MACHINE LEARNING IN THE TASK OF SELF-CALIBRATION OF MOVING ELEMENTS OF ROBOTIC SYSTEMS USING THE EXAMPLE OF CONTROLLING STEPPER MOTORS

Kemenyash Yuriy Mykhailovych
Assistant Professor
Faculty of Air Navigation, Electronics and Telecommunications
National Aviation University, Ukraine

Khotsianivskyi Volodymyr Petrovych
Postgraduate student
Faculty of Air Navigation, Electronics and Telecommunications
National Aviation University, Ukraine

Kuleshov Danylo Oleksandrovich
Student
Aerospace Faculty
National Aviation University, Ukraine

Summary. The paper discusses the use of machine learning for self-calibration of moving elements of robotic systems, particularly for controlling two stepper motors, as they are popular in robotics due to their accuracy and control over the angle of rotation. An algorithm for self-calibration is described, which uses machine learning to improve the accuracy of movement of stepper motors. An example is presented where the accuracy of movement of stepper motors, which were tuned using self-calibration based on machine learning, was compared. The proposed approach demonstrates that using machine learning for self-calibration of moving elements of robotic systems, particularly stepper motors, can improve their accuracy and help ensure more efficient operation of the robotic system.

Key words. Machine learning; microcontrollers; robotic systems; stepper motors; nema; self-calibration;

Introduction. Calibrating moving elements of robots is one of the important tasks in robotics, which involves tuning the parameters of the robot’s moving elements to achieve maximum accuracy and efficiency [1]. Calibrating the moving elements of robots is a crucial stage in their development and operation, as untimely or improper calibration can lead to malfunctions, decreased accuracy and speed of movement, as well as increased risk...
of emergencies occurring[3]. However, traditional methods of calibrating robotic actuators, such as using sensors or manually programming motion parameters, can be difficult to use.

As the use of machine learning methods becomes increasingly popular in robotics, it is advisable to use machine learning to solve this task, since it is one of the key technologies that allows robots and automated systems to interact with the environment and perform complex functions with maximum accuracy and efficiency [2]. For example, deep learning methods can be used for automatic determination of optimal robot motion parameters based on data collected during its operation, as well as for predicting possible deviations in its operation. As a result, this will allow automating and simplifying the process of calibrating the moving elements of robots, ensuring maximum accuracy and speed of operation[4].

Therefore, the subject of research is the application of machine learning methods that will allow for the effective solution of tasks for calibrating the moving elements of robotic systems.

**Problem statement.** Traditionally, calibration is done manually by experts, which requires significant effort and time. However, with the development of machine learning technologies, opportunities have emerged to automate this process and reduce the costs associated with it [5].

Applying machine learning methods for calibrating robot's moving elements has several advantages. They can work with large volumes of data, which allows for more accurate results and reduces the number of errors. In addition, the use of machine learning methods can allow robots to adapt to changes in the environment and ensure maximum efficiency.

One of the possible machine learning methods for calibrating the moving elements of robots is Q-Learning. This method is one type of reinforcement learning, where a robot interacts with the environment and learns to make decisions based on the rewards received. In the context of calibrating robotic actuators, Q-Learning can be used to build a model that determines the optimal parameters of the robot’s motion based on data collected during its operation.

To apply Q-Learning, a model needs to be created that tracks the state of the robot and its actions, as well as determines rewards for each action (see Figure 1). In this case, actions involve changing the parameters of the robot’s motion, while rewards reflect the accuracy of its movement.

![Fig.1. Reinforcement learning, agent and environment](image-url)
Optimal motion parameters based on the data it receives during its operation. This can be done by iteratively choosing actions and updating the model based on the rewards received.

At the beginning of the training process, the model can be initialized with random values for the robot's movement parameters. The robot will move with these parameters and receive rewards for its work. Next, based on the received data, the model will be updated and select new motion parameters. The training cycle will continue until optimal motion parameters are reached, maximizing the rewards for the robot's work.

One of the advantages of Q-Learning is that it can teach a robot to adapt to changes in the environment, as it uses rewards obtained for its performance under different conditions. This allows the robot to continue working effectively even when the environment parameters change.

Fuzzy Q-Learning algorithm:
- Observation of state $x$
- For each activated rule, one action is selected according to the usage strategy.
- A calculation is in progress $a(x)$:

$$a(x) = \frac{\sum_{i=1}^{H} w_i(x) a_i}{\sum_{i=1}^{H} w_i(x)}$$

- The corresponding value of $Q(x, a)$ is being calculated

$$Q(x, a) = \sum_{i=1}^{H} w_i(x) a_i q[x,i]$$

Where $q[x,i]$ is the corresponding q-value of the fired rule $i$ for choosing action $i$ according to the algorithm. Applying $a(x)$ and observing the new rule $x'$.
- Calculation of the reward
- Updating the values of $q$ accordingly to:

$$\Delta q[x,i] = \eta \Delta Q \frac{w_i(x)}{\sum_{i=1}^{H} w_i(x)}$$

$$\Delta Q = R(x, a, x') + \gamma \cdot Q(x', a') - Q(x, a) - Q(x', a^*)$$

Overall, the use of machine learning methods for calibrating the movements of robots is a promising direction in the development of robotics. Using Q-Learning can help ensure maximum accuracy and efficiency of robots in various fields of activity.

Problem solution. The object of the study is a vertical plotter (see Figure 2), which consists of two Nema17 stepper motors, an external camera, and a user
interface developed in Python programming language. The stepper motors are responsible for moving the moving elements of the plotter, while the external camera is used to observe the working process and collect data for further analysis. The user interface provides the ability to interact with the plotter and control its operation.

Fig. 2. Schematic construction of the research stand

To control the operation of the system, an algorithm for searching the moving element object is used, which consists of the following stages:

1. Separation of the moving element image from the background. This stage typically uses background subtraction algorithms or pattern matching to detect the object in the image.

2. Determining the color of the moving object. This stage may use different color models such as RGB, HSV, or HSL to determine the color characteristics of the moving object.

3. Determining the coordinates of the moving object on the image and transmitting this data to the learning system. This stage involves determining the position of the moving object in the image and transmitting this data to the learning system for further analysis and processing.

After determining the coordinates of the moving element on the image and passing this data to the learning system, the algorithm uses Q-Learning to determine the necessary control signals to move the moving element to the desired position.
During the operation of the system, the motors provide movement in four possible directions: clockwise and counterclockwise for each of the motors.

The robot receives a reward or punishment depending on the correctness of the direction of movement towards the goal. Direction towards the goal provides a positive reward, while direction away from the goal provides a negative reward. In addition, a positive reward is given if the robot reaches the boundaries of the working area and places the control object in the goal, while a negative reward is given if the robot does not reach the goal or does not place the control object in the goal (see Fig. 3).

Fig. 3. Visualization of the system operation

Conclusion. So, the method presented in the text has several advantages. One of the main advantages is that the motors can be placed in the robot in any way. Additionally, thanks to the self-tuning algorithm, the system can quickly understand where the boundaries of its working area are, which allows for simpler and more direct control of the robot.

In this way, the application of this technology can improve the efficiency and accuracy of the robot, as well as make it more convenient to use.

References: