Employing of RL Technology to Develop an Adaptive Motion Controller for a Line Follower Robot

Tatyana Kim
Open Laboratory of Artificial Intelligence and Robotics
United Institute of Informatics Problems of NAS of Belarus
Minsk, Belarus
tatyana_kim92@mail.ru

Ryhor Prakapovich
Open Laboratory of Artificial Intelligence and Robotics
United Institute of Informatics Problems of NAS of Belarus
Minsk, Belarus
rprakapovich@robotics.by

Abstract. **The article is focused on the development process of an adaptive motion controller for a line follower robot. The controller learning process took place on the basis of the digital twin of the mobile robot using reinforcement learning technology. The digital twin and the reinforcement learning algorithm were implemented in MATLAB/Simulink. The Twin–Delayed Deep Deterministic Policy Gradient Agents method was used as a learning algorithm. The reward function was taken to minimize the distance between the center of the robot and the middle of the nearest section of the color–contrast line, as well as the difference between the angle of the robot position and the tangent to the current section of the line.**

Keywords: **Reinforcement Learning (RL), MATLAB/ Simulink, Twin-Delayed Deep Deterministic Policy Gradient Agents (TD3), control system, digital twin**

I. INTRODUCTION

Reinforcement learning (RL) is one of the machine learning methods for solving control problems in complex technical systems that cannot or can be problematically described in an analytical form.

The RL method is based on the implementation of the process of maximizing a certain reward (reward) signal when enumerating various behaviors of the studied systems – Agents. The Agent learns to perform those actions that can bring him the greatest reward. In the most interesting and important cases, the Agent's actions can affect not only the local reward received immediately, but also the situation as a whole [1]. Forming a long–term reward is a rather difficult process, since a correctly formed reward will bring the best result and shorten the training time (the better the learning process is).

At present, in addition to classical robotic manipulators, mobile robots (MR) in the form of robotic carts are in high demand in production. As a rule, at the lower level of MR control, PID controllers are most often used [2, 3]. The PID controller allows to adjust the control action of the actuators in such a way as to achieve the required values of the objective function as quickly as possible. Sometimes the selection of coefficients is a rather long process, which does not always lead to success, since there is a chance of overshoot [4, 5]. In closed–loop control systems (CS), the controller uses status observations to improve performance and correct random noise and errors. Engineers use this feedback, as well as the object of control (OC) and the Environment, to design the controller according to the requirements of the system. This concept is easy to put into words, but it will be difficult to implement, since the model can be highly nonlinear or have large spaces of states and actions. There is also a problem related to the fact that for each line type it is required to find the corresponding PID values. This problem can be quickly solved by using RL.

The object of the research is the RoboCake training MR with a differential drive, which is supposed to move along the color contrast line. Its CS is a classic servo drive, including a color contrast line midpoint sensor and a PID controller that controls the angular speed of the wheels. Previously, experiments were carried out on the automatic tuning of the specified PID controller using the software module MATLAB “Interactively Estimate Plant Parameters from Response Data” and the application of genetic algorithms (GA) [6].

The aim of this work is to implement automatic learning of a PID controller using the RL method and compare its effectiveness with other machine learning methods.

II. SUMMARY OF THE PROBLEM

In RL, the object of research is studied, as a result of which an artificial neural network (ANN) is generated that can simulate the desired object. The most important issue in RL remains the generation of the training sample for the specified ANN. There are

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

several types of Actor–Critic learning algorithms (Fig. 1). Actor–Critical Agents use either a stochastic Actor or a deterministic Actor with a value Critic or a Q–value Critic [7].

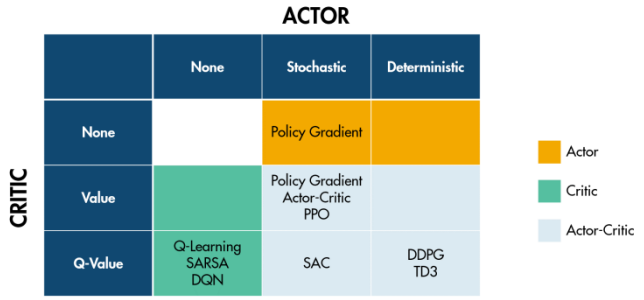

Fig. 1. Learning algorithms for Agent–Actor–Critic combined with stochastic–deterministic Actor and value–Q–value Critic [7]

