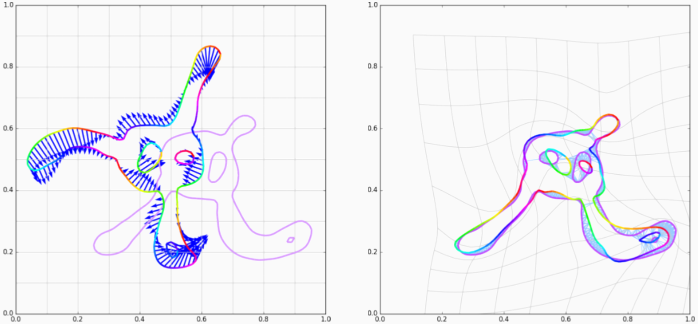
Typical run with OT fidelity



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

Figure 26: Iteration 52.

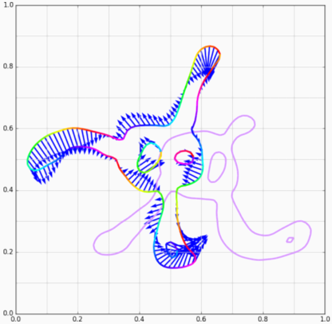
Typical run with OT fidelity



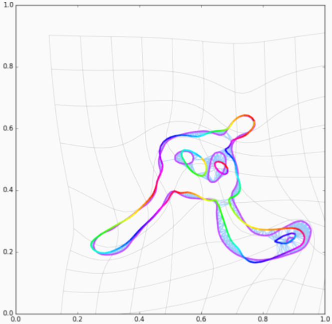
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 53.

Typical run with OT fidelity



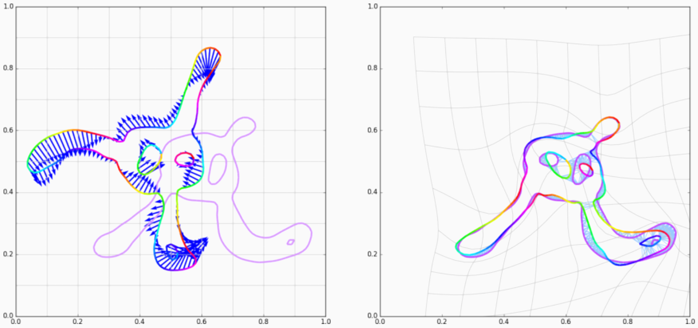
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 54.

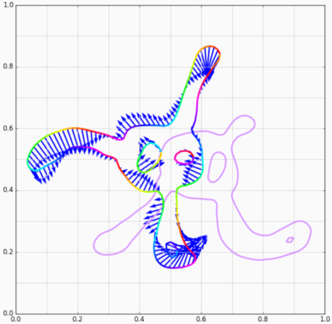
Typical run with OT fidelity



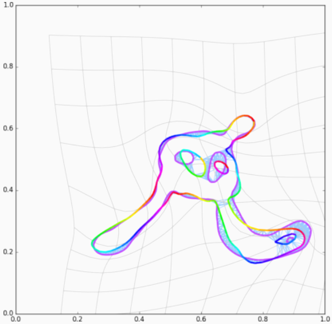
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 55.

Typical run with OT fidelity



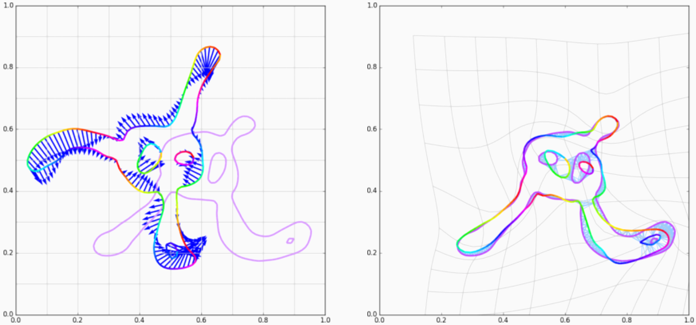
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 56.

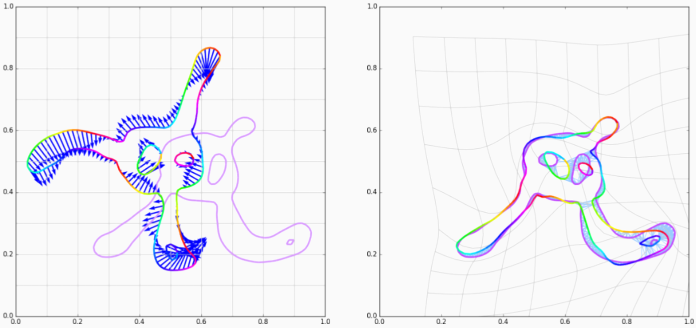
Typical run with OT fidelity



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

Figure 26: Iteration 57.

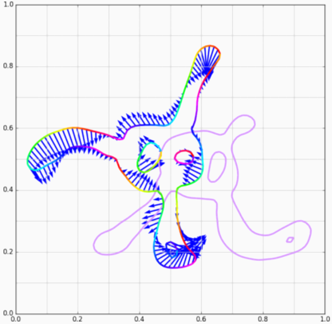
Typical run with OT fidelity



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

Figure 26: Iteration 58.

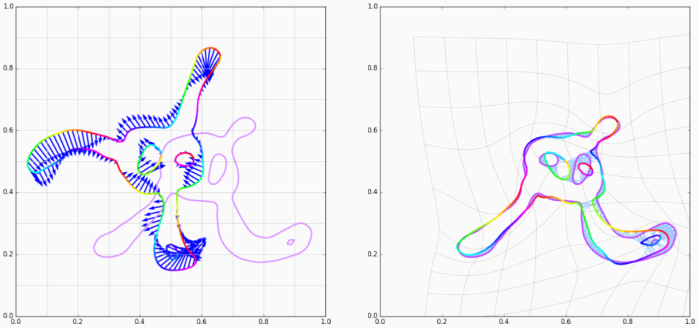
Typical run with OT fidelity



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

Figure 26: Iteration 59.

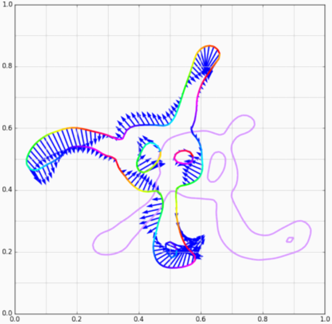
Typical run with OT fidelity



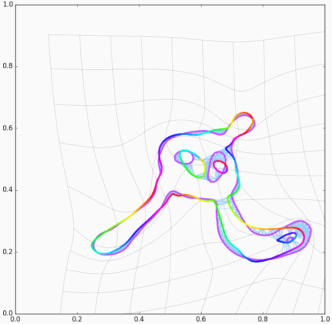
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 60.

Typical run with OT fidelity



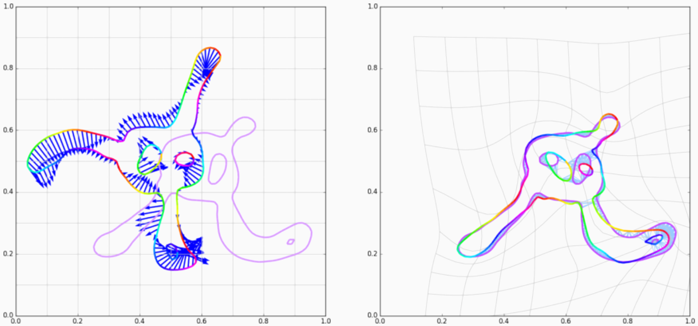
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 61.

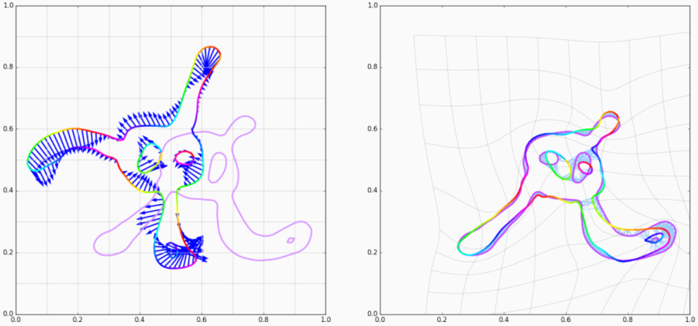
Typical run with OT fidelity



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

Figure 26: Iteration 62.

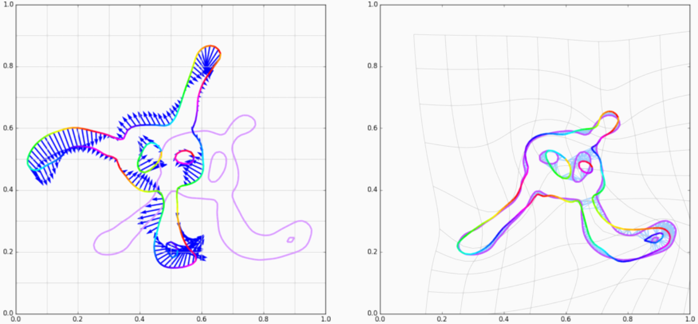
Typical run with OT fidelity



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

Figure 26: Iteration 64.

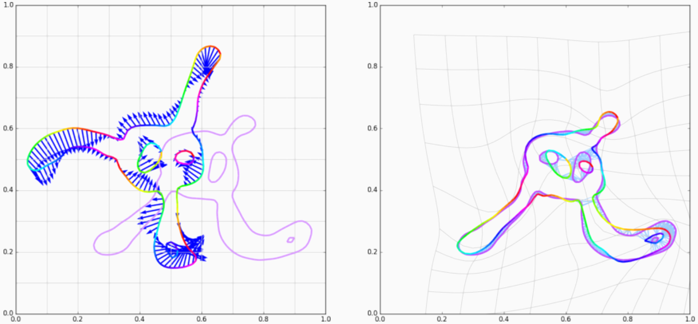
Typical run with OT fidelity



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

Figure 26: Iteration 65.

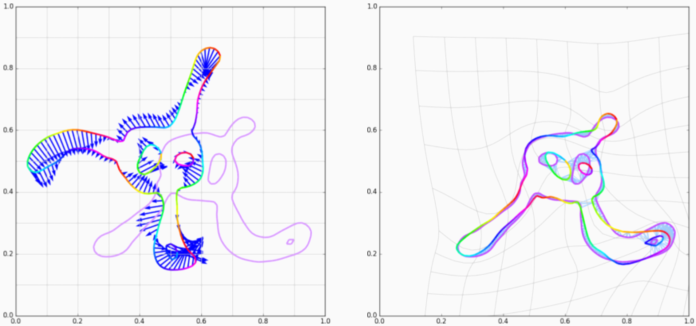
Typical run with OT fidelity



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

Figure 26: Iteration 66.

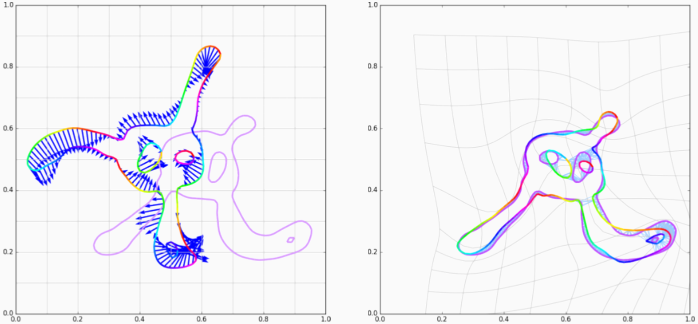
Typical run with OT fidelity



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

Figure 26: Iteration 67.

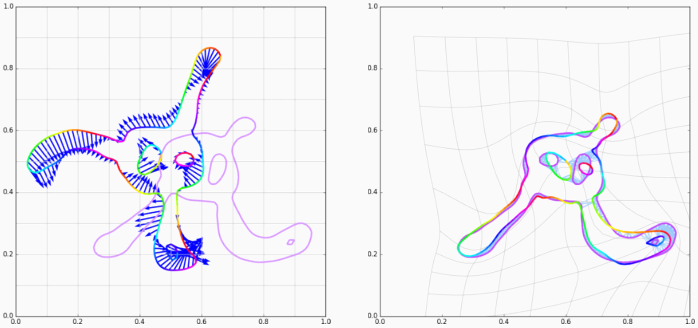
Typical run with OT fidelity



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

Figure 26: Iteration 68.

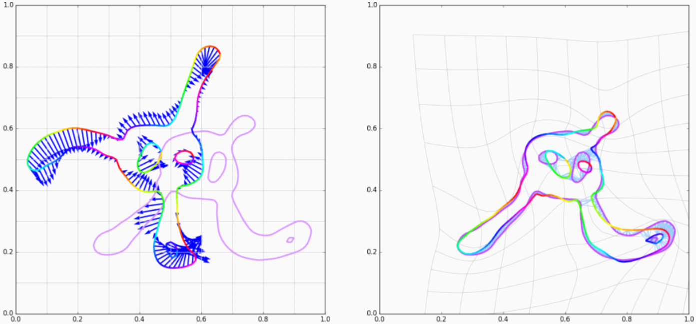
Typical run with OT fidelity



(a) Momentum p_0.

(b) Shooted model q_1.

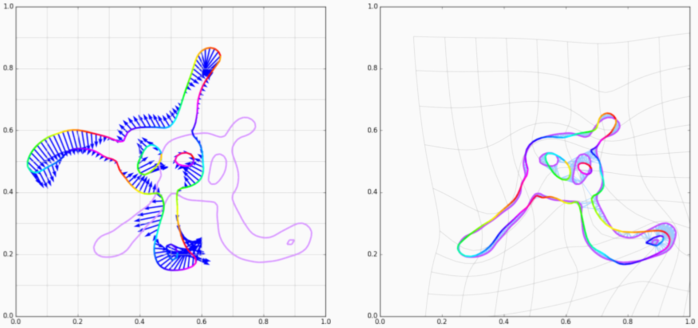
Figure 26: Iteration 69.



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

Figure 26: Iteration 70.

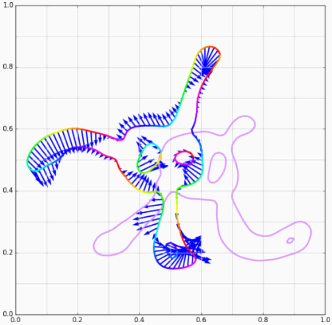
Typical run with OT fidelity



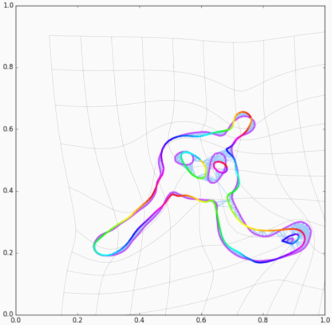
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 71.

Typical run with OT fidelity



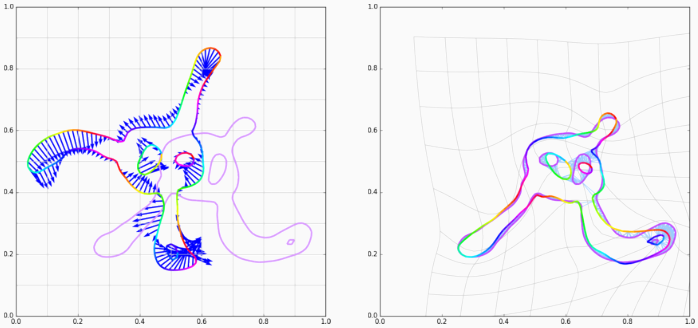
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 72.

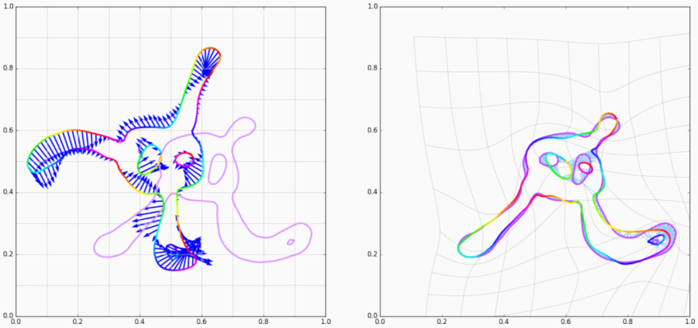
Typical run with OT fidelity



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

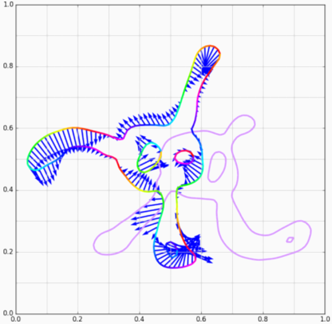
Figure 26: Iteration 73.

Typical run with OT fidelity

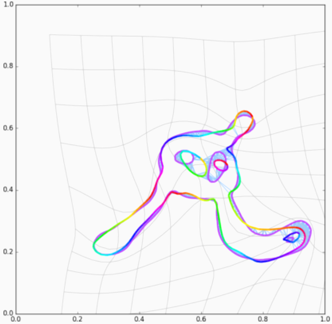


(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 74.



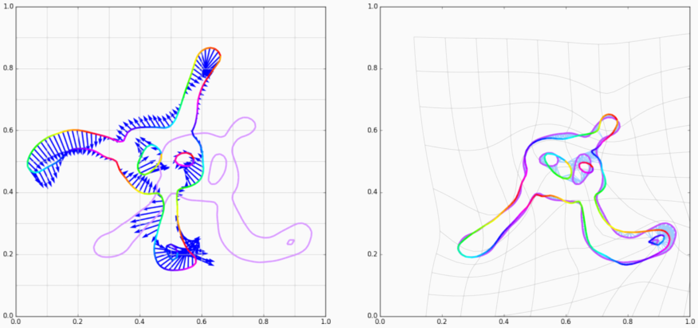
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 75.

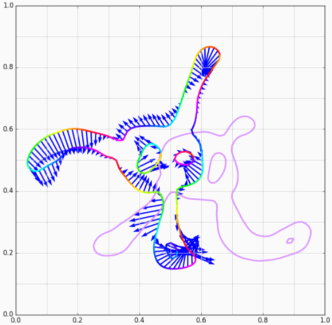
Typical run with OT fidelity



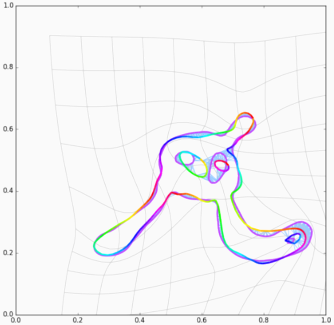
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 77.

Typical run with OT fidelity



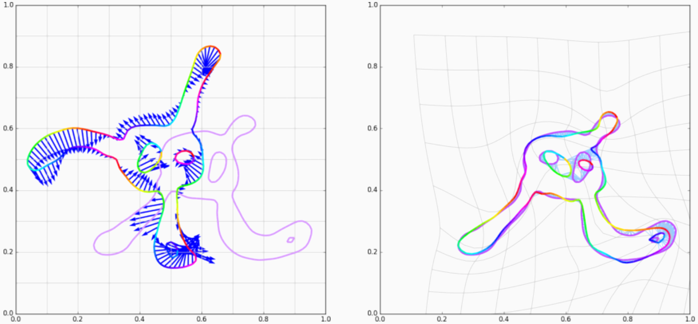
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 78.

