ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2021.1.001

MENDEL Soft Computing Journal

Comparison of Multiple Reinforcement Learning and Deep Reinforcement Learning Methods for the Task Aimed at Achieving the Goal

Roman Parak✉, **Radomil Matousek**✉

Institute of Automation and Computer Science, Brno University of Technology, Czech Republic
Roman.Parak@vutbr.cz✉, RMatousek@vutbr.cz✉

Abstract

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) methods are a promising approach to solving complex tasks in the real world with physical robots. In this paper, we compare several reinforcement learning (Q-Learning, SARSA) and deep reinforcement learning (Deep Q-Network, Deep Sarsa) methods for a task aimed at achieving a goal using robotics arm UR3. The main optimization problem of this experiment is to find the best solution for each RL/DRL scenario, respectively, minimize the Euclidean distance accuracy error and smooth the resulting path by the Bézier spline method. The simulation and real word application are controlled by the Robot Operating System (ROS). The learning environment is implemented using the OpenAI Gym library, which uses the RVIZ simulation tool and the Gazebo 3D modeling tool for dynamics and kinematics.

Keywords: *Reinforcement Learning, Deep neural network, Motion planning, Bézier spline, Robotics, UR3.*

Received: 1 March 2021
Accepted: 9 May 2021
Published: 21 June 2021

1 Introduction

Robotics as a field of science has been evolving for the past several years and modern robots operating in the real world should learn new tasks autonomously, flexibly and adapt smoothly to different changes. These requirements create new challenges in the field of robot control systems. For this purpose, reinforcement learning (RL) methods such as Q-Learning, SARSA (State–action–reward–state–action), etc. are commonly used [27]. A limitation of these learning methods is the need for a large amount of memory.

In recent years, there has been an increase in deep neural network (DNN) methods in several areas of science, technology, medicine, and more, along with significant advances in Deep Reinforcement Learning (DRL) techniques [11]. DRL overcomes the limitations of simple RL methods by combining parallel computation and embedded deep neural networks (DNN).

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) methods are a promising approach to solving complex tasks in the real world with physical robots. RL/DRL methods are also used in real-world applications, such as improvements in the gaming industry for the Go game [24], as well as in robotic applications for manipulation [21], goal achievement [5, 22], Human-Robot Collaboration [9], and more [17].

Planning the trajectory of the robotic arm as one of the most basic and challenging research topics in robotics has found considerable interest from research institutes in recent decades. Traditional task and motion planning methods, such as RRT [18], RRT* [31] can solve complex tasks but require full state observability, a lot of time for problem solving and are not adapted to dynamic scene changes. Advanced RL/DRL techniques can solve motion planning tasks for multiple-axis industrial robot [23].

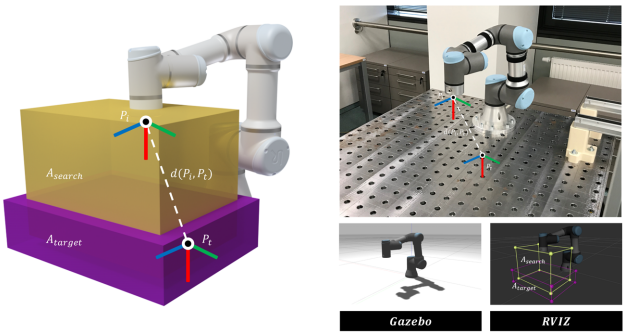


Figure 1: Experimental task aimed at achieving the goal using the UR3 robot. The purple box (A_{target}) here is approximately the area of restriction from which the targets are sampled, and the yellow box (A_{search}) represents the area of safe movement. The distance between the start (P_i) and target (P_t) point is described by Euclidean Distance.

In this paper, we propose several RL/DRL methods for the task aimed at achieving the goal using the cooperating robotic arm UR3 for Universal Robots, more precisely a 6-axis robotic arm [28]. The basic scene of our experiment and the real robot in the initial position position are shown in Fig. 1.

We capitalize the related work in multiple areas of reinforcement learning, deep reinforcement learning, mo-

tion planning, etc. (Section 2 Related Work), and we also summarize the necessary methods needed to create our work (Section 3 Methods).

In the main part of the work, we focus on solving the problem, achieving the goal using advanced methods of motion planning (Section 4 Experiments and Results). Our approach compares different learning techniques (Reinforcement Learning / Deep Reinforcement Learning) to find the trajectory from the initial position to the target position and the resulting trajectory smoothing using a Bézier spline (B-spline) curve.

