Robotica (2016) volume 34, pp. 2806–2822. © Cambridge University Press 2015
doi:10.1017/S0263574715000405

Speed adaptation for self-improvement of skills learned from user demonstrations

Rok Vuga*, Bojan Nemec and Aleš Ude†

Humanoid and Cognitive Robotics Lab, Department of Automatics, Biocybernetics and Robotics, Jožef Stean Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia.
E-mails: bojan.nemec@ijs.si, ales.ude@ijs.si

(Accepted May 12, 2015. First published online: June 15, 2015)

SUMMARY
The paper addresses the problem of speed adaptation of movements subject to environmental constraints. Our approach relies on a novel formulation of velocity profiles as an extension of dynamic movement primitives (DMP). The framework allows for compact representation of non-uniformly accelerated motion as well as simple modulation of the movement parameters. In the paper, we evaluate two model free methods by which optimal parameters can be obtained: iterative learning control (ILC) and policy search based reinforcement learning (RL). The applicability of each method is discussed and evaluated on two distinct cases, which are hard to model using standard techniques. The first deals with hard contacts with the environment while the second process involves liquid dynamics. We find ILC to be very efficient in cases where task parameters can be easily described with an error function. On the other hand, RL has stronger convergence properties and can therefore provide a solution in the general case.

1. Introduction

Autonomous learning, adaptation and generalization are among the main issues that need to be solved by contemporary robotics to widen the applicability of robotic systems. In this study, we deal with the problem of how a robot can autonomously improve its motor skills, taking example from how humans improve their motor skill knowledge. Human motor skill learning often starts with a teacher demonstrating a skill, usually slowly, and then the subject tries to repeat it. The first few attempts might be very slow. After that, we generally try to improve the learned skill and make the execution faster. During this process of improvement, which is still essentially learning, we adapt the motion pattern to our own dynamic capabilities.[2] However, speeding up is not done uniformly. Some parts of the motion pattern can be accelerated more, some less, depending on our own kinematics and dynamics as well as environment constraints. Learning in robot domain can follow a similar pattern of observation and practicing.[3]

Even if the human demonstration is performed with natural speed, the captured solution might not be the optimal one, as the kinematic and dynamic properties of the teacher and the learner are not the same.[4] Therefore, just like with humans, adaptation of the learned movement is needed. While many works deal with adaptation of complete robot trajectories[5–9] we focus on speed optimization only.

Speed optimization has been the subject of many works since the beginning of robotics. One of the early examples is the work by Hollerbach,[10] which deals with uniform time scaling to make trajectories feasible given joint torque constraints. Similarly, Bobrow *et al.*[11] and Shin *et al.*[12] studied optimal control of serial manipulators following a given path by applying bang-bang control. McCarthy and Bobrow[13] extended the approach to more general cases including constraints on the motion of the robot, for example tasks comprised of contacts with the environment and solved them using linear programming. Žlajpah[14] presents a solution using a forward–backward stepping algorithm. Dahl and

* Corresponding author. E-mail: rok.vuga@ijs.si
† Initial results on the topic were presented at the IEEE-RAS International Conference on Humanoid Robots (Humanoids 2013), Atlanta, Georgia.[1]

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

Nilsen[15,16] propose on-line feedback control to deal with errors in modeled dynamics, while Kieffer *et al.*[17] present a method to assure path tracking to the specified tolerance. More recently, Akella and Peng[18] investigated the problem of multiple robots sharing a workspace. They used uniform time scaling to obtain optimal speed profiles, taking into account collision avoidance between the robots as well as their dynamic constraints. Michna *et al.*[19] also optimize speed profiles with collision avoidance and dynamic constraints, applying their work to trajectory planning for mobile robots. Finally, Zhao and Tsiotras[20] investigate the link between minimum-time and minimum-energy solutions of the speed optimization problem.

Essentially all the proposed methods rely on good dynamic models of the robot and the environment. The issue of how to address environmental constraints when environmental models are unavailable remains unsolved. In this paper, we propose an approach that can deal with speed optimization in such cases. Speed profile optimization is governed by an appropriate cost function, which is much easier to obtain than environmental models. As the underlying policy representation, we apply DMP framework[21] and propose a novel formulation that enables non-uniform scaling of speed. For optimization of the policy parameters, we evaluated two techniques: ILC and RL. The former exploits feedback error control law and is useful in cases where enough information about the task is available for an error function to be designed. On the other hand, RL possesses strong convergence properties and can optimize more general cost criteria. We discuss and propose how such criteria can be designed in both cases in order to facilitate speed adaptation.

The proposed methodology and algorithms were evaluated in two experiments from domestic environments. First, we considered speeding up the assembly of a cake decoration tool. In the second experiment, the robot accelerated the carrying of a cup of liquid along a specified path without spilling. The experiments were selected to demonstrate the approach in completely different scenarios. In the first experiment, the robot deals with a process involving hard contacts with the environment. The second experiment is characterized by liquid dynamics. The common feature of the experiments is that they would be very hard to solve using classical, model based approaches.

The paper is organized as follows. In Section 2, we briefly outline the framework of DMP to represent motion patterns describing the desired policy. We extend the DMP framework in order to enable non-uniform speed scaling. Learning of the appropriate speed profile is considered in Section 3, where we propose two new methods for speed profile optimization: one based on ILC and the second based on RL. Experimental results in Section 4 demonstrate the effectiveness of the proposed approaches applied in typical tasks that a humanoid robot might perform in a kitchen environment. Finally, we conclude with a brief discussion on advantages and disadvantages of both approaches.

2. DMPs with Non-Uniform Velocity Scaling

2.1. The original DMP formulation

Many modern policy improvement and adaptation techniques rely on parametric policy description, where a movement trajectory is compactly encoded by a limited set of parameters. This allows representation of complex motor actions that can be flexibly adjusted in simple and intuitive way by modulation of the relatively small number of parameters. A widely used parameterization, the DMP, was proposed by Ijspeert *et al.*.[21,22] Within this framework, the trajectory of every degree of freedom is defined as a response of a dynamic system. For point-to-point movements (also referred to as discrete movements), the dynamic system is specified with the following set of differential equations:

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

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

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \; \Psi_i(x) = \exp\left(-h_i\,(x - c_i)^2\right) \tag{3}$$

$$\tau \dot{x} = -\alpha_x x. \tag{4}$$

Here, y is the variable describing a one-dimensional trajectory, z is an auxiliary variable, and x is the phase variable. This way, y is defined as a response of a second order system, the dynamics of

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

which are governed by constants α_z and β_z. In case of $\alpha = 4\beta$, the response of the system is critically damped, such that y converges from its starting value towards g, which defines the goal. The term $f(x)$ is used to define the more complex path we usually want y to take. It is defined as a weighted combination of non-linear Gaussian kernel functions and is superimposed onto the original second order response. The values of weights w_i thus define the shape of the trajectory y will take while converging towards the goal g.

Equation (4) defines the so-called canonical system. It defines the phase variable x, which serves as a replacement of time. Unlike the dynamical system (1)–(2), the canonical system is a first order system. Starting from an arbitrary starting state, typically $x_0 = 1$, x monotonically converges towards zero as time progresses. This way, we can see why x can be thought of as phase of the movement. The value of $x = 1$ marks the beginning, whereas the case when $x \approx 0$ marks the end of the movement.

The intuition behind the Eqs. (1)–(4) is that the critically damped behavior of the system ensures convergence of y towards the attractor point g as the phase x converges towards zero. This way, g represents the final point of the trajectory. Between $x = 1$ and $x \approx 0$ however, the term $f(x)$ is dominant. Therefore, $f(x)$ describes the shape of the trajectory taken by y. All the open parameters can be independently modulated: changing g will change the final point of the trajectory, whereas changing any of the weights w_i will change its shape. Finally, the parameter τ defines the speed with which the canonical system converges. Concretely, τ is the amount of time it takes for the phase x to become almost zero and thus the trajectory y to reach the goal g.

Note that, y represents a one-dimensional trajectory. To encode a multi-dimensional (either Cartesian or joint space) trajectory, each degree of freedom has to be represented with a separate dynamical system. The phase of the movement is common to all degrees of freedom; only one canonical system is thus needed. Consequently, every DOF has their own shape parameters w_i^d and goal g_d, but the duration τ is common for all.

Given a (possibly multi-dimensional) trajectory, it is necessary to calculate the set of parameters which encode it in the DMP representation. Let us assume that the desired trajectory has been stored as a sequence

$$\{y_{\mathrm{d}}(t_k), \dot{y}_{\mathrm{d}}(t_k), \ddot{y}_{\mathrm{d}}(t_k)\}_{k=1}^{T}, \tag{5}$$

where $y_{\mathrm{d}}(t_k)$, $\dot{y}_{\mathrm{d}}(t_k)$, $\ddot{y}_{\mathrm{d}}(t_k)$ are respectively the measured positions, velocities, and accelerations on the observed trajectory and T is the number of sampling points on the trajectory. By rewriting (1)–(2) as a second order differential equation

$$\tau^2 \ddot{y} + \alpha_z \tau \dot{y} - \alpha_z \beta_z (g - y) = f(x), \tag{6}$$

we obtain one linear equation in $\{w_i\}$ for each data point k in the sequence (5), i.e.,

$$\tau^2 \ddot{y}_{\mathrm{d}}(t_k) + \alpha_z \tau \dot{y}_{\mathrm{d}}(t_k) - \alpha_z \beta_z (g - y_{\mathrm{d}}(t_k)) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x(t_k))}{\sum_{i=1}^{N} \Psi_i(x(t_k))} x(t_k), \tag{7}$$

where $x(t_k)$ are obtained by integrating Eq. (4), which results in $x(t_k) = \exp(-\alpha_x t_k/\tau)$. Weights w_i are thus estimated by solving the overdetermined equation system (7) for all $k = 1, \ldots, T$. Finally, we set $\tau = t_T$ and $g_d = y_d(T)$. Repeating the process for every degree of freedom d, we obtain the parameters of the DMP corresponding to the training trajectory (5).

2.2. Our approach: parameterized velocity scaling

The main goal of our approach is to modulate the speed of the trajectory without affecting their shape. In the original DMP representation, the dynamical system is already uncoupled from time and instead dependent on the phase x. However, the speed of the trajectory is inherited from the combination of the second order response and the values of the weights w_i. It is not possible to modify the speed of the movement without modifying its shape.

In order to be able to modify the speed independently, we include an additional, phase-dependent temporal scaling factor into the DMP equations. The original DMP equations (1)–(4) are extended

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

with the additional temporal scaling factor ν as follows:

$$\tau \dot{z} = \nu(x)(\alpha_z(\beta_z(g - y) - z) + f(x)), \tag{8}$$

$$\tau \dot{y} = \nu(x)z, \tag{9}$$

$$\tau \dot{x} = -\nu(x)\alpha_x x. \tag{10}$$

Note that, we still keep the constant temporal scaling factor τ to preserve the ease of uniform time scaling. Smaller ν results in a slower motion while larger ν increases the speed of motion. The phase-dependent factor ν is parameterized as a weighted combination of M radial basis functions

$$\nu(x) = 1 + \frac{\sum_{j=1}^{M} \theta_j \Psi_j(x)}{\sum_{j=1}^{M} \Psi_j(x)} = 1 + \boldsymbol{\theta}^T \boldsymbol{\Psi}(x), \tag{11}$$

$$\boldsymbol{\theta} = \left[\theta_1, \ldots, \theta_M\right]^T, \tag{12}$$

$$\boldsymbol{\Psi}(x) = \frac{1}{\sum_{j=1}^{M} \Psi_j(x)} \left[\Psi_1(x), \ldots, \Psi_M(x)\right]^T. \tag{13}$$

Just like τ, the scaling factor ν should be the same for all of the robot's degrees of freedom in order to keep them synchronized. The modulation of kernel weights θ_j allows for a non-uniform modulation of the speed profile in the same way as the modulation of weights w_i in Eq. (3) allows for modulation of the shape of the desired trajectory. Initially, the weights θ_j for ν are set to zero, so that $\nu(x) = 1, \ \forall x$. This corresponds to the movement at the original speed. Thus, no re-training of the initial DMP (1)–(4) is needed.

To gain an intuitive insight, you can compare the effect of the scaling factor ν to the constant τ. Setting ν to a very small value has the same effect as setting τ to a very large value. This means that the movement slows down. On the other hand, lowering the value of τ or increasing ν corresponds the movement being performed faster and the goal reached sooner.

This way, we extended the DMP representation to include another set of open parameters, modulation of which changes the speed profile of the represented trajectory. In the following section, we describe how the newly included parameters can be optimized for a given task and thus an optimal speed profile autonomously obtained.

3. Learning of the Speed Profile

The goal of modifying the speed profile is to accelerate the task execution, thus assuring a quicker accomplishment of the task. We assume that initially a joint or Cartesian space trajectory, which successfully accomplishes the task, has been provided and encoded as a DMP. The main idea is simple: increase the speed of task execution until some essential task parameters are violated. These depend on the task and could be anything from excessive forces or joint speeds to more abstract criteria such as spillage of carried liquid. These parameters define the constraints of the task; their violation indicates failure.

This type of optimization can be achieved using iterative learning methods, which operate by improving the performance of the trajectory execution with respect to a selected cost or error function. In order to achieve good results, it is essential for this criteria to be appropriately defined. In this paper, we present a unified design of the cost criteria for ILC and RL. The proposed criteria stimulates fast execution while preventing violation of the imposed constraints.

3.1. Speed profile adaptation using ILC-based approach

A suitable framework for solving the above problem is provided by ILC.[23,24] ILC optimizes the skills by repeating the same action over and over again, which is similar to how humans learn difficult motor skills. Speed adaptation can be cast as a feedback control problem by using the following update rule

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

for the temporal scaling factor:

$$\nu_{j+1}(x) = \kappa(\nu_j(x) + \eta_1 e_j(x + \delta_x)), \tag{14}$$

where x denotes the phase at time t, j is the learning iteration index and κ and η_1 are the ILC gains. At every iteration, error recorded during the previous trial is used to correct the movement. Note that the error is temporally shifted, so that the correction at phase x is calculated from the error recorded at phase $x + \delta_x$. This way, the system anticipates the deviation from the desired behavior and corrects it before it occurs, which is the main advantage of ILC over ordinary feedback control. δ_x denotes the magnitude of the step in the phase domain from which the prediction is calculated. It can be calculated from (10) as follows

$$\delta_{x_k} = -\frac{\alpha_x \nu_j(x)}{\tau} x \delta_t, \tag{15}$$

where δ_t is the user chosen parameter which corresponds to the constant prediction step in the time domain.

