Fast diffeomorphic matching to learn stable nonlinear dynamical systems

Nicolas Perrin[a,b], Philipp Schlehuber-Caissier[a],

[a] *Sorbonne Universités, UPMC University Paris 06, UMR 7222, ISIR, F-75005, Paris, France (e-mail:* {perrin, schlehuber}@isir.upmc.fr*).*
[b] *CNRS, UMR 7222, ISIR, F-75005, Paris, France.*

Abstract

We propose a new diffeomorphic matching algorithm and use it to learn nonlinear dynamical systems with the guarantee that the learned systems have global asymptotic stability. For a given set of demonstration trajectories, we compute a diffeomorphism that maps forward orbits of a reference stable time-invariant system onto the demonstrations, thereby deforming the whole reference system into one that reproduces the demonstrations, and is still stable.

Keywords:
Nonlinear dynamical systems, Diffeomorphic mapping, Imitation learning, Lyapunov stability, Dynamical movement primitives

1. Introduction

We consider the problem of learning dynamical systems (DS) from demonstrations. More precisely, given a list of trajectories $(\mathbf{x}_i(t))$ observed as timed sequences of points in $\mathbb{R}^d$, the objective is to build a smooth autonomous system $\dot{\mathbf{x}} = f(\mathbf{x})$ (i.e. a vector field) that reproduces the demonstrations as closely as possible.

The ability to construct such DS is an important skill in imitation learning (see for example [1]). The learned systems can be used as dynamical movement primitives generating goal-directed behaviors [2].

Modeling motion primitives with DS is convenient for closed loop implementations, and their generalization to unseen parts of the state space provides robustness to spatial perturbations. Moreover, the choice of autonomous (i.e. time-invariant) systems, while not always suitable or preferable, is interesting in many situations as they are inherently robust to temporal perturbations.

The most common motion primitives consist of point-to-point motions, i.e. movements in space that stop at a given target. They correspond to globally asymptotically stable DS. But classical learning algorithms cannot provide the guarantee that their output is always stable. They might produce DS with instabilities or spurious attractors. This issue has recently been studied by Khansari-Zadeh and Billard [3, 4] who proposed several approaches to learn stable nonlinear DS. One of the main ideas they investigated consists in learning a Lyapunov function candidate[1] L that is highly compatible with the demonstrations in the following sense: at almost every point $\mathbf{x}_i(t_j)$, the estimated or measured velocity $\mathbf{v}_i(t_j)$ is such that its scalar product with the gradient of L is negative: $\mathbf{v}_i(t_j) \cdot \nabla L(\mathbf{x}_i(t_j)) < 0$. Once L is found, a learning algorithm optimizes a weighted sum of DS that admit L as a common Lyapunov function, therefore ensuring the global stability of the resulting DS. Alternatively, L can be used to modify motion primitives by correcting trajectories whenever they would violate the compatibility condition.

The main limitation of this method comes from the difficulty to find good Lyapunov candidates. In SEDS (*Stable Estimator of Dynamical Systems*), one of the first algorithms proposed by Khansari-Zadeh and Billard, the Lyapunov function is set to be the l^2-norm squared ($\|\cdot\|^2$), which means that all trajectories produced by the learned DS can only monotonically decrease in norm. In their more recent algorithm CLF-DM (*Control Lyapunov Function-based Dynamic Movements*), the search for a Lyapunov function candidate is done among a set called Weighted Sums of Asymmetric Quadratic Functions (WSAQF). It highly increases the set of DS that can be learned, but the restrictions remain significant (for instance, no WSAQF can take the same value on two distinct points $\mathbf{x}$ and $\lambda\mathbf{x}$ with $\lambda > 0$). Neumann et al. [5] compute a "Neurally Imprinted Lyapunov Candidate" (NILC) via quadratic programming, obtaining similar restrictions to that of WSAQF. In short, a way to efficiently find Lyapunov candidates is to fix a simple stable DS with which they must be compatible ($\dot{\mathbf{x}} = -\mathbf{x}$ in the case of WSAQF and NILC). An issue with this approach is that it necessarily restricts the search to a small convex subset of the whole set of Lyapunov candidates.

To go further, Neumann and Steil [6] suggested to initially compute a Lyapunov candidate with one of the above methods, and then apply a simple diffeomorphism (of the form $\mathbf{x} \mapsto \eta(\mathbf{x})\mathbf{x}$, with $\eta(\mathbf{x}) \in \mathbb{R}_{\geq 0}$) that deforms the space and transforms the Lyapunov candidate into the function $\mathbf{x} \mapsto \|\mathbf{x}\|^2$, thus simplifying the trajectories of the demonstrations. In the deformed space, an algorithm like SEDS is then more likely to learn a stable DS that reproduces faithfully the demonstrations.

In this paper, we propose a more direct diffeomorphism-based approach. Our contribution is twofold.

[1] In this paper, a Lyapunov function candidate (or simply Lyapunov candidate) is a continuously differentiable function from $\mathbb{R}^d$ to $\mathbb{R}_{\geq 0}$ taking the value 0 at the target point $\mathbf{0}$ and with no other local minimum.

- First, we introduce a new algorithm for diffeomorphic matching (Sections 2 and 3) and show from experimental comparisons that it tends to be one or two orders of magnitude faster than one of the state-of-the-art algorithms.
- Then, we explain how it can be used to directly map simple trajectories of a DS like $\dot{\mathbf{x}} = -\mathbf{x}$ onto the trajectories of the training data (Section 4). This gives a new way to generate Lyapunov candidates as well as globally asymptotically stable smooth autonomous systems reproducing the demonstrations.

