Trajectory Optimization for the Handling of Elastically Coupled Objects via Reinforcement Learning and Flatness-Based Control

Daniel Kaczor, Martin Bensch, Moritz Schappler and Tobias Ortmaier

Leibniz Universität Hannover, Institute of Mechatronic Systems
Appelstraße 11a, 30167 Hannover
daniel.kaczor@imes.uni-hannover.de

Abstract. Positioning objects in industrial handling applications is often compromised by elasticity-induced oscillations reducing the possible motion time and thereby the performance and profitability of the automation solution. Existing approaches for oscillation reduction mostly focus on the elasticity of the handling system itself, i.e. the robot structure. Depending on the task, elastic parts or elastic grippers like suction cups strongly influence the oscillation and prevent faster positioning. In this paper, the problem is investigated exemplarily with a typical handling robot and an additional end effector setup representing the elastic load. The handling object is modeled as a base-excited spring and mass, making the proposed approach independent from the robot structure. A model-based feed-forward control based on differential flatness and a machine-learning method are used to reduce oscillations solely with a modification of the end effector trajectory of the robot. Both methods achieve a reduction of oscillation amplitudes of 85 % for the test setup, promising a significant increase in performance. Further investigations on the uncertainty of the parameterization prove the applicability of the not yet widely-used learning approach in the field of oscillation reduction.

Keywords: flatness-based control, reinforcement learning, double deep Q-network, trajectory optimization, oscillation reduction

1 Introduction

Pick and place processes are one of the most common applications in automation technology and robotics and their extensive use in modern production enables significant benefits even by the smallest improvements in accuracy or speed of motion. For this reason, various efforts are being made to increase the performance of industrial robots and, above all, to optimize the accuracy of their end effector positioning.

Classical methods to achieve this are e.g. model-based feedforward control of actuator speed and torque for a given motion profile. The motion profile can further be optimized regarding motion time and smoothness based on the robot dynamics, as shown in [13] for high-speed pick and place applications.

A generalization of feedforward control is given by *flatness-based control* which theoretically makes it possible to design a motion profile for complete vibration reduction [6]. The implementation for high-speed positioning is e.g. presented in [1] for a linear flexible motion system using trapezoidal motion profiles. The concept of adaptive input shaping from [14] allows a more robust parameterization than the feedforward control by using frequency parameters instead of specific physical parameters, which can be difficult to obtain. These approaches all share the necessity of a correct robot model and an identification of the required model parameters, making them partially impractical in their implementation.

The methods of *machine learning* on the contrary do not require system knowledge or any kind of parameter identification, since they are purely based on recorded data. An overview of different learning methods for the feedforward control of a 7-DoF (degrees of freedom) serial link robot arm and an assessment of their performance is given in [9].

The presented methods until here all have the common goal of increasing the accuracy of the robot and its end effector. The objects to be manipulated, especially without rigid coupling, are not part of the considerations. For this reason, only the oscillations occurring at the robot structure can be reduced. In [2] the reduction of shear forces between a suction pad gripper and the object is shown by adjusting the orientation of the end effector while moving. The underlying assumption of the availability of degrees of freedom in the orientation is however not always complied with by the robot or the application. Similiar to the pursued approach in this paper, [3] proofs that the oscillations of a rope-coupled mass attached to a flying drone can be reduced by methods of reinforcement learning.

To extend these results to the oscillation reduction in the positioning of elastically coupled objects with robot manipulators, the contributions of this paper are

1. a structure-independent flatness-based control for handling elastic objects,
2. the comparison of this method with state-of-the-art machine learning.

The remainder of this paper is organized as follows: A brief overview of the theoretical and mathematical background of the presented methods is given in Sec. 2 and their implementation for the specific system at hand is given in Sec. 3. The experimental setup and results are presented in Sec. 4, demonstrating the high potential of the proposed methods. Sec. 5 concludes the paper.

2 Background

First, a brief summary of flatness-based control, reinforcement learning in general and Double Deep Q-Network (Double DQN), the algorithm used within this work, is given.

