Reinforcement learning based compensation methods for robot manipulators

Pane, Yudha P.; Nageshrao, Subramanya P.; Kober, Jens; Babuška, Robert

DOI
10.1016/j.engappai.2018.11.006

Publication date
2019

Document Version
Final published version

Published in
Engineering Applications of Artificial Intelligence

Engineering Applications of Artificial Intelligence 78 (2019) 236–247

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Reinforcement learning based compensation methods for robot manipulators☆

Yudha P. Pane [a], Subramanya P. Nageshrao [b,*], Jens Kober [c], Robert Babuška [c]

[a] Department of Mechanical Engineering, Division PMA, KU Leuven, 3001 Heverlee, Belgium
[b] Green field lab, Ford Motor Company, 3251 Hillview Ave, Palo Alto, CA 94304, USA
[c] Cognitive Robotics Department, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

ARTICLE INFO

Keywords:
Reinforcement learning
Tracking control
Robotics
Actor-critic scheme

ABSTRACT

Smart robotics will be a core feature while migrating from Industry 3.0 (i.e., mass manufacturing) to Industry 4.0 (i.e., customized or social manufacturing). A key characteristic of a smart system is its ability to learn. For smart manufacturing, this means incorporating learning capabilities into the current fixed, repetitive, task-oriented industrial manipulators, thus rendering them 'smart'. In this paper we introduce two reinforcement learning (RL) based compensation methods. The learned correction signal, which compensates for unmodeled aberrations, is added to the existing nominal input with an objective to enhance the control performance. The proposed learning algorithms are evaluated on a 6-DoF industrial robotic manipulator arm to follow different kinds of reference paths, such as square or a circular path, or to track a trajectory on a three dimensional surface. In an extensive experimental study we compare the performance of our learning-based methods with well-known tracking controllers, namely, proportional-derivative (PD), model predictive control (MPC), and iterative learning control (ILC). The experimental results show a considerable performance improvement thanks to our RL-based methods when compared to PD, MPC, and ILC.

1. Introduction

In Industry 4.0, prominently referred to as the fourth industrial revolution, the existing manufacturing processes will be extensively computerized. This will lead to a 'smart factory' which is characterized by modularity, inter-operability, and real-time capabilities. Thanks to these features, the existing mass manufacturing methodology will be eventually replaced by social or custom manufacturing. Manufacturing firms including small and medium enterprises can gain easy and affordable access to robotic technologies that can be customized to meet their needs (Rüßmann et al., 2015). To get superior cost efficiency and to provide better quality of the manufactured products, for each task the industrial robot must be well calibrated. This is also essential to ensure high accuracy and precision (Conrad et al., 2000). Unfortunately calibration is a time consuming process hence in order to achieve faster deployment, the robotic industry may need to change from current fixed control architecture to a flexible control framework (Lu, 2017). That is, an industrial robot which is designed for a fixed and repetitive task must be replaced by a 'smart manipulator'. Here, a smart manipulator is defined as a robotic manipulator that can utilize the operational data to self-optimize. Additionally, a smart manipulator must have the capability to learn and perform a desired task without any explicit task-specific controller. In this work, we augment the standard feedback controller with learning-based compensators that self-optimize to provide optimal performance.

Feedback control methods have been widely used in manufacturing robotics, particularly in motion control problem such as tracking. Precise reference tracking is one of the foremost requirements in the manufacturing robotic applications. This will enable the manipulator arm to move accurately along a predefined trajectory. It has been an active research area for more than three decades (Lewis et al., 2003). Numerous examples of tracking applications using manipulator arms range from simple tasks such as pick-and-place in packaging industry to more complex tasks such as deburring, welding or printing on an irregular surface.

Control methods for reference tracking can be broadly classified into model-based methods (An et al., 1988) and model-free methods (Longman, 2000). For many model-based control approaches, closed-loop stability of the manipulator system can be proven. However, the performance of a manipulator arm, being a physical system used in a complex environment, is often stymied by system non-linearities, sensor noise, and external disturbances. These aberrations can be tedious to model

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2018.11.006.

* Corresponding author.
E-mail address: snageshr@ford.com (S.P. Nageshrao).

https://doi.org/10.1016/j.engappai.2018.11.006
Received 11 September 2017; Received in revised form 13 November 2018; Accepted 17 November 2018
Available online 13 December 2018
0952-1976/

and are difficult to compensate. If they are not corrected appropriately, model uncertainties can lead to degradation of performance during the course of operation (Murray et al., 1994). The current trend in manufacturing requires frequent re-programming of robots, this manual task-specific modeling and tuning is prohibitively expensive. Even an initially well-performing model-based controller provided by the manufacturer may degrade over time. This effect can be due to changes in physical characteristics of the manipulator such as, deteriorated servos, worn out gears, etc. The subsequent re-modeling or re-tuning of the model-based controller can be time consuming and costly. Some aspects of this problem can be addressed by using model-free methods such as learning-based control techniques.

Two of the well-known learning-based control techniques are iterative learning control (ILC) (Bristow et al., 2006) and repetitive control (RC) (Cuiyan et al., 2004). In ILC the objective is to minimize the tracking error iteratively. ILC works as follows, first the controller executes a given task, calculates the tracking error, and uses the error to obtain the control signal for the next iteration. This process is repeated until the error is within an acceptable bound. ILC is prominently used to achieve tracking and/or disturbance rejection of a periodic signal. However, ILC requires the same initial position and velocity of the system in every iteration. For various applications such as manufacturing this requirement can be difficult to satisfy. Repetitive control is based on a similar principle. Additionally, in RC the initialization problem of ILC is addressed by using the internal model, however, this requires the reference trajectory to be periodic. Because of this, a number of control characteristics such as the convergence property needs to be treated differently (Longman, 2000). Also, it is nontrivial to incorporate a measure of optimality when using ILC or RC for any generic nonlinear system.

The stated problem can be (partially) rectified by augmenting a nominal controller with learning capabilities resulting in a combination of model-free and model-based methods (Nguyen-Tuong and Peters, 2010). This leads to a self-adjusting controller that can ensure operational and performance constraints throughout the operational life span of a manipulator arm. The self-adjusting property can be considered as an extra degree of freedom and can be used to compensate for model and parametric uncertainties.

In this paper, we propose two novel reinforcement learning (RL) based methods to improve the performance of a nominal tracking controller. RL is a semi-supervised machine learning approach that is prominently used in sequential decision making problems, where an agent[1] is required to interact and control an uncertain or unknown system. The agent learns to optimize its behavior by maximizing a predefined performance measure. RL has been successfully applied in a wide variety of applications, e.g., games (Tesauro, 1995), human computer interaction (Isbell et al., 2006), and general purpose learning (Mnih et al., 2015).

RL is also prominently used as a control approach in robotics (Kober et al., 2013); well-known examples are autonomous helicopter control (Coates et al., 2010), humanoid robot (Peters et al., 2003), soccer robot (Duan et al., 2007), and manipulators (Bayiz and Babuska, 2014; Bucak and Zohdy, 2001). However, in spite of these promising results, applications of RL in industry or to industrial robotics are rather limited. This can be attributed to the lack of extensive experimental evaluation of RL-based reference tracking methods. The purpose of this article is to bridge this gap and to demonstrate the feasibility of RL in real-world applications such as industrial manipulators. The framework we use is based on the actor-critic scheme that was introduced in Bayiz and Babuska (2014). A major advantage of our methods is that they can be used to augment any existing, stabilizing feedback controller such as PID or LQR. Both simulation and experimental studies have shown a relatively safe learning, compared to pure RL-based control. Additionally, if model-free control, e.g., PID, is used for nominal operation, then there will be no explicit need to identify/learn a system model.

The main contributions of this paper are as follows.

- We extend our initial results from Pane et al. (2016). Based on the RL-based control input compensator from Pane et al. (2016), in this work a novel RL-based method, called the reference compensation method, is developed.
- An extensive experimental evaluation of the introduced methods is performed on a 6-DoF industrial robot, the UR5. The control objective is to follow different types of reference paths, like a square or a circular path, or to track a trajectory on a curved three-dimensional surface.
- The methods developed are compared with well-known tracking-control methods namely, PD, MPC and ILC. PD is used as a baseline for model-free, non adaptive control method, while MPC is used as a reference for model-based control framework and finally ILC is chosen as a baseline for model-free and adaptive method. In Pane et al. (2016) we only compared against PD.

The rest of paper is organized as follows. Section 2 gives an introduction to RL. Following that, in Section 3, the proposed RL-based methods are explained. The implementation of the methods to control a six DoF manipulator and a comparison with PD, MPC, and ILC is given in Section 4. Finally, Section 5 concludes the paper with a note on possible future research.

2. Reinforcement learning preliminaries

Reinforcement learning is an online data-driven machine learning method that enables an agent to perform a desired control task without any prior knowledge of the system's dynamics. This section gives a brief introduction on the theory of reinforcement learning and of the actor-critic method.