The formulation (14) assures that ν converges to the equilibrium where the feedback error function e becomes approximately 0 over the whole duration of motion. An appropriate error function in the case of speed adaptation is in the form

$$e(x) = \xi_\nu(\nu_{\max} - \nu(x)) - \xi_d b(x), \tag{16}$$

where ν is the temporal scaling function defined in (11), b is a scalar function quantifying deviations from prespecified task constraints, $\nu_{\max} > 0$ is the upper bound for the temporal scaling function, x is the phase along the trajectory at time t, $x = x(t)$ and ξ_ν, ξ_d are the prespecified weighting factors.

The function is designed in such a way that, as long as $b = 0$, the controlled system converges to the maximum allowed speed $\nu = \nu_{\max}$. The term b specifies response to violation of task constraints and causes the execution at the corresponding phase to slow down. It needs to be defined so that values $b > 0$ proportionally indicate breaching of the constraints. For example, it might be anything from the norm of excessive forces and torques as in our assembly experiment to a signal indicating the spilling of liquid as in our liquid carrying experiment.

Standard ILC assumptions include: (1) Stable system dynamics, (2) System returns to the same initial conditions at the start of each trial, (3) Each trial has the same length. The third assumption is problematic in our case because we change the execution speed, therefore each trial has a different duration. However, we defined the temporal scaling factor ν as a function of phase in Eq. (10), thus we can sample ν in the phase space and the same number of times in every trial.

To obtain the speed profile in a parameterized form (11), we first define

$$r(x) = \nu(x) - 1. \tag{17}$$

We then need to solve

$$\mathbf{B}\boldsymbol{\theta} = \mathbf{r}, \tag{18}$$

with

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} r(x_1) \\ \vdots \\ r(x_T) \end{bmatrix}, \tag{19}$$

and the system matrix $\mathbf{B} \in \mathbb{R}^{T \times M}$ defined as

$$\mathbf{B} = \begin{bmatrix} \Psi_1(x_1) & \cdots & \Psi_M(x_1) \\ \vdots & \vdots & \vdots \\ \Psi_1(x_T) & \cdots & \Psi_M(x_T) \end{bmatrix}.$$

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

Solution for the above system in the least square sense is obtained by

$$\boldsymbol{\theta} = \mathbf{B}^{+}\mathbf{r}, \tag{20}$$

where $\mathbf{B}^{+}$ denotes the Moore–Penrose pseudo-inverse of $\mathbf{B}$. Such parameterization ensures smoothing of the resulting speed profile, depending on the chosen number of the kernel functions M.

3.2. Speed profile adaptation by reinforcement learning

In the previous section, we showed how to learn the temporal scaling factor ν as continuous function of phase x using ILC. Alternatively, ν can be obtained by RL.

RL can be used to optimize control policies with respect to a specified reward or cost criteria. Roughly, there are two groups of RL algorithms that are commonly used in robotics: value-function approaches and policy search.[25] Value function approaches, where the learning problem is formulated as a Markov decision process, aim to first calculate the benefit of visiting individual states and then derive the control policy based on that. On the other hand, policy search algorithms optimize the control policy directly, for example by estimating the gradient of the cost function with respect to the available policy parameters. Kober *et al.*[25] provide a comprehensive overview of RL and its use in robotics.

In the case of speed adaptation, the speed profile ν is represented as a parameterized policy with parameters $\boldsymbol{\theta}$ and basis functions $\boldsymbol{\Psi}$ (see Eq. (11)). Policy search methods are therefore a perfect match for our problem.

We define the cost function for the speed adaptation problem as follows

$$r(x) = b(x), \tag{21}$$

$$r_T = \xi_T T. \tag{22}$$

Analogous to error function (16) in the ILC case, the cost collected during the movement execution depends on the term b, which accounts for the violation of task constraints. On the other hand, assessment of execution speed is done by the so-called terminal cost r_T. While intermediate rewards $r(x)$ are collected and assigned over the whole execution phase, the terminal cost is given when the system reaches its final state, that is, at the end of each trial. By setting $r_T = \xi_T T$, where T denotes the duration of the trial and ξ_T is a weighting factor, we ensure that faster movements are favored over slower movements.

Compared to ILC, RL offers a more general optimization approach. Typically, policy search RL algorithms search for local maximum of the cumulative reward, or conversely, local minimum of the cumulative cost. Reward/cost is collected during learning and is given by the reward/cost function. In contrast, ILC makes adjustments to the policy under the assumption that the goal is to reduce the error function to zero, and that the correct adjustment of the policy is given by the sign of the error. The ILC error function must therefore be designed accordingly. In RL, however, the direction in which policy is to be adjusted is inferred from exploration. The only general requirement for the reward/cost function is thus monotonicity.

Any policy search RL algorithm can be used to perform the policy updates. In our experiments, we applied the PI^2 algorithm.[26] We refer the reader to[6] for an example implementation of PI^2.

4. Experimental Evaluation

In the experiments, we evaluated how human demonstrated tasks can be autonomously sped up by the proposed approach. Two different manipulation tasks were performed: (1) assembly of a cake decoration tool, and (2) transfer of a cup, full of liquid. The experiments have been chosen to show the applicability of the approach in diverse cases, both of which would be hard to solve using model based methods. We analyzed how ILC and RL performed when applied to the speed adaptation problem.

The experimental setup for the experiments was composed of two Kuka LWR robot arms in upper body humanoid configuration. The left arm was equipped with a three finger Barret hand, and the right one with a two finger TBK RH707 hand. The DMP (8)–(13) was numerically integrated to provide

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press



Fig. 1. Overview of the experimental system architecture. Motion planning (DMP integration, learning of the parameters) is performed on a PC running Matlab. The obtained desired Cartesian positions and velocities are sent to the robot via a real-time server running on an xPC Target system, which is used to communicate with the Kuka Fast Reasearch Interface (FRI). The inverse kinematics, dynamics, and redundancy resolution is solved by the Kuka Robot Controller (KRC), which uses joint torque control to achieve tracking of the desired trajectory with the specified stiffness.



Fig. 2. Cake decoration tool.

trajectory reference for the robot. The reference was passed to Kuka Fast Research interface running at 100 Hz and the tracking was provided by Kuka Robot Controller (KRC).[27] The KRC control law compensates non-linear robot dynamics using[27]

$$\tau_c = J^T(K_p e + D + F) + f_{\text{dynamics}}(\ddot{q}, \dot{q}, q), \tag{23}$$

where τ_c denotes the controlled joint torque, J is the robot Jacobian, e is the Cartesian position error, q are joint angles and F is the end effector force. f_{dynamics} is the term that compensates the non-linear dynamics, including effects of inertia, Coriolis force, gravity and friction. D is the damping term, which assures critical damping of the resulting second order system. K_p is the proportional gain corresponding to task space stiffness and was set to [2000, 2000, 1000] N/m for the three positional axes and to [300, 300, 300] Nm/rad for the rotational axes, respectively. See Fig. 1 for an overview of the control system.

The initial human demonstration in both tasks was performed by kinesthetic guiding, where the operator freely moved the robot along the desired trajectory in gravity compensation mode. The data were sampled at 100 Hz in Cartesian space using quaternions to encode the orientational part. Each position and orientation dimension, as well as speed profile, were encoded as a DMP with a common phase using 20 Gaussian kernel functions ($N = M = 20$). Note that dimensions of the DMP describing quaternion orientations need to be integrated in S^3 space as shown in.[28]

4.1. Cake decoration tool assembly

In the first experiment, the robot had to assemble a cake decoration tool. For that, it had to insert the piston into the barrel containing the creme, marmalade, etc. (see Fig. 2). The barrel was firmly held by the robot's left hand, while the piston was grasped with the right hand. The cake decoration tool was an off-the-shelf commercial product bought in a grocery store. The whole setup is shown in Fig. 3.

