Intelligent Systems with Applications 21 (2024) 200313

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

A Q-learning approach to the continuous control problem of robot inverted pendulum balancing

Mohammad Safeea, Pedro Neto *

University of Coimbra, CEMMPRE, ARISE, Department of Mechanical Engineering, 3030-788, Coimbra, Portugal

ARTICLE INFO

Keywords:
Q-learning
Reinforcement learning
Robotics
Pendulum balancing

ABSTRACT

This study evaluates the application of a discrete action space reinforcement learning method (Q-learning) to the continuous control problem of robot inverted pendulum balancing. To speed up the learning process and to overcome technical difficulties related to the direct learning on the real robotic system, the learning phase is performed in simulation environment. A mathematical model of the system dynamics is implemented, deduced by curve fitting on data acquired from the real system. The proposed approach demonstrated feasible, featuring its application on a real world robot that learned to balance an inverted pendulum. This study also reinforces and demonstrates the importance of an accurate representation of the physical world in simulation to achieve a more efficient implementation of reinforcement learning algorithms in real world, even when using a discrete action space algorithm to control a continuous action.

1. Introduction

Reinforcement learning (RL) is revolutionizing many fields. In robotics, RL is enabling robots to expand their autonomy and learning capabilities beyond strictly structured environments into unstructured environments (Sutton & Barto, 2018, Ju et al., 2022). RL promotes robot learning through trial and error, receiving feedback in the form of rewards or penalties for its actions, and learning complex actions such as human-level skills (Mnih, Kavukcuoglu, Silver, et al., 2015), or to fly a model helicopter (Kim et al., 2004). A reference study addresses policy gradient RL applied to an actuated passive dynamic walker that learned a feedback control policy for performing dynamic walking (Tedrake et al., 2004). The policy was learned in twenty minutes while running online in the real robot. RL has also been successfully applied for robot learning in the context of robot shared autonomy (Rigter et al., 2020) and in position/force control of robot manipulators (Perrusquía et al., 2019).

The problem of environment exploration with sparse rewards is a challenge and prevents the use of RL in some real-world tasks. Researchers propose to use demonstrations to overcome the exploration problem and learn to perform long-horizon multi-step robotics tasks (Nair et al., 2018). To limit the need for real world interactions while learning a policy, models are often used as simulation systems, training the policy in simulation and later transferring such policy to the real world (Zhao et al., 2020). This process is commonly named as RL sim-to-real, featuring successfully application on robot grasping control (Yan et al., 2017) and self-supervised robotic manipulation (Jeong et al., 2020). Training and learning RL policies in simulation brings great advantages to the process, as the learning in real world involves long periods of experiments with robotic equipment to handle disturbances, requiring human supervision, besides that the learning process may damage the equipment. In summary, this process is expensive, unsafe and time consuming. On the other hand, creating accurate simulation models is challenging and sometimes even impracticable. Small errors due to under-modeling can accumulate and make the simulated robot to diverge from the real-world robot, which limits the performance when transferring the learned policies to the real system (Kober et al., 2013). Sim-to-real learning policies often fail to transfer the learning behaviors to real world. In such a context, these systems can be improved by injecting noise in simulation and/or taking data from the real world to minimize the simulation-to-real gap and by this way improve learning policies. Recent studies demonstrate reinforcement learning solutions where only a single demonstration trajectory is required to learn behaviors (Pavse et al., 2020), or a new sim-to-real technique that allows end-to-end training (Karnan et al., 2020).

The application of a discrete action space method like Q-learning in continuous state and action spaces modeling has been studied along the years (Jang et al., 2019, Gaskett et al., 1999, Sabir et al., 2022).