2.1. Introduction to RL

In a reinforcement learning process, an agent learns a specific task by interacting with its environment. The learning process, assuming a discrete-time setting, is as follows. At every time step t the agent applies an action $u_t \in \mathbb{R}^m$ which is a function of the system state $x_t \in \mathbb{R}^n$. This results in the state transition of the environment to a new state x_{t+1}, and the agent also receives a numerical reward $r_{t+1} \in \mathbb{R}$. This process is repeated for T_s samples, which is referred to as a learning episode.

The goal is to learn a controller, also called the policy $u = \pi(x)$, so as to maximize the cumulative discounted sum of rewards, termed the return R^π:

$$R^\pi = \mathbb{E}_\pi \left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{T_s} \gamma^k r_{t+k+1} \right] \tag{1}$$

with the scalar constant $\gamma \in [0, 1)$ is the *discount factor*.

In most RL methods, the learning process is modeled as a Markov decision process (MDP) (Sutton and Barto, 1998). Mathematically, an MDP is represented as a tuple $\langle X, U, f, \rho \rangle$ whose elements are: the state space X, the action space U, the state transition function $x_{t+1} = f(x_t, u_t)$, and the reward function $\rho : X \times U \to \mathbb{R}$ providing the instantaneous reward r_{t+1}. The reward function is devised by the design engineer as per the control objective. The mathematical definition of MDP may include the discount factor γ (Mansley et al., 2011) and the control horizon T_s (Coates et al., 2010) as additional elements of the tuple.

If the agent follows a certain policy π, the return function (1) can be formulated in a recursive form, thus resulting in the value function and Bellman equation in the form:

$$V^\pi(x) = \mathbb{E}_\pi \left[\rho\big(x, \pi(x)\big) + \gamma V^\pi(x') \right].$$

The value function V^π gives the cumulative reward (1) from certain state $x \in X$ and following the policy π from that state. In RL algorithms, the objective is to find a policy π^* that maximizes the value function.

[1] In this article, the term agent is synonymous with actor and controller, and are used interchangeably.

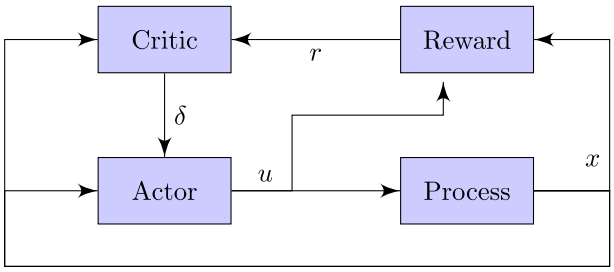

Fig. 1. Actor critic structure (diagram reproduced from Grondman et al. (2012a)).

According to Bellman's optimality principle, the optimal value function is

$$V^*(x) = \max_u \Big(\rho(x,u) + \gamma V^*\big(f(x,u)\big)\Big) \tag{2}$$

where * denotes the optimality. The optimal policy π^* can be derived from V^* (Sutton and Barto, 1998). Most of the widely used RL methods iteratively improve the value function and the policy until the optimality condition is satisfied.

2.2. Actor-critic method

Actor-critic (AC) is one of the solutions to the RL problem based on the temporal difference learning approach (Grondman et al., 2012a). In AC a separate policy (actor) and a value function (critic) are learned simultaneously. Their relation to the environment (process) is visualized in Fig. 1. The AC method works as follows: at time step t the actor senses the current system state x_t and applies an action u_t based on policy π. This leads to a new system state x_{t+1} and a numerical reward r_{t+1}. Using this the temporal difference (TD) error δ at time t is calculated as

$$\delta_t = r_{t+1} + \gamma V(x_{t+1}) - V(x_t). \tag{3}$$

This indicates how well the value function V satisfies the Bellman optimality (2) (Sutton and Barto, 1998). Using the TD error, the critic updates its estimate of the optimal value function. An action u_t that results in a positive TD, $\delta_t > 0$, is favorable as it performs better than expected, hence it must be given a higher preference by the agent. This is achieved by updating the actor to be more preferential for u_t when it encounters a similar state x_t. Conversely, the agent should prefer an action less when it results in a negative TD, i.e., $\delta_t < 0$.

For continuous state and action spaces, such as in the case of a robot manipulator arm, the actor and critic need to be approximated. For this purpose, we use linear-in-parameter approximators with an *a priori* defined basis function vector and a yet to be learned unknown parameter vector. A generic function approximator $F(x)$ is denoted by $F(x,\psi) = \psi^\top \phi(x)$, where $\psi \in \mathbb{R}^{n_\mathrm{p}}$ is the unknown parameter vector of dimension n_p and $\phi(x) \in \mathbb{R}^{n_\mathrm{p}}$ is the user-defined known basis function vector. Using the linear in parameters feature the derivative of $F(x,\psi)$ is $\partial F(x,\psi)/\partial\psi = \phi(x)$. In this work we have used radial basis function (RBF) given by $\tilde{\phi}(x) = e^{-0.5(x-c)^\top B^{-1}(x-c)}$ where $c \in \mathbb{R}^n$ is the center and $B \in \mathbb{R}^{n\times n}$ the covariance matrix of the RBF. The basis function is normalized (Grondman, 2015) $\phi_{\mathrm{b}_i}(x) = \tilde{\phi}_{\mathrm{b}_i}(x) \Big/ \sum_{j=1}^{n_\mathrm{p}} \tilde{\phi}_{\mathrm{b}_j}(x)$ where b signifies which entity it belongs to, i.e., actor a or critic c, for the ith element of the RBF vector. The actor and critic are approximated as

$$\hat{\pi}(x,\vartheta) = \vartheta^\top \phi_\mathrm{a}(x), \tag{4}$$

$$\hat{V}(x,\theta) = \theta^\top \phi_\mathrm{c}(x), \tag{5}$$

respectively, where $\vartheta \in \mathbb{R}^{n_\mathrm{a}}$ and $\theta \in \mathbb{R}^{n_\mathrm{c}}$ are the unknown actor and critic parameters, respectively.

The vanilla actor-critic method, which is used in this work, is given in Algorithm 1. A Gaussian exploration noise Δu_t is added to the output of the actor, i.e., $u_t = \hat{\pi}(x_t,\vartheta_t) + \Delta u_t$ (see line 10 in Algorithm 1). Thanks

Algorithm 1 Actor-critic algorithm.

1: Initialize λ, γ, α_a, α_c
2: Initialize ϑ_0, θ_0
3: **for** each episode **do**
4: Initialize x_0
5: Obtain a random initial action u_0
6: Initialize eligibility trace $\zeta_0 = 0$
7: $t \leftarrow 0$
8: **repeat**
9: calculate the exploration term Δu_t
10: calculate the current action $u_t = \hat{\pi}(x_t,\vartheta_t) + \Delta u_t$
11: apply u_t, measure x_{t+1}
12: obtain reward $r_{t+1} = \rho(x_{t+1}, u_t)$
13: $\delta_t = r_{t+1} + \gamma \hat{V}(x_{t+1},\theta_t) - \hat{V}(x_t,\theta_t)$
14: $\zeta_{t+1} = \lambda\gamma\zeta_t + \frac{\partial \hat{V}(x,\theta)}{\partial\theta}\Big|_{x=x_t,\theta=\theta_t}$
15: $\theta_{t+1} = \theta_t + \alpha_c \delta_t \zeta_{t+1}$
16: $\vartheta_{t+1} = \vartheta_t + \alpha_a \delta_t \Delta u_t \frac{\partial \hat{\pi}(x,\vartheta)}{\partial\vartheta}\Big|_{x=x_t,\vartheta=\vartheta_t}$
17: $t \leftarrow t+1$
18: **until** t = T_s number of samples
19: **end for**

to the exploration Δu_t an agent can visit various states multiple times. For a given approximated value function (5) at state x_{t+1}, reward r_{t+1}, and value at state x_t, i.e., $\hat{V}(x_t,\theta_t)$, the TD-error δ_t in (3) can be easily obtained (see line 13). Eligibility traces ζ_t, are used to increase the convergence rate of the critic (Sutton and Barto, 1998; Grondman et al., 2012b). Finally, the actor and critic parameters, ϑ and θ, are updated using the TD-error (see line 15 and 16 in Algorithm 1) (Grondman et al., 2012b).

3. RL compensation methods

In this section, we present the two control methods, namely, RL-based input compensation and RL-based reference compensation. We start by deriving a general framework which incorporates the RL compensation into a nominal feedback controller. Afterwards, we continue on to a detailed explanation of each method.

3.1. General framework

Let the dynamic model of the manipulator arm in discrete-time be

$$x_{t+1} = f(x_t, u_t) \tag{6}$$

where $f : \mathbb{R}^{n\times m} \to \mathbb{R}^n$ is an unknown nonlinear function of system state x and control input u. The discrete-time assumption can be justified due to the computer control of the manipulator arm. The system output is

$$y_t = C x_t \tag{7}$$

where $y \in \mathbb{R}^l$ and $C \in \mathbb{R}^{l\times n}$ denote the output and output matrix respectively. For accurate tracking the output y must follow a given reference trajectory $y_\mathrm{ref} \in \mathbb{R}^l$. We assume an existing nominal feedback controller

$$u = g(y_\mathrm{ref} - y) \tag{8}$$

where $g : \mathbb{R}^l \to \mathbb{R}^m$ is a function of the tracking error $e = y_\mathrm{ref} - y$. The existing controller g is assumed to include a feedforward term that cancels the various forces acting on the manipulator arm, e.g., gravity and Coriolis terms. Typically, this controller is provided by the robot manufacturer. However, as the extended operation can lead to wear and tear, an existing nominal feedforward controller may no longer compensate for the changes in the dynamics. This will result in a deteriorated or unacceptable control performance.

