Optimal Control Problem Solving Using an Identified Neural Network-based Dynamic Model of a Car-like Robot*

Shmalko E.Yu., Eliseev N.A. and Gromov I.A.

Abstract— **This study focuses on solving the optimal control problem for a wheeled car-like robot by leveraging its neural network-based dynamic model. The neural network model captures nonlinear dynamic effects—such as variations in wheel-surface adhesion and battery charge depletion—enhancing the model's accuracy and realism. However, employing such a model restricts the applicability of classical optimal control methods. To overcome this limitation, the paper introduces a direct numerical approach for solving the optimal control problem under phase constraints for a neural network-represented system. The proposed method approximates control inputs as piecewise polynomial functions, with parameter optimization performed using state-of-the-art evolutionary algorithms. An experimental analysis of the most widely used algorithms demonstrates that hybridizing them improves the efficiency of the solution.**

I. Introduction

Optimal control allows minimizing energy costs, improving the smoothness of movement, minimizing time and increasing the accuracy of task execution by a mobile robot. The optimal control is calculated based on mathematical models of control objects. The model is essentially a simplified version of the object itself, covering only its main essential properties. As a rule, fairly simple, often linear models are selected for calculations, for which the analytical apparatus is well implemented. However, in practice, roboticists are faced with the need to refine the obtained control laws due to the discrepancy between the obtained calculated modeling results and real data.

Thus, on the one hand, simple kinematic models do not provide the necessary accuracy, but detailed dynamic models that accurately describe the behavior of robots are too complex, it is difficult to apply existing methods to them, they are usually redundant, or may be completely unavailable.

This paper discusses the use of a neural network dynamic model of a robot to solve an optimal control problem. Unlike a simple kinematic model that does not take into account the dynamics of the robot's motion, using a neural network as a dynamic model for calculating the control action allows us to take into account various factors that affect the robot's behavior in a real environment, such as the effect of battery charge, changes in the coefficient of adhesion of the wheels to the surface, slippage, and other nonlinear factors.

Unlike traditional analytical models, a neural network dynamic model is able to adapt to changing conditions by learning on data obtained in real operating conditions of the robot. This allows us to take into account nonlinear effects that are difficult to formalize.

The problem of optimal control of a wheeled robot is to find a control function that ensures the achievement of the target state while observing the physical limitations of the system and minimizing a given quality criterion. In this paper, an identified neural network model is used to describe the robot's dynamics, which allows us to take into account nonlinear effects and improve control accuracy in real conditions.

Employing a neural network model of the controlled system makes it challenging to apply the maximum principle [1, 2] for optimal control computation. This difficulty arises because, at each integration step, one must determine control inputs that maximize the Hamiltonian - yet the control variables are not explicitly present in the Hamiltonian itself but instead serve as inputs to the neural network. Furthermore, incorporating various phase constraints introduces additional complexity. A numerical solution based on this approach appears computationally intensive and inefficient.

To address these challenges, this paper advocates for a direct approach using piecewise polynomial control approximation, transforming the problem into a nonlinear programming framework. Considering the non-unimodality and non-convexity of the functional in the problems of optimal control of a mobile robot with complex phase constraints, it is necessary to use modern global optimization algorithms. In the experimental part of the work, this approach is demonstrated using a neural network model of a mobile robot, a study of the most popular evolutionary algorithms of global optimization is conducted and it is shown that their hybridization allows increasing the efficiency of solving the problem.

This scientific research was partially supported by Educational Organizations in 2024-2026 Project under Grant FSFS-2024-0012

E.Yu. Shmalko is with the Department of Robotics Control, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333, Vavilova str. 44/2, Moscow, Russia (e-mail: e.shmalko@gmail.com).

N. A. Eliseev is with the Department of Robotics and Mechatronics, Bauman Moscow State Technical University, 105005, Moscow, 2nd Baumanskaya str., 5, p. 1 (e-mail: eliseevna@student.bmstu.ru).

I. A. Gromov is with the Department of Robotics Control, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333, Vavilova str. 44/2, Moscow, Russia (e-mail: 8357743@gmail.com).