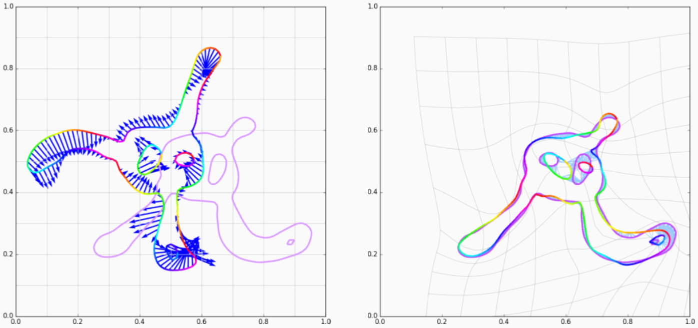
Typical run with OT fidelity



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

Figure 26: Iteration 79.

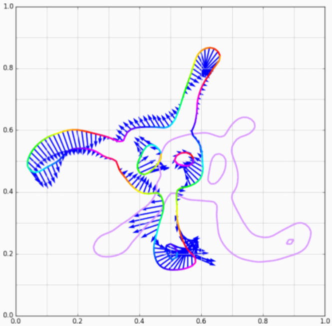
Typical run with OT fidelity



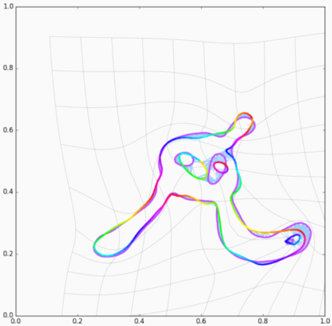
(a) Momentum p_0. (b) Shooted model q_1.

Figure 26: Iteration 80.

Typical run with OT fidelity



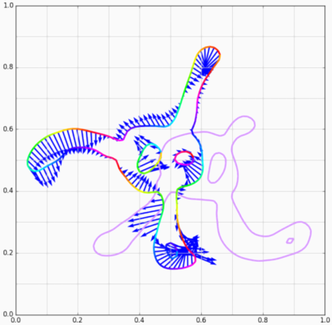
(a) Momentum p_0.



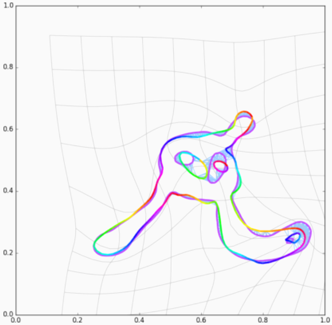
(b) Shooted model q_1.

Figure 26: Iteration 81.

Typical run with OT fidelity



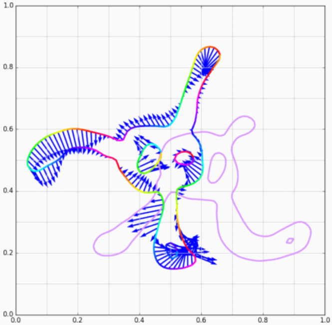
(a) Momentum p_0.



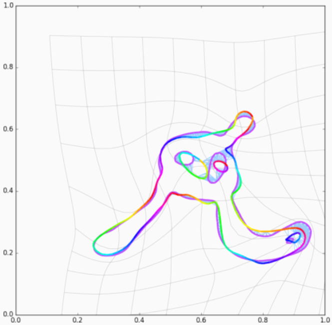
(b) Shooted model q_1.

Figure 26: Iteration 82.

Typical run with OT fidelity

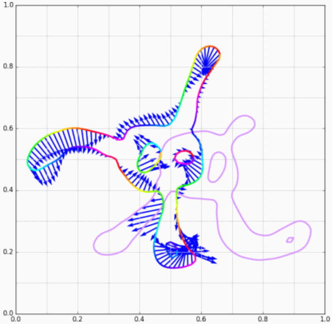


(a) Momentum p_0.

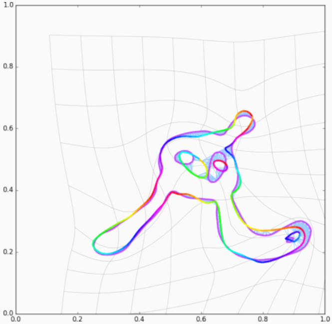


(b) Shooted model q_1.

Figure 26: Iteration 83.



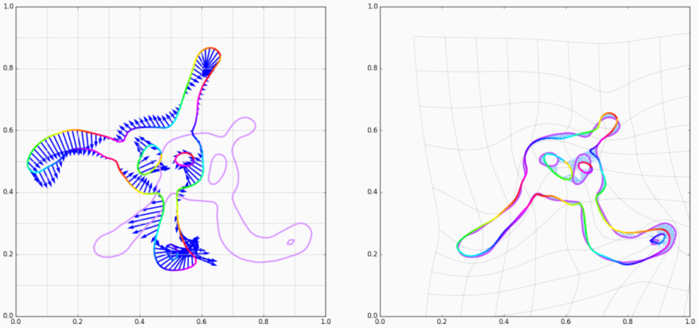
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 26: Iteration 85.

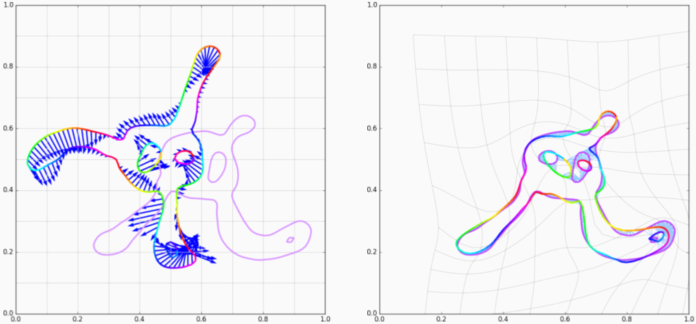
Typical run with OT fidelity



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

Figure 26: Iteration 86.

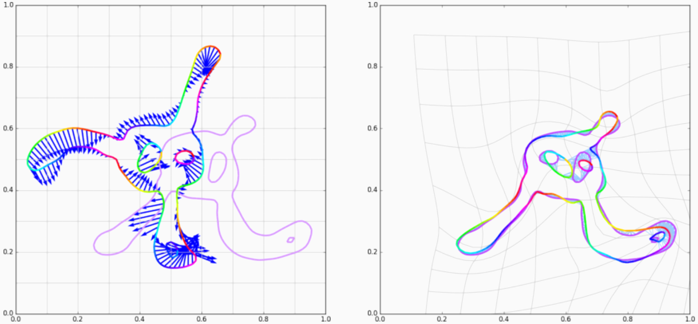
Typical run with OT fidelity



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

Figure 26: Iteration 87.

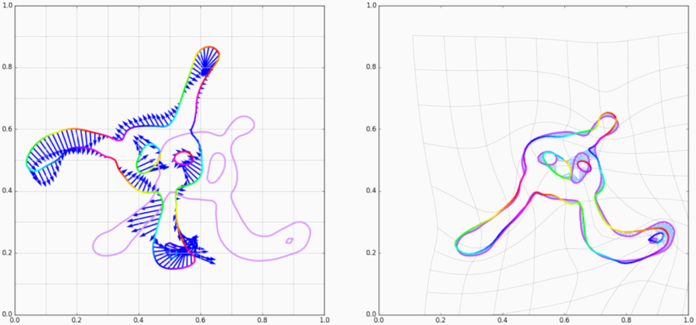
Typical run with OT fidelity



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

Figure 26: Iteration 88.

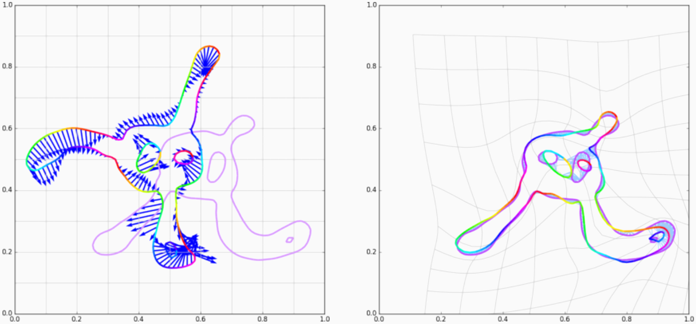
Typical run with OT fidelity



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

Figure 26: Iteration 89.

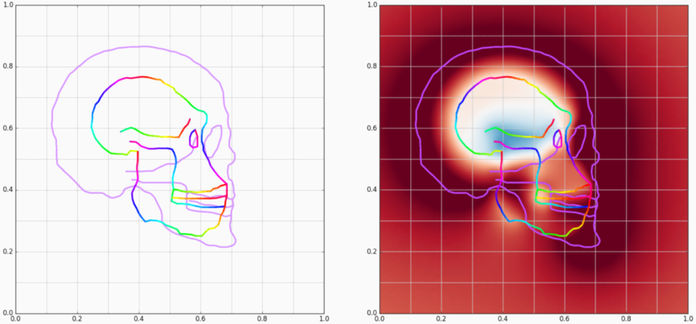
Typical run with OT fidelity



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

Figure 26: Iteration 90.

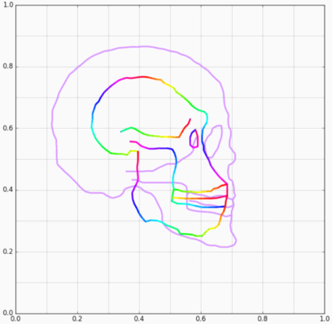
Typical run with kernel fidelity



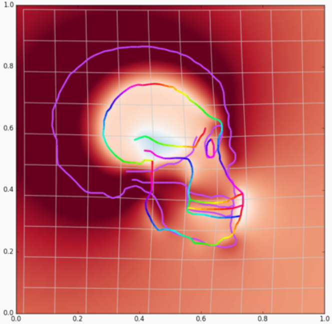
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 27: Iteration 0.

Typical run with kernel fidelity

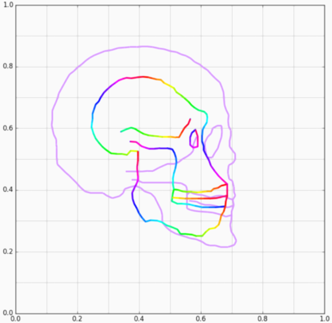


(a) Momentum p_0.

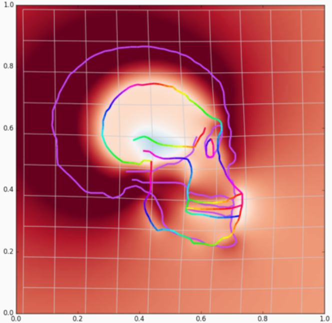


(b) Shooted model q_1.

Figure 27: Iteration 3.

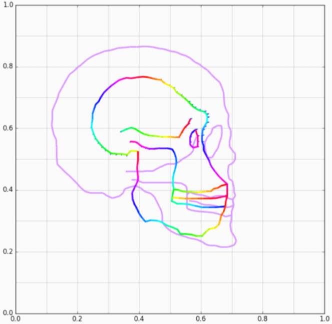


(a) Momentum p_0.

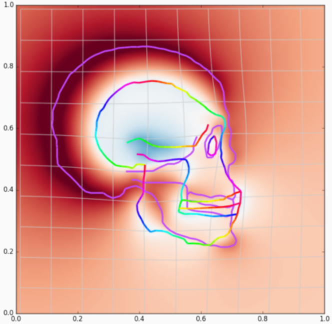


(b) Shooted model q_1.

Figure 27: Iteration 4.



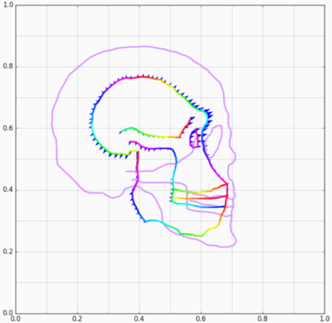
(a) Momentum p_0.



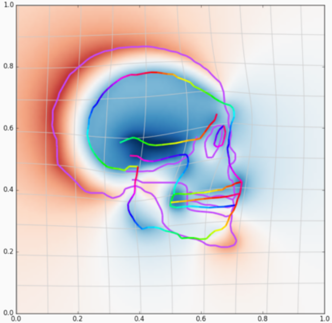
(b) Shooted model q_1.

Figure 27: Iteration 5.

Typical run with kernel fidelity



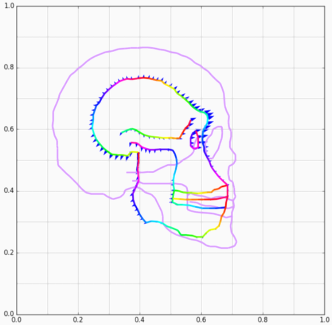
(a) Momentum p_0.



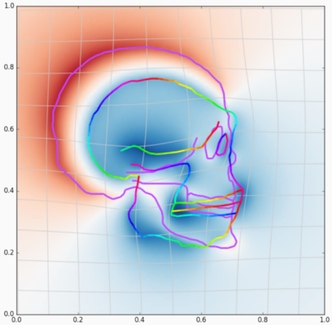
(b) Shooted model q_1.

Figure 27: Iteration 6.

Typical run with kernel fidelity



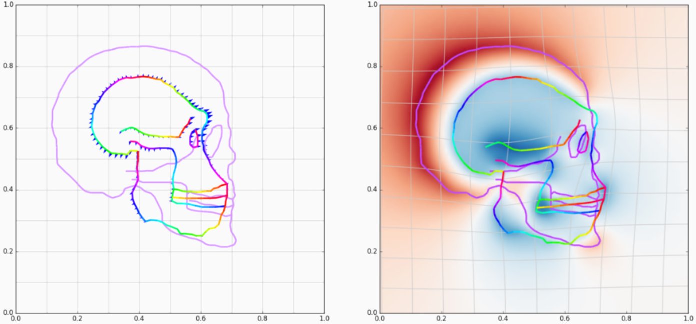
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 7.

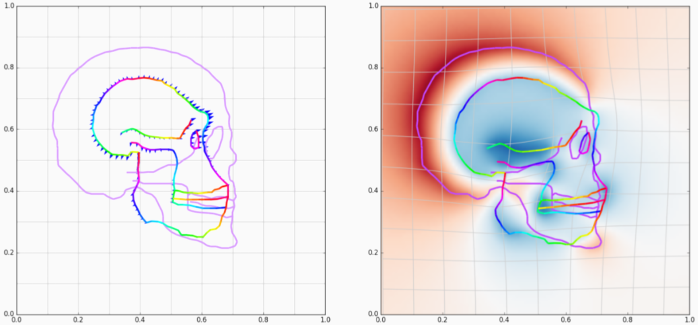
Typical run with kernel fidelity



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

Figure 27: Iteration 8.

Typical run with kernel fidelity

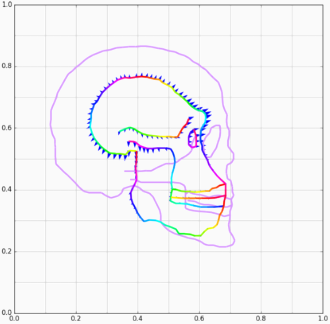


(a) Momentum p_0.

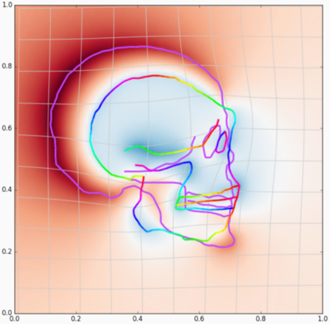
(b) Shooted model q_1.

Figure 27: Iteration 9.

Typical run with kernel fidelity



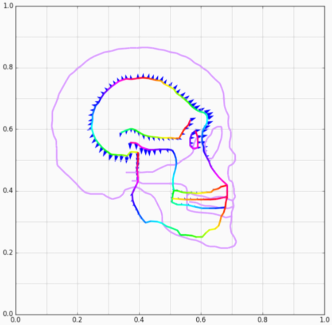
(a) Momentum p_0.



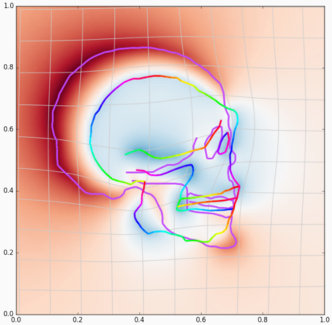
(b) Shooted model q_1.

Figure 27: Iteration 10.

Typical run with kernel fidelity



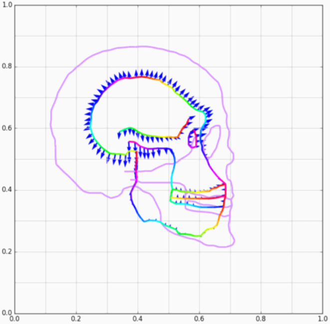
(a) Momentum p_0.



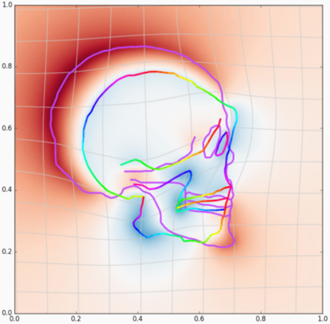
(b) Shooted model q_1.

Figure 27: Iteration 11.

Typical run with kernel fidelity



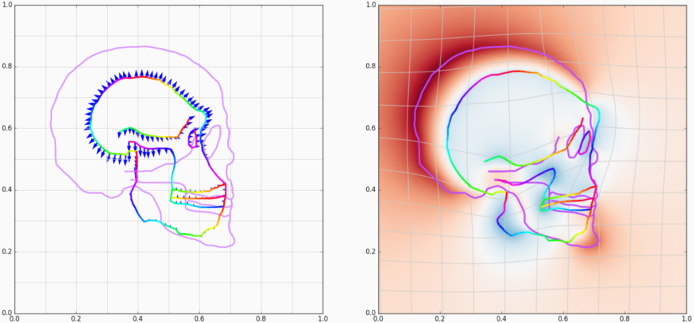
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 12.

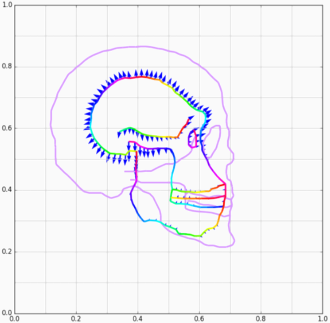
Typical run with kernel fidelity



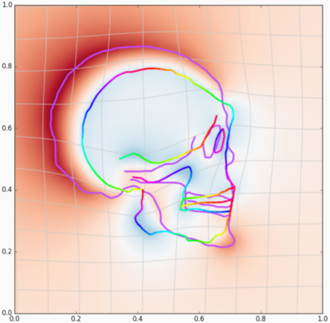
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 27: Iteration 13.

Typical run with kernel fidelity



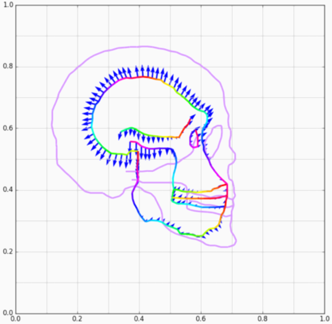
(a) Momentum p_0.



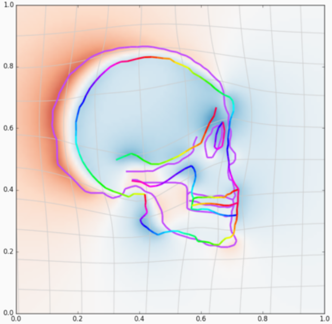
(b) Shooted model q_1.

Figure 27: Iteration 14.

Typical run with kernel fidelity



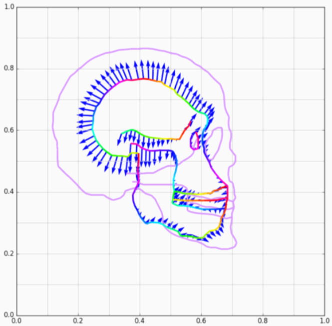
(a) Momentum p_0.