In the final part of the paper, we focus on the challenges we have encountered, the current limitations, and future extensions of our work (Section 5 Conclusion and Future Work).

2 Related Work

Our approach to finding a point in Cartesian space using multiple RL/DRL techniques is based on previous work in the areas of reinforcement learning, deep reinforcement learning, and motion planning. In the following section, we will briefly discuss previous work on each of the relevant topics.

In research in the field of robotic motion planning, the concept of machine learning emerges. In particular, Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) techniques are an area of growing interest for the robotic research community.

Researchers at Erle Robotics have created a framework for testing various RL/DRL algorithms called OpenAI Gym [6, 32]. Various robot simulation tools are used as an extension of the OpenAI toolkit, with Gazebo [1, 15] and PyBullet [8] being the most commonly used today. The connection of the robotic tool with the robust physical core and the Gym toolkit is created using the ROS (Robot Operating System) [25].

RL/DRL methods are used in robotic applications in the real world for several experiments. One of the approaches is focused on unscrewing operations in robotic disassembly of electronic waste using the Q-Learning method [16], other approaches have used a robotic arm to achieve a goal using the Deep Reinforcement Learning method DQN (Deep Q-Network) [5], TRPO(Trust Region Policy Optimization)[22] and tested the result of the experiment in a real application. Some approaches use 2D/3D cameras and some other sensors to observe the robotic environment [5, 10, 12], others use only dynamic simulation with the specified environment [13, 16] or use real-time robot learning techniques [19].

Motion planning is one of the most fundamental research topics in robotics. Some of the approaches have used traditional planning methods, such as RRT [18], RRT* [31], where structured tree methods are used to find the curve from point A to point B. Other approaches use modern techniques, such as RL/DRL, but both methods use Bézier curves to characterize complex trajectories and smooth motion planning [23].

3 Methods

This section provides a brief introduction to the theory of Reinforcement Learning, Deep Reinforcement Learning, as well as path smoothing techniques using the Bézier spline curve. In each subsection, we present two methods of RL (Q-Learning, SARSA) / DRL (Deep Q-Network, Deep Sarsa) control and the last subsection is focused on trajectory smoothing.

3.1 Markov Decision Process

Markov Decision Process (MDP) is a classical formulation of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations or states, and through those future rewards [27]. MDPs include late reward and the need to compromise with immediate and late reward.

The MDP contains a structure of four basic elements: $(s_t; a_t; P(s_{t+1}|s_t; a_t); R(s_{t+1}|s_t; a_t))$, where s_t and s_{t+1} elements represents the current and next state, a_t part represents the action, $P(s_{t+1}|s_t; a_t)$ means the probability of transition to the state s_{t+1} when taking action a_t in state s_t, and the last part of elements $R(s_{t+1}|s_t; a_t)$ represents the immediate reward received from the environment after the transition from s_t to s_{t+1}. The agent and environment interact at each in a sequence of discrete time steps, $t = 0, 1, ...$ Probability of transition in the MDP structure is depending on the current state s_t and chosen action a_t [11, 27, 33].

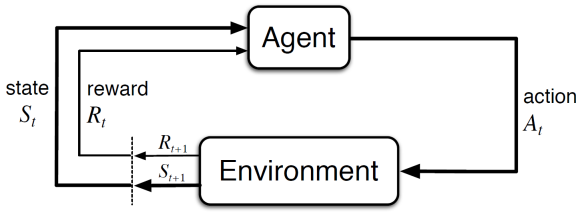


Figure 2: Basic structure of agent-environment interaction in Markov's decision-making process (MDPs) [11, 27]

3.2 Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning that deals with gradual decision-making. The main task of this method is to learn how agents ought to take sequences of actions in an environment to maximize cumulative rewards. Markov decision processes (MDP) are an ideal mathematical formulation for RL problems, for which a direct learning methodology to achieve the goal is proposed. The agent decides to receive not only the current remuneration, but also cumulative remuneration in the next learning state [11, 27, 33].

The agent and MDP together form a sequence that contains a number of state-action pairs represented as $\tau = ((s_0, a_0), (s_1, a_1), \dots)$. The return is defined as the

discounted return for the sequence τ at the time steps t:

$$R_t(\tau) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}, \tag{1}$$

where γ is a discount factor ($(0 \leq \gamma \leq 1)$), $r_{t'}$ is the reward at time steps t'.