II. Problem Statement

The problem of optimal control of a wheeled robot is to find such a control action that ensures the achievement of the target state while observing the physical limitations of the system and minimizing the specified quality criterion. In this paper, an identified neural network model is used to describe the robot dynamics, which allows taking into account nonlinear effects and increasing control accuracy in real conditions.

The wheeled robot is considered as a dynamic system, the state of which is determined by the vector $\mathbf{x}(t)$. Control is performed using the control vector $\mathbf{u}(t)$, which specifies the linear and angular velocity of the robot. The dynamics of the system is described by the equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_{NN}\big(\mathbf{x}(t), \mathbf{u}(t)\big), \tag{1}$$

where $\mathbf{f}_{NN}$ are functions of the right-hand sides of the system of differential equations of the control object model, described by an artificial neural network trained on data obtained during the actual movement of the robot.

To solve the optimization problem, it is necessary to minimize the generalized cost functional J. We define a functional for which the robot will travel from point A to point B in the minimum time without crossing obstacles and with a minimum coordinate error at the final point B.

$$J = k_{time} \cdot t_{end} + k_{goal} \sqrt{\sum_{i=1}^{3} \big(x_i(t_{end}) - x_j^{end}\big)^2} + k_{traj} \cdot L_m + k_{obstacle} \int_0^{t_{end}} \Big(\sum_{j=1}^{k} \vartheta\big(h_j\big)\Big) dt\,, \tag{2}$$

where

$$t_{end} = \begin{cases} t, if \sqrt{\sum_{i=1}^{3} \big(x_i(t_{end}) - x_j^{end}\big)^2} < \varepsilon \\ t^* - otherwise, \end{cases}$$

x_i are phase variables defining the state of the mobile robot (position coordinates on the plane and heading angle); $\vartheta\big(h_j\big)$ is a Heaviside function taking the value of unity if the robot collides with the j-th obstacle; $h_j(\mathbf{x})$ are geometric equations of the j-th obstacle, where obstacles can be of various shapes, both round, describing various pillars, and rectangular, defining walls and other constraints; ε is a permissible deviation of coordinates from the specified target state at the end point; t^* is maximum permissible time of the robot's travel along the specified route, k_{time} is a weighting coefficient of time; k_{goal} is a weighting coefficient of the distance to the end point; k_{traj} is a weighting coefficient of the trajectory length; $k_{obstacle}$ is a weighting coefficient of collision with obstacles; L_m – trajectory length.

Taking into account the trajectory length in the functional helps the optimization algorithm to find such controls that provide a smooth trajectory:

$$L_m = \sum_{i=1}^{idx_{goal}} M_i \cdot dist_i,$$

where idx_{goal} is the index of the time interval in which the robot reaches the end point; $dist_i$ is the Euclidean distance between adjacent points; M_i is a binary mask that filters distances beyond a given index.

Let us set the initial conditions: $x_i^0 = x_i(0)$, and final values of the robot state coordinates $x_i^{t_{end}} = x_i(t_{end}), i = 1,2,3$. Let us also take into account the constrints on the robot's control and speed of movement in the problem: $0 \le u_1 \le 1$ are constraints on the linear speed control; $-1 \le u_2 \le 1$ are constraints on the angular speed control.

III. Training a Neural Network

In this paper, we used a neural network dynamic model of a mobile robot of the automobile type, shown in Fig. 1. A mixed approach was used to identify the model of the control object, when part of the object model is known on the basis of physical laws, and the other part is determined by a neural network trained on the basis of a specially formed training sample.

Next, we describe the process how we trained such model.

To begin with, not all robots have a corresponding virtual model. Moreover, the presence of such a model does not guarantee an adequate description of the real object. Therefore, a high-quality model of a real object can only be formed based on data obtained directly from the physical device. Unlike a computer model, a real robot has unique characteristics that must be taken into account when collecting data (Slippage on the movement surface; The influence of battery charge level on the robot's movement speed (for example, with the same control signals, the movement speed will vary depending on whether the charge level is low or high); Failures and interference in data acquisition from sensors or in the transmission of control signals).

