Appendix A. Converging dynamics

A problem often encountered when using statistical models to learn the dynamics $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ based on given demonstrations is what we term *drift* error and is caused by integrating small but consistent errors in the estimated velocities when comparing with the velocities given in the demonstrations. Enforcing global asymptotic stability based on, for instance, Lyapunov theory does not solve this problem. Enforcing the Lyapunov function $V(\boldsymbol{x})$ using the standard constraint

$$\nabla_{\boldsymbol{x}} V(\boldsymbol{x}).f(\boldsymbol{x}) < 0. \tag{1}$$

does not allow to make any non-trivial statements about the resulting trajectories for initial positions *close* to each other or about the distance between the demonstration and the resulting trajectory for the same initial point.

To be able to make statements about the closeness of the resulting trajectories one has to resort to contraction theory. Contraction theory is based on the idea that the distance, according to some possibly state-dependent metric $M(\boldsymbol{x})$, between any two points subject to the system dynamics should monotonically decrease. This can be proven by looking at the virtual displacement induced by the dynamics $f(\boldsymbol{x})$. However this does not mean that the distance between the demonstrations and the resulting trajectories also decreases at every moment or even on average. Secondly in order to be able to represent *complex* motions using a DS contracting with respect to the metric $M(\boldsymbol{x})$, $M(\boldsymbol{x})$ has to be non-flat (so actually depend on $\boldsymbol{x}$). This state-dependence of the metric however does in turn forbid any meaningful statement on the maximal distance in an l_2-norm sense between two trajectories starting close to each other.

In order to reduce the drift error we propose a statistical model to construct a vector field that locally converges towards the demonstrations. Note that we use this model only to generate the control-space dynamics. Therefore we do not care about the magnitude of the velocity, only the direction, and normalize the resulting velocity vector. In other cases, in which the magnitude does play a role, appropriate changes would have to be made to the model.

The proposed model combines objects called locally converging dynamics. They are composed of a weight function $w : \mathbb{R}^d \mapsto \mathbb{R}_0^+$ (in this case non-normalized Gaussian kernels) and a *direction* $d : \mathbb{R}^d \mapsto \mathbb{R}^d$ function of the form:

Algorithm 1: weightedDirec Weighted direction function

Input: point $\boldsymbol{x}$, target point $\boldsymbol{x}^*$, base direction $\boldsymbol{d}^*$, factor $\alpha \geq 0$, factor $\beta \leq 0$
Output: direction $\boldsymbol{d}$
$\boldsymbol{dy} \leftarrow P_{\boldsymbol{d}^*}.(\boldsymbol{x} - \boldsymbol{x}^*)$
$\boldsymbol{dy}_s \leftarrow \text{sgn}(\boldsymbol{dy})$
$\boldsymbol{dy}_a \leftarrow \text{abs}(\boldsymbol{dy})$
$\boldsymbol{dy}_a \leftarrow \max(0, \boldsymbol{dy}_a - \alpha)$
$\boldsymbol{d} \leftarrow \boldsymbol{d}^* + \sum_i P_{\boldsymbol{d}^*}^{\mathsf{T}}[:, i]\boldsymbol{dy}_a[i]\boldsymbol{dy}_s[i]$
$\boldsymbol{d} \leftarrow \boldsymbol{d}w(\boldsymbol{x})$ **return** $\boldsymbol{d}$

In the first line, the distance vector between the point considered $\boldsymbol{x}$ and the target point of the direction function $\boldsymbol{x}^*$ is projected into the nullspace of the base direction $\boldsymbol{d}^*$, denoted $\boldsymbol{dy} \in \mathbb{R}^{d-1}$. This vector is then element-wise split into its absolute value and sign. The absolute value is then lowered by the factor α and bounded from below by 0. The desired direction $\boldsymbol{d}$ is then given as the sum of the base direction and the sum of the basis vectors

forming the nullspace ($P^{\mathsf{T}}_{\boldsymbol{d}^*}[:, i]$ is the $i-th$ basis vector) multiplied with the projected distance between the point considered and the target point lowered by α and scaled by the factor β. Finally we scale the resulting direction using the weight function w. The resulting vector field therefore converges towards the straight line going through $\boldsymbol{x}^*$ and points in the same direction as the base direction until leaving some neighbourhood defined by α and is shown in 1.
In order to obtain an expressive model, we can combine multiple such locally converging dynamics.

