Modified dynamic movement primitives for joining movement sequences

Tomas Kulvicius, KeJun Ning, Minija Tamosiunaite and Florentin Wörgötter
Georg-August-Universität Göttingen
Bernstein Center for Computational Neuroscience
Department for Computational Neuroscience
III Physikalisches Institut - Biophysik
Friedrich-Hund Platz 1, DE-37077 Göttingen, Germany
Email: {tomas; ning; minija; worgott}@physik3.gwdg.de

***Abstract*—The generation of complex movement patterns, in particular in cases where one needs to smoothly and accurately join trajectories, is still a difficult problem in robotics. This paper presents a novel approach for joining of several dynamic movement primitives (DMPs) based on a modification of the original formulation for DMPs. The new method 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 implemented 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 trajectory learning and generation and its accuracy and modular character has potential for various robotics applications.**

I. Introduction

Recently, dynamic movement primitives (DMPs) have become very popular for motor control in robotics and led to numerous studies [1]–[15]. DMPs are units of actions which describe a particular movement trajectory and are formalized as stable nonlinear attractor systems [16]–[19]. Such movement primitives can be used to generate a movement trajectory either in joint- or task-space. DMPs have advantages over other trajectory generators, like splines [20], [21], for several reasons: 1) robustness to perturbations, 2) ability to generalize, and 3) ability to apply learning. However, the original DMPs [1], [5], [16], [17] have some drawbacks, too. There is a trade-off between path and end-point control of the trajectory. Briefly: accurate path following will lead to inaccurate endpoint and vice versa. There exist several approaches to address this problem [7], [8]. Here we propose a simple alternative way to modify the original DMPs [1], [5] for solving this trade-off problem and show that it's possible to have control over the full movement trajectory while at the same time guaranteeing convergence to the end-point.

DMPs have been applied in different robotics applications, like movement with obstacle avoidance [7], [8], lifting [10], pouring [9], [11], [15], hitting and batting [6], [12], [17], flight control [4], drumming [14], [22], walking [2], [19] and jumping [13]. However, not so much work has been done with respect to joining of dynamic movement primitives [4], [11]. Another drawback of the original DMPs is that usually their velocity profile has a long and shallow tail which is inappropriate for obtaining natural speed profile of composite movements when joining several dynamic movement primitives. This, however, is needed to reproduce handwriting composed from single letters as well as in many other cases in robotics.

In the following we will first describe the formalism for DMP modification and our novel joining approach. Then we will compare two joining approaches. Finally, we will conclude our study with a Discussion section where we will compare our approach to state-of-the-art methods.

II. Methods

A. Modification of original DMPs

The original DMP system is formalized by second order differential equations and consists of two dynamic systems: the transformation system and the canonical [1], [16], [17] or exponential decay system [5]. Here we will use the exponential decay system since it is more closely related 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 exponential decay system is described by

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

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

$$f = \frac{\sum\limits_{i=1}^{n} \psi_i \, w_i \, v}{\sum_i \psi_i}, \tag{5}$$

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. For more details see [1], [5].

To solve the above discussed trade-off problem of the original DMPs we modify the system by changing the delayed goal function r and the exponential decay function v (Eq. 3, 4, see Fig. 1). The exponential goal function r is replaced by a piecewise-linear function and formalized as follows:

$$\tau \dot{r} = \begin{cases} (g-s)/(T-\Delta_t), & \text{if } t \leq T \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Here g and s 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 (here we used $\Delta_t = 200\,ms$).

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

$$\dot{v} = \left(1 + e^{[\alpha_v^s(\tau T - t)]^2}\right)^{-1} \alpha_v^s \, e^{\alpha_v^s(\tau T - t)}. \tag{8}$$

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

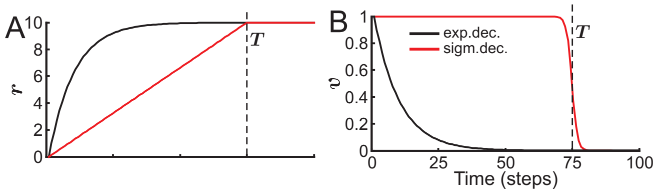

Fig. 1. Modification of the original DMP system. **A**) Goal functions r and **B**) scaling functions v are shown for the exponential decay system (original DMPs) and the sigmoidal decay system (modified DMPs). The following parameters were used: $g = 10$, $\alpha_g = 0.1$, $T = 75$, $\alpha_v^e = 0.1$ and $\alpha_v^s = 1.0$. For more details see section II-A.