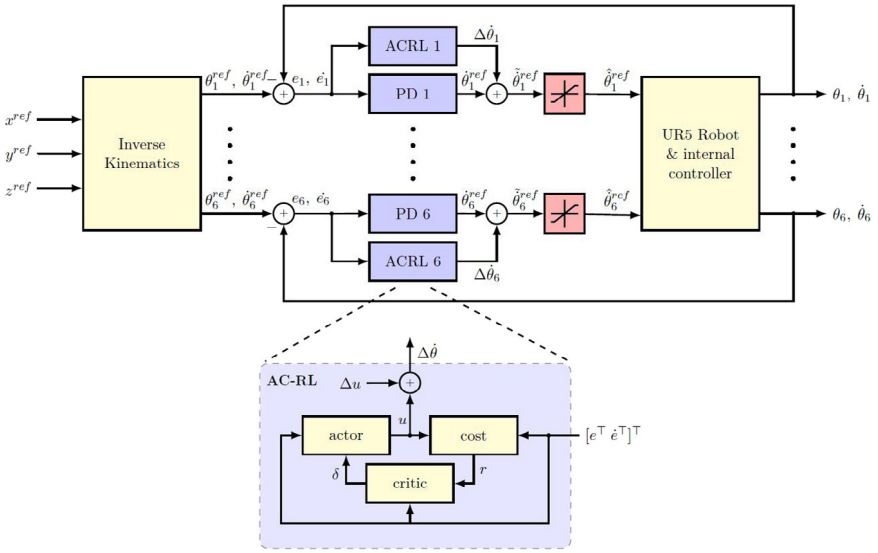

Fig. 2. RL control input compensation framework for a 6-DoF robot, e.g., the UR5. A RL-based compensator is learned for each joint separately using the actor-critic framework. Here the correction is added to the control action.

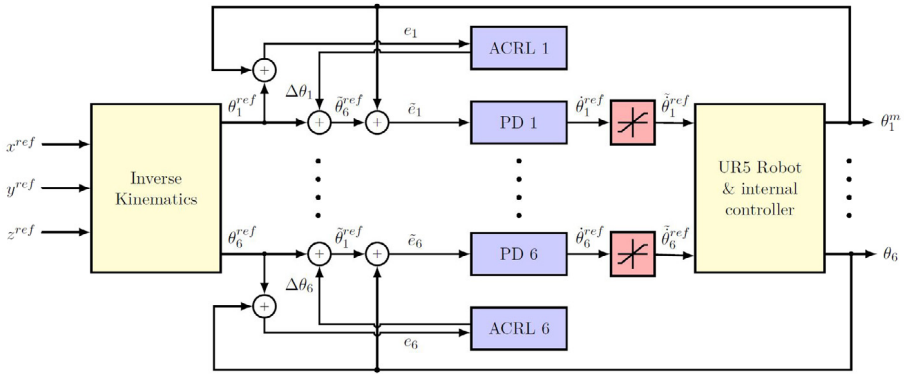

Fig. 3. RL reference compensation framework for the joint space trajectory. In this case, the method is applied to a 6-DoF robot, e.g., the UR5. A RL-based compensator is learned for each joint separately using the actor-critic framework. Here the correction is added to the reference signal.

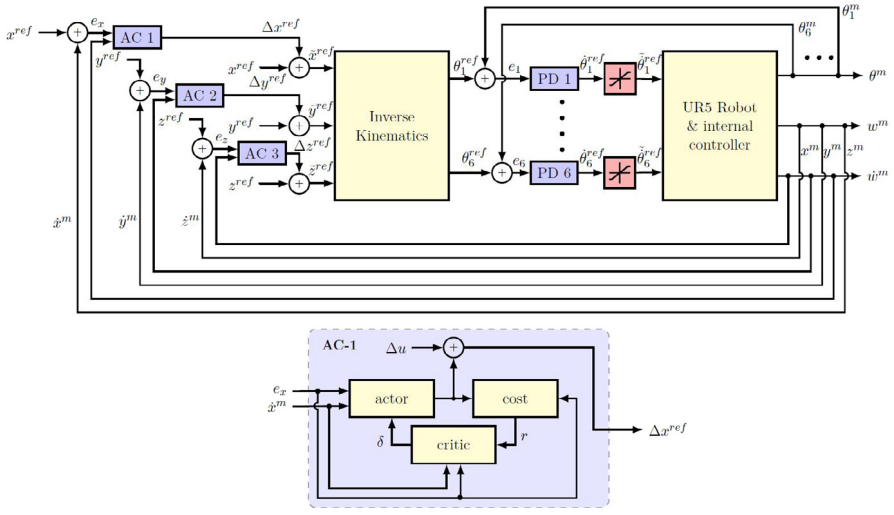

Fig. 4. RL reference compensation framework for the Cartesian space trajectory. A RL-based compensator is learned for each Cartesian direction separately using the actor-critic framework. Here the correction is added to the reference signal.

In this paper we propose two RL based compensation methods that can reduce the tracking error thanks to their online learning capabilities. The first approach is called the *reinforcement learning based control input compensation* method and was introduced in Bayiz and Babuska (2014), where model learning AC (Grondman et al., 2012b) was used to learn a compensator for a 1-DoF robot in a simulation environment. In this paper, we use the vanilla actor-critic method instead of the model

learning AC. This will reduce the number of parameters to be learned, thus simplifying the learning algorithm.

A correction signal is added to the nominal control input. The resulting control input to the system at time t is

$$u_t = g(e_t) + h(e_t) \tag{9}$$

where $h : \mathbb{R}^l \rightarrow \mathbb{R}^m$ is a yet to be learned RL agent. Similar to the nominal controller, the RL policy h is also a function of the tracking error e.

The second approach is called the *reference compensation* method. As the name suggests, the correction is added to the reference signal instead, resulting in a modified reference

$$\tilde{y}_{\mathrm{ref}_t} = y_{\mathrm{ref}\,t} + p(e_t) \tag{10}$$

where $p : \mathbb{R}^l \rightarrow \mathbb{R}^l$ is an error dependent RL-based reference compensator.

The compensators h and p are parameterized in terms of the predetermined basis function vector and the parameter vector to be learned, see (4). Similarly, the value function (critic) V is also approximated using a function approximator (5).

Apart from acting on a different state (reference signal or control input, which in our experiments correspond to position and velocity respectively), the two methods also differ in the space of signal to be compensated. While the control input compensation only allows to compensate in joint space, the reference compensation can provide corrections in either joint or Cartesian space. Furthermore, as will be explained in more details in the following sections, our experiments show that there are trade-offs between the two methods. The RL input compensation generally results in a lower tracking error, but introduces a higher amplitude of jitter in the presence of noise/disturbance. The RL reference compensation converges faster, but the error is larger and the rise-time is slower.

3.2. RL-based input compensation method

For the manipulator arm, the RL based input compensator is formulated as a function of the joint space error and its derivative. Therefore, the RL state is

$$x = \begin{bmatrix} \theta_{\mathrm{ref}} - \theta \\ \dot{\theta}_{\mathrm{ref}} - \dot{\theta} \end{bmatrix} = \begin{bmatrix} e_\theta \\ \dot{e}_\theta \end{bmatrix}, \tag{11}$$

where $\theta, \dot{\theta} \in \mathbb{R}$ are the position and velocity of the arm, and $e_\theta \in \mathbb{R}$ is joint error. In Bayiz and Babuska (2014) the reference signal was also included while formulating the input compensator. Including the reference signal has the drawback that a different RL compensator has to be learned whenever a new reference trajectory is provided.

A major advantage of the input compensation framework is its scalability, i.e., it can be applied to a multiple DoF manipulator arm in a straightforward fashion. For instance, if we apply this method to a 6-DoF UR5 robot, a separate RL compensator is learned for each joint, resulting in a total of 6 ACs. A schematic representation of the RL based input compensation framework is given in Fig. 2. As depicted in the diagram, the compensation signal $\Delta\dot{\theta}_i$ is added to the control input of each joint, i.e., $u = \dot{\theta}_{\mathrm{ref}}$. This signal is the output of the learned compensation policy h which is a function of the state x (see (9) and (11)). Following the actor-critic scheme explained in Section 2.2 and Algorithm 1, the policy is approximated by parameterizing a number of basis functions. Furthermore, the resulting compensated control signal $\tilde{\dot{\theta}}_{\mathrm{ref}}$ is bounded by saturation limits in order to ensure safe operation of the systems.

3.3. RL-based reference compensation method

Instead of modifying the control signal, the second method directly compensates the reference signal fed to the system. Depending on the tracking task, the correction signal can be added either to the joint space reference or to the Cartesian space reference. Figs. 3 and 4 shows the diagram of the RL reference compensation applied to the joint and Cartesian space respectively.

The RL based joint space reference compensator is a function of joint space error and its derivative. The RL state becomes

$$x = \begin{bmatrix} e_\theta \\ \dot{e}_\theta \end{bmatrix} \tag{12}$$

where $e_\theta = r - y$ is the joint space error. For the Cartesian space counterpart, the RL state is a vector of Cartesian error and velocity given by

$$x = \begin{bmatrix} e_w \\ \dot{w} \end{bmatrix} \tag{13}$$

where e_w and $\dot{w}$ are the error and the velocity in one of the Cartesian axes respectively.

Similar to the RL-based control input compensation, the correction policy is learned by using one of the above state vectors as its variable. As the learning progresses, improved approximations of the policy functions are obtained by adjusting the basis functions weights.

Compensating in a different signal space may require different number of AC agents. For the joint space compensation method, the required number of learning agents is the same as the number of DoFs. For the Cartesian space compensation method, at most three RL-based compensators are needed. The choice is generally dictated by the trade-off between a fast response of the system and the oscillatory behavior caused by the measurement noise. This trade-off will be a topic of the discussion in the next section.

4. Experimental results

In this section, experimental evaluations of the proposed methods are presented. We start by describing the robot setup and then we define the different tracking references used for evaluation. Following this we briefly introduce the three benchmark controllers: PID, MPC, and ILC. Finally, we analyze the tracking results of the RL methods in comparison to the benchmark controllers.

4.1. UR5 robot

The UR5 is a 6-DoF industrial manipulator produced by Universal Robots (see Fig. 5). The robot has a manufacturer-provided, internal controller to compensate the gravity and Coriolis forces. The controller and the robot model are not available due the manufacturer's proprietary reasons. This controller can hence be considered as a black-box system.

The UR5 is chosen as the platform for a robotized 3D printer system that was developed at TU Delft. A print head and a laser scanner sensor are attached to the robot's end-effector. The objective is to print on a 3D curved surface by taking the advantage of the manipulator arms large workspace. First a CAD model of the 3D surface is built by using the laser scanner. The model is then used in the subsequent printing stage.

The UR5 can be controlled by sending a velocity or position command either in the joint or Cartesian coordinates. In this work, we choose joint-space velocity command since it results in a smoother motion and also avoids singularity problems. As a consequence of this control input, an implementation of an external feedback controller to track the position reference is required.

4.2. Tracking tasks

Three different tracking tasks have been designed to assess the performance of the learning compensator. The first task is to track a square-shaped trajectory in the $x-z$ plane with the objective to minimize the z axis error (see Fig. 5 for the orientation of the robot's axis frames). The reference trajectory is first generated in the Cartesian space and then transformed to the joint space by using inverse kinematics. In the second task, the reference is a circular trajectory in the x–y plane. The

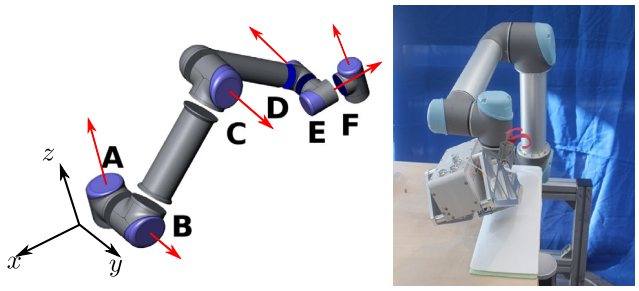

Fig. 5. The UR5 Robot. Left: The joint's axis positioning and the reference frame used in this paper. The joints in alphabetical order (A to F): base, elbow, shoulder, wrist 1, wrist 2, and wrist 3. Picture is courtesy of Universal Robots. Right: the 3D printing system with the robot moving on top of the surface of a curved object.

objective is to minimize both the x and y error. For the last task, the robot minimizes the position errors in the $x-y-z$ axes while following a path above a smooth curved surface as shown in the right panel of Fig. 5.

4.3. Benchmark controllers

Three different types of controllers namely, PD, MPC, and ILC, are used as a basis to compare the performance of the developed RL methods. PD is used as an example for a model-free, non adaptive controller, MPC is used as a reference for model-based controllers and finally ILC represents model-free and adaptive controllers. Brief descriptions of these controllers are given below.

4.3.1. Proportional-derivative controller (PD)

The nominal controller is a standard PD described by the following discrete-time transfer function

$$C(z) = K_p + K_d \frac{z-1}{T_s} \tag{14}$$

where K_p and K_d are the P and D gains, respectively, and T_s is the sampling time. The PD controller regulates each joint of the UR5 robot. Since the UR5 internal controller compensates the dominant nonlinearities, the joints become decoupled. This way, the PD controllers can be tuned independently.

4.3.2. Model predictive control (MPC)

Model predictive control is a model based control method that is prominently used in the process industry (Richalet et al., 1976; Cutler and Ramaker, 1980). MPC uses the system model to predict, at each time step, the future states and to compute the corresponding control inputs up to a specified horizon. The control input is calculated so as to minimize a cost function subject to pre-defined constraints.

In this paper, we use linear MPC that was previously implemented in de Gier (2015). Each UR5 joint is modeled as a SISO system with commanded velocity and joint position as the input and output, respectively. The model parameters are identified using the subspace identification method (Verhaegen and Verdult, 2007). This model is used to predict the system states up to N_p steps. The optimal control input is calculated by minimizing the following cost function

$$J_t = \sum_{i=t}^{N_p-1+t} (y_{i+1}^{\text{ref}} - y_{i+1})^\top W_e (y_{i+1}^{\text{ref}} - y_{i+1}) + \Delta u_i^\top W_u \Delta u_i \tag{15}$$

where y^{ref} denotes the reference signal, $\Delta u_t = u_t - u_{t-1}$, N_p is the prediction horizon, W_e and W_u are the error and control weight (diagonal) matrix, respectively.

Similarly to the PD controller, the MPC is implemented for each joint separately. In order to ensure real-time performance, no constraints are imposed on the MPC. This results in an unconstrained quadratic optimization problem whose globally optimal solution can be obtained easily.

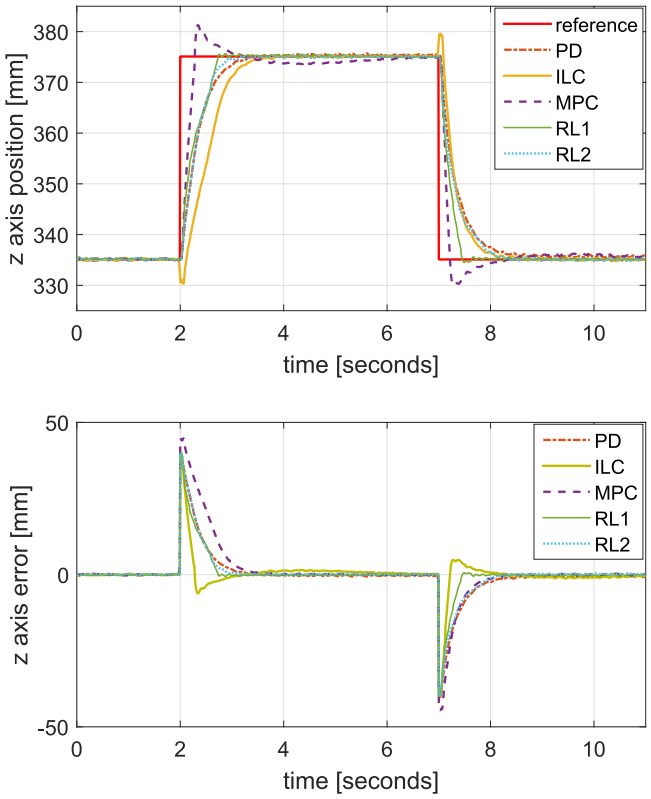

Fig. 6. The tracking performance of the proposed RL compensation methods compared to the benchmark controllers for the square reference.

Table 1
The z-axis tracking performance of RL compensation methods compared to the other controllers for square reference tracking.

Error (mm)	PD	MPC	ILC	RL-1	RL-2
Final steady state	−0.5858	**0.0185**	−0.4798	0.0412	0.1423
RMS	7.4669	9.676	**5.502**	6.4721	7.2051

4.3.3. Iterative Learning Control (ILC)

Iterative learning control is a model-free, adaptive control approach. It is based on the premise that in the absence of an explicit external correction the tracking errors in a repetitive task remain unchanged. To compensate for the repetitive error, an ILC correction is added to a nominal tracking controller thus resulting in a gradual error minimization. The working principle of the ILC-based compensation method is similar to the RL based methods described before.

In this paper we use linear ILC to control each joint of the UR5 robot. The ILC control law is based on a PD-type learning rule

$$\hat{u}_t^{j+1} = Q(q)\left[\hat{u}_t^j + k_p e_{t+1}^j + k_d (e_{t+1}^j - e_t^j)\right] \tag{16}$$

where superscript j denotes the jth iteration, e is the tracking error, $Q(q)$ is a discrete-time low pass filter to improve robustness (Bristow et al., 2006), k_p and k_d are the proportional and derivative gain respectively.

