Movement Primitives with Multiple Phase Parameters

Marco Ewerton[1], Guilherme Maeda[1], Gerhard Neumann[2], Viktor Kisner[3], Gerrit Kollegger[4], Josef Wiemeyer[4] and Jan Peters[1,5]

***Abstract*— Movement primitives are concise movement representations that can be learned from human demonstrations, support generalization to novel situations and modulate the speed of execution of movements. The speed modulation mechanisms proposed so far are limited though, allowing only for uniform speed modulation or coupling changes in speed to local measurements of forces, torques or other quantities. Those approaches are not enough when dealing with general velocity constraints. We present a movement primitive formulation that can be used to non-uniformly adapt the speed of execution of a movement in order to satisfy a given constraint, while maintaining similarity in shape to the original trajectory. We present results using a 4-DoF robot arm in a minigolf setup.**

I. Introduction

Learning from human demonstrations has been playing an increasingly important role in robotics, especially since robots are getting better at adapting demonstrated movements to new situations and demands [1], [2], [3].

A commonly required adaptation is due to the necessity of executing a certain demonstrated movement faster or slower. For example, a robot playing table tennis has to hit the ball at different speeds than the ones of the movements demonstrated to the robot [2]. In a task such as golf, the necessity of adapting a certain movement in order to achieve new speeds also arises. Depending on the distance between the ball and the hole or on the shape and friction properties of the floor, a robot has to hit the ball faster or slower in order to sink it.

When executing a golf swing, a human normally moves the golf club first away from the ball and then towards it. In order to hit the ball twice as fast, the human does not need to move away from the ball twice as fast as well. Uniformly accelerating the whole movement would be energy inefficient. Therefore, adapting a golf swing to hit the ball with different speeds requires a non-uniform change in the speed of execution of the movement.

[1]Intelligent Autonomous Systems group, department of Computer Science, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {ewerton, maeda, peters}@ias.tu-darmstadt.de

[2]Computational Learning for Autonomous Systems group, department of Computer Science, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany geri@robot-learning.de

[3]Fachgebiet Regelungstechnik und Mechatronik, Technische Universität Darmstadt, Landgraf-Georg-Str. 4, 64283 Darmstadt, Germany vkisner@iat.tu-darmstadt.de

[4]Institut für Sportwissenschaft, Technische Universität Darmstadt, Magdalenenstr. 27, 64289 Darmstadt, Germany {kollegger, wiemeyer}@sport.tu-darmstadt.de

[5]Max Planck Institute for Intelligent Systems, Spemannstr. 38, 72076 Tuebingen, Germany jan.peters@tuebingen.mpg.de

This paper presents an approach to achieve such non-uniform changes in speed of execution in order to adapt a demonstrated movement to new situations and demands. This approach consists of optimizing with respect to a given reward multiple parameters that define the phase function associated to the movement. The phase function is a monotonically increasing function of time. It describes the speed of execution of the movement at each time step and its overall duration.

Fig. 1. Human demonstrating a putt to a robot via kinesthetic teaching.

The remainder of this paper is organized as follows: Section II presents related work. Section III explains our movement primitive formulation with multiple phase parameters. Section IV shows how to use a reinforcement learning algorithm to optimize the parameters of the phase function and the amplitude of a movement. Section V presents two sets of experiments. The first experiments explain the importance of non-uniformly adapting the phase function of a movement instead of simply adjusting some parameters that define its shape or uniformly rescaling its speed of execution. The second set of experiments has been conducted with an elastically actuated robot arm and shows how the proposed approach performs in practice at the task of adapting a demonstrated golf swing to achieve desired speeds at specific positions. Section VI presents conclusions and ideas for future work.

II. Related Work

A number of distinct movement representations have been proposed that allow for adapting demonstrations to new requirements. Dynamical movement primitives (DMPs) [5], for instance, allow for changing the goal, the speed and the duration of a movement while preserving its overall shape. The original DMP formulation, however, is not able to reach a certain goal with an arbitrary velocity, because it enforces zero velocity at the end of the movement.

Kober et al. [6] have designed an alternative DMP formulation to achieve arbitrary velocities at the end of the

movement, however, as pointed out by Mülling et al. [2], this solution is not necessarily accurate and may produce very large accelerations if the desired final position and the start position are very close to each other.

Mülling et al. [2] have used a two-stage movement primitive in order to achieve the desired velocity at an intermediary position along the movement. In a table tennis setup, the stage of the movement primitive is switched when the racket gets in contact with the ball. In our work, it is possible to specify desired velocities at intermediary positions using one single movement primitive with one single stage.

Nemec et al. [7] also proposed a method for adapting the velocity of motor skills learned from user demonstration. They have extended DMPs with a scaling factor $\nu(x)$, which is a function of the phase variable x. By changing $\nu(x)$, it is possible to accelerate or decelerate movements in order to satisfy for example certain contact forces or contact torques during the execution of a task. This acceleration or deceleration does not need to be uniform. In their method, the change in $\nu(x)$ depends on the difference between desired and actual forces and torques at successive time steps along the execution of the trajectory.

In [8], a similar approach has been adopted in order to speed up the movement of a robot carrying a cup without spilling the water inside it. In this approach, DMPs have been extended with a scaling factor $\nu(t)$, which allows for non-uniform change in speed. The change in $\nu(t)$ depends on an intermediate cost function with value γ if the water is about to be spilled or value 0 otherwise.

In contrast, our method deals with the problem of accelerating, decelerating or changing the amplitude of trajectories in order to satisfy velocity constraints. Our method allows for non-uniform change in velocity also if there is only a final reward, computed at the end of the execution of a trajectory.

In [4], van den Berg et al. showed how to speed up the movement of a surgical robot by increasing a factor s that rescales the time steps according to $\Delta t = \Delta t_0 / s$. By increasing s, the time steps get shorter. Since the number of time steps N is kept constant, the speed of the movement increases uniformly and its duration decreases. In our work, we deal with changing the speed of execution of a movement in a non-uniform fashion. As previously discussed, when adapting movements such as a golf putt swing to achieve new desired velocities at specific positions, a non-uniform change in the speed of execution is more suitable.

Englert et al. [9] presented a method for adapting online the path and the phase of a movement to events such as changes in the positions of obstacles and of the goal. Kim et al. [10] have also developed a method for adapting online the velocity of the movement of a robot to make it reach a certain position at a certain time, for example when catching objects on the fly. In contrast, we try to optimize the phase function and the amplitude of a movement according to a given reward, which can depend for instance on the difference between the achieved velocity and the desired velocity at a specific position.

In a remarkable work involving a minigolf setup, Kronander et al. [11] proposed a method to infer the hitting speed and hitting angle to sink the ball given multiple successful demonstrations. In our experiments, the amplitude of the movement and the phase function with multiple parameters have been optimized starting from only one demonstration.

III. Movement Primitive with Multiple Parameters for Shape and Phase

This section explains the proposed movement primitive formulation. This formulation comprises a set of parameters to define the shape of the movement in space and a set of parameters to define its speed profile and duration.

A. Shape Parameterization

The shape parameterization used in this work is basically the same one used in Probabilistic Movement Primitives (ProMPs) [12]. A trajectory $\boldsymbol{\tau} = [q_1, q_2, \cdots, q_T]^T$ comprising positions q_t sampled at a certain number T of time steps can be approximated by a weighted sum of normalized Gaussian basis functions ψ_n. This approximation can be expressed by

$$\boldsymbol{\tau} \approx \boldsymbol{\Psi} \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{w} = [w_1, w_2, \cdots, w_N]^T$ is the vector of weights w_n for each normalized Gaussian basis function ψ_n and

$$\boldsymbol{\Psi} = \begin{bmatrix} \psi_1(z(1)) & \psi_2(z(1)) & \cdots & \psi_N(z(1)) \\ \psi_1(z(2)) & \psi_2(z(2)) & \cdots & \psi_N(z(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(z(T)) & \psi_2(z(T)) & \cdots & \psi_N(z(T)) \end{bmatrix} \tag{2}$$

is a matrix with each ψ_n evaluated over the phase $z(t)$, which is a monotonically increasing function of time.

Given a trajectory $\boldsymbol{\tau}$ and a matrix of normalized Gaussian basis functions $\boldsymbol{\Psi}$, the vector of shape parameters $\boldsymbol{w}$ can be determined by

$$\boldsymbol{w} = \left(\boldsymbol{\Psi}^T \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^T \boldsymbol{\tau}, \tag{3}$$

which is the ordinary least squares solution for linear regression [13].

B. Phase Parameterization

The phase function $z(t)$ is a monotonically increasing function of time assuming non-negative real values between 0 and Z. In this work, $Z = 100$ has been chosen. When $z(t) = 0$, the movement is just starting. When $z(t) = Z$, the movement is over. We propose defining the phase function with a few parameters that will be useful to manipulate the evolution of a movement in time, as the shape parameters w_n allow for manipulating the trajectory of a movement in space.

The rate of change of phase in relation to time can be written as the vector $\dot{\boldsymbol{z}} = [\dot{z}(1), \dot{z}(2), \cdots, \dot{z}(T-1)]^T$. In fact, we parameterize $\dot{\boldsymbol{z}}$ according to

$$\dot{\boldsymbol{z}} = \boldsymbol{\Phi} \boldsymbol{\alpha}. \tag{4}$$

The matrix $\boldsymbol{\Phi}$ comprises M normalized Gaussian basis functions evaluated from time step $t = 1$ until time step

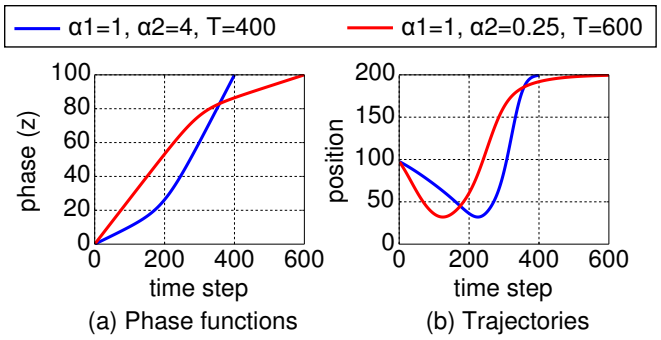

Fig. 2. Effects of changes in phase parameters. (a) Two different phase functions. The blue one has phase parameters $\alpha_1 = 1$, $\alpha_2 = 4$ and $T = 400$. The red one has phase parameters $\alpha_1 = 1$, $\alpha_2 = 0.25$ and $T = 600$. (b) Respective trajectories.

$t = T - 1$, where T is the total number of time steps of the movement. This matrix can be written as

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1(1) & \phi_2(1) & \cdots & \phi_M(1) \\ \phi_1(2) & \phi_2(2) & \cdots & \phi_M(2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(T-1) & \phi_2(T-1) & \cdots & \phi_M(T-1) \end{bmatrix}. \quad (5)$$

The vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_M]^T$ contains the weights α_m for each basis function ϕ_m. Those weights are such that $\alpha_m > 0, \forall m \in \{1, 2, \cdots, M\}$, ensuring that the phase z always increases with time.

Once $\dot{\boldsymbol{z}}$ has been defined according to (4), the phase $z(t)$ can be computed via Euler integration with $z(0) = 0$ and $z(t+1) = z(t) + \Delta t \dot{z}(t)$ until $t = T - 1$.

In order to ensure that the phase achieves its maximum value exactly at the last time step of the movement, i.e. $z(T) = Z$, the phase is normalized with

$$\boldsymbol{z}_{\text{norm}} = \frac{\boldsymbol{z}}{z(T)} Z. \quad (6)$$

In summary, the weights α_m and the total number of time steps T are the phase parameters. They define the rate of change $\dot{z}$ of the phase in relation to time, which, via Euler integration, defines the phase z of a movement.

Fig. 2 shows two phase functions. Each of them was defined using two normalized Gaussian basis functions $\phi(t)$. One Gaussian is centered at $t = 0$, the other at $t = T$. The variance of those Gaussians is $30 \times T$. The number of time steps T is 400 for the phase depicted in blue and 600 for the phase depicted in red. Observe how the proportion between the parameters α_m influence the phase function and consequently the evolution in time of the trajectories generated with the same shape parameters $\boldsymbol{w}$. The first half of the blue trajectory is slow compared to the second half, because $\alpha_1 < \alpha_2$. The opposite happens in the red trajectory.

In [14], we proposed a similar phase function parameterization as in this paper. In that formulation, it was necessary to define the maximum number of time steps of the demonstrated movements. By treating T as a phase parameter and applying the normalization described by (6) we have eliminated in this new formulation the necessity of defining a maximum number of time steps.

IV. Reinforcement Learning of Movement Amplitude and Phase

This section explains how to use reinforcement learning by Reward-weighted Regression (RWR) [15] in order to optimize the amplitude and the phase parameters of a movement with the objective of achieving a desired velocity at a specific position.

The reward function has been defined as

$$R = \exp(-\beta \|\dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}^*\|), \quad (7)$$

where β is a task specific parameter tuned by the user and $\|\dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}^*\|$ is the L^2 norm of the difference between $\dot{\boldsymbol{v}}$ and $\dot{\boldsymbol{v}}^*$. The term $\dot{\boldsymbol{v}}$ is the actual velocity at the specific position $\boldsymbol{q}^*$, while the term $\dot{\boldsymbol{v}}^*$ is the desired velocity at this same specific position. Velocities have been computed using the finite difference approximation

$$\dot{\boldsymbol{q}}(t) = \frac{\boldsymbol{q}(t+1) - \boldsymbol{q}(t)}{\Delta t}. \quad (8)$$

The term β has been set equal to 10 in all our experiments, because this value led to achieving the desired speed with high precision within a reasonable number of reinforcement learning iterations, compared to the values 0.1, 1 and 100.

The parameters that need to be optimized are given by the vector $\boldsymbol{\theta} = [\alpha_2, \alpha_3, \cdots, \alpha_M, T, \lambda]^T$. Due to the normalization (6), not the absolute values of the parameters α_m, but the proportions between their values is important. Therefore, α_1 is simply always equal to 1 and it is not necessary to optimize it. The scalar λ multiplies all the shape parameters w_n and determines the amplitude of the movement.

Let us adopt an upper-level policy given by a Gaussian distribution $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. Our objective is thus to maximize the expected reward with respect to $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$. This can be done iteratively according to

$$\{\boldsymbol{\mu}_{\boldsymbol{\theta}}^{k+1}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{k+1}\} = \underset{\{\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\}}{\arg\max} \sum_{i=1}^{S} R_i \mathcal{N}(\boldsymbol{\theta}_i; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}), \quad (9)$$

where S is the number of sampled parameter vectors $\boldsymbol{\theta}_i$ from the previous policy $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}^k, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^k)$.

The solution of (9) is given by

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{k+1} = \frac{\sum_{i=1}^{S} R_i \boldsymbol{\theta}_i}{\sum_{i=1}^{S} R_i}, \quad (10)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{k+1} = \frac{\sum_{i=1}^{S} R_i (\boldsymbol{\theta}_i - \boldsymbol{\mu}_{\boldsymbol{\theta}}^k)(\boldsymbol{\theta}_i - \boldsymbol{\mu}_{\boldsymbol{\theta}}^k)^T}{\sum_{i=1}^{S} R_i}. \quad (11)$$

This iterative process continues until the expected reward converges. The best parameters are then given by $\boldsymbol{\mu}_{\boldsymbol{\theta}}$.

The convergence properties of the Expectation-Maximization (EM) algorithm [16] guarantee that the reward converges to some local maximum. The initial parameters $\boldsymbol{\mu}_{\boldsymbol{\theta}}^0$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^0$ influence the maximum expected reward achieved by RWR. As it will be explained in

Section V, those parameters have been initialized in our work by using a human demonstration and choosing an exploration noise.

V. Experiments

This section presents a number of experiments in which the proposed movement primitive formulation with multiple phase parameters has been applied. In these experiments, the BioRob, a robot arm consisting of four elastically actuated joints [17], has been used.

In the first experiments, we have assumed that the desired trajectory could be perfectly tracked by the robot. Using this simplifying assumption, a comparison between optimizing in simulation different parameters has been made. Afterwards, both the phase parameters and the amplitude of a putt swing have been optimized such that the real robot arm could pass through a specific position with a desired velocity.

A. Comparison between optimizing different parameters

In these experiments, a putt swing was demonstrated to the robot via kinesthetic teaching as depicted in Fig. 1. Subsequently, the robot tried to track the demonstrated trajectory[1]. The trajectory executed by the robot was then recorded. It comprises 5959 positions[2] sampled at regular intervals of $1/480s$. We refer to this recorded trajectory as the "original trajectory".

The shape of the original trajectory has been parameterized according to (1). Twenty normalized Gaussian basis functions $\psi(z)$ have been used. These basis functions have variance equal to 100 and their means are evenly distributed between $z = 0$ and $z = 100$. In a first experiment, the phase function was defined as $z(t) = \alpha t$, where $\alpha = 100/T$. The vectors of shape parameters $\boldsymbol{w}$ for each of the four DoFs of the robot have been determined with (3).

The shape parameters have then been optimized according to (9), using however $\boldsymbol{\theta} = \boldsymbol{w}$, to generate a trajectory in which the double of the original velocity of the 1^{st} joint is achieved when this joint crosses position $0.1\,\mathrm{rad}$ for the first time. The desired velocity for the other three joints is the same as the original velocity and is close to zero. Our algorithm iterated over equations (10) and (11) 100 times. In each iteration, fifty vectors of parameters $\boldsymbol{\theta}$ were sampled. Results are depicted in Fig. 3. The plot on the left shows the time at which the original trajectory and the trajectory after optimization cross the joint position $0.1\,\mathrm{rad}$ for the first time. The plot on the right shows the joint velocity of both trajectories. The shape of the trajectory has changed considerably. Such changes in shape may result in unnecessary movements, collisions, reaching joint limits, etc.

[1]The control of the robot has been performed using a PD controller for joint and motor angles with compensation for stiction and gravity. System identification methods have been used to estimate gravitational torques, elastic transmission and stiction.

[2]Cubic spline interpolation has been used to compress the original trajectory to 500 time steps before determining the parameters of the movement primitive and running reinforcement learning. The optimized trajectory is then decompressed also with cubic spline interpolation in the experiments where the real robot executes the trajectories.

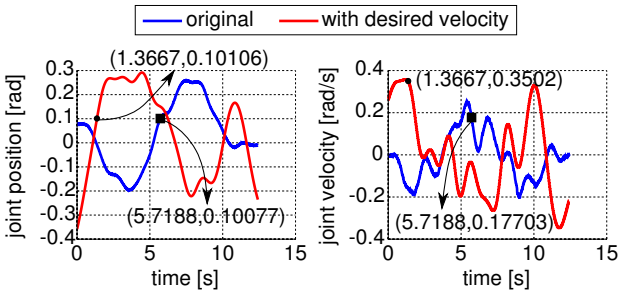

Fig. 3. Optimizing only shape parameters to achieve double the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\mathrm{rad}$ for the first time. Assuming the desired trajectory could be exactly tracked by the robot.

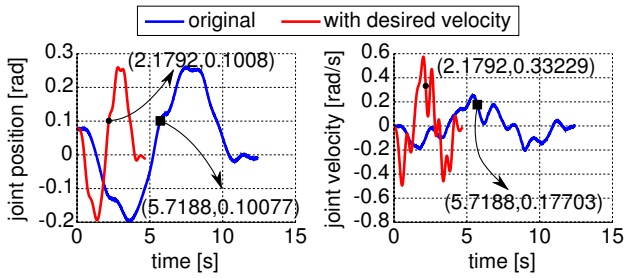

Fig. 4. Optimizing a phase function of the form $z(t) = \alpha t$ to achieve double the velocity of the original movement when the 1^{st} joint angle crosses the value $0.1\,\mathrm{rad}$ for the first time. Assuming the desired trajectory could be exactly tracked by the robot.

Further constraints would be required in order to keep the shape of the movement similar to the original one when optimizing only the shape parameter vectors $\boldsymbol{w}$ as in this example.

By optimizing not the shape parameters, but only the phase parameter α, assuming $z(t) = \alpha t$ and $\alpha = 100/T$, we have obtained solutions as the one depicted in Fig. 4. This solution rescales the velocity of the movement uniformly and results in unnecessary accelerations. Speeds are considerably higher than the original ones also when the joint position is far away from the position of interest.

Finally, the same experiment was performed optimizing multiple phase parameters and the amplitude of the movement, as in Section IV. The shape parameterization is the same as the one in the previous experiment. The phase function has been defined as in Section III-B, with nine Gaussian basis functions $\phi(t)$. The basis functions $\phi(t)$ have variance equal to the duration T and their means are equally distributed between $t = 0$ and $t = T$. The upper-level policy $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ was initialized with mean

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{0} = \left[\alpha_2 = 1, \alpha_3 = 1, \cdots, \alpha_9 = 1, T = 500, \lambda = 1\right]^T$$

and a diagonal covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{0}$ with its diagonal defined by the vector

$$\left[\sigma_{\alpha_2}^2 = 10, \cdots, \sigma_{\alpha_9}^2 = 10, \sigma_T^2 = 1000, \sigma_\lambda^2 = 0.1\right].$$

The elements $\sigma_{\alpha_m}^2$ represent the variance of the parameters α_m. The element σ_T^2 represents the variance of the duration

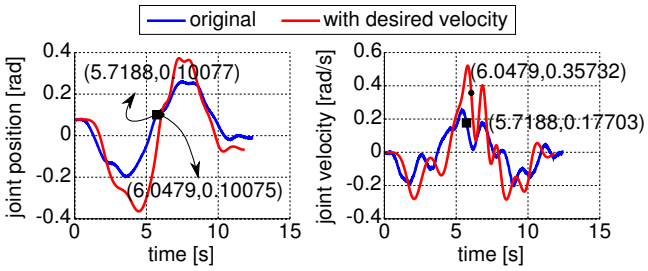

Fig. 5. Optimizing multiple phase parameters and amplitude to achieve double the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. Assuming the desired trajectory could be exactly tracked by the robot.

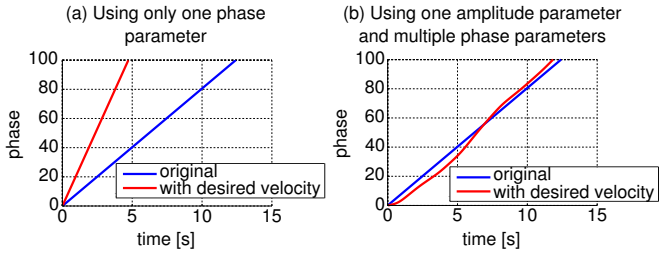

Fig. 6. Phase functions before and after optimization. (a) Phase functions of the form $z(t) = \alpha t$. (b) Phase functions with multiple phase parameters.

T of the movement. Finally, σ_λ^2 represents the variance of the amplitude parameter λ.

Fig. 5 depicts the results of optimizing the phase parameters and the amplitude parameter after 100 iterations with 50 samples $\boldsymbol{\theta}_i$ each. The optimized trajectory has shape similar to the original one, except for the change in its amplitude. Velocities are not much higher than original ones far from the position of interest. Fig. 6 shows the phase functions before and after optimization with one and multiple phase parameters.

B. Optimizing trajectories executed by the robot

In these experiments, the same Gaussian basis functions for shape and phase were used as in Section V-A. The upper level policy $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$ was initialized with the same mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}^0$ as in Section V-A. The initial covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^0$ was again a diagonal matrix and its diagonal was defined by the vector

$$\left[\sigma_{\alpha_2}^2 = 10, \cdots, \sigma_{\alpha_9}^2 = 10, \sigma_T^2 = 50000, \sigma_\lambda^2 = 0.25\right]^T.$$

In a first experiment, the phase parameters were optimized to reach the half of the original velocity when the 1^{st} joint crosses position $0.1\,\text{rad}$ for the first time. In a second experiment, those same parameters were optimized to reach the double of the original velocity when the 1^{st} joint crosses this same specific position for the first time. In both experiments, there were 20 reinforcement learning iterations with 30 samples $\boldsymbol{\theta}_i$ each[3]. Fig. 7 shows how the

[3]The experiments with the real robot were time-consuming, since the robot took approximately 32 seconds on average to position the golf club and perform the putt swing. This motivated the choice of running 20 iterations with 30 samples $\boldsymbol{\theta}_i$ each.

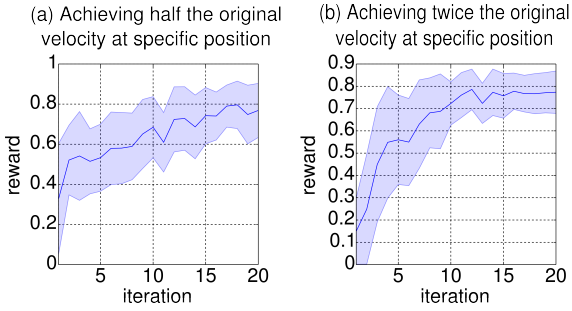

Fig. 7. Iteration of reinforcement learning algorithm versus expected reward with mean and standard deviation. (a) Evolution of the expected reward by trying to achieve half the original velocity at position $0.1\,\text{rad}$. (b) Evolution of expected reward by trying to achieve twice the original velocity at position $0.1\,\text{rad}$.

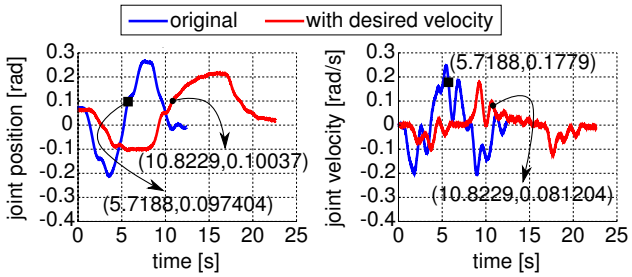

Fig. 8. Optimizing phase parameters and amplitude to achieve half the velocity of an original movement when the 1^{st} joint angle crosses the value $0.1\,\text{rad}$ for the first time. In this case, trajectories executed by the real robot have been evaluated.

expected reward changed with the number of iterations of the reinforcement learning algorithm. Figs. 8 and 9 show the trajectories found by the reinforcement learning algorithm that achieve the desired velocity in comparison to the original trajectories.

In the solution for achieving half the original speed at a certain position, the overall duration of the movement is considerably longer than the original duration. The change in velocity is non-uniform, as can be clearly observed by the different slopes along the red curve in Fig. 10a.

In the solution for achieving twice the original speed at a certain position, the overall duration of the movement is slightly longer than the original duration. However, the movement accelerates in between, producing the desired velocity. This acceleration can be noticed by the slight change in slope along the red curve in Fig. 10b.

VI. Conclusion and Future Work

This paper presented a movement primitive formulation with multiple phase parameters. Changes in these phase parameters allow for non-uniform acceleration or deceleration of a movement. We have shown that, using a reinforcement learning algorithm, it is possible to adapt the phase parameters and the amplitude of a movement demonstrated by a human in order to satisfy new velocity constraints. This was demonstrated in practice with experiments where an elastically actuated robot arm learned how to execute a golf putt swing, achieving new desired velocities at a specific

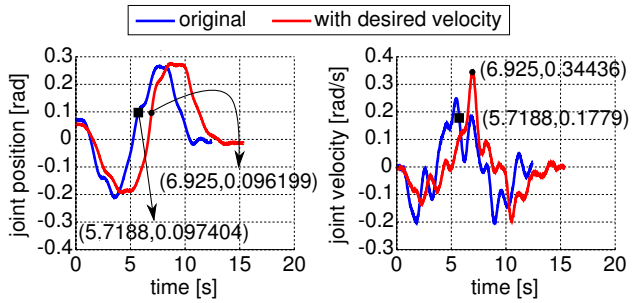

Fig. 9. Optimizing phase parameters and amplitude to achieve double the velocity of an original movement when the 1^{st} joint angle crosses the value 0.1 rad for the first time. In this case, trajectories executed by the real robot have been evaluated.

