Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting

Tomas Kulvicius[1], KeJun Ning[1], Minija Tamosiunaite[1,2], Florentin Wörgötter[1]

***Abstract*—The generation of complex movement patterns, in particular in cases where one needs to smoothly and accurately join trajectories in a dynamic way, is an important problem in robotics. This paper presents a novel joining method based on the modification of the original dynamic movement primitive (DMP) formulation. The new method can reproduce the target trajectory with high accuracy regarding both, position and velocity profile, and produces smooth and natural transitions in position as well as velocity space. The properties of the method are demonstrated by applying it to simulated handwriting generation also shown on a robot, where an adaptive algorithm is used to learn trajectories from human demonstration. These results demonstrate that the new method is a feasible alternative for joining of movement sequences which has high potential for all robotics applications where trajectory joining is required.**

***Index Terms*—Joining of dynamic movement primitives, overlapping kernels, delta learning rule, handwriting generation**

I. Introduction

Dynamic motion control in robotics requires to accurately produce trajectories and control their dynamic parameters (e.g., position, velocity and acceleration). Several state-of-the-art methods exist for this. These can be based on splines [1], [2], Gaussian Mixture Models [3] or dynamic movement primitives (DMPs, [4]), to name only some of the most important ones. Recently much focus has been directed onto DMPs and they have been used in many studies [5]–[17]. DMPs are units of actions which describe a particular movement trajectory and are formalized as stable attractor systems [4], [18]–[20]. Such movement primitives can be used to generate a movement trajectory either in joint- or task-space. One observes that DMPs are robust to perturbations, allow for generalization of the trajectory and that they also allow for learning. Only few attempts exist which address the problem of DMP joining [7], [14], [15]. Here we present a modification of the original DMP formulation and a novel method for joining of movement sequences based on overlapping kernels. The novel method allows us to apply such DMPs to very complex trajectories and to also address the problem of dynamically joining them, which is the main novel contribution of this study.

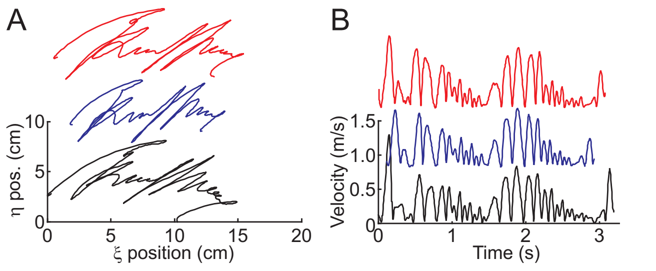


Fig. 1. Three samples of signatures signed by the same person. **A)** Position profiles and **B)** corresponding velocity profiles (here we show velocities as defined in Eq. 14).

Humans are able to perform complex movement sequences, for example in a manual construction task, in a highly elegant and dynamic fashion. Robots begin to become as dexterous, too. However, the problem to accurately combine trajectories and control position *and* velocity, which exists in many dexterous robotics applications still leaves many questions open, in particular when wanting to do this in a dynamic and perturbation-resistant way. We use human handwriting as our test-bed. Although, in handwriting accuracy at the joining point might not be so important due to "co-articulation", i.e., the next letter is started before the previous one is completely finished, we have nonetheless chosen it as a useful and complex example. Consider, for example, signatures. Those are not only defined in position- but also in velocity space and modern biometric signature recognition makes use of this [21]. In Fig. 1 A we show three samples of signatures from the same person. One can see that, although all three signatures are slightly different, the corresponding velocity profiles (see Fig. 1 B) have highly similar patterns. How can such complex dynamic profiles be generated in a simple, robust and generalizable way? We solve this problem by presenting a novel method for joining movement sequences which can be also applied for other potential uses, especially where precision at the joining point would be crucial, e.g., for grasping and lifting an object.

In the following we will first describe the formalism for

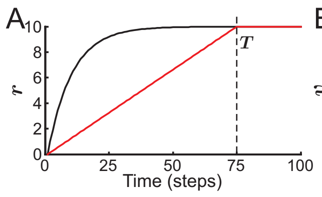
The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience. This research was also supported by BMBF (Federal Ministry of Education and Research), BCCN (Bernstein Center for Computational Neuroscience) - Göttingen project W3 and BFNT project 3a.

Authors are greatly thankful to Dr. Armin Bies from MPIDS Göttingen who provided a pen tablet and helped with handwriting recordings.

[1]Georg-August-Universität Göttingen, Bernstein Center for Computational Neuroscience, Department for Computational Neuroscience, III Physikalisches Institut - Biophysik, Friedrich-Hund Platz 1, D-37077 Göttingen, Germany

[2]Department of Informatics, Vytautas Magnus University, Vileikos g. 8, Kaunas, Lithuania

E-mail: {tomas; worgott}@physik3.gwdg.de, m.tamosiunaite@if.vdu.lt



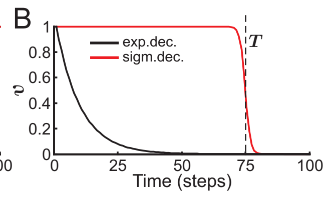


Fig. 2. Modification of the original DMP system. **A**) Goal functions r and **B**) functions v are shown for the exponential decay system (original DMPs) and the sigmoidal decay system (modified DMPs). The following parameters were used: $g = 10$, $\alpha_g = 0.1$, $T = 75$, $\alpha_v^e = 0.1$ and $\alpha_v^s = 1.0$. For more details see section II-A.

DMP modification and our novel joining approach. One goal here is to provide a powerful but simple method. Then we will compare our approach to a basic DMP system. We want to strongly emphasize that this comparison is not made to devalue the old approach, but rather to show that the new approach can perform equally well additionally allowing us to join movement sequences in a smooth and accurate way. To make this very clear we will, in the discussion section, try to provide a scholarly and fair comparison of some of the existing DMP methods.

II. Methods

A. Modification of original DMPs

The original DMP system is formalized by second or third order differential equations and consists of two dynamic systems: the transformation system and the canonical system [4], [8], [18], [22]. Here we will use the third order system since it more closely relates to our modifications. The transformation system is described as follows:

$$\tau \dot{z} = \alpha_z \left(\beta_z \left(r - y\right) - z\right) + f, \tag{1}$$

$$\tau \dot{y} = z, \tag{2}$$

$$\tau \dot{r} = \alpha_g \left(g - r\right), \tag{3}$$

where g is a known goal state (end-point), α_z , β_z and α_g are time constants, τ is a temporal scaling factor (in this study we used $\tau = 1$), $\dot{z}$, $\dot{y}$ and y correspond to acceleration, velocity and position, respectively. Here r defines the delayed goal function.

The canonical system is described by an exponential decay term:

$$\tau \dot{v} = -\alpha_v^e \, v, \tag{4}$$

where α_v^e is a time constant. The nonlinear function f is given by:

$$f = \frac{\sum_{i=1}^{n} \psi_i \, w_i \, v}{\sum_i \psi_i} \left(g - y_0\right), \tag{5}$$

with

$$\psi_i = e^{-h_i \, (v - c_i)^2}, \tag{6}$$

where ψ_i denote Gaussian kernels, h_i - width of the $i - th$ kernel, c_i - centers of the kernels, w_i - weights, and n - number of kernels. Here $(g - y_0)$ is the scaling term which guarantees proper scaling when start- or end-point is changed. For more details see [8]. In addition, coupling terms such as a phase stopping can be added to the DMP system. In this case one would need to replace Eq. 4 by the following equation as suggested in [8]:

$$\tau \dot{v} = -\frac{\alpha_v^e \, v}{1 + \alpha_c (y_{desired} - y_{actual})^2}, \tag{7}$$

where α_c is a time constant and defines the strength of the phase stopping. The implications of systems with or without phase-stopping will be addressed in the Discussion section. Here, if not mentioned differently, we are going to use Eq. 4.

In order to apply our new joining method using overlapping kernels we modify the original DMP system by changing the delayed goal function r and the exponential decay function v (see Eq. 3, 4 and Fig. 2). The non-linear goal function r is replaced by a piecewise-linear function and formalized as follows:

$$\tau \dot{r} = \begin{cases} \frac{\Delta_t}{T} \left(g - s\right), & \text{if } t \leq T \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Here s and g define the start- and end-point of the movement trajectory, respectively; T is the duration of the movement and Δ_t is the sampling rate. In this study we used $\Delta_t = 5 \, ms$ which corresponds to $200 \, Hz$ and is consistent with the sampling rate of the pen tablet used for the recordings.

Instead of the exponential decay function we use a sigmoidal decay function which is given by:

$$\dot{v} = -\frac{\alpha_v^s \, e^{\frac{\alpha_v^s}{\Delta_t} (\tau T - t)}}{(1 + e^{\frac{\alpha_v^s}{\Delta_t} (\tau T - t)})^2}, \tag{9}$$

where α_v^s defines the steepness of the sigmoidal function centred at time moment T.