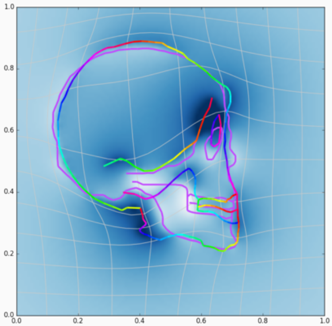
(b) Shooted model q_1.

Figure 27: Iteration 15.

Typical run with kernel fidelity



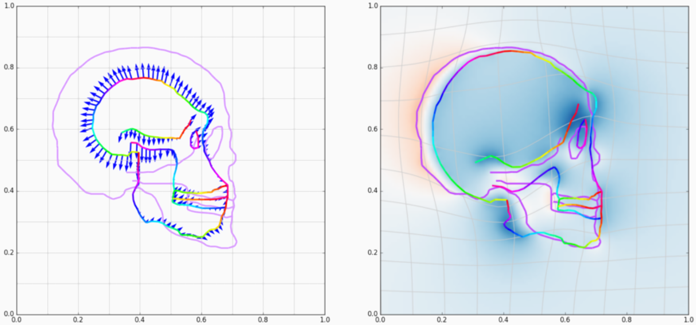
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 16.

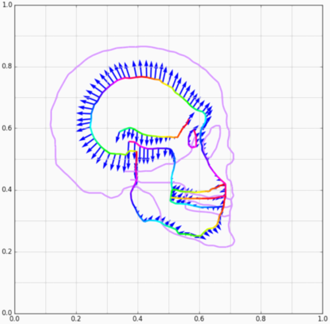
Typical run with kernel fidelity



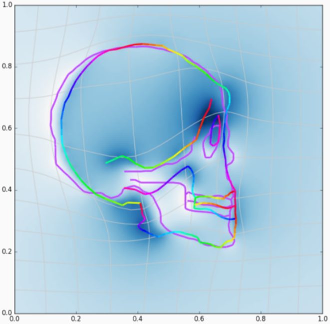
(a) Momentum p_0. (b) Shooted model q_1.

Figure 27: Iteration 17.

Typical run with kernel fidelity



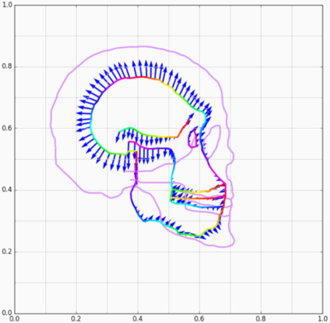
(a) Momentum p_0.



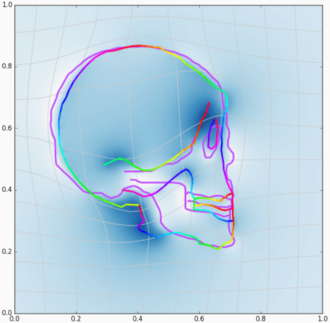
(b) Shooted model q_1.

Figure 27: Iteration 19.

Typical run with kernel fidelity



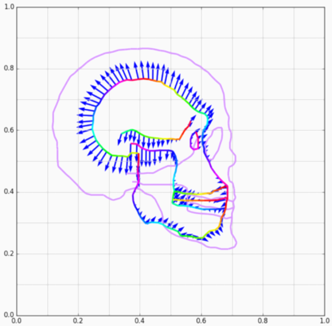
(a) Momentum p_0.



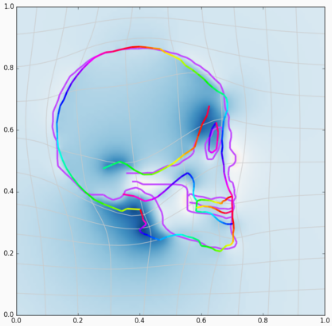
(b) Shooted model q_1.

Figure 27: Iteration 20.

Typical run with kernel fidelity



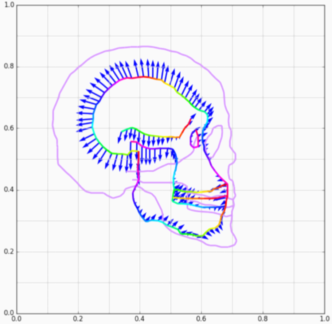
(a) Momentum p_0.



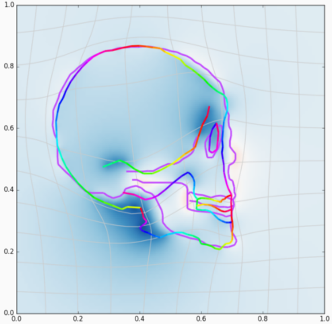
(b) Shooted model q_1.

Figure 27: Iteration 21.

Typical run with kernel fidelity



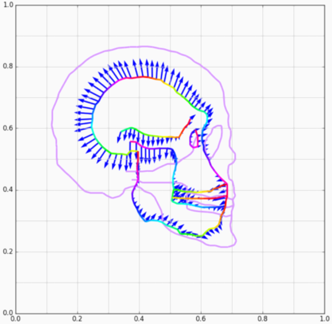
(a) Momentum p_0.



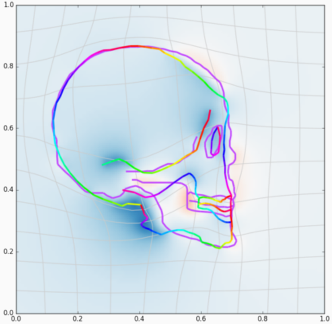
(b) Shooted model q_1.

Figure 27: Iteration 22.

Typical run with kernel fidelity

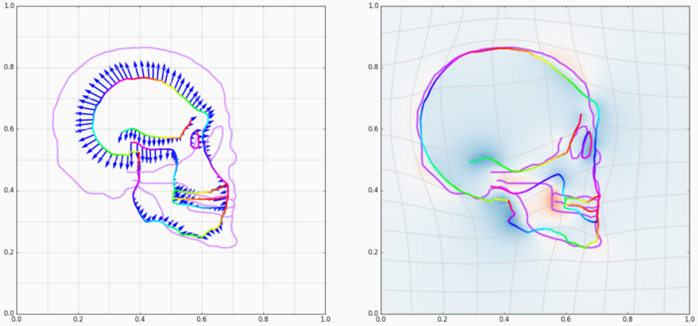


(a) Momentum p_0.



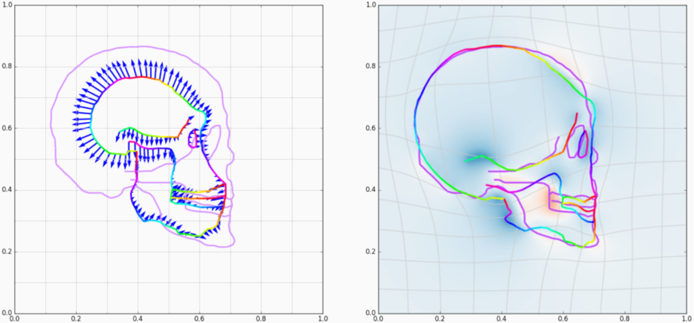
(b) Shooted model q_1.

Figure 27: Iteration 23.



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

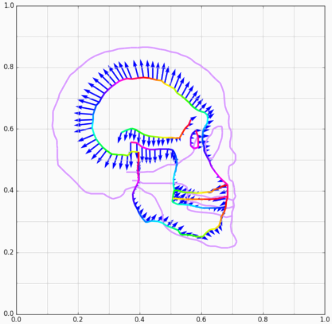
Figure 27: Iteration 24.



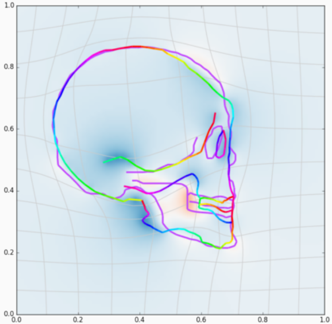
(a) Momentum p_0. (b) Shooted model q_1.

Figure 27: Iteration 25.

Typical run with kernel fidelity



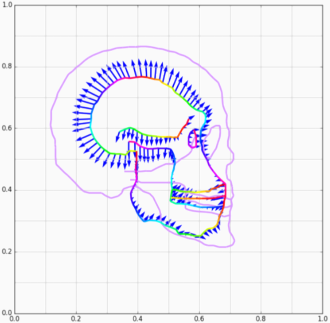
(a) Momentum p_0.



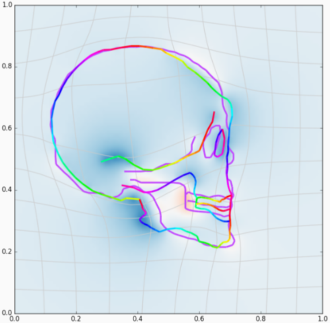
(b) Shooted model q_1.

Figure 27: Iteration 26.

Typical run with kernel fidelity



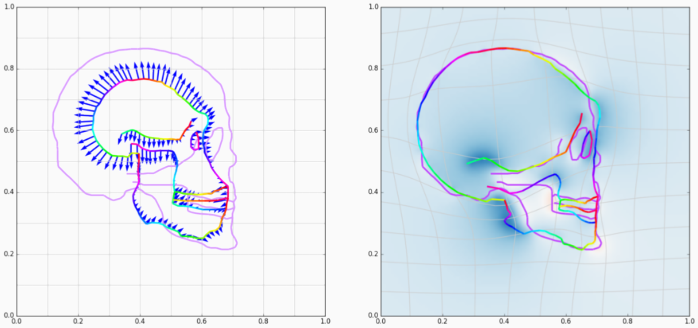
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 27.

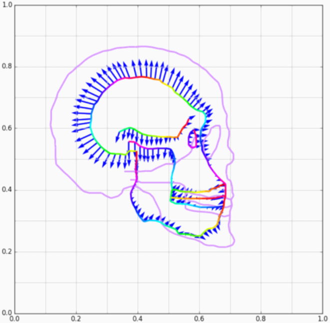
Typical run with kernel fidelity



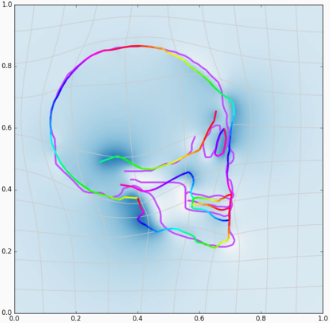
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 27: Iteration 28.

Typical run with kernel fidelity



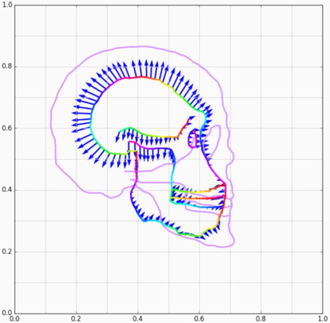
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 30.

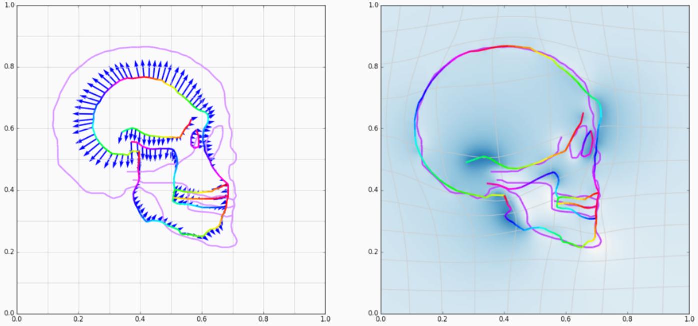
Typical run with kernel fidelity



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

Figure 27: Iteration 31.

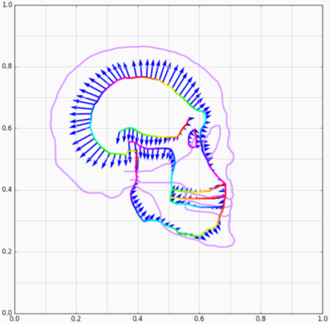
Typical run with kernel fidelity



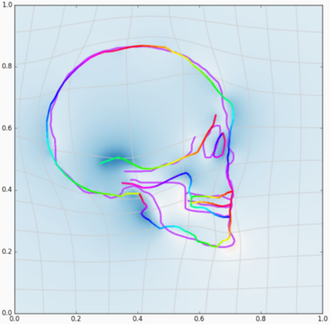
(a) Momentum p_0. (b) Shooted model q_1.

Figure 27: Iteration 32.

Typical run with kernel fidelity



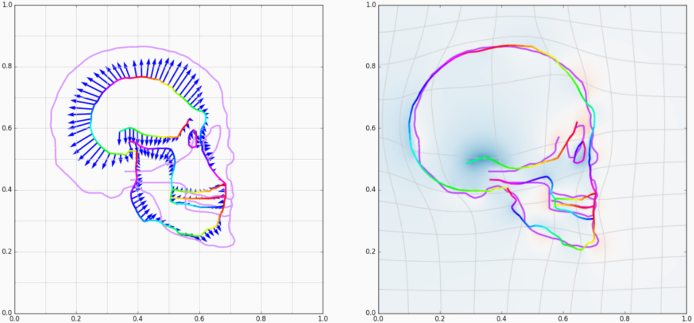
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 33.

Typical run with kernel fidelity

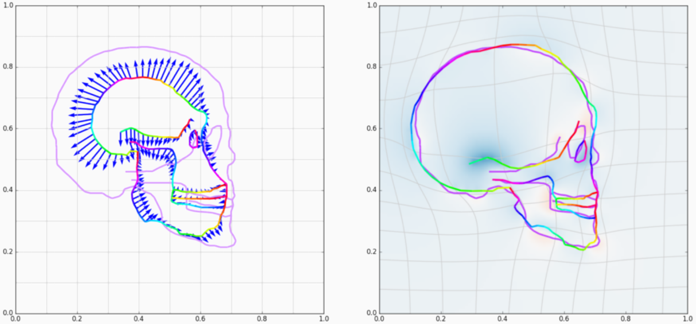


(a) Momentum p_0.

(b) Shooted model q_1.

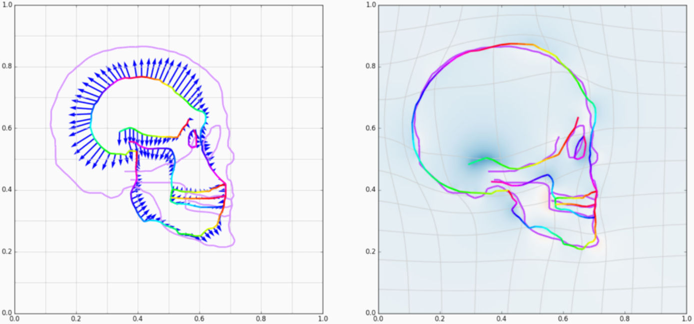
Figure 27: Iteration 34.

Typical run with kernel fidelity



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

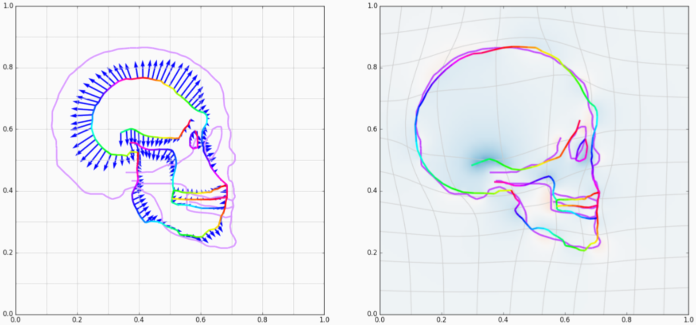
Figure 27: Iteration 36.



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

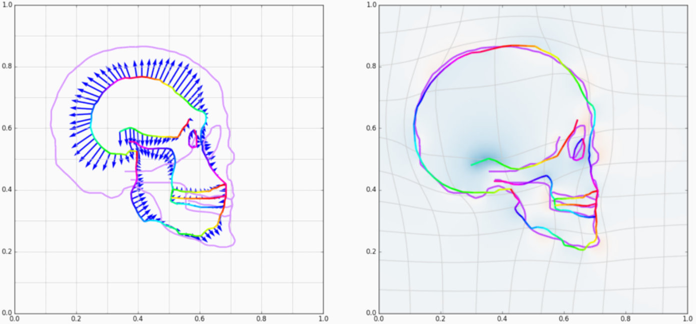
Figure 27: Iteration 37.

Typical run with kernel fidelity



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

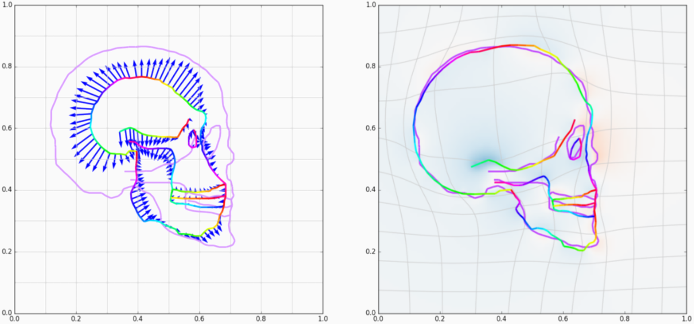
Figure 27: Iteration 38.



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

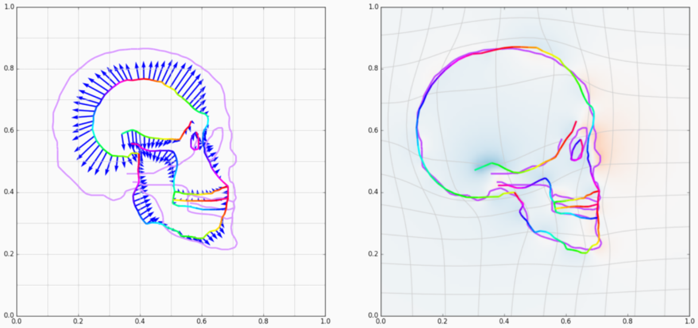
Figure 27: Iteration 39.

Typical run with kernel fidelity



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

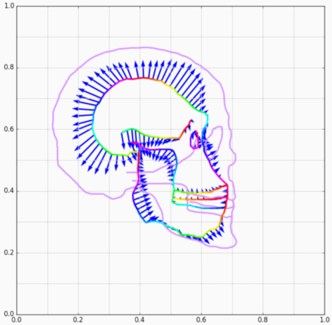
Figure 27: Iteration 40.



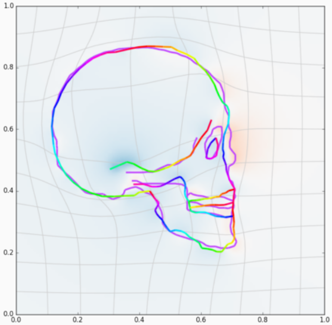
(a) Momentum p_0. (b) Shooted model q_1.

Figure 27: Iteration 41.

Typical run with kernel fidelity



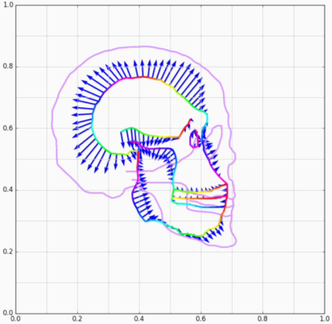
(a) Momentum p_0.