Algorithm 2: COMBINEDWEIGHTEDDIREC Weighted direction function

Input: point $\boldsymbol{x}$, list of target points $\boldsymbol{x}_i^*$, list of base direction $\boldsymbol{d}_i^*$, list of factors $\alpha_i \geq 0$, list of factors $\beta_i \leq 0$, factor base convergence $\epsilon < 0$

Output: direction $\boldsymbol{d}$

$\boldsymbol{d} \leftarrow \epsilon \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$

$\boldsymbol{d} \leftarrow \boldsymbol{d} + \sum_i \text{WEIGHTEDDIREC}(\boldsymbol{x}, \boldsymbol{x}_i^*, \boldsymbol{d}_i^*, \alpha_i, \beta_i)$

$\boldsymbol{d} \leftarrow \frac{\boldsymbol{d}}{\|\boldsymbol{d}\|_2}$

return $\boldsymbol{d}$

This means that we add up all the results of all locally weighted directions and add a *small* radial convergence. Finally we normalize the vector to obtain a direction. A resulting direction field with three components is shown in the bottom right image of Figure 1.

To construct a such a direction field from demonstrations we proceed as following: Given a curve in natural discretized form $\mathcal{C}^{dis}$ and a number of locally weighted converging dynamics n_c, we compute a new discretized representation of the curve with $n_c + 1$ points. We then define the $i-th$ locally converging dynamics as follows: the center of the weighting function is positioned at $(\mathcal{C}^{dis}[i]+\mathcal{C}^{dis}[i+1])/2$[1]. The covariance matrix Σ_i is chosen such that one eigenvector is aligned with $\mathcal{C}^{dis}[i+1] - \mathcal{C}^{dis}[i]$, the eigenvalue is chosen such that the weight $\mathcal{C}^{dis}[i]$ is equal to some a priori fixed value γ. The other eigenvalues (the eigenvalues associated to the basis vectors of the nullspace of $\mathcal{C}^{dis}[i+1] - \mathcal{C}^{dis}[i]$) are chosen as $\zeta\gamma$ (with ζ usually between 0.1 and 2). The values α and β are fixed *a priori*, whereas $\boldsymbol{x}_i^*$ is equal to $\mathcal{C}^{dis}[i+1]$ and $\boldsymbol{d}^*$ is set to $(\mathcal{C}^{dis}[i+1]-\mathcal{C}^{dis}[i])/\left\|\mathcal{C}^{dis}[i+1]-\mathcal{C}^{dis}[i]\right\|_2$. This approach yields a direction field representing the geometry of a given curve, as shown in the left image of Figure 2. If a list of curves is given then we can perform the above procedure for each curve and sum up over all components obtaining a statical model for the given list of curves, as shown in the right image of Figure 2.

1. The notation $\mathcal{C}^{dis}[i]$ is used to denoted the $i-th$ point of the discretized form of the curve $\mathcal{C}$.