We also use a slightly different nonlinear function f:

$$f = \alpha_w \frac{\sum_{i=1}^{n} \psi_i \, w_i \, v}{\sum_i \psi_i}, \tag{9}$$

$$\psi_i = e^{-(\frac{t}{\tau T} - c_i)^2 / 2\sigma_i^2}, \tag{10}$$

where σ_i is the width of the $i-th$ kernel. Kernels are placed evenly along the trajectory in time and normalized 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 α_w is the scaling factor for the learned weights (note that during learning we always set $\alpha_w = 1$). In case of $\alpha_w = 1$ we obtain a squeezing or stretching effect compared to the original trajectory when different start- and/or end-points are chosen. If we use a scaling factor $\alpha_w \neq 1$ then we get a scaling of the original trajectory which leads either to a smaller ($\alpha_w < 1$) or bigger ($\alpha_w > 1$) trajectory. Clearly, choosing α_w (with scaling or without) would depend on the particular application and further in this study we will always use $\alpha_w = 1$.

With such a system we assure that the end-point g of the movement trajectory y will be reached in time $T + \xi$ ($\xi \ll T$) with a given accuracy $|g - y(t)| \leq \varepsilon$, where ξ depends on the steepness of the sigmoidal function (Eq. 8) defined by the parameter α_v^s. Note that in the original DMP system the exponential function (see Eq. 4) acts as a phase as well as a weight scaling variable, whereas in our modified DMP system the sigmoidal function (Eq. 8) acts only as a weight scaling variable.

B. Trajectory learning

The shape of a trajectory in the DMP system is parametrized 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 [16] or global regression methods [11]. We propose a simpler learning rule to train weights w_i. Thus, here we use 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. In our case this can be formalized by the following equation:

$$\Delta w_i^j = \mu \, [\gamma(k) - y^j(k)], \tag{11}$$

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. Note that in case we have more than one dimension 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.

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, couplets (e.g. "ab") and whole words. Data were obtained by using a pen tablet (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 (training) trajectory. We calculated the position deviation by

$$d_{pos} = \sqrt{(y_{x_t} - y_{x_l})^2 + (y_{y_t} - y_{y_l})^2} \tag{12}$$

and the velocity deviation as

$$vel_{t,l} = \sqrt{\dot{y}_{x_{t/l}}^2 + \dot{y}_{y_{t/l}}^2}, \tag{13}$$

$$d_{vel} = |vel_t - vel_l|, \tag{14}$$

where $y_{x/y}$ and $\dot{y}_{x/y}$ are position and velocity for x and y profiles, and indices t and l stand for the target and learnt 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 start the next DMP at this 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 drawbacks due to the trade-off problem of the original DMPs. As we will see later, either we get a slow down (around zero velocity at the joining point) or we do not reach the-end point of the first DMP with good enough accuracy for joining. Here we propose a novel method for DMP joining which can solve these problems. 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:

$$c_j^i = \begin{cases} \frac{T_1\,(i-1)}{T'\,(n-1)}, & \text{if } j=1 \\ \frac{T_j\,(i-1)}{T'\,(n-1)} + \frac{1}{T'}\sum\limits_{k=1}^{j-1} T_k, & \text{otherwise,} \end{cases} \tag{15}$$

$$c' = c_1^1, c_1^2, \ldots, c_1^n; c_2^1, c_2^2, \ldots, c_2^n; \ldots; c_m^1, c_m^2, \ldots, c_m^n. \tag{16}$$

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

$$\overline{\sigma}_j^i = \frac{\sigma_j^i T_j}{T'}, \tag{17}$$

$$\sigma' = \overline{\sigma}_1^1, \overline{\sigma}_1^2, \ldots, \overline{\sigma}_1^n; \overline{\sigma}_2^1, \overline{\sigma}_2^2, \ldots, \overline{\sigma}_2^n; \ldots; \overline{\sigma}_m^1, \overline{\sigma}_m^2, \ldots, \overline{\sigma}_m^n, \tag{18}$$

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:

$$w' = w_1^1, w_1^2, \ldots, w_1^n; w_2^1, w_2^2, \ldots, w_2^n; \ldots; w_m^1, w_m^2, \ldots, w_m^n. \tag{19}$$

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

$$\tau\dot{r}' = \begin{cases} (g_j - s_j)/(T_j - \Delta_t), & \text{if } \sum\limits_{k=1}^{j-1} T_k \leq t \leq \sum\limits_{k=1}^{j} T_k \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

where j=1…m. We use the same equation for the sigmoidal decay function as given in Eq. 8, 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. 2, where we show signals obtained from joining two letters "a" and "b" (see also Fig.3 C).