* Corresponding author.
E-mail addresses: ms@uc.pt (M. Safeea), pedro.neto@dem.uc.pt (P. Neto).

https://doi.org/10.1016/j.iswa.2023.200313
Received 7 February 2022; Received in revised form 23 October 2023; Accepted 28 November 2023
Available online 5 December 2023

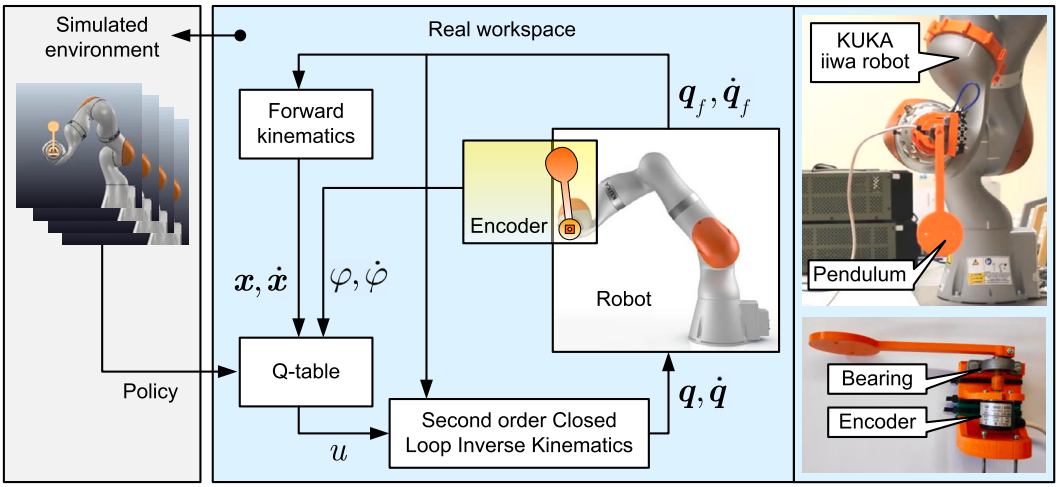

Fig. 1. Architecture of the proposed Q-learning approach to the problem of robot inverted pendulum balancing, featuring in/out monitoring and control data. The hardware elements of the system are highlighted, namely the robot and the pendulum mechanism.

Recently, normalized advantage functions allowed to apply Q-learning with experience replay to continuous tasks, substantially improving performance on robot control (Gu et al., 2016). Continuous action Q-learning demonstrated good performance in heavily constrained environments when compared to RL algorithms on continuous control problems (Ryu et al., 2019). Deep Q-network (DQN) algorithms, combining Q-learning with deep neural networks (DNNs), allows an agent to learn directly from high dimensional sensory inputs (Mnih, Kavukcuoglu, et al., 2015). DQN has been used for active simultaneous localization and mapping (SLAM) for autonomous navigation featuring static and dynamic obstacles (Wen et al., 2021), motion attitude control of humanoid robots (Shi et al., 2020), and for assembly sequence planning problem (Neves & Neto, 2022).

When solving complex high-dimensional problems, standard policy gradient methods often require many iterations and are prone to poor local optima. To solve these difficulties the algorithm Guided Policy Search (GPS) uses trajectory optimization to guide the policy away from poor local optima (Levine et al., 2015). In the presence of a small number of real-world samples, the resulting neural network (considering control policies represented as high-dimensional neural networks deriving robot actions based on states) is only robust in the neighborhood of the trajectory distribution explored by real-world interactions. Generative Motor Reflexes (GMR) was introduced to tackle this exact problem, improving robustness by using motor reflexes and stabilizing actions (Ennen et al., 2019). A robot manipulator was trained in a reaching task and in a task where the robot was expected to place a block in a hole using GMR, both in simulation and in the real-world. GMR proved itself as more robust than the GPS algorithm (Ennen et al., 2019). As discussed above, learning real world tasks from scratch is a complex problem requiring significant training time and high sample-complexity. To tackle this issue, an asynchronous version of the Normalized Advantage Function (NAF) algorithm, parallel NAF, was used in a variety of robotic tasks both in simulation and in the real-world (Gu et al., 2017). This algorithm allows several robots to learn simultaneously in an asynchronous fashion, reducing the training time proportionally to the number of robots in use. RL has also been used to improve assembly efficiency recurring to dual-arm robots, reducing the efforts in robot teaching and process planning (Watanabe & Inada, 2020).