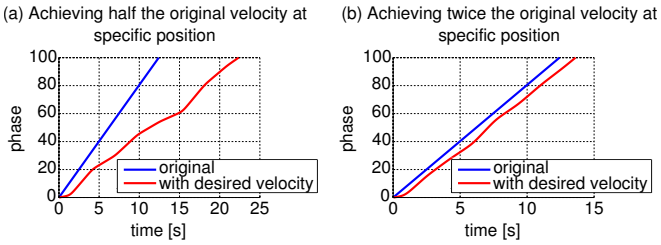

Fig. 10. Phase functions before and after optimization. The blue curve represents the original phase function, which was defined with constant slope. (a) The red curve represents the phase function after optimization to achieve half the original velocity when the 1^{st} joint reaches 0.1 rad for the first time. (b) The red curve represents the phase function after optimization to achieve double the original velocity when the 1^{st} joint reaches 0.1 rad for the first time.

position.

Experiments with more sophisticated reward functions might be performed with the real robot. For example, the reward function could favor energy-efficient movements. Our phase function with multiple parameters might play an important role in this case, since it allows for non-uniform changes in speed of execution. Unnecessary accelerations or decelerations could then be avoided.

In the future, we will use a camera to detect the ball and a reward will be assigned to the movement of the robot depending on the final distance between the ball and the hole. Furthermore, we will evaluate the applicability of our movement primitive formulation to other tasks such as throwing or catching an object, moving objects around in an energy-efficient and safe way, etc.

So far in our experiments, only a small change in the shape of the movement through an amplitude parameter has been allowed. In a future work, we intend to allow for more general changes in shape alongside changes in phase in order to let the robot adapt to new constraints in space as well.

VII. Acknowledgments

The research leading to these results has received funding from the project BIMROB of the "Forum für interdisziplinäre Forschung" (FiF) of the TU Darmstadt and from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand).

The authors would also like to thank Rudolf Lioutikov, Hany Abdulsamad, Alexandros Paraschos and Elmar Rueckert. Their help was essential for us to perform the real robot experiments.

We are also indebted to Jérôme Kirchhoff, who has provided us with a great technical support, and to Josef Noll, who has made an excellent work in building a support to attach the golf club to the end effector of the BioRob.

References

[1] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 2, pp. 286–298, 2007.

[2] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[3] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*, 2013.

[4] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2074–2081.

[5] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1398–1403.

[6] J. Kober, K. Mulling, O. Kromer, C. Lampert, B. Scholkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 853–858.

[7] B. Nemec, A. Gams, and A. Ude, "Velocity adaptation for self-improvement of skills learned from user demonstrations," in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE, 2013, pp. 423–428.

[8] R. Vuga, B. Nemec, and A. Ude, "Speed profile optimization through directed explorative learning," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 547–553.

[9] P. Englert and M. Toussaint, "Reactive phase and task space adaptation for robust motion execution," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 109–116.

[10] S. Kim, E. Gribovskaya, and A. Billard, "Learning motion dynamics to catch a moving object," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*. IEEE, 2010, pp. 106–111.

[11] K. Kronander, M. S. Khansari-Zadeh, and A. Billard, "Learning to control planar hitting motions in a minigolf-like task," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 710–717.

[12] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2616–2624.

[13] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.

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

[15] J. Peters and S. Schaal, "Reinforcement learning by reward-weighted regression for operational space control," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 745–750.

[16] C. J. Wu, "On the convergence properties of the em algorithm," *The Annals of statistics*, pp. 95–103, 1983.

[17] T. Lens and O. von Stryk, "Design and dynamics model of a lightweight series elastic tendon-driven robot arm," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4512–4518.