The most direct applications of this work are in motor control and robotics, but we believe that learning stable nonlinear DS and computing Lyapunov candidates can be useful for various types of systems and control design problems.

2. Diffeomorphic locally weighted translations

Given a smooth (symmetric positive definite) kernel function $k_\rho(\mathbf{x}, \mathbf{y})$, depending on some parameter ρ, such that $\forall \mathbf{x}, k_\rho(\mathbf{x}, \mathbf{x}) = 1$ and $k_\rho(\mathbf{x}, \mathbf{y}) \to 0$ when $\|\mathbf{y} - \mathbf{x}\| \to \infty$, given a "direction" $\mathbf{v} \in \mathbb{R}^d$ and a "center" $\mathbf{c} \in \mathbb{R}^d$, we consider the following *locally weighted translation*:

$$\psi_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{x}) = \mathbf{x} + k_\rho(\mathbf{x}, \mathbf{c})\mathbf{v}.$$

Theorem 1. *If* $\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^d \times \mathbb{R}^d$, $\frac{\partial k_\rho}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{v} > -1$, *then* $\psi_{\rho,\mathbf{c},\mathbf{v}}$ *is a diffeomorphism.*

Proof. For $\mathbf{x} \in \mathbb{R}^d$, let us define:

$$h_{\mathbf{x}} : r \in \mathbb{R} \mapsto r + k_\rho(\mathbf{x} + r\mathbf{v}, \mathbf{c}) \in \mathbb{R}.$$

If $\frac{\partial k_\rho}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{v} > -1$, we get: $\forall r \in \mathbb{R}$, $\frac{dh_{\mathbf{x}}}{dr}(r) > 0$. Since $h_{\mathbf{x}}(r)$ tends to $-\infty$ when r tends to $-\infty$, and to $+\infty$ when r tends to $+\infty$, we deduce that there exists a unique $s(\mathbf{x}) \in \mathbb{R}$ such that $h_{\mathbf{x}}(s(\mathbf{x})) = 0$. Moreover, $\psi_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{x} + r\mathbf{v}) = \mathbf{x} + h_{\mathbf{x}}(r)\mathbf{v}$, thus $\psi_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{x} + r\mathbf{v}) = \mathbf{x}$ if and only if $r = s(\mathbf{x})$ or $\mathbf{v} = \mathbf{0}$. Besides, any $\mathbf{y} \in \mathbb{R}^d$ verifying $\psi_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{y}) = \mathbf{x}$ is necessarily of the form $\mathbf{x} + r\mathbf{v}$, so $\psi_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{y}) = \mathbf{x}$ implies $\mathbf{y} = \mathbf{x} + s(\mathbf{x})\mathbf{v}$.

We conclude that $\psi_{\rho,\mathbf{c},\mathbf{v}}$ is invertible, and:

$$\psi^{-1}_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{x}) = \mathbf{x} + s(\mathbf{x})\mathbf{v}.$$

Finally, the inverse function theorem can be applied to prove that $\psi_{\rho,\mathbf{c},\mathbf{v}}$ is a diffeomorphism. □

With Gaussian Radial Basis Function (RBF) kernel:

We now consider the following kernel (with $\rho \in \mathbb{R}_{>0}$):

$$k_\rho(\mathbf{x}, \mathbf{y}) = \exp\left(-\rho^2 \|\mathbf{x} - \mathbf{y}\|^2\right).$$

We have:

$$\frac{\partial k_\rho}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{v} = -2\rho^2 \exp\left(-\rho^2 \|\mathbf{x} - \mathbf{y}\|^2\right)(\mathbf{x} - \mathbf{y}) \cdot \mathbf{v},$$

with the lower bound:

$$\frac{\partial k_\rho}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{v} \geq -2\rho^2 \exp\left(-\rho^2 \|\mathbf{x} - \mathbf{y}\|^2\right) \|\mathbf{x} - \mathbf{y}\| . \|\mathbf{v}\|.$$

The expression on the right takes its minimum for $\|\mathbf{x}-\mathbf{y}\| = \frac{1}{\sqrt{2}\rho}$, which yields:

$$\frac{\partial k_\rho}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{v} \geq -\sqrt{2}\|\mathbf{v}\|\rho \exp\left(-\frac{1}{2}\right).$$

Applying Theorem 1, $\mathbf{v} = 0$ or $\rho < \rho_{\max}(\mathbf{v}) = \frac{1}{\sqrt{2}\|\mathbf{v}\|} \exp\left(\frac{1}{2}\right)$ implies that $\psi_{\rho,\mathbf{c},\mathbf{v}}$ is a diffeomorphism.

In that case, $s(\mathbf{x})$, and as a result $\psi^{-1}_{\rho,\mathbf{c},\mathbf{v}}(\mathbf{x})$, can be very efficiently computed with Newton's method.

3. A diffeomorphic matching algorithm

In this section we are interested in the following problem: given two sequences of distinct points $\mathbf{X} = (\mathbf{x}_i)_{i\in\{0,\dots,N\}}$ and $\mathbf{Y} = (\mathbf{y}_i)_{i\in\{0,\dots,N\}}$, compute a diffeomorphism Φ that maps each $\mathbf{x}_i$ onto $\mathbf{y}_i$, either exactly or approximately. More formally, defining $\mathsf{dist}(\mathbf{A}, \mathbf{B}) = \frac{1}{N+1}\sum_i \|\mathbf{a}_i - \mathbf{b}_i\|^2$ for two sequences $\mathbf{A}$ and $\mathbf{B}$ of $N + 1$ points, and denoting by $\Phi(\mathbf{X})$ the sequence of points $(\Phi(\mathbf{x}_i))_{i\in\{0,\dots,N\}}$, we want to find a diffeomorphism Φ that minimizes $\mathsf{dist}(\Phi(\mathbf{X}), \mathbf{Y})$.

