Self-Collision Avoidance Control of Dual-Arm Multi-Link Robot Using Neural Network Approach

Vadim Kramar [1], Oleg Kramar [2], Aleksey Kabanov [3*]
Department of Informatics and Control in Technical Systems, Sevastopol State University, Sevastopol, Russia
Email: [1] kramarv@mail.ru, [2] rolets@yandex.ru, [3] kabanovaleksey@gmail.com
*Corresponding Author
The research was supported by the Russian Science Foundation grant No. 22-19-00392, https://rscf.ru/project/22-19-00392/

***Abstract*—The problem of mutual collisions of manipulators of a dual-arm multi-link robot (so-called self-collisions) arises during the performance of a cooperative technological operation. Self-collisions can lead to non-fulfillment of the technological operation or even to the failure of the manipulators. In this regard, it is necessary to develop a method for online detection and avoidance of self-collisions of manipulators. The article presents a method for detecting and avoiding self-collisions of multi-link manipulators using an artificial neural network by the example of the dual-arm robot SAR-401. A comparative analysis is carried out and the architecture of an artificial neural network for self-collisions avoidance control of dual-arm robot manipulators is proposed. The novelty of the proposed approach lies in the fact that it is an alternative to the generally accepted methods of detecting self-collisions based on the numerical solution of inverse kinematics problems for manipulators in the form of nonlinear optimization problems. Experimental results performed based on MATLAB model, the simulator of the robot SAR-401 and on the real robot itself confirmed the applicability and effectiveness of the proposed approach. It is shown that the detection of possible self-collisions using the proposed method based on an artificial neural network is performed approximately 10 times faster than approaches based on the numerical solution of the inverse kinematics problem while maintaining the specified accuracy.**

Keywords—multilink manipulator; dual-arm robot; neural network; self-collision avoidance

I. INTRODUCTION

The multilink manipulators of modern robots allow solving high-precision technological problems. As a rule, these manipulators are redundant, have a degree of freedom (DOF) of at least six, which allows them to move the end-effector of the manipulator in any position within their workspace.

The synthesis of the motion control systems for such manipulators can be carried out based on various approaches. The technique used, as well as the way it is implemented, can have a significant impact on the performance of the manipulator and on the possible range of applications. On the other hand, the mechanical design of the manipulator affects the type of control circuit used. For example, the control problem of a Cartesian manipulator differs significantly from the control problem of a multilink manipulator.

Regardless of the specific type of mechanical manipulator, it should be noted that the target setting (movement and efforts of the end-effector) is usually carried out in the operational space, and the control actions (generalized forces of the actuators) – in the joint space. This fact leads to the consideration of two types of general automatic control schemes: a control scheme in the joint space and a control scheme in the operational space [1].

Some robots with multilink manipulators make it possible to implement a control design based on the "Programming by Demonstration" approach [1 – 4]. The "Programming by Demonstration" approach is based on the robot teaching to perform the various operations based on the results of the operator's (teacher's) actions. In this case, the operator should be able to "demonstrate" either by directly moving the joints of the manipulator, or with the help of special master devices [5]. As a rule, such master devices have a kinematic similarity with an actuating mechanism of the manipulator. In this case, the operator using the master device makes the required movements of the manipulator. The presence of a master device makes it possible to organize a copy-type control of the manipulators, forcing the robot to copy the movements of the operator in real time, for example, when the robot is operating in an unstructured and dangerous environment.

Self-collision detection is a basic feature that plays an important role in performing fully or partially autonomous manipulations in an unstructured environment. In practice, it usually turns out that the seemingly large working space of the manipulator is significantly reduced due to the danger of collision with other objects, the presence of joint limits, special positions and zones of limited controllability [2, 6].

The solution of the problem of detection and preventing the critical movements of manipulator becomes much more complicated if the robot has several manipulators with possible mutual influence on each other. In particular, for robots with several manipulators, there is a danger that manipulators will collide with each other. Such collisions are called self-collisions [7-11]. For robots with a copy-type control, the self-collisions control problem falls on the operator. There may be cases when self-collisions of manipulators will be possible (for example, due to poor calibration of the master device, or as a result of erroneous actions of the operator, or, for example, as a result of a mismatch between anthropomorphic characteristics of the operator and the master device).

Often in the literature, methods of analysis of the critical positions of manipulators that lead to collisions are reduced to the analysis of the kinematic model of the manipulators

and the search of the coordinates and their combinations at which these critical positions arise [7–24]. In [7], the authors presented a solution of the problem of two manipulators motion control using a constraint-based programming approach, which was used to generate online motion plans. The control problem is reduced to the analysis of the interaction of only the end-effectors of the manipulators, ensuring the avoidance of their self-collisions and collisions with external obstacles. Obviously, this approach does not guarantee the prevention of collisions between manipulators links. Another approach is based on the search for restrictions on control signals that exclude the movement of manipulators with the intersection of the trajectories of all the parts of the manipulators. In [8, 9, 25], the authors proposed a simple approach based on linear algebra methods for self-collisions control of multilink manipulators. In [14], to detect the manipulators collisions and represent them in a two-dimensional coordinate system, the so-called collision map was proposed as a collision zone. In this case, collision avoidance is achieved by temporarily planning the trajectory of the movement of the robot that received the move instruction. The manipulator, which becomes an obstacle in the path of another one receives an escape instruction. To generate real-time movements of manipulators to avoid collision, a reliable reactive algorithm called the "skeleton algorithm" is proposed [15]. According to this algorithm, the structure of the robot is divided into segments, so that arbitrary points can be selected and controlled. The algorithm determines the collision points on such segments and generates the appropriate control for collision avoidance. This algorithm requires sensory information about the position of each link while moving in order to plan avoidance trajectories before possible collisions. An occupied-free approach can be seen in [16] (with the provision that the approach is applicable for a two-dimensional space). In [19] the authors developed a backward quadratic search algorithm as an option for solving inverse kinematics problems in obstacle avoidance. The algorithm executes a backward search for possible obstacle collisions, from the end-effector to the base, and avoids obstacles by utilizing a hybrid inverse kinematics scheme. In [20] a non-collision trajectory-planning algorithm in three-dimensional space was presented. The algorithm is based on velocity potential field for robotic manipulators, which can be applied to collision avoidance among serial industrial robots and obstacles, and path optimization in multi-robot collaborative operation. In [23] the approach of collision prediction of a dual-arm robot based on a 3D point cloud is proposed. According to the approach, a simplified 3D model of the robot is generated using a segmented point cloud, and a self-identification control is based on the over-segmentation approach and the forward kinematic model of the robot. In [24] a self-collision between a manipulator and a mobile robot was considered. Authors introduced the concept of a distance buffer border, which is a 3D curved surface enclosing a buffer region of the mobile robot. When the distance between the manipulator and the mobile robot is less than the buffer distance, the proposed strategy is to move the mobile robot away from the manipulator.

All the above methods are united by the fact that the solution of the kinematic problems necessary for their implementation is performed based on classical approaches. At the same time, a number of articles [26-32] propose approaches to design kinematic control of multilink manipulators based on neural networks, where the data obtained because of solving the direct kinematics problem (DKP) are used as a set of training parameters. At the same time, there are studies that show the effectiveness of the neural network approach, primarily in terms of computational speed. It is especially important for real-time self-collisions control for robots with copy-type master devices. In this regard, it is very useful to consider the problem of the self-collision control of robot manipulators using a neural network approach without solving the inverse kinematics problem and analytical approaches. This, according to the authors, should lead to a reduction in the time to solve the problem of self-collision analysis, which will allow a more efficient online control implementation.