Our model represents a modified version of TRAXXAS E-Revo v2.0:

Figure 1. The robot used for collecting training data

The ROS environment allows interaction with the robot through control signals. The rover utilizes Ackermann steering, meaning that when turning, the wheels on opposite sides of the robot travel different distances. The rover receives commands for linear and angular velocity, the robot's previous state (its prior linear and angular velocities), and the time interval for executing the control action.

On the output side, we can obtain data from sensors and the navigation system. In our case, the output consists of the robot's angular and linear velocities, as well as its current coordinates. In a two-dimensional environment, we obtain the coordinates x, y, and the heading angle (yaw). The input-output scheme of the neural network is shown in Fig. 2.

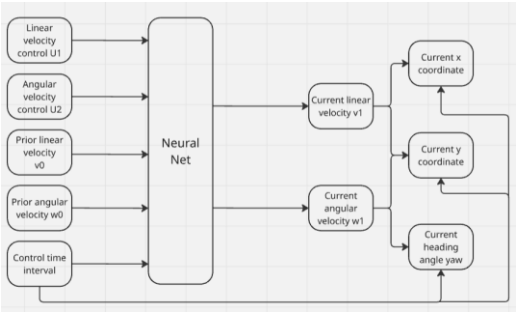

Figure 2. The robot we used for collecting training data

In the general case, it is possible to identify the entire right-hand side of the system of differential equations at once, i.e. to consider the vector function $\mathbf{f}_{NN}$ from (1) as completely unknown and requiring determination using a neural network. However, as the number of dependencies to be identified increases, so does the number of required resources, which include the size and diversity of the training sample and the complexity of the neural network structure. There is a need to simplify the computational load. The work uses a mixed approach to identification [3], which implies partial knowledge of the vector function $\mathbf{f}$, based on kinematic relations. For the numerical implementation of the identification problem, we present the model in finite-difference form with a time step Δt:

$$\begin{cases} x_{k+1} = x_k + \Delta t \cdot v_k \cdot cos(yaw_k) \\ y_{k+1} = y_k + \Delta t \cdot v_k \cdot sin(yaw_k) \\ yaw_{k+1} = yaw_k + \Delta t \cdot w_k \\ v_{k+1} = F^v(v_k, w_k, u_k^v, u_k^w, \Delta t) \\ w_{k+1} = F^w(v_k, w_k, u_k^v, u_k^w, \Delta t) \end{cases} \quad (4)$$

In the problem of mixed neural network identification, it is necessary to determine the functions of change in linear and angular velocities $F^v(v_k, w_k, u_k^v, u_k^w, \Delta t)$ and $F^w(v_k, w_k, u_k^v, u_k^w, \Delta t)$, using a neural network, which express the influence of various factors on the robot's motion, such as friction, inertia, uneven distribution of the robot's mass, and the operation of an unknown low-level controller. The trained neural network function with parameters takes the current state of the robot, the control vector, and the time step as input, and outputs the state of the robot at the next moment in time.

The task of structural identification of the neural network model is not the focus of this work, therefore MultiLayer Perceptron was chosen as the structure of the neural network based on the recommendations presented in [4, 5].

When using neural networks in the task of identifying dynamic models, a number of problems arise that require additional development and research. For example, when solving the problem of object recognition using neural networks, the desired result is known in advance, whereas in the task of obtaining a high-quality model of a control object, it is first necessary to determine which part of the object's dynamics must be taken into account. Thus, the training sample must reflect the properties most important for modeling. Representativeness is the property of the sample to reflect the parameters of the universal population that are significant for the task (in practice, with unknown characteristics of the universal population, this property is ensured by an adequate size and completeness of the sample). The completeness of the sample is determined by the class provision of samples. Homogeneity of the sample shows how evenly the sample samples are distributed across classes.

Unlike a computer environment, the real-world one introduces additional challenges in data collection and processing.

First, low data quality can significantly reduce the effectiveness of model training. Noisy, non-representative, or partially missing data often cause errors, decreasing model accuracy. It is crucial to ensure the representativeness of the collected information to cover various scenarios that the robot may encounter in the real world [6, 7].

Moreover, data quality issues can lead to difficulties in understanding the general movement patterns of the robot. If the model is trained on a limited dataset, its predictions may be accurate only under specific conditions but fail to generalize to others. For example, if a neural network is trained only on data where the robot moves straight, it should not be expected to perform correctly in turning scenarios. [8]

The environment significantly affects the data obtained from the robot's cameras, posing a serious challenge. For instance, varying lighting conditions can lead to errors in object recognition or depth measurement using RGB-D cameras. Complex textures and dynamic objects add additional noise, requiring sophisticated filtering and processing methods to keep the data usable. [9, 10]

Even though the selected camera has built-in filtering for such outliers, they still occasionally appear in the dataset and must be detected and processed to improve data quality. Experimental proof of the impact of outliers on training quality will be provided in the experimental section of this study.

Collecting data from the same starting point of the trajectory is a critically important aspect. This helps minimize deviations related to the robot's initial position, which is particularly important when using methods sensitive to the system's initial state. Consistent initial conditions standardize the data collection process, simplifying subsequent analysis and training. However,

this does not affect data bias, as our model is trained on velocities as the robot's state parameters rather than its coordinates. This approach allows us to both minimize camera initialization errors and select a convenient starting point from which the robot can always begin.

Furthermore, fixed starting points ensure comparability of data collected in different sessions. This helps identify systematic errors or deviations and simplifies the task of constructing a comprehensive picture of the robot's behavior in various environmental conditions.

Data normalization is one of the key steps in data preprocessing. It eliminates differences in scale and dynamic range between different types of sensor data, such as readings from cameras, LiDARs, and inertial sensors. Unnormalized data can complicate the training process, making the model more prone to overfitting or misinterpreting signals. Normalization also helps accelerate the convergence of training algorithms. Data brought to a unified scale enables models to find optimal parameters more efficiently, reducing both time and computational costs. This process is particularly important for mobile robots operating under limited computational resources.

The collected data must possess the following properties to be used for building an identified model:

- Data should be obtained under various robot motion scenarios (moving straight, turning right, turning left).
- Data should be collected under different lighting conditions.

Additionally, the quality of the data can be indirectly assessed by the model's prediction accuracy—high accuracy with minimal trajectory deviation from the ground truth should be observed in the validation dataset.

In the collected datasets, the robot moves straight, right, and left. The natural light source remains stationary, resulting in varying illumination throughout the robot's trajectory.

As part of this study, only straight, right, and left movements were considered. Therefore, reverse motion was neither accounted for in data collection nor in model training.

To evaluate the model's performance and trajectory prediction accuracy, we use the Absolute Translation Error (ATE) metric as it punishes the model more for predicting too low or too high values:

$$ATE = \frac{1}{N}\sum_{t=0}^{N}\sqrt{(x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2},$$

where: N – number of trajectory points; $(x_t,\ y_t)$ – actual trajectory coordinates; $(\hat{x}_t,\ \hat{y}_t)$ – predicted trajectory coordinates

For yaw angle prediction accuracy, we use the Mean Absolute Error (MAE) metric as it is less sensitive to outliers:

$$MAE = \frac{1}{N}\sum_{t=0}^{N}|yaw_t - \hat{yaw}_t|$$

Since our task involves time series forecasting, we apply a multi-step loss function to prevent the model from predicting positions close to the trajectory center. Mathematically, during training, we minimize the deviation of predicted states $\{\mathrm{x_k}\}_{k=n}^{n+N}$ from the true states $\{\hat{\mathrm{x}}_\mathrm{k}\}_{k=n}^{n+N}$ over the trajectory segment $k \in [n, n+N]$:

$$Loss = \sum_{k=n}^{n+N}(\mathrm{x_k} - \hat{\mathrm{x}}_\mathrm{k})^2$$

The starting segment n is chosen randomly

The parameter N is tunable

Experiments shown [11] that trajectory prediction improves when N > 1, because data noise and limited training samples cause overfitting; multi-step loss significantly reduces gradient variance when using gradient descent optimization.