We also use a slightly different nonlinear function f:

$$f = \alpha_w \frac{\sum_{i=1}^{n} \psi_i \, w_i \, v}{\sum_i \psi_i}, \tag{10}$$

$$\psi_i = e^{-(\frac{t}{\tau T} - c_i)^2 / 2\sigma_i^2}, \tag{11}$$

where σ_i is the width of the $i - th$ kernel. Here kernels are placed evenly along the trajectory in time and spaced between 0 and 1, where 0 denotes the beginning of the movement trajectory and 1 the end. The shape of the movement trajectory is defined by weights w_i. To learn the weights we use the delta rule as explained in section II-B. Here we use α_w as a general scaling factor for all learned weights (note that during learning we always set $\alpha_w = 1$), and in this study, if not specifically mentioned, after learning we also set it to 1. For the influence of α_w on the movement trajectory see Fig. 4 D-G.

The modified system will reach the end-point g of the movement trajectory y in time $T + \lambda$ (usually $\lambda \ll T$) with a given accuracy $|g - y(t)| \leq \varepsilon$, where λ depends on the steepness α_v^s, the center point of the sigmoidal function (Eq. 9), and the movement trajectory itself. I.e., depending on how far one is from the target point at the end T of the movement, the additional time to arrival (λ) at the target may vary. This strongly depends on how well the nonlinear function f fits the target trajectory. If it fits well, then the trajectory might arrive at the target within little additional time. Otherwise, the additional time required could be considerable, such that one cannot always ensure reaching the target at a specific time.

In the original DMP system the exponential function (see Eq. 4) acts as a phase as well as a weight scaling variable. In our modified DMP system the sigmoidal function (see Eq. 9) acts only as a weight scaling variable as our DMP system operates on a scalable time axis. We will in the last section discuss the differences between a phase-defined or a time-defined DMP system because advantages and disadvantages of the different formulations are not easily discernible.

B. Trajectory learning

The shape of a trajectory in the DMP system is parameterized by Gaussian kernels ψ_i and their corresponding weights w_i. Several methods exist for weight adaptation in order to learn a target trajectory, like locally weighted regression methods [4] or global regression methods [14]. As learning is not in the focus of this study, here we just used the delta learning rule as in artificial neural networks [23], where the error between target signal and system's output is used to modify the weights. This rule is simple, but serves our purposes. It can be formalized by the following equation:

$$\Delta w_i^j = \mu\,[\gamma(k) - y^j(k)], \tag{12}$$

where γ is the target trajectory (training trajectory), y is the system's output, μ is the learning rate, and k defines the center of the i-th Gaussian kernel in the time period $t = 0, \ldots, T$; where T is the duration of the training trajectory γ. Here $j = 1, \ldots, L$; where L is the number of learning iterations. In this study we used the same training trajectory γ for each training iteration j, but in general one can use different training trajectories, e.g., from several demonstration examples. In case we have more than one dimension then we have to learn weights for each dimension separately (e.g. for x and y position in a 2D case), which can be done independently and in parallel. Note that here we learn weights in position space as acceleration is noisier compared to position.

The presented simple learning approach is not optimal for a single training trajectory. It is more suitable for cases where many different training trajectories are used and this method will then lead to an average trajectory. If desired any other regression method (or more sophisticated methods like LWPR) can be used to learn the kernel weights.

As already mentioned above, we tested our modified DMPs on trajectories obtained from handwriting. Handwriting samples were taken from one person, where the person was asked to sign, write single letters and couplets (e.g. "ab"). Data were obtained by using a pen tablet (Wacom Intuos3 A3 Wide DTP) with a size of $48.8 \times 30.5\,cm$, resolution of $5,080\,lpi$ and a sampling rate of $200\,Hz$.

In order to evaluate the performance of our method and compare it to the original DMPs we looked at the position and velocity deviation of the learned trajectory from the target

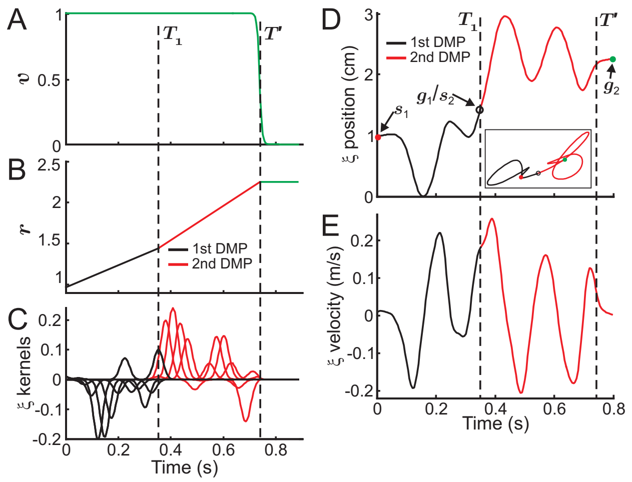


Fig. 3. Illustration of the DMP joining method with overlapping kernels. Here we show only signals for the ξ profile of joining two letters "a" and "b". **A)** Sigmoidal decay function v, **B)** goal function r, **C)** kernels ($\psi_i' w_i' v$), **D)** ξ position (y_ξ); in the inset we show the trajectory of the joined "a" and "b", and **E)** ξ velocity ($\dot{y}_\xi$).

(training) trajectory. We calculated the position deviation by

$$d_{pos} = \sqrt{(y_{\xi_t} - y_{\xi_l})^2 + (y_{\eta_t} - y_{\eta_l})^2} \tag{13}$$

and the velocity deviation as

$$vel_{t/l} = \sqrt{\dot{y}^2_{\xi_{t/l}} + \dot{y}^2_{\eta_{t/l}}}, \tag{14}$$

$$d_{vel} = |vel_t - vel_l|, \tag{15}$$

where $y_{\xi/\eta}$ and $\dot{y}_{\xi/\eta}$ are position and velocity, ξ and η denote two coordinates in 2D Cartesian space and indices t and l stand for the target and learned profiles, respectively.

C. Joining of DMPs

A simple way to join several DMPs is to perform one DMP until it reaches the end-point and then to start the next DMP at that point (in the text we will call this *simple joining*), i.e., to use the end-point of the first DMP as the start-point of the second DMP. This approach is very simple, but it has some drawback due to the close to zero velocities at the end of the movement trajectory in the original DMPs. Here we propose a novel method for DMP joining which can solve this problem and lead to natural and smooth transitions. In our approach we construct a single set of overlapping kernels ψ' defined by centers c' and width of kernels σ' for the whole joint trajectory in the following way. We place centers c' along the joint trajectory in time as follows:

$$c_j^i = \begin{cases} \frac{T_1\,(i-1)}{T'\,(n-1)}, & \text{if } j = 1 \\ \frac{T_j\,(i-1)}{T'\,(n-1)} + \frac{1}{T'}\sum_{k=1}^{j-1} T_k, & \text{otherwise,} \end{cases} \tag{16}$$

$$c' = c_1^1, c_1^2, \ldots, c_1^n; c_2^1, c_2^2, \ldots, c_2^n; \ldots; c_m^1, c_m^2, \ldots, c_m^n. \tag{17}$$

Here $i = 1, \ldots, n$ and $j = 1, \ldots, m$, where n is the number of kernels for one DMP and m defines the number of DMPs, T_k