This paper considers the solution of self-collisions control problem for multi-link robot manipulators with a copy-type master device based on a neural network approach. As an object, the robot SAR-401 with two multilink (anthropomorphic) manipulators controlled through a master device in the form of a copy suit is considered. The proposed design is based on the use of the special choice of the structure of the input dataset and experimental choice of the structure of the neural network. The approach proposed in the article made it possible to reduce the computation time by approximately 10 times compared to the application of the computational approach to the analysis of the self-collision [8, 9] while maintaining the specified accuracy. By reducing the computation time, the approach proposed in the article becomes an effective tool for developing a control program with an online mode for detecting and avoiding self-collisions of multi-link manipulators. The proposed approach can be extended to other anthropomorphic robots. Note that the approach proposed in the article is based on solving a regression problem, the use of which for detecting and avoiding a self-collision, in our opinion, is more effective than the classification approach, which gives a prediction with a certain probability of less than 100%, which as a result may lead to an incorrect result.

The rest of the article is structured as follows. Section 2 presents the problem statement and the description of the robot SAR-401. The kinematic analysis of the robot SAR-401 manipulators is shown in section 3. The problems of the neural network design, including training dataset, architecture are discussed in section 3. Section 4 is devoted to the neural network learning process. Section 5 presents the results of the operation of the self-collision control system.

II. THE DESCRIPTION OF THE ROBOT SAR-401 AND PROBLEM STATEMENT

The robot SAR-401 is a torso manipulation robot with anthropomorphic structure and consisting of a base, a body mounted on it. Two manipulators with five-finger grippers and a head module with a computer stereo vision system are mounted on the robot's body (Fig. 1) [33, 34]. Fig. 1 shows: 1 – head module; 2 – compartment for stereo video cameras; 3 - rotary mechanism of the head module; 4 - body; 5,6,7,8 - rotational kinematic pairs of the manipulator; 9 - five-

fingered gripper; 10 - rack of SAR-401. Each manipulator of the robot SAR-401 is an anthropomorphic.

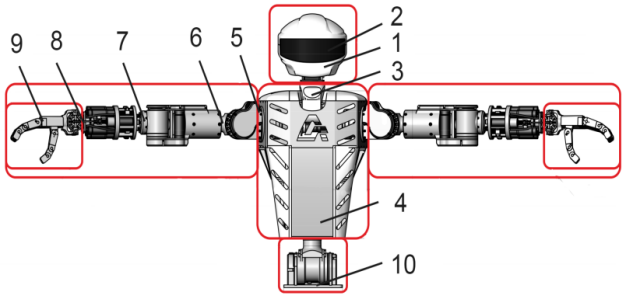
The parameters of the anthropomorphic manipulator of the robot SAR-401 are shown in Table 1.

TABLE I. THE PARAMETERS OF THE ROBOT SAR - 401

Parameters	Manipulator of the SAR-401
Weight	70.0 kg (with hand 1.2 kg)
Height	825 mm
Articulation types of the manipulator	revolving
Robot manipulator length	shoulder to elbow (290 mm) elbow to wrist (280 мм)
Joint Motor Swing Speed Swing speed of the gripper motors	170 grad/s 110 grad/s
Joint Positioning Accuracy	12 bit (4 096 pulses per rotation)

The master device, for example, a copy suit (Fig. 1(a)), makes it possible to organize a copy-type control of the manipulators, forcing the robot to copy the movements of the operator in real time. In this case, it is necessary to control the self-collisions of the manipulators.

a)



b)

Fig. 1. Robot SAR-401: a) robot SAR-401 and an operator in a master device; b) structure of robot SAR-401

The problem of the self-collision control in the copy-type control mode of the robot SAR-401 is to develop a method for determining the configurations of both manipulators, which can lead to self-collisions with a high probability. To determine the current configuration, signals from the motors of the robot's manipulators are used. With the help of these signals, the values of the generalized coordinates of the manipulators are formed, knowing which the coordinate values of all the manipulators links are calculated through the direct kinematics problem solution.

Based on the information about the planned movement of the manipulator links received in real time from the master device, the possibility of a self-collision during movement is analyzed based on the calculation of the minimum distance between the manipulator links. The results obtained are transmitted to the control system, and it blocks this movement and provides information to the operator about a possible self-collision if the minimum distance between the manipulators is less than a certain threshold value. At the same time, the analysis of the relationship between configurations with a minimum distance between manipulators in this paper is proposed to be performed on the basis of a neural network. The training data set for neural network can be obtained by solving the direct kinematics problem.

III. KINEMATICS ANALYSIS

To construct a kinematic diagram of the manipulators of the robot SAR-401, we use the Denavit-Hartenberg notation and methodology [35, 36]. Consider the left manipulator.

To represent the geometry of the manipulator, we will use a kinematic diagram (Fig. 2), which sequentially displays the representation of the links of the manipulator connected by joints. Since the manipulator has only a rotational type of joints, the relative position of adjacent links will be determined by the angular variable θ_i. The Denavit-Hartenberg parameters for the left manipulator of the robot SAR-401 are shown in Table II.

TABLE II. THE DENAVIT-HARTENBERG PARAMETERS FOR THE LEFT MANIPULATOR OF THE ROBOT SAR - 401

Link, i	A_i, m	α_i, rad	d_i, m	θ_i, rad
1	0	π/2	0	θ_i
2	0	π/2	0	$\theta_i - \pi/2$
3	0	π/2	-0.3	$\theta_i - \pi/2$
4	0	π/2	0	$\theta_i + \pi$
5	0	π/2	-0.3	$\theta_i + \pi$
6	-0.24	0	0	$\theta_i + \pi/2$

In Table 2, the following designations are adopted: A_i is the distance between the Z_{i-1} and Z_i axes along the X_i axis, α_i is the angle from the Z_{i-1} axis to the Z_i axis about the X_i axis, d_i is the distance from the origin of the frame $i-1$ to the X_i axis along the Z_{i-1} axis, θ_i are generalized coordinates (the angles of rotation of joints) (Fig. 2). The parameters $\theta_i, i = \overline{1,6}$ are variables and are initially 0.

For the right manipulator, the Denavit-Hartenberg parameters are identical. Restrictions on the movements of the joint servos of the SAR-401 manipulators (servo limits) are shown in Table III.

TABLE III. THE LIMITS OF SERVOS OF MANIPULATORS

№	Min	Max	Min	max
0	-90°	0°	-90°	0°
1	0°	105°	-105°	0°
2	-40°	40°	-40°	40°
3	-110°	5°	-110°	5°
4	-40°	40°	-40°	40°
5	-10°	10°	-10°	10°

When solving the direct kinematics problem two coordinate systems are considered: the initial one, associated with the base joint x_0, y_0, z_0, and the final one, associated with the end-effector of the manipulator x_n, y_n, z_n. Extended position vectors $\hat{X}_0 = [x_0 \quad y_0 \quad z_0 \quad 1]^T$ and $\hat{X}_n = [x_n \quad y_n \quad z_n \quad 1]^T$, for the specified coordinate systems are related by the formula

$$\hat{X}_0 = T_n^0 \hat{X}_n \tag{1}$$

where $T_n^0 = \left[\begin{array}{c:c} R_{3x3} & p_{3x1} \\ \hdashline f_{1x3} & t \end{array}\right]$ is a 4x4 matrix of a homogeneous transformation that carries information about the translation (vector p_{3x1}) and the rotation of one coordinate system relative to another (matrix R_{3x3}), f_{1x3} is a 1x3 vector that specifies the perspective transformation, t is a global scaling factor. In our case $f_{1x3} = [0\ 0\ 0]$, $t = 1$.