The initial trajectory was generated by kinesthetic guiding. The demonstrator tried to keep the contact forces between the piston and the barrel were as small as could be achieved. When executed on the robot, the speed-up of the trajectory resulted in tracking errors due to compliant control, which increased the contact forces. Basically, the robot had to learn that the trajectory can be faster in the starting section, but had to be slow during the insertion section to prevent excessive forces. The contact forces were computed from joint torques, which were measured by built-in torque sensors.

4.1.1. Adaptation using ILC. The error function for ILC was defined as explained in Eq. (16), with constraints b chosen as

$$b(x) = \xi_f \|\mathbf{f}(x)\| + \xi_m \|\mathbf{m}(x)\|, \tag{24}$$

where $\mathbf{f}$ is the vector of contact forces, $\mathbf{m}$ is the vector of contact torques, $x = x(t)$ is the phase along the trajectory at time t, and ξ_f, ξ_m are the corresponding weighting factors. The drawback of the

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

Table I. Parameter settings used in assembly experiment.

Maximal scaling factor	$\nu_{\max}$	2.5
Maximal force	$f_{\max}$	5 N
Maximal torque	$m_{\max}$	0.2 Nm
ILC error gain	ξ_ν	1
ILC force scale	ξ_f	0.5
ILC torque scale	ξ_m	5
ILC gain	κ	1.02
ILC gain	η_1	0.06
ILC delay	δ_t	0.01
RL exploration noise	Σ^2	0.7
RL term. reward weight	ξ_T	50

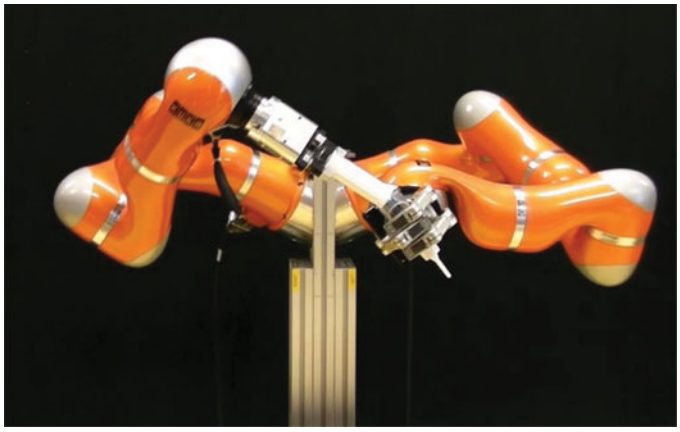


Fig. 3. Experimental setup.

formulation (24) is that it also penalizes forces that might be advantageous, e. g., forces required by the given assembly task. Therefore, weighting scalars ξ_f and ξ_m are defined as

$$\xi_{f,m} = \begin{cases} 0, & \text{if } \|\mathbf{f}\| < f_{\max},\ \|\mathbf{m}\| < m_{\max} \\ \xi_{f,m} & , \end{cases} \tag{25}$$

where $f_{\max}$ and $m_{\max}$ are maximal allowable force and torque norms.

We have also implemented a safety procedure, which interrupts the trajectory if high contact forces are detected. In order to prevent damage to the robot and other equipment, which might occur at high velocity impacts, the upper limit of the temporal scaling factor ν was set to 2.5.

Unfortunately, there exist no explicit recipe for setting the ILC parameters κ, η_1 and δ_t. We can however, make a good guess based on the effects their values have on the learning process. κ represents gain; in our case we want the process to speed up at every iteration, therefore κ needs to be slightly bigger than 1. The error function in our experiments in defined dependent on the forces and torques at the robot's tool, therefore, we set η_1 to a value, which brings the resulting output to the same range as ν. The ILC delay δ_t is the most challenging parameter to set, as its optimal value depends on the task dynamics. It should be lower for the systems with quick response and higher for systems with slower response. In our case, the value of 0.01 proved appropriate. All the parameter settings for this experiment are summarized in Table I.

We performed seven learning iterations using ILC as described in Section 3.1. Results are shown in Fig. 4. It is evident that the ILC approach anticipates the contact force error and can successfully slow down the execution of the task prior to the occurrence of high contact forces. Moreover, with the parameters set as shown in Table I, the robot could speed up the initial policy in the first five cycles. Afterwards, due to the increased contact forces, the ILC algorithm actually slowed down the execution and the 6th and 7th attempt turned to be slower than the 5th. On the other hand, peak

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

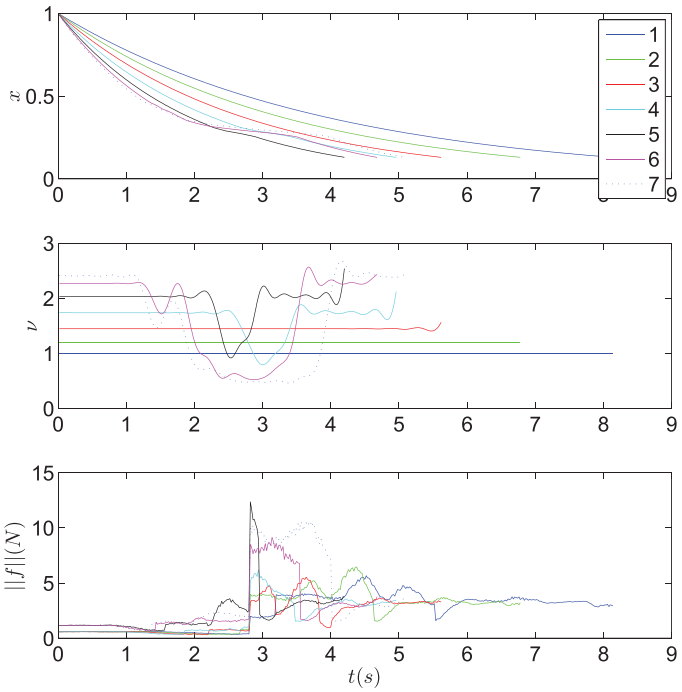


Fig. 4. Phase, temporal scaling factor, and the resulting force norm using the ILC-based approach. Numbers 1–7 in the legend denote the learning iterations.

impact forces were lower in the 6th and 7th adaptation cycle. This turns out to be the biggest problem with the speed adaptation using ILC and the proposed error function (16), (24). After reaching the near-optimal policy, the movement start oscillating around the optimal policy due to the design of the error function.

It is worth mentioning that the ILC approach can also benefit from unsuccessful attempts, when the trajectory was interrupted due to excessive contact forces. High forces and consequently high errors occurred when the robot had missed to insert the piston into the barrel. This error is then used in the next iteration to modify the temporal scaling factor. Here, a problem arises how to recover the control and the error signal when the trajectory was interrupted. In such a case, the missing part of the control and error signal was recovered from the last successfully executed iteration.

4.1.2. Adaptation using reinforcement learning. In our next set of experiments adaptation was performed as described in Section 3.2. The intermediate cost $r(x) = b(x)$ was based on force and torque constraints, as defined by Eqs. (24) and (25). The terminal cost r_T was defined based on total duration of the motion in seconds (22).

Policy updates were performed using RL algorithm PI^2, whose only tuning parameter (besides parameters included in the cost functions) was exploration noise Σ. Σ defines variance of the noise added to the policy when exploring the space in its vicinity. Higher values therefore correspond to bigger exploration steps at each iteration, which facilities faster learning. On the other hand, smaller Σ leads to slower convergence, but generally succeeds in obtaining a better result and provides more safety due to smaller deviations from the current policy. In our experience, setting Σ to 10% of the typical range of the optimized variable presents a good compromise between speed and stability of the convergence.

We performed 40 roll-outs of the experiment, each 10th was executed without the exploration noise in order to check the convergence of the learning algorithm. The results are shown in Fig. 5, whereas Fig. 6 shows collected terminal rewards during learning.

Compared to ILC, the RL approach generated smoother velocity profile, results in lower force/torque norm and required less tuning. However, the number of learning trials needed to achieve a good result was significantly higher. See Fig. 7 for an illustration of final trajectories in both ILC and RL cases.

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

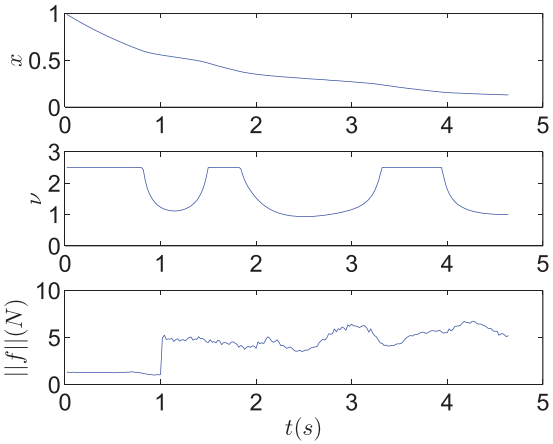


Fig. 5. Phase, temporal scaling factor and the resulting force norm using reinforcement learning.

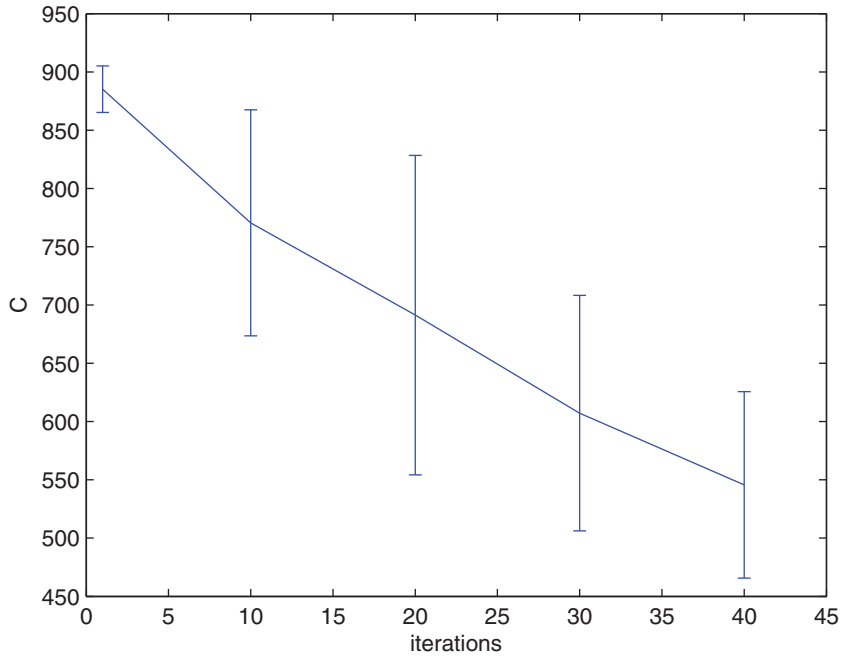


Fig. 6. Convergence of the total cost (sum of all intermediate and terminal rewards) when applying reinforcement learning algorithm PI^2 to learn the cake decoration assembly task. The cost was computed in noiseless trials, performed after every tenth iteration. The bars show standard deviation of six learning series.

4.2. Liquid carrying experiment

The liquid carrying task is a typical example of a task for which it would be very difficult to find a model based solution. The robot had to carry a vessel full of liquid along a specified path. The goal was to speed up the task as much as possible without spilling the liquid. In order to detect spilling, the cup was equipped with a resistance sensor (Fig. 8). Whenever the level of the liquid reached the conductive ring at the edge of the cup, an increase in voltage was detected. The duration of the demonstrated trajectory was 10 s.

4.2.1. Adaptation using ILC. First, we tested the ILC approach explained in Section 3.1. The function defining constraints of this task was constructed using the spilling sensor output:

$$b(x) = U_s(x) - U_0, \tag{26}$$

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

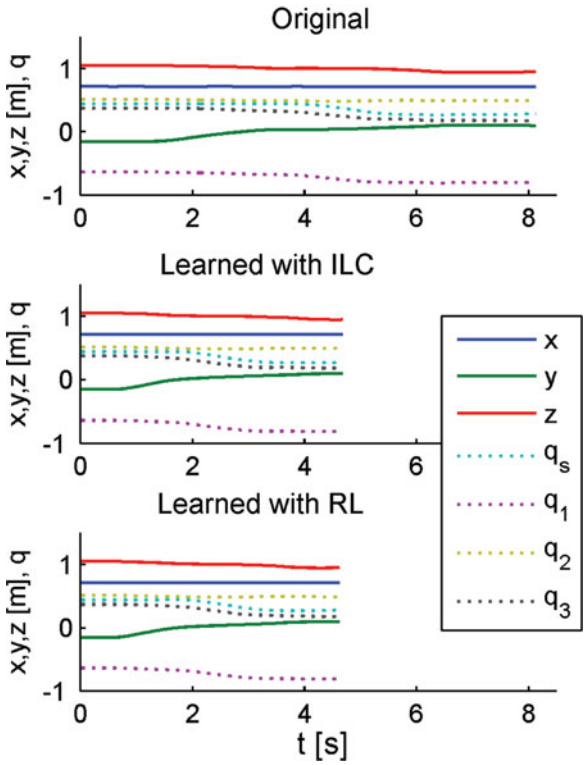


Fig. 7. Top: the original, human demonstrated trajectory. Middle: the trajectory learned with ILC. Bottom: the trajectory learned with RL. The graphs show Cartesian coordinates (solid lines), as well as the four quaternion dimensions (dashed lines).

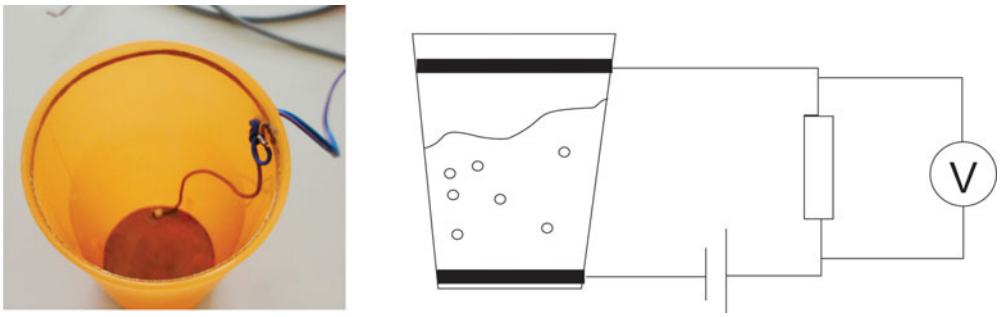


Fig. 8. Sensory system used to detect spilling. The circuit was powered with a 9 volt battery, and voltage measured with a 10 MΩ resistor. When the water closes the circuit the resistor takes an overwhelming part of the voltage drop and the reading increases from 0 to 9 V.

where U_s denotes sensor's output voltage. U_0 denotes the threshold value of the voltage, exceeding of which signifies that the liquid was spilled. Consequently, for U_s smaller than U_0, b is negative and facilitates speeding up. On the other hand, when the water rises above the conducting ring on the edge of the cup, the sensor voltage U_s increases and becomes higher than the threshold U_0. This causes the relevant part of the movement to slow down.