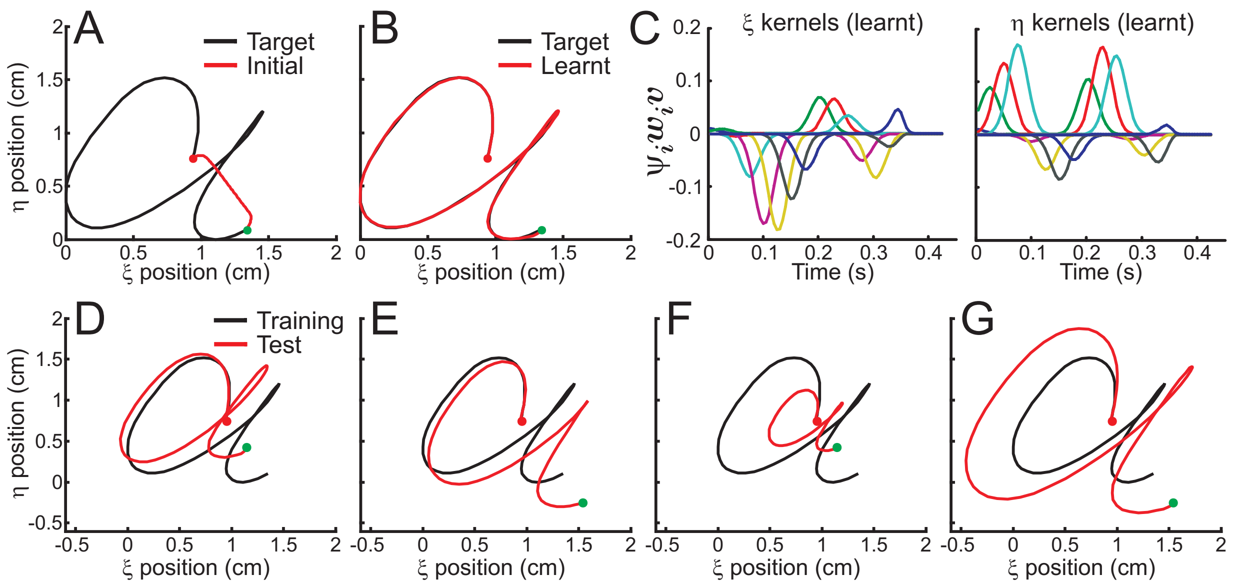


Fig. 4. Results of trajectory learning using modified DMPs (sigmoidal decay system). **A)** Trajectories before learning and **B, C)** trajectories and kernels ($\psi_i w_i v$) for ξ and η position profiles after learning (initially all weights were set to 0.01). **D-G)** Examples of generalization when different end-points are used. **D, E)** Trajectories without weight scaling ($\alpha_w = 1$) and **F, G)** with weight scaling $\alpha_w = |(g_t - s_t)/(g_l - s_l)|$ (similar to the weight scaling used in the original DMP system, [8]), where s_l and g_l denote the start- and end-points of the training trajectory used for learning, and s_t and g_t are start- and end-points of the test trial. The following system parameters were used for all cases: number of kernels $n = 15$, width of kernels $\sigma_i = 0.05$, $i = 1, \ldots, n$, number of learning iterations $L = 100$, and learning rate $\mu = 0.1$. Red and green dots represent start- and end-points of the movement trajectory, respectively.

is the duration of the k-th DMP. $T' = \sum_{k=1}^{m} T_k$ is the duration of the joint trajectory. We define the width of kernels σ' by

$$\overline{\sigma}_j^i = \frac{\sigma_j^i T_j}{T'}, \tag{18}$$

$$\sigma' = \overline{\sigma}_1^1, \overline{\sigma}_1^2, \ldots, \overline{\sigma}_1^n; \overline{\sigma}_2^1, \overline{\sigma}_2^2, \ldots, \overline{\sigma}_2^n; \ldots; \overline{\sigma}_m^1, \overline{\sigma}_m^2, \ldots, \overline{\sigma}_m^n, \tag{19}$$

where we scale the width of kernels σ_j^i of each DMP with respect to the duration of the joint trajectory T'. Corresponding weights w' for kernels ψ' are obtained from the learned weights w_j^i of each dynamic movement primitive j:

$$w' = w_1^1, w_1^2, \ldots, w_1^n; w_2^1, w_2^2, \ldots, w_2^n; \ldots; w_m^1, w_m^2, \ldots, w_m^n. \tag{20}$$

Finally we describe the goal function r' by the following equation:

$$\tau \dot{r}' = \begin{cases} \frac{\Delta_t}{T_j}(g_j - s_j), & \text{if } \sum_{k=1}^{j-1} T_k \le t \le \sum_{k=1}^{j} T_k \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

where j=1,…,m. We use the same equation for the sigmoidal decay function as given in Eq. 9, only here we use T' instead of T. This way we construct a more "complex" DMP from several "simpler" DMPs. A graphical illustration of this DMP joining method is shown in Fig. 3, where we show signals obtained from joining two letters "a" and "b".

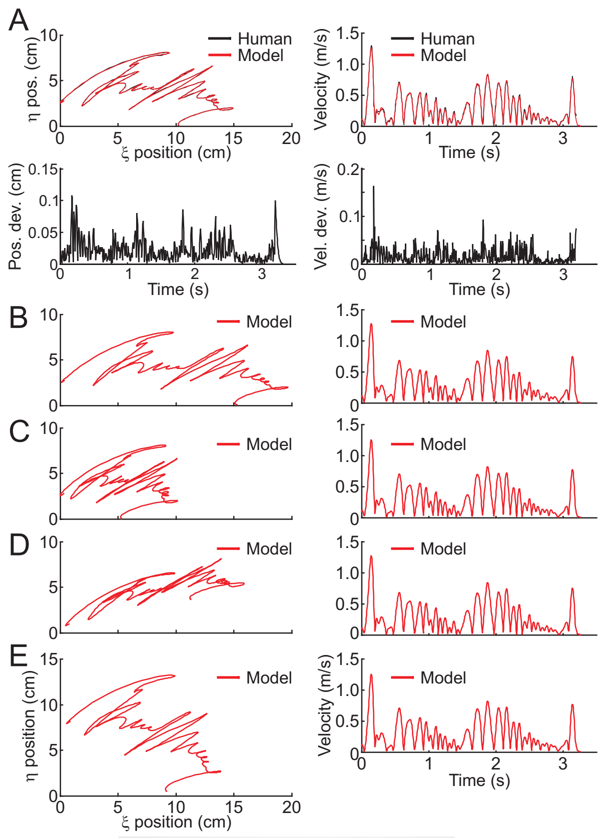
The joining method is designed in such a way that one can control time of the whole joint DMP by setting T' and spatial stretching and squeezing by setting g_j and s_j (see Eq. 21). Note, similar to conventional DMPs, one can also join our DMPs in a "reactive" way. To do so, one has to set start- and end-points for the new joint DMP (see Eq. 21) and recalculate kernel parameters (c', σ' and ω') as soon as it is known that a new DMP will be triggered. Then one continues with the new DMP by using Eqs. 1, 2, 9-11 and 21 with the new parameters. If one knows early that a new DMP is triggered (e.g., in the middle of the currently executed DMP) then there will be no differences at the joining point as compared to "non-reactive" (pre-planned) joining. If we join the next DMP when the current one is almost finished (around the joining point T_j) then there will be a slowing-down in the velocity profile (similar to conventional DMPs).

III. Results

A. Learning and generalization

An example for trajectory learning is presented in Fig. 4. As a training (target) trajectory we used a letter "a". Position profile before learning and position profile and corresponding kernels for ξ and η positions after 100 learning iterations are shown in panels A-C. We can see that initially the system generates more or less a linear trajectory from the start-point (red dot) to the end-point (green dot), whereas during learning the weights adapt in such a way that after learning we obtained a trajectory which is almost identical to the training trajectory.

To show how well our model can generalize we made an experiment where we tested the response of the model after learning when the end-point of the movement trajectory was changed. In one case we set the end-point closer to the start-point as compared to the target trajectory; in the second case the end-point was moved further away from the start-point. In



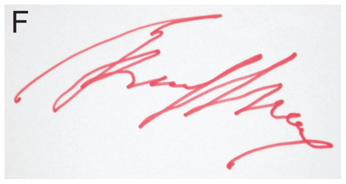


Fig. 5. **A)** Comparison of signatures signed by a person and generated by the model (modified DMPs). Position profiles (top-left), corresponding velocity profiles (top-right) as well as position deviation (bottom-left) and velocity deviation (bottom-right) are shown. Those have been obtained by using Eq. 13 and 15. **B-E)** Signatures generated by the model when different start- and end-points where used, which correspond to horizontal stretching and squeezing **(B, C)**, and diagonal skewing **(D, E)**. Position profiles (left) and corresponding velocity profiles (right) are shown for each case. **F)** Signature performed by the robot (compare to the case shown in panel **A**). The following system parameters were used: number of kernels $n = 100$, width of kernels $\sigma_i = 0.01$, $i = 1 \dots n$, number of learning iterations $L = 100$ and learning rate $\mu = 0.1$.

Fig. 4 D, E we show movement trajectories when the weight scaling factor α_w (see Eq. 10) was set to 1 (no scaling). In this case we obtain a squeezing (D) or stretching (E) effect compared to the original trajectory. If we use a scaling factor which is proportional to the relation between start- and end-points of the training and test trajectories (for equation see the caption of Fig. 4) then we get a scaling of the original trajectory which leads either to a smaller (F) or bigger letter "a" (G). The scaling effect presented in panels F and G is similar to the scaling used in the original DMPs [8] (data not shown). Clearly, choosing α_w would depend on the particular application and further in this study we will always use $\alpha_w = 1$.