2.1 Flatness-Based Control

Dynamic systems with the property of differential flatness are characterized by the existence of a flat output y_f which in the single input single output case allows transformations

$$\boldsymbol{x} = \boldsymbol{\psi}_x \left(y_\mathrm{f}, \dot{y}_\mathrm{f}, \ldots, y_\mathrm{f}^{(k-1)} \right) \quad \text{and} \quad u = \psi_u \left(y_\mathrm{f}, \dot{y}_\mathrm{f}, \ldots, y_\mathrm{f}^{(k)} \right) \tag{1}$$

leading to a representation where the state vector $\boldsymbol{x}$ and the input u can be expressed by means of a flat output y_f and a finite number of its derivatives up to the order of the the system's relative degree k [6]. While no general method for finding a flat output of a nonlinear dynamic system exists, in the case of linear controllable systems the controllable canonical form allows determining the flat output directly [6]. After a flat output is found, one can use (1) to calculate an input signal $u(t)$ that leads to the desired output trajectory $y(t)$ for the system.

2.2 Reinforcement Learning

In the context of reinforcement learning an agent finds itself in a state S_t and interacts with its environment through an action A_t derived by a policy π. Acting on its environment results in a change from state S_t to S_{t+1} and an observation of a reward R_{t+1}, valuing the chosen action and new state. The agent's objective is to maximize the discounted return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2}$$

where $\gamma \in [0, 1)$ describes the importance of future rewards [11]. The true value for a specific state s and action a, under a policy π then becomes

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right], \tag{3}$$

where the optimal value is denoted as $Q_*(s, a) = \max_\pi Q_\pi(s, a)$ and the optimal policy is made up of choosing always the highest valued action in each state [12]. In case of large or continuous state spaces, it becomes inadequate to store the Q-value for each discrete time step t separately. Instead, the objective becomes learning a parameterized value function $Q(s, a; \boldsymbol{\theta})$, were $\boldsymbol{\theta}$ denotes the parameterization. The function approximator can be linear in its weights $\boldsymbol{\theta}$ or nonlinear (e.g. an artificial neural network, ANN) [11]. The proposed approach uses an algorithm from the latter class of methods.

The standard update rule for the parameterized Q-learning algorithm is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left[U_t - Q(S_t, A_t; \boldsymbol{\theta}_t) \right] \nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t), \tag{4}$$

where α is the step size and the target U_t is defined by

$$U_t = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t). \tag{5}$$

In the Double DQN approach from [12], two ANNs are used for function approximation. Both networks share the same network architecture but deviate in their weight vectors, resulting in the target

$$U_{\mathrm{DoubleDQN},t} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-). \tag{6}$$

Here, $\boldsymbol{\theta}_t$ parameterizes the online network and $\boldsymbol{\theta}_t^-$ the target network. Inspired by Double Q-Learning [4] the online network is used to select the current action, while the target network estimates its value. Similar to [8] the weights of the target network $\boldsymbol{\theta}_t^-$ are updated by or copied from the online network $\boldsymbol{\theta}_t$ every N time steps and are kept constant otherwise. To break correlations between training samples it is beneficial to use experience replay [5], where tuples of observed transitions $(S_t, A_t, S_{t+1}, R_{t+1}, \text{terminal})$ are stored in a buffer with size N_B and sampled uniformly in order to train the network; where 'terminal' describes whether S_{t+1} is a terminal state.

3 Implementation

As motivated in Sec. 1, the considered scenario is a pick and place task where the object is not rigidly but elastically coupled to the tool center point (TCP), e.g. by elastic grippers like suction cups. Therefore, the coupling between the robot's TCP and the object is modeled as a base-excited mass, illustrated in Fig. 1 (a), where the robot structure is assumed to be rigid. To investigate the principal relations, a 1-DoF replacement model is considered for the robot and the TCP. Thereby y and $\dot{y}$ denote the objects Cartesian position and velocity, while u

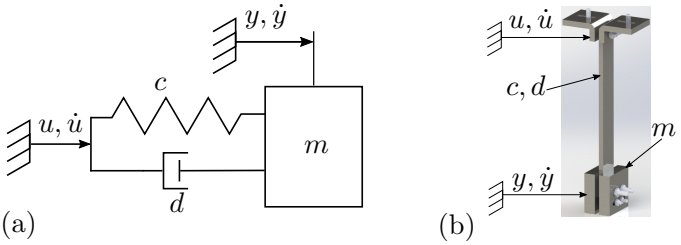

Fig. 1. (a) Base-excited mass and (b) testbed construction.

and $\dot{u}$ are the TCP position and velocity. Furthermore, c and d denote spring stiffness and damping, which altogether results in the differential equation

$$m\ddot{y} + d\dot{y} + cy = d\dot{u} + cu, \tag{7}$$

with disappearing initial values for y and u and their derivatives. This model is employed for both the flatness-based control and the training of the agent.