2. Proposed approach and background

RL is a powerful learning tool. However, its application in real-world robotic systems is still challenging. In this study it is proposed to apply RL, tabular Q-learning, to the continuous control problem of robot inverted pendulum balancing, Fig. 1. The proposed methodology is trained in simulation recurring to the Virtual Robot Experimentation Platform (V-REP, CoppeliaSim) (Rohmer et al., 2013), and then evaluated on a real-world robot. A mathematical model of the system dynamics is deduced by curve fitting on data acquired from the real-world system. The setup is composed by a robot manipulator and the pendulum mechanism. The pendulum and the encoder are attached to the same spindle hoisted inside the bearing. The bearing and the encoder are mounted on a 3D printed base that is attached to the flange of the robot.

2.1. Background

Given a Markov Decision Process (MDP) defined by a set of states s, a set of actions a, and a transition function $P(\grave{s}|s,a)$, it is required to find an optimal policy π^* which maximizes the expected reward. Q-learning is a model free RL method that can be implemented to achieve the control policy π^* (Watkins & Dayan, 1992), while the agent learns during its interaction with the environment. The table covers the discretized state-action space, containing scalar values indicating how fit an action is for a given state. During the learning process the table is updated in each iteration to maximize the expected reward while the agent is interacting with the environment. The learning process is represented mathematically by:

$$Q(s,a) = (1-\alpha)\,Q(s,a) + \alpha\left(r + \gamma max_{\grave{a}}\left(Q(\grave{s},\grave{a})\right)\right) \quad (1)$$

Where $Q(s,a)$ is the state-action value for the action a at the state s, α is the learning rate, r is the reward, γ is the discount ratio, and $\grave{s}$ is the resulting (observed state) after applying the action a. The algorithm can be applied for controlling dynamical systems where the resulting policy can be used for generating the control signals required to stabilize the physical system.

3. Methodologies

3.1. Robot acceleration tracking

The accelerations required to balance the pendulum are the RL actions. However, the robot is a series of rigid links connected by joints, and thus, the robot moves by actuating such joints. The robot can track the commanded acceleration of the flange by using a second order Closed Loop Inverse Kinematics (CLIK) algorithm (Siciliano, 1990). From differential kinematics the following relationship between the acceleration of the joints and the acceleration of the end-effector (EEF) can be established:

$$\Gamma = \mathbf{J}\ddot{\boldsymbol{q}} + \dot{\mathbf{J}}\dot{\boldsymbol{q}} \tag{2}$$

Where, Γ is the acceleration of the EEF (robot flange), $\mathbf{J}$ is the Jacobian matrix of the robotic manipulator, $\ddot{\boldsymbol{q}}$ is the angular acceleration vector of the robot joints, $\dot{\mathbf{J}}$ is the time derivative of the Jacobian matrix, and $\dot{\boldsymbol{q}}$ is the angular velocity vector of the robot joints. The robot is 7 degrees of freedom (DOF) redundant manipulator. However, to simplify the experiment we fix the redundant joint (third) of the robot. Thus, the resulting Jacobian is square and the joints angular accelerations are calculated:

$$\ddot{\boldsymbol{q}} = \mathbf{J}^{-1}(\Gamma + \mathbf{K}_d\dot{\boldsymbol{e}} + \mathbf{K}_p\boldsymbol{e} - \dot{\mathbf{J}}\dot{\boldsymbol{q}}) \tag{3}$$

Where $\mathbf{K}_d$ and $\mathbf{K}_p$ are the derivative and proportional gains, both positive definite matrices. The position/orientation error at the EEF level is $\boldsymbol{e}$, and $\dot{\boldsymbol{e}}$ is the linear/angular velocity error at the EEF.