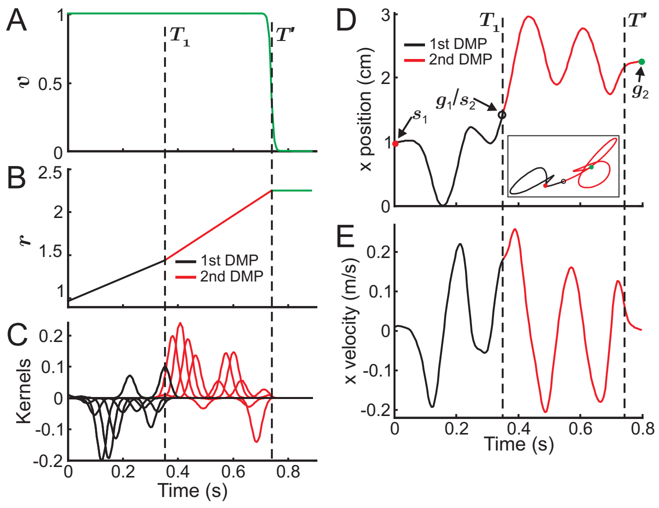

Fig. 2. Illustration of the DMP joining method with overlapping kernels. Here we show only signals for the x position of joining two letters "a" and "b". **A)** Sigmoidal decay function v, **B)** goal function r, **C)** kernels ($\psi_i' w_i' v$), **D)** x position (y_x); in the inset we show the trajectory of the joined "a" and "b", and **E)** x velocity ($\dot{y}_x$).

III. Results

A. Joining of several 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. 3. 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. Results for the simple joining method are presented in Fig. 3 A, B where in panel A we show results for steep exponential function ($\alpha_v^s = 0.1$) and in panel B for shallow one ($\alpha_v^s = 0.02$). Here we can see that in case of a large exponent we are reaching the joining point (see panel A, left), however, at this point the system's velocity is almost zero (see panel A, right), whereas in human handwriting velocity at the junction is relatively high. If we use a small exponent then velocity does not drop to zero (B, right), however, here we do not reach the target joining point (B, left). These drawbacks are inherited from the behavior of the original DMP system. Results for the joining method with overlapping kernels based on modified DPMs are shown in panel Fig. 3 C, where, in contrast to the simple joining method, we obtain high accuracy at the junction point with respect to both, position and velocity, profiles. Better precision could be obtained if one would use more kernels. Note that joining with overlapping kernels based on original DMPs would not improve the results much due to the suppressed influence of the kernels at the junction point.

B. Statistical evaluation

We have also performed statistical evaluation of our joining method and compared it to the corresponding statistical

