2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Trajectory Representation by Nonlinear Scaling of Dynamic Movement Primitives

Aleš Ude[1,2], Rok Vuga[1], Bojan Nemec[1], and Jun Morimoto[2†]

***Abstract*— An effective robot trajectory representation should encode all relevant aspects of the desired motion. For kinematic representations, this means that both the spatial course of the trajectory and its speed profile must be specified. The concept of dynamic movement primitives (DMP) provides a kinematic representation that fully specifies these two aspects of motion. They are, however, not separated from each other within the DMP representation. This can be problematic when movements with significant speed variations are compared within movement recognition and skill learning algorithms. In such comparisons it is often important to distinguish between the spatial and temporal aspects of motion. In this paper we propose a new representation based on dynamic movement primitives, where spatial and temporal aspects are well separated. We demonstrate the effectiveness of the proposed representation for statistical learning of robot skills and movement recognition and compare the performance with standard DMPs, where temporal and spatial aspects of motion are intertwined.**

I. Introduction

Nonlinear dynamic systems, i. e. nonlinear systems of differential equations, can be effective at representing kinematic control policies that create kinematic target values for robot control [1]. Among various approaches based on dynamic systems, dynamic movement primitives (DMPs) of Ijspeert et al. [2] have proven to be a popular choice due to their flexibility. They can be used to represent robot movements with complex trajectories, they include free parameters for learning and optimization, they are not explicitly dependent on time, are robust to perturbations, allow easy modulation to adjust to joint limits or speed up or slow down the movement, etc. For these reasons DMPs are widely used in robotics, including for movement imitation [3], policy search [4], [5], statistical learning from multiple demonstrations [6], [7], sharing of knowledge for efficient generalization [8], adaptation of trajectories to new endpoints [9], adaptation to the recorded sensory data [10], [11], and coupling of dual-arm movements [12], [13].

The focus of our recent work was on an extension of dynamic movement primitives for learning of speed profiles [14], [15]. Additional parameters were introduced in the proposed representation that allow for nonuniform temporal scaling of the desired trajectories. With this representation, we were able to speed up or slow down trajectories without affecting the spatial course of movement. In this paper we show that the representation proposed in [14], [15] is related to the parametrization of dynamic systems by arc length instead of time and develop a new representation, which we call arc length dynamic movement primitives (AL-DMPs). Spatial and temporal aspects of motion are well separated in AL-DMPs. We demonstrate that the newly developed representation significantly improves the power of DMPs when applied to the problem of statistical learning of robot movements and movement recognition.

Other recent research that addresses the problem of time alignment within the framework of DMPs includes the work of Englert et al. on probabilistic movement primitives [16] and Ben Amor et al. on interaction primitives [17], where multiple trajectories are time aligned using dynamic time warping [18]. On the other hand, Ewerton et al. [19] applied a probabilistic approach to solve the alignment problem. The advantage of our approach is the parametrization of movement through a natural parameter, i. e. the arc length, which provides alignment without the need for an algorithmic comparison between different trajectories.

In the rest of the paper we first introduce the concept of arc length dynamic movement primitives. Next we discuss the estimation of AL-DMPs and robot control with AL-DMPs. In Section V-B we analyze the application of AL-DMPs to the problems of statistical learning of motor skills and movement recognition. The significant advantages of AL-DMPs compared to standard DMPs have been demonstrated.

II. Trajectory Parameterization and Arc Length of a Curve

A DMP for a single degree of freedom point-to-point movement y is defined by the following set of nonlinear differential equations [2]

$$\tau \dot{z} = \alpha_z(\beta_z(g-y)-z)+F(x), \quad (1)$$

$$\tau \dot{y} = z, \quad (2)$$

where x is the phase variable, z is the scaled velocity of the movement, g is the desired final position on the movement, and F is a nonlinear forcing term. With the choice $\tau > 0$ and $\alpha_z = 4\beta_z$, the linear part of equation system (1) – (2) becomes critically damped and y, z monotonically converge to a unique attractor point at $y = g$, $z = 0$ [2]. $F(x)$ is defined as a linear combination of N nonlinear radial basis functions, which enables the robot to follow any

†The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no. 270273, Xperience, and from the Slovenian Research Agency under grant agreement no. J2-7360. It was further supported by MIC-SCOPE, by the New Energy and Industrial Technology Development Organization (NEDO), by JSPS KAKENHI JP16H06565, and by the Strategic Research Program for Brain Science from AMED.

[1]Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia.

[2]Dept. of Brain Robot Interface, ATR Computational Neuroscience Laboratories, Kyoto, Japan.