3.2. Mathematical model of the pendulum

The state of the pendulum at any instant is specified by the vector $\boldsymbol{s}$:

$$\boldsymbol{s} = \begin{bmatrix} x \\ \dot{x} \\ \varphi \\ \dot{\varphi} \end{bmatrix} \tag{4}$$

Where x and $\dot{x}$ are the linear displacement and velocity of the robot's flange along the x axis of its base, respectively. The angular position φ and velocity $\dot{\varphi}$ of the pendulum are measured from the vertical upright position (in the unstable fixed point). Thus, the control problem requires to stabilize the system at the state $\boldsymbol{s}^* = [0,0,0,0]^{\mathrm{T}}$. On the other hand, the robot is controlled at the joints position level. By moving its joints, the robot constrains the motion of its flange along the x axis and tries to balance the pendulum in the unstable fixed point (vertical upright position). Thus, the control input u is the acceleration of the flange along the x axis:

$$u = \ddot{x} \tag{5}$$

The tracking acceleration of the robot along the x axis is given by:

$$\grave{u} = n^{\mathrm{T}}\grave{\Gamma} \tag{6}$$

Where n^{T} is equal to the vector $[1,0,0,0,0,0]$, and $\grave{\Gamma}$ is the actual (tracking) acceleration at the robot flange due to joints acceleration/velocities (2), as a result of the control command u, and by considering the error dynamics e and $\dot{e}$. In such a case the state transition equation of the proposed system near the unstable fixed point is given by:

$$\dot{\boldsymbol{s}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{mgl}{I} & \frac{-b}{I} \end{bmatrix} \boldsymbol{s} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ \frac{-ml}{I} \end{bmatrix} \grave{u} \tag{7}$$

Where I is the moment of inertia of the pendulum and the rotary part of the encoder around the rotation axis, b is the viscus friction coefficient, m is the mass of the pendulum, g is the gravity acceleration, l is the distance from the center of mass of the pendulum to the rotation axis, and $\grave{u}$ is the actual (tracking) acceleration of the robot's flange along the x axis.

For the proposed system, the state equation (7) is used, so that the system learns the commanded accelerations u required to balance the pendulum according to both the pendulum's dynamics and the robot's acceleration tracking dynamics. However, from (7) it is noticed that there are parameters that need to be estimated, specifically I and b. In such a context, it was conducted an experiment to achieve a precise estimation of the aforementioned constants. The mathematical equation of a pendulum rotating around a fixed axis without an external excitation is given by:

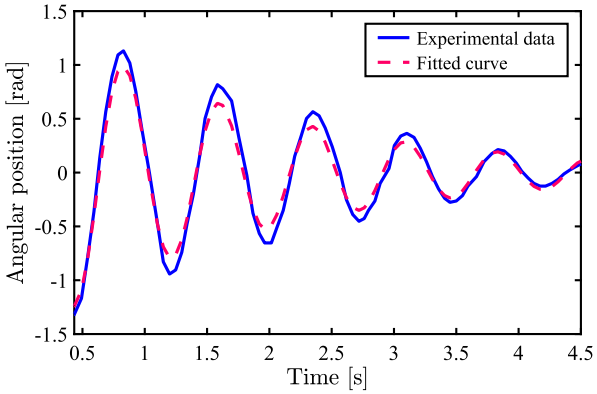

Fig. 2. Free oscillation of the pendulum. Encoder experimental data are plotted in continuous line, while the fitted curve is represented in dashed line.

$$I\ddot{\theta} + b\dot{\theta} + mgl\,sin(\theta) = 0 \tag{8}$$

Where θ is the pendulum's angle measured from the vertical down position (the stable fixed point), so that $\theta + \varphi = \pi$. The angular velocity and acceleration of the pendulum under free oscillation are represented by $\dot{\theta}$ and $\ddot{\theta}$, respectively. From equation (8), the values of I and b can be estimated by the angular coordinate θ of the pendulum (from the encoder) while it is freely oscillating due to an initial angular displacement, i.e., inducing free oscillations in the pendulum while capturing θ, and then by fitting a curve into the resulting data according to the mathematical model in (8). Fig. 2 shows a plot of the captured data in addition to the resulting fitted curve, corresponding to the resulting estimates of I and b. In this experiment, the pendulum has a small distance from the center of mass to the rotation axis l. This is reflected on the characteristics of the plot in Fig. 2, where the period of the natural oscillation is around 0.75 seconds, and therefore, the task of balancing the inverted pendulum is challenging, even for a human.

