INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 9(5):539-554, October, 2014.

ANN Method for Control of Robots to Avoid Obstacles

E. Ciupan, F. Lungu, C. Ciupan

Emilia Ciupan, Florin Lungu, Cornel Ciupan*
Technical University of Cluj-Napoca
Romania, 400641 Cluj-Napoca, Bd. Muncii, 103-105
emilia.ciupan@mis.utcluj.ro, florin.lungu@mis.utcluj.ro
*Corresponding author: cornel.ciupan@muri.utcluj.ro

Abstract: The avoidance of obstacles placed in the workspace of the robot is a problem which makes controlling them more difficult. The known avoidance methods used for the robots control are based on bypass trajectory programming or on using the sensors that detect the position of the obstacle. This paper describes a method of training industrial robots in order for them to avoid certain obstacles in the workspace. The method is based on the modelling of the robot's kinematics by means of an artificial neural network and by including the neural model in the robot's controller. The neural model simulates the robot's inverse kinematics, and provides the joint coordinates, as referential values for the controller. The novelty of the method consists in the deliberately erroneous training of the network, so that, when programming a direct trajectory in the workspace, the robot avoids a known obstacle.
Keywords: Artificial Neural Network (ANN), control, robot, obstacle avoidance.

1 Introduction

Proportional-integrative-derivative PID controller is widely used for the control of robots, because it is model-free, and its parameters can be adjusted easily and separately. An integrator in a PID controller reduces the bandwidth of the closed-loop system. In order to remove steady-state error caused by uncertainties and noise, the integrator factor has to be increased, having the effect of reducing the performance of transient regime [19].

The application of neural networks to robots control is well known [10], [11] and an alternative to the adaptive control is represented by the neural controllers [21].

Lewis et al. [10] demonstrate that neural networks do indeed fulfil the promise of providing model-free learning controllers for a class of nonlinear systems. Neural network control offers two specific advantages over adaptive control:

- neural network controller works for any rigid robot arm without computing a regression matrix or performing any preliminary analysis
- neural networks provide a basis set for any smooth function, while the linear in the parameters equation provides a basis set only for linear systems.

There are several approaches to combine PID control with the intelligent control, such as the neural control. The first way is to form neural networks into PID structure [5], [6], [10], [17]. By proper updating laws, the parameters of PID controllers are changed so that the closed-loop systems are stable. The second method used intelligent techniques to tune the parameters of PID controllers, such as fuzzy tuning [11], neural tuning [7], [18], and expert tuning [8].

All known approaches require the set point (the reference values) determination that consists in drive joints coordinates which are obtained from the inverse kinematic analysis.

In the inverse kinematic analysis [14], [16], the coordinates and the effector's orientation (X,Y,Z,ψ,θ,φ) are considered to be known, and the coordinates of the joints (represented by q_i, i=1, ..., m, where m is the number of kinematic axes, equal to the number of the degrees of

freedom) are to be determined. Although an apparently easy task, determining the coordinates of the joints becomes more complicated when robots with complex kinematic structure, such as the parallel robots, are at stake [20].

Given its advantages, neural computing is often used to solve the problem of inverse kinematics. The training of the neural network and the getting of the neural model implies solving two important problems [7]:

a) getting the training data, especially when the mathematical model is not known, and measurements on the physical model are necessary

b) performing the training process and obtaining an acceptable error, in the case of a large amount of training data.

The proper control of the robot is carried out by the robot's control equipment, by means of generating a control input for each joint, so that it achieve coordinate q_i resulted from the inverse kinematics, and the effector pass through the points that belong to the trajectory. Therefore an important problem is to determine the coordinates of joints.

Figure 1 shows a neural controller for the positioning of the effector, which uses a neural model NM for the generation of the coordinates of the joints q_i, PID controllers and feedback loops. The method implies the completion of the following stages: providing the coordinates of the some points that define the robot's trajectory, generating some additional points on this trajectory and determining the coordinates of the joints by using an NM neural model implemented onto the robot's control equipment, transmitting the coordinates of the joints to the controllers of the PID_i axes, which generate the actuating quantity e_i corresponding to the M_i motors of the robot.