The main optimization problem of the RL method is the need to find the optimal policy π^*, which is defined such as maximizing the expected return

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}[R_t(\tau)]. \tag{2}$$

Q-Learning (QL):

is one of the most popular methods of Reinforcement Learning. The QL method was developed as an off-policy TD (Temporal difference) control algorithm, defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)], \tag{3}$$

where α is a learning rate ($0 < \alpha \leq 1$), $\max_a Q(s_{t+1}, a)$ is estimate of the optimal future value, and other parameters are described in the previous section [11, 27].

State–action–reward–state–action (SARSA):

is an on-policy TD control method, very similar to the previous Q-Learning method. SARSA method is characterized by using the next action [27].

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)], \tag{4}$$

3.3 Deep Reinforcement Learning

Deep reinforcement learning (DRL) combines artificial neural networks (ANN) with a learning reinforcement architecture that allows agents to learn the best possible actions in an individual environment to achieve their goals. The main approach of this method is to approximate the function and optimize the goals, mapping the state-action pairs to the expected rewards [11, 33]. This area of research can solve a wide range of complex decision-making tasks that were previously out of reach for a machine.

The learning agent's DRL methods with auxiliary tasks within a jointly learned representation can significantly increase the effectiveness of the learning sample. The learning agent's DRL methods with auxiliary tasks within a jointly learned representation can significantly increase the effectiveness of the learning sample. This is based on simultaneously maximizing a number of pseudoreward functions, such as immediately predicting the reward ($\gamma = 0$), predicting changes on the next observation, or predicting the activation of some hidden unit of the agent's neural network [11].

Deep Q-Network (DQN):

is a combination of Q-learning with deep convolutional ANN and reinforcement learning method, multilayer and deep ANN specialized in the processing of spatial data fields. For a given state of the neural network s, the output is a vector of action values $Q(s, a; \theta)$, where θ are the network parameters [11, 29]. The two most important components of the DQN algorithm are the use of the target network and the use of the experience replay. The formula used by DQN is then:

$$Q(s_t, a_t; \theta_t) \leftarrow Q(s_t, a_t; \theta_t) + \alpha[(r_{t+1} + \gamma \max_a Q(s_{t+1}, a_t; \theta_{\bar{t}}) - Q(s_t, a_t; \theta_t))^2], \tag{5}$$

where a target network with parameters $\theta_{\bar{t}}$, is the same as the online network except that its parameters are copied every τ steps from the online network, so that then $\theta_{\bar{t}} = \theta_t$, which implies that weight of the neural network [29].

Deep SARSA (DSARSA):

is similar to the previous part. In the structure of learning, the approximation of the value function is with the convolutional neural network (CNN), that uses Q-network to obtain Q value like DQN. The Function is represented by a CNN with weights θ and an output that represents the Q-values for each action[20].

$$Q(s_t, a_t; \theta_t) \leftarrow Q(s_t, a_t; \theta_t) + \alpha[(r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}; \theta_{t-1}) - Q(s_t, a_t; \theta_t))^2], \tag{6}$$

3.4 Bézier-Spline Curves

Bézier curves use Bernstein's polynomials, which were described in 1912 by the Russian mathematician Sergei Bernstein [30].

B-spline curves, like Bézier curves, use polynomials to generate a curve segment. The main difference between simple Bézier curves and a B-spline is that B-Spline is used as a series of control points to determine the local geometry of the curve. This feature ensures that only a small portion of the curve is changed when a control point is moved [30]. A Bézier spline curve of degree p is defined by $n+1$ control points $P_0, P_1, .., P_n$ [2]:

$$B(t) = \sum_{i=0}^{n} N_{i,p}(t) P_i, \tag{7}$$

where $N_{i,p}(t)$ is a normalized B-Spline curve defined over the nodes.

4 Experiments and Results

In this section, we present an application that includes a task using several RL/DRL techniques for the point of Cartesian space achievement problem using an collaborative manipulator with 6-DOF (Degrees of Freedom). The experiment will be demonstrated in simulation and in the real world.

4.1 Setting up the learning environment

For our experiment, we will use a cooperating robotic arm UR3 from Universal Robots (Fig. 1), more precisely a 6-axis robotic arm with a working radius of 500 mm/19.7 inches, a payload of 3 kg/6.6 pounds and a repeatability of ±0.1 mm [28].