4. Results and discussion

4.1. Discretization

Fig. 3 shows the policy training process in simulation and the robotic system operating in real world environment (video in multimedia materials). Different quantities were discretized, namely:

1. The control command u (linear acceleration of the robot flange) was discretized into eight different actions: $\{-2,-1.5,-1,-0.5,0.5,1,1.5,2\}$ m/s^2;
2. The state of the system, i.e., the angular position of the pendulum φ (taken from the vertical upright position) was discretized into six different states in the intervals: $]-11°,-5°[$, $[-5°,-1°[$, $[-1°,0°[$, $[0°,1°[$, $[1°,5°[$ and $[5°,11°[$ degrees;
3. The angular velocity of the pendulum $\dot{\varphi}$ was discretized into five different states in the intervals: $]-\infty,-50]$, $]-50,-10]$, $]-10,10[$, $[10,50[$ and $[50,+\infty[$ degrees per second;
4. The linear position of the flange x was discretized into three different states in the intervals: $]-0.22,-0.08]$, $]-0.08,0.08[$ and $[0.08,0.22[$ meters;
5. The linear velocity of the flange $\dot{x}$ was discretized into three different states in the intervals: $]-\infty,-0.5]$, $]-0.5,0.5[$ and $[0.5,+\infty[$ m/sec.

A failure is declared when the angular position φ exceeds any of the limits $\{+11°,-11°\}$ degrees, or when the flange exceeds any of the limits $\{+0.22,-0.22\}$ meters.

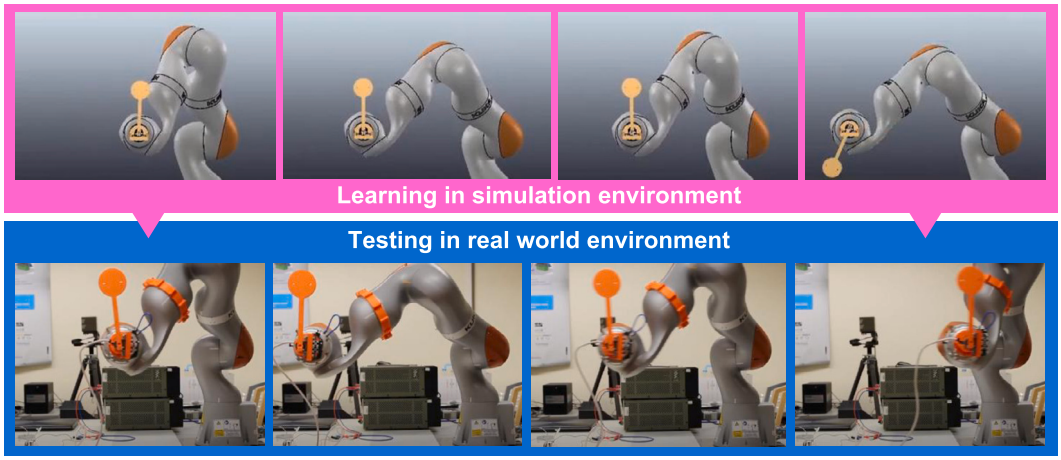

Fig. 3. RL policy training in V-REP simulation environment and policy evaluation on the real world environment.

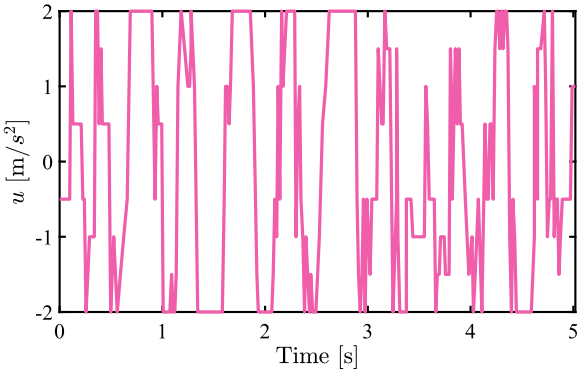

Fig. 4. Discretized control commands u featuring the linear acceleration of the robot flange.