4.4. Reference tracking results

The tracking results of both RL compensation methods compared to the three benchmark controllers are provided as follows. For all tasks, the control loop is executed with a sampling time of 0.008 s (125 Hz). A video of the tracking experiments can be obtained from the website from the supplementary material.

Table 2
The parameters of the RL-based input compensation for the first reference tracking task. Note that there are four actor-critics for the first four joints.

Parameter	Symbol	base	elbow	shoulder	wrist-1
Actor learning rate	α_a	0.04	0.06	0.03	0.03
Critic learning rate	α_c	0.8	0.7	0.9	0.9
No. of actor RBFs	–	[19 9]	[19 9]	[19 9]	[19 9]
No. of critic RBFs	–	[21 9]	[21 9]	[19 9]	[19 9]
Actor RBF variance	B_a	$\begin{bmatrix} 3e{-}7 & 0 \\ 0 & 9e{-}3 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 3e{-}2 \end{bmatrix}$	$\begin{bmatrix} 3e{-}4 & 0 \\ 0 & 9e{-}2 \end{bmatrix}$	$\begin{bmatrix} 3e{-}4 & 0 \\ 0 & 9e{-}2 \end{bmatrix}$
Critic RBF variance	B_c	$\begin{bmatrix} 3e{-}7 & 0 \\ 0 & 6e{-}3 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 8e{-}3 \end{bmatrix}$	$\begin{bmatrix} 1e{-}4 & 0 \\ 0 & 4e{-}2 \end{bmatrix}$	$\begin{bmatrix} 3e{-}4 & 0 \\ 0 & 7e{-}2 \end{bmatrix}$
Reward matrix	Q	$\begin{bmatrix} 8e4 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 4e5 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 5e5 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 5e5 & 0 \\ 0 & 10 \end{bmatrix}$

4.4.1. Task 1: Square reference

A square trajectory along the x direction is used as reference to evaluate the developed tracking control method. Since the end effector orientation does not change throughout the trajectory, only four learning agents, namely for the base, shoulder, elbow and the wrist 1, are needed. The reward function is formulated as the following quadratic function

$$\rho(e_i, \dot{e}_i) = [e_i \quad \dot{e}_i] Q_i [e_i \quad \dot{e}_i]^\top \tag{17}$$

where e is the joint error, $Q \in \mathbb{R}^{2\times 2}$ is a diagonal reward matrix and i is the joint index.

For each agent, the AC parameters are tuned separately using the following approach. The number of RBFs are obtained by iterating through a range of values; a value which balances the trade-off between under-fitting and excessive computational cost is chosen. The diagonal values of Q and α are first initialized with low values in order to yield a relatively small actor output. These values are then gradually increased to achieve a faster, yet monotonic convergence.

Similarly, the PD and MPC controllers are also tuned with heuristics. The PD gains are first initialized with low values. The proportional gain is then gradually increased to reduce the error until a slight overshoot occurs. Finally, the derivative gain is increased to suppress the overshoot. As for the MPC, the diagonal elements of W_e are first initialized with small values while those of W_u with large values in order to have a less aggressive control input. Then, we gradually increase W_e and decrease W_u in order to have a faster controller and smaller errors. Finally, for the ILC controller, we follow the tuning rule described in Bristow et al. (2006).

Fig. 6 shows reference tracking in the z axis for the two RL methods in comparison to the benchmark controllers. Two performance criteria, final steady state and RMS errors, are compared in Table 1.[2] Compared to the nominal PD controller, the two learning-based methods successfully reduce both the steady-state and the RMS error. Compared to MPC, the RL controllers achieve a lower RMS error, while the steady state error is larger. The opposite result is obtained when it is compared to ILC. The reason of the large RMS error is that we limit the RL to learn the compensation policy within the continuous region to avoid large spikes in the error derivatives, making the compensation signal outside that region almost zero, i.e., uncompensated. Although the RL controllers outperforms MPC and ILC only in one of the two performance criteria, Fig. 6 shows that its step response is still preferable since it exhibits neither nonminimum-phase behavior nor overshoot. Furthermore, the RL controller's step responses also show faster settling times.

If we compare the two RL methods, the RL reference compensation has a slightly larger RMS error than the RL input compensation. A possible explanation for this difference is that the latter method modifies a reference trajectory instead of the control input directly. The result is a less aggressive response which is unable to reduce the error as quickly as the RL control input compensation method.

[2] In Tables 1, 9, Figs. 6, 8, and 11, RL-1 and RL-2 corresponds to input-compensation and reference-compensation, respectively.

Table 3
The ACs parameters of the RL-based reference compensation for the first reference tracking task. There is only one actor critic that corrects the z-axis reference.

Parameter	Symbol	z-axis AC
Actor learning rate	α_a	0.002
Critic learning rate	α_c	0.5
No. of actor RBFs	–	[19 11]
No. of critic RBFs	–	[20 10]
Actor RBF variance	B_a	$\begin{bmatrix} 2e{-}6 & 0 \\ 0 & 3e{-}3 \end{bmatrix}$
Critic RBF variance	B_c	$\begin{bmatrix} 9e{-}7 & 0 \\ 0 & 9e{-}4 \end{bmatrix}$
Reward matrix	Q	$\begin{bmatrix} 5e8 & 0 \\ 0 & 0.1 \end{bmatrix}$

Table 4
The MPC parameters used for all three tracking tasks.

Parameter	Symbol	Value
Prediction & control horizon	N_p	30
Error cost matrix	Q	$\begin{bmatrix} 1000 & 0 \\ 0 & 10 \end{bmatrix}$
Input cost matrix	R	1

Table 5
The ILC parameters for the first and second reference tracking task.

Joint	Parameter		
	k_p	k_d	Filter time constant τ
base	0.4	4	0.35
shoulder	0.1	10	0.5
elbow	0.1	15	0.5
wrist-1	0.1	10	0.35
wrist-2	0.1	1	0.35
wrist-3	0.1	1	0.35

The discounted return or learning curve for the first RL method is shown in Fig. 7. As evident from the figure, the return is monotonically converging for all joints. On the other hand, the learning curve for the RL reference compensation method shows an erratic behavior, as shown in Fig. 7. There are two possible reasons for this behavior. First, it might be caused by the RBFs which are initialized with inappropriate values, causing the learning curve to deteriorate before improving. The second explanation is that the reference discontinuities introduce very large TD errors. This causes the policy and value function to change rapidly during the first 80 trials before finally settling to more "stable" parameters.

The convergence time for RL input and reference compensation is approximately 350 and 170 trials, respectively (each trial consists of 1375 samples). Both RL methods are still slower than ILC which reaches convergence in 55 trials. The final tracking performance improves with the number of learning trials.

The RL-based input and reference compensation parameters are reported in Tables 2 and 3, respectively. Meanwhile, the parameters for PD, MPC and ILC are listed in Tables 2, 4 and 5, respectively.

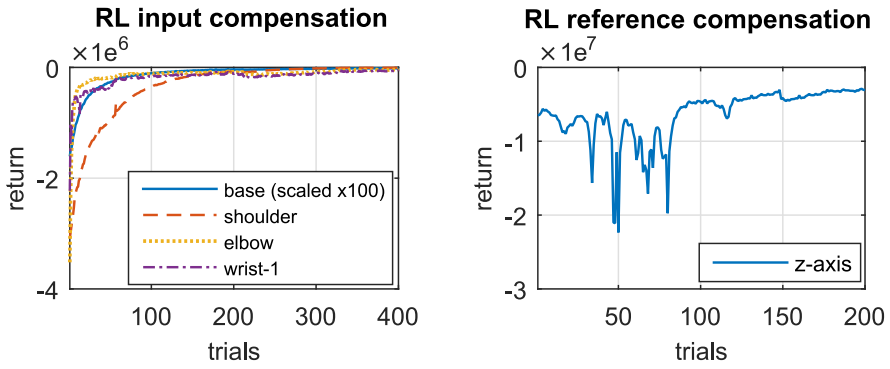

Fig. 7. The sum of rewards (return) of all the learning agents for the square reference. Left: RL input compensation, right: RL reference compensation. Note that appropriate scaling to the discounted return curves is applied for readability.

Table 6
Tracking performance comparison for the circular reference.

Error measure (mm)	MPC	ILC	PD	RL-1	RL-2
RMS x	1.0613	0.5109	4.2388	**0.4847**	0.4962
RMS y	1.0108	0.4662	1.9859	0.3215	**0.2408**
Max absolute x	1.6935	1.0565	6.0253	1.0529	**0.9557**
Max absolute y	1.6938	1.9269	3.0798	**1.1213**	1.6508

4.4.2. Task 2: Circular reference

The second reference tracking task is to follow a circular path in the x-y plane with a fixed end-effector orientation. This causes all joints except for wrist 3 to move. Therefore, a total of five learning agents are needed for the RL-based input compensation. As for the RL-based reference compensation, Cartesian space compensation is chosen again. Since the goal is to minimize errors in the x and y axes, two actor-critics are needed.