The simulation is controlled through communication between the RVIZ [7, 26] simulation tool and the Gazebo 3D modeling tool [1, 15]. In our experiment, we use the UR3 URDF model (Universal Robotic Description Format) without a gripper and the official ROS driver for Universal Robots [3], which is used to control real / simulation robots.

The main structure of our experiment is shown in Fig. 1. Fig. 3 shows a simplified structure of the UR3 model in the RVIZ simulation tool. The blue sphere represents the initial position of the robot and the target to be reached by the robot is represented as a red sphere. The purple box (A_{target}) is approximately the area of restriction from which targets are removed, and the yellow box (A_{search}) represents the area of safe movement. The safe area is created with given to the individual robot model to avoid collisions with the target area (imagine that the target area is a bin for an object selection problem).

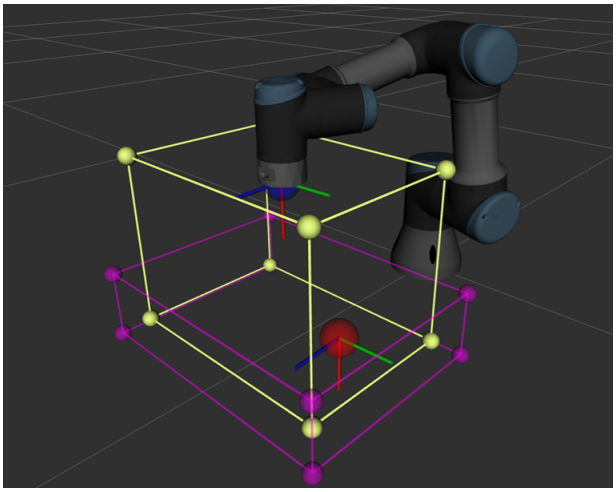


Figure 3: Learning environment of the UR3 model in the RVIZ simulation tool.

4.2 Definition of experiment

In our problem, we choose a task aimed at achieving a goal in Cartesian space using a robotics arm. The task aims to autonomously find the trajectory from the initial position to the target position using various learning techniques (Reinforcement Learning / Deep Reinforcement Learning) and using the Bézier spline method to smooth the resulting trajectory.

First, the robot position is initialized and the target position is randomly selected. When selecting a target position, we start the learning process for each learning technique. The distance between the start P_i and target P_t point is described by Euclidean Distance d_t (Eq. 8).

$$d_t(P_i, P_t) = \sqrt{\sum_{i=1}^{n} (P_i - P_t)^2}, \tag{8}$$

4.3 Learning process

We implement the learning environment within the OpenAI Gym library [6, 32], which provides a interface to train and test the learning process.

To evaluate our algorithm, we performed identical experiments with several RL/DRL techniques. In the first experiment we use classical RL techniques (QL, SARSA) and in the next experiment we use modern DRL techniques (DQN, DSARSA). Hyper-parameters of individual RL/DRL techniques are given in the table (Tab. 1, Tab. 2).

Table 1: Hyper-parameters used for Reinforcement Learning method (Q-Learning, DARSA)

Hyper-parameter	Symbol	Value
Episodes of Training	M_{min}, M_{max}	500, 1000
Steps per Episode	T	100
Discount Factor	γ	0.75
Learning Rate	α	0.3

Table 2: Hyper-parameters used for Deep Reinforcement Learning method (DQN, DSARSA)

Hyper-parameter	Symbol	Value
Episodes of Training	M_{min}, M_{max}	500, 1000
Steps per Episode	T	100
Discount Factor	γ	0.95
Learning Rate	α	0.0003
Batch Size	N	64
Replay Buffer Size	B	2000
Optimizer	-	Adam [14]

The reward function of the goal achievement experiment in each learning technique is defined as:

$$R_t = \frac{d_t(P_p, P_t) - d_t(P_a, P_t)}{d_t(P_i, P_t)} + R_s, \tag{9}$$

where $d_t(P_i, P_t)$ is the initial Euclidean distance between the start P_i and target point P_t, and