To get a better feeling for the dynamic reshaping and generalization, Fig. 5 A shows a comparison between a human signature and a signature generated by our modified DMPs. Here, as in the previous case, we show results after 100 learning iterations, however, in this case narrower kernels were used (for parameters see the caption of Fig. 5). We can see that trajectories of both, signature shapes (left) and corresponding velocities (right), are almost identical as can be judged by the very small errors shown in the position and velocity deviation curves (lower panels in Fig. 5 A). In Fig. 5 B-E we present results of the model for stretching (B), squeezing (C) and for two cases of diagonal skewing (D and E) to give an impression how the new model generalizes when positions of the start- and end-points are changed. The velocity profiles shown on the right side remain almost identical for panels (B) to (E) showing that the dynamics of the changed signatures remain the same. We have also implemented our method on a simple robotic manipulator platform (Neuro-Robotics, Sussex) in our laboratory, letting the robot sign. The resulting trajectory[1] is shown in Fig. 5 F (compare to the signature shown in panel A). Note that we scaled the output of the system in space and time in order to implement it on the robotic platform.

B. *Original vs. modified DMPs*

A comparison between both systems, i.e., original and modified DMPs, is shown in Fig. 6 (for statistics see Fig. 8 A, B). Here we trained both systems with the letters "a" and "b". Results obtained with the original DMPs are shown in panels A and C. We can see that in both cases the original DMPs can follow the target trajectories very well. We can also observe the drawback of the original DMPs which is a relatively long and shallow tail of the velocity profile (see panels A3 and C3). This means that the velocity decays slowly at the end of the movement until the end-point of the trajectory is reached. In comparison, results for the modified DMPs (sigmoidal decay system) are shown in panels B and D, where one can see that, as in the original DMPs, we can reproduce trajectories very well, too. We note that modified DMPs reach the end-point earlier compared to the original DMPs[2]. This is due to the fact that we use a steep sigmoidal function at the end-point of the movement trajectory which results in quicker convergence to the goal. In our approach we obtain bigger errors compared to the whole trajectory only at the end of the movement trajectory (see panels E1, F1) due to the suppressed influence of the kernels by the sigmoidal function, which is necessary to assure convergence to the end-point. We observe that we can reproduce not only the position but also the velocity profile (see panels B3, D3). Here we get a relatively large error at the end of the trajectory and this is due to the fact that here humans lift the pen, whereas our system always stops at the end of the trajectory. Also, in the modified approach we have a much quicker velocity decay at the end of the movement compared to the original DMPs (compare A3 to B3 and C3 to D3). In general, we observe that modified DMPs can compete

[1]See supplementary video.

[2]Here we stopped the system as soon as it reached an accuracy of $\varepsilon = 0.1\,mm$ with respect to the end-point.

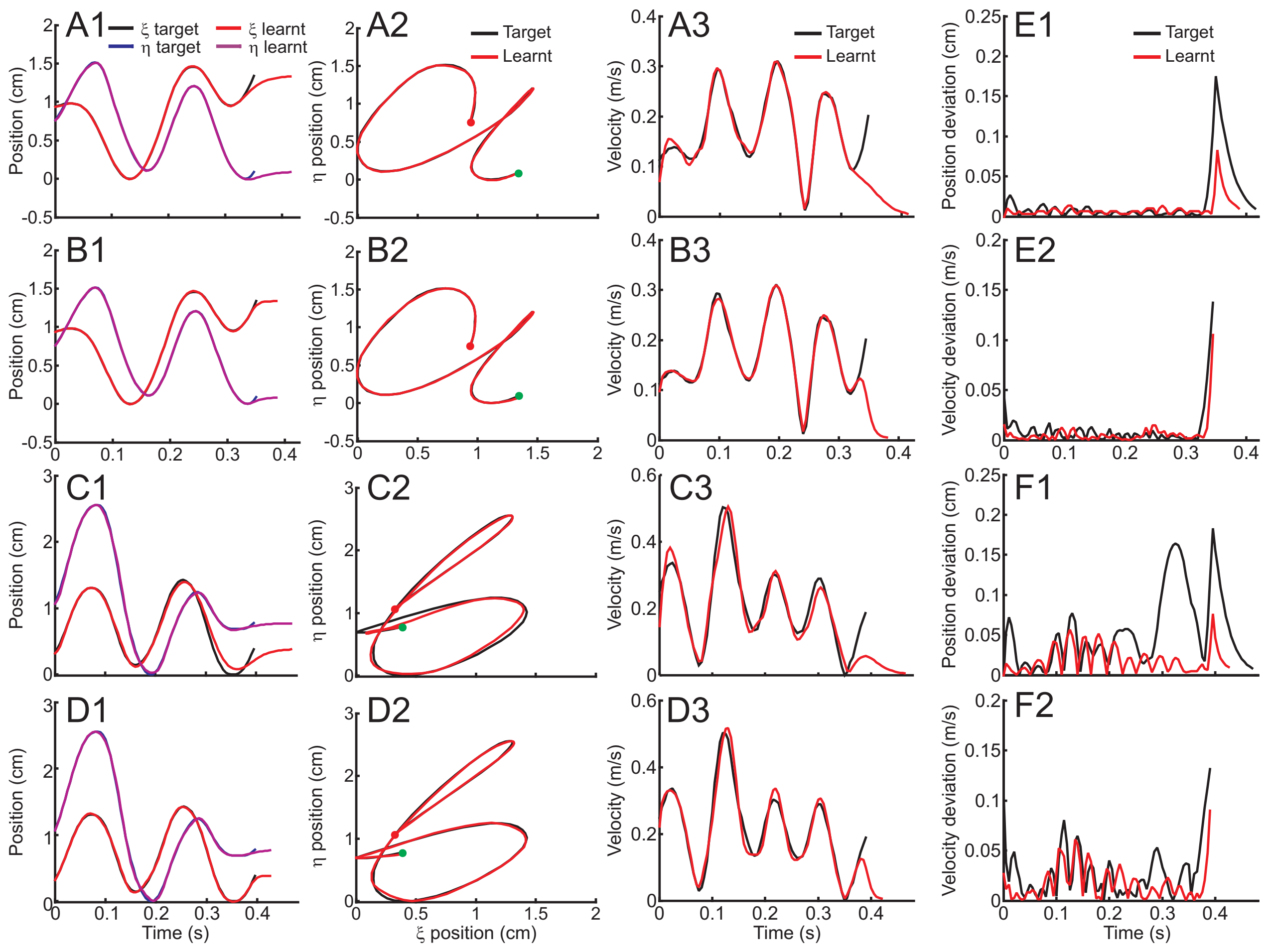


Fig. 6. Comparison between original DMPs (exponential decay system) and modified DMPs (sigmoidal decay system) for letters "a" and "b". **A, C)** Results for the exponential decay system. Here we used α_v^s such that the exponential decay function is 95% converged by the time the end-point g is to be reached as suggested in [8]). **B, D)** Results for the sigmoidal decay system (we used $\alpha_v^s = 1.0$). In panels **A-D** (from left to right) we show ξ and η trajectories plotted over time, position profiles and the corresponding velocity profiles. **E1, E2)** show position (Eq. 13) and velocity (Eq. 15) deviations for the letter "a" and **F1, F2)** for the letter "b". The following system parameters were used for both systems: number of kernels $n = 15$, width of kernels $\sigma_i = 0.05$ (for the exponential decay system we tuned h such that it has the same kernel width in time space as for the sigmoidal decay system), and learning rate $\mu = 0.1$. Number of learning iterations L for the exponential decay system was 500 and for the sigmoidal decay system it was 100. Red and green dots represent start- and end-points of the movement trajectory, respectively.

with original approach very well with respect to position and velocity accuracy (see panels E and F).