The x-y reference trajectory and the tracking error are given in Fig. 8. Two performance measures, RMS and maximum absolute errors, are provided in Table 6. Clearly, the proposed RL controllers outperform PD, MPC and ILC. The only drawback is that the RL based controllers produce high-frequency jitters. This can be attributed to the inherent jitter in the robot manufacturer's velocity controller that influences the learned policy. This jitter cannot be removed due to the black-box nature of the robot's velocity controller. The amplitude of the jitter is, however, not larger than in the nominal case. The learning curves for both methods are shown in Fig. 9. The RL-based input and reference compensation required 70 and 30 trials, respectively. This is comparatively faster than ILC which requires about 90 iterations.

Between the two proposed methods, the RL control input compensation method is better as it quickly minimizes the tracking error. This is because it directly compensates the joint velocity. The RL reference compensation has a slower response since the corrected trajectory is tracked by the nominal PD controller. However, the RL-based reference compensation converges much faster. Furthermore, since this method modifies the position reference instead of the control input, a slightly smoother behavior is obtained. The parameters for RL-based methods are listed in Tables 7 and 8, while the ILC and MPC parameters remain unchanged.

4.4.3. Task 3: Printing trajectory

For the third tracking task, the robot follows a trajectory along a smooth curved surface while keeping the printing head aligned with the normal of the surface (see the right panel of Fig. 5). This trajectory simulates the path that the robot must execute during a 3D printing process. Since the task is performed in a configuration where the robot arm stretches out, a slight deviation in the joint position significantly affects the y and z position. Therefore, this task is expected to require more iterations to achieve minimal joint errors. A total of 5 actor-critic agents are required for the RL-based input compensation. For wrist-3, no learning agent is needed as it does not change the position of the end-effector. As for the RL-based reference compensation method, the correction is performed in the joint space instead of Cartesian space. The reason is that the inverse kinematics algorithm of the UR5 controller is apparently not reliable as it sometimes returns non-smooth joint-space trajectories. Therefore, 5 actor-critics are also employed for the RL-based reference compensation.

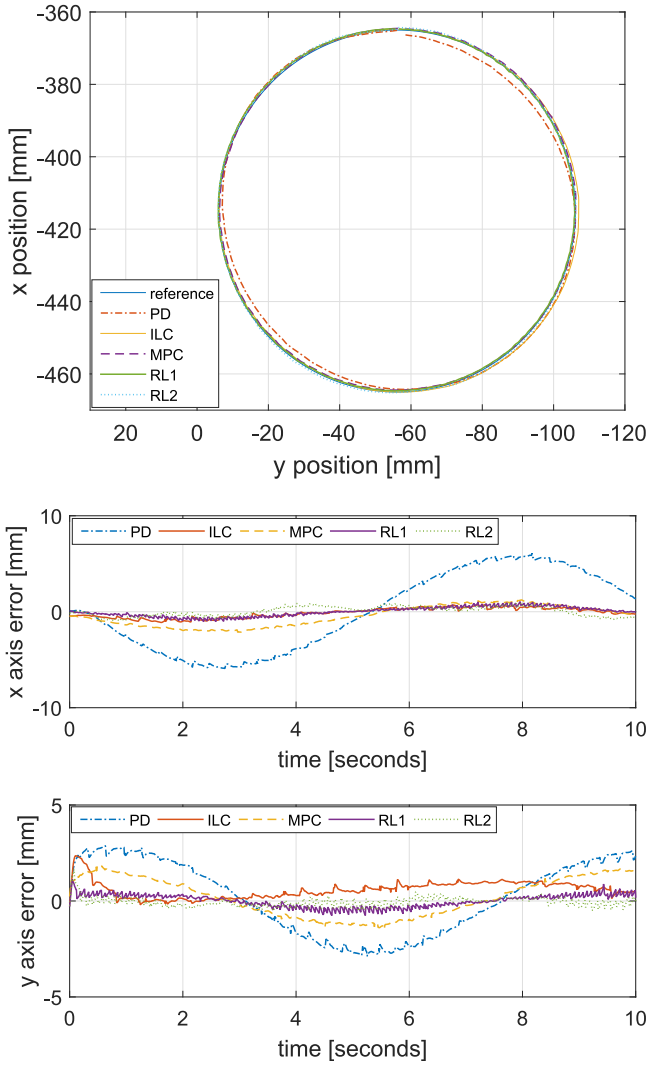

Fig. 8. Top to bottom: the $x-y$ trajectory of the RL methods compared to other control methods for the circular reference, x axis error, y axis error.

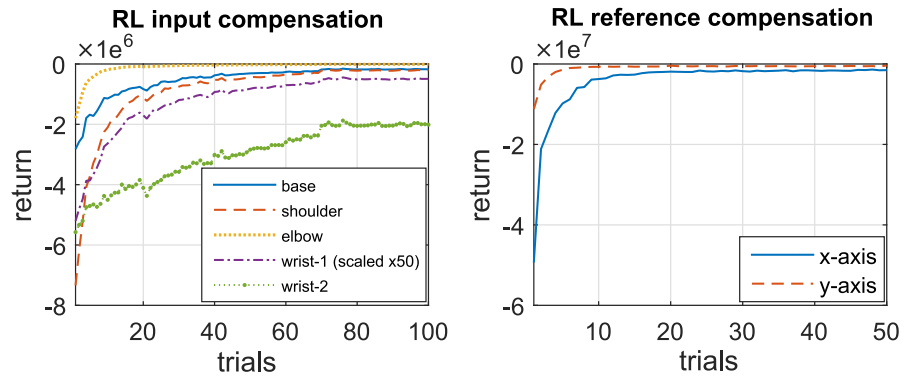

Fig. 9. The sum of rewards (return) of all the learning agents for the circular reference. Left: RL-input compensation, right: RL reference compensation. Note that appropriate scaling to the discounted return curves is applied for readability.

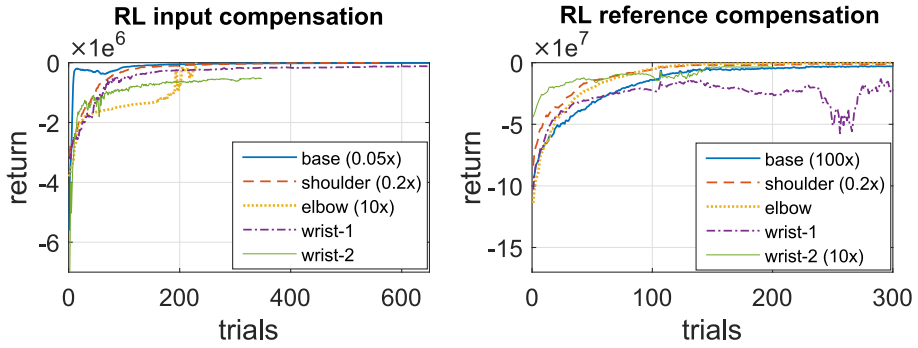

Fig. 10. The discounted sum of rewards (return) of each learning agent for the 3D printing reference. Note that appropriate scaling to the discounted return curves is applied for readability. Furthermore, the number of trials per agent can be different, as shown with different number of trials per agent.

Table 7
The ACs parameters of the RL-based control input compensator for the second reference tracking task.

Parameter	base	shoulder	elbow	wrist-1	wrist-2
Actor learning rate α_a	0.04	0.06	0.03	0.04	0.04
Critic learning rate α_c	0.8	0.7	0.9	0.8	0.8
No. of actor RBFs	[19 9]	[19 3]	[19 3]	[19 3]	[19 3]
No. of critic RBFs	[21 9]	[21 9]	[19 9]	[19 9]	[19 9]
Actor RBF variance B_a	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 6 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 8 \end{bmatrix}$	$\begin{bmatrix} 3e-6 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 6 \end{bmatrix}$
Critic RBF variance B_c	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 9e{-}3 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 3e{-}6 & 0 \\ 0 & 3e{-}2 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 0.3 \end{bmatrix}$
Reward matrix Q	$\begin{bmatrix} 1e6 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e6 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e6 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e6 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e6 & 0 \\ 0 & 10 \end{bmatrix}$

Table 8
The ACs parameters of the RL-based additive reference modifier for the second reference tracking task.

Parameter	Symbol	AC-1	AC-2
Actor learning rate	α_a	0.02	0.02
Critic learning rate	α_c	0.5	0.5
No. of actor RBFs	–	[19 11]	[19 11]
No. of critic RBFs	–	[20 10]	[20 10]
Actor RBF variance	B_a	$\begin{bmatrix} 3e-6 & 0 \\ 0 & 0.05 \end{bmatrix}$	$\begin{bmatrix} 5e-7 & 0 \\ 0 & 0.1 \end{bmatrix}$
Critic RBF variance	B_c	$\begin{bmatrix} 3e-6 & 0 \\ 0 & 0.05 \end{bmatrix}$	$\begin{bmatrix} 5e-7 & 0 \\ 0 & 0.1 \end{bmatrix}$
Reward matrix	Q	$\begin{bmatrix} 5e8 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 5e8 & 0 \\ 0 & 0.1 \end{bmatrix}$

Table 9
The tracking performance comparison for the 3D printing reference.