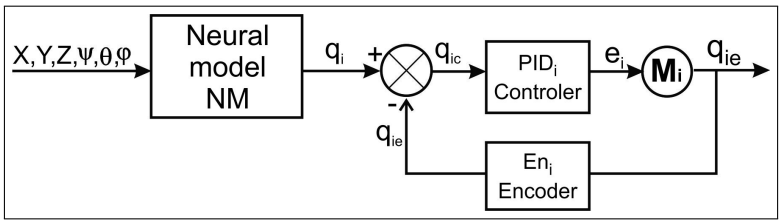

Figure 1: Neural network controller

In order to obtain and implement the NM neural model, it is necessary to complete the following steps: creating a neural network, creating a set of training examples, training the neural network, which results in the creation of the neural model, testing and validating the neural model, and using the neural model by implementing it in the control equipment. The set of training examples consists of pairs of input-output data which are determined by the choice of a point cloud in the robot's workspace. The input signals are considered to be the positioning coordinates of the points in the workspace, while the output signals, the coordinates of the joints associated with these positions.

The disadvantage of this method is that when programming a straight trajectory between two points the robot cannot avoid an obstacle, even though the volume covered by the obstacle has been excluded from the set of training examples. At the same time, another disadvantage consists in the large number of training examples needed to cover the entire workspace of the robot.

In the majority of the handling applications, the robot's task is to complete linear movement between points that belong to the workspace, points where it has to retrieve or deliver objects, or where the robot perform operations. This general method can also be used to obtain a neural

model in the case of the effector's move between two points, so that it can avoid an obstacle. In this case, the set of training examples is constituted by the association of coordinates of the points on the deviation trajectory with the coordinates of the joints corresponding to them. It is necessary for the trajectory to be described in this case.

A way of describing the trajectory is by its mathematical expression, which has the disadvantage of having to determine it. The next step consists in the determination of the coordinates of the joints for a set of points that belong to the bypass trajectory. For complex structure robots, such as the parallel robots with 6 degrees of freedom, the expression of the joints determination is complicated. The general form of the expression is $q_i=f_i(X,Y,Z,\psi,\theta,\varphi)$, i=1, ...,6. In this case, the calculation is complex and it involves a large number of mathematical operations. This is a disadvantage, especially in the case of robots that operate at high speeds, given that calculations are made in real time.

Also, in the patent literature [9], [12], [13], there are many methods of robot control using neural networks. Patent CN102346489 discloses a pulse neural network method of robot object tracking control. The collision avoidance is done by processing a set of information from the sensors. There is no information regarding a method to avoid a static obstacle in the robot's workspace to be achieved exclusively by neural network training, without the use of visual sensors to identify the obstacle [15].

Some of the authors have been previously involved in research concerning the use of neural networks for economic applications, or robot control. The program presented in [1], [2] was designed with the purpose of using neural computing in the modelling and the simulation of processes or activities. It is suitable for the study of any activity for which a three-layer perceptron neural network may serve as a model.

Also, [3] shows a neural model for the kinematical analysis of six parallel robot. For reasons related to simplification, there has been considered the move of the effector in a cube with a side of 10 cm, without taking into account the variation of the position angles. For the training of the network, there have been generated 130 training examples, and then the neural model for the move of the robot on different trajectories has been validated. In the application of the neural model, there has been noticed that the training of the neural model in a larger working space, specific to a robot, is difficult, especially when the robot has to avoid an obstacle. That is why the authors have aimed to develop a more effective method of control for the cases in which the robot has to avoid an obstacle.

In [4] there has been presented the main principle of a method of obstacle avoidance, by means of an erroneous instruction of the network. The experiments have been completed in a smaller part of the robots workspace (a cube with the side 10 cm). Further research illustrated the difficulty of obtaining an effective neural model that can lead the effector with a precision that is appropriate to the application, and avoid the obstacle. There have been circumstances in which the neural model did not offer the coordinates that lead the effector beside the obstacle [3]. The validation of the models presented [3], [4] has been made based on sets of data that have not been used as training data; there has been completed no cross-validation.

In the current paper the authors have developed the method in terms of the generation of the set of the training examples. In this sense, there have been stated clear rules that establish the training examples. In a first stage, there has been completed a neural model tested through a 4-fold cross-validation technique. The case studies have been carried out in a workspace that had the shape of a cube with a side of 600 mm, corresponding to the majority of the applications, by using an obstacle with a cube shape with a side of 200 mm. In order to avoid the collision, there have been studied three types of envelopes.