C. Joining of DMPs

A comparison between the simple joining method based on original DMPs and joining with overlapping kernels based on modified DMPs is shown in Fig. 7. Here we show the joining of two letters "a" and "b" where samples "a" and "b" were obtained from a handwritten couplet "ab" by splitting it apart. Systems were trained separately with letters "a" and "b" in order to obtain two DMPs and then these DMPs were joined by using one joining method or the other. Note that in the simple approach we start a new DMP at time moment T_1, i.e., when the first DMP is supposed to finish according to the training trajectory. As mentioned above, accuracy at the joining point depends on the overlap of the kernels at that point. To investigate this we varied the number of kernels n and the width σ and analysed trajectories. In comparison we also show results for different kernel parameters using original DMPs. Results for four different cases are shown in Fig. 7 (parameters are given in the caption) where in A we show results for the simple joining method and in B for the new approach. We observe that using more kernels leads to better accuracy at the joining point for both approaches and that the new method is more accurate at the joining point as compared to the simple approach, especially, with respect to the velocity profile. In the following we will discuss the influence of kernel overlap at the joining point on the accuracy at that point for our new method. In B1 we show position and velocity profiles where we used few but relatively wide kernels ($n = 10$, $\sigma = 0.1$). One can see that, although the transition between two letters is relatively smooth, position and velocity accuracy is much worse as compared to the case when more narrower kernels ($n = 15$, $\sigma = 0.05$) are used as shown in B2. If we use even narrower kernels ($n = 30$,

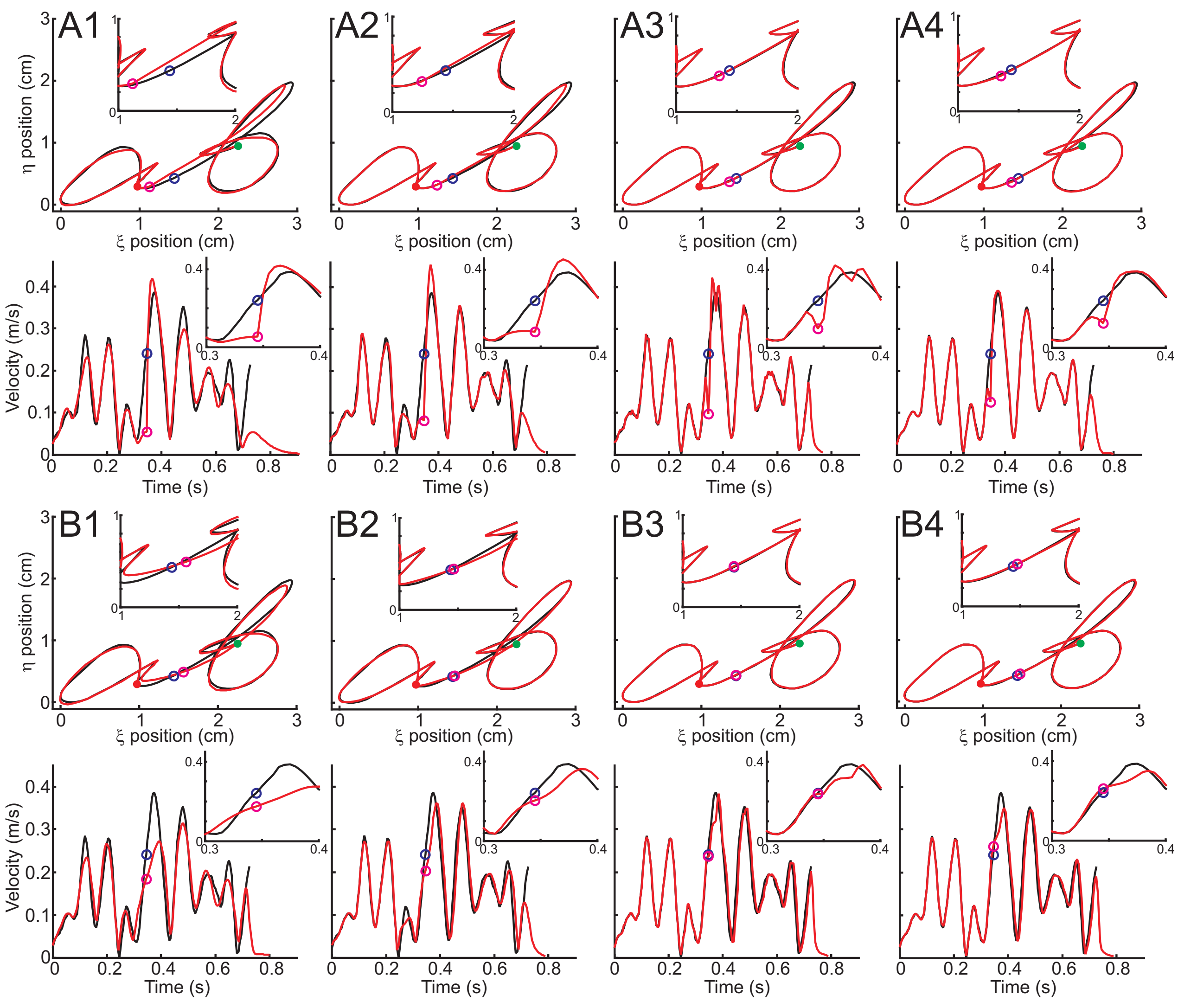


Fig. 7. Comparison between **A)** simple DMP joining (exponential decay system) and **B)** joining with overlapping kernels (sigmoidal decay system) when using different kernel parameters n (number of kernels) and σ (width of kernels): **1)** $n = 10$, $\sigma = 0.1$; **2)** $n = 15$, $\sigma = 0.05$; **3)** $n = 30$, $\sigma = 0.015$; **4)** $n = 30$, $\sigma = 0.03$. In the top panels we show position profiles, corresponding velocity profiles are shown in the bottom panels. In the insets we show magnified trajectories around the joining point. The same system parameters were used as given in the caption of Fig 6. Red and green dots represent start- and end-point of the movement trajectory, respectively, whereas circles represent junction points between two DMPs.

$\sigma = 0.015$; see B3) as compared to the case B2, we can achieve better accuracy with respect to both, position and velocity, profiles; however, this can lead to small jerks in the velocity profile (see inset in panel B3, bottom) due to such overly slim kernels. One can improve smoothness of the velocity by increasing the width of the kernels as shown in B4, but this might lead to bigger inaccuracy at the joining point in the position profile due to larger overlap at this point. Position and velocity deviations at the junction point for both approaches are given in Table I. In general, one can see that the new method performs better with respect to the accuracy at the joining point as compared to the simple method. Concerning the new approach, we can summarize that more kernels lead to better accuracy, but if kernels are too narrow this might produce jerky velocity profiles, whereas if kernels are too wide this might lead to bigger inaccuracies at the joining point.

D. *Statistical evaluation*

We have also performed a statistical evaluation of our system's performance in order to compare it to the original DMPs. First of all we compared the performance of both systems on single letter experiments as presented in Fig. 6. For this we used all handwritten letters from the English alphabet, except "i", "j" and "t" (in total 23 letters). Those letters were not included in the statistics due to an additional dimension (pen-up/down). We looked at the position and velocity deviation (d_{pos}, d_{vel}) from the target trajectory at specific time points along the movement trajectory. Results of such experiments are shown in Fig. 8 A, B and are consistent with the results shown in Fig. 6. We get similar performance along the whole trajectory for both systems except at the end of the movement where the modified DMP system has smaller deviations around the time where the movement is suposed to finish, which leads

TABLE I
DEPENDENCE OF ACCURACY AT THE JOINING POINT ON THE KERNEL PARAMETERS n (NUMBER OF KERNELS) AND σ (WIDTH OR KERNELS).

		Original DMPs		Modified DMPs	
n	σ	$d_{pos}\,(cm)$	$d_{vel}\,(m/s)$	$d_{pos}\,(cm)$	$d_{vel}\,(m/s)$
10	0.100	0.3454	0.1867	0.1366	0.0669
15	0.050	0.2269	0.1601	0.0276	0.0383
30	0.015	0.1002	0.1440	0.0081	0.0036
30	0.030	0.1052	0.1158	0.0464	0.0203

to faster convergence compared to the original DMPs (see panel A). Concerning velocities (see panel B) there are no significant differences and both systems perform equally well except for the fact that modified DMPs have faster velocity decay at the end of the movement (see also Fig. 6 A3, B3 and C3, D3).

Results for the statistical evaluation of the joining methods are presented in Fig. 8 D, E. For the statistics we used 20 different couplets as shown in Fig. 8 C and the same procedures as explained in section III-C. Here we compare position and velocity deviation of joint DMPs at their junction point from the target junction point (denoted by the red dot). As demonstrated previously in Fig. 7 we observe that the new joining method produces significantly smaller errors at the joining point than the simple approach with respect to both, position and velocity. We also see that accuracy at the joining point depends on the overlap of the kernels and can be improved by increasing the number of kernels. Note that many too wide kernels might lead to worse accuracy in the position profile due to large overlap of kernels at the joining point (see case $n = 30, \sigma = 0.03$ in panel D).

E. Handwriting generation

Finally, we applied the novel DMP joining method to the problem of generating handwriting. For this we trained the system on separately written letters as shown in Fig. 9 A. In this case we used samples with loops in order to avoid overlapping letters when joining DMPs. After that we let the system generate three words: “dmp”, “demo” and “norway” (where two of the authors had gone fishing together). Position and corresponding velocity profiles generated by the system for the first two words are shown in Fig. 9 B-F, where we can observe that connections between letters are smooth and natural. As in the previous case this handwriting generation was also implemented on the robotic manipulator platform[3] and resulting trajectories are shown in Fig. 9 D, G. In addition we also show the system’s performance by letting it generate different handwriting styles for the word “norway”. This was done by choosing different end-points with respect to the start-point and rescaling the start- and end-points s_j, g_j (see Eq. 21). Similarly, as already shown for the single letter “a”, we get squeezing/stretching/skewing effects of the whole word by setting the end-point closer/further away and/or higher/lower with respect to the start-point and setting the scaling parameter $\alpha_w = 1$ (words shown in the corners of panel H). If we move the end-point further away but set $\alpha_w > 1$ we obtain spatial magnification of the word (corresponds to bigger font size) and vice versa (see words in the middle of the panel H), moving the end-point closer and setting $\alpha_w < 1$ leads to shrinking of the word (corresponds to smaller font). By doing so, we can see that we retain smooth and natural transitions between the letters.