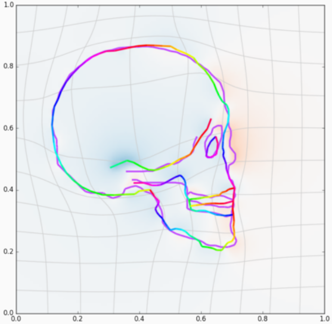
(b) Shooted model q_1.

Figure 27: Iteration 42.

Typical run with kernel fidelity



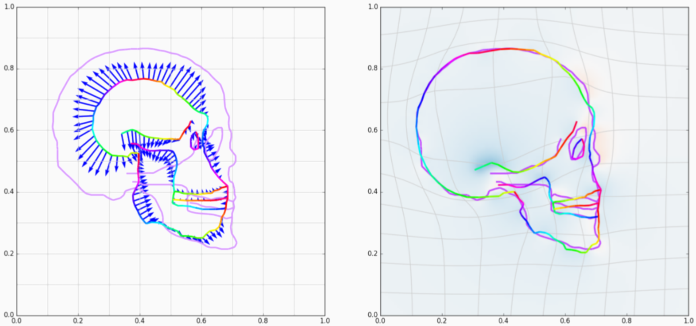
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 44.

Typical run with kernel fidelity

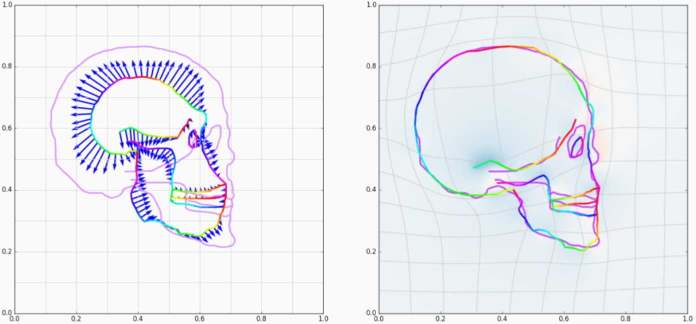


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 27: Iteration 45.

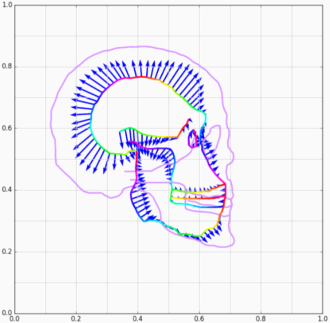
Typical run with kernel fidelity



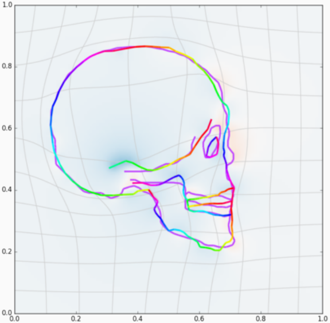
(a) Momentum p_0. (b) Shooted model q_1.

Figure 27: Iteration 46.

Typical run with kernel fidelity



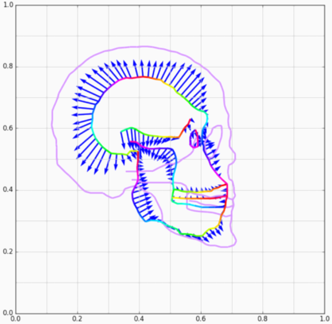
(a) Momentum p_0.



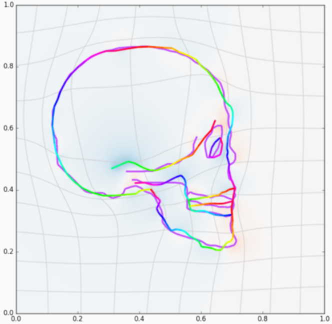
(b) Shooted model q_1.

Figure 27: Iteration 47.

Typical run with kernel fidelity



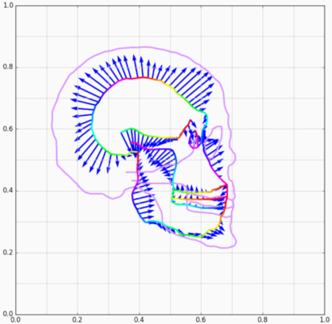
(a) Momentum p_0.



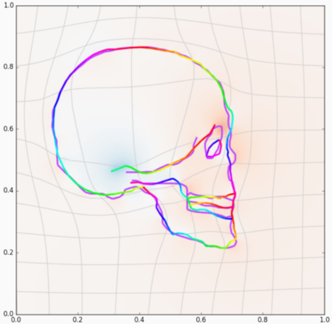
(b) Shooted model q_1.

Figure 27: Iteration 50.

Typical run with kernel fidelity



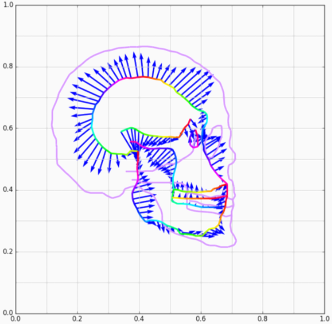
(a) Momentum p_0.



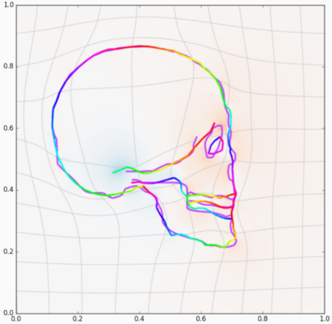
(b) Shooted model q_1.

Figure 27: Iteration 70.

Typical run with kernel fidelity



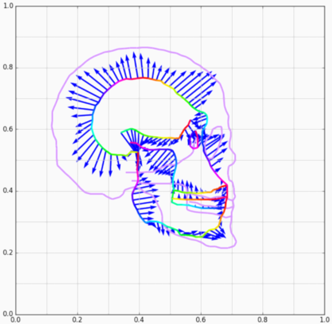
(a) Momentum p_0.



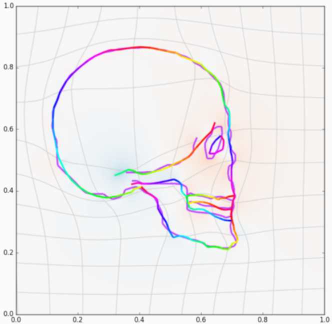
(b) Shooted model q_1.

Figure 27: Iteration 90.

Typical run with kernel fidelity



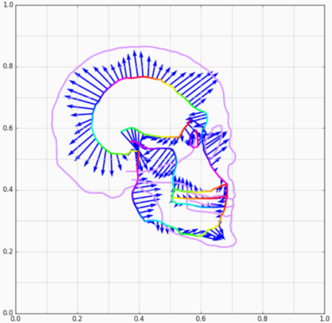
(a) Momentum p_0.



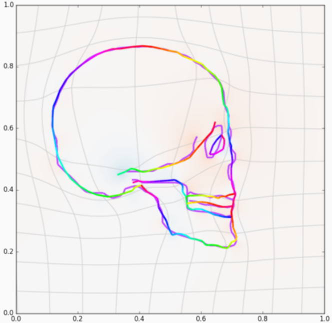
(b) Shooted model q_1.

Figure 27: Iteration 110.

Typical run with kernel fidelity



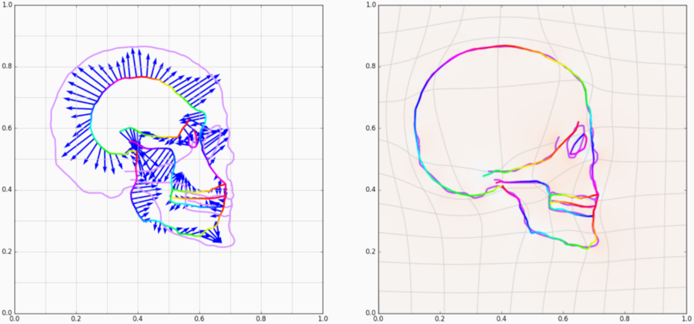
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 130.

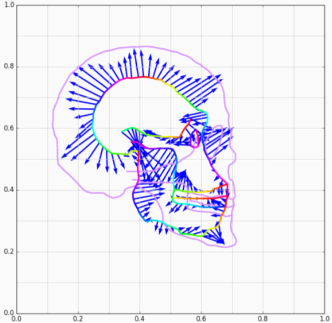
Typical run with kernel fidelity



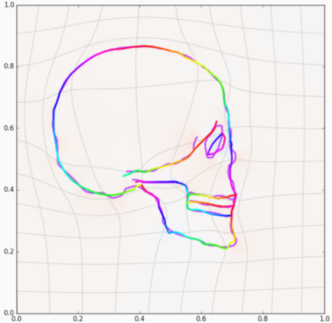
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 27: Iteration 150.

Typical run with kernel fidelity



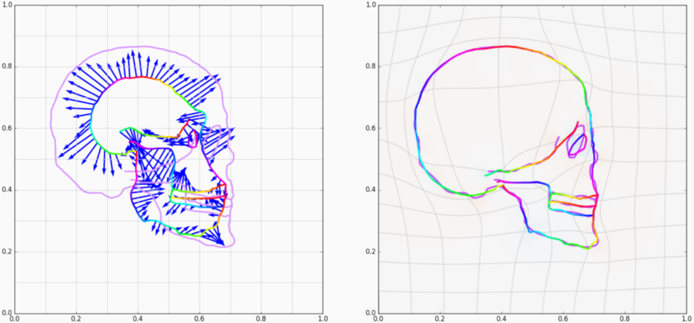
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 27: Iteration 170.

Typical run with kernel fidelity

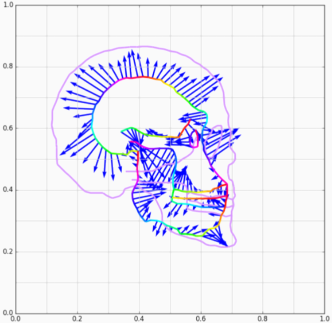


(a) Momentum p_0.

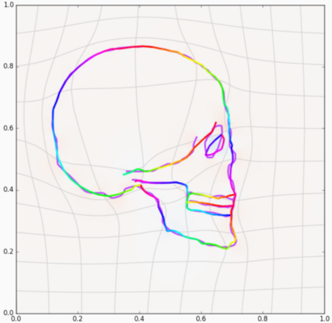
(b) Shooted model q_1.

Figure 27: Iteration 200.

Typical run with kernel fidelity



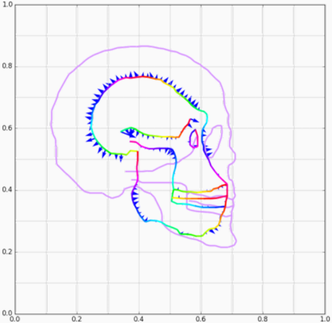
(a) Momentum p_0.



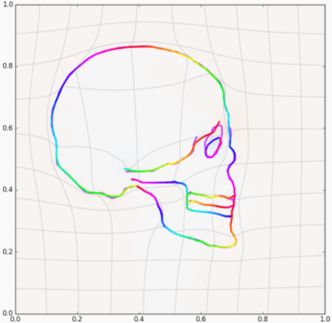
(b) Shooted model q_1.

Figure 27: Iteration 240.

Influence of the kernel width



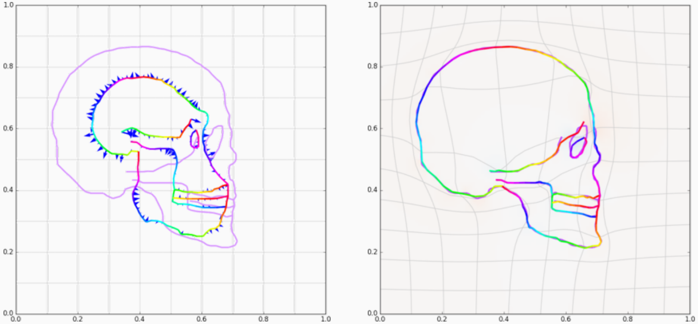
(a) Momentum p_0.



(b) Shooted model q_1.

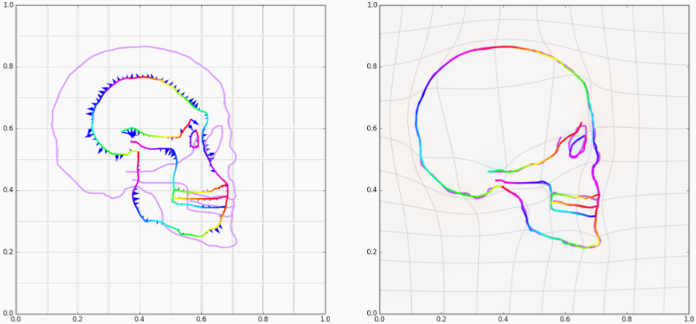
Figure 28: Final matching, $\sigma = \texttt{.01}$.

Influence of the kernel width



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

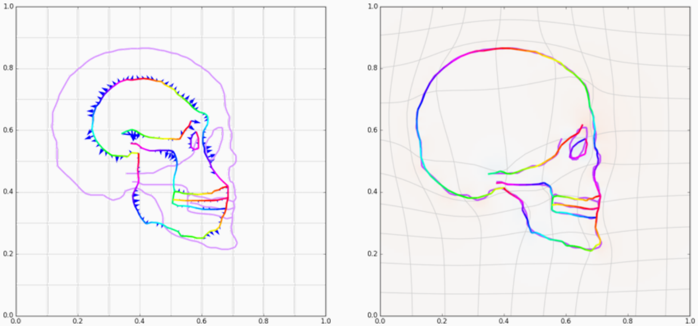
Figure 28: Final matching, $\sigma = \texttt{.02}$.



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

Figure 28: Final matching, $\sigma = \texttt{.03}$.

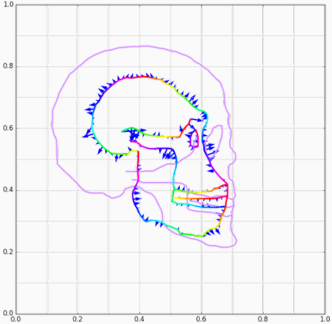
Influence of the kernel width



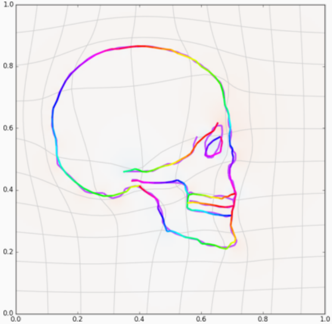
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.04}$.

Influence of the kernel width



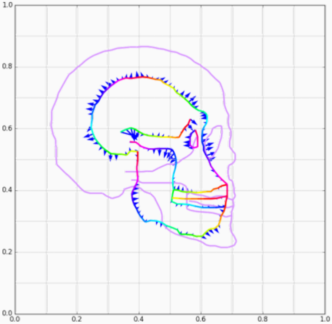
(a) Momentum p_0.



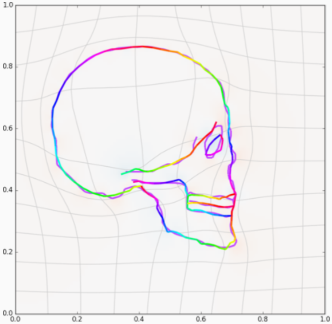
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.05}$.

Influence of the kernel width



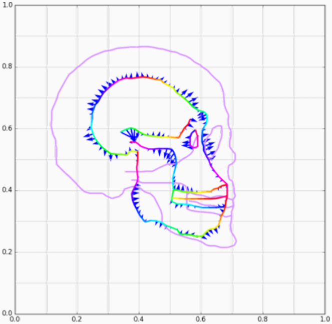
(a) Momentum p_0.



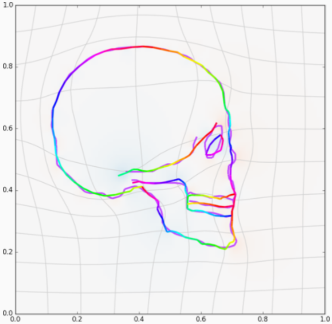
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.06}$.

Influence of the kernel width



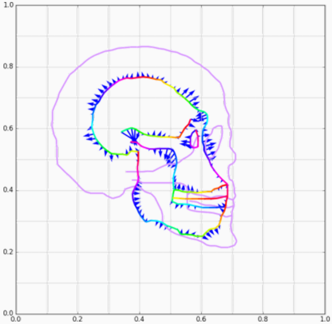
(a) Momentum p_0.



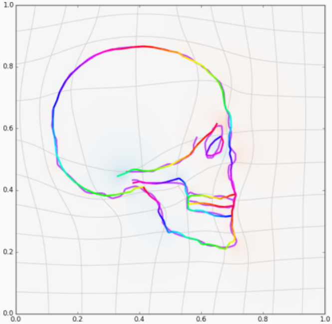
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.07}$.

Influence of the kernel width



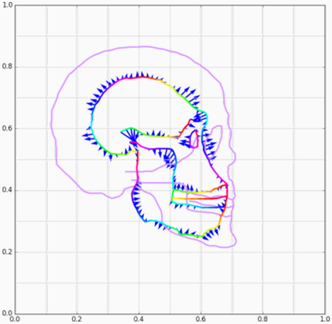
(a) Momentum p_0.



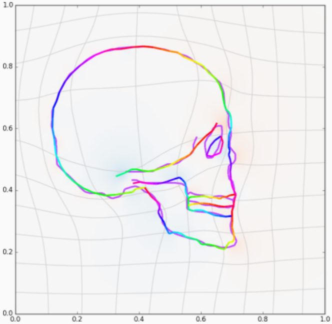
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.08}$.

Influence of the kernel width



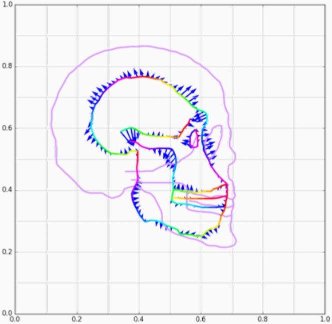
(a) Momentum p_0.



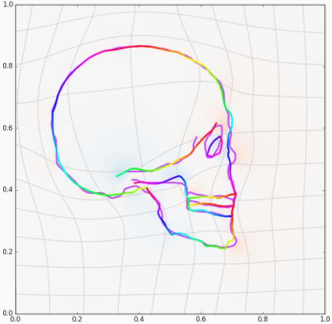
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.09}$.

Influence of the kernel width



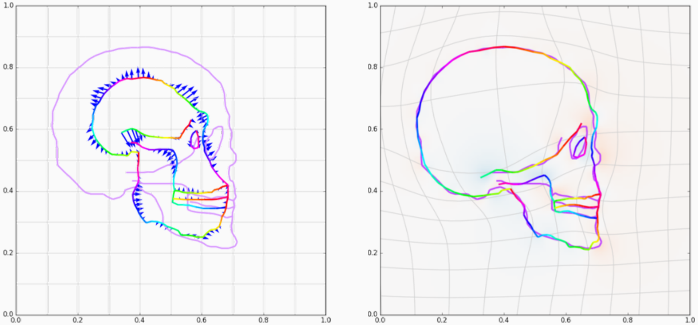
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .1$.

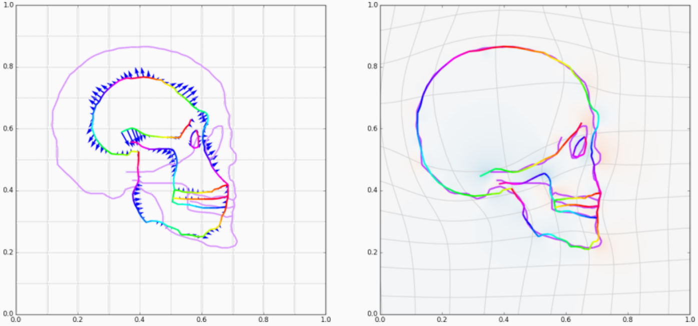
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .11$.