Since the sequences $\mathbf{X}$ and $\mathbf{Y}$ can be very different in shape, to the best of our knowledge the state-of-the-art existing techniques to solve this problem are based on the Large Deformation Diffeomorphic Metric Mapping (LDDMM) framework introduced in the seminal article by Joshi and Miller [7]. Its core idea is to work with a time dependent vector field $v(\mathbf{x}, t) \in \mathbb{R}^d$ ($t \in [0, 1]$), and define a flow $\phi(\mathbf{x}, t)$ via the transport equation:

$$\frac{d\phi(\mathbf{x}, t)}{dt} = v(\phi(\mathbf{x}, t), t),$$

with $\phi(\mathbf{x}, 0) = \mathbf{x}$. With a few regularity conditions on v (see [8] for specific requirements), $\mathbf{x} \mapsto \phi(\mathbf{x}, t)$ is a diffeomorphism. The resulting diffeomorphism $\Phi(\mathbf{x}) = \phi(\mathbf{x}, 1)$ is given by:

$$\Phi(\mathbf{x}) = \mathbf{x} + \int_0^1 v(\phi(\mathbf{x}, t), t)dt.$$

Using an appropriate Hilbert space for the vector fields $\mathbf{x} \mapsto v(\mathbf{x}, t)$, they can be associated with an infinitesimal cost whose integration is interpreted as a deformation energy.

Various gradient descent algorithms have been proposed to optimize v with respect to a cost that depends both on the deformation energy and on the accuracy of the mapping, whether the objective is to map curves [9], surfaces [10], or, as in our case, points [11].

The LDDMM framework has several advantages. For example, it tries to minimize the deformation, and allows the computation of similarity measures between diffeomorphic geometrical objects. However, Φ is not in closed-form, so once obtained, evaluating it requires an integration that can be slightly time-costly. In our context, Φ can be used inside a control law, so its evaluation (and that of Φ^{-1}) needs to be very fast.

We propose a completely different approach to diffeomorphic matching, based on the diffeomorphic locally weighted translations presented in the previous section, which are functions that can be evaluated extremely quickly. The proposed algorithm is based on simple principles, yet it does not seem to have been studied in the past.

3.1. The algorithm

We fix a number of iterations K, and two parameters $0 < \mu < 1$ and $0 < \beta \leq 1$. Typically, on the examples presented in this paper, we use $K = 150$, $\mu \approx 0.9$ and $\beta \approx 0.5$.

Initially, we define $\mathbf{Z} := \mathbf{X}$. Every iteration j updates $\mathbf{Z}$, and can be briefly described by the three following steps:

1. we select the point $\mathbf{p}_j$ in $\mathbf{Z}$ that is the furthest from its corresponding target $\mathbf{q}$ in $\mathbf{Y}$;
2. we consider the locally weighted translation $\psi_{\rho_j,\mathbf{p}_j,\mathbf{v}_j}$ of direction $\mathbf{v}_j = \beta(\mathbf{q}-\mathbf{p}_j)$, center $\mathbf{p}_j$, and Gaussian RBF kernel k_{ρ_j}, optimizing $\rho_j \in [0, \mu\rho_{\max}(\mathbf{v}_j)]$ to minimize the error between $\psi_{\rho_j,\mathbf{p}_j,\mathbf{v}_j}(\mathbf{Z})$ and $\mathbf{Y}$;
3. we perform the update: $\mathbf{Z} := \psi_{\rho_j,\mathbf{p}_j,\mathbf{v}_j}(\mathbf{Z})$.

The resulting diffeomorphism is:

$$\Phi = \psi_{\rho_K,\mathbf{p}_K,\mathbf{v}_K} \circ \cdots \circ \psi_{\rho_2,\mathbf{p}_2,\mathbf{v}_2} \circ \psi_{\rho_1,\mathbf{p}_1,\mathbf{v}_1}$$

Here is a description of the algorithm in pseudo-code:

1: **Input:** $\mathbf{X} = (\mathbf{x}_i)_{i\in\{0,\dots,N\}}$ and $\mathbf{Y} = (\mathbf{y}_i)_{i\in\{0,\dots,N\}}$
2: **Parameters:** $K \in \mathbb{N}_{>0}$, $0 < \mu < 1$, $0 < \beta \leq 1$
3:
4: $\mathbf{Z} = (\mathbf{z}_i)_{i\in\{0,\dots,N\}}$
5: $\mathbf{Z} := \mathbf{X}$
6: **for** $j = 1$ to K **do**
7: $\quad m := \underset{i\in\{0,\dots,N\}}{\arg\max} (\|\mathbf{z}_i - \mathbf{y}_i\|)$
8: $\quad \mathbf{p}_j := \mathbf{z}_m$
9: $\quad \mathbf{q} := \mathbf{y}_m$
10: $\quad \mathbf{v}_j := \beta(\mathbf{q} - \mathbf{p}_j)$
11: $\quad \rho_j := \underset{\rho\in[0,\mu\rho_{\max}(\mathbf{v}_j)]}{\arg\min} (\mathsf{dist}(\psi_{\rho,\mathbf{p}_j,\mathbf{v}_j}(\mathbf{Z}), \mathbf{Y}))$
12: $\quad \mathbf{Z} := \psi_{\rho_j,\mathbf{p}_j,\mathbf{v}_j}(\mathbf{Z})$
13: **end for**
14: **return** $(\rho_j)_{j\in\{1,\dots,K\}}$, $(\mathbf{p}_j)_{j\in\{1,\dots,K\}}$, $(\mathbf{v}_j)_{j\in\{1,\dots,K\}}$