[3]See supplementary video.

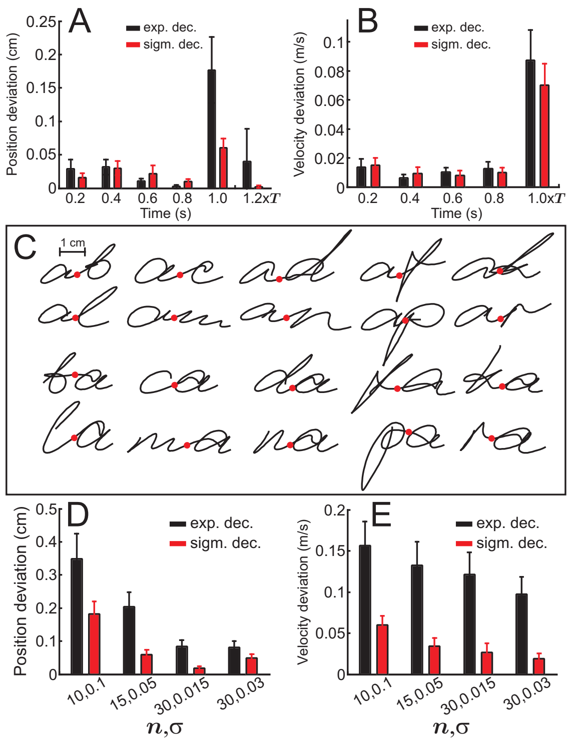


Fig. 8. **A, B)** Statistical comparison between original DMPs (exponential decay system) and modified DMPs (sigmoidal decay system). **A)** Average position deviation d_{pos} and **B)** velocity deviation d_{vel} obtained from 23 letters. **C-E)** Statistical comparison between joining with the simple method based on original DMPs and the joining method with overlapping kernels based on modified DMPs when using different kernel parameters n (number of kernels) and σ (width of kernels). **C)** 20 couplets used for the statistical evaluation shown in panels D, E. Red dots denote the junction points. **D)** Average position deviation d_{pos} and **E)** velocity deviation d_{vel} computed at the junction points as shown in panel C. Here error-bars denote confidence intervals of the mean (95%). The same system parameters were used as given in the caption of Fig. 6.

IV. DISCUSSION

In this study we presented a modification of the original DMP formulation [4], [8], [22] and a novel method for joining several DMPs which was applied for handwriting generation and also implemented on a robot. Our new method has led to a high accuracy for reproducing complex trajectories and joining movement sequences. We have also statistically evaluated our system’s performance and compared it against the standard DMPs. In the following we will discuss our methods in more detail and compare them to other approaches of DMP joining, focusing on a robotics viewpoint. We think, it is clear that handwriting was just chosen as a demonstration scenario and we are neither interested nor will discuss details and implications of handwriting biometry.

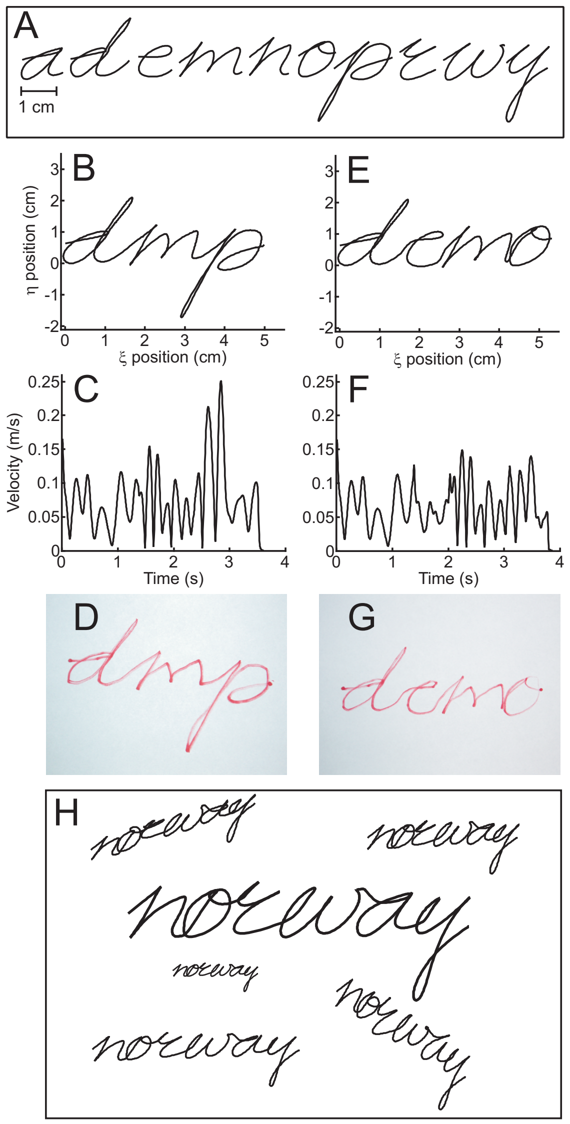


Fig. 9. Results of handwriting generation using modified DMPs (sigmoidal decay system). **A)** Samples of single letters used to generate the words "dmp" (panels **B-D**), "demo" (panels **E-G**) and "norway" (panel H). **B, E)** Position profiles and **C, F)** corresponding velocity profiles generated by the modified DMP system. **D, G)** Trajectories of handwriting produced by the robot. **H)** Example of different handwriting generation by changing end-points (simulation results). The same system parameters were used as given in the caption of Fig. 6.

In the first part of this study we proposed a modification of the original DMP formulation in order to apply our new joining method based on overlapping kernels. Here we made two modifications: 1) instead of the delayed goal function r we used a piece-wise linear goal function (see Eq. 3, 8) and 2) we replaced the exponential decay function v with a relatively steep sigmoidal function centered at the end-point of the trajectory (see Eq. 4, 9). These modifications allow reproducing target trajectories with respect to position as well as velocity profile, while at the same time guaranteeing good convergence to the end-point (see Fig. 6 B, D). Note that we use the linear goal function r together with kernels ψ which are independent of the function v, i.e., our system is directly dependent on time (how to alter this will be discussed below). This makes our approach more similar to splines, but we have maintained attractive features of original DMPs, like robustness to perturbations (see Fig. 10 B1, B2) and generalization (see Fig. 4 D-G and Fig. 5 B-E). We think that such features are very useful and provide some advantages over splines, where they are not immediately present. An unequivocal decision in favor or against any method, however, depends for all these approaches much on the task and sometimes splines might still be a better solution. We also would like to stress that our system is easily scalable with respect to the dynamics of the system (τ) and space without losing the qualitative trajectory appearance that was originally coded in the DMP weights. Depending on the application, two generalization options for the trajectory are possible with our DMPs. In one case, if $\alpha_w = 1$ (see Eq. 10), then we get a stretching or squeezing effect of the learned trajectory when start- and/or end-point are changed (see Fig. 4 D, E), whereas when $\alpha_w \neq 1$ we obtain a smaller or larger position profile (see Fig. 4 F, G; similar to scaling proposed in [8]).

As already mentioned in the Introduction section, several methods exist for trajectory generation for example those based on splines, dynamic movement primitives (DMPs), or Gaussian Mixture Models (GMMs). In the following paragraph we will discuss the behavior of different types of DMPs for some specific cases and at the end compare those also to results from the literature obtained with GMMs. Differences will emerge, but before discussing those, it is important to note that the now following comparison should not be used to deduce quality assessments of the different methods. Many times the actual quality of a method depends critically on the task and a method that performs badly under some conditions can be superior under different ones. Thus, fair comparisons are difficult if not impossible. Still we think that the behavior displayed in Fig. 10 might be helpful to appreciate and understand the similarities and differences of the shown methods.

