International Journal of Advanced Robotic Systems

Generating a Style-adaptive Trajectory from Multiple Demonstrations

Regular Paper

Yue Zhao[1], Rong Xiong[1,*], Li Fang[1] and Xiaohe Dai[1]

1 Key Laboratory of Industrial Control Technology, Zhejiang University, China
* Corresponding author E-mail: rxiong@iipc.zju.edu.cn

Received 11 Dec 2013; Accepted 29 May 2014

DOI: 10.5772/58723

Abstract Trajectory learning and generation from demonstration have been widely discussed in recent years, with promising progress made. Existing approaches, including the Gaussian Mixture Model (GMM), affine functions and Dynamic Movement Primitives (DMPs) have proven their applicability to learning the features and styles of existing trajectories and generating similar trajectories that can adapt to different dynamic situations. However, in many applications, such as grasping an object, shooting a ball, etc., different goals require trajectories of different styles. An issue that must be resolved is how to reproduce a trajectory with a suitable style. In this paper, we propose a style-adaptive trajectory generation approach based on DMPs, by which the style of the reproduced trajectories can change smoothly as the new goal changes. The proposed approach first adopts a Point Distribution Model (PDM) to get the principal trajectories for different styles, then learns the model of each principal trajectory independently using DMPs, and finally adapts the parameters of the trajectory model smoothly according to the new goal using an adaptive goal-to-style mechanism. This paper further discusses the application of the approach on small-sized robots for an adaptive shooting task and on a humanoid robot arm to generate motions for table tennis-playing with different styles.

Keywords Dynamic Movement Primitives, SADMPs, Trajectory Learning

1. Introduction

Trajectory generation is a fundamental issue in the field of robotics, the object of which is to output a smooth and safe path that a moving robot can follow through space. The generated trajectory can be described as a function of pose or velocity with respect to time. Traditional methods generally adopt some predefined functions to describe the trajectory, then determine the parameters of the function according to optimization objectives and constraints [1–3]. This category of method has been widely and successfully applied in various robotic systems. However, an artificially predefined function with a fixed form can hardly be used to accurately describe complex trajectories. It is also difficult to modify these functions to produce similar trajectories to adapt to different dynamic situations, such as a changed destination or a moving obstacle, without replanning the whole trajectory. The most recent idea is to learn the trajectory from demonstrations, which brings challenges to the modelling of the trajectory but eases the cost of real-time replanning for new situations. Promising progress has been made in the published literature, such as GMM [4], affine functions [5] and DMPs [6].

The methods mentioned above can learn and generalize the motion styles of the demonstrated trajectories. Here, a motion style is referred to as a motion sequence which is distinct in its shape or curve tendency. However, in many situations, different styles of trajectories will be adopted for different goals. An example is depicted in Figure

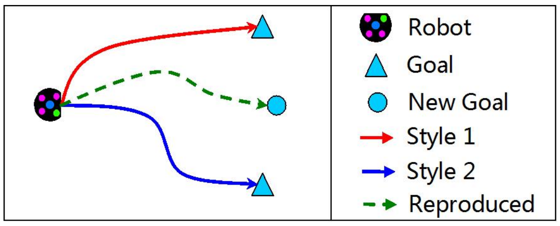


Figure 1. A two-boundary trajectory planning example. The robot's task is to reach the goal. Trajectories in different styles could be adopted to the triangular goals. As a new circle goal is set, a new different stylistic trajectory (dashed) should be generalized to reach the new goal.

1, where two motion trajectories of two different styles with the same start point but different goal points are learned. Based on the styles of the these two motions, any trajectories that start at the same point and end with a different goal are expected to share similarities in style with the learned ones and with changes that are as smooth as possible. A desired reproduced trajectory is shown in Figure 1, the style of which is different from that of the two learned trajectories but which integrates their features. The main focus of this paper is to address the problem of style-adaptive motion trajectory generation for different goals based on trajectories of various styles learned from multiple demonstrations.

In this paper, we propose a style-adaptive trajectory generation method based on DMPs, which is called 'Style Adaptive Dynamics Movement Primitives' (SADMPs). In SADMPs, demonstrated trajectories are first modelled and clustered to a series of style-different principal trajectories using PDM [8]. Next, each principal trajectory is trained using the DMPs to get its weight parameters. Finally, an adaptive goal-to-style mechanism merges the weight parameters of different styles to obtain new weight parameters according to the changes in goal position.

The main contributions of this paper are:

- A framework for learning and generating style-adaptive trajectories is proposed, which supports the generation of new trajectories based on multiple learned motion examples and which needs fewer demonstrations.
- An algorithm for fusing and adapting different motion styles is proposed, which consists of the Least Mean Squares (LMS) method and a goal-to-style mechanism. This algorithm blends different motion styles to generate a new trajectory with a smooth transition.

SADMPs has been successfully applied to our robot in the Small Size League (SSL) for an adaptive shooting task, as well as to a humanoid robot arm to generate motions for table tennis playing with different styles.

The remainder of this paper is organized as follows. Section 2 introduces the related work about learning trajectories from demonstrations. In Section 3, we describe the discrete formalism of the original DMPs. Next, we propose a framework for the SADMPs and discuss the learning procedure for the SADMPs in Section 4. Section 5 demonstrates the effectiveness of the proposed method for the SSL robot and for a striking ball task in the humanoid robot. Section 6 concludes the paper.

2. Related Work

Billard proposed to model the non-linear dynamical point-to-point robot motions as a time-independent system using an iterative algorithm to estimate the form of the dynamical system through a mixture of Gaussian distributions [4]. Although this method is suitable for processing large amounts of motion data due to the parametric model, it generates discontinuities in the trajectories. GMM has been used in different robotics applications, such as gesture imitation [9] and handwriting [10].

Pham et al. proposed a method using affine trajectory deformation for motion imitation [5]. This method is based on the affine invariance of human motion [11], and makes no use of an exogenous basis function. This method also allows the deformed motions to preserve the similar motion styles in relation to the observed one. The style of the generated motions can only be altered by the modification of the observed trajectories.

Another pioneering work introduced by Ijspeert et al. [6] considered a robot's movement to be a linear spring system coupled with an external forcing term, which can be described by a set of differential equations. In order to obtain the weight parameters using the DMPs formulations, Ijspeert et al. applied Locally Weighted Learning (LWL). Schaal introduced an advanced version algorithm named 'Receptive Field Weighed Regression' (RFWR) to solve the problem [12] in which the centre and width of the kernel functions were adjusted automatically. The RFWR could also select the appropriate number of kernel functions to fit the training trajectory. The effectiveness of this method has been validated through various applications, such as walking [13], flight control [14] and hitting and batting [15].

The DMPs employed in [6] used a very basic model to learn a two-boundary trajectory without considering obstacles and kinematic restrictions. Motivated by the requirement of high-order continuity, studies[15, 16] were also conducted to develop extensions and modifications to DMPs in planning with a non-zero terminal velocity. It is also easy to extend the DMPs formula with a dynamic potential field item to achieve obstacle avoidance [17, 18]. Furthermore, a method for the comparison and clustering of robotic trajectories with human motion data is proposed in [19], which is applied to achieving motion by learning from a large set of demonstrations.

The approaches proposed in [4, 6] provide good results for style-fixed tasks - in other words, the demonstrated style is always suitable for the new goal. However, in real environments, the motion styles have functional roles to solve specific tasks, such as hitting balls or avoiding obstacles. Matsubara et al. proposed an approach called 'Stylistic Dynamic Movement Primitives' (SDMPs) [7] which controls the motion style by manipulating a style parameter added to the original DMPs. However, the mapping between the style parameter and the perceptual feedback of the motion goal is set by hand in their work, and is neither automatic nor smooth.

3. Dynamic Movement Primitives

We firstly briefly introduce the original DMPs using the same description as in [18]. DMPs explain a movement using the following set of differential equations for a one degree-of-freedom (DoF) motor system:

$$\tau\dot{v} = K(g-x) - Dv - K(g-x_0)u + Kf(u) \quad (1)$$

$$\tau\dot{x} = v \quad (2)$$

$$\tau\dot{u} = -\alpha u \quad (3)$$

where

- x is the position,
- v is the velocity,
- x_0 is the start position,
- g is the goal position,
- K is the spring constant,
- D is the damping term,
- u is the phase corresponding to time changing from 1 towards 0,
- τ is a temporal scaling factor,
- α is a pre-defined constant,
- f is a non-liner function which simulates an external force.

Equations (1) and (2) are referred to as the *transformation system* while the differential equation(3) is called the *canonical system*. In a practical multi-DoF motor system, every DoF effects the transformation system independently, but they could share the same canonical system. Usually, K and D are chosen with special care so that the *transformation system* is critically damped. Moreover, τ and α are also chosen to make u close to zero. These equations are time invariant and can alter the duration of a movement by simply changing τ. In addition, once the non-linear function f is defined, the style of the resulting trajectory will be determined as well. Specifically, f can be given as:

$$f(u) = \frac{\sum_i^N w_i\psi_i(u)u}{\sum_i^N \psi_i(u)} \quad (4)$$

$$\psi_i(u) = exp(-h_i(u-c_i)^2) \quad (5)$$

where ψ_i are Gaussian basis functions with a centre c_i and a width h_i, N is the total number of Gaussian basis functions, and w_i are the weights to learn. An important advantage of DMPs is that one demonstration can be imitated by one-shot learning. In order to learn from the demonstrated trajectory, first the movement $x(t)$ should be recorded, then its derivative $v(t)$ and $\dot{v}(t)$ for each time step, $t = 0, ..., T$, can be solved. By combining the transformation system and canonical system together, $f_{target}(u)$ is obtained by:

$$f_{target}(u) = \frac{\tau\dot{v} + Dv}{K} + (g-x_0)u - (g-x), \quad (6)$$

where $x_0 = x(0)$ and $g = x(T)$. The weights w_i in (4) can be calculated by minimizing the error criterion $J = \sum(f_{target}(u) - f(u))^2$, which is a linear regression problem and can be solved efficiently. Once w_i is determined for a demonstration input, the trajectory with a new goal position can be generalized by resetting $g = g_{new}$, $x_0 = x_{current}$ and $t = 0$. After this setup procedure, τ is adjusted manually to determine the duration of the movement.

4. Style-adaptive Dynamic Movement Primitives

4.1. System Architecture

Figure 2 shows the architecture of the SADMPs. We first collect and analyse different motions demonstrated by a human. The captured motions are modelled and clustered using PDM [8] in order to get the principal trajectory of each cluster, which is the average of the trajectories in a cluster [19]. Next, we train these principal trajectories using the DMPs trainer separately to get their weight parameters. Finally, we use the adapter to combine the training results with the new goal to reproduce new motions. Here, a new desired goal g_{new} acts not only on the transformation system - which is similar to the original DMPs - but also helps to generate a fused style of the reproduced motion.

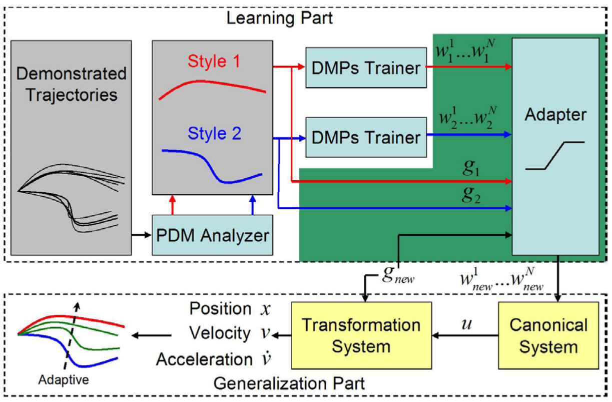


Figure 2. The architecture of the proposed SADMPs model. This method comprises two parts: one is for learning and the other is for generalization. The dark green-shaded components constitute an adaptive goal-to-style mechanism. Two points should be noted for the extension: 1) the architecture is for one-DoF - J copies of this architecture could be employed for a J-DoF application; 2) Although there are only two clusters obtained in this figure, the SADMPs can work for a situation involving multiple clusters.

PDM is a useful tool for analysing trajectories. In order to analyse the trajectories using PDM, an adequate number of movements must be collected from demonstrations. The recorded trajectories will be cropped so that they start at the same point and then re-sampled using cubic spline interpolation so that they contain the same number of points. The detailed procedure of PDM is depicted in [8]. Figure 3 shows an example of the data processing procedure using the PDM.

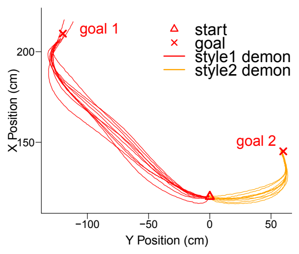
4.2. Algorithm for Learning

The learning of $\mathbf{w_{new}}$ in SADMPs can be accomplished in two steps: (I) Obtaining the weight parameters $\mathbf{w}_k = [w_k^1 \cdots w_k^N]^T$ [1] for the kth principal trajectory; (II) Employing an adaptive goal-to-style mechanism to merge different $\mathbf{w}_k$.

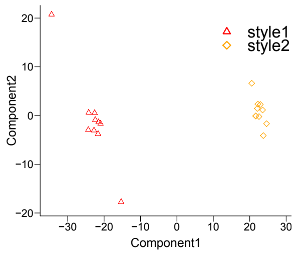
4.2.1. Training the Principal Trajectory by LMS

In the first step, and distinct from the training method used by [6, 12], an LMS method is used to train the parameters

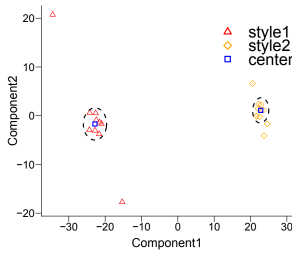
[1] The representation of $\mathbf{w}_k$ is sightly different from Section 3. Here, k is the kth principal trajectory, N is the number of kernel functions.



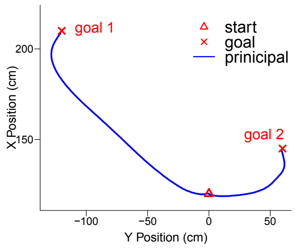
(a)



(b)



(c)



(d)

Figure 3. The data processing using the PDM. In (a), two goals are set and 10 reaching motions are captured for each of the two goals. Next, the first two components are employed to present all the trajectories and perform a *k*-means clustering in the PCA space, as shown in (b) and (c). Finally, the principal trajectories are computed, which are the averages of the trajectories for each cluster in the trajectory space, as in (d).

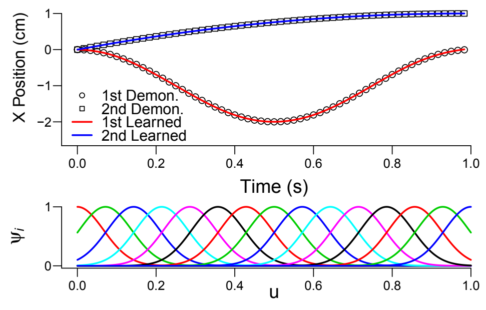
$\mathbf{w_k}$ of every principal trajectory in this paper. The main reason for this design is that the LMS method can obtain the weight parameters by using the same kernel functions for different principal trajectories in the same dimension, which is the basis of the goal-to-style mechanism in the next step. Compared with the LWL method in [6], LMS can decrease the computational cost when generating a new trajectory, particularly when there are many principal trajectories to be blended in the adapting process. Another method named 'RFWR' in [12] can adjust the centre and width of the kernel functions automatically, but it cannot guarantee the production of the same kernel functions for different principal trajectories. So, LMS is chosen as the learning algorithm in our system. The update rule is:

$$w_k^i \leftarrow w_k^i - \alpha_r \frac{\partial (f_{target}(u(t)) - f(u(t)))^2}{\partial w_k^i}, \tag{7}$$

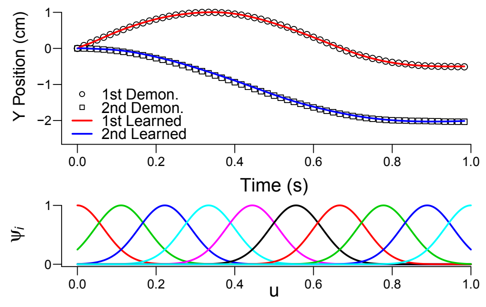
where α_r is the learning rate. Replacing $f(u(t))$ with (4) and $u(t) = e^{(-\alpha t/\tau)}$ according to (3), we employ the training method corresponding to the batch gradient descent on every example in the entire training set at every time step. As such, (7) can be rewritten as:

$$w_k^i \leftarrow w_k^i + \alpha_r \sum_{t=0}^{T} \frac{(f_{target}(u(t)) - f(u(t)))\psi_i(u(t))u(t)}{\sum_{i=0}^{N} \psi_i(u(t))}, \tag{8}$$

Usually, a principal trajectory has more than one dimension and the weight parameters of each dimension should be learned independently and in parallel. Thus, in our training process, we consider two rules. First, for different principal trajectories in the same dimension, they must have the same kernel functions for training, which means that the kernel functions are the same in shape, number and distribution. Second, for one principal trajectory in different dimensions, different kernel functions could be chosen for training. For example, there are two principal trajectories in Figure 4, and each of them has two dimensions, respectively, in X an Y. Both of them use the same kernel functions for training, whether in the X dimension or in the Y dimension. However, the kernel functions of the X dimension are different from those of the Y dimension, and the difference includes the number and distribution of the kernel functions.



(a) X translational direction



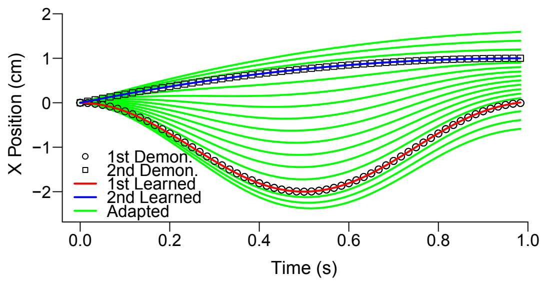
(b) Y translational direction

Figure 4. Training using LMS: these two demonstrated movements are approximated using 15 kernel functions for the X translational direction in (a) and 10 kernel functions for the Y translational direction in (b)

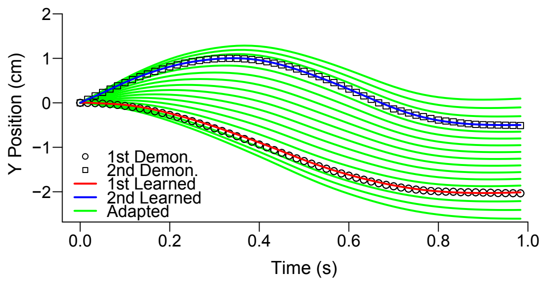
4.2.2. Adapting to New Goal

In the next step, an adapter is responsible for the adaptive goal-to-style mechanism. In the original DMPs formulations, the goal position g and the temporal scaling factor τ determines the style of the trajectory. In SADMPs, we further coupled g to the weight parameters; thus, the style of a generated motion changes smoothly between movement primitives of different styles.

The goal of the kth principal trajectory in one-DoF is g_k. We sort the one-DoF goals of all the principal trajectories in ascending order so that $g_1 < g_2 < \cdots < g_M$, M is the total number of the principal trajectories. Note that for different DoFs, the sorting result might be different. If $g_k \le g_{new} \le g_{k+1}$, then g_{new} can be represented as:



(a) X translational direction



(b) Y translational direction

Figure 5. Reproduced adaptive trajectories in two dimensions - two principal trajectories are used in this example. The styles of the reproduced trajectories (green) change smoothly according to the goal.

$$g_{new} = \frac{d(g_k)g_{k+1} + d(g_{k+1})g_k}{d(g_k) + d(g_{k+1})} \tag{9}$$

where $d(g_k) = |g_{new} - g_k|$. Considering the two transformation systems learned for g_k and g_{k+1}, these can be represented as:

$$f_k(u) = \frac{\tau\dot{v} + Dv}{K} + (g_k - x_0)u - (g_k - x), \tag{10}$$

$$f_{k+1}(u) = \frac{\tau\dot{v} + Dv}{K} + (g_{k+1} - x_0)u - (g_{k+1} - x). \tag{11}$$

Since $(f_k(u) * d(g_{k+1}) + f_{k+1} * d(g_k))/(d(g_k) + d(g_{k+1}))$, we have:

$$f_{new}(u) = \frac{\tau\dot{v} + Dv}{K} + (g_{new} - x_0)u - (g_{new} - x), \tag{12}$$

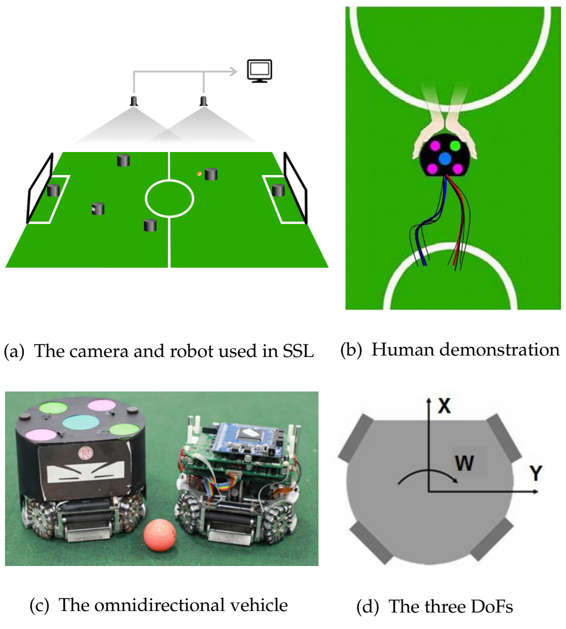
Then, the new weight parameters $\mathbf{w_{new}}$ for generalization can be calculated as:

$$w^i_{new} = \begin{cases} \dfrac{d(g_k)w^i_{k+1} + d(g_{k+1})w^i_k}{d(g_k) + d(g_{k+1})} & g_k \le g_{new} \le g_{k+1}, \\ w^i_1 & g_{new} < g_1, \\ w^i_M & g_{new} > g_M. \end{cases} \tag{13}$$

It is clear that the $\mathbf{w}_{new}$ is determined by the distance between the new goal and the goals of the principal trajectories. In other words, the closer a goal is to the new goal, the more likely it is to be selected in $\mathbf{w}_{new}$.

When a new goal is set, the non-linear term of DMPs will drive the system away from its initial state by resetting the parameters, which is similar to the original DMPs mentioned in Section 3. The difference with SADMPs is that the weight parameters are also changed to adapt to a new goal. Figure 5 shows the adaptively learned trajectories using formula (13).

The fundamental property of the DMP formulation is that it is spatially invariant. If we obtain $\mathbf{w}_{new}$ using the linear superposition rule, such as (13), when $g_k \le g_{new} \le g_{k+1}$, we can ensure that the trajectory will converge to the new goal accurately.



(a) The camera and robot used in SSL (b) Human demonstration

(c) The omnidirectional vehicle (d) The three DoFs

Figure 6. The robot and scene for motion-capture

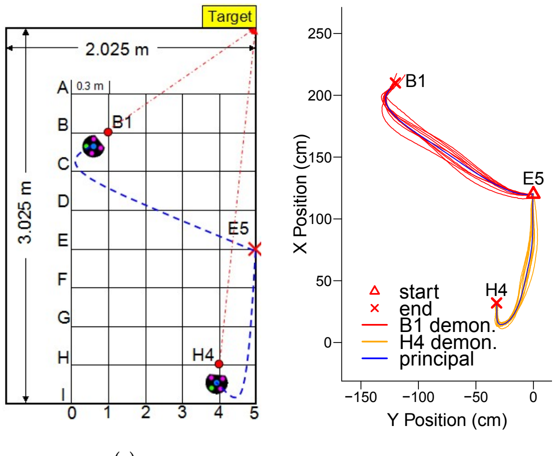
5. Experiments

In this section, we test the proposed method by conducting an adaptive shooting task on an SSL platform and compare our method with the original DMPs quantitatively. Furthermore, the SADMPs are applied to a humanoid robot performing a table tennis task.

5.1. Shooting Ball Task in SSL

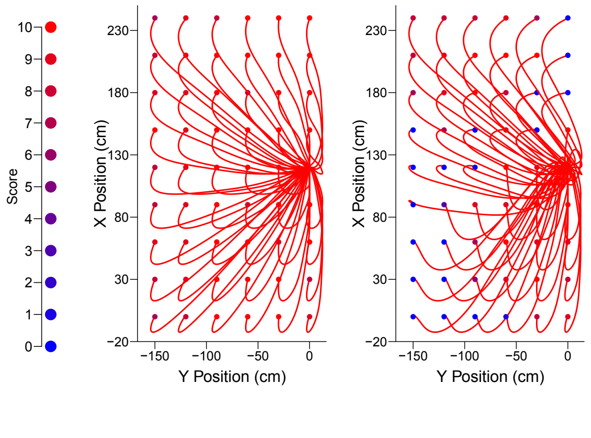
5.1.1. Testing Platform and Training

The Small Size League (SSL) is the fastest and most intense game among RoboCup's soccer competitions. The basic rules of SSL are based on the rules of a FIFA soccer game, but each team consists of only six robots playing on field that is 6.05 m long by 4.05 m wide. There are two cameras mounted over the field to capture images at 60 Hz for further processing in a shared vision system named 'SSL-Vision' [20]. This system recognizes and locates the position and orientation of the robots and the position of the ball, as shown in Figure 6(a), and then broadcasts the information package to each team via a network. The team that scores the most goals wins the game.



(a) (b)

Figure 7. The scene and demonstrated trajectories for the shooting ball task. (a) A quarter field in the SSL is used for this experiment, and the shooting task was demonstrated 10 times for every predefined pose. (b) The recorded data from multiple human demonstrations.



(a) (b)

Figure 8. Comparison of DMPs and SADMPs in trajectory generalization and scores for every shooting point. (a) is the result of the SADMPs method, while (b) is the result of the DMPs method. For the representation of the score, a dot is drawn at the end of the trajectory, whereby the colour of the dot denotes the score (0-blue, 10-red).

The robot we used is an omnidirectional vehicle with four wheels and which can shoot the ball using an electromagnet-controlled device installed in the front, as shown in Figure 6(c). The motion of the robot can be decoupled kinematically into three DoFs: a X translational direction, a Y translational direction and a W rotational direction, see Figure 6(d), and our proposed method will be employed in these three DoFs independently.

In our experiment, we perform a shooting task to test the method. In this task, the robot should accomplish two basic subtasks: one is to reach the correct goal pose without touching the ball; the other is to kick the ball to the shooting target [2]. As the ball may appear anywhere on the field, different stylistic trajectories will be adopted according to the position of the ball. As such, we hope that the robot moves to the ball in a suitable motion style according to the ball's location.

In order to accomplish the shooting task, we set a 9×6 grid on the quarter field, such that a golf ball could be placed at 53 points except for start point, as shown in Figure 7(a). The initial pose of the robot was at $E5$, and the initial orientation was facing towards the shooting target. Furthermore, the speed of the robot would be zero when reaching the shooting pose because of the built-in shooting device. For the learning process, two rules should be followed when choosing the demonstrated goals: 1) the stylistic distinction between any two demonstrated trajectories should be obvious; 2) most of the other goals should be located between any two demonstrated goals. According to the two rules, in our experiment, two demonstrated goal poses were set up, namely $B1$ and $H4$.

The experiment was carried out as follows: first, a human demonstrator moved the robot to perform a shooting ball task at the goal pose by hand, see Figure 6(b), and 10 demonstrated trajectories for each shooting pose were collected, see Figure 7(b). Next, we employed the training algorithm mentioned in Section 4.2 to train different stylistic trajectories. Note that the different trajectories share the same kernel functions and canonical system for each DoF. Finally, the adapter merged the learned $\mathbf{w_k}$ in each dimension to generalize to new goals. In the generalization process, the orientation to the shooting target was the direction of the red dashed vector which was drawn from the ball-set position to the target, as depicted in Figure 7(a).

5.1.2. Results and Comparison

We compare the performance of the proposed approach and that of the original DMPs method. Because the original DMPs do not take multiple stylistic trajectories into account, we choose the weight parameters of the training pose closest to the new goal for generalization.

In the shooting ball task, the robot should not only reach the goal pose without touching the ball but also kick the ball to the shooting target. We repeated the shooting task 10 times in each goal pose and counted the score, as shown in Figure 8.

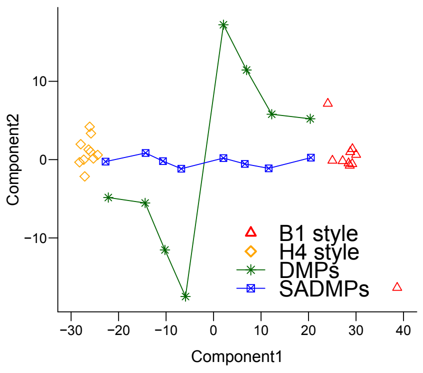


Figure 9. Comparison of the difference between the SADMPs method and the DMPs method in the PCA space

[2] In the experiment, 'goal pose' means that the robot needs to reach both the correct position and orientation, while 'shooting target' denotes the soccer goal that the ball should be kicked towards

Obviously, both methods can reproduce motions which generalize to the new goal accurately. However, some trajectories reproduced by the DMPs knock the ball away, such as the motion to $F0$ shown in Figure 8(b), and the robot cannot score in this situation. The SADMPs method cannot only move the robot to the new target pose successfully, but it can also kick the ball to the shooting target with a high score, with the average score percentage at about 92%, as shown in Figure 8(a). Compared with the collecting and training method in [19], which made demonstrations for all the goal poses in a 8×4 grid, our method only needed to demonstrate two situations.

Figure 9 shows the locations of the 20 demonstrated trajectories in the space formed by the first two components of the PDM, which contribute 94.15% to the total deformation energy in the PCA space. It is noticeable that both clusters are separated. In order to verify that SADMPs can generalize an adaptive motion style, we project several reproduced trajectories with different goals to the PCA space. In Figure 9, the selected goal poses are $C1$, $C2$, $D2$, $E2$, $E3$, $F3$, $G3$ and $G4$. Compared with the reproduced trajectories by DMPs, it is clear that the style points in the PCA space array smoothly from the $B1$ cluster to the $H4$ cluster, rather than the large jump between $E2$ and $E3$ as by the DMPs method.

5.2. Striking Ball Task in Table Tennis

The proposed method is also used on a humanoid robot to play table tennis. We set up a table tennis playing scenario, as shown in Figure 10(a). The humanoid robot that we designed is 165 cm in height and 58 kg. Two seven-DOF arms are equipped on each robot so that they can achieve flexible table tennis playing and serving. Each robot stands on one side of a standard table tennis table, which is 2.74 m long, 1.525 m wide and 76 cm high, with a 15.25 cm high net in the middle of the table. During play, the speed of the ball varies from 3 m/s to 10 m/s. The detailed joint parameters are shown in Table 1.

In order to learn from the human, we use an OptiTrack S250e motion capture system to record the human motion trajectories which consists of the 3D positioning of optical markers attached on human actors, acquired over 8 ms intervals, as show in Figure 10(b).

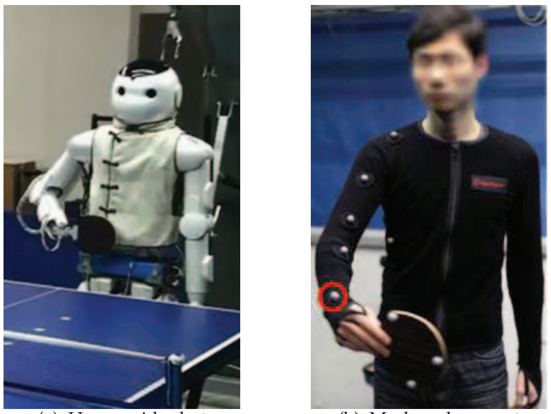

(a) Humanoid robot (b) Marker placement

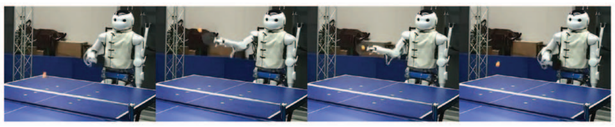
Figure 10. Marker placement for the optical motion capture system and the humanoid robot platform. The motion of the marker in the red circle is recorded and trained in Euclidean space.

Joints	Range (deg)	Maximal angular velocity (rad/s)	Maximal angular acceleration (rad/s^2)	Continuous output torque (mNm)
Shoulder pitch	-120∼90	13	293	12000
Shoulder roll	-130∼15	18	225	9720
Shoulder yaw	-180∼90	12	518	5550
Elbow roll	-60∼120	20	380	9600
Elbow yaw	-180∼180	11	190	2600
Wrist roll	-90∼90	10	277	1080
Wrist pitch	-45∼100	4.5	198	1520

Table 1. Joint parameters for seven-DOF manipulation

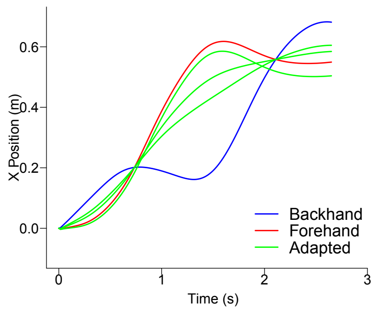
In such a hitting ball task, the forehand and backhand are considered to be two totally different hitting styles, as shown in Figure 11. From the end-effector's (paddle) point of view, different sides are used, as determined by the motion of the wrist joints. Considering the end-effector's trajectory, its motion style will affect the motion planning of other joints. As such, in practice the paddle style (forehand or backhand) is determined according to the ball's speed and impact point, ensuring the accuracy of striking each round. Here, our proposed method is only used to decide the trajectory of the marker in the red circle (Figure 10(b)) in a 3D Euclidean space.

In the application, the demonstrated striking position of the forehand trajectory is $[0.5456, 0.2291, 0.6100]m$, and the demonstrated striking position of the backhand trajectory is $[0.6750, 0.4697, 0.5002]m$. Next, we employed our method to learn the weight parameters for the demonstrations. In the generalizing procedure, in order to predict the ball's current position and velocity more accurately, the raw 3D positions are filtered by a Kalman filter and the striking point and velocity are calculated

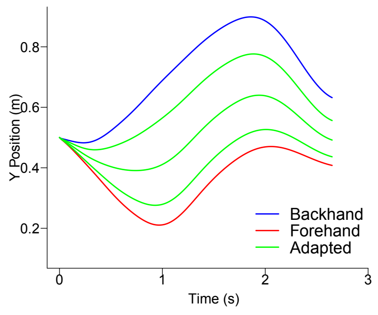

(a) Forehand striking

(b) Backhand striking

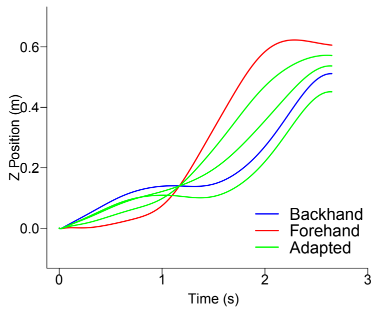
Figure 11. Hitting ball task for a humanoid robot playing table tennis



(a) X dimension



(b) Y dimension



(c) Z dimension

Figure 12. The result of reproducing a style-fused trajectory in each dimension

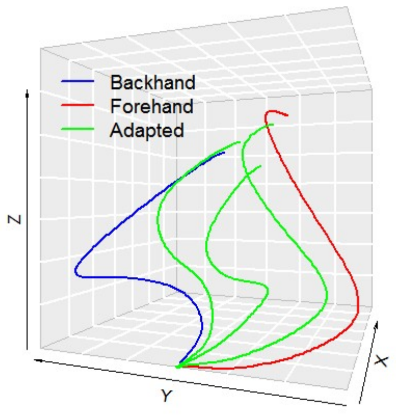


Figure 13. Generalizing to new goals in application to table tennis

using the method mentioned in [21]. Note that the robot should strike the ball at a virtual point with a goal-oriented velocity, so a moving target should be introduced to the original DMPs differential equations, detailed in [15]. Finally, the striking primitives will be generated by SADMPs according to the predicted virtual striking point and be executed on the robot in a joint-space using an inverse dynamics control law.

In our experiment, the robot successfully hit the ball at different positions near the demonstrated positions if kinematically feasible. Figure 12 is the result of reproducing a style-fused trajectory in each dimension - three new reproduced trajectories are shown in the figure, the striking positions of which are $[0.6021, 0.3912, 0.5305]m$, $[0.5786, 0.3211, 0.4386]m$, $[0.5020, 0.2619, 0.5683]m$. Figure 13 shows the result in a 3D Euclidean space.

6. Conclusion

In this paper, we proposed a style-adaptive trajectory generation method based on DMPs called SADMPs for learning multiple trajectories from a human's demonstrations. This method needs fewer demonstrations and leads to a smooth transition between different motion styles. In SADMPs, we first use the Point Distribution Model to cluster our demonstrated movements in a PCA space and then calculate the principal trajectory of every cluster. Next, we train the principal trajectories independently to obtain the weight parameters based on Dynamic Movement Primitives. Finally, we proposed a goal-to-style mechanism for the adaptive changes between different motion styles. We evaluate this novel approach in a SSL robot and a humanoid robot. In these cases, our novel motor primitives generalize smooth and adaptive trajectories according to the goals.

Our future work will focus on extending the method to more complicated applications of a humanoid robot and multiple-agent motion planning in a competitive and dynamic environment, such as a SSL game.

7. Acknowledgements

This work is supported by the National Nature Science Foundation of China (Grant No. NSFC: 61075078).

8. References

[1] Biagiotti L and Melchiorri C (2008) Trajectory planning for automatic machines and robots. Springer, 2008.

[2] Howard TM and Kelly A (2007) Optimal rough terrain trajectory generation for wheeled mobile robots. In: The International Journal of Robotics Research, 26(2), pp. 141-166.

[3] Ude A, Atkeson CG, and Riley M (2000) Planning of joint trajectories for humanoid robots using B-spline wavelets. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on vol. 3, pp. 2223-2228.

[4] Khansari-Zadeh SM and Billard A (2010) BM: An iterative algorithm to learn stable non-linear dynamical systems with Gaussian mixture models. In: Proc. Int. Conf. Robotics Automation, Anchorage, Alaska, 2010, pp. 2381-2388.

[5] Pham QC and Nakamura Y (2012) Affine trajectory deformation for redundant manipulators. In: Robotics: Science and Systems, 2012.

[6] Ijspeert AJ, Nakanishi J, and Schaal S (2002) Learning attractor landscapes for learning motor primitives. In: Advances in Neural Information Processing Systems, vol. 15, 2002, pp. 1523-1530.

[7] Matsubara T, Hyon S, and Morimoto J (2010) Learning stylistic dynamic movement primitives from multiple demonstrations. In: Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., Oct. 2010, pp. 1277-1283.

[8] Roduit P, Martinoli A, and Jacot J (2007) A quantitative method for comparing trajectories of mobile robots using point distribution models. In: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2007, pp. 2441-2448.

[9] Calinon S, D'halluin F, Sauser E, Caldwell D, and Billard A (June 2010) Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression. In: IEEE Robot. Autom. Mag., vol. 17, no. 2, pp. 44-54.

[10] Khansari-Zadeh SM, and Billard A (2011) Learning stable nonlinear dynamical systems with Gaussian mixture models. In: IEEE Transactions on Robotics, vol. 27, no. 5, pp. 943-957.

[11] Bennequin D, Fuchs R, Berthoz A, and Flash T (2009) Movement timing and invariance arise from several geometries. PLoS Comput Biol, 5(7): e1000426, Jul 2009. doi: 10.1371/journal.pcbi.1000426.

[12] Schaal S (1997) Learning from demonstration. In: Advances in neural information processing systems, 1997, pp. 1040-1046.

[13] Nakanishi J, Morimoto J, Endo G, Cheng G, Schaal S, and Kawato M (2004) Learning from demonstration and adaptation of biped locomotion. In: Robot. Auton. Syst., vol. 47, nos. 2-3, 2004, pp. 79-91.

[14] Perk BE, and Slotine JJE (2008) Motion primitives for robotic flight control. In: Cornell Univ., Ithaca, NY, Rep. cs/0609140, Sep. 2008.

[15] Kober J, Mulling K, Kromer O, Lampert CH, Scholkopf B, and Peters J (2010) Movement Templates for Learning of Hitting and Batting. In: Proc. ICRA IEEE Int. Conf. Robotics and Automation, Anchorage, 2010, pp. 853-858.

[16] Ning K, Kulvicius T, Tamosiunaite M, and Worgotter F (2011) Accurate position and velocity control for trajectories based on dynamic movement primitives. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 5006-5011). IEEE.

[17] Park D, Hoffmann H, Pastor P, and Schaal S (2008) Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: Humanoid Robots, 2008. 8th IEEE-RAS International Conference on (pp. 91-98). IEEE.

[18] Pastor P, Hoffmann H, Asfour T, and Schaal S (2009) Learning and generalization of motor skills by learning from demonstration. In: Proc. IEEE Int. Conf. Robot. Autom., Kobe, Japan. 2009, pp. 763-769.

[19] Stulp F, Oztop E, Pastor P, Beetz M, and Schaal S (2009) Compact models of motor primitive variations for predictable reaching and obstacle avoidance. In: 9th IEEE-RAS International Conference on Humanoid Robots, 2009, pp. 589-595.

[20] Zickler S, Laue T, Birbach O, Wongphati M, and Veloso M (2010) SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: RoboCup 2009: Robot Soccer World Cup XIII, pp. 425-436.

[21] Zhang Y, Xiong R, Zhao Y, and Chu J (2012) An adaptive trajectory prediction method for ping-pong robots. In: Intelligent Robotics and Applications, Springer Berlin Heidelberg, 2012, pp. 448-459.