Remarks:

- Here we have presented a version of the algorithm in which the parameter β is constant, but we can also make it vary iteration after iteration, for example by slowly increasing towards 1.
- The line 11 of the algorithm performs a nonlinear optimization, but since it depends only on one bounded real variable, a minimum can be found very quickly and precisely.
- We can add a fixed upper bound $\rho_M > 0$ for all ρ_j, and a regularization term in the cost of the optimization problem of line 11, to prevent the diffeomorphism from overly deforming the space to get a perfect matching. Simply using sequences with a large number of points has a similar effect (and it almost does not slow the algorithm down).
- Again in line 11, dist can be replaced by any other distance. In practice, writing $\mathbf{X}$ and $\mathbf{Y}$ as $(N+1)$-by-2 matrices and taking the largest singular value norm of $(\mathbf{X}-\mathbf{Y})$ seems to give slightly better results than with the Euclidean distance.

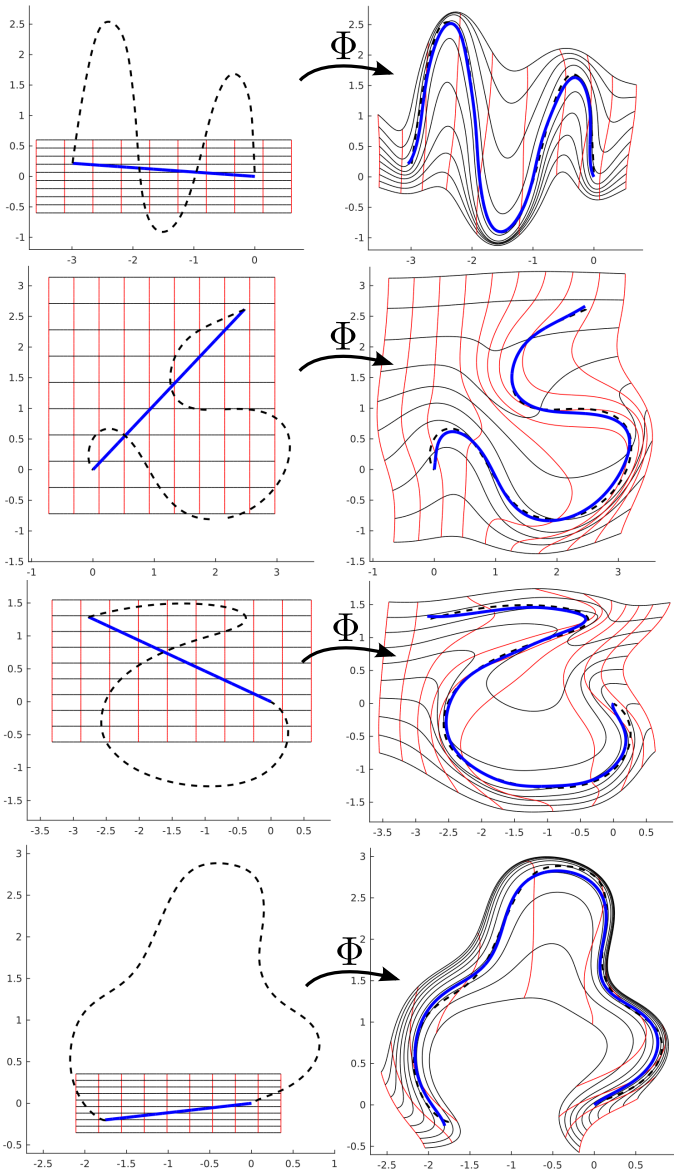

Figure 1: On the left, the dashed curve is a trajectory represented by a sequence of points $\mathbf{Y} = (\mathbf{y}_i)_{i\in\{0,\dots,N\}}$. The solid line is $\mathbf{X} = (\mathbf{y}_0 + \frac{i}{N}(\mathbf{y}_N - \mathbf{y}_0))_{i\in\{0,\dots,N\}}$. The right side shows the result of the application of the diffeomorphism Φ constructed by our algorithm to map $\mathbf{X}$ onto $\mathbf{Y}$.

- Nothing prevents the algorithm from getting stuck in a local minimum, so a general proof of convergence cannot be found. However, as we show in the next sections, experimental results give empirical evidence that the algorithm is efficient and converges quickly in practice, even on difficult matching problems. In future work, we will try to further improve the algorithm and get convergence proofs under realistic assumptions.

3.2. Experimental evaluation

We compare our algorithm to an implementation of diffeomorphic matching in the LDDMM framework developed by J. Glaunès (the "`Matchine`" software [12]).