Original DMPs are designed in such a way that they are not directly dependent on time, which allows adding coupling terms or phase resetting techniques [8]. By replacing time with a phase variable one can manipulate the time evolution of phase, i.e. by slowing down or speeding up the movement as appropriate. One example of such a coupling is *phase stopping* which prevents the actual trajectory from moving too far away in case of a perturbation [8]. In Fig. 10 A1 and A2 we show the behavior of original DMPs in case of a perturbation without (see Eq. 4) and with phase stopping (see Eq. 7), respectively. Indeed, we can see that DMPs with phase stopping behave differently compared to DMPs without phase stopping by pushing the trajectory more towards the perturbation point. Note, this example was specifically chosen to show this particular effect and the response characteristics of the system might be to some degree different for different perturbations at other times. The effect of phase stopping in this example is significant but not very strong and more time will be required to allow for this compensation to happen, i.e., this makes the time of the movement execution longer. We

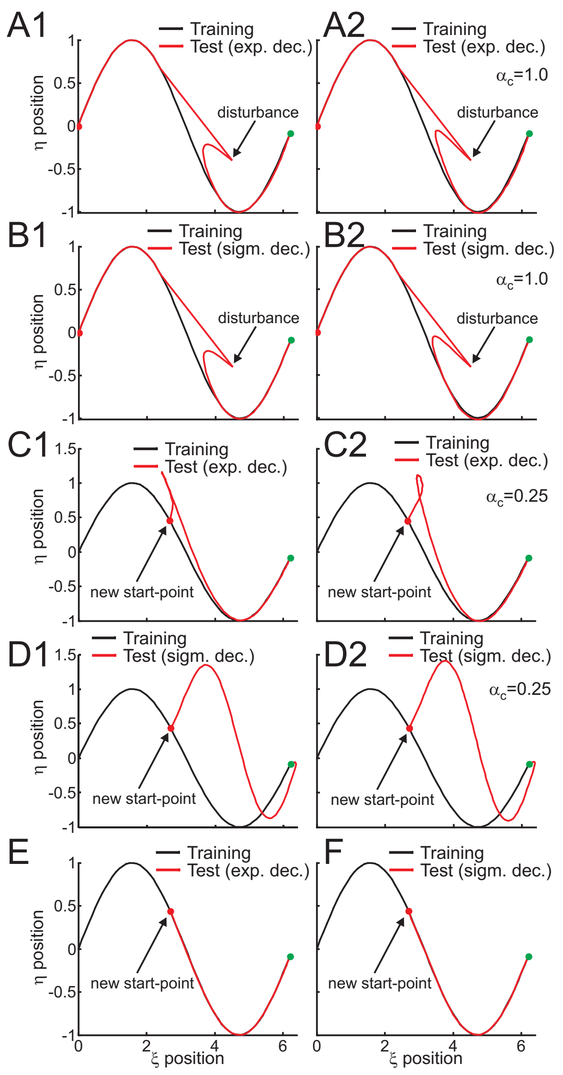


Fig. 10. Comparison between **A, C, E)** original (exponential decay system) and **B, D, F)** modified (sigmoidal decay system) DMPs. In panels **A-D** on the left side we show results without phase stopping and on the right side with phase stopping (see Eq. 7 and Eq. 22). **A, B)** Performance of different DMP systems when a disturbance is applied. We disturbed the system by $\Delta y_\xi(t) = \Delta y_\xi(t) + 2$ and $\Delta y_\eta(t) = \Delta y_\eta(t) - 1$ at the time moment $t = 0.4\,T$). **C-F)** Performance when the start-point is changed, where in case **E)** a phase-cut and in case **F)** a time-cut was applied at the new start point. Note that in cases C1, C2 and E we used Eq. 5 without scaling $(g - y_0)$ in order to make it comparable to modified DMPs. Other system parameters were as given in the caption of Fig. 6, only in this case we used 30 kernels with width $\sigma = 0.025$.

can also extend our approach by adding the phase variable in order to make it not directly dependent on time and obtain phase stopping. This can be done by the following equations. Let us add an additional phase variable

$$\tau\dot{\theta} = \frac{\Delta_t}{1 + \alpha_c(y_{desired} - y_{actual})^2}, \tag{22}$$

with $\theta_0 = 0$ and replace Eq. 8 by

$$\dot{r} = \begin{cases} \dot{\theta}\,(g - s)/(T), & \text{if } \theta \le T \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

In doing this, we also need to replace everywhere the time variable t with the phase variable θ. As demonstrated in Fig. 10 B1 and B2 we can see that the modified DMPs behave similarly to the original method.

In a second set of experiments, we compared the behavior of the systems when the start-point is moved further away along the trajectory as compared to the training trajectory. With this we want to compare generalization properties of the different systems. Results of such experiments are shown in Fig. 10 C-F where a different behavior is observed for the original DMPs (with and without phase stopping) as compared to the modified DMPs (compare panels C1, C2 and D1, D2). In case C1 the system first of all tries to go back to the originally trained start-point and only later continues to follow the target (training) trajectory, whereas phase stopping (C2) prevents the system from going backwards. The behavior of our modified DMPs is presented in panel D1, D2 where a squeezing effect of the original trajectory is observed in both cases. Here the phase stopping has no effect on the trajectory since our modified system follows the new (squeezed) trajectory and the phase stopping caused by the deviation from the training trajectory only slows down the motion process but does not change the shape of the trajectory. Instead of backtracking or squeezing (C, D) one can also easily obtain a behavior where the trajectory just begins at the shifted start-point (see panels E and F). For this, one needs to set appropriate phase (E) or time (F) at the corresponding position.

Another type of attractor-based trajectory generators are Gaussian Mixture Models (GMMs, [3], [24]). In GMMs a movement is described by a nonlinear time-invariant dynamical system, where the convergence to the end-point of all trajectories is ensured starting from any point in the target space forming a trajectory-embedding surface which is obtained by learning from a set of demonstrations (see Fig. 10 in [24]). The behavior of GMMs in case of perturbations much depends on the obtained trajectory-embedding surface, which may vary depending on the used learning method. So, in case of a perturbation a GMM might first of all come back to the target (training) trajectory and later continue along the training trajectory or go directly from perturbation point to end-point. In case the start-point is moved further away along the trajectory as compared to the training trajectory the GMM would continue following the target trajectory directly from the shifted start-point (similar to the DMP behavior in Fig. 10 E, F), since GMMs are time-invariant. Clearly, choosing a system for movement generation much depends on the task and on which type of behavior is desired. For example, whether it is important to follow the trajectory or reach the goal at a specified time, continue following the trajectory from the new start-point or return to the trained start-point, and so on.

Several methods exist for the imitation learning of a target trajectory, like locally weighted regression methods [4], [8], global regression methods [14] or Gaussian Mixture Regression [25]. For instance, by using locally weighted regression one can learn not only the appropriate weights w_i of the kernels but also the number of kernels n, their centers c_i and widths h_i, automatically. Here we used a simpler learning rule to train weights w_i, known as delta learning rule in artificial neural networks [23], which is not optimal for a single training sample (regression methods would succeed in one-shot learning in this case). On the other hand, in our case position information obtained from demonstration is enough for determining the DMP weights. We do not have to measure acceleration or velocity (or obtain these by calculating derivatives) as required for regression methods in DMP weight learning procedures [8]. Although, here we adapt only the weights of the kernels, while leaving their centers and widths fixed, we can still achieve high learning accuracy. In general we obtained that for trajectories with relatively slow dynamics, like letters, one needs fewer but wider kernels to accurately approximate the trajectory, whereas for relatively complex trajectories with fast dynamics, like signatures, more but narrower kernels are required. Similar to other methods, such a learning procedure would lead to an average trajectory when different training samples are used. Clearly, our simple learning method could be replaced by any of the more advanced approaches (e.g., regression methods), which might be advisable in cases of single training samples or where the dynamics of the trajectory changes strongly ”along the way”.

Our approach assumes a single reference trajectory and it can generalize quite well for points close enough to this trajectory. Generalization, however, for example drops when motion starts/ends at points too far away from this trajectory. For robot tasks that require generalization over a larger domain, one might choose other learning alternatives, such as GMM’s which forms a trajectory-embedding surface from a set of demonstrations [3], [24]. Also, in our setup we learnt weights along each dimension separately. However, such an assumption of independent dimensions is not generally true for many motions, especially in case of perturbations. Depending on the application, one might use other methods, like *Stable Estimator of Dynamical Systems* [24], which is a single multi-dimensional model and considers correlations across several dimensions.

A classical way to join several trajectories are spline based solutions [1], [2] which allow joining trajectories smoothly with predefined velocity and/or acceleration. This approach, however, is not an on-line trajectory generator, and lacks attractive features of DMPs, like robustness to perturbations and generalization. Koga et al. [26] proposed a motion planner for object manipulation which allows sequencing of actions. In their approach complex motion is divided into *transit paths* (e.g., approaching an object) that do not move objects and *transfer paths* that move objects (e.g., grasping and carrying an object). A manipulation is performed by concatenating alternating sequences of such paths that connect the initial system’s configuration to a final (goal) configuration. This is a kinematic approach where the solution is based on the configuration of manipulators and objects and, different from DMPs, does not consider the dynamics of the movement, i.e., velocities and accelerations.