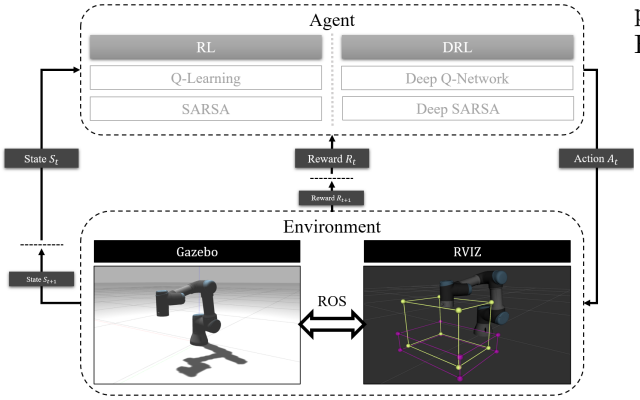


Figure 4: Agent-environment interaction in Markov's decision-making process (MDPs) in our problem. The environment in our problem represents the communication between the RVIZ [7, 26] simulation tool and the Gazebo [1, 15] 3D modeling tool via ROS [25]. The agent represents the RL/DRL methods used in our problem.

$(d_t(P_p, P_t) - d_t(P_a, P_t))$ is the Euclidean distance difference in real time (P_p – previous position, P_a – actual position). The parameter R_s assumes values higher than 0 when the condition is successfully done, which is determined by the accuracy of the results (Eq. 10).

$$R_s = \begin{cases} 0.25, & \text{if } \delta_d \leq 5\%. \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

where δ_d is the error of accuracy of the Euclidean distance, defined as:

$$\delta_d = \frac{d_t(P_a, P_t)}{d_t(P_i, P_t)}.100. \tag{11}$$

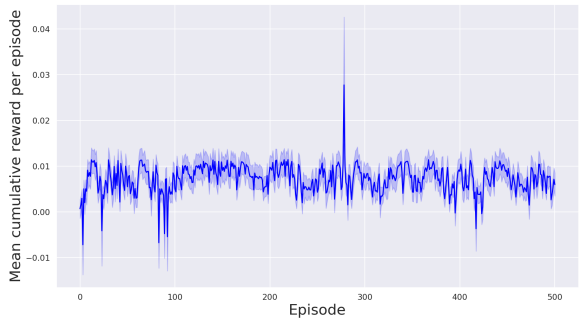
The learning process of the RL/DRL agent begins by examining the environment by performing actions from the initial state to the target state and collecting appropriate rewards (Eq. 9). In our experiment, the agent can select one of six possible actions ($a_t = (0..5)$) in each state ($s_t = (X_+, X_-, Y_+, Y_-, Z_+, Z_-)$), and the available actions correspond to fixed discrete steps 5 mm of the Tool Center Point (TCP).

Once the agent moves from A_{search}, the episode M ends in the current step T, otherwise the process continues until the maximum number of episodes.

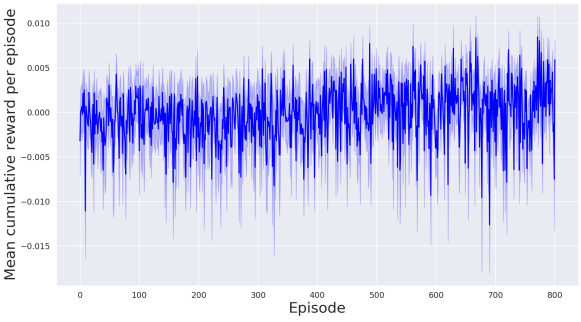
4.4 Experimental results

The training results of the proposed target achievement experiment using the UR3 robotic arm are shown in Fig. 5 (a. Q-Learning, b. SARSA, c. Deep Q-Network, d. Deep SARSA), including the mean cumulative reward with the number of iterations in each of the episodes.

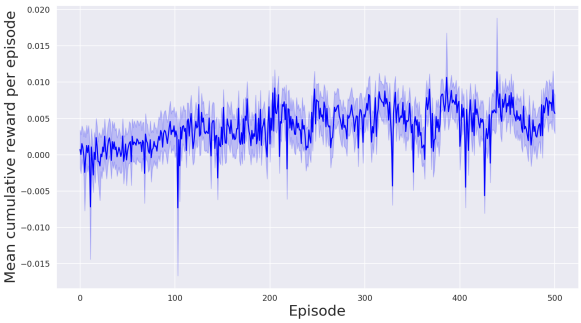
The main goal of the optimization problem in the case of the learning process was to maximize the expected cumulative reward (Fig. 5) and to minimize the Euclidean distance accuracy error (Fig. 6).



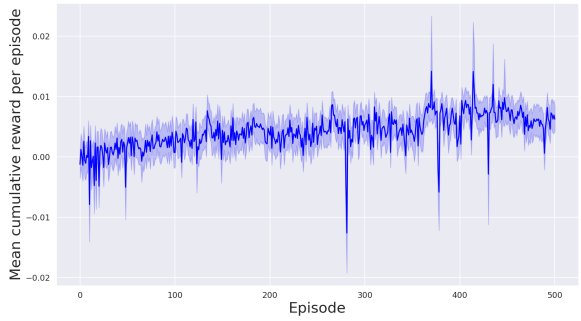
(a) Q-Learning



(b) SARSA



(c) Deep Q-Network



(d) Deep SARSA

Figure 5: Training results of RL/DRL techniques using the environment to achieve the goal of the UR3 robot.

$\delta_d \leq 5.0\%$ $\delta_d \leq 10.0\%$ $\delta_d > 10.0\%$

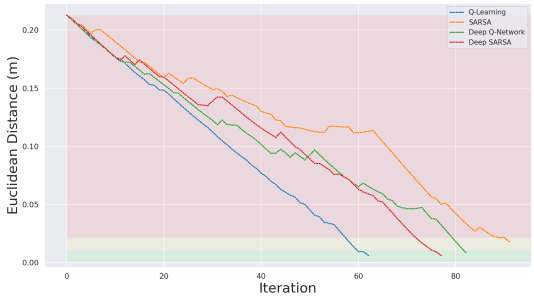
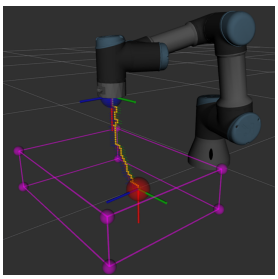
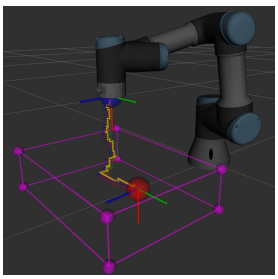


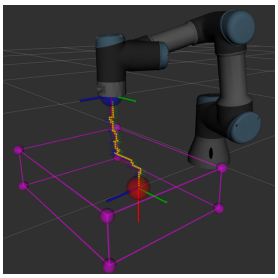
Figure 6: Minimization of Euclidean distance accuracy error for each of the learning techniques.



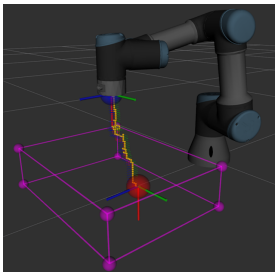
(a) Q-Learning



(b) SARSA

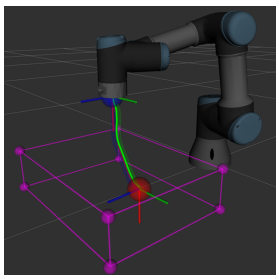


(c) Deep Q-Network

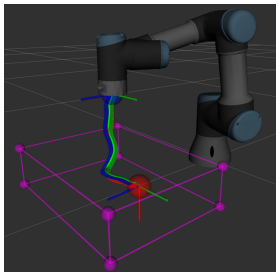


(d) Deep SARSA

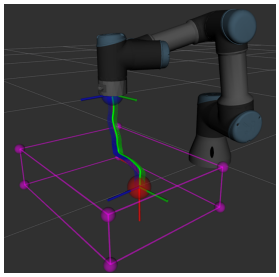
Figure 7: Test case of a robotic arm for the task of achieving a goal: Without trajectory optimization.