	N	our algorithm	LDDMM
Learning: average duration of the construction of Φ	20	0.25 s	2.78 s
	50	0.25 s	14.5 s
	100	0.26 s	53.3 s
Forward evaluation: average duration of the computation of $\Phi(\mathbf{X})$	20	3.05 ms	157 ms
	50	3.35 ms	804 ms
	100	3.72 ms	3130 ms
Backward evaluation: average duration of the computation of $\Phi^{-1}(\mathbf{Y})$	20	29.8 ms	145 ms
	50	35 ms	798 ms
	100	38.5 ms	3110 ms
Accuracy: average value of $\mathsf{dist}(\Phi(\mathbf{X}), \mathbf{Y})$	20	3.49×10^{-3}	18.2×10^{-3}
	50	8.32×10^{-3}	22.2×10^{-3}
	100	9.51×10^{-3}	22×10^{-3}

Table 1: Comparison of experimental results

Given a sequence of points $\mathbf{Y} = (\mathbf{y}_i)_{i\in\{0,\dots,N\}}$ representing a trajectory, we set $\mathbf{X} = (\mathbf{x}_i)_{i\in\{0,\dots,N\}} = \left(\mathbf{y}_0 + \frac{i}{N}(\mathbf{y}_N - \mathbf{y}_0)\right)_{i\in\{0,\dots,N\}}$ and apply our algorithm or the LDDMM algorithm to construct a diffeomorphism Φ such that $\Phi(\mathbf{X})$ and $\mathbf{Y}$ match. Figure 1 displays the result of our algorithm on four 2D trajectories, and Table 1 shows a comparison of the results obtained on these trajectories with our algorithm and the LDDMM algorithm. For each trajectory, we try with representations as sequences of 21, 51 and 101 points (i.e. $N = 20$, $N = 50$, $N = 100$). For both algorithms, the same parameters are kept across all the trials.

In all cases, our algorithm provides a substantial speedup. For example, with 51 points, Φ is learned in average 58 times faster and evaluated 240 times faster, while the error $\mathsf{dist}(\Phi(\mathbf{X}), \mathbf{Y})$ is 2.67 times smaller. The tests were made on an Intel(R) Core(TM) i7-4700MQ @ 2.4 GHz with 4GB of RAM.

4. Learning stable nonlinear dynamical systems

4.1. Overview of the method

Two autonomous systems $\dot{\mathbf{x}} = f(\mathbf{x})$ and $\dot{\mathbf{x}} = g(\mathbf{x})$ are said diffeomorphic, or smoothly equivalent, if there exists a diffeomorphism $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ such that:

$$g(\Phi(\mathbf{x})) = J_\Phi(\mathbf{x}) f(\mathbf{x}),$$

where $J_\Phi(x)$ is the Jacobian matrix: $J_\Phi(\mathbf{x}) = \frac{\partial \Phi}{\partial \mathbf{x}}(\mathbf{x})$. When this equation is verified, Φ maps the orbits of the DS $\dot{\mathbf{x}} = f(\mathbf{x})$ onto the orbits of the DS $\dot{\mathbf{x}} = g(\mathbf{x})$. In particular, if one of the two diffeomorphic DS is globally asymptotically stable, then both are. And if L is a Lyapunov function for the DS $\dot{\mathbf{x}} = f(\mathbf{x})$, then $L \circ \Phi^{-1}$ is a Lyapunov function for the DS $\dot{\mathbf{x}} = g(\mathbf{x})$.

The objective of our approach is to learn a diffeomorphism Φ that maps forward orbits of the DS $\dot{\mathbf{x}} = -\mathbf{x}$ (i.e. line segments) onto the observed trajectories. As a consequence, $\mathbf{x} \mapsto \|\Phi^{-1}(\mathbf{x})\|$ will be a Lyapunov candidate. Additionally, it will be possible to directly get a stable DS of the form $\dot{\mathbf{x}} = -\gamma(\Phi^{-1}(\mathbf{x}))J_\Phi(\Phi^{-1}(\mathbf{x}))\Phi^{-1}(\mathbf{x})$, with $\gamma : \mathbb{R}^d \to \mathbb{R}_{>0}$, that reproduces approximately the demonstrations. Remark: this system can be written $\mathbf{x} = \Phi(\mathbf{z})$, with $\dot{\mathbf{z}} = -\gamma(\mathbf{z})\mathbf{z}$ (the DS $\dot{\mathbf{z}} = -\gamma(\mathbf{z})\mathbf{z}$ and $\dot{\mathbf{z}} = -\mathbf{z}$ have the same orbits).

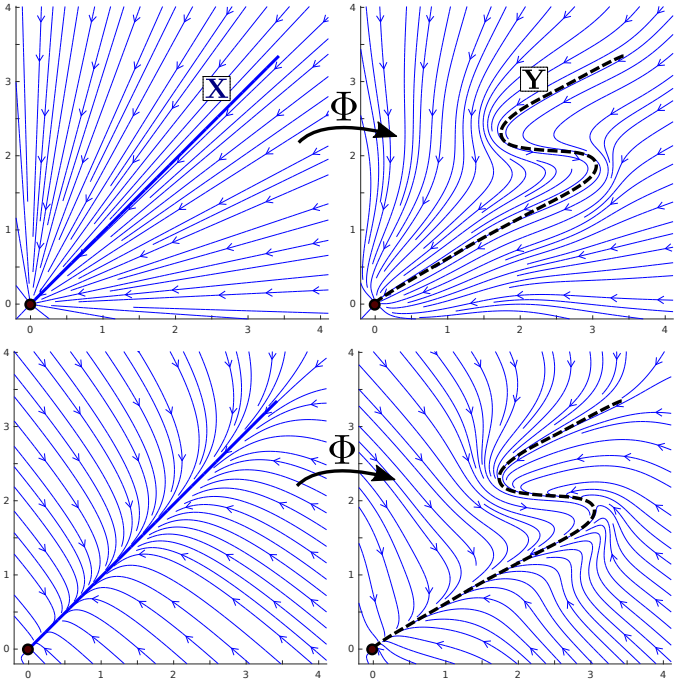

Figure 2: The diffeomorphism Φ, that maps the trajectory $\mathbf{X}$ onto $\mathbf{Y}$, transforms a stable DS with $\mathbf{X}$ as a forward orbit into a stable DS with $\mathbf{Y}$ as a forward orbit (*top row*). Using this property, we can modulate the initial vector field while keeping $\mathbf{X}$ unchanged to obtain systems with different behaviors that all reproduce the demonstration $\mathbf{Y}$.