DMPs have been applied for different robotics applications, like movement with obstacle avoidance [10], [11], lifting [13], pouring [12], [14], hitting and batting [9], [15], [18], flight control [7], drumming [17], [27], walking [5], [20] and jumping [16]. However, not so much work has been done with respect to joining of dynamic movement primitives [7], [14], [15]. So, in this study we were mainly concerned with the problem of joining several DMPs. This was motivated by a drawback of the original DMP system, namely, its velocity profile which produces a relatively long and shallow decay until it reaches the end-point. This means that the system has to stop (to reach zero velocity) before a new DMP can be started. To avoid jumps in velocity and acceleration, a simple solution for this problem, as demonstrated in [14], would be to apply a first order low-pass filter (augment original DMPs to a third order system). However, if positions and velocities of both DMPs are at the joining point too different from each other, one would need to smooth the trajectory a lot, which, as a consequence produces large deviations from the target trajectory due to the delay produced by the first order filter. Another approach for joining movement sequences based on a modification of the original DMPs is introduced by Kober et al. [15] for learning of hitting and batting. In this case the original DMPs are augmented by introducing two additional terms, namely, moving target and final boundary velocity. Such a method allows joining several DMPs at a specific position with a predefined velocity. Perk et al. [7] proposed a method to get smooth transitions when joining several DMPs based on partial contraction theory [28]. In their approach the trajectory of one DMP is forced to converge to the other leading to a smooth transition between two DMPs. In contrast to the presented methods, in our approach we do not augment the system by the additional equations or terms but simply create one joint DMP by overlapping kernels of several single DMPs. This way, a smooth, natural and accurate transition, as exemplified on handwriting, emerges from the system itself without a deceleration at the joining point (see Fig. 3). This method inherits the properties of the modified DMPs such as scalability and generalization, like stretching and squeezing, when start- and/or end-points are changed as shown in Fig. 9 H. In this case one only needs to appropriately scale start- and end-points (s_j and g_j, see Eq. 21) at the joining points. The trajectory around the joining point is dependent on the overlap of kernels (see Fig. 7 B-E) and some tuning would be required in cases where high accuracies at the joining point are required.

We believe that the attractive properties of our joining method, like scalability, generalization and smoothness, have some potential for different kind of robotics applications. Using robots to actually forge signatures will however (luckily!) still remain only a possible topic for entertaining science fiction. After all, it is unlikely that human-like (”android”) machines will exist in the next 50 or more years, with an appearance that could deceive a customer clerk and let him/her forget that actually a robot hand does the writing.

References

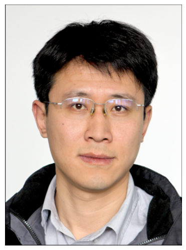
[1] R. H. Castain and R. P. Paul, “An on-line dynamic trajectory generator,” *Int. J. Robot. Res.*, vol. 3, no. 1, pp. 68–72, 1984.

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and control*. Springer Publishing Company, 2009.

[3] S. M. Khansari-Zadeh and A. Billard, “BM: An iterative method to learn stable non-linear dynamical systems with gaussian mixture models,” in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2381–2388.

[4] J. A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. 2002 IEEE Int. Conf. Robotics and Automation*, 2002, pp. 1398–1403.
[5] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robot. Auton. Syst.*, vol. 47, no. 2-3, pp. 79–91, 2004.
[6] J.-X. Xu, W. Wang, P. Vadakkepat, and L. W. Yee, "ANN based internal model approach to motor learning for humanoid robot," in *Proc. 2006 Int. Joint Conf. Neural Networks*, 2006, pp. 4179–4186.
[7] B. E. Perk and J. J. E. Slotine, "Motion primitives for robotic flight control," Tech. Rep. arXiv:cs/0609140v2, Sep. 2008.
[8] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control–a unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.
[9] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, no. 4, pp. 682–697, 2008.
[10] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. 8th IEEE-RAS Int. Conf. Humanoid Robots*, 2008, pp. 91–98.
[11] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *Proc. 2009 IEEE Int. Conf. Robotics and Automation*, 2009, pp. 1534–1539.
[12] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. 2009 IEEE Int. Conf. Robotics and Automation*, 2009, pp. 763–768.
[13] S. Bitzer and S. Vijayakumar, "Latent spaces for dynamic movement primitives," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, 2009, pp. 574–581.
[14] B. Nemec, M. Tamosiunaite, F. Woergoetter, and A. Ude, "Task adaptation through exploration and action sequencing," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, 2009, pp. 610–616.
[15] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, 2010, pp. 1–6.
[16] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2397–2403.
[17] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robot.*, vol. 26, pp. 800–815, Sep. 2010.
[18] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Proc. 15th Advances in Neural Information Processing Systems*, 2003.
[19] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robots," in *Proc. 2003 Int. Symp. Adaptive Motion of Animals and Machines*, 2003.
[20] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *Workshop Bilateral Paradigms on Humans and Humanoids, 2003 IEEE Int. Conf. Intelligent Robots and Systems*, 2003.
[21] A. I. Al-Shoshan, "Handwritten signature verification using image invariants and dynamic features," in *Proc. 2006 Int. Conf. Computer Graphics, Imaging and Visualisation*, 2006, pp. 173–176.
[22] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Proc. 2003 Int. Symp. Robotics Research*, 2004, pp. 561–572.
[23] S. Haykin, *Neural networks: A comprehensive foundation*. Prentice Hall, 1999.
[24] S. M. Khansari-Zadeh and A. Billard, "Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming," in *Proc. 2010 IEEE Int. Conf. Intelligent Robots and Systems*, 2010, pp. 2676–2683.
[25] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Trans. Syst. Man and Cybern. B*, pp. 286–298, Apr. 2007.
[26] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, "Planning motions with intentions," in *Proc. 21st Ann. Conf. Computer Graphics and Interactive Techniques*, 1994, pp. 395–408.
[27] S. Schaal, "Movement planning and imitation by shaping nonlinear attractors," in *12th Yale Workshop Adaptive and Learning Systems*, 2003.
[28] W. Wang and J. J. Slotine, "On partial contraction analysis for coupled nonlinear oscillators," *Biol. Cybern.*, vol. 92, no. 1, pp. 38–53, 2005.

Tomas Kulvicius received his M.S. degree in Computer Science (2003) from Vytautas Magnus University, Kaunas, Lithuania. He received the Ph.D. degree in Computer Science (2010) from the University of Göttingen, Germany. In his Phd thesis he investigated development of receptive fields in closed loop learning systems. Currently he is a Postdoctoral Researcher at the Department for Computational Neuroscience in the University of Göttingen. His research interests include closed loop behavioural systems, learning algorithms, receptive fields, robotics, dynamic movement primitives, biosignal analysis, biological system modelling.



KeJun Ning received the B.S. (Hons.) and M.S. (Hons.) degrees in mechanical engineering from the Northeastern University, Shenyang, China, in 1999 and 2002, respectively, and the Ph.D. degree in mechatronics engineering from Shanghai Jiao Tong University, Shanghai, China, in 2006. From May 2006 to July 2007, he was a Senior Mechatronic System Engineer with Shanghai Grandar Robotics Co. Ltd. From August 2007 to May 2008, he was a Senior Engineer and Researcher with the Robotics Department, China Corporate Research Center (CNCRC), Shanghai Branch, ABB (China) Ltd., China. He is currently a Postdoctoral Researcher with the Bernstein Center for Computational Neuroscience, University of Göttingen, Germany. His research interests include the areas of robotics, mechanism design, embedded system, and autonomous systems.

Minija Tamosiunaite has received a Ph.D. in Informatics in Vytautas Magnus University, Lithuania, in 1997. Currently she works as a senior researcher at the Bernstein Center for Computational Neuroscience, Inst. Physics 3, University of Göttingen.Her research interests include machine learning, biological signal analysis, and application of learning methods in robotics.

Florentin Wörgötter has studied biology and mathematics at the University of Düsseldorf, Germany. He received his PhD for work on the visual cortex from the University of Essen, Germany, in 1988. From 1988 to 1990, he was engaged in computational studies with the California Institute of Technology, Pasadena. Between 1990 and 2000, he was a Researcher at the University of Bochum, Germany where he was investigating the experimental and computational neuroscience of the visual system. From 2000 to 2005, he was a Professor of computational neuroscience with the Psychology Department, University of Stirling, U.K., where his interests strongly turned towards "Learning in Neurons". Since July 2005, he has been the Head of the Computational Neuroscience Department at the Bernstein Center for Computational Neuroscience, Inst. Physics 3, University of Göttingen, Germany. His current research interests include information processing in closed-loop perception-action systems, sensory processing, motor control, and learning/plasticity, which are tested in different robotic implementations. His group has also developed the RunBot, which is a fast and adaptive biped-walking robot.