Now for each joint, we can define a homogeneous transformation matrix T_i that specifies the sequence of transformations according to the Denavit-Hartenberg notation: rotation around the axis X by an angle α_i, shift along the axis X by A_i, rotate along the axis Z by an angle θ_i, shift along the axis Z by d_i

$$\begin{aligned} T_i &= T_{z\,d_i} T_{z\,\theta_i} T_{x\,A_i} T_{x\,\alpha_i} = \\ &= \begin{bmatrix} I & p_{d_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{z\,\theta_i} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & p_{A_i} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{x\,\alpha_i} & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{2}$$

The calculation of the homogeneous transformation matrix (depending on the angles $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$) is carried out using the relation

$$T_n^0 = T_1 T_2 T_3 T_4 T_5 T_6. \tag{3}$$

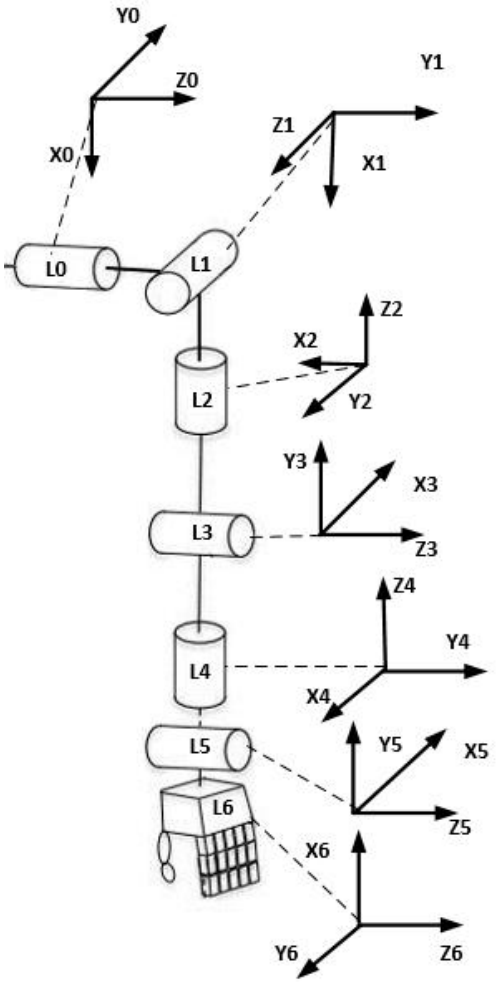


Fig. 2. The kinematic diagram of the left manipulator of the robot SAR-401

As a result, we get the transformation matrix

$$T_n^0 = \begin{pmatrix} n_x & o_x & a_x & x \\ n_y & o_y & a_y & y \\ n_z & o_z & a_z & z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4}$$

To form a dataset for training the neural network, we need to know the coordinates of each joint. It is necessary to compose 6 transformation matrices for each joint of the manipulator, which are used to generate data for training the neural network

$$\begin{gathered} T_1^0 = T_1,\ T_2^0 = T_1 T_2,\ T_3^0 = T_1 T_2 T_3, \\ T_4^0 = T_1 T_2 T_3 T_4,\ T_5^0 = T_1 T_2 T_3 T_4 T_5, \\ T_n^0 = T_1 T_2 T_3 T_4 T_5 T_6. \end{gathered} \tag{5}$$

IV. PROPOSED ARTIFICIAL NEURAL NETWORK DESIGN

A. The Training Dataset

The output data for the training set. The output data of the neural network will be the minimum distance between the

robot manipulators. All the distances between the links of the manipulators are calculated, and the minimum distance for each configuration in the robot workspace for the corresponding input data is used as an output value.

The input data for the training set. To collect an input dataset, it is necessary to generate a set of configurations of robot manipulators in a random order. Data collection using the copy-type master device consists of the random positioning of the manipulators by the operator and receiving the coordinates of all the joints of the manipulators from the robot, as well as coordinates x, y, z and Euler angles φ, θ, γ of the end-effectors [37]. Similarly, the collection of the input dataset using the solution of the direct kinematics problem consists in choosing randomly the positions of the manipulators and finding the coordinates x, y, z and Euler angles φ, θ, γ of the end-effectors. The position of the manipulators changes until a sufficient amount of data necessary for training the neural network is accumulated.

Data (random values of the possible positions of the robot servo drives in different places of its working space) will be collected for two manipulators and stored in vectors

$$X_1 = [q_1^1, q_2^1\ q_3^1\ q_4^1\ q_5^1\ q_6^1],$$
$$X_2 = [q_1^2, q_2^2\ q_3^2\ q_4^2\ q_5^2\ q_6^2], \quad (6)$$

where q_i^j is the angle of rotation of the servo of the i-th joint, j-th manipulator, X_1, X_2 are the vectors of positions of the servo drive of the left and right manipulator, respectively. As an input dataset we will use the values of the servos rotation angles for each joint of two manipulators: a vector $X_{12} = [X_1\ X_2]$.

We will call the neural network with the specified input dataset as Network 1.

As an alternative, we will also create a second input dataset (Network 2), in which we will feed the elements of the transformation matrix to the input of the neural network - nine elements of the rotation matrix and three values of the position vector for each manipulator:

$$R_{3x3} = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix},\ p_{3X1} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \quad (7)$$

In favor of such a choice of the input dataset for training the neural network is the effectiveness that is obtained when using such a neural network to solve the inverse kinematics problem (some results on this issue can be found in [12]).

As the second input dataset, it is proposed to supply a vector X_i, consisting of 24 elements of two transformation matrices for the left and right manipulators:

$$X_i = [Ln_x\ Ln_y\ Ln_z\ Lo_x\ Lo_y\ Lo_z\ \mathrm{K}\ La_x\ La_y\ La_z Lx\ Ly\ Lz\ Rn_x\ \mathrm{K}\ Rn_y\ Rn_z\ Ro_x\ Ro_y\ Ro_z\ Ra_x\ \mathrm{K}\ Ra_y\ Ra_z\ Rx\ Ry\ Rz]. \quad (8)$$

Here $TL_n^0 = \begin{pmatrix} Ln_x & Lo_x & La_x & Lx \\ Ln_y & Lo_y & La_y & Ly \\ Ln_z & Lo_z & La_z & Lz \\ 0 & 0 & 0 & 1 \end{pmatrix}$ is the transformation matrix for the left manipulator, $TR_n^0 = \begin{pmatrix} Rn_x & Ro_x & Ra_x & Rx \\ Rn_y & Ro_y & Ra_y & Ry \\ Rn_z & Ro_z & Ra_z & Rz \\ 0 & 0 & 0 & 1 \end{pmatrix}$ is the transformation matrix for the right manipulator.

B. Amount of the training dataset

For the primary analysis, the generation of initial data was carried out based on solving the direct kinematics problem based on the kinematic model of manipulators. To determine the training dataset, it was proposed to make 2000 random positions for each manipulator. As a result, for the combination "each with each", we get 4 000 000 random positions.