As the investigated model, the work uses the DT [8] MR RoboCake, but developed in the MATLAB / Simulink packages. Following the RL methodology, in order to achieve this goal, it is required to describe the software Agent that needs to develop a policy for managing the educational institution. The specified MR is required to move along an elliptical curve (color–contrast line) with the maximum possible speed and the minimum deviation from its center. The term "policy" means a mapping that selects the appropriate actions of the OS on the corresponding changes in the Environment [9]. In the process of training, the Agent uses the following data: the readings of the sensor of the middle of the line (consisting of 3 light sensors), the distance from the center of the MR to the center of the color–contrast line, as well as the angle between the normal of the nearest section of the line and the direction of movement of the MR itself. The result of the observation is the selection of the angular speeds of rotation of the 2 MR wheels. In order for an Agent to form actions correctly, he needs to train repeatedly and for the training to be fruitful, for each action he must be encouraged (rewarded) or fined. Also, stopping criteria are used to reduce the learning time. Each episode of the learning process can stop if: 1) the simulation exceeds the time allotted for movement along a full ellipse; 2) the robot has exceeded the distance to the center of the ellipse line. During each episode, the Agent chooses an action (forms a policy), after the end, he updates his parameters based on the actions and receives the maximum reward. This process continues until the Agent learns to move correctly along the line with the specified conditions [9]. When developing this algorithm, the Reinforcement Learning Toolbox library was used, which provides a policy and a reward function using deep neural networks (DNNs) [10].

III. LEARNING PROCESS

Before starting the learning process, it is required to create a virtual Environment for the functioning of the MR and an interface for interacting with it. Next, you should configure the Agent module from the "Reinforcement Learning Toolbox" library [10]. To implement the Agent, the TD3 algorithm was chosen, which is characterized by the fact that in addition to the Agent, which offers specific actions of the Actor on certain indications of the sensory system, two new entities are also used – two Critics that form a long–term reward. Next, the Agent is configured and formed. The final stage is training and verification of the results of the trained Agent.

After a positive completion of the learning process, using the *getLearnableParameters()* command, weights are extracted from the trained ANN, which are the desired coefficients of the PID controller.

A. Reward function

The reward process is an important step in RL, as it affects the performance of the Agent in relation to the goal that CS MR seeks to achieve. A correctly selected reward signal forces the Agent to move in the right direction with a minimum deviation from the line and maximize the total reward received by the Agent over a long period of time, stimulating him (Agent) for long–term rewards. The reward is formed in the form of a scalar signal that is received by the Agent and generated by the Environment.

"Two criteria are taken into account as a reward for a committed action. First, the distance (r) from the center of the robot (x,y) to the nearest unvisited point (x_0 y_0) of the ellipse located on the line is calculated using the formula for calculating the distance between two points (1). The closer the Agent, the more reward he will receive, the further, the less reward" [11]. Secondly, calculate the angle (φ) between the tangent to the ellipse at the point (x_0 y_0), where the robot should be, and the robot guide using formula (2). The smaller the angle (φ), the more the Agent will receive a reward (Fig. 2).

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2} \quad (1)$$

$$\varphi = \arctan\left(\left|\frac{-b^2 * sin\,sin\,(\alpha) * R}{a^2 cos\,cos\,(\alpha) * R}\right|\right) \quad (2)$$

where b is the semi–minor axis, a is the semi–major axis, α is the angle between the radius (R) and the semi–major axis (a), R is the radius of the ellipse [12].

Since our reward signal consists of 2 signals combined into a scalar, then we determine that the distance between the line and the robot is of paramount importance, and the resulting angle (φ) we reduce the influence to get the reward.

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

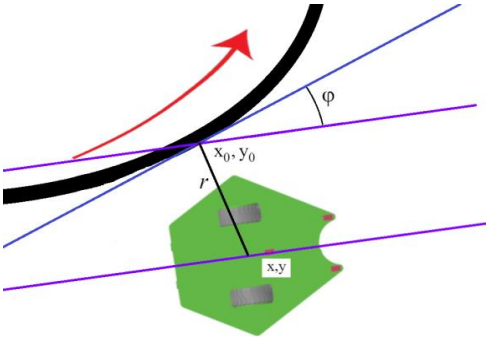

Fig. 2. The direction of movement of the robot, r is the minimum distance to the nearest point on the elliptic curve

B. Implementation of the Environment

The Environment is a DT MR implemented in the MATLAB application package [9], which generates the following states for the Agent:

– the position of the robot;

– indication from the sensor.