Trajectories being represented as sequences of points, this problem of orbits mapping can be cast as diffeomorphic matching. In the case of a unique demonstration $\mathbf{Y} = (\mathbf{y}(t_i))_{i\in\{0,\dots,N\}}$, with $t_i = i\Delta t$, we want to find a diffeomorphism that maps $\mathbf{X} = \left(\mathbf{y}(0) + \frac{i}{N}(\mathbf{0} - \mathbf{y}(0))\right)_{i\in\{0,\dots,N\}}$ onto $\mathbf{Y}$ (the trajectory is assumed to arrive at the target: $\mathbf{y}(t_N) = \mathbf{0}$). To do so, we simply use the algorithm presented in the previous section. The diffeomorphism Φ_K obtained after K iterations can be such that $\Phi_K(\mathbf{0}) \neq \mathbf{0}$, so we add an extra iteration that picks $\mathbf{p}_{K+1} = \Phi_K(\mathbf{0})$ and $\mathbf{v}_{K+1} = \mathbf{0} - \Phi_K(\mathbf{0})$. This ensures that the final diffeomorphism Φ verifies $\Phi_K(\mathbf{0}) = \mathbf{0}$. Remark: the structure of Φ makes it easy to efficiently compute $J_\Phi(\mathbf{x})$ at any given point.

4.2. Results

The top row of Figure 2 shows the result of mapping the straight trajectory $\mathbf{X}$ (on the left) onto the trajectory $\mathbf{Y}$ (on the right). The diffeomorphism Φ that realizes this matching also transforms the entire DS $\dot{\mathbf{x}} = -\mathbf{x}$ into a nonlinear stable DS that reproduces the trajectory $\mathbf{Y}$ (as the forward orbit of $\mathbf{y}(0)$).

Modifying the initial DS without changing the forward orbit of $\mathbf{x}(0)$ leads, by application of Φ, to another DS that still reproduces $\mathbf{Y}$. On the bottom row of Figure 2, we use a linear system that keeps $\mathbf{x}(0)$ as an eigenvector associated with eigenvalue -1 (ensuring that its orbit is not modified), but has an eigenvalue smaller than -1 in the orthogonal direction (unlike the DS $\dot{\mathbf{x}} = -\mathbf{x}$). This results in a transformed DS that "tracks" more agressively the trajectory $\mathbf{Y}$.

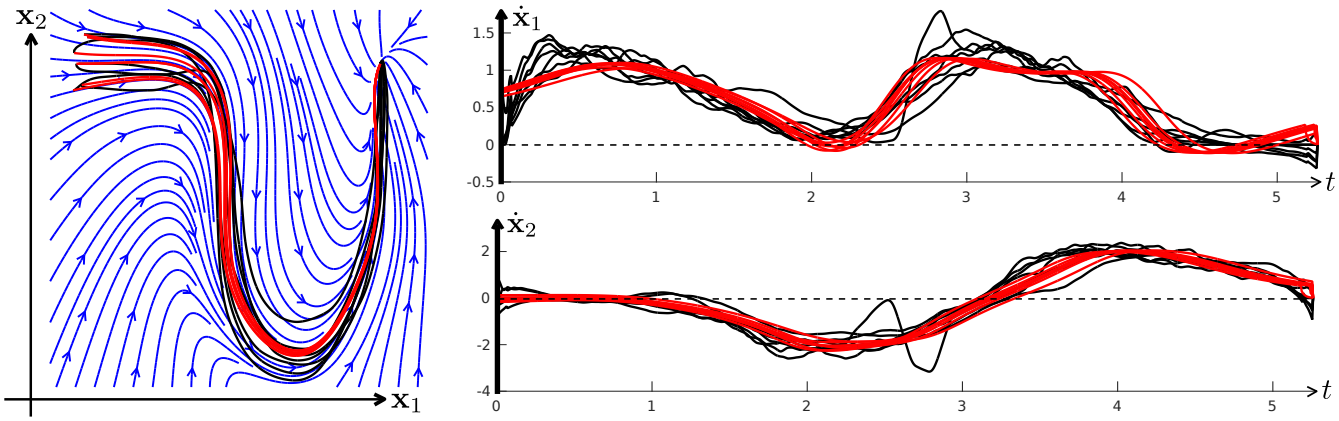

Figure 3: On the left: a smooth autonomous systems, learned with our method, that reproduces a motion pattern (demonstrations are in black, reproduced trajectories in red). The trajectories on the right show that the velocity profiles are quite accurately reproduced as well (again, demonstrations in black and reproductions in red).