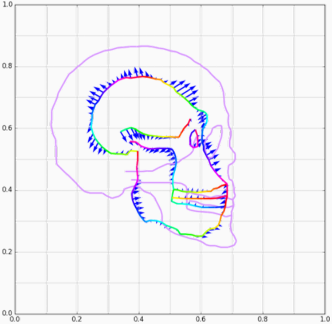
Influence of the kernel width



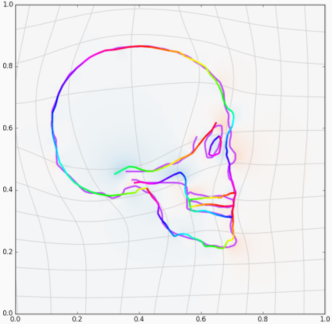
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .12$.

Influence of the kernel width



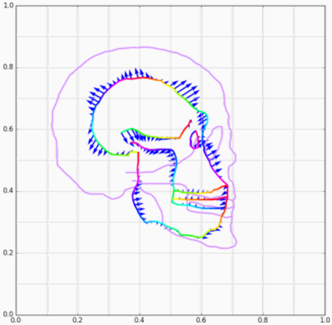
(a) Momentum p_0.



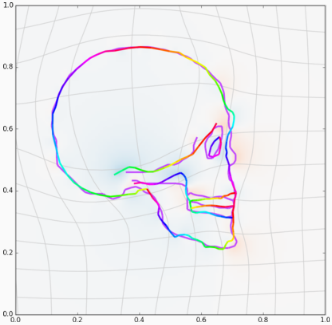
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .13$.

Influence of the kernel width



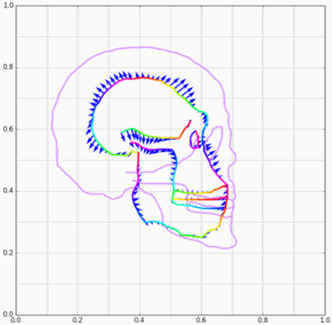
(a) Momentum p_0.



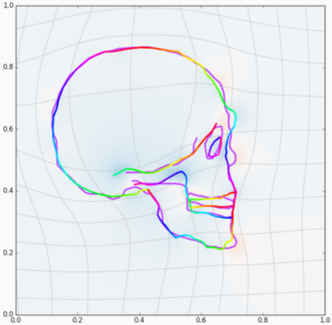
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .14$.

Influence of the kernel width



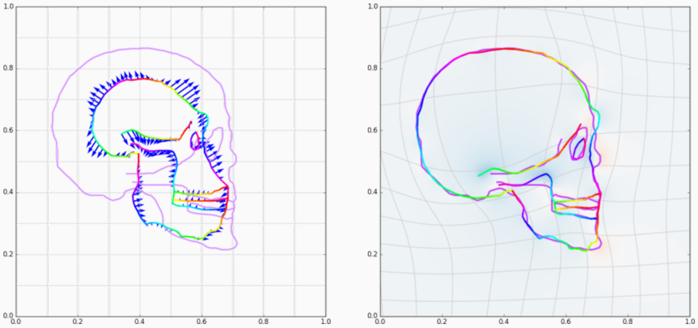
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .15$.

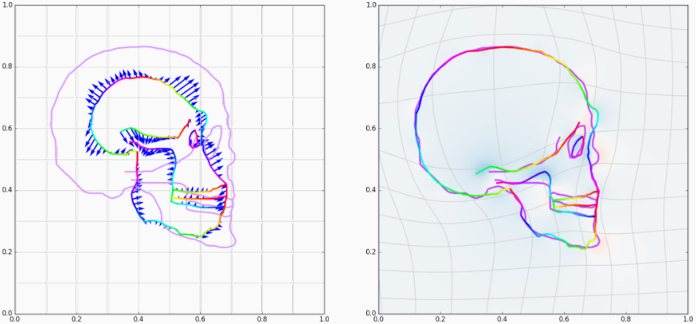
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .16$.

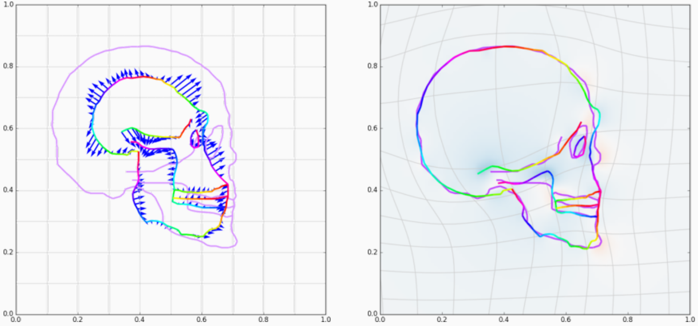
Influence of the kernel width



(a) Momentum p_0.

(b) Shooted model q_1.

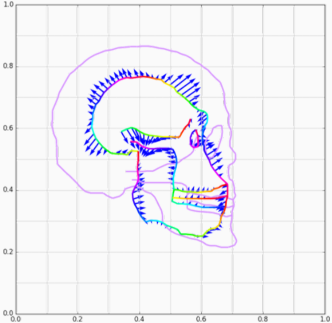
Figure 28: Final matching, $\sigma = .17$.



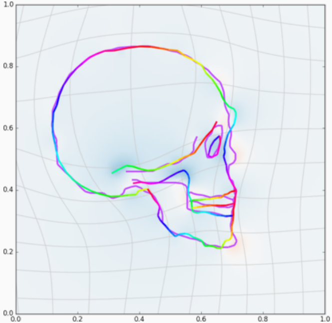
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .18$.

Influence of the kernel width

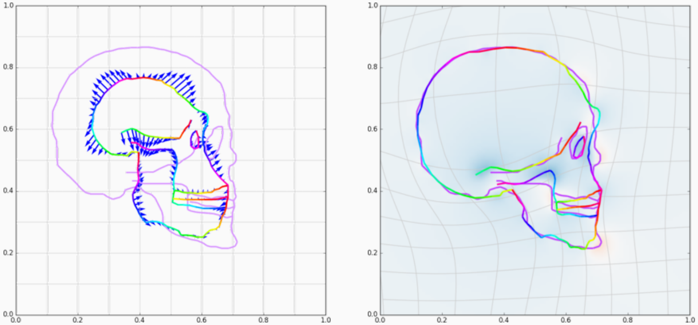


(a) Momentum p_0.



(b) Shooted model q_1.

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

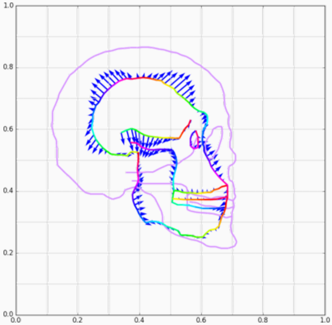


(a) Momentum p_0.

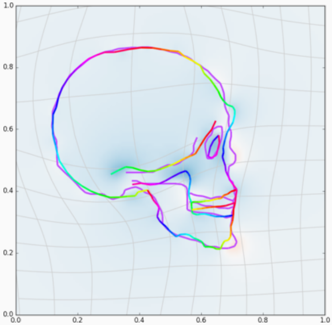
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .2$.

Influence of the kernel width



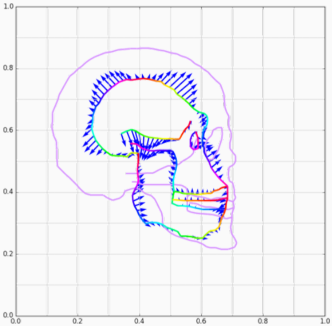
(a) Momentum p_0.



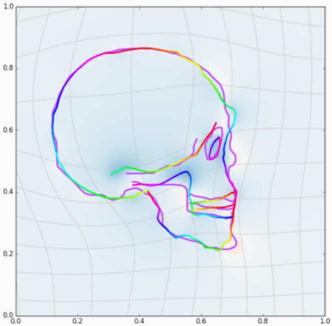
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .21$.

Influence of the kernel width



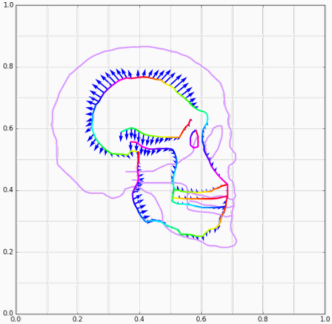
(a) Momentum p_0.



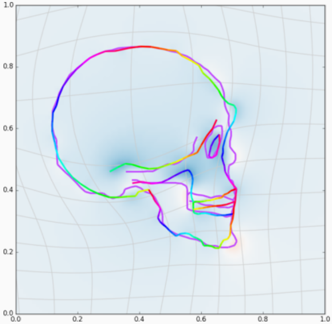
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .22$.

Influence of the kernel width



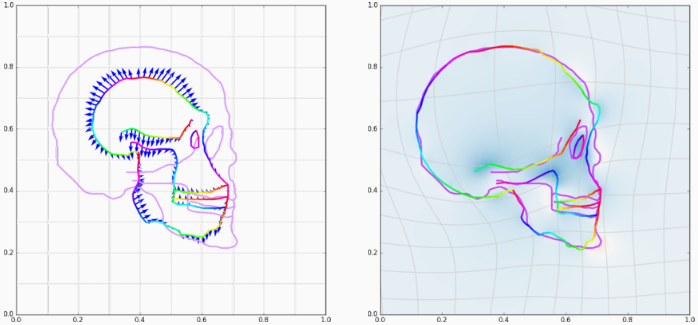
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .23$.

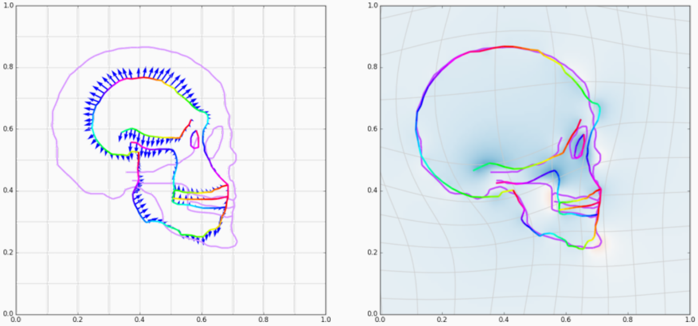
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = \mathtt{.24}$.

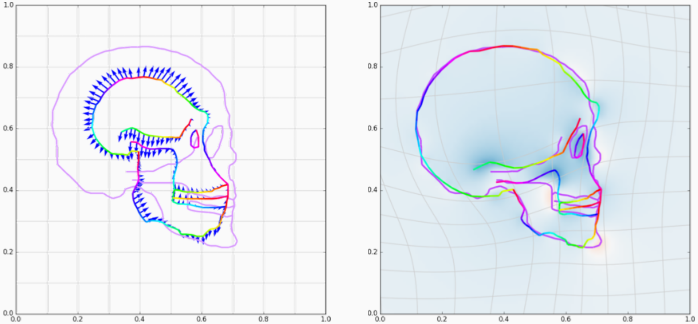
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .25$.

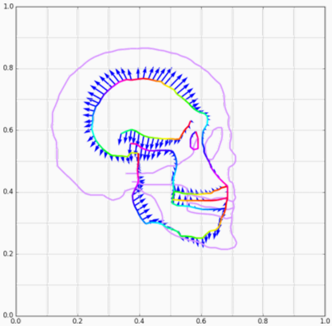
Influence of the kernel width



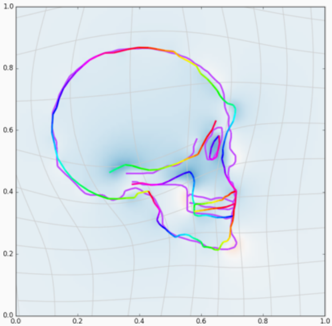
(a) Momentum p_0.　(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .26$.

Influence of the kernel width



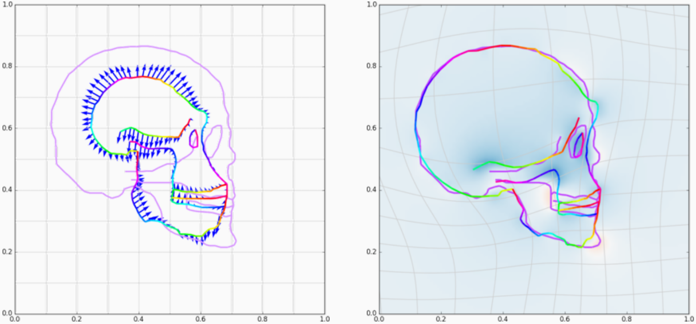
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .27$.

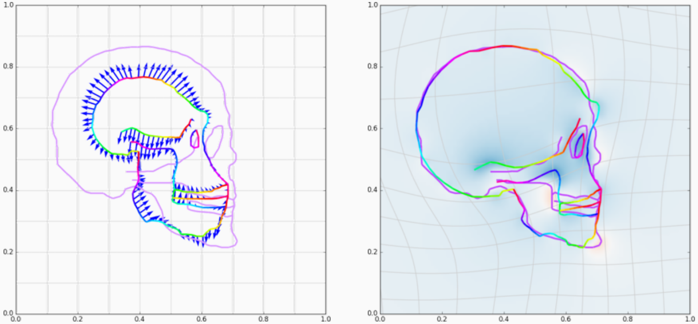
Influence of the kernel width



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

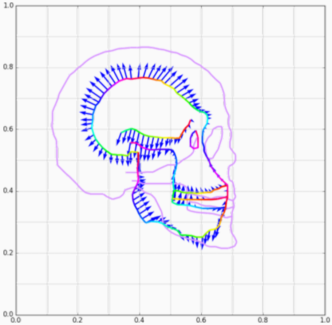
Figure 28: Final matching, $\sigma = .28$.

Influence of the kernel width

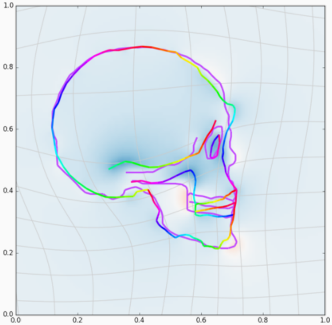


(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .29$.



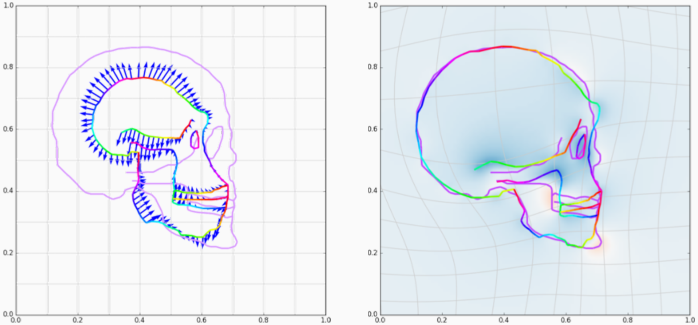
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .3$.

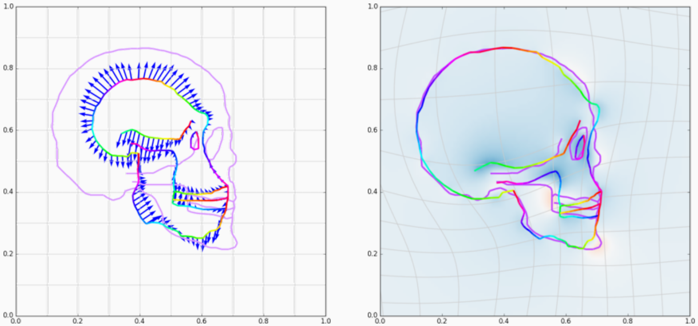
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .31$.

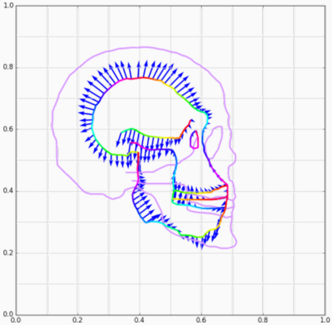
Influence of the kernel width



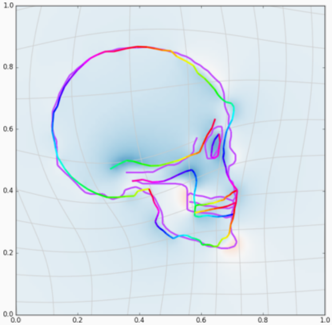
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .32$.

Influence of the kernel width



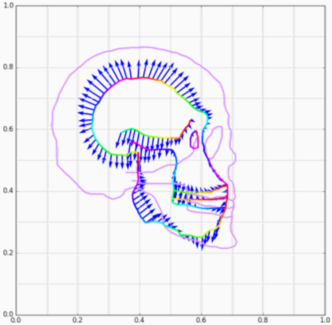
(a) Momentum p_0.



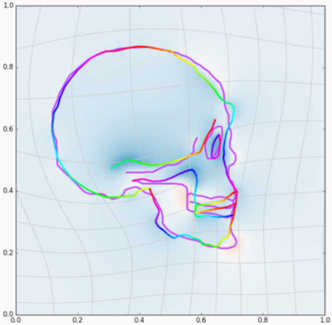
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .33$.

Influence of the kernel width



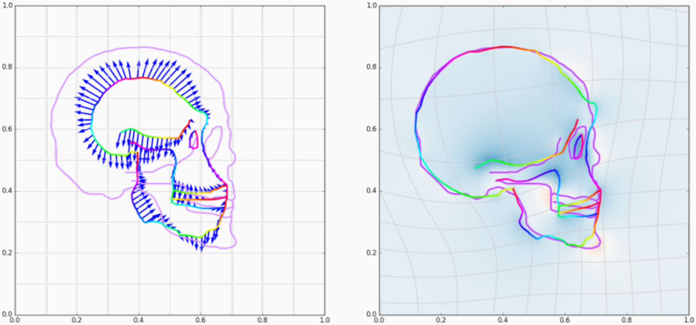
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.34}$.

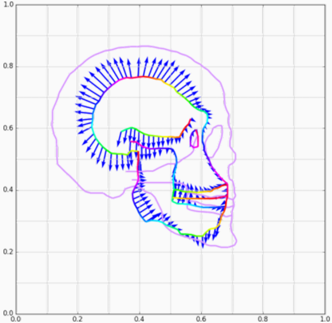
Influence of the kernel width



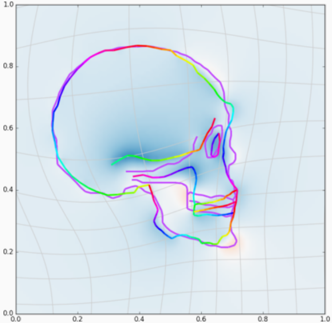
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .35$.

Influence of the kernel width

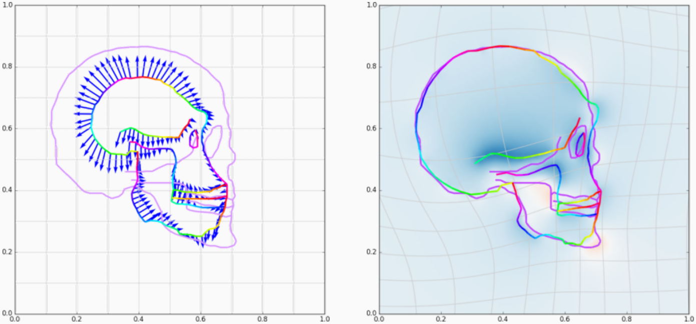


(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .36$.