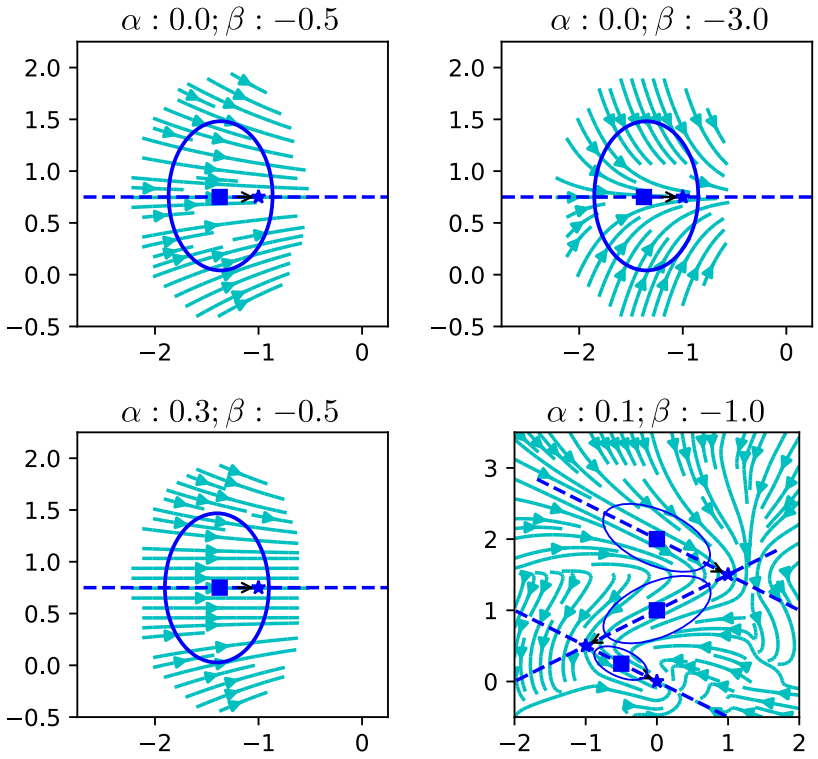

Figure 1: The first three images show the resulting vector field for differently parametrized locally converging dynamics. The dashed line, called base line, and the arrow are aligned with the base direction. The blue square indicates the center of the weight function, the blue star corresponds to $\boldsymbol{x}^*$. The ellipsoids showcase the quantitative shape of the Gaussian kernels. The factor β determines how fast points converge towards the base line, the factor α determines the size of the neighborhood around the baseline where the resulting direction is aligned with the base direction. In the bottom right plot the direction field for a combination of three locally converging dynamics corresponding to alg. 2 with $\epsilon = 10^{-6}$ is shown.

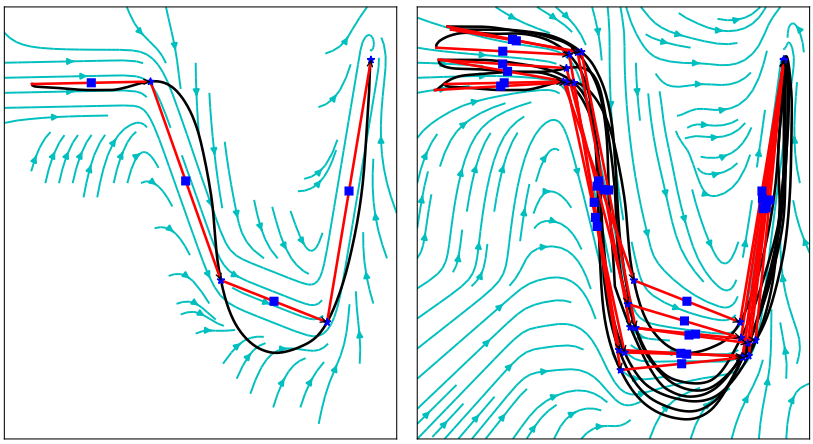

Figure 2: The left image shows the original curve in black. The discretization used to construct the vector field consisting of four components is shown in red. The parameters $\alpha = 2$ and $\beta = -1$ apply to both images. The coefficient ϵ is set to zero n the left image and 10^{-6} in the right image. As one can see, even with very few components the geometry is somewhat well represented. On the right we add the components generated with the same parameters for the other demonstrations. The resulting direction field represent well the variance existing within the set of demonstrations.

Appendix B. Appendix: Heuristic algorithm

In this section of the appendix we will present in greater detail the core component of the heuristic algorithm, the function ComputeCenterAndDir, used to perform approximative curve matching and some other aspects of the algorithm.
The purpose of the function ComputeCenterAndDir is to compute a center and a translation direction which are likely to significantly reduce the distance between the curves when used to construct a locally weighted translation based on statistical considerations.
One of the main reasons why the heuristic proposed in Perrin and Schlehuber-Caissier (2016) fails when presented with multiple demonstrations, is how the center and translation is deduced from the current source curve and error. The center is chosen as the point in the current source curve having the largest error-norm while the translation is taken to be the corresponding error multiplied with an a priori fixed scaling factor α typically between 0.3 and 0.8. This approach creates loops when presented with configurations that are not compatible, leading to undesirable results as shown in Figure 3. The image in Figure 3 depicts an incompatible configuration as we wish to match the (dashed) blue curves onto the (dashed) red curves. Due to the incompatibility, the initial configuration is already a good *solution* for this problem. However the heuristic is unable to detect this resulting in a loop degrading the solution. The first three iterations of the loop are depicted in cyan, orange then magenta. The coloured dots correspond to the chosen center for the next iteration, the line starting at the center depict the maximal error vector. In contrast our approach detects this situation and does not construct any kernels at all, but returns $\Phi = Id$.