3.1 Flatness-Based Control

Equation (7) describes a linear system with a relative degree of one. A transformation into its corresponding state space representation leads to a state space model in controllable canonical form (with $y_\mathrm{f} \neq y$)

$$\begin{pmatrix} \dot{y}_\mathrm{f} \\ \ddot{y}_\mathrm{f} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{c}{m} & -\frac{d}{m} \end{pmatrix} \begin{pmatrix} y_\mathrm{f} \\ \dot{y}_\mathrm{f} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u \tag{8}$$

$$y = \begin{pmatrix} \frac{c}{m} & \frac{d}{m} \end{pmatrix} \begin{pmatrix} y_\mathrm{f} \\ \dot{y}_\mathrm{f} \end{pmatrix}, \tag{9}$$

where y_f represents a flat output of the system (7). The dependency on time t for states y_f and $\dot{y}_\mathrm{f}$, input u and output y is omitted for a better readability. For the objective of a rest-to-rest motion two approaches are pursued:

Approach 1 (Neglection of damping): Neglecting d in (7) results to

$$m\ddot{y} + cy = cu, \tag{10}$$

where the physical state y itself already is a flat output and the control input u can be obtained after dividing both sides by c.

Approach 2: Instead of transforming the whole reference trajectory y_ref for the physical output y into flat coordinates, only the initial and end conditions for the output are transformed. Since the goal is to achieve a rest-to-rest motion for the output y, only the dynamics at start and end of the motion have to be constraint. After transforming initial and end values of the physical trajectory into flat coordinates, a polynomial of degree 9 for the reference trajectory $y_\mathrm{f,ref}$ in flat coordinates is used. The constraints on derivatives up to order 4 avoid exciting unmodeled robot dynamics. For the transformation of initial as well as end conditions (9) is used to obtain

$$y(T) = y_T = \tfrac{c}{m} y_\mathrm{f}(T) + \tfrac{d}{m}\dot{y}_\mathrm{f}(T) \quad \text{and} \quad \dot{y}(T) = \dot{y}_T = \tfrac{c}{m}\dot{y}_\mathrm{f}(T) + \tfrac{d}{m}\ddot{y}_\mathrm{f}(T) \tag{11}$$

with T denoting the end of motion. In condition (11) the second derivative of y_f appears, which then can be substituted by the last row in (8)

$$\dot{y}_T = \tfrac{c}{m}\dot{y}_\mathrm{f}(T) + \tfrac{d}{m}\left(-\tfrac{c}{m} y_\mathrm{f}(T) - \tfrac{d}{m}\dot{y}_\mathrm{f}(T) + u(T)\right). \tag{12}$$

At the end of motion the mass has to be in rest at a specific position, which leads to the conditions

$$y(T) = y_T \quad \text{and} \quad \dot{y}(T) = 0 \tag{13}$$

and, in addition, the robot's TCP should simultaneously arrive at the same end position as the mass, leading to

$$u(T) = y(T) = y_T. \tag{14}$$

Finally, the transformation for position and velocity at time T appears to be

$$y_\mathrm{f}(T) = \tfrac{m}{c} y_T \quad \text{and} \quad \dot{y}_\mathrm{f}(T) = 0 \tag{15}$$

while initial values equal zero.

3.2 Double Deep Q-Network

For the learning algorithm a Double Deep Q-Network is used where the agent is trained in a simulated environment using the same mathematical model of (7) as for the flatness-based control approach. The agent's objective is to learn a

correction for the reference trajectory y_{ref}, which is again a polynomial of degree 9, but in physical coordinates, leading to the position profile

$$p(t) = y_{\text{ref}}(t) + a(s), \tag{16}$$

where $a(s)$ denotes the agent's action in state s. The agent selects out of 101 equally spaced positions a ranging from $-0.03\,\text{m}$ to $0.03\,\text{m}$. For the state representation the error in position and velocity between the mass and the reference trajectory

$$\boldsymbol{e} = \begin{pmatrix} y_{\text{ref}} - y \\ \dot{y}_{\text{ref}} - \dot{y} \end{pmatrix} = \begin{pmatrix} e \\ \dot{e} \end{pmatrix} \tag{17}$$

is used, since learning can be sped up if the ANNs inputs have a mean of zero [10]. Further, the reward signal penalizes error in distance, velocity and large changes by the agent through

$$r = -\{e\} - \{\dot{e}\} - 0.5\{a\}, \tag{18}$$

where SI-units are removed with the $\{\cdot\}$ operator. The network consists of three fully connected layers, two inputs, 48 neurons in the hidden layer and 101 neurons in the output layer. Training in a simulated environment led to a discontinuous position profile, which was unsuitable for highly dynamic motions. Hence, a polynomial of degree 19 was fitted to the position profile $p(t)$ altered by the agent. Finally, Table 1 summarizes the most important hyper parameters.