C. Agent implementation

The Agent is an CS that studies the CO and receives the following states from the Environment:

– observation, which includes:

– the previous formed action;

– the difference between the current state of the sensor and the desired value;

– the previous generated award;

– the current reward for the performed action;

– stopping criteria.

The Agent interacting with the Environment "implements the following basic functions necessary for the work of RL:

– list of available actions;

– handling status and awards from the Environment;

– obtaining the accumulated experience, presented in the form of a neural network (NN) for the Actor and the Critic" [11].

The list of available actions includes the angular speed from the left and right wheels, where the constant speed is 12 rad/s with a wheel radius of 0.025 cm.

The Agent receives readings from 3 sensors, such as the encoders of the right and left wheels and 1 light contact sensor, which determine the speed of the wheels and the location of the robot, as well as the previous action that the Agent formed.

The Agent receives the current reward when it minimizes the distance between the nearest point on the ellipse and the robot's location, while also minimizing the slope between the tangent to the ellipse and the robot's rail. The reward function for the RL Agent was defined as negative, since the RL Agent maximizes this reward, thereby minimizing the error.

The conditions are the criteria for stopping, formed in such a way as to shorten the training time. If the robot has exceeded the specified value (20 cm), then training starts over.

The learning process was built on the basis of TD3. This TD3 algorithm is the next version of the DDPG (Deep Deterministic Policy Gradient) algorithm, which is more reliable, increases the stability [13] of learning, and "Eliminates function approximation errors in methods of criticizing actors" [14]. The exceptional nature of this algorithm is that it combines 3 main algorithms for RL, such as Double Deep Q–Learning [15], Policy Gradient [16] and Actor–Critic [17].

This algorithm is based on an Agent–Critic and an Agent–Actor, which use a Critic with a Q–value and a deterministic Actor, respectively.

"The advantage of this method is that the TD3 Agent approximates the long–term reward, taking into account the observation and action, using the 2 presented Critics. When constructing neural networks for Actor, radial descent optimization can lead to negative weights. To avoid negative weights, we replaced the normal *fullConnectedLayer()* with *fullConnectedPILayer()*. This layer ensures that the weights are positive" [18]. After training, we extract these weights for our PID controller and apply them to the test model.

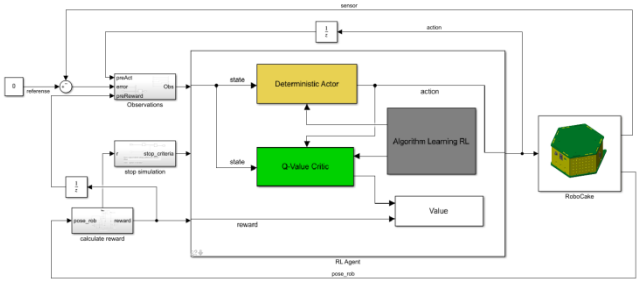

Fig. 3. Architecture of Reinforcement Learning based on DT in the MATLAB modeling Environment

D. Learning Algorithm

At this stage, we are ready to train the Robot, this is a rather long procedure, and in this case the training process took 2.15 hours and as a result, everything that the CS setup specialist had to do to set the correct architecture and calculate the CS parameters, all this did RL.

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

IV. THE RESULTS OBTAINED, THEIR SIGNIFICANCE AND COMPARISON WITH PREVIOUS WORK

In the current research work, a software Agent was developed using the TD3 algorithm for a simulated DT of a 2–wheeled robot moving along an elliptical curve in the MATLAB application package. A simulation Environment was implemented and a reward function was developed. Comparing the results obtained earlier by the GA method [6], with the new results obtained by the RL method, we can conclude that the latter method reduces the number of episodes and training time by several times (Table I).

TABLE I. LEARNING OUTCOMES OF TD3 AND GA METHOD

PID controller	***P***	I	D	***Learning Algorithm***	***Training time (hours)***	***Number of episodes***
1	1.3593	1.4066		TD3	0.45	100
2	3.676	-0.159	-4.665		2.15	500
3	0.0903	0.0085	0.7691		2.90	800
4	-4.627	0.15	0.3011	GA	5.55	2000
5	-1	0	-0.0621		4.15	1500
6	-0.4164	-0.5453	-0.1568		6.94	2500