We evaluated our approach on the LASA Handwriting Dataset, similarly to [3, 4, 6]. On all cases shown in Figure 4, a set of 7 trajectories ending at the same point demonstrate a single pattern of handwriting motion. For each of these patterns, we create an average timed sequence of points $\mathbf{Y} = (\mathbf{y}(i\Delta t))_{i\in\{0,\dots,N\}}$ based on the 7 trajectories, and apply our matching algorithm to construct a diffeomorphism Φ that maps $\mathbf{X} = (\frac{N-i}{N}\mathbf{y}(0))_{i\in\{0,\dots,N\}}$ onto $\mathbf{Y}$. This gives a Lyapunov candidate $\mathbf{x} \mapsto \|\Phi^{-1}(\mathbf{x})\|$. We compare it to the optimized WSAQF Lyapunov candidates obtained with the method of Khansari-Zadeh and Billard [4] also used in [6]. On the *1st column* are displayed level sets of the WSAQF Lyapunov candidates, and on the *2nd column* level sets of our Lyapunov candidates. We can observe that the level sets of the Lyapunov candidates produced by our method have a richer geometry and exhibit variations that are more suitably adapted to the training data.

Of course, following the method of Khansari-Zadeh and Billard [4], these Lyapunov candidates can be used to correct any learned DS to ensure global asymptotic stability. But as mentioned above, the diffeomorphism also provides a way to directly get a stable DS that reproduces the motion pattern. We define a function $\gamma : \mathbb{R}^d \to \mathbb{R}_{>0}$ such that, starting at $\mathbf{x}(0) = \mathbf{y}(0)$ with $t = 0$, the DS $\dot{\mathbf{x}} = -\gamma(\mathbf{x})\mathbf{x}$ produces a trajectory that passes by the points $\frac{N-i}{N}\mathbf{y}(0)$ at times $i\Delta t$, for $i \in \{1, \dots, N-1\}$, and converges asymptotically towards $\mathbf{0}$ for $t > (N-1)\Delta t$. A simple choice for γ is $\gamma(\mathbf{x}) = \frac{\|\mathbf{y}(0)\|}{N\Delta t\|\mathbf{x}\|}$ for $\|x\| \geq \frac{\|\mathbf{y}(0)\|}{N}$ and $\gamma(\mathbf{x}) = \frac{\|\mathbf{y}(0)\|}{N}$ otherwise (but it is easy to design a smoother function with the same desired properties).

Φ transforms the DS $\dot{\mathbf{x}} = -\gamma(\mathbf{x})\mathbf{x}$ into one that reproduces the demonstrations and their velocity profiles, as shown in Figure 3. The eigenvalue in the direction orthogonal to $\mathbf{y}(0)$ can be adjusted according to the variability of the 7 demonstrations. The vector fields we obtained are shown on the *4th column* of Figure 4, and the vector fields obtained with the τ-SEDS (WSAQF) method of Neumann and Steil [6] are shown on the *3rd column*.

5. Conclusion

In this paper, we presented a new algorithm for diffeomorphic matching and a way to use it to learn stable nonlinear autonomous dynamical systems from demonstrations.

While the demonstrations were 2D single motion patterns in the results we presented, our algorithm scales well to higher dimensions (because all its parameters are dimension-independent) and can handle multiple motion patterns, although in some cases topological issues may prevent the convergence of the matching. It should also be noted that we can only produce vector fields that are diffeomorphic to the DS $\dot{\mathbf{x}} = -\mathbf{x}$, which is not true for all globally asymptotically stable smooth autonomous systems. A related limitation concerns 2D spiral trajectories, which cannot be reproduced. In future work, we will try to combine our approach with existing methods to extend its possibilities. For instance, Kronander et al. [13] suggest to iteratively reshape DS by locally applying full-rank modulations such as scalings and rotations. It does not guarantee the global asymptotic stability of the resulting DS, but we see in this work an approach that could be complementary to ours.

The main advantages of our method are:

1. *speed*: unlike [4] and [6], we do not need a second learning phase once the Lyapunov candidate has been found, and we do not rely on numerical optimization of parameters whose number rapidly increases with the dimensionality; instead, the simple iterative algorithm we use has a constant number of parameters, and we have empirically verified its quick convergence for many difficult matching problems.
2. *simplicity*: of implementation because the algorithm is very short, but also of use thanks to the small number of parameters to adjust.

For these reasons, we believe it can be applied with ease to efficiently learn a large variety of stable autonomous systems, with applications in dynamic movement primitives construction or more generally in control design. If instead of $\dot{\mathbf{x}} = -\mathbf{x}$, we start with an initial DS that has a stable limit cycle, our approach can be adapted to learn limit cycle systems.

Finally, being significantly faster than a state-of-the-art algorithm, our diffeomorphic matching algorithm itself might be of interest for completely different applications, such as for example image registration.