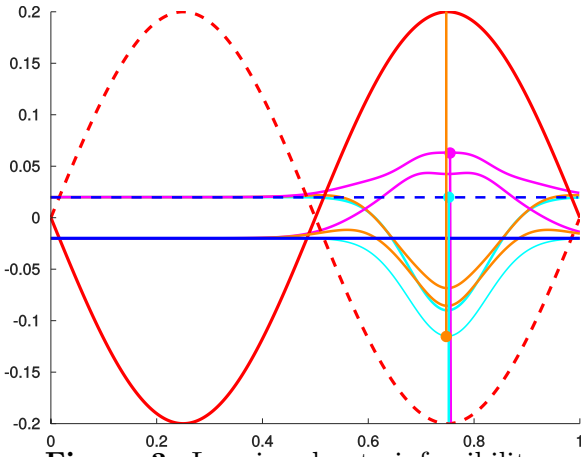

Figure 3: Looping due to infeasibility

The reason for this is the clustering of the concatenation of the source curves and the respective error and the search for the ideal norm of the translation vector. The function ComputeCenterAndDir takes the source curves $\mathcal{C}_s ki^{dis}$ and the current error e_i. In a first step we cluster the concatenation of the source curves and error using the k-means algorithm so

$$m_{cluster} \leftarrow \text{k-means}\left(\begin{pmatrix} \mathcal{C}_{ski}^{dis} \\ e_i \end{pmatrix}_i\right) \tag{2}$$

then we can interpret each center of a cluster $m_{cluster,k}$ as the concatenation of a desirable center and translation of a locally weighted translation

$$\begin{pmatrix} \boldsymbol{c}_k \\ \boldsymbol{u}_{mean,k} \end{pmatrix} = m_{cluster,k} \ . \tag{3}$$

Of all clusters computed we want to select the one which is most like to increase the resemblance of the source and target curve. Therefore we want to chose the one having the largest norm of the translation vector $\boldsymbol{u}_{mean,k}$ but we are also interest in mainly transforming the source curve in a normal direction, as translations in tangential direction correspond to a reparametrization of the curve. We therefore select the pair center, translation

$$\underset{k}{\operatorname{argmax}} \left\|\boldsymbol{u}_{mean,k}\right\|_2 \left(\zeta - \frac{\boldsymbol{u}_{mean,k}^{\mathsf{T}}.\tau}{\left\|\boldsymbol{u}_{mean,k}\right\|_2}\right) \tag{4}$$

where ζ is used to give preference to the norm or direction and is usually chosen between 1.2 and 2 and τ is the average tangential direction of all current source curves in a neighbourhood of $\boldsymbol{c}_k$.
This leads to two possibilities when confronted with incompatibilities: either the error norms are small and therefore k-means seeks to cluster the source curves without regard of the error or vice-versa, so the clusters are mainly chosen to respect the errors. In the first case, the norm of the translation vector is small and the cluster will be discarded. In the second case, the norm of the translation is large, however if this cluster is chosen, due to the relative closeness of source curve points with errors pointing in a different direction, the optimization step will find that the ideal translation is very small and we can discard this cluster. This way our approach can *detect* incompatible settings and avoid the problems of the approach in Perrin and Schlehuber-Caissier (2016).