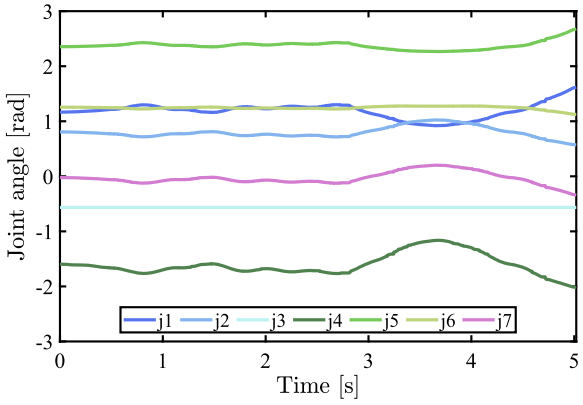

Fig. 5. Robot joints positions reflecting the control commands in Fig. 4.

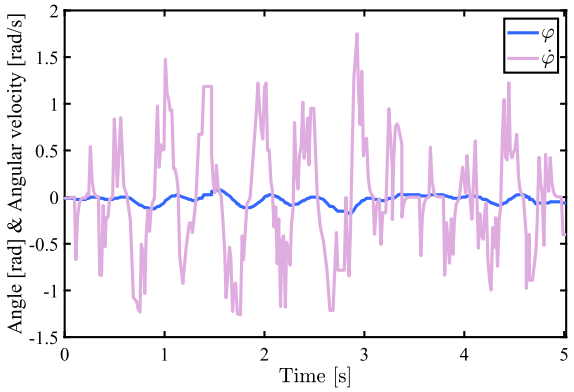

Fig. 6. Angular position and velocity of the pendulum in real world experiments.

4.2. Simulation

The simulations were performed in V-REP, Fig. 3, using the remote-API to control it from MATLAB where the mathematical model of the system was implemented, equations (7) and (3). The Q-learning control policy was trained in 10,000 episodes, where for each episode was injected a small noise into the estimated parameters I and b. A time step $h = 0.01$ seconds is used and a small noise is injected into the error e for each iteration.

4.3. Real world system

The policy trained in simulation is transferred to control the real system. A computer running a real time operating system is used to control the robot (Safeea & Neto, 2019), and a trajectory generation algorithm is implemented on the robot controller. The computer is also used to acquire data from the encoder and the motion feedback from the robot. From the acquired data, an appropriate action u is chosen according to the policy trained in simulation. Afterwards, the robot angular quantities, accelerations, velocities and positions are calculated according to the CLIK algorithm. The robot control commands defined are shown in Fig. 4 and the resulting robot commanded angular positions (robot joints) are in Fig. 5.

The angular position/velocity of the pendulum is shown in Fig. 6. From this figure it can be seen that the angular position of the pendulum stays inside the limits. The linear position/velocity plot of the robot's flange is shown in Fig. 7, where the linear position is calculated as the displacement from a starting home position. In the proposed experiment the system was able to balance the pendulum for the first 5 seconds, however, it failed eventually, due to the violation of the flange's position, exceeding the limit at −0.22 meters, Fig. 7 (at this instant the pendulum's angle is −3.44).