We conducted a grid-search for hyperparameters tuning. As a result, our best model was trained using these parameters:

- Learning rate 0.001;
- Optimization algorithm – Adam;
- Number of hidden layers – 4;
- Number of training epochs – 50;
- Miltistep parameter N – 300;
- Number of neurons in each hidden layer – 64;
- Activation function – ReLU;
- Batch size – 1000;

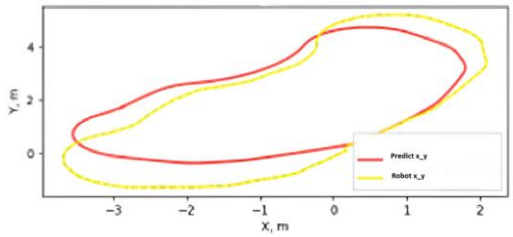

Figure 3. Validation trajectory result using the best model we trained

IV. Direct Approach to Optimal Control

Since an analytical solution to the optimal control problem using a neural network model is not feasible, we employ a direct approach based on approximating the control input as follows:

$$u_i = \sum_{j=1}^{t_{end}} u_{ij} \cdot \Delta t,$$

where Δt is the time interval of the j-th control signal u_{ij}.

The parameters u_{ij}will be optimized using an optimization algorithm, the choice of which will be described further.

The simplest methods for finding function minima are gradient-based methods. These methods rely on computing gradients and searching for the global minimum of a function in the direction of the negative gradient.

Standard gradient descent methods, such as BGD (Batch Gradient Descent), SGD (Stochastic Gradient Descent), and Mini-batch Gradient Descent, exhibit slow convergence [12]. To accelerate convergence, momentum-based and adaptive approaches have been developed. These methods dynamically adjust learning rates but are more sensitive to initial conditions and can be challenging to balance between momentum and adaptive components. Second-order methods, such as Newton's method and quasi-Newton methods, offer faster convergence but are computationally more complex than the previously mentioned approaches.

Since most applied optimal control problems with phase constraints in robotics have non-unimodal functionality and the space of possible solutions is non-convex, modern global optimization algorithms based on population and evolutionary search are becoming the most popular among direct methods for solving such problems. Based on the results of the studies presented in [13], the swarm algorithm and the genetic algorithm show the greatest efficiency for the optimal control problem. In this regard, they were used in the experimental part of this work.

The genetic algorithm (GA) is an evolutionary optimization method inspired by natural selection and genetics. It is effective for global optimization and does not require gradient information. However, it can be computationally expensive and may require significant resources and time to converge, especially for high-dimensional problems [14].

The particle swarm optimization (PSO) method is an evolutionary algorithm inspired by the collective behavior of bird flocks or fish schools. It employs a population of particles that update their positions in the search space based on both individual and collective experience [15].

In PSO, a swarm of particles is used, where each particle represents a potential solution to the optimization problem. Let N be the number of particles in the swarm, then the position of the i-th particle in the solution space is represented as a vector:

$$x_i = (x_i^1, x_i^2, \ldots, x_i^D),$$

where D - is the dimensionality of the search space.

The velocity of a particle is denoted asv_i. Each particle stores its best-found position (personal best)p_i. The best position found by the entire swarm (global best) is denoted as g.

At each iteration, the velocity and position of a particle are updated using the following formulas:

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (p_i - x_i) + c_2 r_2 (g - x_i)$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

where ω is the inertia coefficient regulating the influence of the previous velocity; c_1, c_2 are learning coefficients (typically $c_1, c_2 \approx 1.5 - 2$; r_1, r_2 are random numbers from the range $[0,1]$; p_i is the best position of the given particle along the d-th coordinate; g is the best position among all particles along the d-th coordinate.

The algorithm runs until one of the following conditions is met:

1. The maximum number of iterations is reached.
2. The desired solution accuracy is achieved.
3. The particle velocities become very small, indicating solution stabilization.

The output of the algorithm is the optimal value g, corresponding to the minimum (or maximum) of the objective function.

Thus, PSO does not require gradient computation and can efficiently find global optima in complex and nonlinear spaces. Furthermore, PSO typically converges faster than genetic algorithms and is simpler to implement [16].