Since most of the generated positions were far from potential self-collisions, it was decided to reduce the sample. Only those cases were left in the sample when one or both manipulators fall into a given corridor ± 0.07 m relative to the central axis of the robot, or pass through it (Fig. 3). The corridor is determined by the thickness of the manipulators. This value is the threshold value, which should not exceed the minimum distance between the links of the manipulators. This condition for the considered robot SAR-401 means that the coordinates along the central axis of the right manipulator are greater than 0.07 m, or the coordinates along the central axis of the left manipulator are greater than 0.07 m (Fig. 3).

As a result, three states remain in the sample:

- the left manipulator approaches the central axis of the robot, and the right manipulator is positioned arbitrarily;

- the right manipulator approaches the central axis of the robot, and the left manipulator is positioned arbitrarily.

- the left and the right manipulators approach the central axis of the robot.

Because of the reduction, out of 4 000 000 configurations, about 250 000 values remained. Of the 250 000 values, about 2 000 led to self-collisions of the manipulators.

According to the standard approach, the input dataset is divided in the following proportion: 70% is used for training data, 15% for validation, and 15% for testing.

Fig. 4 shows the situation where data is excluded.

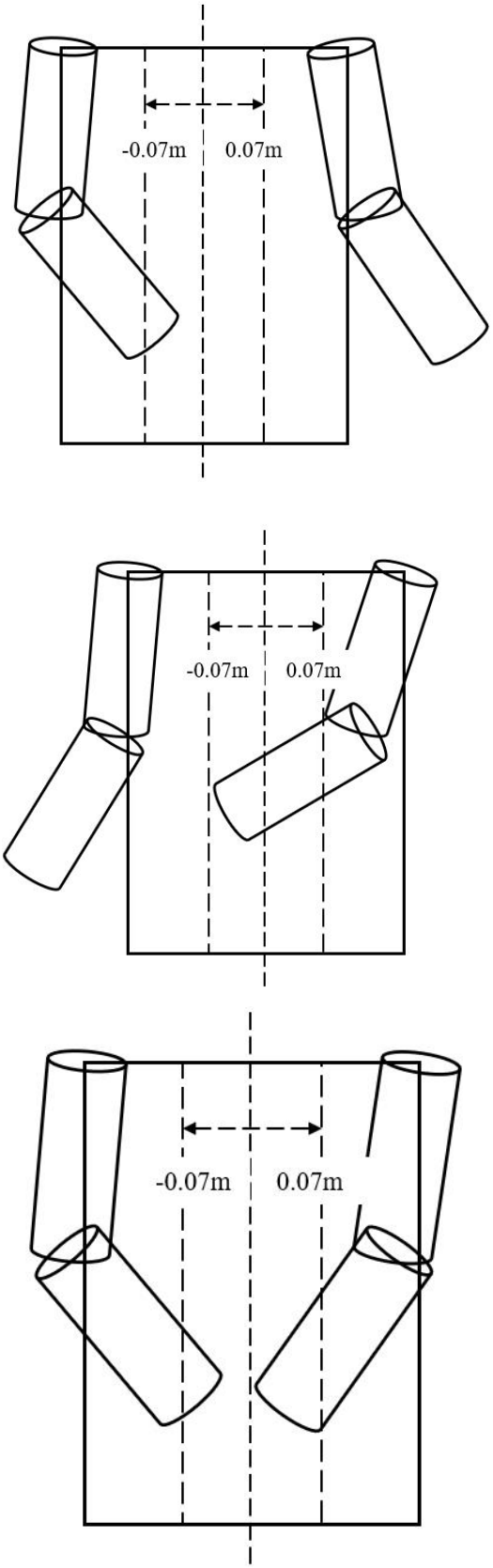


Fig. 3. The variants of the location of the manipulators for forming the input dataset

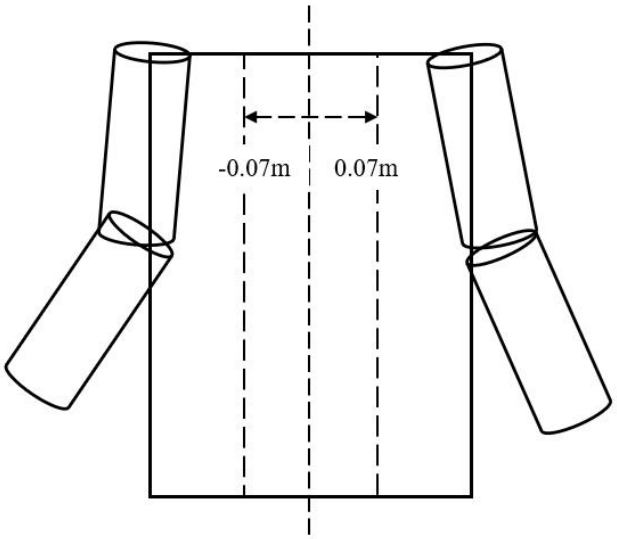


Fig. 4. Options for the manipulators to exclude from the input dataset

As a result of the experiments, we will choose the sigmoid function as the activation function, the loss function will be estimated using the Mean Squared Error method, and we will choose the Levenberg-Marquardt algorithm as the training function.

C. Hidden layers

Taking into account the recommendations for the hidden layers design [38, 39], we will prepare two neural networks for research to choose the most accurate solution. The neural network parameters are shown in Table IV.

TABLE IV. PARAMETERS OF NEURAL NETWORKS FOR SELF-COLLISION ANALYSIS

Network No.	Amount of training data	Number of hidden layers	Number of neurons in hidden layers
Network 1	250 000	1	11
Network 2	250 000	1	47

D. Neural network configuration

We will explore a multi-layered perceptron neural network structure (Fig. 5) [40, 41].

To calculate the error, we use the Mean Squared Error (MSE) method

$$MSE = \frac{1}{m}\sum_{i=1}^{n}(y_k - y_n)^2, \quad (9)$$

where y_k are the true output values of the robot, y_n are the output values that the NN received, n is the total amount of data.

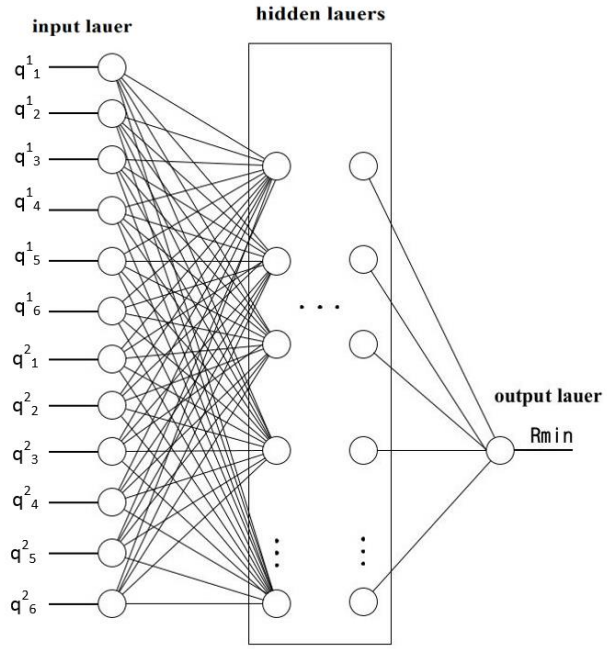
For the designed neural network, we need an activation function that will repeat all the properties of the sigmoid function but lie in the range from -1 to 1. For this task, we will use the hyperbolic tangent function

$$\tanh(n) = \frac{e^{2n} - 1}{e^{2n} + 1}. \quad (10)$$

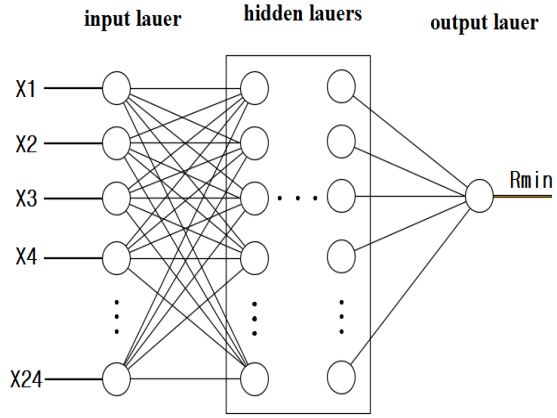
In MATLAB, we will use the tansig(x) function, which is equivalent to the function that implements the hyperbolic tangent tanh(x).

According to the standard approach, the input data is divided into the following proportion: 70% are used as training data, 15% as validation data, and 15% are used for testing [40, 41]. The duplicate input data should be removed [40, 41].

Based on the above, in Table V, we will give the characteristics of the designed neural network.



a)



b)

Fig. 5. Neural network architecture; a) Network 1, b) Network 2

The inputs and outputs of the NN (Fig. 5) are described as follows:

$$[Q_1] = ANN\_Net(X_1, X_2),$$

$$[Q_1] = ANN\_Net(q_1^1, q_2^1, q_3^1, q_4^1, q_5^1, q_6^1, q_1^2, q_2^2, q_3^2, q_4^2, q_5^2, q_6^2). \quad (11)$$

$$[Q_2] = ANN\_Net[Ln_x\ Ln_y\ Ln_z\ Lo_x\ Lo_y\ Lo_z\ La_x\ La_y\ La_z\ Lx\ Ly\ Lz\ Rn_x\ Rn_y\ Rn_z\ Ro_x\ Ro_y\ Ro_z\ Ra_x\ Ra_y\ Ra_z\ Rx\ Ry\ Rz]. \quad (12)$$

TABLE V. CHARACTERISTICS OF NEURAL NETWORKS 1 AND 2

Characteristics	Network 1	Network 2
Hidden layers	1	
The number of neurons in the first layer	11	47
Layer activation function	tansig	
Output layer activation function	Linear	
Loss function	MSE	
Learning function	trainlm	
Number of epochs	1000	
Number of inputs	6	24
Number of output	1	

V. TRAIN A NEURAL NETWORK AND RESULTS

We will train the considered neural networks (Section IV) using the backpropagation method on the obtained training data set to detect a possible self-collision of the SAR-401 robot manipulators. Self-collision is detected when the minimum distance between all manipulator links is less than 0.14 m. If the distance between any manipulator links is less than 0.14 m, we will signal that a collision can take place and block the motion.

To train neural networks, we will use a previously prepared training dataset consisting of 177 979 variants of the position of manipulators. The loss function for each of the neural networks is shown in Table VI.

TABLE VI. RESULTS OF NEURAL NETWORKS TRAINING

Network No.	Number of training data	Number of hidden layers	Number of neurons in hidden layers	Loss function (meters)
Network 1	177979	1	11	8.5806e-06
Network 2	177979	1	47	1.3562e-06

Fig. 6 shows the solution for Network 1, where the neural network out is shown in red, and the reference solution is shown in blue. d is the minimum distance between manipulators, t is the amount of training dataset. Fig. 7 shows the solution for Network 2, where, as above, the neural network out is shown in red, and the reference solution is shown in blue. We supply 12 values q_i^j of the left and the right manipulator to the input of Network 1. At the output of Network 1, we get one value - the minimum distance between manipulators d. The average error is 8.5806e-06 meters.

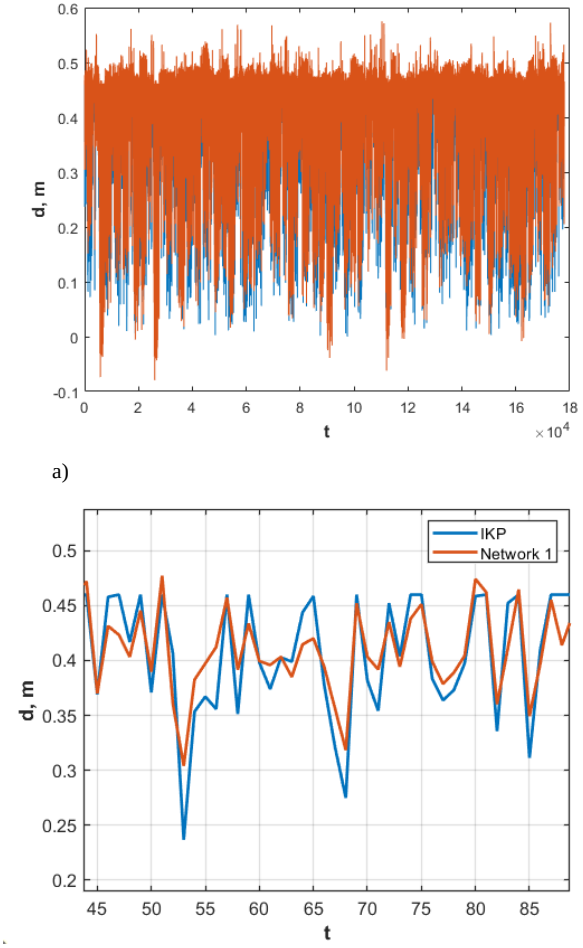


Fig. 6. a) Network 1 solutions; b) Scaled Network 1 solutions plot

At the input of Network 2, we supply the values of the transformation matrices – 24 values. At the output of the Network 2, we get one value - the minimum distance between the manipulators. The average error is 1.3562e-06 meters. It can be seen from the graphs that Network 2 shows a more accurate result.

Let's carry out an additional test of neural networks for the accuracy of a self-collision detection. For testing, we will select from a random sample of positions only those positions that lead to the self-collision (when the minimum distance between the manipulators is less than 0.14 meters). The value of 0.14 meters was chosen based on the condition of guaranteed blocking of the manipulators motion before the self-collision occurs.

The solution of neural networks on these data gives more demonstrative results (Fig. 8, Fig. 9). It can be seen that Network 2 is more accurate too. The average error when testing only self-collisions: Network 1 defines an accuracy of 0.0255 meters and Network 2 – 0.0029 meters.

Training of both neural networks showed good results in estimating the minimum distance between the manipulators and analyzing the self-collision between the robot manipulators with a minimum error. Thus, finally, we will stop our choice on Network 2, where elements of the transformation matrix are used as input dataset.

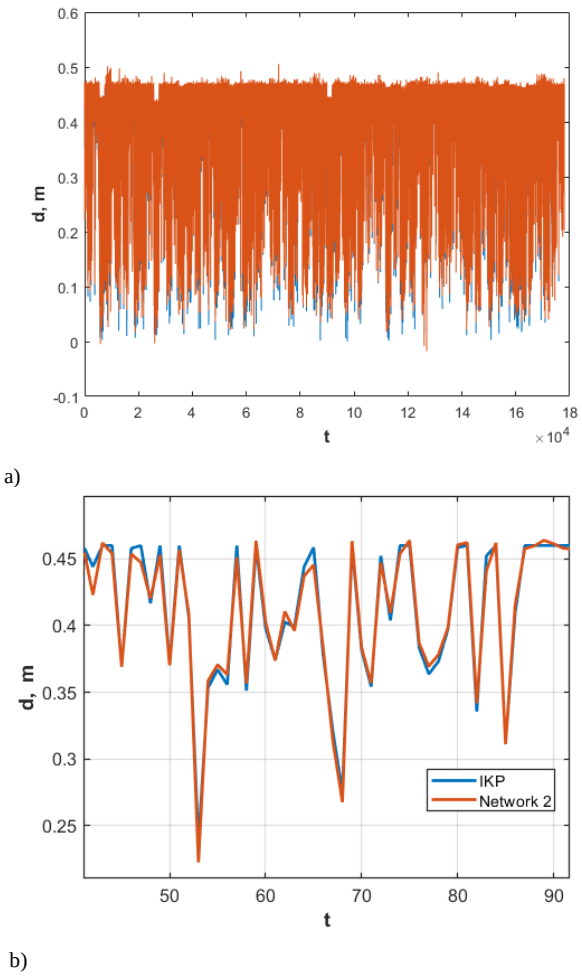
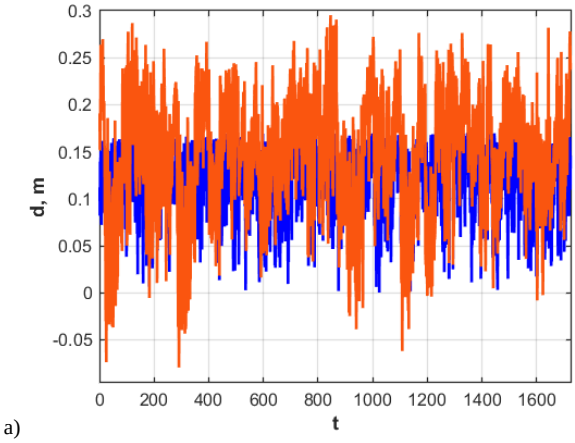


Fig. 7. a) Network 1 solutions; b) Scaled Network 2 solutions plot

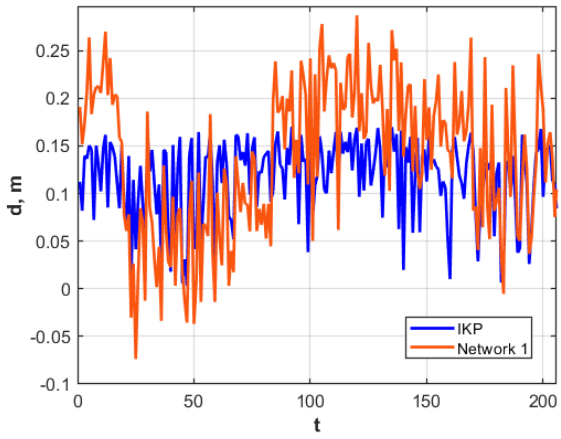
VI. TESTS

Using the developed neural network (Network 2), we will test the correct operation of the self-collision control algorithm. The tests will be carried out using MATLAB to check the operation of the control algorithm, using the SAR-401 robot simulator and the robot itself to check the copy-type control mode. As before, we set the minimum distance between the manipulators to 0.14 meters.

Based on the developed neural network, a universal approach to the self-collision avoidance control of dual-arm robot manipulators is proposed. The proposed system for the self-collisions avoidance of manipulators includes a serial connection of a software-mathematical module for determining the position of the manipulators, a collision detection module and a collision avoidance module, which are functionally connected in series. Before each new movement, an analysis is carried out using a trained neural network for the collision detection. If the neural network estimates the distance between the arms to be less than 0.14 m for this example (an occurrence of the self-collision), then this movement of the robot arms is blocked, thereby preventing the collision. Since the analysis of a possible collision is carried out using the neural network, the detection speed of which is higher than other methods, the collision check process does not slow down the work of the robot as a whole and allows you to work in real time, even if there is a need to break the trajectory into 100 or more parts.

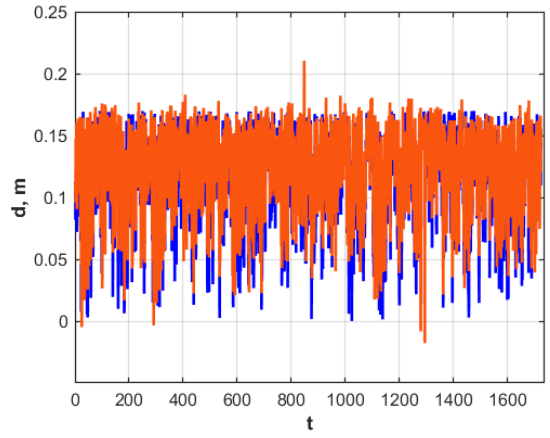


a)

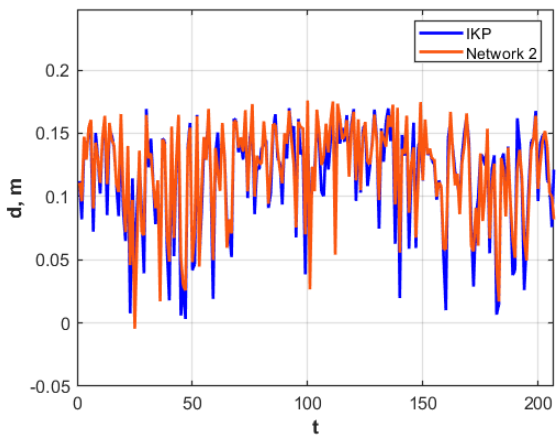


b)

Fig. 8. a) Network 1 solutions; b) Scaled Network 1 solutions plot



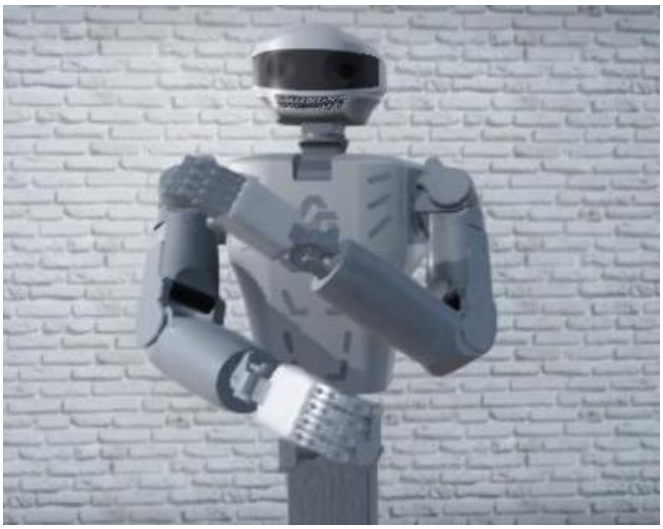
a)



b)

Fig. 9. a) Network 2 solutions; b) Scaled Network 2 solutions plot

As a result of the work, in Figs. 10-12 it can be seen that the developed neural network works correctly, providing information to block the movements of the manipulators in case their possible self-collision is detected.



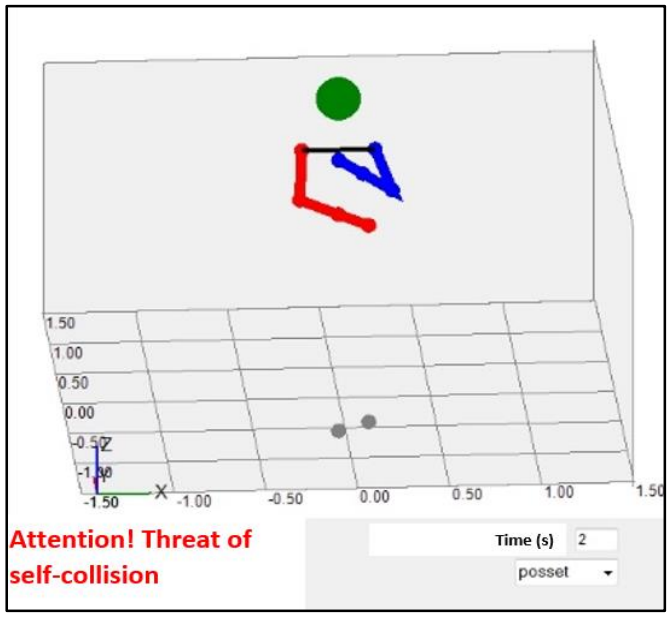


Fig. 10. The result of the algorithm in MATLAB and in the simulator

Fig. 11. Manipulator self-collision blocking on the robot SAR-401 simulator

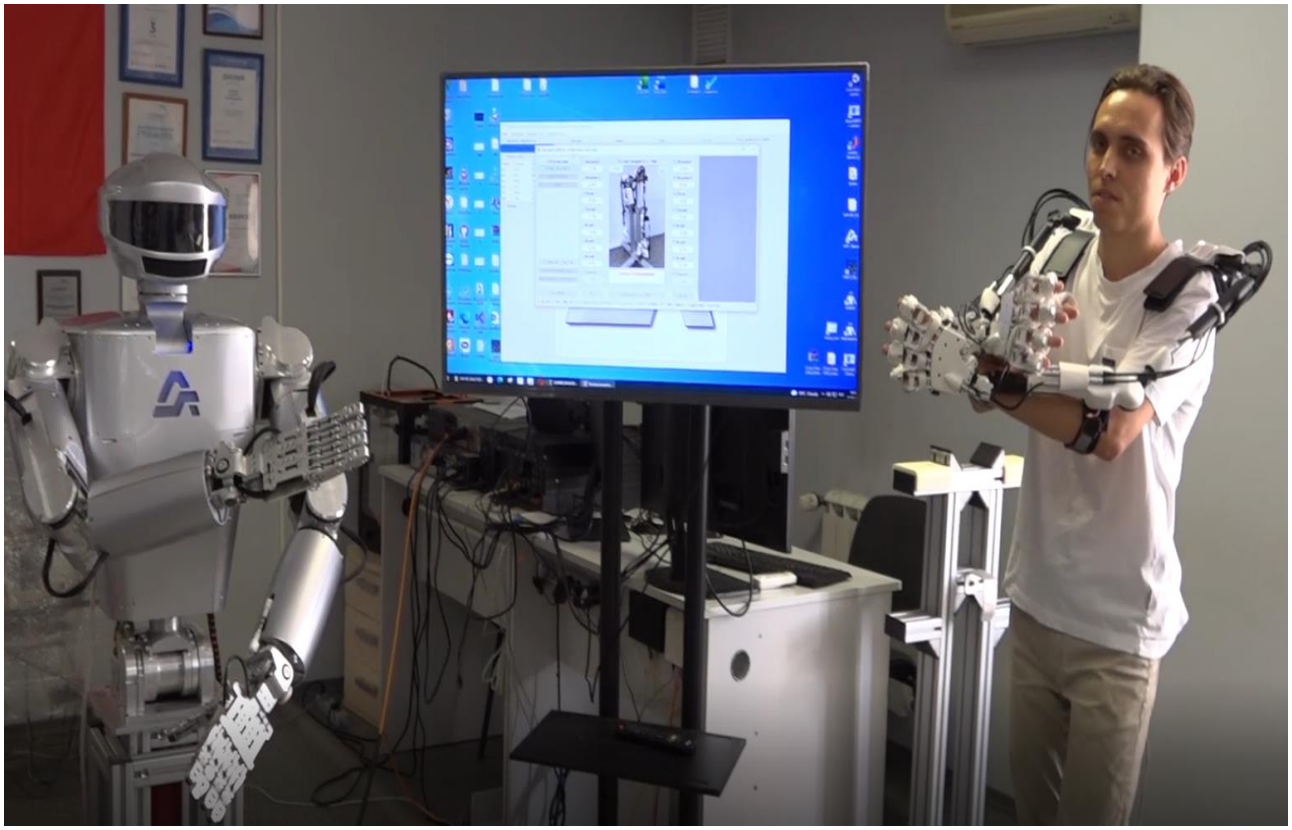


Fig. 12. Manipulator self-collision blocking on the robot SAR-401

VII. CONCLUSION

In the control design for multilink manipulators, neural network approaches have been actively used recently. An important component of the problem of control design for multilink manipulators during their cooperative work is the control of their possible self-collision. This task becomes especially important when implementing a copy-type control.

The article proposes an approach to the analysis of a self-collision of multilink manipulators during their cooperative work. The proposed approach, in contrast to the approaches based on algebraic methods, is based on the use of neural networks. A feature of solving the problem is its solution using the regression method, in contrast to the classification approach, which gives a forecast with a certain probability of less than 100%, which consequently may lead to an incorrect result.

Therefore, an efficient solution was obtained in terms of algorithm computational speed and accuracy compared to algebraic approaches based.

In this paper, the self-collision control problem is solved based on a regression approach using neural networks. Although in the future the authors see the solution of this problem based on the classification approach to obtain an unambiguous answer - whether there is a self-collision, without additional processing of the neural network output.

REFERENCES

[1] M. Ostanin, D. Popov, and A. Klimchik, "Programming by Demonstration Using Two-Step Optimization for Industrial Robot," IFAC-PapersOnLine, vol. 51, no. 11, pp. 72–77, 2018.

[2] S. Calinon, "Learning from Demonstration (Programming by Demonstration)," Encyclopedia of Robotics, M. H. Ang, O. Khatib, B. Siciliano, Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 1–8.

[3] V. Alchakov, V. Kramar, and A. Larionenko, "Basic approaches to programming by demonstration for an anthropomorphic robot," IOP Conference Series: Materials Science and Engineering, vol. 709, no. 2, 2020, p. 022092.

[4] J. Peters, D. Lee, J. Kober, D. Nguyen-Tuong, J. Bagnell and S. Schaal, Chapter 15: Robot Learning, Springer Handbook of Robotics. Springer International Publishing, 2017.

[5] Z. Zhu and H. Hu, "Robot Learning from Demonstration in Robotic Assembly: A Survey," Robotics, vol. 7, no. 2, p. 17, 2018.

[6] X. Tian, Q. Xu, and Q. Zhan, "An analytical inverse kinematics solution with joint limits avoidance of 7-DOF anthropomorphic manipulators without offset," Journal of the Franklin Institute, vol. 358, no. 2, pp. 1252–1272, 2021.

[7] Y. Wang, P. Ogren, C. Simth, F. Vina, Y. Karayiannidis, "Dual arm manipulation using constraint based programming," Proc. 19th World Congress IFAC,Cape Town, vol 19, 2014, pp. 311-319.

[8] V. Kramar, A. Kabanov, and V. Alchakov, "Application of linear algebra approaches for predicting self-collisions of dual-arm multi-link robot," International Journal of Mechanical Engineering and Robotics Research., vol. 9, no. 11, pp. 1521-1525, 2020.