Error (mm)	MPC	ILC	PD	RL-1	RL-2
RMS x	1.9287	0.6111	19.3509	**0.46153**	0.82856
RMS y	**0.0616**	0.0979	0.53016	0.20632	0.1972
RMS z	**0.3107**	0.3440	2.6643	0.36981	0.43733
Max absolute x	2.3499	3.9082	20.8915	**1.2626**	1.9246
Max absolute y	**0.2963**	0.4854	2.0685	0.60674	0.5844
Max absolute z	1.1019	1.5896	4.2995	1.3896	**1.0341**

The evaluation of the RL control law and of the benchmark controllers is given in Fig. 11. The performance measures are provided in Table 9. The comparison shows that both RL controllers significantly improve the nominal performance. However, compared to the MPC and ILC controllers, the first RL method performs worse in the y and z axis tracking. In the x axis, it outperforms all other controllers. Meanwhile for the second RL method, in comparison to ILC, it loses in all RMS errors. However, the maximum absolute errors attained in the x and z axes show a better result. This implies that the ILC is superior at reducing the overall error, but inferior at minimizing the error variance. This is verified in Fig. 11 which shows that there are several spikes in the ILC errors.

In the experiment, it is again found that the RL controllers suffer from the inherent jitter caused by the robot's internal velocity controller. This issue may be rectified by using a low pass filter for the joint state

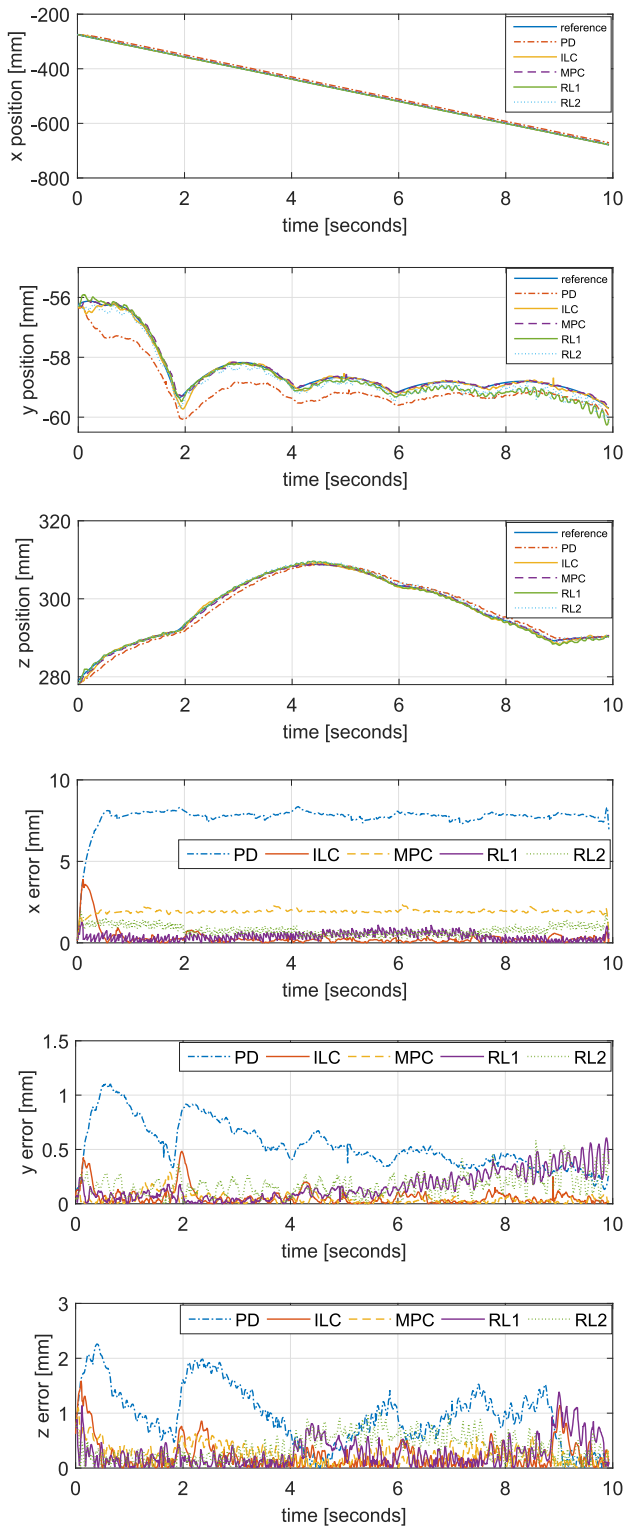

Fig. 11. Reference tracking result of the 3D printing reference task using RL controllers. Top to bottom: the reference and measured x, y and z trajectory of the end-effector, absolute tracking error on x, y and z axes.

measurement. This was not carried out during the experiment since it may result in a lower overall bandwidth due to the filtering delay. The learning curves for all the RL-compensated joints are visualized in Fig. 10. The proposed methods need around 650 and 300 trials (each one consisting of 1250 samples), respectively, to reach the optimal policy. It is important to highlight that in the experiments, the individual actor-critic compensators are not necessarily trained simultaneously. This is due to the difficulty in finding learning parameters which would synchronize them, i.e., achieve a similar learning duration for all of them. This is the reason why the number of trials for each joint may be different, as Fig. 10 shows. For this task, the RL-based controller parameters are reported in Tables 10 and 11, while the ILC parameters are shown in Table 12.

Based on the experimental results, we showed that RL-based compensation methods can significantly reduce the tracking errors without relying on a model, which is an advantage compared to a model-based controller such as MPC. Furthermore, another benefit of using RL-based methods is that the control engineers can flexibly define the reward function so that it is most suitable for the tasks at hand. For example, a higher penalty with respect to the larger error can be imposed by using a higher-order polynomial function. Nevertheless, some limitations still exist. One is that for some tasks, the learning time can be quite slow, as indicated in the last tracking example. Another drawback is the number of parameters to specify is larger than with the MPC and ILC methods.

5. Conclusion & further research

In this paper we have developed and implemented two RL-based compensation schemes to improve the suboptimal tracking performance of a feedback controller in a multi DoF robot arm. The capacity to self-optimize the controllers of robot arms is essential in the Industry 4.0 setting. This capability is required in order to cope with frequent changes in the manufacturing process, to guarantee high accuracy and precision, and hence to ensure cost efficiency and high quality of the manufactured products. For both methods, the technique of additive compensation is used. The first method compensates the control input given by the nominal controller whereas the second method compensates the nominal reference trajectory. The compensation is realized as a continuous state policy function which is constructed by an actor-critic algorithm. Three reference tracking tasks are devised to test the methods. Furthermore, PD, MPC, and ILC controllers are also implemented and their performances are compared.

The RL control input compensation method has an advantage in a faster response since it compensates in the velocity space, thus a higher bandwidth is obtained. Furthermore, it also achieves a smaller error compared to the second method. However, the first RL method is more susceptible to oscillatory behavior. The oscillation is typically induced by the measurement noise or an uncertainty in the robot's servo system (e.g., inherent jitter). Moreover, since the learning process must be kept safe, it results in a slower learning speed. On the other hand, RL reference compensation is advantageous with respect to the smoothness of the tracking response. This is because it only changes the reference while the gain of the controller is kept intact. Another advantage is it converges faster compared to the first method. The limitation of the second method, however, is that the response is less aggressive and the tracking error is slightly larger than that of the first RL method.

The comparative experimental study shows that, for a discontinuous reference such as the square trajectory, the RL-based method results in a more favorable response than the MPC and the ILC. For a simpler smooth trajectory such as the circular reference, the RL-based methods successfully outperform both the ILC and the MPC. However, in a more complex task, like following the printing trajectory, the RL-based controller performance is still slightly inferior to the MPC and ILC. For all tasks carried out in the experiment, we always assume that the reference is known. Had the assumption been invalid, RL would lose the Markov property and hence convergence would no longer be guaranteed.

There are at least two issues which are interesting for future research. First is to see how the proposed methods perform in a torque controlled robot manipulator. The UR5 robot used in our experiments only allows for velocity commands to its internal controller. This method, however,

Table 10
The ACs parameters of the RL-based control input compensator for the third reference tracking task.