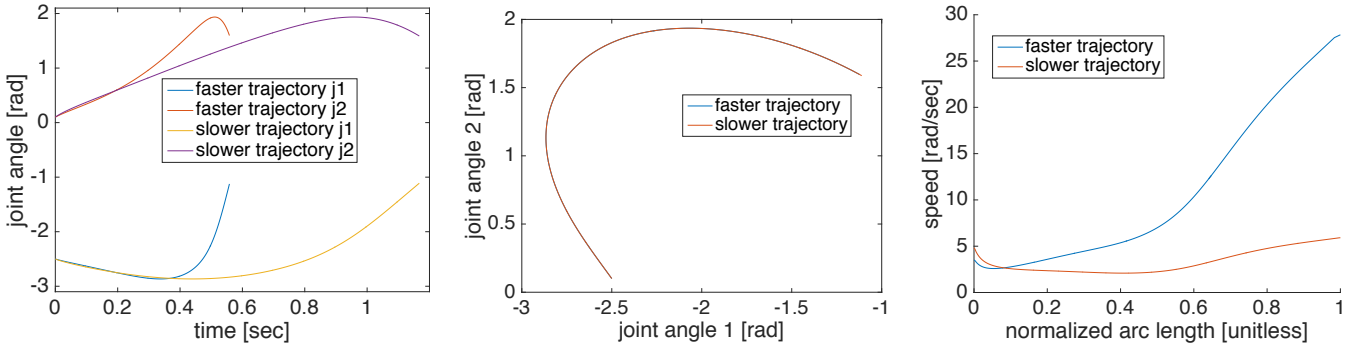


Fig. 1: Simulation results for movement estimation with AL-DMPs. The left graph shows trajectories of a simulated two degrees of freedom robot executed at two different speeds, resulting in spatially equivalent but temporally different movements. The faster movement was obtained by multiplying the speed of the slower movement by a factor of $2(t+1)e^{(t+1)^2-2},\ t \in [0,T],\ T = 0.5345$, where T is the interval of the faster movement. This is a nonlinear speed scaling that completely changes the spatial parameters $\mathbf{w}$ defining the forcing term of the standard DMP. The middle graph shows the reproduction of the spatial course of movement by AL-DMP. Parameters $\mathbf{w}$ of AL-DMPs and consequently the spatial courses of the two movements are identical up to the errors caused by numerical differentiation. The right graph shows the estimated speed $\dot{s}$ of both movements as a function of normalized arc length, i. e. s/L. It is clear that the change is nonlinear, hence it cannot be reproduced by simply changing the parameter τ in the standard DMP. These graphs demonstrate that the proposed representation effectively separates the spatial and temporal aspects of movements.

smooth trajectory from the initial position y_0 to the final configuration g

$$F(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0), \tag{3}$$

$$\Psi_i(x) = \exp\left(-h_i \left(x - c_i\right)^2\right). \tag{4}$$

c_i are the centers of Gaussians distributed along the phase of the movement and h_i their widths. The phase x has been introduced to avoid explicit time dependency. Its evolution follows the first order linear dynamics

$$\tau \dot{x} = -\alpha_x x, \tag{5}$$

where $\alpha_x > 0$ and x(0) = 1. For each degree of freedom, the weights w_i should be adjusted so that the desired behavior is achieved. To control a robot with more than one degree of freedom, the trajectory of every degree of freedom is represented with its own equation system (1) – (2), but with the common phase (5) to synchronize them.

The above formulation has been shown to have many advantageous properties for robot control [1], [6], [2]. With this representation, however, it is often difficult to compare different movement promitives between each other because Eqs. (1) – (2) contain time derivatives. Time derivatives are dependent both on speed and shape of movement and do not distinguish between the two. As it is well known from basic calculus, speed and form can be separated by parameterization with the arc length. The arc length of a time-parameterized trajectory $\mathbf{y}(t)$ is given by

$$s(t) = \int_0^t \|\dot{\mathbf{y}}(u)\| \mathrm{d}u. \tag{6}$$

The speed of movement can be calculated as the time derivative of s, which is given by

$$\dot{s}(t) = \|\dot{\mathbf{y}}(t)\|. \tag{7}$$

We will also need the length of the trajectory, which we denote here by L and can be calculated as follows

$$L = \int_0^T \|\dot{\mathbf{y}}(t)\| \mathrm{d}t, \tag{8}$$

where T is the duration of the movement.

Let's assume now that a trajectory is parameterized by the arc length instead of time. Just like the time-parameterized movements, we can approximate arc length-parameterized movements with a second order dynamic system

$$Lz' = \alpha_z(\beta_z(g - y) - z) + F(x), \tag{9}$$

$$Ly' = z, \tag{10}$$

where all derivatives are now taken not with respect to time but with respect to the arc length (this derivative is denoted by $'$). Similarly, the phase equation (5) becomes

$$Lx' = -\alpha_x x. \tag{11}$$

The time constant τ, which is used in standard DMPs to speed-up or slow-down the movement during its execution by the robot, was replaced by arc length $L > 0$, which is defined by Eq. (8). We name this new formulation *arc length dynamic movement primitive*, or *AL-DMP*. Figure 1 shows that an AL-DMP effectively separates the spatial and temporal component of motion. Just like the time-dependent equation (5), the arc length dependent phase equation has an analytical solution. It is given by

$$x(s) = \exp\left(-\alpha_x \frac{s}{L}\right). \tag{12}$$

Note that due to the replacement of τ with arc length L, the phase is always defined on the same interval regardless of the length of the trajectory. Consequently, $1 \geq x \geq \exp(-\alpha_x),\ \forall s \in [0, L]$.

III. Estimation of AL-DMPs

To estimate the parameters of DMPs, we are usually given training data in a form of a sequence of trajectory points

$$\mathcal{G} = \{\boldsymbol{y}_k, t_k\}_{k=1}^K, \boldsymbol{y}_k \in \mathbb{R}^d, \tag{13}$$

where d is the number of the robot's degrees of freedom, $t_1 = 0$, $t_K = T$. The time derivatives $\dot{\boldsymbol{y}}_k, \ddot{\boldsymbol{y}}_k$ are also needed to compute a standard DMP, but they are usually estimated numerically. To approximate these data with the equation system (9) – (11), we first sample the trajectory with respect to s. This can be achieved by calculating the arc length steps

$$s_k = \int_0^{t_k} \|\dot{\mathbf{y}}(t)\| \mathrm{d}t \sim \mathrm{Trapzd}(k), \tag{14}$$

where the trapezoidal rule was used for numerical quadrature

$$\mathrm{Trapzd}(k) = \begin{cases} \Delta t \left(\frac{1}{2}\|\dot{\boldsymbol{y}}_1\| + \sum_{n=2}^{k-1} \|\dot{\boldsymbol{y}}_n\| + \frac{1}{2}\|\dot{\boldsymbol{y}}_k\| \right), & k \geq 2 \\ 0, & k = 1 \end{cases} \tag{15}$$

to approximate the above integrals. Here we assumed that the time step is constant, i. e. $\Delta t = t_{k+1} - t_k$, $\forall k$, but this assumption can be relaxed by applying a different numerical quadrature formula. Similarly, the length of the movement can be estimated by

$$L = \int_0^T \|\dot{\mathbf{y}}(t)\| \mathrm{d}t \sim \mathrm{Trapzd}(K), \tag{16}$$

The time derivatives $\|\dot{\boldsymbol{y}}_n\|$ can be computed using standard numerical differentiation formulas, e. g.

$$\|\dot{\boldsymbol{y}}_n\| = \frac{\|\boldsymbol{y}_{n+1} - \boldsymbol{y}_n\|}{\Delta t}. \tag{17}$$

From $s_k, k = 1, \ldots, K$, and the input data $\mathcal{G}$, we can calculate y'_k and y''_k by numerical differentiation, e. g.

$$\boldsymbol{y}'_k = \frac{\boldsymbol{y}_{k+i_k} - \boldsymbol{y}_k}{s_{k+i_k} - s_k}, \ \boldsymbol{y}''_k = \frac{\boldsymbol{y}'_{k+i_k} - \boldsymbol{y}'_k}{s_{k+i_k} - s_k}. \tag{18}$$

In practice we need to make sure that the arc length step Δs is large enough for numerically stable calculation of derivatives (18) because unlike the time step, the arc length step is not uniform in the sampled trajectory. This can be done by choosing i_k to be the smallest index $|i_k|$ (positive or negative) so that $|s_{k+i_k} - s_k| \geq \delta$, where $\delta > 0$ is a constant specifying the minimum step. Otherwise the calculation of numerical derivatives can become unstable at locations with zero speed, i. e. where $\dot{s} = 0$.

We are now ready to estimate the free parameters of AL-DMPs, i. e. weights w_i of radial basis functions as specified in Eq. (3). As with standard DMPs, we first rewrite the equation system (9) – (10) as a single second-order system

$$F(x) = L^2 y'' - \alpha_z(\beta_z(g - y) - Ly'). \tag{19}$$

From these formulas we can calculate w_i by solving the following system of linear equations

$$\boldsymbol{A}\boldsymbol{w} = \boldsymbol{f}, \tag{20}$$

with

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} f_1 \\ \vdots \\ f_K \end{bmatrix}, \tag{21}$$

the $(K \times N)$ system matrix $\boldsymbol{A}$ defined as

$$\boldsymbol{A} = (g - y_0) \begin{bmatrix} \frac{\psi_1(x_1)x_1}{\sum_{j=1}^N \psi_j(x_1)} & \cdots & \frac{\psi_N(x_1)x_1}{\sum_{j=1}^N \psi_j(x_1)} \\ \vdots & \vdots & \vdots \\ \frac{\psi_1(x_K)x_K}{\sum_{j=1}^N \psi_j(x_K)} & \cdots & \frac{\psi_N(x_K)x_K}{\sum_{j=1}^N \psi_j(x_K)} \end{bmatrix},$$

and

$$f_k = L^2 y''_k - \alpha_z(\beta_z(g - y_k) - Ly'_k), \tag{22}$$

$$x_k = \exp\left(-\alpha_x \frac{s_k}{L}\right). \tag{23}$$

Here we assumed that F is defined as in (3). For a given number N of radial basis functions Ψ_i, which are defined as in (4), we calculate their parameters as follows: $c_i = \exp\left(-\alpha_x \frac{i-1}{N-1}\right)$, $h_i = \frac{2}{(c_{i+1} - c_i)^2}$, $i = 1, \ldots, N-1$, $h_N = h_{N-1}$. A set of weights that solves linear equation system (20) in a least squares sense is then calculated using

$$\boldsymbol{w} = \boldsymbol{A}^+ \boldsymbol{f}, \tag{24}$$

where $\boldsymbol{A}^+$ denotes the Moore-Penrose pseudo-inverse of the system matrix $\boldsymbol{A}$.

Eqs. (9) – (11) do not contain all information needed to control the robot. What's missing is the speed of motion. At each phase x on the trajectory $\{\boldsymbol{y}, \boldsymbol{y}', \boldsymbol{y}''\}$, we need to know also the speed, or equivalently, the time derivative of the arc length, i. e. $\dot{s}$. We use radial basis functions to approximate $\dot{s}$ at phase x

$$\dot{s}(x) = 1 + \frac{\sum_{i=1}^N v_i^s \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x. \tag{25}$$

The time derivative of s can be calculated using formula (7). Thus we obtain the following system of linear equations

$$\|\dot{\boldsymbol{y}}_k\| - 1 = \frac{\sum_{i=1}^N v_i^s \Psi_i(x_k)}{\sum_{i=1}^N \Psi_i(x_k)} x_k, \ k = 1. \ldots, K. \tag{26}$$

(26) is a standard overdetermined linear equation system in v_i^s that can be solved in a similar way as equation system (20). Note that beyond the demonstrated trajectory, $\dot{s}$ as defined by Eq. (25) converges to 1 as the phase x tends to zero. This property is important to ensure the convergence of an AL-DMP to its desired final configuration g. To enable torque control, the derivative of speed is also needed. We write

$$\ddot{s}(x) = \frac{\sum_{i=1}^N v_i^a \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x. \tag{27}$$

The data for estimating the parameters v_i^a are obtained by numerical differentiation of s as estimated by Eq. (14). Unlike $\dot{s}$ defined by (25), which converges to 1, $\ddot{s}$ defined by (27) converges to zero beyond the end of the demonstrated trajectory.

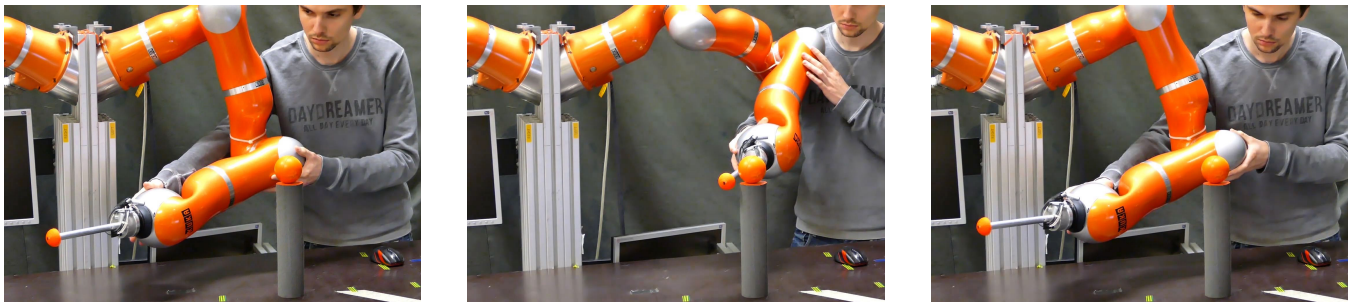


Fig. 2: Teaching of joint space trajectories from the unique starting point (left) to via point (middle) and back (right). The trajectories were sampled at 100 Hz. Note that the robot's end-effector only approximately reaches the via point and that the end point is similar but not exactly the same as the starting point.

IV. AL-DMPs in Time Domain

While AL-DMPs have many favourable properties for learning and recognition, they cannot be used directly for control because a robot is controlled at constant time steps, not constant arc length steps. Equivalent, time-dependent equations can be derived by exploiting the relationship between time and arc length derivatives

$$\dot{y} = \frac{\mathrm{d}}{\mathrm{d}t} y(s(t)) = y' \dot{s}, \tag{28}$$

$$\ddot{y} = \frac{\mathrm{d}^2}{\mathrm{d}t^2} y(s(t)) = y'' \dot{s}^2 + y' \ddot{s}. \tag{29}$$

We can now express arc length derivatives in terms of time derivatives

$$y' = \frac{1}{\dot{s}} \dot{y}, \tag{30}$$

$$y'' = \frac{1}{\dot{s}^3} \left(\ddot{y}\dot{s} - \dot{y}\ddot{s} \right). \tag{31}$$

From (30) – (31) and (9) - (10) we obtain

$$z = Ly' = \frac{L}{\dot{s}} \dot{y} \tag{32}$$

$$\dot{z} = L \frac{\ddot{y}\dot{s} - \dot{y}\ddot{s}}{\dot{s}^2}, \tag{33}$$

$$z' = Ly'' = \frac{1}{\dot{s}} L \frac{\ddot{y}\dot{s} - \dot{y}\ddot{s}}{\dot{s}^2} = \frac{1}{\dot{s}} \dot{z}, \tag{34}$$

and finally, by respectively inserting (28) and (34) into (9) and (10),

$$L\dot{z} = \dot{s} \left(\alpha_z (\beta_z (g - y) - z) + F(x) \right), \tag{35}$$

$$L\dot{y} = \dot{s} z. \tag{36}$$

Similarly, the phase equation (11) can be rewritten in an equivalent, time-dependent form

$$L\dot{x} = -\dot{s} \alpha_x x. \tag{37}$$

A. Control with AL-DMPs

To control a robot at constant time steps, we can integrate Eq. (35) – (37), which are expressed in terms of time derivatives, instead of (9) – (11), which are expressed in terms of arc length derivatives. The initial values are set to $y = y_0$, $z = L\dot{y}_0/\dot{s}_0$, $x = 1$. Note that $\dot{s}$ approximated by Eq. (25) is given as a function of phase. If the second derivatives are also needed by the controller, as is the case with torque-controlled systems, they can be calculated from (33)

$$\ddot{y} = \frac{1}{L} \frac{\dot{s}^2 \dot{z} + \dot{y}\ddot{s}}{\dot{s}}. \tag{38}$$

This equation involves $\ddot{s}$, which we estimate using Eq. (27).

B. Stability of AL-DMPs

The arc length s and its derivative $\dot{s}$ can be learnt only along the demonstrated trajectory. Beyond the end of the learned trajectory, i. e. for $s > L$, we have no information about the arc length and the speed of movement. Under normal circumstances, the end of the executed movement coincides with the attractor point $y = g$, $z = 0$, but in case of perturbations this might not be the case. However, since $\dot{s}$ is strictly positive outside of the interval of learning, x defined by (37) tends to zero as time increases, consequently $\dot{s}$ defined by (25) tends to 1 and the equation system (35) – (36) tends to the second order linear system

$$L\dot{z} = \alpha_z (\beta_z (g - y) - z), \tag{39}$$

$$L\dot{y} = z, \tag{40}$$

which has a unique attractor point at $y = g$, $z = 0$. Thus just like standard DMPs, AL-DMPs are guaranteed to converge to the desired final configuration. Note that $\dot{s}$ is equal to the speed of movement only within the training interval; thus the fact that $\dot{s}$ tends to 1 outside of the training interval does not mean that the speed of movement tends to 1. The actual speed of movement is specified by Eq. (39) – (40) and tends to 0, thus the robot is guaranteed to come to rest at the end of integration process.

V. Experimental results

A. Statistical skill learning with AL-DMPs

The proposed formulation of AL-DMPs has a potential to significantly improve the performance of statistical learning algorithms because it can ensure the proper alignment of the training data. This is essential for many learning algorithms that analyze the interdependencies between the available example trajectories. For example, in statistical learning algorithms, e. g. [6], multiple user demonstrations are locally averaged to generate new movements. Such an averaging process requires the alignment of example trajectories, which is

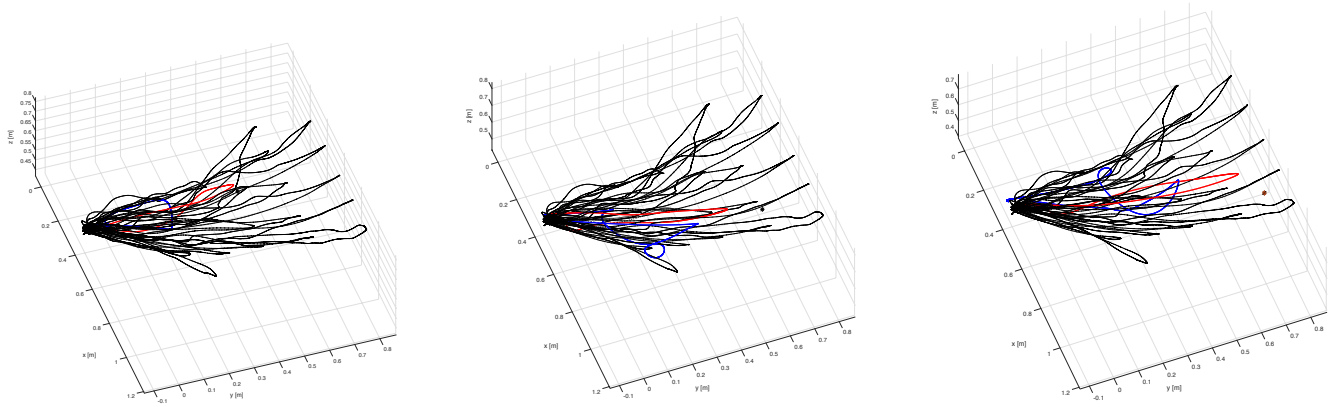


Fig. 3: Three spatial paths generated by statistical trajectory learning using DMP (blue) and AL-DMP representation (red). The training trajectories are shown in black. It is clear that the AL-DMP paths (red) are a better approximation of the data, that they come closer to the via points (shown as stars) and that they exhibit less artifacts not present in the training data than the DMP paths (blue).

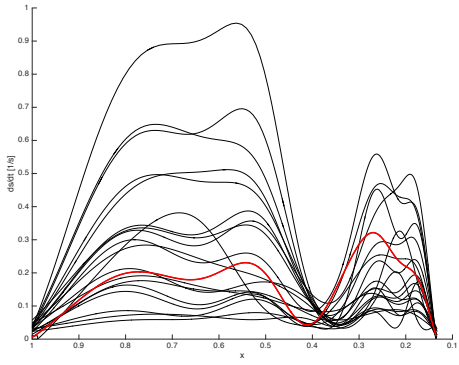


Fig. 4: Speed profiles of the training data. While the speed profiles in the first part of the movements transition between each other fairly smoothly, this is not the case for the second part (after reaching the via point), where the speed profiles are very different. This spoils the results of statistical learning with standard DMPs. The profile shown in red was used as speed profile for trajectories generated by AL-DMPs.

automatically provided by AL-DMPs. In our first experiment we analyzed the effectiveness of the proposed representation for this type of statistical learning.

We compared the performance of the statistical generalization method described in [6] when using standard DMPs and AL-DMPs, where the problem was the learning of via point movements. Fig. 2 shows the experimental setup, where the orange ball specifies the via point, which position can be measured by vision. The trajectory learning results (in Cartesian space) are shown in Fig. 3. It is clear that the trajectories calculated using AL-DMP representation are much closer to the training data than trajectories calculated using standard DMPs. Both the spatial shape of the trajectories as well as the via points are better approximated when AL-DMPs are used. The via points are not exactly reached because they were not reached even in the training data. The better quality of AL-DMP-based trajectories is even more obvious in the **video** accompanying this paper.

The reason for this becomes evident if we analyze the speed profiles of the training data (see Fig 4). Due to the discontinuous transitions between speed profiles after reaching the via point, the DMP-based learning cannot generate trajectories similar to the training data. Since AL-DMPs separate the spatial from the temporal component of motion, they do not suffer from this issue. However, in this case we cannot generalize the temporal component of movement using statistical techniques. Instead, we used a fixed speed profile extracted from one of the example trajectories. This is not possible if the standard DMP representation is used.

B. Recognition with AL-DMPs

Ijspeert et al. [3] showed that the shape weights in the original DMP formulation can be used for recognition of trajectories with "similar velocity profiles". They classified their trajectories by computing correlations between parameter vectors of all example trajectories. They obtained a higher correlation value for trajectories belonging to the same class compared to trajectories from belonging to different classes.

However, in human demonstrated trajectories, the speed of movement may vary across inter-class examples, especially when the demonstrations are performed by several subjects. Since AL-DMPs encode the shape of the trajectory in a manner consistent across examples performed at varying speeds, the recognition performance can be substantially improved compared to the original DMP formulation.

1) Simulated data: To test the performance of recognition with AL-DMPs, we created a simulated dataset that consisted of 805 reaching trajectories of a two degrees of freedom simulated robot. The trajectories were first generated in Cartesian space using a fifth order polynomial with specified initial and final positions, velocities, and accelerations. The Cartesian space trajectories where divided into five classes, where each of the classes consisted of 161 reaching movements to a given distance in x direction, whereas the final destinations in y direction where different. The trajectories were converted into joint space and estimated with standard DMPs and AL-DMPs. The resulting parameters w_i were used to test the performance of classification. Since the

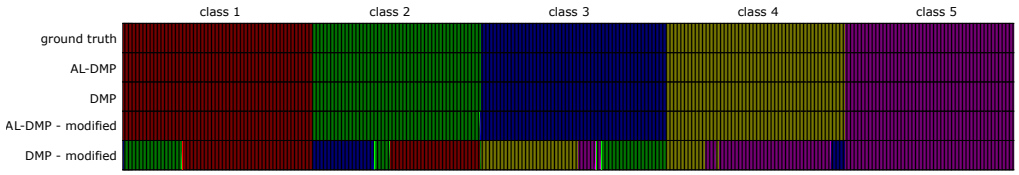


Fig. 5: Classification performance with simulated data. Each colored bar corresponds to an example trajectory from the dataset; its color denotes the class determined by the SVM. For example, red corresponds to the trajectory being classified as a trajectory reaching to the nearest line (see Fig. 6). The above picture is best viewed at increased scaling.