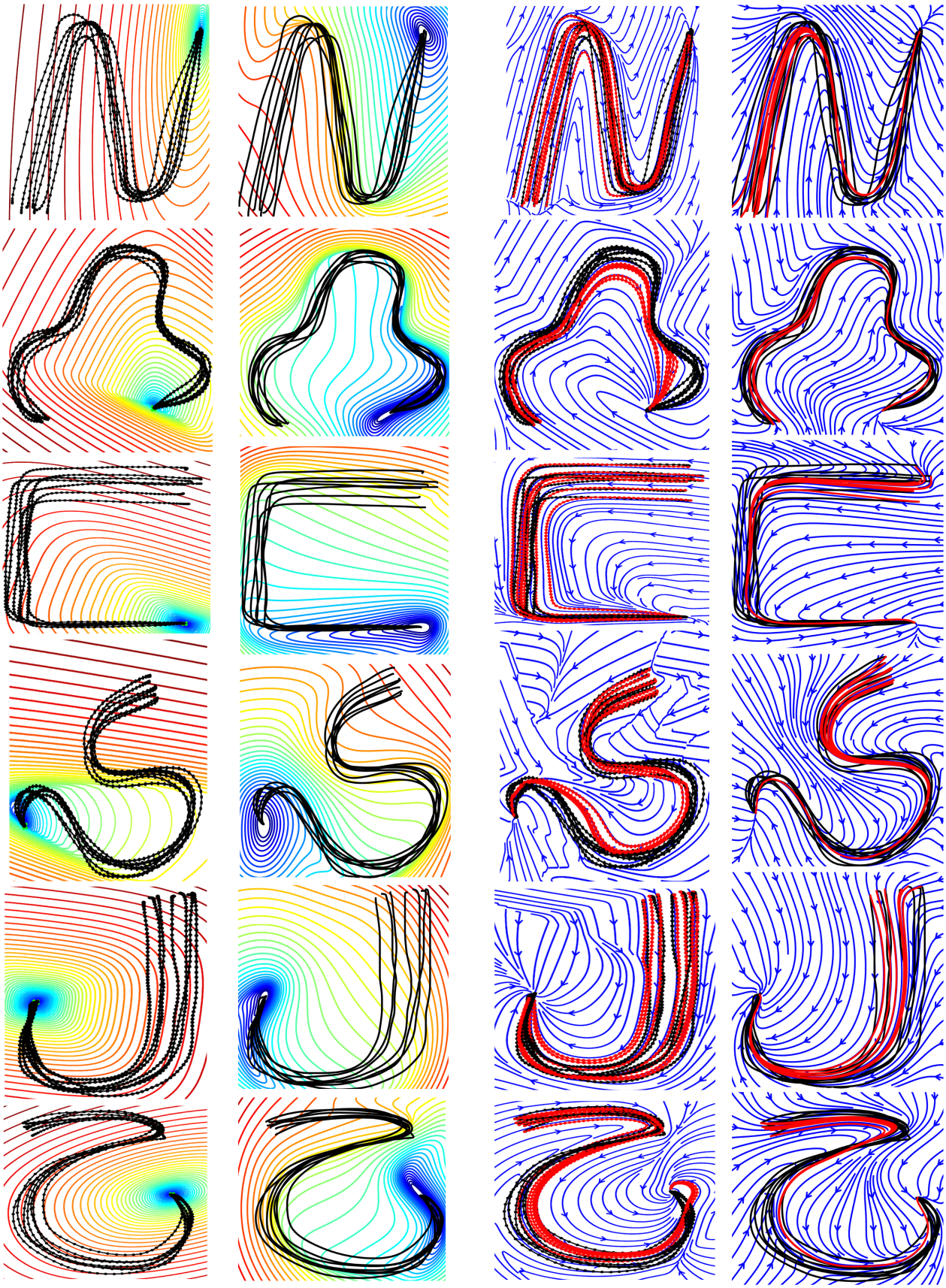

Figure 4: • *1st column*: level sets of the WSAQF Lyapunov candidates [4, 6]. • *2nd column*: level sets of the Lyapunov candidates obtained with our approach. • *3rd column*: streamlines of the stable DS produced by the τ-SEDS (WSAQF) approach [6]. • *4th column*: streamlines of the stable DS produced by our approach. • *All columns*: the demonstrations are displayed in black. • *3rd & 4th columns*: the reproduced trajectories are in red.

References

[1] S. Schaal, A. Ijspeert, A. Billard, Computational approaches to motor learning by imitation, Philosophical Transactions of the Royal Society B: Biological Sciences 358 (1431) (2003) 537–547.

[2] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural computation 25 (2) (2013) 328–373.

[3] S. M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with gaussian mixture models, IEEE Transactions on Robotics 27 (5) (2011) 943–957.

[4] S. M. Khansari-Zadeh, A. Billard, Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions, Robotics and Autonomous Systems 62 (6) (2014) 752–765.

[5] K. Neumann, A. Lemme, J. J. Steil, Neural learning of stable dynamical systems based on data-driven lyapunov candidates, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 1216–1222.

[6] K. Neumann, J. J. Steil, Learning robot motions with stable dynamical systems under diffeomorphic transformations, Robotics and Autonomous Systems 70 (2015) 1–15.

[7] S. C. Joshi, M. Miller, et al., Landmark matching via large deformation diffeomorphisms, IEEE Transactions on Image Processing 9 (8) (2000) 1357–1370.

[8] P. Dupuis, U. Grenander, M. I. Miller, Variational problems on flows of diffeomorphisms for image matching, Quarterly of applied mathematics 56 (3) (1998) 587–600.

[9] J. Glaunès, A. Qiu, M. I. Miller, L. Younes, Large deformation diffeomorphic metric curve mapping., Int J Comput Vis 80 (3) (2008) 317–336.

[10] M. Vaillant, J. Glaunès, Surface matching via currents, in: Information Processing in Medical Imaging, 2005, pp. 381–392.

[11] H. Guo, A. Rangarajan, S. Joshi, Diffeomorphic point matching, in: Handbook of Mathematical Models in Computer Vision, Springer, 2006, pp. 205–219.

[12] "Matchine" software by J. A. Glaunès, Copyright (C) Université Paris Descartes, http://www.mi.parisdescartes.fr/~glaunes/matchine.zip.

[13] K. Kronander, M. Khansari, A. Billard, Incremental motion learning with locally modulated dynamical systems, Robotics and Autonomous Systems 70 (C) (2015) 52–62.