[9] V. A. Kramar, "The method for predicting self-collisions of multi-link manipulators," Journal of Physics: Conference Series, vol. 1661, p. 012052, 2020.

[10] M. Lei, T. Wang, C. Yao, H. Liu, Z. Wang, and Y. Deng, "Real-Time Kinematics-Based Self-Collision Avoidance Algorithm for Dual-Arm Robots," Applied Sciences, vol. 10, no. 17, p. 5893, 2020.

[11] Y. Liu, C. Yu, J. Sheng, and T. Zhang, "Self-collision Avoidance Trajectory Planning and Robust Control of a Dual-arm Space Robot," Int. J. Control Autom. Syst., vol. 16, no. 6, pp. 2896–2905, 2018.

[12] A. A. Kabanov and D. A. Tokarev, "Self-collision Avoidance Method for a Dual-arm Robot," Proc. of 2019 3rd International Conference on Control in Technical Systems, 2019, pp. 273-276.

[13] C. Scoccia, G. Palmieri, M. C. Palpacelli, and M. Callegari, "A Collision Avoidance Strategy for Redundant Manipulators in Dynamically Variable Environments: On-Line Perturbations of Off-Line Generated Trajectories," Machines, vol. 9, no. 2, p. 30, Feb. 2021.

[14] A. Y. Afaghani, Y. Aiyama, "On-line collision avoidance between two robot manipulators using collision map and simple escaping method," Proc. IEEE/SICE Int. Symposium on System Integration, 2013, pp. 105-110.

[15] A. Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, G. Hirzinger, "The skeleton algorithm for real-time collision avoidance of a humanoid manipulator interacting with humans," Proc. IEEE/ASME international conference on advanced intelligent mechatronics, Zurich, 2007, 9871732.

[16] Y. Wang, P. Ogren, C. Simth, F. Vina, and Y. Karayiannidis, "Dual-arm manipulation using constraint-based programming," in Proc. 19th World Congress IFAC, Cape Town, vol. 19, 2014, pp. 311-319.

[17] S. Tarbouriech, B. Navarro, P. Fraisse, A. Crosnier, A. Cherubini "Dual-arm relative tasks performance using sparse kinematic control," IROS: Intelligent Robots and Systems, 2018, pp.6003-6009.

[18] T. Kivelä, J. Mattila, J. Puura, and S. Launis, "On-Line Path Planning With Collision Avoidance for Coordinate-Controlled Robotic Manipulators," Proc. ASME/BATH 2017 Symposium on Fluid Power and Motion Control, 2017, p. V001T01A048.

[19] T. Hu, T. Wang, J. Li, and W. Qian, "Obstacle Avoidance for Redundant Manipulators Utilizing a Backward Quadratic Search Algorithm," International Journal of Advanced Robotic Systems, vol. 13, no. 3, 2016, p. 119.

[20] X. Xu, Y. Hu, J. Zhai, L. Li, and P. Guo, "A novel non-collision trajectory planning algorithm based on velocity potential field for robotic manipulator," International Journal of Advanced Robotic Systems, vol. 15, no. 4, p. 172988141878707, 2018.

[21] S. Chen, M. Luo, and F. He, "A universal algorithm for sensorless collision detection of robot actuator faults," Advances in Mechanical Engineering, vol. 10, no. 1, p. 168781401774071, 2018.

[22] Z. Hou, S. Ma, Q. Zeng, and A. Li, "Kinematics analysis and self - collision detection of Truss type multi-robot cooperative welding platform," Procedia CIRP, vol. 81, 2019, pp. 488–493.

[23] X. Wang, C. Yang, Z. Ju, H. Ma, and M. Fu, "Robot manipulator self-identification for surrounding obstacle detection," Multimed Tools Appl, vol. 76, no. 5, pp. 6495–6520, 2017.

[24] K. Jang, S. Kim, and J. Park, "Reactive Self-Collision Avoidance for a Differentially Driven Mobile Manipulator," Sensors, vol. 21, no. 3, p. 890, 2021.

[25] A. A. Kabanov, O. A. Kramar, Kramar, V.A. "Collision Avoidance Control of Multi-link Robotic Manipulators with a Copy-type Master Device," Proc. International Russian Automation Conference, RusAutoCon, 2021, pp. 639–643.

[26] Y. Li, S. Li, и B. Hannaford, "A Novel Recurrent Neural Network for Improving Redundant Manipulator Motion Planning Completeness," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 2956–2961.

[27] Z. Liao, G. Jiang, F. Zhao, X. Mei, и Y. Yue, "A novel solution of inverse kinematic for 6R robot manipulator with offset joint based on screw theory," International Journal of Advanced Robotic Systems, vol. 17, no. 3, p. 172988142092564, 2020.

[28] M. Rouhani and S. Ebrahimabadi, "Inverse kinematics of a 7-DOF redundant robot manipulator using the active set approach under joint physical limits," Turk J Elec Eng & Comp Sci, vol. 25, pp. 3920 – 3931, 2017.

[29] O. M. Omisore, "Non-iterative geometric approach for inverse kinematics of redundant lead-module in a radiosurgical snake-like robot," BioMed Eng OnLine, vol. 16, no. 1, p. 93, dec. 2017.

[30] A. R. J. Almusawi, L. C. Dülger, and S. Kapucu, "A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242)," Computational Intelligence and Neuroscience, vol. 2016, pp. 1–10, 2016.

[31] T. Uhl, Advances in Mechanism and Machine Science: Proceedings of the 15th IFToMM World Congress on Mechanism and Machine Science, V. 73. Cham: Springer International Publishing, 2019.

[32] V. Kramar, O. Kramar, and A. Kabanov, "An Artificial Neural Network Approach for Solving Inverse Kinematics Problem for an Anthropomorphic Manipulator of Robot SAR-401," Machines, vol. 10, no. 4, p. 241, 2022.

[33] A. Bogdanov, E. Dudorov, A. Permyakov, A. Pronin, и I. Kutlubaev, "Control System of a Manipulator of the Anthropomorphic Robot FEDOR," 2019 12th International Conference on Developments in eSystems Engineering (DeSE), 2019, pp. 449–453.

[34] A. A. Kabanov and A. N. Balabanov, "The modeling of an anthropomorphic robot arm," MATEC Web of Conferences, vol. 224, 2018, p. 141807.

[35] K. M. Lynch and F. C. Park, Modern robotics: mechanics, planning, and control, Cambridge, UK: Cambridge University Press, 2017.

[36] P.I. Corke, Robotics, vision and control fundamental algorithms in Matlab: Springer, Cham, 2017.

[37] Shuai Li, Yunong Zhang, Long Jin, "Kinematic Control of Redundant Manipulators Using Neural Networks," IEEE Transactions On Neural Networks And Learning Systems, vol. 28, no. 10, pp. 2243-2254, 2017.

[38] Ng, A. Machine Learning Yearning; GitHub eBook (MIT Licensed): San Francisco, CA, USA, 2018, p. 118.

[39] J. Heaton, Deep learning and neural networks. St. Louis, MO: Heaton Research, Inc, 2015.

[40] C. E. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," Proc. of the 2nd International Conference on Computational Sciences and Technology, 2020, pp. 124 – 133.

[41] A. Bhoi, P. Mallick, CM. Liu, V. Balas, Bio-inspired Neurocomputing. Studies in Computational Intelligence, Springer, Singapore, 2021.