Based on conducted research [13], classical gradient-based methods cannot be effectively used for solving optimal control problems where the properties of the objective function do not meet the algorithm's requirements. According to a comparative study of evolutionary algorithms [16], the Particle Swarm Optimization (PSO) method is more suitable for this task, as it does not require gradient and has moderate computational complexity. This makes it the preferred choice for solving optimal control problems of wheeled robots in our experiments using neural network dynamic models.

V. Computational Experiments

In our experiments, we decided to use the previously obtained neural network-based dynamic model of our rover-like robot.

Robot is moving from point A $[0.0, 0.0, 1.5]$ to the goal $[-2.0, 3.0, 0.0]$ in the environment with following obstacles:

$(-0.5, -0.4, 3, -0.1), (-1, -2, -0.4, 4), (0.5, 0, 1.2, 10),$
$(-2, 2.5, -1, 3), (-6, 2.5, -3, 3), (-4, 0, 1.7),$
$(-6, -2.5, -3, -2), (-2, -2.5, 1, -2), (-9, 3, -5.5, 6),$
$(-6, 5, 1, 10), (-10, -2, -6, 2.5)$

We use following weight coefficients: $k_{time} = 0$; $k_{goal} = 1$; $k_{traj} = 1$; $k_{obstacle-rectangular} = 1000$; $k_{obstacle-circular} = 100$. We defined a collision with rectangular obstacles to be more punishable because these are actual walls while circular obstacle is the area, we want robot to avoid.

We conducted series of experiments with various parameters and gathered the most relevant ones:

TABLE I. Experiments results

Algorithm	*Paramteres*	*J value*	*Computational time*
PSO	Num iterations = 400; Num particles = 15000; $t^+ = 35\ sec$; $t_{alg} = 1\ sec$; $dt = 0.03\ sec$;	22.989 ($k_{time} = 1$)	6108,8 sec
PSO	Num iterations = 700; Num particles = 15000; $t^+ = 30\ sec$; $t_{alg} = 0.15\ sec$; $dt = 0.03\ sec$;	13.497 ($k_{time} = 0$)	9419,9 sec

Algorithm	*Paramteres*	*J value*	*Computational time*
PSO + GA	Num iterations (PSO) = 300; Num particles (PSO)= 8000; $t^+ = 30\ sec$; $t_{alg} = 0.2\ sec$; $dt = 0.03\ sec$; Num iterations (GA)=2000; Num particles (GA)=200;	12.592 ($k_{time} = 0$)	2153,1 sec (PSO) +655.3 sec (GA)

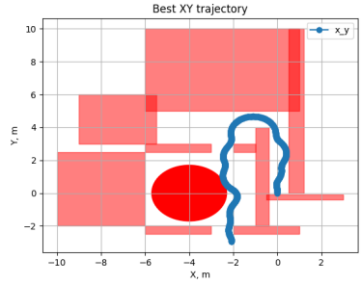

Figure 4. PSO algorithm with $k_{time} = 1$ experiment result.

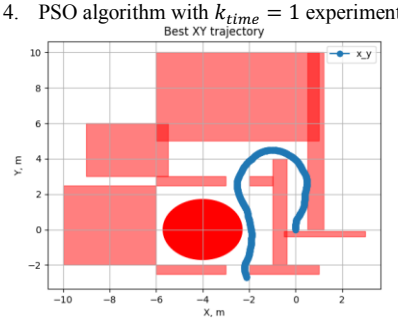

Figure 5. PSO algorithm with $k_{time} = 0$ experiment result

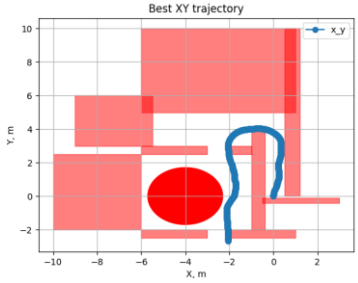

Figure 6. PSO+GA algorithms experiment result.

PSO algorithm has proved to be an effective for optimal control task. According to the Figure 4 this solution can be enhanced. Setting k_{time} to 0 we managed to get a smoother trajectory. However, this path is too far away from obstacles. We implemented a combination of PSO and GA to tackle this problem. Doing so we not only found a solution with lower J value and a smother trajectory, but also it allowed us to limit PSO iterations and particles which resulted in almost 4 times computational time decrease.

VI. Conclusion

The research introduces a novel optimal control methodology based on high-accuracy modeling. The precision model was developed using state-of-the-art machine learning approaches, where an artificial neural network was trained to represent the controlled system. The resulting trained neural network serves as the foundation for solving the robot's spatial motion optimal control problem with phase constraints. As a result, we have conducted the research on solving optimal control problem specifically with the usage of neural network-based dynamic model of wheeled robot. We also selected a combination of PSO method and GA as our final solution which has shown both the increase in performance and decrease in computational time. Experimental results demonstrate that such accurate modeling enables even open-loop control systems to achieve satisfactory motion quality.

References

[1] L.C. Young, *Lectures on the Calculus of Variations and Optimal Control Theory*. Chelsea Pub. Co., New York, 2nd edition, 1980.

[2] D. Karamzin, F.L. Pereira, On a Few Questions Regarding the Study of State-Constrained Problems in Optimal Control. *J Optim Theory Appl*, 2019, 180, pp.235–255.

[3] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, vol. 1, issue 1, 1990, pp. 4-27.

[4] A.A. Dyda, D.A. Oskin, A.V. Artemiev Robot dynamics identification via neural network // *IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Warsaw, Poland, 2015, pp. 918-923

[5] Xu Jian'an Zhang Mingjun, Zhang Jian. Kinematic model identification of autonomous mobile robot using dynamical recurrent neural networks. *IEEE International Conference Mechatronics and Automation*, 2005, Niagara Falls, ON, Canada.

[6] J. Wu, J. Gao, J. Yi, P. Liu and C. Xu, "Environment Perception Technology for Intelligent Robots in Complex Environments: A Review," *2022 7th International Conference on Communication, Image and Signal Processing (CCISP)*, Chengdu, China, 2022, pp. 479-485.

[7] C. Jing, J. Potgieter, F. Noble and R. Wang, "A comparison and analysis of RGB-D cameras' depth performance for robotics application," *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, Auckland, New Zealand, 2017.

[8] M. Haroush, T. Frostig, R. Heller, D. Soudry, A statistical framework for efficient out of distribution detection in deep neural networks, ICLR 2022, Mar 2022.

[9] D. Maier, A. Hornung and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, Osaka, Japan, 2012, pp. 692-697

[10] S. Dasari, F. Ebert, S. Tian, S. Nair "RoboNet: Large-Scale Multi-Robot Learning", Conference on Robot Learning (CoRL) 2019, Osaka, Japan, pp. 293-208.

[11] E.Y. Shmalko, Y. A. Rumyantsevm R.R. Bainazarov, K.L. Yashmanov, Identification of Neural Network Model of Robot to Solver the optimal Control Problem, "Computer Technology and Automatization", Moscow, Russia, 2021, vol. 20, ch. 6, pp 1254-1278.

[12] O. F. Razzouki, A. Charroud, Z. E. Allali, A. Chetouani and N. Aslimani, "A Survey of Advanced Gradient Methods in Machine Learning," *2024 7th International Conference on Advanced Communication Technologies and Networking (CommNet)*, Rabat, Morocco, 2024.

[13] Diveev, A.I., Konstantinov S.V. Study of the Practical Convergence of Evolutionary Algorithms for the Optimal Program Control of a Wheeled Robot. *Journal of Computer and Systems Sciences International*, 2018, Vol. 57, No. 4.

[14] David E. Goldberg. 1989. Genetic Algorithms in Search, Optimization and Machine Learning (1st. ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[15] Gad, A.G. Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. Arch Computat Methods Eng, 2022, 29, 2531–2561.

[16] A.A. Zaytsev, V.V. Kureychik, A.A. Polupanov, A review of evolutionary optimization methods based on the swarm intelligence, SFU, Rostov-on-Don, Russia, 2010, pp 7-12.