There are several types of neural networks that can be used in modelling a system. They can be classified based on criteria such as the structure or the instruction types (networks of

perceptron type, radial basis function networks, Kohonen self-organization networks, Hopfield networks, fuzzy neural networks, networks with supervised or non-supervised training and so on). In all the modelling activities that are referred to in this paper there have been used neural networks of the type of a three-layer perceptron, in which the initial layer has 6 neurons corresponding to the position (X,Y,Z,ψ,θ,φ) of the effector. In the case of some of these models, the final layer has 6 neurons corresponding to the coordinates of the joints q_i, i=1, ...,6, or only one neuron, corresponding to the coordinate of a single joint (q_3 for instance). The number of the neurons of the hidden layer has been determined by trial, throughout several training sessions, based on the criteria of the minimization of the mean square error. The chosen activation functions have been log-sigmoid for the neurons of the hidden layer, and the function purelin for the neurons on the output layer. The training of the networks has been completed using the Levenberg-Marquardt method. Unlike the descent gradient method, the process of training through the Levenberg-Marquardt method could converge quickly when close to the solution. As it is a method based on Hessian, there is no risk that in such a circumstance the solution be lost, as it can occur in the training with the descent gradient method, when a higher rate of learning is used. In the modelling activity, there has been used the Matlab application.

2 Method to avoid obstacle

2.1 Method presentation

The problem that this paper solves consists in the elaboration of an industrial robot training method based on neural network modelling and training, so that, when programming a straight trajectory between two points through which the robot's effector has to pass, the robot avoid an obstacle that it encounters.

The method of training robots to avoid obstacles is based on the modelling, training and use of three-layer perceptron type neural networks, having k neurons in the input layer, which corresponds to the number of degrees of freedom, m neurons in the output layer, which corresponds to the number of kinematic axes, and a number n, consisting of 15 to 50 neurons, which corresponds to the hidden layer. Figure 2 shows the scheme of a neural network of this type that has 6 neurons in the input layer, corresponding to the 6 degrees of freedom, 6 neurons in the output layer, corresponding to the 6 kinematic axes and an unspecified number n, in the hidden layer.

The training data are determined from the mathematical model or by experimenting on the physical model of the robot, by choosing a point cloud contained in a work plane included in the robot's workspace, plane in which the robot has to operate at least one move between two given points. The avoidance of the obstacle in the robot's trajectory is achieved by the training of the network, with input data corresponding to the coordinates of some points on the robot's direct trajectory between two points, and output data (the joints coordinates) corresponding to the bypass trajectory.

The set of training examples will have as input signals the coordinates of the cloud of points, while as output ones, the coordinates of the joints, calculated according to the rule R below:

R1) for a point in the work plane outside the obstacle or its envelope, it is established as pair the coordinates of the correct joints (in accordance with the mathematical model which describes the robot's kinematics, or in accordance with the experimental measurements)

R2) for a point in the work plane that belongs to the obstacle or its envelope, it is established as pair the coordinates of the joints that belong to a different point, the latter being situated on the surface of the obstacle, or on its envelope, depending on the case.

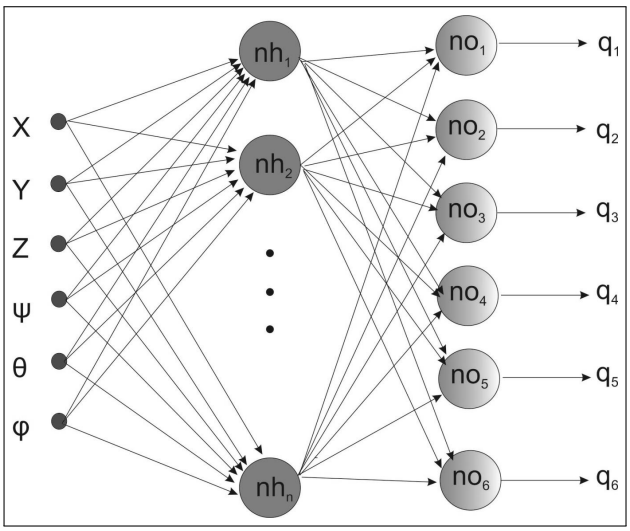

Figure 2: Neural network diagram

The number of neurons in the hidden layer is chosen by means of trials, a practice which is used in neural computing.