Table 1. Hyper parameters for the Double Deep Q-Network.

Parameter	**Symbol**	**Occurrence**	**Value**
Learning rate	α	(4)	10^{-3}
Discount factor	γ	(2)(3)(5)(6)	0.99
Size of *replay buffer*	N_B	-	10^6
Size of *mini batch*	-	-	128
Update period of *target network*	N	-	30
Exploration decay	ϵ	-	0.0036

4 Experiments

The effectiveness of both presented methodologies is studied and verified using a 4-DoF delta robot. Following a short introduction of the experimental setup in Section 4.1, the experimental results and the comparison of approaches are presented in Section 4.2.

4.1 Experimental Setup

Exemplarily for a typical industrial pick and place robot for highly dynamic applications, the delta robot Codian D4-1100 controlled by a standard industrial PLC and servo inverters is selected for an evaluation of the proposed methods for oscillation reduction. Fig. 2 (a) shows the experimental setup including the

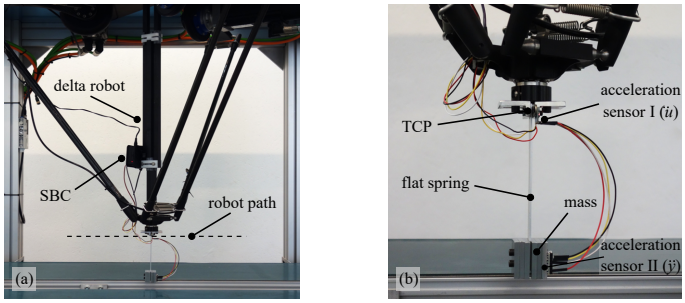

Fig. 2. (a) Delta robot with attached test construction, (b) measurement setup detail.

whole robot structure. A detailed view of the TCP-mounted test construction is shown in Fig. 2 (b). To evaluate both methods without further unknown and unmodeled effects, a test construction following the mathematical model was created. It consists of a flat spring with an exchangeable additional mass, designed to be mountable to the TCP of the robot, as shown in Figure 1 (b). To identify the mechanical parameters of the construction, free oscillations test were performed, where the spring stiffness was identified as $c = 702\,\frac{\mathrm{N}}{\mathrm{m}}$ and the damping as $d = 0.135\,\frac{\mathrm{Ns}}{\mathrm{m}}$ with a measured mass of $m = 0.281\,\mathrm{kg}$. Two acceleration sensors were used: one rigidly connected to the TCP (sensor I) and the other one attached to the mass (sensor II), see Fig. 2 (b).
The influence of different speeds and different masses was investigated. The agent learned its correction $a(s)$ for one motion time T and was also tested for slower reference trajectories. Though the agent was trained with the identified mechanical parameters, it was also tested with varying masses by changing the mass attached to the flat spring. In Summary, 10 movements per time and weight case were recorded. All following results are also transferable to other structures as the approaches are not restricted to a specific hardware or robot type.

4.2 Experimental Results and Comparison of Approaches

The variations of motion time T and mass m were performed using four different methods for calculating the TCP trajectory $u(t)$:

- using flatness-based control with all parameters of (9) ("FBC damping"),
- without damping by using (10) ("FBC"),
- using the Double DQN approach from Sec 3.2 ("Double DQN"),
- for comparison purposes, neglecting stiffness and damping ("solid").

Fig. 3 shows exemplary acceleration signals for all three methods each compared to the solid model for an attached mass of $m = 0.281\,\mathrm{kg}$. The highlighted area I indicates the time of motion for the TCP and the area II the time after arriving at the goal position with remaining mass oscillations. The evaluation concentrates on the remaining accelerations after arriving the goal position. Fig. 4 shows the results for the time variation tests as box plots. The agent was trained for a motion time of 0.4 s, mass of 281 g and a linear movement of 0.55 m. The ordinate shows the maximal remaining acceleration after the TCP arrives its end position

and the abscissa corresponds to the motion time. Note that the axes scales of Fig. 4 and Fig. 5 are different, since using the solid model (directly using the reference trajectory) leads to higher oscillations. The highest improvement in relation to the solid model of around 85% can be seen for the shortest motion time in Fig. 4. After investigating different motion times, the influence of incorrectly identified masses is examined, for a motion time of 0.4 s in Fig. 5. As opposed to Fig. 4, different masses attached to the bending beam are given on the ordinate. All methods were able to achieve a vibration reduction except for a mass of 98 g, for which the solid model performed better. One possible explanation for such a behavior is that a small mass tends to vibrate rather than a large one. At the same time, the flatness-based control reacts sensitive to deviations of identified parameters with respect to the real parameters [14].

The experiments indicate that both methods lead to a vibration reduction. For the present case, the effect of the spring stiffness is much greater than the damping, the flatness-based control approach with consideration of damping was just slightly superior to the one without it.

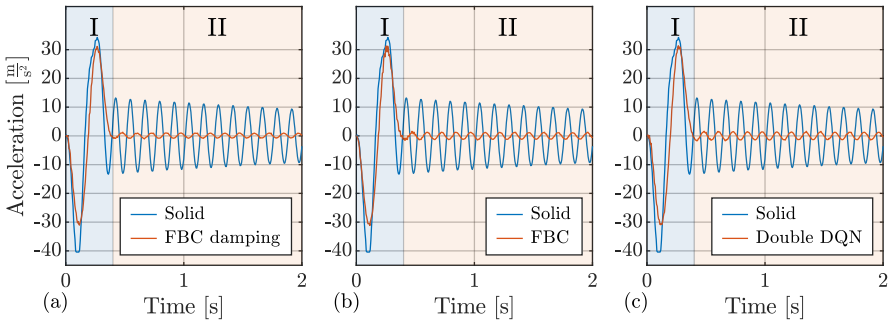

Fig. 3. Exemplary acceleration signals in comparison to solid model for (a) Flatness-based control with damping, (b) FBC without damping and (c) Double DQN.

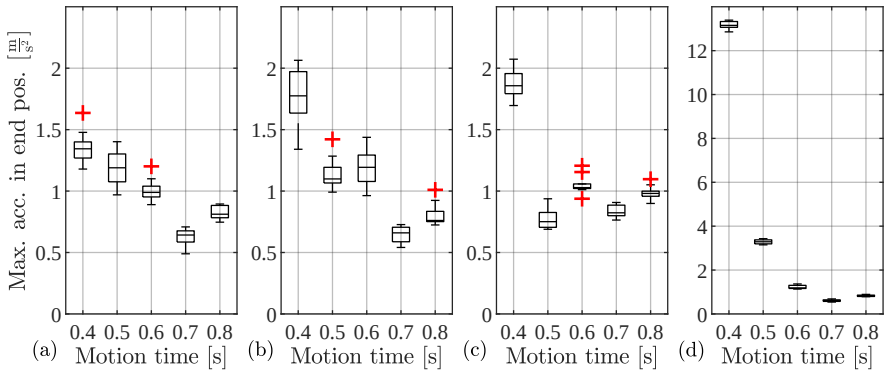

Fig. 4. Variation of the motion time with a constant mass; (a) FBC damping, (b) FBC, (c) Double DQN and (d) solid model. 10 test motions per box with a mass of 281 g.

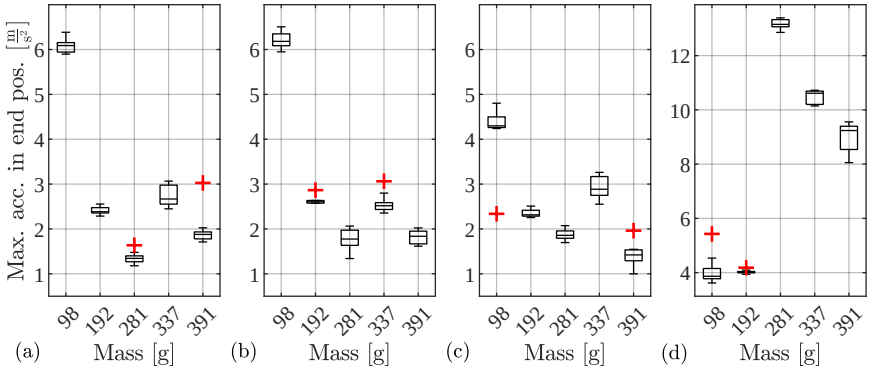

Fig. 5. Variation of the mass with a constant motion time; (a) FBC with damping, (b) FBC, (c) Double DQN and (d) solid model. 10 test motions per box.