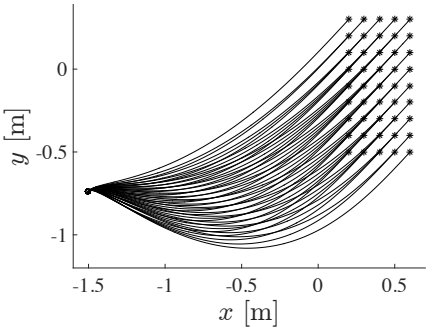


Fig. 6: Cartesian paths of the reaching movements that define 5 classes in the dataset used for testing classification with simulated data. Reaching distance in the x direction defines one class. For each class 161 example trajectories were generated (only 9 of them are shown), with uniformly distributed reaching distances in y direction.

number of basis functions N was 20 and the trajectories were two-dimensional, this amounts to 40 features for each example.

We performed classification using support vector machines (SVM, [20]). SVMs are a non-probabilistic classifier which builds a model of the input pattern by optimally separating the input data. In this experiment we used multiclass (one-vs-one) SVMs with linear kernels. We used Matlab Classification Learner App to train the classifiers. The training set for this experiment consisted of 161 examples (20% of the whole dataset).

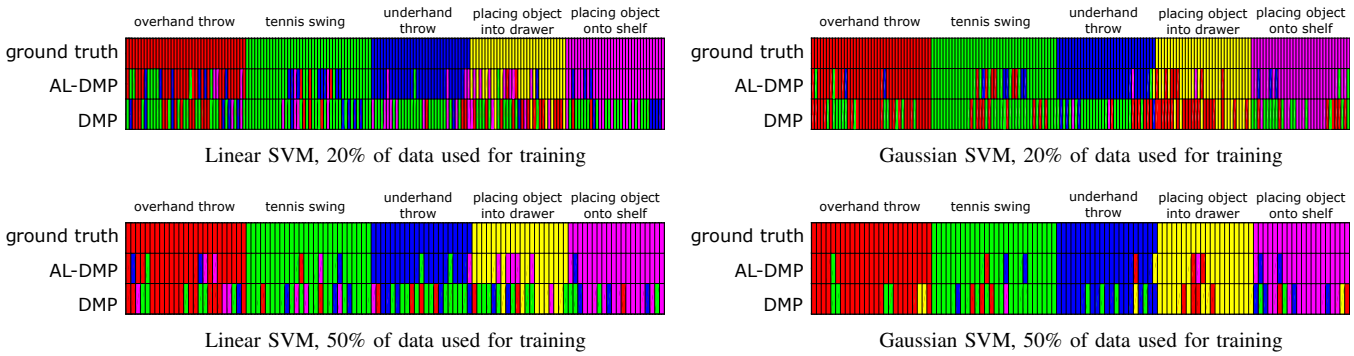
In this rather simple experiment, all of the remaining 644 examples were classified by the SVM correctly, both when AL-DMPs were used to compute the input features, as well as when standard DMPs were used.

Next, we introduced non-linear scaling of speed into the examples. We achieve this by multiplying the speed of the simulated trajectories by $2(t+1)e^{(t+1)^2-2}$, $t \in [0, T]$, where T is the duration of motion. We estimated the newly obtained examples with both DMPs and AL-DMPs and tested the classification performance. As can be seen in Figure 5, recognition rate with DMP-based features was severly affected. Only 40.1% of the examples in the test set were classified correctly. The cause of this result is obvious: the introduced temporal scaling resulted in significant modifications of the values of the shape weights in the underlying DMP representation. Therefore, the features of the new examples are not representative of the classes trained with the original data. On the other hand, the performance

Fig. 7: Movement classes in the dataset used in the classification experiment with real data. Top to bottom: overhand ball throw, tennis swing, underhand ball throw, placing an object onto a shelf, placing an object into a drawer. Note the marker on the demonstrator's wrist.

of AL-DMP-based classification was virtually unaffected. Variations in the speed profile of the trajectories do not change the parameters corresponding to the shape of the trajectory. Therefore, the modified trajectories were still classified correctly. This experiment confirms that classification with AL-DMP features is more robust compared to standard DMP features when the speed of motion varies nonlinearly between examples.

2) Real data: In this experiment we evaluated the recognition performance of AL-DMPs on a dataset consisting of 5 motion classes, in total containing 225 hand trajectories performed by 4 subjects. Of the 5 classes, 2 deal with manipulation (placing a book on a shelf, placing an object into a drawer) and 3 encompass sports actions (tennis swing, 2 versions of a ball throw). See Figure 7 for a graphical



Linear SVM, 20% of data used for training

Gaussian SVM, 20% of data used for training

Linear SVM, 50% of data used for training

Gaussian SVM, 50% of data used for training

Fig. 8: Results of SVM classification where different SVM classifiers and different percentages of data were used for training. Each colored bar corresponds to an example trajectory from the set; its color denotes the class determined by the SVM. For example, red corresponds to the trajectory being classified as overhand throw.