This way, some improvement compared to the initially demonstrated trajectory was achieved. However, the result is still not optimal (see Fig 9 for an example of the learned speed profile after 20 iterations). Figure 10 shows the adaptation performance and also demonstrates the main problem of the ILC approach with error function (16), (26). The error converged towards zero until approximately sixth iteration, when the trajectory was still slow enough to not cause any spilling and the error was decreased by simply speeding up the next iteration. However, when the trajectory got fast enough to cause spilling (after iteration 6), the resulting behavior started to oscillate around the optimal

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

Table II. Parameter settings used in liquid carrying experiment.

Maximal scale factor	$\nu_{\max}$	10
ILC error gain	ξ_v	0
ILC gain	κ	0.98
ILC gain	η_1	0.01
ILC delay	δ_t	0.5
RL exploration noise	Σ^2	0.5
RL term. reward weight	ξ_T	50
spilling threshold	U_0	5 V

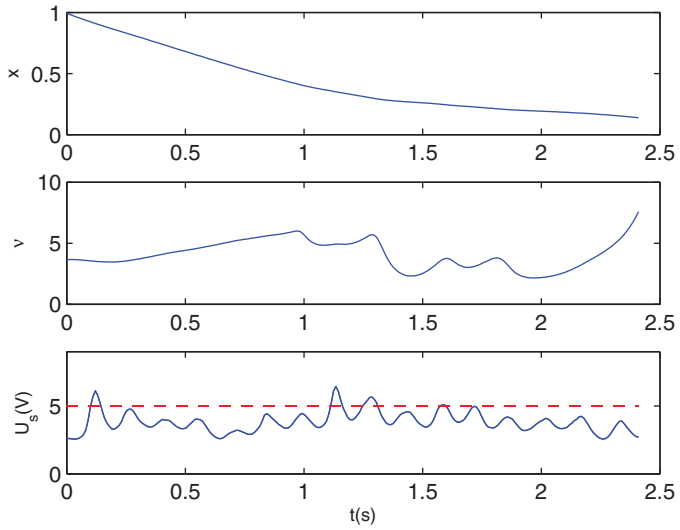


Fig. 9. The evolution of phase, the learned speed profile, and the measured voltages after 20 iterations of iterative learning control. The dashed line in the bottom plot shows U_0-the value of the sensor voltage which signifies spilling. The learned profile is far from optimal. While the robot motion could be accelerated compared to the initially demonstrated movement, the spilling constraint was violated.

behavior. If the liquid is spilled, the corresponding part of trajectory is slowed down to correct the problem. In the next cycle however, there is no spilling, so the algorithm accelerates again.

We need to note here that parameter tuning for this problem turned out to be quite challenging. The biggest problem presented tuning of the prediction time step δ_t, which should be set appropriately to account for the fact that jerky cup motion results in spilling after some time needed by the liquid to rise. The parameters for the liquid carrying experiment are shown in Table II.

4.2.2. Adaptation using reinforcement learning. Next we considered the PI^2 RL algorithm described in Section 3.2 to learn the desired speed profile. We defined the intermediate cost to represent the spilling constraints analogous to the ILC case:

$$r(x) = b(x) = U_s - U_0, \tag{27}$$

and the terminal cost based on the duration of the trajectory in seconds multiplied by a weighting factor

$$R_T = 50T. \tag{28}$$

Figure 12 shows the reduction of the cost in 50 learning iterations, while Fig. 11 shows some of the learned speed profiles. Note that the trajectory is sped up non-uniformly, with three slower sections, corresponding to the parts of the path where robot made sharp turns. This is the expected result. The optimized movement took approximately three times less time than the initially demonstrated movement.

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

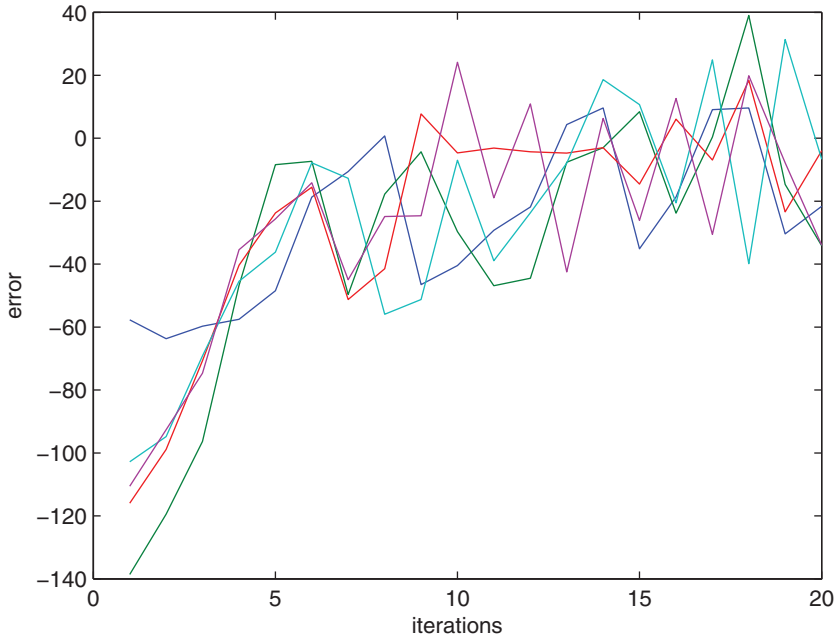


Fig. 10. Convergence of ILC in five learning experiments. The error shown after each iteration is calculated as a sum of all intermediate errors. In all experiments, after iteration six ILC stopped converging as the system got caught between speeding up and spilling the liquid.

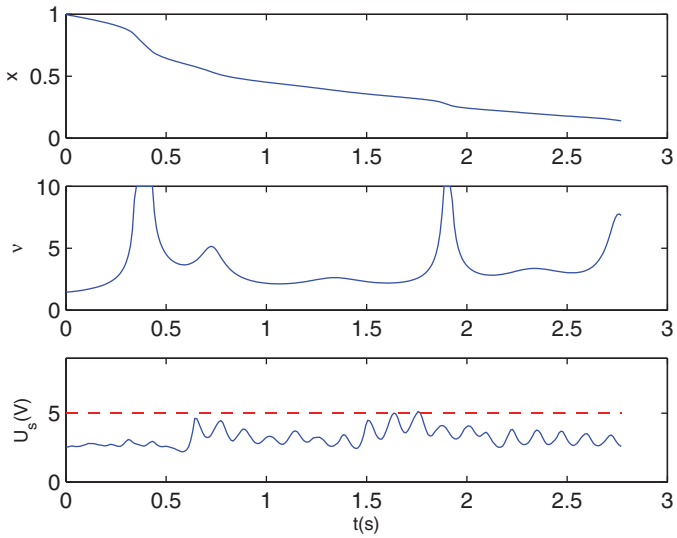


Fig. 11. Learned speed profile ν for a rectangular path in the liquid carrying experiment. The figure shows the result after 50 iterations of the PI^2 algorithm. The dashed line in the bottom plot shows U_0-the threshold value of the sensor voltage which signifies spilling. The learned profile resulted in no spilling, and the final movement took less than 3 s. Maximum value $\nu_{\max} = 10$ was hard-coded for safety reasons.