Appendix C. Appendix: Diffeomorphic multitranslations

The proposed diffeomorphic transformation is based on the composition of so called local multitranslations of the form

$$\Psi(\boldsymbol{x}) = \boldsymbol{x} + \sum_i k_{p_i,b_i,\boldsymbol{c}_i}(\boldsymbol{x}).\boldsymbol{u}_i \tag{5}$$

where $k_{p_i,b_i,\boldsymbol{c}_i}$ denotes the piecewise radial basis function defined as

$$k_{p_i,b_i,\boldsymbol{c}_i} = \begin{cases} p_i(\left\|\boldsymbol{x}-\boldsymbol{c}_i\right\|_2) & \text{if } \left\|\boldsymbol{x}-\boldsymbol{c}_i\right\|_2 \leq b_i \\ 0 & \text{else} \end{cases} \ . \tag{6}$$

This local transformation has zero influence outside of a hypersphere centred at $\boldsymbol{c}_i$ with radius b_i. Inside this hypersphere the weight corresponds to the value of some weight function $p_i(.) : [0, b_i] \mapsto [0, 1]$ evaluated at $r = \left\|\boldsymbol{x}-\boldsymbol{c}_i\right\|_2$.
In order ensure that the each transformation is indeed a diffeomorphism, we need to ensure the following properties:

- $p_i(0) = 1$
- $\frac{d}{dr}p_i(0) = 0$
- $p_i(b_i) = 0$
- $\frac{d}{dr}p_i(b_i) = 0$
- $\max_{0\leq r\leq b_i} |\frac{d}{dr}p_i(r)| \left\|\boldsymbol{u}_i\right\|_2 < 1$.
- $p_i(r)$ is strictly monotonically decreasing from 0 to b_i

Additionally one has to ensure that sum of the translations is also a diffeomorphism, which can be done by checking a conservative lower bound of the smallest eigenvalue of the Jacobian and is based on the following reasoning:
The Jacobian of a multitranslation is given as

$$\begin{aligned} J_\Psi(\boldsymbol{x}) &= \frac{\partial \Psi(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ &= \frac{\partial}{\partial \boldsymbol{x}} \left[\boldsymbol{x} + \sum_i k_{p_i,b_i,\boldsymbol{c}_i}(\boldsymbol{x}).\boldsymbol{u}_i \right] . \end{aligned}$$

As only the region within the base b_i is of interest, assuming the worst case scenario in which the critical point is within the base of all rbfs, we can write

$$\begin{aligned} J_\Psi(\boldsymbol{x}) &= \frac{\partial}{\partial \boldsymbol{x}} \left[\boldsymbol{x} + \sum_i p_i(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2).\boldsymbol{u}_i \right] \\ &\text{with } \|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2 = \sqrt{(\boldsymbol{x}-\boldsymbol{c}_i)^\mathsf{T}.(\boldsymbol{x}-\boldsymbol{c}_i)} \\ &= Id + \sum_i p_i'(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2).\frac{\boldsymbol{u}_i.(\boldsymbol{x}-\boldsymbol{c}_i)^\mathsf{T}}{\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2} . \end{aligned}$$

So we see that each translations adds a rank-1 matrix to the total Jacobian. Now we have to construct a conservative approximation for the smallest eigenvalue that can possibly occur anywhere. Therefore we have to take a closer look at each of the rank-1 matrices

$$J_i = p_i'(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2).\frac{\boldsymbol{u}_i.(\boldsymbol{x}-\boldsymbol{c}_i)^\mathsf{T}}{2.\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2} . \tag{7}$$

For which we can derive an upper bound for the absolute value of the largest eigenvalue

$$\max_\lambda J_i \leq |p_i'(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2)|.\frac{\|\boldsymbol{u}_i\|_2 . \|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2}{\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2} = |p_i'(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2)|.\|\boldsymbol{u}_i\|_2 \tag{8}$$

So as one can see the upper bound for the eigenvalue depends on the norm of the translation and the derivative of the weight function. In this work we use fifth order polynomials as weight functions therefore these values can be easily computed. So the simplest conservative constraint to ensure that Ψ is diffeomorphic is

$$\sum_i |p_i'(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2)|.\|\boldsymbol{u}_i\|_2 < 1. \tag{9}$$

where $p_i'(.)$ denotes the derivative of the weight function.
More involved criteria, which are nonetheless cheap to compute, can be found by separating the space into regions and take into account the bases, the direction of $\boldsymbol{u}_i$ and the sign of $p_i'(\|(\boldsymbol{x}-\boldsymbol{c}_i)\|_2)$ with respect to the region, which have the advantage of being less conservative.

References

Nicolas Perrin and Philipp Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96: 51–59, 2016.