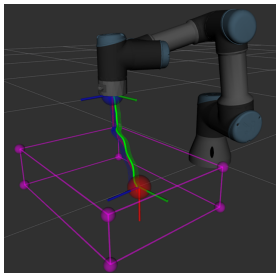
(a) Q-Learning



(b) SARSA



(c) Deep Q-Network

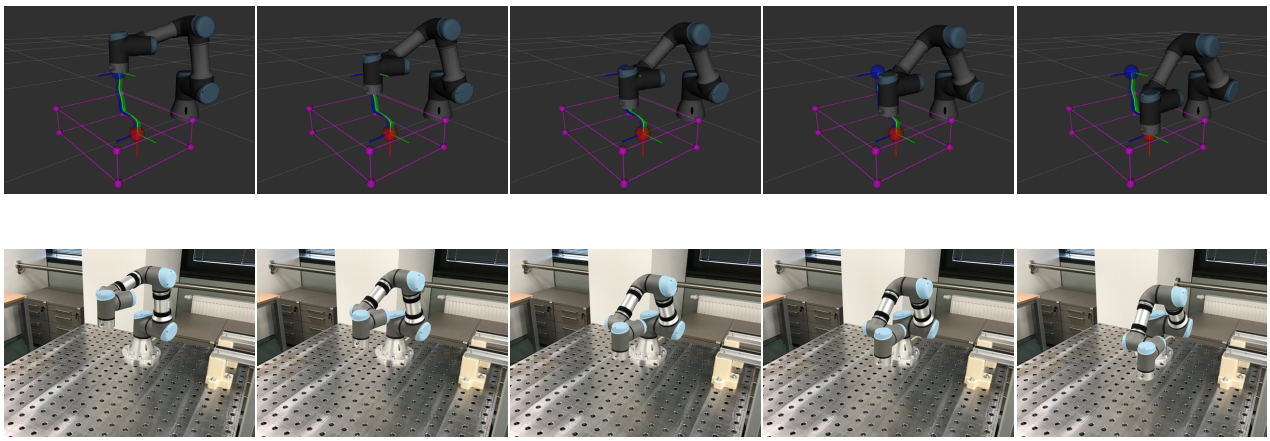


(d) Deep SARSA

Figure 8: Test case of a robotic arm for the task of achieving a goal: With trajectory optimization by smoothing the path using B-Spline.

After the learning process, the robotic arm UR3 is tested with the RVIZ simulation tool and the Gazebo 3D modeling tool, which communicate through ROS. In each learning technique, we choose the best solution and execute the movement of the planning path. The resulting path is smoothed by the Bézier spline method using the GEOMDL library [4]. The trajectory results of the proposed experiment in each scenario are shown in Fig. 7 without B-Spline, and Fig. 8 with B-Spline.

The result of the learning experiment is also tested in a real robotic application using the official ROS driver [3]. A demonstration of the proposed optimization path using the DQN learning technique is given in Fig. 9.



(a) 0% (b) 25% (c) 50% (d) 75% (e) 100%

Figure 9: Test case of a robotic arm for the task of achieving a goal in a real robot application. The experiment using the proposed optimization path is performed by the DQN learning technique.

Table 3: Results of RL/DRL learning techniques for the goal achievement experiment

Learning Technique	Number of episodes	Best solution (Episode; Time)	Number of points from P_i to P_t	Accuracy error δ_d	Time Default	Time B-Spline
Q-Learning	500	(279; 04:55 hr.)	63	2.98%	5.25 sec.	2.27 sec.
SARSA	800	(772; 18:29 hr.)	92	8.48%	7.83 sec.	4.33 sec.
Deep Q-Network	500	(440; 15:48 hr.)	83	4.16%	6.57 sec.	2.89 sec.
Deep SARSA	500	(371; 13:25 hr.)	78	2.90%	6.79 sec.	2.44 sec.

5 Conclusion and Future Work

In this work, we provided the experimental study of multiple reinforcement learning (RL)/deep reinforcement learning algorithms (DRL), namely Q-Learning (QL), SARSA for RL methods, and Deep Q-Network (DQN), Deep Sarsa (DSARSA) for DRL methods. We presented related work on the problem of achieving the goal using RL/DRL techniques. In this work, we also introduced in more detail the various learning methods and techniques for trajectory smoothing. The simulation is controlled by communication between the simulation tool RVIZ and the Gazebo 3D modeling tool via the Robot Operating System (ROS). The main optimization problem of this experiment is to find the best solution for each RL/DRL scenario, respectively, minimize the Euclidean distance accuracy error δ_d. RL (QL) and DRL (DQN, DSARSA) techniques completed the conditions for the required accuracy represented by R_s, but in the perspective of the future research are techniques based on deep neural network are more stable and efficient. The resulting path found in each scenario is smoothed by the Bézier spline method and tested in a real robotic application using the official ROS driver.

This work can provide a foundation for future research on motion planning in the field of robotics using advanced deep reinforcement learning methods such as DDPG (Deep Deterministic Policy Gradient), TD3 (Twin Delayed Deep Deterministic Policy Gradient) and more. This research can also provide a suitable basis for other areas of learning, such as the Pick & Place task, Bin-picking, etc., and the use of other robotic arm models.

Acknowledgement: This work was supported by Internal grant agency of BUT: FME-S-20-6538 “Industry 4.0 and AI methods”.