4.2.3. Human speed profiles. Since we perform human-inspired learning on a human-like robot platform, it is interesting to consider how human-like are the final learned movements. To compare, we asked a human subject to perform the liquid carrying task and measured the corresponding speed profile.

The subject had to trace the same rectangular path of which the robot was learning the speed profile. The path was drawn onto the table surface to provide visual reference for the subject. A marker was placed on the subject's hand and a contact-less motion tracking system (3dInvestigator, Northern Digital Inc.) was used to record the experiment. The setup is shown in Fig. 14. We did

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

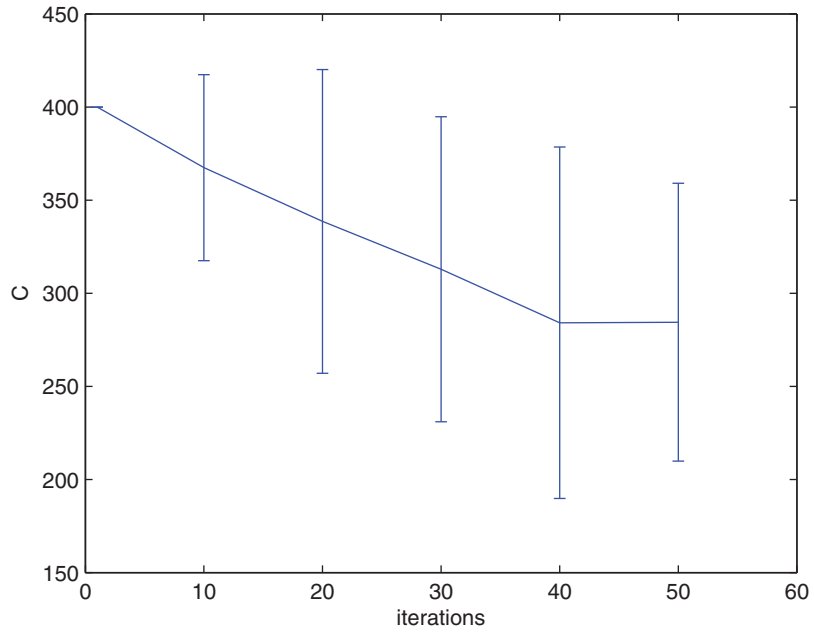


Fig. 12. Convergence of the total cost (sum of all intermediate and terminal rewards) when applying reinforcement learning algorithm PI^2 in the liquid carrying experiment. The cost was computed in noiseless trials, performed after every tenth iteration. The bars show standard deviation of seven learning iterations.

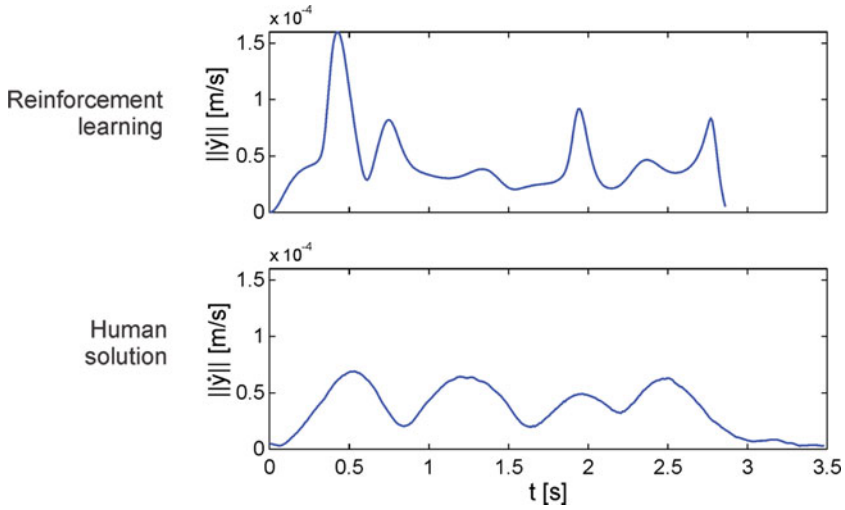


Fig. 13. Speed profiles of robot and human in the liquid transfer experiment. The upper graph shows robot speed profile learned using reinforcement learning, whereas the lower graph show speed profile of a human execution of the task measured with a motion capture system. The human speed profile is much smoother and more tightly corresponds to the shape of the path.

not use the sensor shown in Fig. 8, as we assumed that the subject himself can accurately judge the spilling or the lack thereof.

Several recordings were made. Compared to the robot, the human did not need any learning of the process, and the trajectories were more or less the same in all recorded trials. We contribute this to the fact that liquid carrying is a very common task in everyday human life and as such we already learn it as children. Figure 13 shows a speed profile of the human trajectory along with the result from RL for reference. In this comparison, we did not consider ILC because it did not succeed in learning a spilling-free policy.

From the results, we can see that the human speed profile strongly correlates with the shape of the path, with slow sections at the corners and accelerated sections along the edges. The human

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press



Fig. 14. Setup of human liquid transfer. The human was asked to follow the path outlined in white as fast as possible without spilling the liquid. Note the marker on the cup.

movement along straight lines has a bell-shaped speed profile, as has been noted before.[29,30] On the other hand, the speed profile learned by the robot is much less smooth. The cost criteria for learning was purely goal driven, i.e., it was dependent only on the duration of execution and the spilling of liquid. Therefore, the final solution was selected without regard to smoothness or energy consumption, which differs from the human trajectory planning. The robot could be made to learn a more human-like solution by employing a cost function which would optimize similar criteria as humans do. There exist several hypotheses regarding what such criteria may be. For example, it has been suggested that humans perform smooth movements by optimizing the squared magnitude of hand jerk,[31] joint torques,[30] variance of final positions,[32] or by stochastic feedback control.[33] These principles plausibly explain relatively simple motor actions like point-to-point movements. In the case of liquid carrying experiment, however, there exists another, overlying objective, that of not-spilling the liquid. This objective is arguably more abstract and may possibly be in contrast with the aforementioned motor planning principles. It is not clear how objectives on different levels of abstraction should be combined to form a unifying cost criteria. This will need to be addressed in the future in order to facilitate human-like robot behavior.

Note, however, that in the case of learning speed profiles comparison of humans and robots can be problematic. In this paper, we presented a way for the robot to optimize the speed profile without modifying the given path, whereas humans adapt both at the same time. This was evident in our experiment, where the subject tended to naturally round the corners of the trajectory or even tilt the cup to prevent the liquid from spilling. The subject was then given verbal instructions to avoid such behavior and focus on speed profile only. A haptic device could be used on the subject's arm to constrain the movement and prevent deviations from the given path, however, such task would not be natural to the human.

Furthermore, human joints are actuated by relatively complex combinations of antagonistic muscles. This specific structure plays a central part in the human trajectory planning process.[34] Since such actuation is not typical in contemporary robotic system, optimality criteria employed by the human central nervous system may not be directly applicable to problems involving robots with classical serial kinematic structures.

5. Conclusions

In this paper, we presented a novel approach for non-uniform speedup of learned policies which does not require any explicit environment/robot model. For this purpose, we extended the DMP framework with additional parameters that encode the speed profile of the trajectory. The proposed formulation allows for non-uniform adaptation of the speed profile by modifying weights of the linear combination of Gaussian kernel functions, analogous to DMP trajectory shape adaptation. We proposed two algorithms that enable a robot to obtain near optimal speed profile in an autonomous way: (1) adaptation with ILC, and (2) RL algorithm PI^2. The two approaches were experimentally evaluated on two challenging tasks: assembly of a cake decoration tool and transfer of a vessel full of liquid.