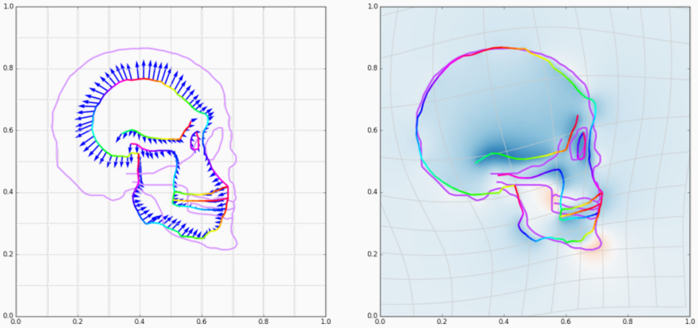


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .37$.

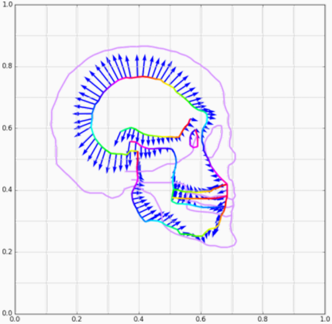
Influence of the kernel width



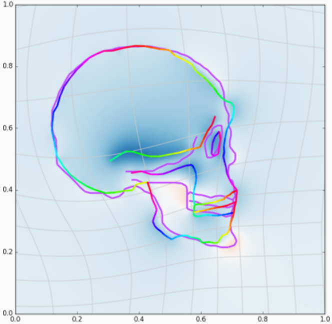
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .38$.

Influence of the kernel width



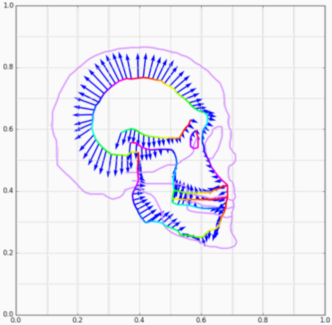
(a) Momentum p_0.



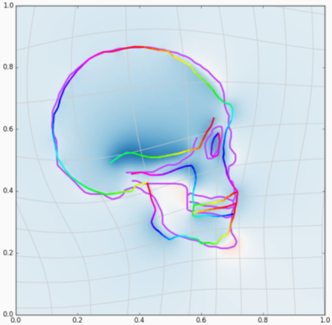
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .39$.

Influence of the kernel width



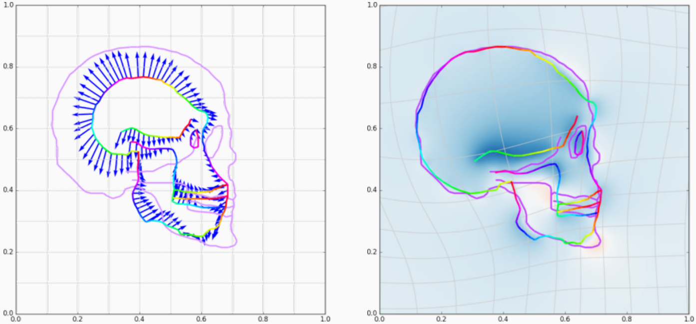
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .4$.

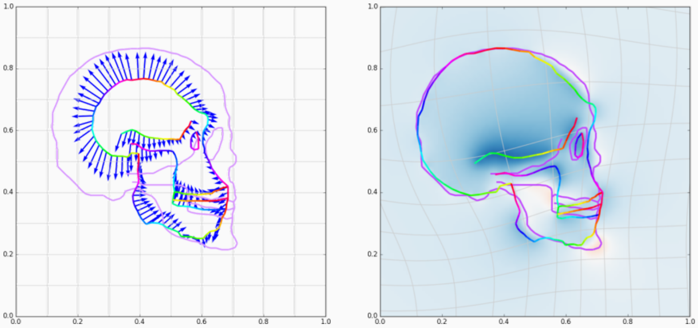
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .41$.

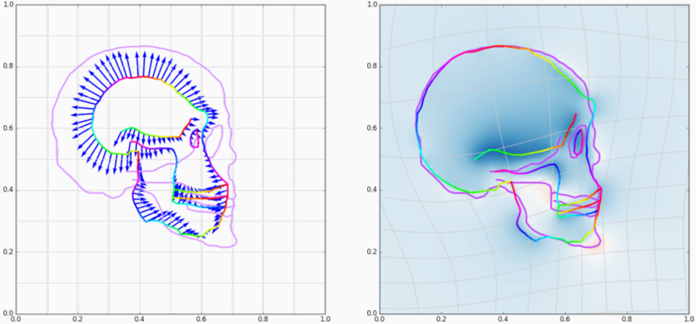
Influence of the kernel width



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

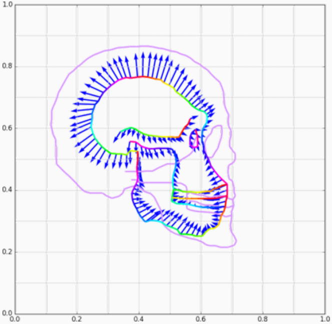
Figure 28: Final matching, $\sigma = .42$.

Influence of the kernel width

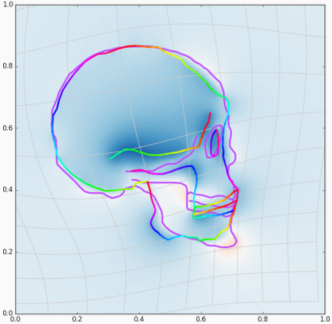


(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .43$.



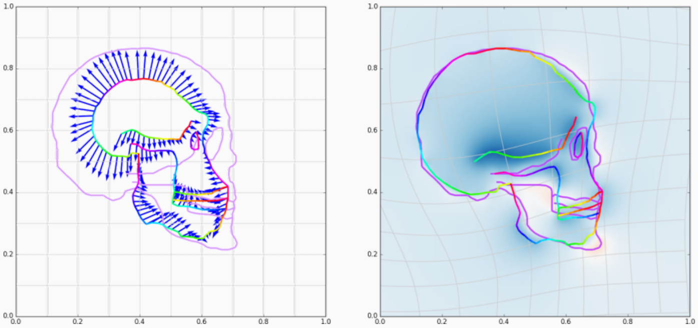
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .44$.

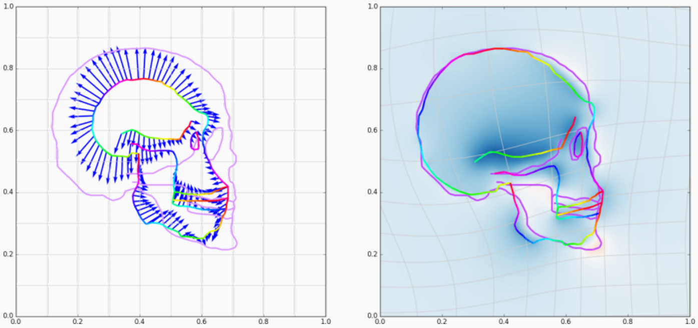
Influence of the kernel width



(a) Momentum p_0.

(b) Shooted model q_1.

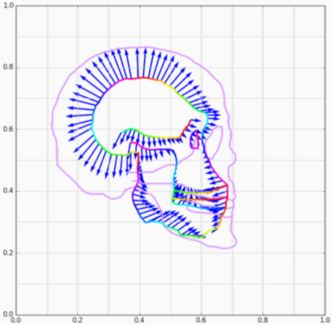
Figure 28: Final matching, $\sigma = .45$.



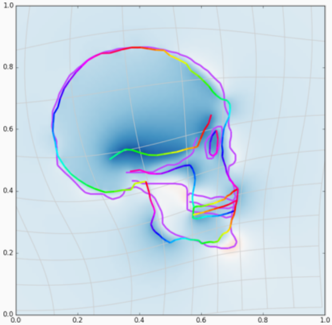
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .46$.

Influence of the kernel width



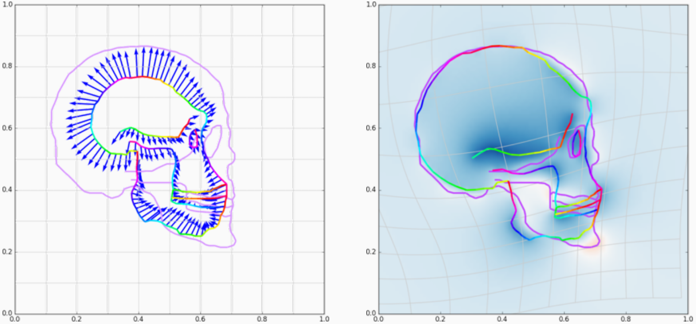
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .47$.

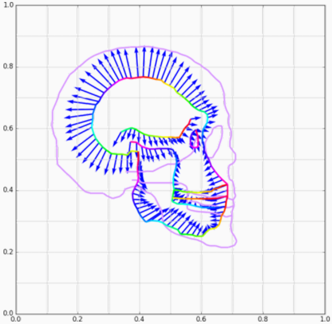
Influence of the kernel width



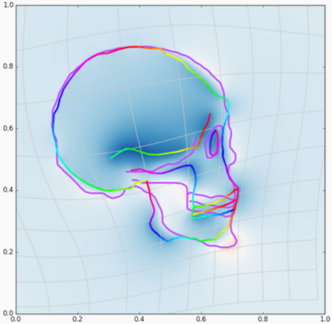
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .48$.

Influence of the kernel width



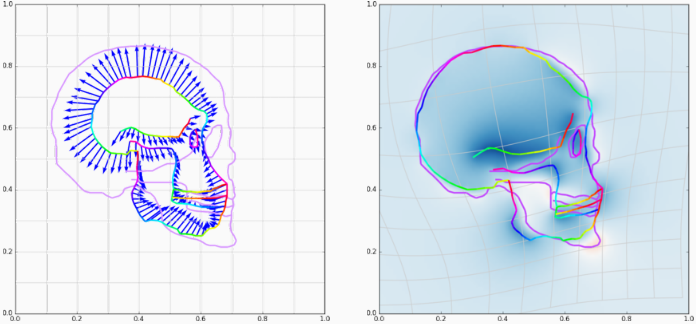
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .49$.

Influence of the kernel width

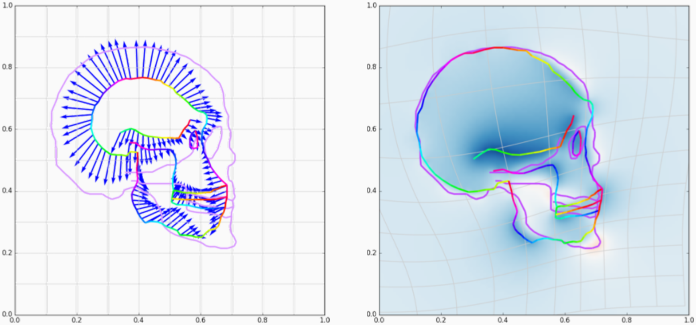


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .5$.

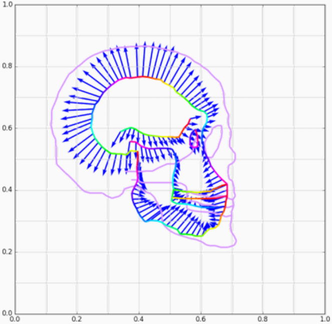
Influence of the kernel width



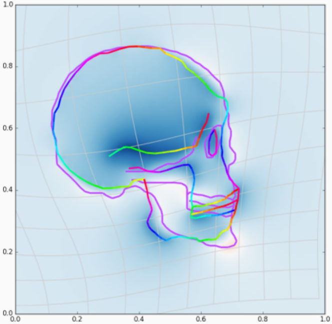
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \mathbf{.51}$.

Influence of the kernel width



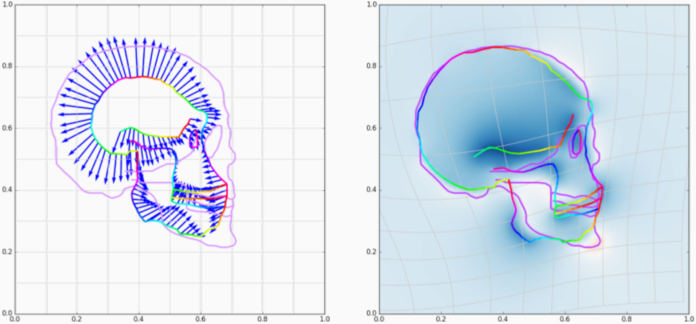
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .52$.

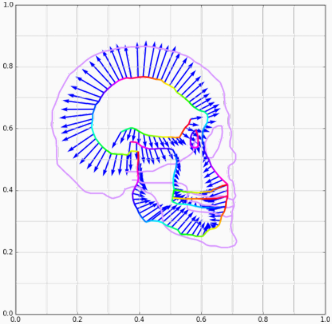
Influence of the kernel width



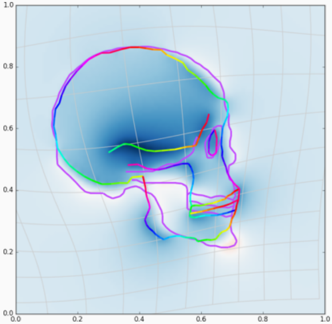
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .53$.

Influence of the kernel width



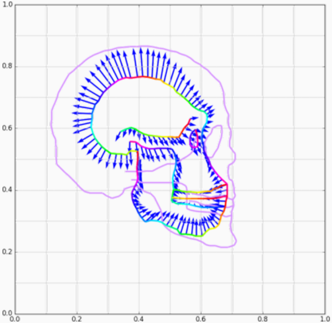
(a) Momentum p_0.



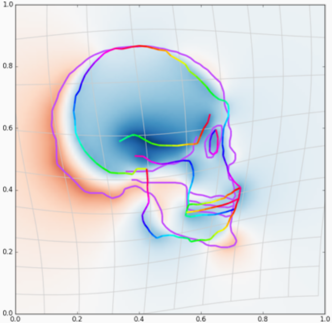
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .54$.

Influence of the kernel width



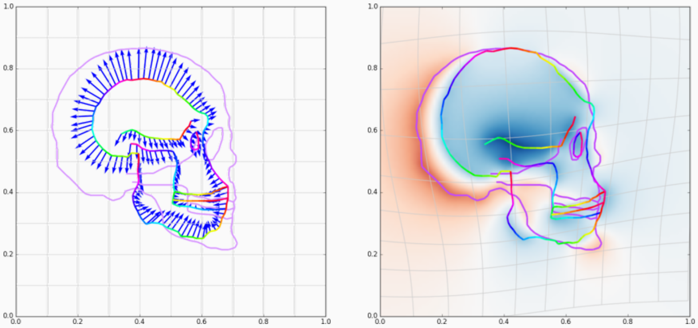
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .55$.

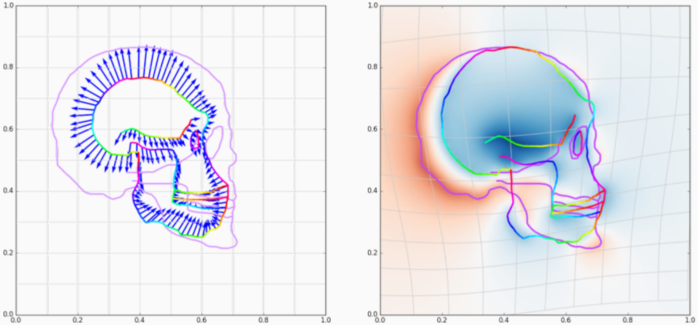
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .56$.

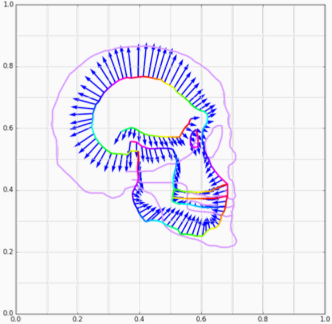
Influence of the kernel width



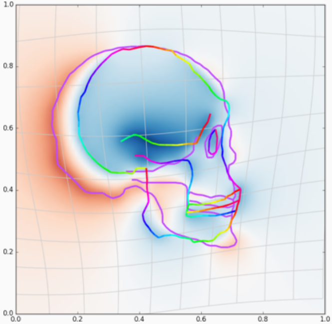
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .57$.

Influence of the kernel width



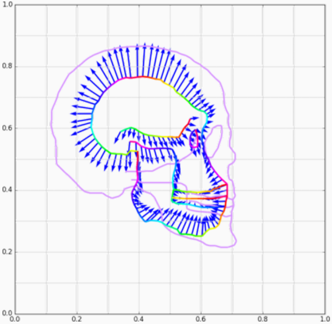
(a) Momentum p_0.



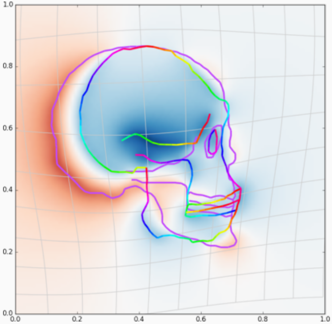
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .58$.

Influence of the kernel width



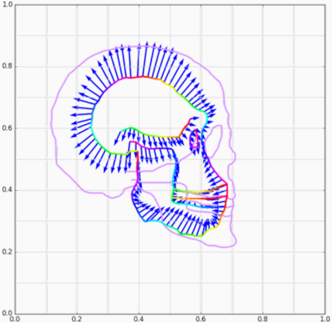
(a) Momentum p_0.



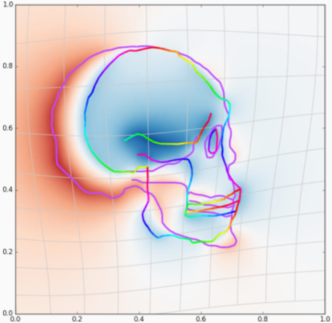
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .59$.

Influence of the kernel width



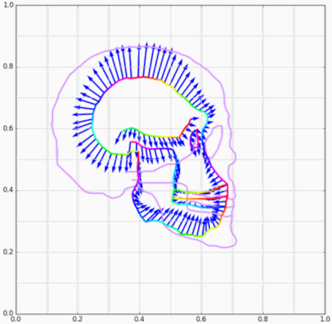
(a) Momentum p_0.



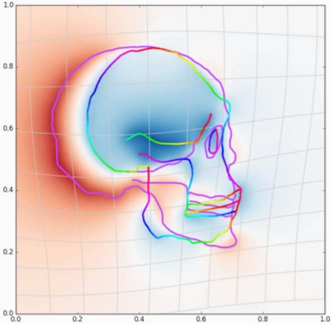
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .6$.

Influence of the kernel width



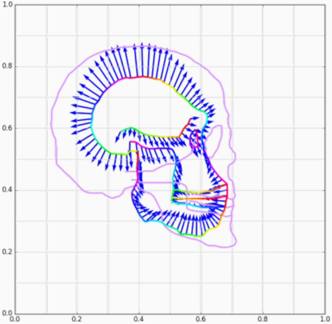
(a) Momentum p_0.



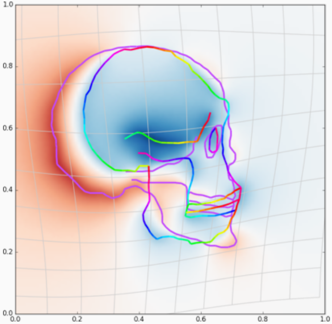
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .61$.

Influence of the kernel width



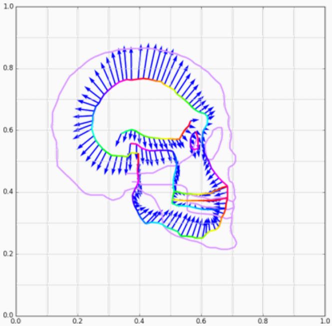
(a) Momentum p_0.