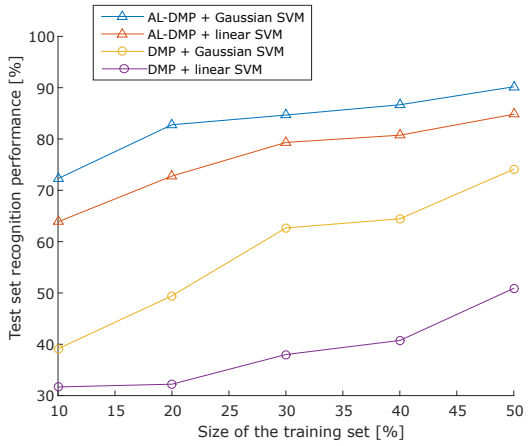


Fig. 9: Classification performance as a function of the training set size. Blue and red graphs show the percentage of correctly classified examples using the Gaussian and linear SVM, respectively, with AL-DMP weights used as features. Yellow and magenta show classification performance when standard DMP weights were used as features.

illustration of the demonstrated manipulation tasks. The trajectories were recorded using Optotrak motion capture system. Some examples of the demonstrated movements are shown in the second part of the accompanying **video**.

Same as before, the trajectories were encoded with AL-DMPs as well as standard DMPs for comparison, with the shape weights w_i being used as classification input features. Again, the number of weights N was 20. In this experiment the trajectories were three dimensional (wrist positions), thus the number of features for each example was 60.

Besides testing the performance of classification when using SVMs with linear kernels, we also evaluated the classification results when using SVMs with Gaussian kernels. Linear SVMs assume that the classes can be separated with hyperplanes in the feature space, whereas Gaussian kernel SVMs support non-linear class boundaries. For this reason, Gaussian kernel SVMs are better suited for classifying complex data. We tested the recognition performance using different training and test set ratios. Namely, we tested the cases where 10, 20, 30, 40 and 50 percents of the trajectories in each class were randomly chosen to form the training sets, with the test sets being composed of the remaining examples.

Figure 9 shows the results. Note that AL-DMP-based features significantly outperform the standard DMP-based features. This is true for both types of SVMs we evaluated. The graph also shows that the application of the more advanced Gaussian-kernel SVM has a greater impact on the result when conventional DMPs instead of AL-DMPs are used. This is due to the fact that the shape weights of AL-DMPs better reflect the changes in the shape of movement. Consequently, the demonstrated movements can be effectively separated even when using simpler linear boundaries.

Figures 8 shows per-sample results of classification for the test cases when 20 or 50 percent of the trajectories in each class were randomly chosen to form the training set. Note that many more example movements are classified correctly when AL-DMP parameters are used as classification features instead of standard DMP parameters.

VI. Conclusion

The main result of this paper is a novel representation for robot movements called *arc length dynamic movement primitive (AL-DMP)*, which effectively separates the spatial and temporal aspects of motion. We have shown in our experiments that the proposed formulation can significantly improve the performance of statistical learning of robot skills and movement recognition algorithms, which are based on dynamic movement primitives.

References

[1] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control – a unifying view," *Progress in Brain Research*, vol. 165, no. 6, pp. 425–445, 2007.

[2] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computations*, vol. 25, no. 2, pp. 328–373, 2013.

[3] A. J. Ijspeert, J. Nakanishi, T. Shibata, and S. Schaal, "Nonlinear dynamical systems for imitation with humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Tokyo, Japan, 2001, pp. 219–226.

[4] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2010.

[5] A. Colomé and C. Torras, "Dimensionality reduction and motion coordination in learning trajectories with dynamic movement primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, 2014, pp. 1414–1420.

[6] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[7] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[8] E. Rückert and A. d'Avella, "Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems," *Frontiers in Computational Neuroscience*, vol. 7, 2013.

[9] A. D. Dragan, K. Muelling, J. A. Bagnell, and S. S. Srinivasa, "Movement primitives via optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, 2015, pp. 2339–2346.

[10] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, 2011, pp. 365–371.

[11] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[12] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[13] T. Kulvicius, M. Biehl, M. J. Aein, M. Tamosiunaite, and F. Wörgötter, "Interaction learning for dynamic movement primitives used in cooperative robotic tasks," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1450–1459, 2013.

[14] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Atlanta, GA, 2013, pp. 423–428.

[15] R. Vuga, B. Nemec, and A. Ude, "Speed adaptation for self-improvement of skills learned from user demonstrations," *Robotica*, pp. 1–17, 2015, doi: 10.1017/S0263574715000405.

[16] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth, "Probabilistic model-based imitation learning," *Adaptive Behavior*, vol. 21, no. 5, pp. 388–403, 2013.

[17] H. Ben Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, pp. 2831–2837.

[18] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[19] M. Ewerton, G. Maeda, J. Peters, and G. Neumann, "Learning motor skills from partially observed movements executed at different speeds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[20] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.