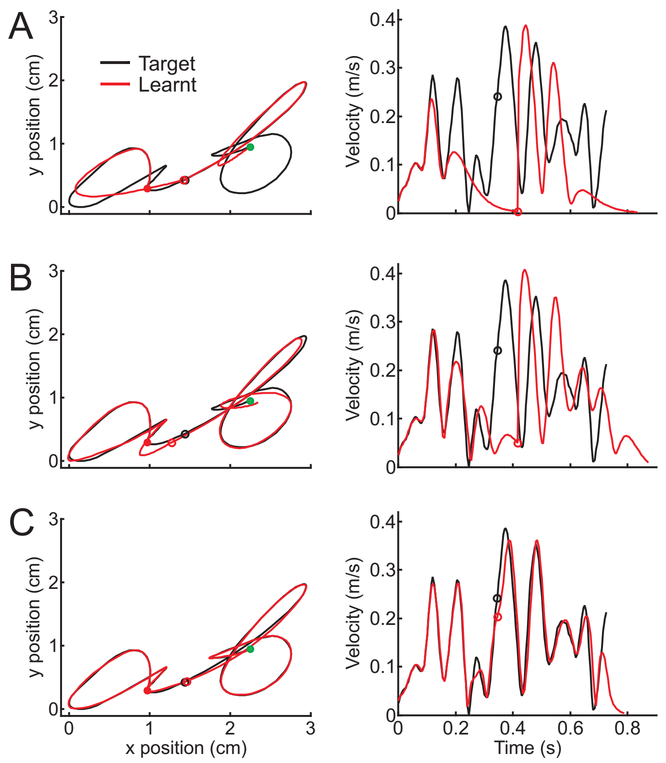

Fig. 3. Comparison between simple DMP joining (exponential decay system) and joining with overlapping kernels (sigmoidal decay system). **A)** Results for the simple DMP joining when a large exponent is used ($\alpha_v^e = 0.1$) and **B)** when a small exponent is used ($\alpha_v^e = 0.02$). **C)** Results for the joining method with overlapping kernels. On the left side we show position profiles, whereas corresponding velocity profiles are shown on the right side. Red and green dots represent start- and end-point of the movement trajectory, respectively, whereas circles represent junction points between two DMPs. The following system parameters were used for both systems: number of kernels $n = 15$, width of kernels $h_i = 150$ (exp. dec.) and $\sigma_i = 0.05$ (sigm.dec.), $i = 1, \ldots, n$, 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.

evaluations of joining the original DMPs. For the statistics we used 20 different couplets as shown in Fig. 4 A and the same procedures as explained in section III-A. Here we compare position (d_{pos}) and velocity deviation (d_{vel}) of joint DMPs at the junction point from the target junction point (denoted by the red dot). Results of such an experiment are shown in Fig. 4 B, C. We can see that the simple joining method based on original DMPs with relatively steep exponential function ($\alpha_v^e = 0.1$), as well as the joining method with overlapping kernels based on modified DMPs, produce significantly less deviation from the target junction point as compared to the case when a relatively shallow exponential function ($\alpha_v^e = 0.02$) is used (Fig. 4 B). As already presented in Fig. 3, we also find that the proposed joining method is significantly more accurate with respect to the velocity at the junction point than the simple joining approach (Fig. 4 C).

C. Handwriting generation

Finally, we applied the novel DMP joining method based on modified DMPs to the problem of generating handwriting by a robot. For this we trained the system on separately written letters as shown in Fig. 5 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 two words: "dmp" and "demo". Position and corresponding velocity profiles generated by the system for these words are shown in Fig. 5 B, D and C, E, where we can observe that connections between letters are smooth and natural. This handwriting generation was also implemented on a simple robotic manipulator platform (Neuro-Robotics, Sussex) in our laboratory and the resulting trajectories[1] are shown in Fig. 5 F, G. Note that we scaled the output of the system in space and time in order to implement it on the robotic platform.

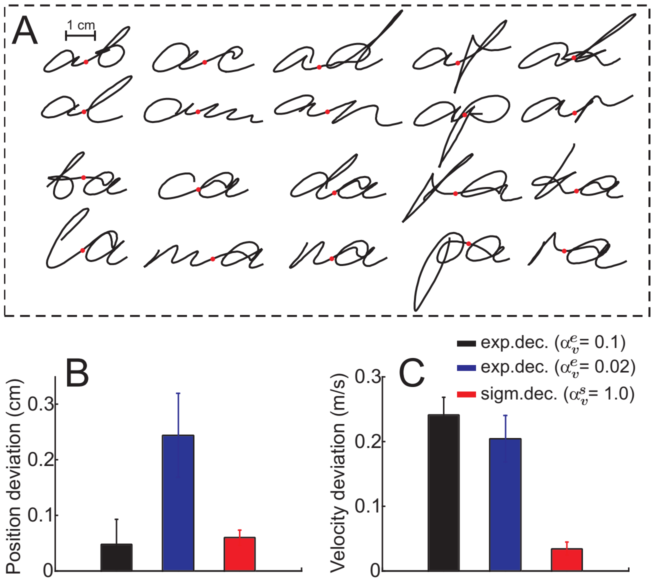

Fig. 4. Statistical comparison between joining using the simple method based on original DMPs and the joining method with overlapping kernels based on modified DMPs. **A)** 20 couplets used for the statistical evaluation are shown in panels B, C. Here red dots denote the junction points. **B)** Average position deviation d_{pos} and **C)** 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 legend of Fig. 3.

IV. Discussion

In this study we presented a novel method for joining of several dynamic DMPs based on a modification of the original DMP formulation [1], [5], [16]. The new method was applied to handwriting generation and also implemented on a robot. Furthermore, we used standard delta-rule learning to adapt the weights. These three components are all quite simple and efficient. We have also statistically compared our system against the simple joining with 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

[1]Videos of these experiments can be downloaded at http://sites.google.com/site/ktomsite/handwriting-with-dmps

was just chosen as a demonstration scenario and we are neither interested nor will discuss details and implications of handwriting biometry.

As already discussed above, original DMPs suffer from a trade-off problem: either one can achieve good control along the trajectory but then one will not reach the end-point of the desired movement well enough or vice versa. Another drawback of the original DMP system is the velocity profile, which produces a long and shallow decay before reaching the end-point. This means that the system has to stop (to reach zero velocity) before a new DMP can be started, which is not appropriate for joining several DMPs. Therefore, first of all we proposed a modification of the original DMP formulation in order to solve the trade-off problem. Here we made two modifications: 1) instead of the delayed goal function r we used a piece-wise linear goal function (see Eq. 3, 20) 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, 8). Such 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. Here the steepness of the sigmoid sets the limits. Note that use of the linear goal function r together with kernels ψ which are independent of v (see Eq. 6, 10) allows better generalization, especially in cases when start and goal positions are relatively far away compared to the training trajectory. We also would like to stress that our system is easily scalable with respect to the dynamics of the system (τ), the time (T) and space without losing the qualitative trajectory appearance that was originally coded in 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. 9), we get a stretching or squeezing effect of the learned trajectory when start- and/or end-point are changed, whereas when $\alpha_w \neq 1$ we obtain a smaller or larger position profile.

Several methods exist for imitation learning in order to learn a target trajectory, like locally weighted regression methods [5], [16] or global regression methods [11]. 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 proposed a simpler learning rule to train weights w_i, known as delta learning rule in artificial neural networks [23]. Although, here we adapt only the weights of the kernels, while leaving their centers and widths fixed, we can still achieve very 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. Clearly, our simple learning method could be replaced by any of the more advanced approaches, which might be advisable in case where the dynamics of the trajectory changes strongly "along the way".

There are a few approaches existing for joining of dynamic

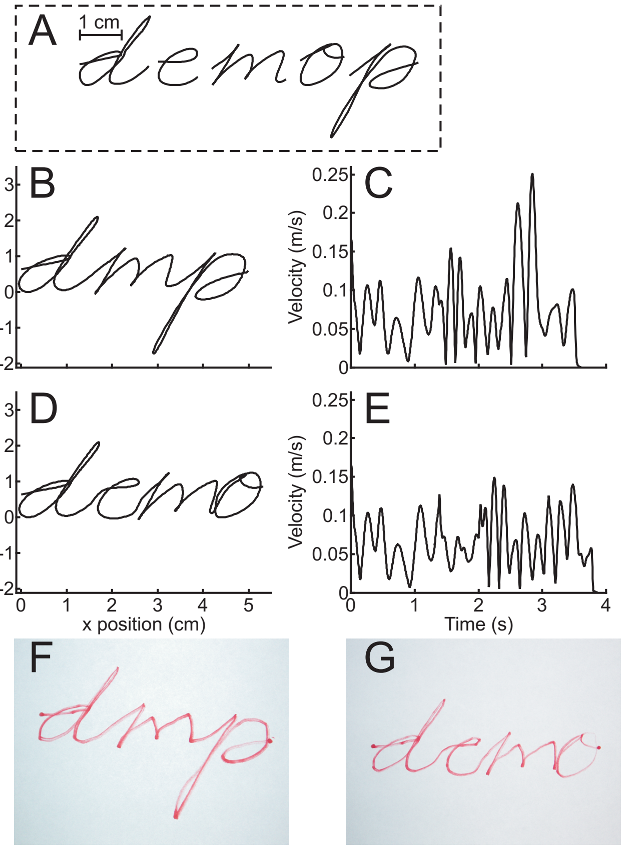

Fig. 5. Results of handwriting generation using modified DMPs (sigmoidal decay system). **A)** Samples of single letters used to generate the words "dmp" (panels B, C and F) and "demo" (panels D, E and G). **B, D)** Position profiles and **C, E)** corresponding velocity profiles generated by the modified DMP system. **F, G)** Trajectories of handwriting produced by the robot. The same system parameters were used as given in the legend of Fig. 3.