Parameter	base	shoulder	elbow	wrist-1	wrist-2
Actor learning rate α_a	0.005	0.06	0.03	0.01	0.05
Critic learning rate α_c	0.1	0.5	0.9	0.1	0.5
No. of actor RBFs	[35 5]	[35 5]	[35 5]	[21 5]	[19 5]
No. of critic RBFs	[21 9]	[21 9]	[21 9]	[19 9]	[19 9]
Actor RBF variance B_a	$\begin{bmatrix} 1e{-}5 & 0 \\ 0 & 0.1 \end{bmatrix}$	$\begin{bmatrix} 4e{-}5 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 3e{-}5 & 0 \\ 0 & 0.5 \end{bmatrix}$	$\begin{bmatrix} 1e{-}4 & 0 \\ 0 & 0.3 \end{bmatrix}$	$\begin{bmatrix} 2e{-}5 & 0 \\ 0 & 0.3 \end{bmatrix}$
Critic RBF variance B_c	$\begin{bmatrix} 4e{-}5 & 0 \\ 0 & 2e{-}3 \end{bmatrix}$	$\begin{bmatrix} 5e{-}5 & 0 \\ 0 & 0.01 \end{bmatrix}$	$\begin{bmatrix} 7e{-}5 & 0 \\ 0 & 0.01 \end{bmatrix}$	$\begin{bmatrix} 1e{-}4 & 0 \\ 0 & 0.01 \end{bmatrix}$	$\begin{bmatrix} 2e{-}5 & 0 \\ 0 & 0.01 \end{bmatrix}$
Reward matrix Q	$\begin{bmatrix} 1e6 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e3 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e3 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 1e5 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 2e4 & 0 \\ 0 & 10 \end{bmatrix}$

Table 11
The ACs parameters of the RL-based reference compensator for the third reference tracking task.

Parameter	base	shoulder	elbow	wrist-1	wrist-2
Actor learning rate α_a	0.002	0.002	0.002	0.002	0.002
Critic learning rate α_c	0.5	0.5	0.5	0.5	0.5
No. of actor RBFs	[19 11]	[19 11]	[19 11]	[19 11]	[19 11]
No. of critic RBFs	[20 10]	[20 10]	[20 10]	[20 10]	[20 10]
Actor RBF variance B_a	$\begin{bmatrix} 6e{-}7 & 0 \\ 0 & 1e{-}3 \end{bmatrix}$	$\begin{bmatrix} 1e{-}5 & 0 \\ 0 & 5e{-}3 \end{bmatrix}$	$\begin{bmatrix} 7e{-}6 & 0 \\ 0 & 5e{-}3 \end{bmatrix}$	$\begin{bmatrix} 7e{-}6 & 0 \\ 0 & 5e{-}2 \end{bmatrix}$	$\begin{bmatrix} 7e{-}7 & 0 \\ 0 & 1e{-}1 \end{bmatrix}$
Critic RBF variance B_c	$\begin{bmatrix} 6e{-}7 & 0 \\ 0 & 8e{-}4 \end{bmatrix}$	$\begin{bmatrix} 1e{-}5 & 0 \\ 0 & 3e{-}3 \end{bmatrix}$	$\begin{bmatrix} 7e{-}6 & 0 \\ 0 & 3e{-}3 \end{bmatrix}$	$\begin{bmatrix} 7e{-}6 & 0 \\ 0 & 8e{-}3 \end{bmatrix}$	$\begin{bmatrix} 7e{-}7 & 0 \\ 0 & 7e{-}2 \end{bmatrix}$
Reward matrix Q	$\begin{bmatrix} 2e7 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 4e7 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 4e7 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 8e7 & 0 \\ 0 & 0.8 \end{bmatrix}$	$\begin{bmatrix} 6e7 & 0 \\ 0 & 0.8 \end{bmatrix}$

Table 12
The ILC parameters for the third reference tracking task.

Joint	Parameter		
	k_p	k_d	Filter time constant τ
base	0.6	1	0.35
shoulder	0.4	1	0.5
elbow	0.4	1	0.5
wrist-1	0.4	1	0.35
wrist-2	0.4	1	0.35
wrist-3	0.4	1	0.35

is limited in terms of the control bandwidth. An access to the motor torques means a higher control bandwidth therefore the possibly of reducing the tracking error even more.

Secondly, for the RL control input compensation method, it would be interesting to investigate the effect of formulating the reward function in terms of the Cartesian errors instead of the joint errors. In our implementation, we only work with joint errors because this was seen as the most feasible approach since the compensation signal is sent to each joint of the robot.

The proposed RL-based methods are relevant for Industry 4.0 where a much wider variety of products are manufactured while, at the same time, quality must be maintained. For applications that require high positioning accuracy, fine tuning the controller for each task will be infeasible, hence a self-learning capability will be necessary. The proposed methods are also well aligned with the data-driven philosophy of Industry 4.0, in which the logged data can be continuously exploited to better the performance.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.engappai.2018.11.006 or https://www.dropbox.com/s/tdkhp4io6yojh94/RL_based_compensation.wmv?dl=0.

References

An, C.H., Atkeson, C.G., Hollerbach, J.M., 1988. Model-Based Control of a Robot Manipulator. MIT Press, Cambridge, MA, USA.

Bayiz, Y.E., Babuska, R., 2014. Nonlinear disturbance compensation and. reference tracking via reinforcement. learning with fuzzy approximators. In: Preprints 19th IFAC World Congress (IFAC-14), Cape Town, South Africa. pp. 5393–5398.

Bristow, D., Tharayil, M., Alleyne, A., 2006. A survey of iterative learning control. IEEE Control Syst. 26 (3), 96–114.

Bucak, I.O., Zohdy, M.A., 2001. Reinforcement learning control of nonlinear multi-link system. Eng. Appl. Artif. Intell. 14 (5), 563–575.

Coates, A., Abbeel, P., Ng, A., 2010. Autonomous helicopter flight using reinforcement learning. In: Sammut, C., Webb, G. (Eds.), Encyclopedia of Machine Learning. Springer US, pp. 53–61.

Conrad, K.L., Shiakolas, P.S., Yih, T., 2000. Robotic calibration issues: Accuracy, repeatability and calibration. In: Proceedings of the 8th Mediterranean Conference on Control and Automation (MED2000), Vol. 1719. Rio, Patras, Greece.

Cuiyan, L., Dongchun, Z., Xianyi, Z., 2004. A survey of repetitive control. In: Int. Conf. on Intelligent Robots and Systems, Vol. 2. pp. 1160–1166.

Cutler, C.R., Ramaker, B., 1980. Dynamic matrix control - a computer control algorithm. In: Joint American Control Conferences.

Duan, Y., Liu, Q., Xu, X., 2007. Application of reinforcement learning in robot soccer. Eng. Appl. Artif. Intell. 20 (7), 936–950.

de Gier, M.R., 2015. Control of a Robotic Arm: Application to On-Surface 3D-Printing. (Master's thesis), Delft Center for Systems and Control, TU Delft, The Netherlands.

Grondman, I., 2015. Online Model Learning Algorithms for Actor-Critic Control (Ph.D. thesis), Delft Center for Systems and Control, TU Delft, The Netherlands.

Grondman, I., Busoniu, L., Lopes, G.A.D., Babuska, R., 2012a. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) 42 (6), 1291–1307.

Grondman, I., Vaandrager, M., Busoniu, L., Babuska, R., Schuitema, E., 2012b. Efficient model learning methods for actor critic control. IEEE Trans. Syst. Man Cybern. Part B: Cybern. 42 (3), 591–602.

Isbell, Jr., C.L., Kearns, M., Singh, S., Shelton, C.R., Stone, P., Kormann, D., 2006. Cobot in LambdaMOO: An adaptive social statistics agent. Auton. Agents Multi-Agent Syst. 13 (3), 327–354.

Kober, J., Bagnell, J.A., Peters, J., 2013. Reinforcement learning in robotics: A survey. Int. J. Robot. Res. 32 (11), 1238–1274.

Lewis, F.L., Dawson, D.M., Abdallah, C.T., 2003. Robot Manipulator Control: Theory and Practice. CRC Press.

Longman, R.W., 2000. Iterative learning control and repetitive control for engineering practice. Internat. J. Control 73 (10), 930–954.

Lu, Y., 2017. Industry 4.0: A survey on technologies, applications and open research issues. J. Ind. Inf. Integr. 6, 1–10.

Mansley, C.R., Weinstein, A., Littman, M.L., 2011. Sample-based planning for continuous action markov decision processes. In: Twenty-First International Conference on Automated Planning and Scheduling.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Murray, R.M., Li, Z., Sastry, S.S., 1994. A Mathematical Introduction to Robotic Manipulation. CRC press.

Nguyen-Tuong, D., Peters, J., 2010. Using model knowledge for learning inverse dynamics. In: IEEE International Conference on Robotics and Automation. pp. 2677–2682.

Pane, Y.P., Nageshrao, S.P., Babuška, R., 2016. Actor-critic reinforcement learning for tracking control in robotics. In: IEEE 55th Conference on Decision and Control (CDC). pp. 5819–5826.

Peters, J., Vijayakumar, S., Schaal, S., 2003. Reinforcement learning for humanoid robotics. In: Proceedings of the Third IEEE-RAS International Conference on Humanoid Robots. pp. 1–20.

Richalet, J., Rault, A., Testud, J.L., Papon, J., 1976. Algorithmic control of industrial processes. In: 4th IFAC Symposium on Identification and System Parameter Estimation.

Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., Harnisch, M., 2015. Industry 4.0: The future of productivity and growth in manufacturing industries, Vol. 9. Boston Consulting Group.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction, Vol. 28. MIT press.

Tesauro, G., 1995. Temporal difference learning and TD-Gammon. Commun. ACM 38 (3), 58–68.

Verhaegen, M., Verdult, V., 2007. Filtering and System Identification: A Least Squares Approach. Cambridge University Press.