References

[1] Aguero, C., et al. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on 12*, 2 (April 2015), 494–506.

[2] Ammad, M., and Ramli, A. Cubic b-spline curve interpolation with arbitrary derivatives on its data points. In *2019 23rd International Conference in Information Visualization – Part II* (2019), pp. 156–159.

[3] Andersen, T. T. Optimizing the universal robots ros driver. Technical University of Denmark, Department of Electrical Engineering (2015).

[4] Bingol, O. R., and Krishnamurthy, A. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX 9* (2019), 85–94.

[5] Bogunowicz, D., Rybnikov, A., Vendidandi, K., and Chervinskii, F. Sim2real for peg-hole insertion with eye-in-hand camera. *arXiv:2005.14401* (05 2020).

[6] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv:1606.01540* (2016).

[7] Coleman, D., Şucan, I. A., Chitta, S., and Correll, N. Reducing the barrier to entry of complex robotic software: a moveit!case study. *Journal of Software Engineering for Robotics 5* (2014), 3–16.

[8] Coumans, E., and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[9] El-Shamouty, M., Wu, X., Yang, S., Albus, M., and Huber, M. F. Towards safe human-robot collaboration using deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 4899–4905.

[10] Franceschetti, A., Tosello, E., Castaman, N., and Ghidoni, S. Robotic arm control and task training through deep reinforcement learning. *arXiv:2005.02632* (01 2021).

[11] Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. An introduction to deep reinforcement learning. *arXiv:1811.12560* (2018).

[12] Hundt, A., et al. "good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters PP* (08 2020), 1–1.

[13] Hůlka, T., Matoušek, R., Dobrovský, L., Dosoudilová, M., and Nolle, L. Optimization of snake-like robot locomotion using ga: Serpenoid design. *MENDEL 26*, 1 (Aug. 2020), 1–6.

[14] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014).

[15] Koenig, N., and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sendai, Japan, Sep 2004), pp. 2149–2154.

[16] Kristensen, C., Sørensen, F., Nielsen, H., Andersen, M., Bendtsen, S., and Bøgh, S. Towards a robot simulation framework for e-waste disassembly using reinforcement learning. *Procedia Manufacturing 38* (01 2019), 225–232.

[17] Kudela, J. Social distancing as p-dispersion problem. *IEEE Access 8* (2020), 149402–149411.

[18] Lin, C., and Li, M. Motion planning with obstacle avoidance of an ur3 robot using charge system search. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)* (2018), pp. 746–750.

[19] Mahmood, A., Korenkevych, D., Komer, B., and Bergstra, J. Setting up a reinforcement learning task with a real-world robot. *arXiv:1803.07067* (03 2018).

[20] Mesquita, A., Nogueira, Y., Vidal, C., Cavalcante-Neto, J., and Serafim, P. Autonomous foraging with sarsa-based deep reinforcement learning. In *2020 22nd Symposium on Virtual and Augmented Reality (SVR)* (2020), pp. 425–433.

[21] Nguyen, H., and La, H. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)* (2019), pp. 590–595.

[22] Rupam Mahmood, A., Korenkevych, D., Komer, B. J., and Bergstra, J. Setting up a reinforcement learning task with a real-world robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 4635–4640.

[23] Scheiderer, C., Thun, T., and Meisen, T. Bézier curve based continuous and smooth motion planning for self-learning industrial robots. *Procedia Manufacturing 38* (2019), 423 – 430. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[24] Silver, D., et al. Mastering the game of go with deep neural networks and tree search. *Nature 529* (01 2016), 484–489.

[25] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[26] Sucan, I. A., and Chitta, S. Moveit. [online] Available at: moveit.ros.org.

[27] Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction*, second ed. The MIT press, 2018.

[28] Universal Robots. Ur3. [online] Available at: https://www.universal-robots.com.

[29] van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. *arxiv:1509.06461* (2015).

[30] Vince, J. *Mathematics for Computer Graphics*, fifth ed. Springer, London, 2017.

[31] Xinyu, W., Xiaojuan, L., Yong, G., Jiadong, S., and Rui, W. Bidirectional potential guided rrt* for motion planning. *IEEE Access 7* (2019), 95046–95057.

[32] Zamora, I., Lopez, N., Vilches, V., and Cordero, A. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv:1608.05742* (08 2016).

[33] Zeng, X. Reinforcement learning based approach for the navigation of a pipe-inspection robot at sharp pipe corners. University of Twente, September 2019.