4.4. Discussion

The proposed approach, training the control policy using Q-learning in simulation environment and transferring it to real world, offers various advantages, namely:

1. Speeding up the training process (time efficiency);
2. Reducing the risk to damage hardware;

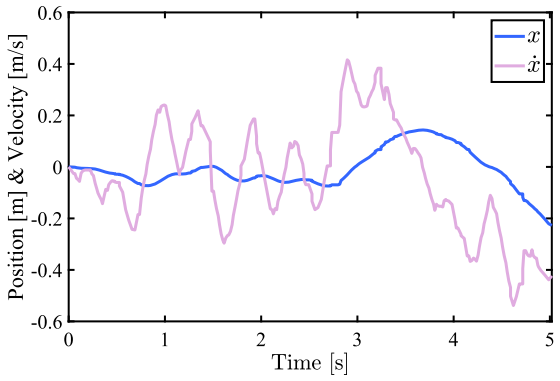

Fig. 7. Linear displacement and velocity of the flange along x axis in real world experiment.

3. The ability to start the process from various initial states which might not be easily performed using real equipment.

The RL policy was trained using 10,000 episodes. While it took few minutes for training on a computer, performing the same number of trials on the real robotic system would take considerable amount of time. Note that the required time for the experiment also includes the preparation phase, to be performed at the beginning of each trial (training episode). For example, the encoder used in the experiment is an incremental encoder, consequently, at the beginning of each training episode it shall be referenced by allowing the pendulum to settle in the vertical downward position (stable fixed point) which is used as the reference. The referencing is done by pressing a button. The server application on the robot should be running, and the control program on the computer turned on. Finally, a time-delay is used in the control program to allow the user to place the pendulum in the vertical upright position before starting the control loop.

The real world system requires the user to set the pendulum upwards near the vertical by hand at the beginning of each training episode. Afterwards, the user shall release the pendulum immediately when the control loop is activated (prompted by a message on the computer screen). This procedure, the coordination between the sight of the visual notification and the immediate release, proved challenging for the users, especially because in this particular application the period of the natural oscillations of the pendulum is small, resulting in the pendulum falling rapidly and colliding with the user's hand, ruining the experiment.

An accurate dynamical model is key to achieve good results in sim-to-real transfer. However, the process to obtain such model is challenging, as well as the process to achieve good results in the presence of an inaccurate model. It is shown that even tabular Q-learning can achieve a feasible solution despite the loss of information due to discretization.

5. Conclusion

This study evaluated the performance of a discrete action space RL method (Q-learning) to the continuous control problem of robot inverted pendulum balancing. The control policy, trained in simulation environment and transferred to the real robot, demonstrated feasible when the system is accurately modeled. In such a condition, it can be concluded that a discrete action space algorithm can be used to control a continuous action. The advantages the simulation brings to the process are multiple, namely the reduced training time, reduced risk to damage hardware and the ability to start the training from various initial states. The learned policy can fail in some conditions due to some remaining unmodeled dynamics and the perturbations in the real system which will always exist. Future work will be dedicated to improving the control policy with more information from the real world environment.

CRediT authorship contribution statement

Mohammad Safeea: Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **Pedro Neto:** Data curation, Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This research is sponsored by national funds through Fundação para a Ciência e a Tecnologia, under the project UIDB/00285/2020 and LA/P/0112/2020.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.iswa.2023.200313.

References

Ennen, P., Bresenitz, P., Vossen, R., & Hees, F. (2019). Learning robust manipulation skills with guided policy search via generative motor reflexes. In *2019 international conference on robotics and automation (ICRA)* (pp. 7851–7857).

Gaskett, C., Wettergreen, D., & Zelinsky, A. (1999). Q-learning in continuous state and action spaces. In *Australasian joint conference on artificial intelligence* (pp. 417–428). Springer.

Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)* (pp. 3389–3396).

Gu, S., Lillicrap, T., Sutskever, I., & Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *International conference on machine learning* (pp. 2829–2838). PMLR.

Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, *7*, 133653–133667. https://doi.org/10.1109/ACCESS.2019.2941229.

Jeong, R., Aytar, Y., Khosid, D., Zhou, Y., Kay, J., Lampe, T., Bousmalis, K., & Nori, F. (2020). Self-supervised sim-to-real adaptation for visual robotic manipulation. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 2718–2724). IEEE.