The experimental evaluation in both cases has shown that ILC quickly improves the initially trained policy. However, after near optimal solution is found, ILC starts oscillating around the optimal policy, trying to find an equilibrium between the acceleration and task constraints. This is due to the design of the error function. No further improvement can be achieved at this point.

The performance of ILC depends on a number of tuning parameters. Once these parameters are properly tuned, ILC can improve the speed profile within few trials.

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

RL, on the other hand, requires less tuning and consequently less detailed analysis of the task. Besides the appropriate cost function b, the user only needs to provide exploration noise. RL thus proved easier to tune in both experiments. This comes at a cost, however, since less task knowledge provided to the algorithm results in more iterations needed for convergence.

It could also be argued that RL models the learning processes of living beings, including humans. In this paper however, we presented a purely goal-directed design of the cost functions directing the learning process. Human trajectory planning is undoubtedly more complex; therefore, the speed profiles obtained by the robot are not same as the speed profiles used by a human performing the same task.

We conclude that learning of trajectory speed profiles using ILC can be advantageous over RL in the first few learning cycles due to faster convergence. In this way, the robot can quickly find a better initial approximation for RL, which can then be applied to find near optimal solution, albeit with slower convergence. The results of our work thus show a promising possibility for the development of a learning algorithm as a combination of both approaches. This way, we expect to achieve faster adaptation while retaining robust convergence of RL. The first results of this combined approach have been shown in.[35]

References

1. B. Nemec, A. Gams and A. Ude, "Velocity Adaptation for Self-Improvement of Skills Learned from User Demonstrations," *Proceedings of IEEE-RAS International Conference on Humanoid Robots, Humanoids 2013*, Atlanta, Georgia, USA (2013).
2. D. M. Wolpert, J. Diedrichsen and J. R. Flanagan, "Principles of sensorimotor learning," *Nature Rev. Neurosci.* **12**(12), 739–751 (2011).
3. D. C. Bentivegna, C. G. Atkeson and G. Cheng, "Learning tasks from observation and practice," *Robot. Auton. Syst.* **47**(2–3), 163–169 (2004).
4. H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Netw.* **9**(8), 1281–1302 (1996).
5. J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.* **21**(4), 682–697 (2008).
6. F. Stulp, E. Theodorou and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Trans. Robot.* **28**(6), 1360–1370 (2012).
7. S. Calinon, F. Guenter and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Trans. Syst. Man Cybern. Part B* **32**(2), 286–298 (2007).
8. P. Kormushev, S. Calinon and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Adv. Robot.* **25**(5), 581–603 (2011).
9. J. Buchli, F. Stulp, E. Theodorou and S. Schaal, "Learning variable impedance control," *Int. J. Robot. Res.* **30**(7), 820–833 (2011).
10. J. M. Hollerbach, "Dynamic Scaling of Manipulator Trajectories," *American Control Conference* (Jun., 1983) pp. 752–756.
11. J. E. Bobrow, S. Dubowsky and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.* **4**(3), 3–17 (1985).
12. K. Shin and N. D. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control* **30**(6), 531–541 (Jun. 1985).
13. J. McCarthy and J. Bobrow, "The Number of Saturated Actuators and Constraint Forces During Time-Optimal Movement of a General Robotic System," *Proceedings of 1992 IEEE International Conference on Robotics and Automation*, vol. 1 (May 1992) pp. 542–546.
14. L. Žlajpah, "On Time Optimal Path Control of Manipulators with Bounded Joint Velocities and Torques," *Proceedings of 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota (1996) pp. 1572–1577.
15. O. Dahl and L. Nielsen, "Torque Limited Path Following by On-Line Trajectory Time Scaling," *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, vol. 2 (May 1989) pp. 1122–1128.
16. O. Dahl, "Path Constrained Robot Control with Limited Torques-Experimental Evaluation," *Proceedings of 1993 IEEE International Conference on Robotics and Automation*, vol. 2 (May 1993) pp. 493–498.
17. J. Kieffer, A. Cahill and M. James, "Robust and accurate time-optimal path-tracking control for robot manipulators," *IEEE Trans. Robot. Autom.* **13**(6), 880–890 (Dec. 1997).
18. S. Akella and J. Peng, "Time-scaled Coordination of Multiple Manipulators," *Proceedings of 2004 IEEE International Conference on Robotics and Automation, ICRA '04*, vol. 4 (Apr. 2004) pp. 3337–3344.
19. V. Michna, P. Wagner and J. Cernohorsky, "Constrained Optimization of Robot Trajectory and Obstacle Avoidance," *Proceedings of 2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA)* (Sep. 2010) pp. 1–4.
20. Y. Zhao and P. Tsiotras, "Speed Profile Optimization for Optimal Path Tracking," *American Control Conference (ACC)* (Jun. 2013) pp. 1171–1176.

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press

21. A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Comput.* **25**(2), 328–373 (2013).
22. S. Schaal, P. Mohajerian and A. Ijspeert, "Dynamics systems versus optimal control – a unifying view," *Prog. Brain Res.* **165**(6), 425–445 (2007).
23. K. L. Moore, Y. Chen and H.-S. Ahn, "Iterative Learning Control: A Tutorial and Big Picture View," *Proceedings of 2006 45th IEEE Conference on Decision and Control* (Dec. 2006) pp. 2352 –2357.
24. D. Bristow, M. Tharayil and A. Alleyne, "A survey of iterative learning control," *IEEE Trans. Control Syst.* **26**(3), 96–114 (Jun. 2006).
25. J. Kober, D. Bagnell and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.* **32**(11), 1238–1274 (2013).
26. E. A. Theodorou, J. Buchli and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.* **11**(11), 3137–3181 (2010).
27. G. Schreiber, A. Stemmer and R. Bischoff, "The Fast Research Interface for the Kuka Lightweight Robot," *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications - How to Modify and Enhance Commercial Controllers (ICRA 2010)* (May 2010) pp. 73–77.
28. A. Ude, B. Nemec, T. Petrič and J. Morimoto, "Orientation in Cartesian Space Dynamic Movement Primitives," *International Conference on Robotics and Automation (ICRA)*, Hong Kong, China (2014) pp. 2997–3004.
29. W. Abend, E. Bizzi and P. Morasso, "Human arm trajectory formation," *Brain* **105**(2), 331–348 (1982).
30. Y. Uno, M. Kawato and R. Suzuki, "Formation and control of optimal trajectory in human multijoint arm movement. Minimum torque-change model," *Biol. Cybern.* **61**(2), 89–101 (1989).
31. T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *J. Neurosci.* **5**(7), 1688–1703 (1985).
32. C. M. Harris and D. M. Wolpert, "Signal-dependent noise determines motor planning," *Nature* **394**(6695), 780–784 (1998).
33. E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature Neurosci.* **5**(11), 1226–1235 (2002).
34. N. Hogan, "Adaptive control of mechanical impedance by coactivation of antagonist muscles," *IEEE Trans. Autom. Control* **29**(8), 681–690 (Aug. 1984).
35. R. Vuga, B. Nemec and A. Ude, "Speed Profile Optimization through Directed Explorative Learning," *IEEE-RAS 14th International Conference on Humanoid Robots, HUMANOIDS* (Nov. 2014) pp. 547–553.

https://doi.org/10.1017/S0263574715000405 Published online by Cambridge University Press