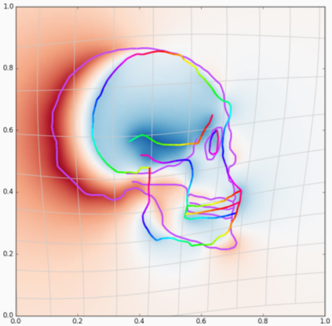
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .62$.

Influence of the kernel width



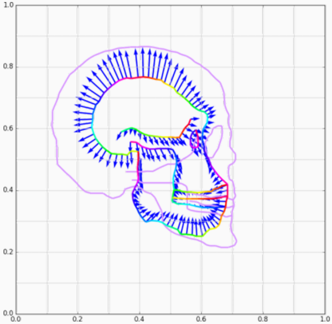
(a) Momentum p_0.



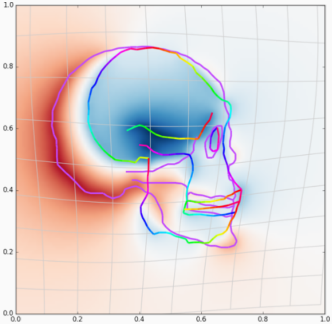
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .63$.

Influence of the kernel width

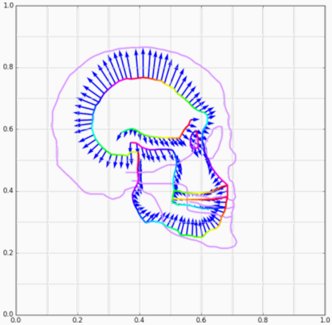


(a) Momentum p_0.

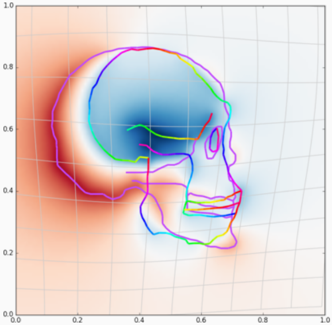


(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .64$.



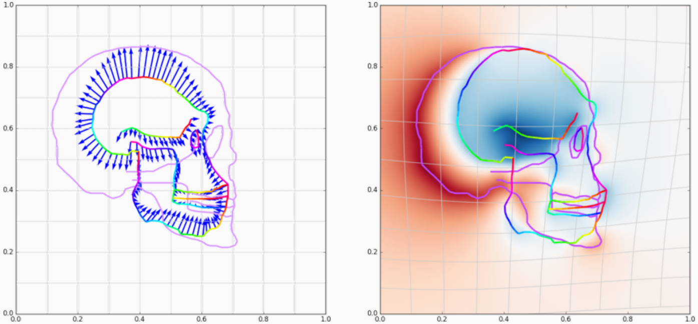
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .65$.

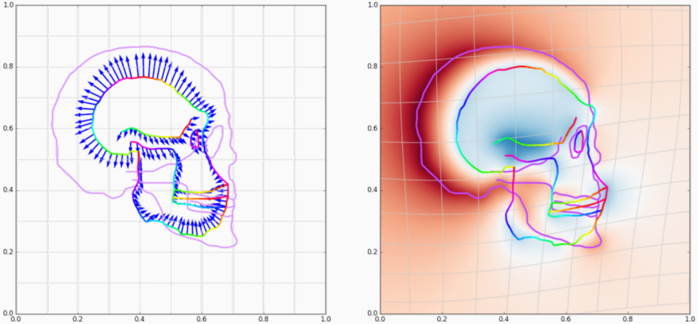
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .66$.

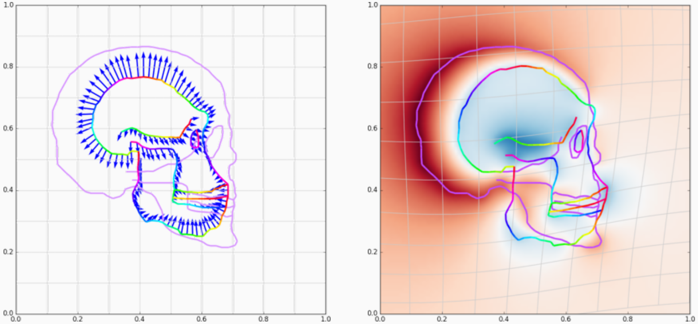
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .67$.

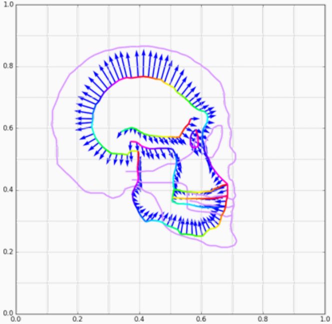
Influence of the kernel width



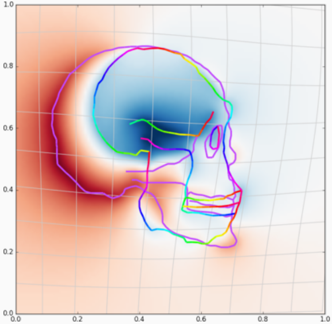
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .68$.

Influence of the kernel width



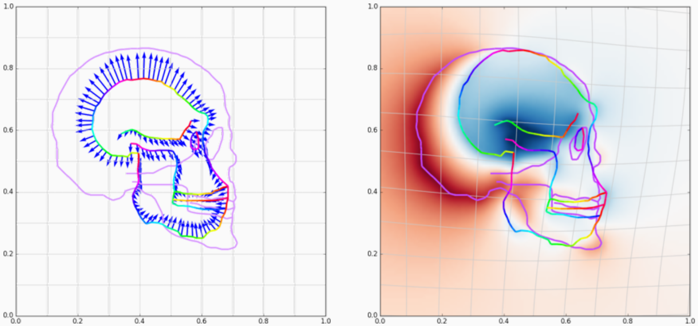
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .69$.

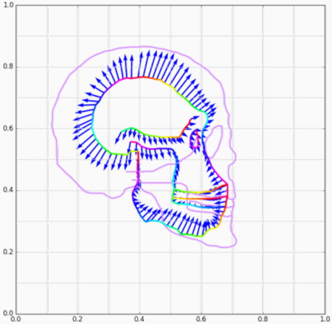
Influence of the kernel width



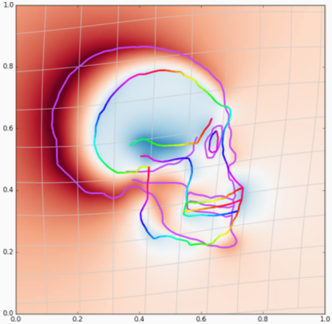
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \texttt{.70}$.

Influence of the kernel width



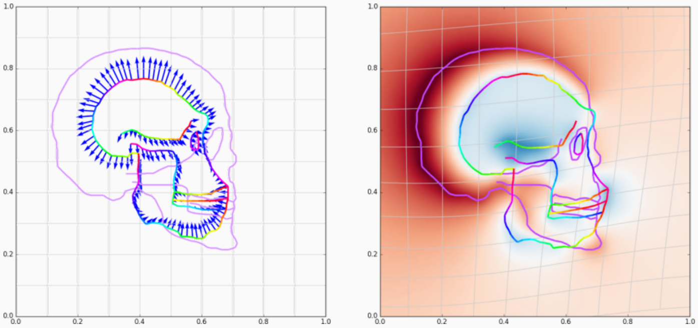
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .71$.

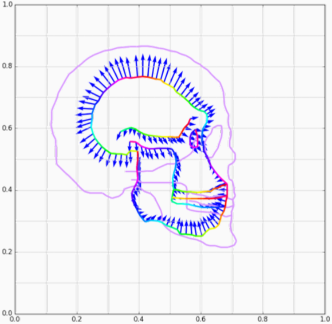
Influence of the kernel width



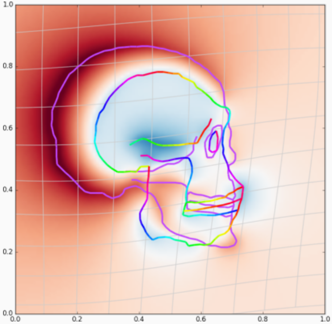
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .72$.

Influence of the kernel width



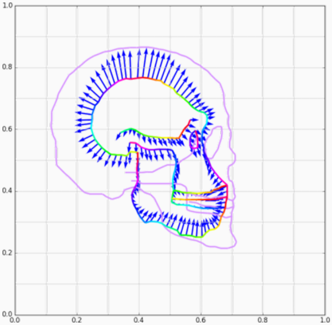
(a) Momentum p_0.



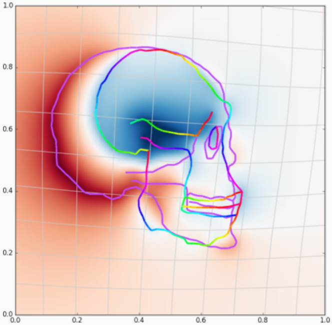
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .73$.

Influence of the kernel width



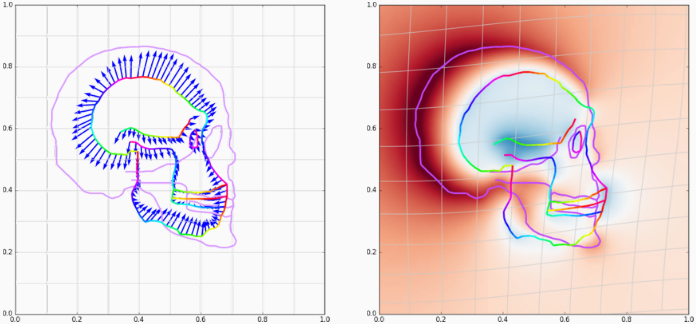
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .74$.

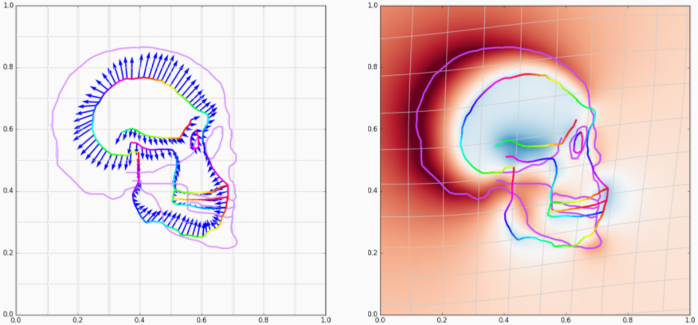
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .75$.

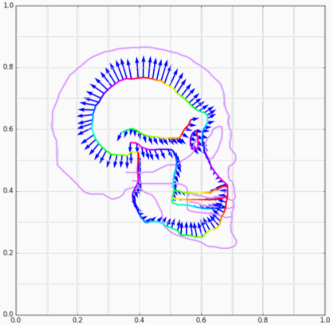
Influence of the kernel width



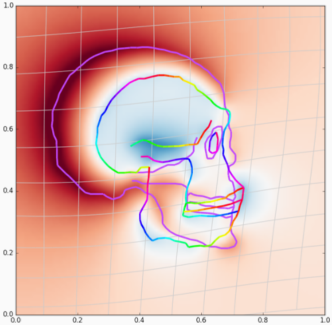
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .76$.

Influence of the kernel width



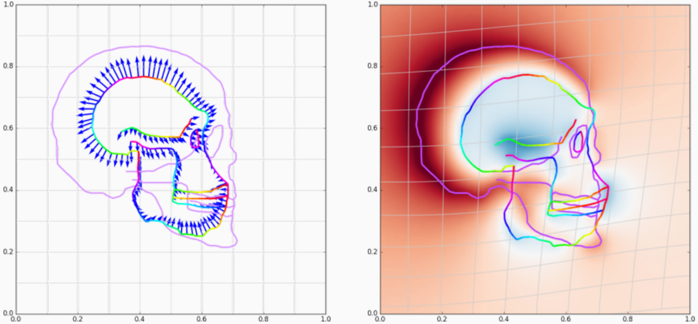
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .77$.

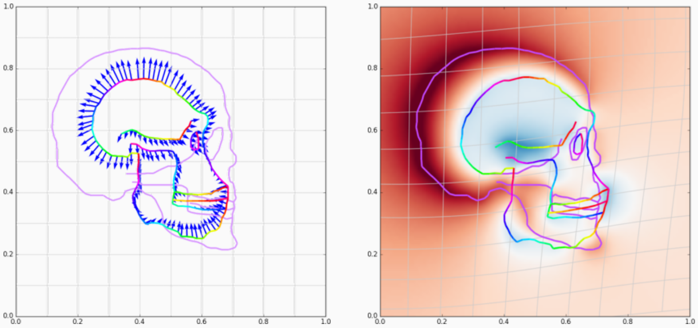
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .78$.

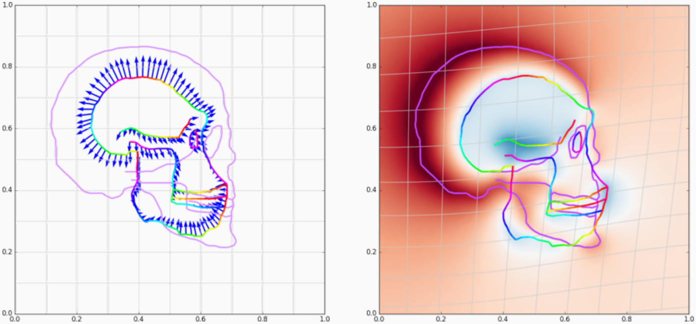
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .79$.

Influence of the kernel width

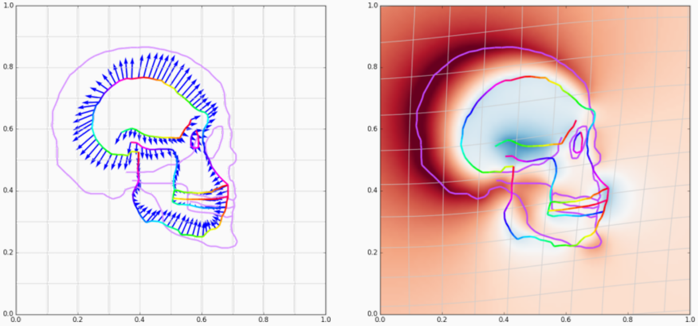


(a) Momentum p_0.

(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .8$.

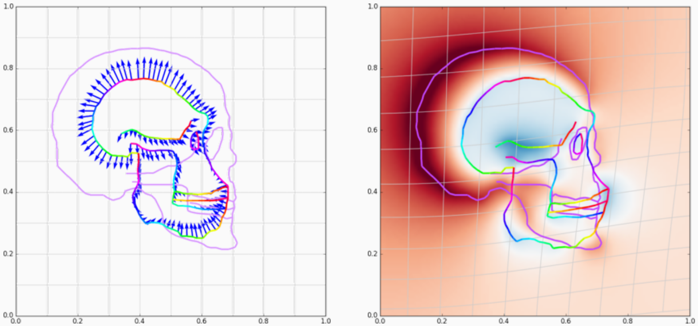
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .81$.

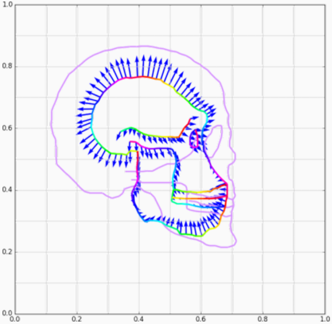
Influence of the kernel width



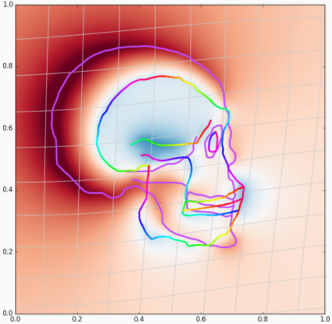
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .82$.

Influence of the kernel width



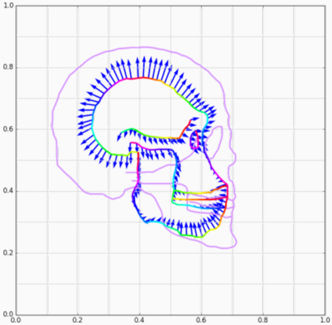
(a) Momentum p_0.



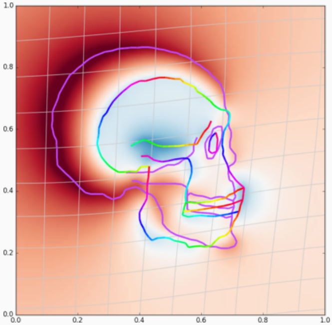
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .83$.

Influence of the kernel width



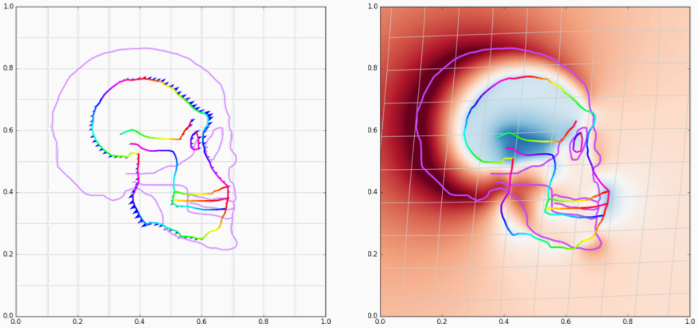
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .84$.

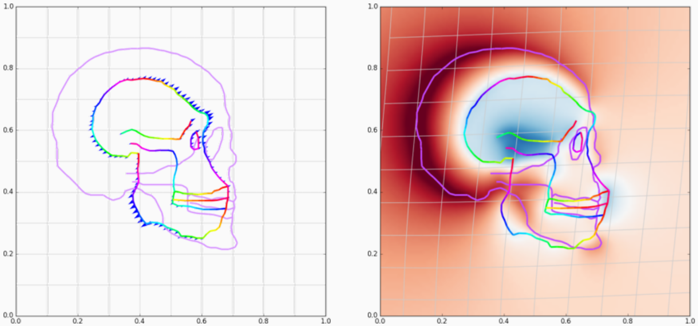
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .85$.

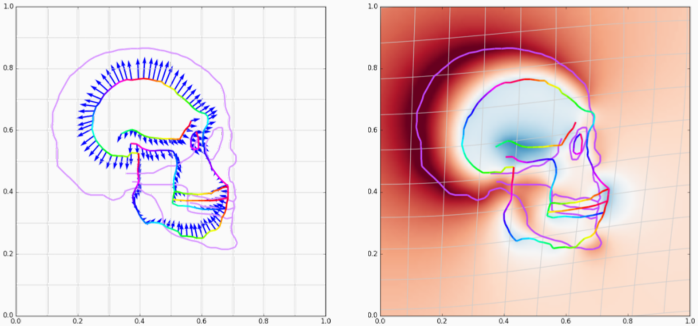
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .86$.

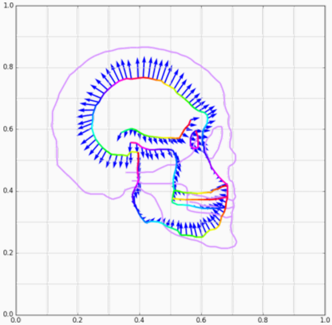
Influence of the kernel width



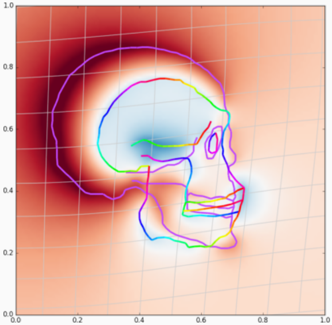
(a) Momentum p_0.   (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .87$.