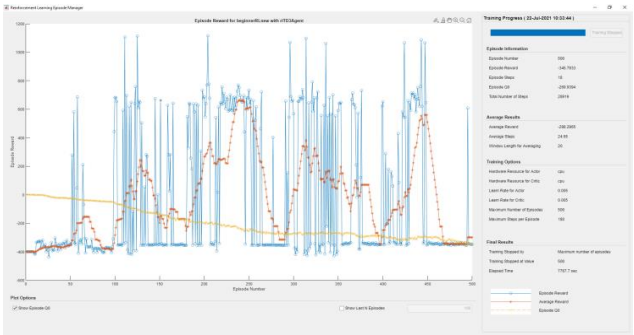

Fig. 4. Result of robot training after 2.15 hours, Episode Reward – the reward received by the robot for each episode, Average Reward – the average reward for window length for Averaging equal to 20, Episode Q0 – a critical assessment of the long–term reward for each episode

This graph shows that the highest reward is 1113, the average is 715, where the robot can drive an elliptical curve in 8 seconds minimizing the distance to the curve.

V. CONCLUSION

In this article, using the RL methods, we have chosen the optimal values of the PID controller that controls the movement of the 2–wheel MR along the color contrast line. For this, a virtual environment for the functioning of the MR was implemented, a software Agent was configured, a reward function was developed, training was implemented using the TD3 method using a deterministic Actor and a Q–value Critic. As a result, the tuned PID controller allows the MR to accurately move along the curved color–contrast line at a speed of 0.66 m /s.

REFERENCES

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, London, England, The MIT Press, 2014, pp. 13.

[2] A. V. Philippov, M. A. Kosolapov, I. A. Maslov, and G. I. Tarasova, “Automated tuning of the PID controller for the control object of the tracking system using the MATLAB Simulink software package”, Science, technology and education, no. 12 (18), 2015, pp. 53–59 (in Russian).

[3] W. S. Levine, “PID Control,” in The Control Handbook, Ed. Piscataway, NJ, IEEE Press, 1996, pp. 198–209.

[4] F. G. Martins, Tuning PID controllers using the ITAE criterion, January 2005 International Journal of Engineering Education 21(5). Intern. J. Engng Ed. Vol. 21, no. 5, pp. 867–873, 2005 0949–149X/91 Printed in Great Britain.

[5] J. G. Ziegler, N. B. Nichols, “Optimum settings for automatic controllers,” Trans. ASME, 1942, vol. 64, pp. 759–768,

[6] T. Yu. Kim and G. A. Prokopovich, “Optimization of the parameters of the PID controller of the educational mobile robot control system using genetic algorithms”, unpublished (in Russian).

[7] Reinforcement Learning Agents (Release 2021a). [Online]. Available:https://www.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html.

[8] T. Yu. Kim, “Development of a digital twin of a mobile robot for research and educational purposes based on MATLAB / Simulink”, XVIII International Conference of Young Scientists "Youth in Science – 2.0'21", submitted for publication (in Russian), in press.

[9] What Is Reinforcement Learning? (Release 2020b), [Online]. Available:https:// www.mathworks.com/help/reinforcement–learning/ug/what–is–reinforcement–learning.html.

[10] Reinforcement Learning Toolbox™ User's Guide 2021a, [Online]. Available: https://www.mathworks.com/help/pdf_doc/reinforcement–learning/rl_ug.pdf.

[11] M. P. Mayorov, “Reinforcement Learning Algorithm for Solving a Robot Motion Problem”, master’s thesis, South Ural State University, National Research University, Chelyabinsk, Russian, 2019. [Online]. Available: https://dspace.susu.ru/xmlui/bitstream/handle/0001.74/29488/2019_222_majorovmp.pdf?sequence=1 (in Russian).

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

[12] “Ellipse”. [Online]. Available: https://ru.wikipedia.org/wiki/Эллипс.

[13] S. Fujimoto, H.–H. Meger, D. Meger, Addressing Function Approximation Error in Actor–Critic Methods, Cornell University, Oct., 2018. [Online]. Available: https://arxiv.org/pdf/1802.09477.pdf.

[14] S. Fujimoto, H. Herke, D. Meger, Addressing Function Approximation Error in Actor–Critic Methods, Cornell University, Oct., 2018. [Online]. Available: https://arxiv.org/abs/1802.09477.

[15] H. Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double Q–learning. In AAAI, pp. 2094–2100, 2016.

[16] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, Deterministic policy gradient algorithms, In ICML, 2014.

[17] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, 2000, pp. 1057–1063.

[18] Tune PI Controller using Reinforcement Learning (Release 2021a), [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/tune-pi-controller-using-td3.html.

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.