movement primitives. A simple solution to avoid jumps in velocity and acceleration when joining two DMPs, as demonstrated by [11], 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 at the joining point are 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 method to get smooth transitions when joining several DMPs is presented by [4]. These authors use partial contraction theory [24] to force the trajectory of one DMP to converge to the other. In contrast to the presented methods, in our approach we do not augment the system by the additional equations but simply create one joint DMP by overlapping kernels of several single DMPs. This way, a smooth and natural transition, as shown by handwriting generation, emerges from the system itself without a deceleration at the joining point (see Fig. 3 C). This method inherits the properties of the modified DMPs such as scalability and generalization, like stretching and squeezing, when start- and end-points are changed. In this case one only needs to appropriately scale time points T_i

(see Eq. 20). A drawback of our approach is that the trajectory around the joining point will be slightly transformed and less accurate compared to the target (training) trajectory due to the overlap of kernels (see Fig. 3 C). To increase the accuracy around the joining points one can use more but narrower Gaussian kernels.

In summary, we presented a novel and easy to implement method for joining several DMPs based on modified DMPs and demonstrated that such method results in smooth and natural transitions. We believe that attractive properties of our model, like scalability, generalization and smoothness, have some potential for different kind of robotics applications.

Acknowledgements

This research was supported by the European funded PACO-PLUS project as well as 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.

References

[1] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *International Symposium on Robotics Research (ISRR 2003)*, 2005, pp. 561–572.

[2] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robotics and Autonomous Systems*, no. 2-3, pp. 79–91, 2004.

[3] J.-X. Xu, W. Wang, P. Vadakkepat, and L. W. Yee, "ANN based internal model approach to motor learning for humanoid robot," in *International Joint Conference on Neural Networks (IJCNN 2006)*, 2006, pp. 4179–4186.

[4] B. E. Perk and J. J. E. Slotine, "Motion primitives for robotic flight control," Tech. Rep. arXiv:cs/0609140v2, Sep 2008.

[5] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control–a unifying view," *Progress in Brain Research*, vol. 165, pp. 425–445, 2007.

[6] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[7] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *IEEE-RAS International Conference on Humanoid Robotics*, 2008, pp. 91–98.

[8] 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 *International Conference on Robotics and Automation (ICRA 2009)*, 2009, pp. 2587–2592.

[9] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *International Conference on Robotics and Automation (ICRA 2009)*, 2009, pp. 763–768.

[10] S. Bitzer and S. Vijayakumar, "Latent spaces for dynamic movement primitives," in *9th IEEE-RAS International Conference on Humanoid Robots*, 2009, pp. 574–581.

[11] B. Nemec, M. Tamosiunaite, F. Woergoetter, and A. Ude, "Task adaptation through exploration and action sequencing," in *9th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2009)*, 2009, pp. 610–616.

[12] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, 2010, pp. 1–6.

[13] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, 2010.

[14] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics and Automation*, vol. accepted, 2010.

[15] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour combining goal and shape learning for dynamic movement primitives," *Robotics and Autonomous Systems*, vol. submitted, 2010.

[16] J. A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *International Conference on Robotics and Automation (ICRA 2002)*, 2002, pp. 1398–1403.

[17] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2003, pp. 1547–1554.

[18] S. Schaal, "Dynamic movement primitives - a framework for motor control in humans and humanoid robots," in *The International Symposium on Adaptive Motion of Animals and Machines (AMAM 2003)*, 2003.

[19] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Control, planning, learning, and imitation with dynamic movement primitives," in *Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003.

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

[21] R. H. Castain and R. P. Paul, "An on-line dynamic trajectory generator," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 68–72, 1984.

[22] S. Schaal, "Movement planning and imitation by shaping nonlinear attractors," in *Proceedings of the 12th Yale Workshop on Adaptive and Learning Systems*, 2003.

[23] S. Haykin, *Neural Nnetworks: A comprehensive foundation*. 2nd edition, Prentice Hall, 1999.

[24] W. Wang and J. J. Slotine, "On partial contraction analysis for coupled nonlinear oscillators," *Biological Cybernetics*, vol. 92, pp. 38–53, 2005.