Ju, H., Juan, R., Gomez, R., Nakamura, K., & Li, G. (2022). Transferring policy of deep reinforcement learning from simulation to reality for robotics. *Nature Machine Intelligence*, *4*, 1077–1087. https://doi.org/10.1038/s42256-022-00573-6.

Karnan, H., Desai, S., Hanna, J. P., Warnell, G., & Stone, P. (2020). Reinforced grounded action transformation for sim-to-real transfer. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4397–4402).

Kim, H. J., Jordan, M. I., Sastry, S., & Ng, A. Y. (2004). Autonomous helicopter flight via reinforcement learning. In *Advances in neural information processing systems* (pp. 799–806).

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*, 1238–1274. https://doi.org/10.1177/0278364913495721.

Levine, S., Wagener, N., & Abbeel, P. (2015). Learning contact-rich manipulation skills with guided policy search. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 156–163).

Mnih, V., Kavukcuoglu, K., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533.

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*, 529–533.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 6292–6299).

Neves, M., & Neto, P. (2022). Deep reinforcement learning applied to an assembly sequence planning problem with user preferences. *The International Journal of Advanced Manufacturing Technology*, *122*, 4235–4245. https://doi.org/10.1007/s00170-022-09877-8.

Pavse, B. S., Torabi, F., Hanna, J., Warnell, G., & Stone, P. (2020). Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration. *IEEE Robotics and Automation Letters*, *5*, 6262–6269.

Perrusquía, A., Yu, W., & Soria, A. (2019). Position/force control of robot manipulators using reinforcement learning. *Industrial Robot*, *46*, 267–280.

Rigter, M., Lacerda, B., & Hawes, N. (2020). A framework for learning from demonstration with minimal human effort. *IEEE Robotics and Automation Letters*, *5*, 2023–2030. https://doi.org/10.1109/LRA.2020.2970619.

Rohmer, E., Singh, S. P., & Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE (pp. 1321–1326).

Ryu, M., Chow, Y., Anderson, R., Tjandraatmadja, C., & Boutilier, C. (2019). Caql: Continuous action q-learning. arXiv preprint arXiv:1909.12397.

Sabir, Z., Said, S. B., Al-Mdallal, Q., & Ali, M. R. (2022). A neuro swarm procedure to solve the novel second order perturbed delay Lane-Emden model arising in astrophysics. *Scientific Reports*, *12*. https://doi.org/10.1038/s41598-022-26566-4.

Safeea, M., & Neto, P. (2019). Kuka sunrise toolbox: Interfacing collaborative robots with matlab. *IEEE Robotics & Automation Magazine*, *26*, 91–96.

Shi, Q., Ying, W., Lv, L., & Xie, J. (2020). Deep reinforcement learning-based attitude motion control for humanoid robots with stability constraints. *Industrial Robot*, *47*, 335–347. https://doi.org/10.1108/IR-11-2019-0240.

Siciliano, B. (1990). A closed-loop inverse kinematic scheme for on-line joint-based robot control. *Robotica*, *8*, 231–243.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Tedrake, R., Zhang, T. W., & Seung, H. S. (2004). Stochastic policy gradient reinforcement learning on a simple 3d biped. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE cat. no. 04CH37566)*. *IEEE: Vol. 3* (pp. 2849–2854).

Watanabe, K., & Inada, S. (2020). Search algorithm of the assembly sequence of products by using past learning results. *International Journal of Production Economics*, *226*, Article 107615. https://doi.org/10.1016/j.ijpe.2020.107615. http://www.sciencedirect.com/science/article/pii/S0925527320300037.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Wen, S., Lv, X., Lam, H., Fan, S., Yuan, X., & Chen, M. (2021). Probability dueling dqn active visual slam for autonomous navigation in indoor environment. *Industrial Robot*. https://doi.org/10.1108/IR-08-2020-0160.

Yan, M., Frosio, I., Tyree, S., & Kautz, J. (2017). *Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control.*

Zhao, W., Queralta, J. P., & Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *2020 IEEE symposium series on computational intelligence (SSCI)* (pp. 737–744). IEEE.