Influence of the kernel width



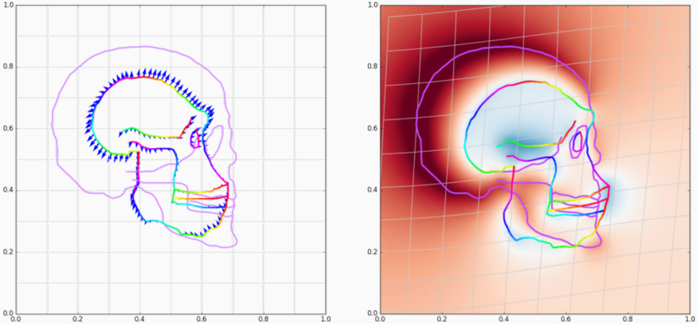
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .88$.

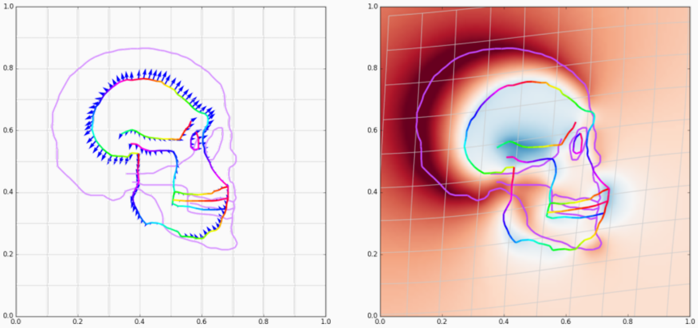
Influence of the kernel width



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

Figure 28: Final matching, $\sigma = .89$.

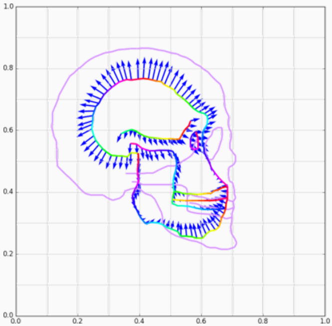
Influence of the kernel width



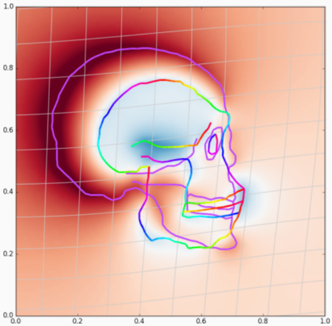
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .9$.

Influence of the kernel width



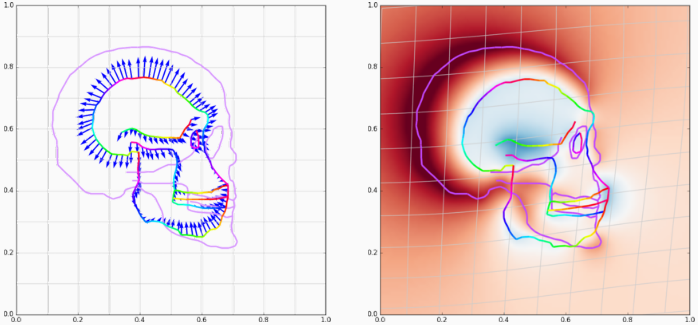
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .91$.

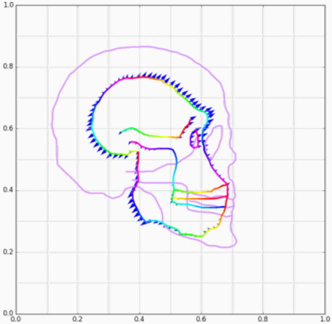
Influence of the kernel width



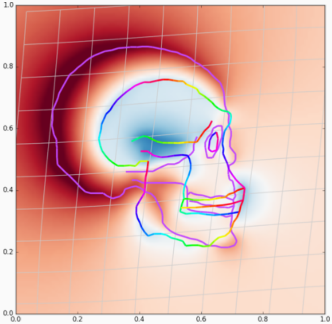
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .92$.

Influence of the kernel width



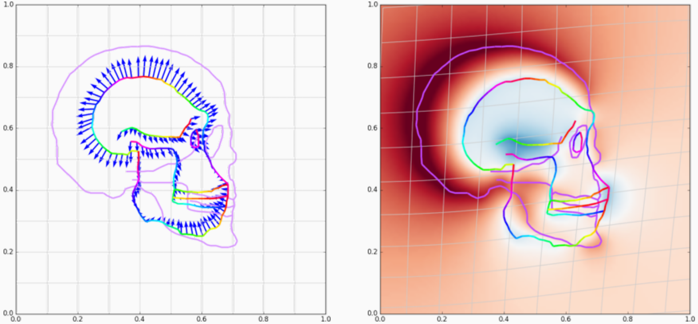
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .93$.

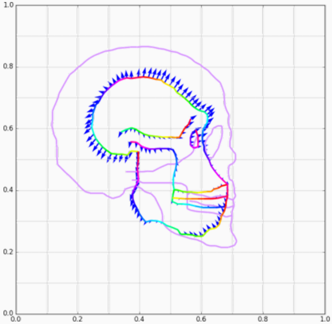
Influence of the kernel width



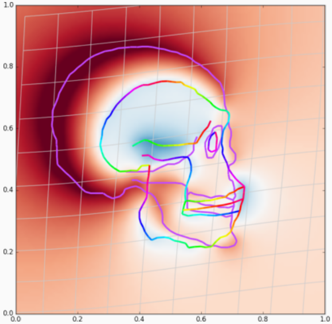
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .94$.

Influence of the kernel width



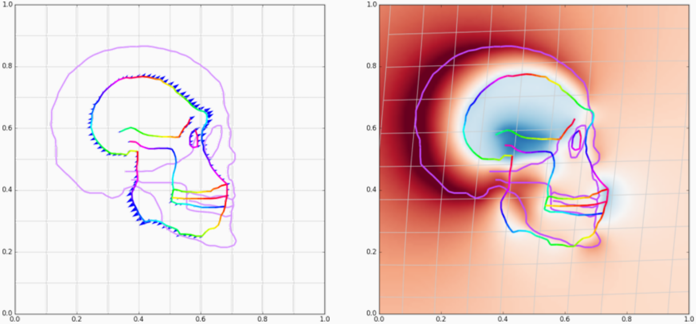
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .95$.

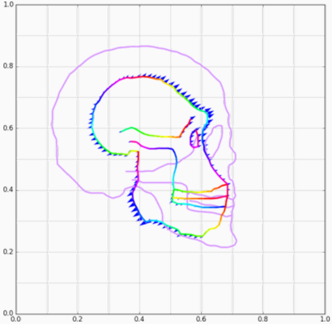
Influence of the kernel width



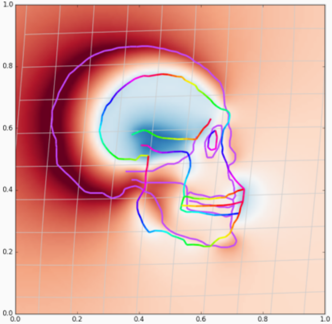
(a) Momentum p_0. (b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .96$.

Influence of the kernel width



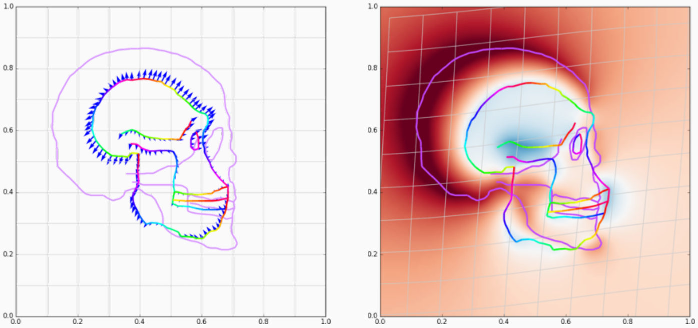
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .97$.

Influence of the kernel width

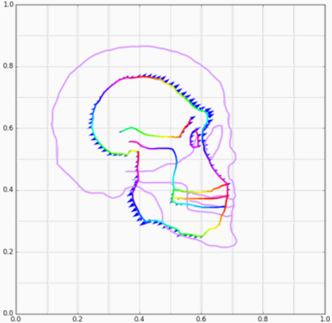


(a) Momentum p_0.

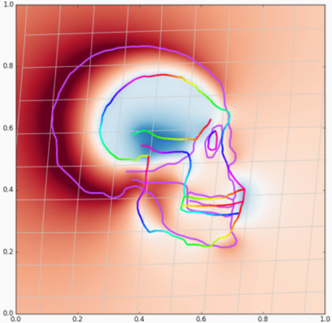
(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .98$.

Influence of the kernel width



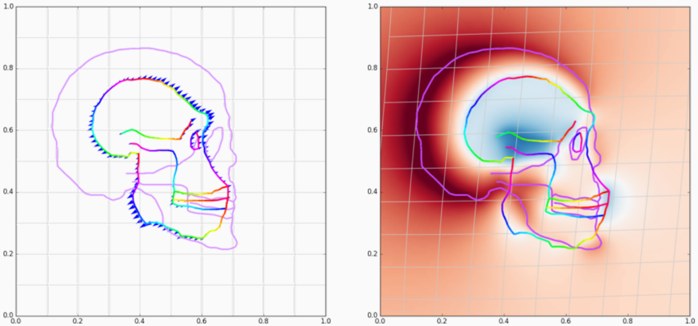
(a) Momentum p_0.



(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = .99$.

Influence of the kernel width



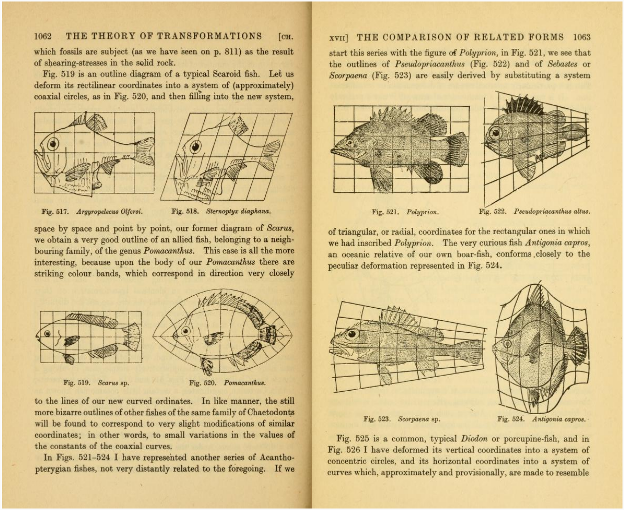
(a) Momentum p_0.

(b) Shooted model q_1.

Figure 28: Final matching, $\sigma = \mathtt{1.0}$.

Conclusion

We can now emulate D’Arcy Thompson’s work



1062 THE THEORY OF TRANSFORMATIONS [CH.

which fossils are subject (as we have seen on p. 811) as the result of shearing-stresses in the solid rock.

Fig. 519 is an outline diagram of a typical Scaroid fish. Let us deform its rectilinear coordinates into a system of (approximately) coaxial circles, as in Fig. 520, and then filling into the new system,

Fig. 517. *Argyropelecus Olfersi.* Fig. 518. *Sternoptyx diaphana.*

space by space and point by point, our former diagram of *Scarus*, we obtain a very good outline of an allied fish, belonging to a neighbouring family, of the genus *Pomacanthus*. This case is all the more interesting, because upon the body of our *Pomacanthus* there are striking colour bands, which correspond in direction very closely

Fig. 519. *Scarus sp.* Fig. 520. *Pomacanthus.*

to the lines of our new curved ordinates. In like manner, the still more bizarre outlines of other fishes of the same family of Chaetodonts will be found to correspond to very slight modifications of similar coordinates; in other words, to small variations in the values of the constants of the coaxial curves.

In Figs. 521–524 I have represented another series of Acanthopterygian fishes, not very distantly related to the foregoing. If we

XVII] THE COMPARISON OF RELATED FORMS 1063

start this series with the figure of *Polyprion*, in Fig. 521, we see that the outlines of *Pseudopriacanthus* (Fig. 522) and of *Sebastes* or *Scorpaena* (Fig. 523) are easily derived by substituting a system

Fig. 521. *Polyprion.* Fig. 522. *Pseudopriacanthus altus.*

of triangular, or radial, coordinates for the rectangular ones in which we had inscribed *Polyprion*. The very curious fish *Antigonia capros*, an oceanic relative of our own boar-fish, conforms closely to the peculiar deformation represented in Fig. 524.

Fig. 523. *Scorpaena sp.* Fig. 524. *Antigonia capros.*

Fig. 525 is a common, typical *Diodon* or porcupine-fish, and in Fig. 526 I have deformed its vertical coordinates into a system of concentric circles, and its horizontal coordinates into a system of curves which, approximately and provisionally, are made to resemble

Figure 29: Excerpt from the seminal book of D’Arcy Wentworth Thompson (1860-1948), *On Growth and Forms.*

Transfer of anatomical data

Figure 30: Video presentation of the (non-LDDMM) paper *Anatomy Transfer Fast Forward*, Siggraph Asia 2013 by Ali-Hamadi, Liu, Gilles et al.

Statistics on a Riemannian manifold

Biologists, Neurologists and Physicians would like to conduct statistical surveys such as :

- Linear regression
- Mean computation + Principal Component Analysis
- Transport of tangential information

Problem : no meaningful algebraic structure $(+, \times)$ on shapes.

Statistics on a Riemannian manifold

Biologists, Neurologists and Physicians would like to conduct statistical surveys such as :

- Linear regression
- Mean computation + Principal Component Analysis
- Transport of tangential information

Problem : no meaningful algebraic structure $(+, \times)$ on shapes.

Given a mere Riemannian distance, we provide :

- Geodesic regression
- Fréchet Mean + PCA on shooting momentums
- Parallel transport

Construction of anatomical atlases

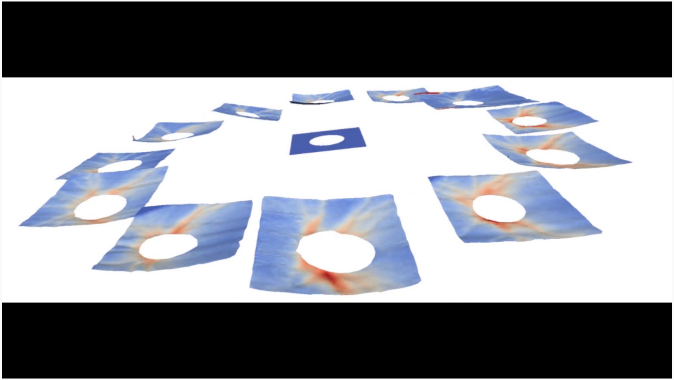


Figure 31: Taken from the personal web page of Benjamin Charlier.

A continuum of professions

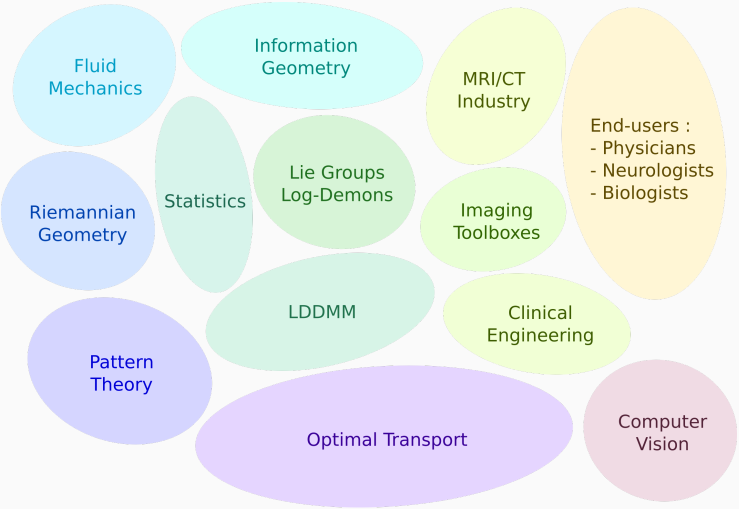


Figure 32: A (very) schematic view of the fields related to Computational Anatomy.

A continuum of professions

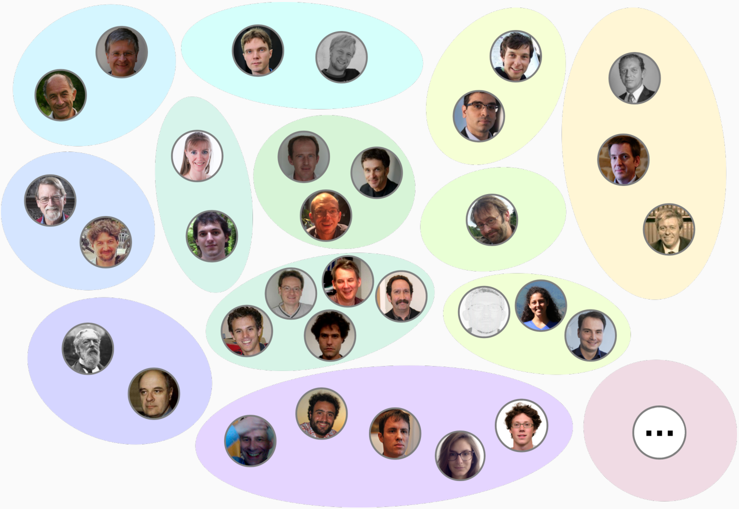
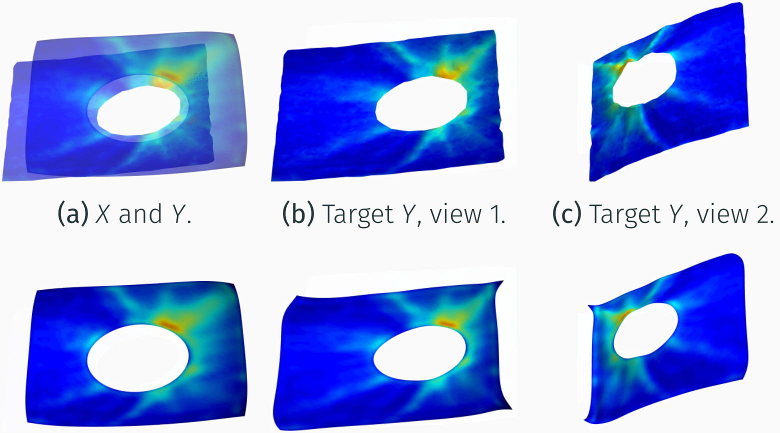


Figure 32: The people behind the labels.

Matchings of partially observed shapes



(a) X and Y. (b) Target Y, view 1. (c) Target Y, view 2.

(d) Source X. (e) $f(X)$, view 1. (f) $f(X)$, view 2.

Figure 33: Matching artifacts for the *retina* dataset.

Questions?

References I

P. Addis, P. Melis, R. Cannas, M. S. F. Tinti, C. Piccinetti, and A. Cau.
A morphometric approach for the analysis of body shape in bluefin tuna: preliminary results.
Collect. Vol. Sci. Pap. ICCAT, 65(3):982–987, 2010.

S. Clausen, K. Greiner, O. Andersen, K.-A. Lie, H. Schulerud, and T. Kavli.
Automatic segmentation of overlapping fish using shape priors.
In *Scandinavian conference on Image analysis*, pages 11–20. Springer, 2007.

References II

Y. Nikulin and R. Novak.
Exploring the neural algorithm of artistic style.
arXiv preprint arXiv:1602.07188, 2016.