The tests with respect to incorrectly identified masses showed similar results, however, with deterioration for the lightest mass. Overall, both methods led to similar performances in vibration reduction.

5 Conclusion and Future Work

This paper illustrates two methods for reducing the vibrations of not rigidly coupled objects during highly dynamic robot movements. Especially the object vibrations at the end of the movement are observed and can be reduced with both methods by up to 85 %. This results in a significant increase in placement accuracy and a time saving during object manipulation. It can thus be shown that both flatness-based control and reinforcement learning with the Double DQN are methodically suitable for the aim of vibration reduction of elastically coupled handling objects. This clear improvement was also shown with inaccurately identified object parameters. Both methods are therefore (to a certain extent) robust against parameter uncertainties. It is important to mention that both methods are completely independent of the considered robot structure, since they are based purely on the TCP position and therefore no consideration of the robot dynamics has to be carried out.

Due to the simple underlying linear model of the base-excited mass, there are clear advantages for flatness-based control in terms of implementation effort. For this reason we consider the flatness-based control approach to be superior to the Double DQN, since programming the reinforcement learning algorithm, including the time for tuning hyper parameters and training, took much more time than designing and implementing the control $u(t)$. However, it should be noted that these described advantages decrease with increasing complexity of the model. In addition, it is not guaranteed that a flat output can be defined for more complex models. In this case, flatness-based control is no longer applicable so that the Double DQN approach would be advantageous. In addition, reinforcement learning offers the possibility to be used in combination with convolutional neural networks, enabling it to learn from raw image data [7].

Future work will focus on the comparison of the methods for more complex (non-linear) systems regarding the achievable vibration reduction and the implementation effort. Furthermore, the investigation of adapting the approaches to three-dimensional space curves is still an open question. In this paper a value-based algorithm of reinforcement learning was used, which is limited to a discrete action space. An actor-critic algorithm with a continuous action space promises further improvements.

References

1. Beckmann, D., Schappler, M., Dagen, M., and Ortmaier, T. New approach using flatness-based control in high speed positioning: Experimental results. In *International Conference on Industrial Technology* (2015), IEEE, pp. 351–356.
2. Chen, S. J. *Time-Optimized Generation of Robot Trajectories Considering Object Dynamic Constraints*. Universität Karlsruhe, 2007. PhD thesis.
3. Faust, A., Palunko, I., Cruz, P., Fierro, R., and Tapia, L. Learning swing-free trajectories for UAVs with a suspended load. In *International Conference on Robotics and Automation* (may 2013), IEEE.
4. Hasselt, H. V. Double q-learning. In *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 2613–2621.
5. ji Lin, L. Self-improving reactive agents based on reinforcement learning, planning and teaching. In *Machine Learning* (1992), pp. 293–321.
6. Lévine, J. *Analysis and Control of Nonlinear Systems*. Springer Berlin Heidelberg, 2009.
7. Mattner, J., Lange, S., and Riedmiller, M. Learn to swing up and balance a real pole based on raw visual input data. In *Neural Information Processing* (2012), T. Huang, Z. Zeng, C. Li, and C. S. Leung, Eds., Springer Berlin Heidelberg, pp. 126–133.
8. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (feb 2015), 529–533.
9. Nguyen-Tuong, D., Peters, J., Seeger, M., and Schölkopf, B. Learning inverse dynamics: A comparison, in proceedings of the european symposium on artificial neural networks. In *European Symposium on Artificial Neural Networks* (2008), IEEE.
10. Orr, G. B., and Müller, K.-R., Eds. *Neural Networks: Tricks of the Trade*. Springer, 1999.
11. Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series)*. A Bradford Book, 2018.
12. Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence* (2016).
13. Zhang, Y., Huang, R., Lou, Y., and Li, Z. Dynamics based time-optimal smooth motion planning for the delta robot. In *International Conference on Robotics and Biomimetics* (2012), IEEE.
14. Öltjen, J., Kotlarski, J., and Ortmaier, T. On the reduction of vibration of parallel robots using flatness-based control and adaptive inputshaping. In *International Conference on Advanced Intelligent Mechatronics* (jul 2016), IEEE.