The novelty of this method consists in the way the training set is built. This leads to a deliberately erroneous training, so that in the recall phase it is not necessary to know the bypass trajectory.

The set of training examples is constituted by input-output data pairs, in which the input signals correspond to some points on the robot's direct Td trajectory. The output signals, in the training phase, are the coordinates of the joints, but in "deliberately erroneous" way, they are not the coordinates associated to the direct trajectory that crosses the obstacle, but to the output signals corresponding to the points situated outside the obstacle, on a Ta avoidance trajectory. Thus, in the recall phase, the model is going to behave erroneously. This means that for the points on a direct trajectory (a line, in most of the cases) which does not bypass the obstacle, transmitted as input data, the neural model will generate, as output, the coordinates of the joints that will lead the robot beside the obstacle.

In order to achieve the neural model NM, one has to complete the following steps:

- create a neural network that has, in its input layer, a number of neurons equal to the number of the robot's degrees of freedom, and in its output layer, a number of neurons equal to the number of joints q_i

- create a set of training examples formed by pairs of effector coordinates, which belong to the robot's work plane, and corresponding coordinates of the joints q_i, determined according to rule R described above

- train a neural network with the sets of training data, the result of the training process being called "neural model"

- test the neural model achieved previously and validate it, in case acceptable errors are obtained

- use the neural model, which means that the neural model receives exclusively input data which consists of robot effector positions, and generates the coordinates of the joints.

2.2 Cross validation of the method

Let us consider the case of a serial robot with six degrees of freedom (Figure 3), given by three positioning movements (X, Y and Z), and other three effector orientation movements (ψ, θ, φ). Based on the robot's kinematic scheme (Figure 3a), one can describe the connecting relations

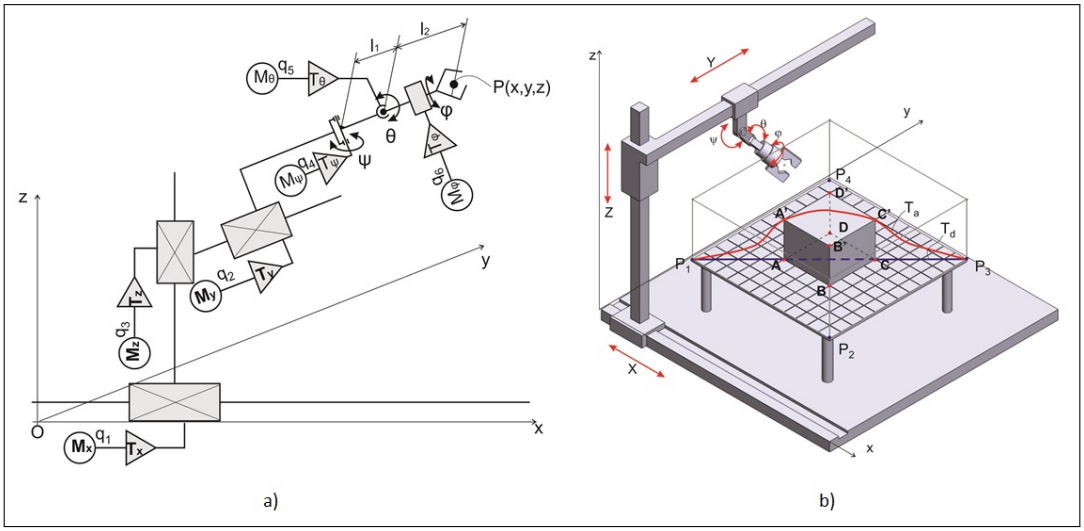

Figure 3: a. Kinematic diagram; b. The robot architecture

between the joints coordinates q_i, as well as the effector's position (X, Y, Z, ψ, θ, φ). Thus, the mathematical model for the inverse kinematics is obtained by solving the system of equations (1)-(6):

$$X = X_0 + q_1 \cdot i_{Tx} + l_2 sin(\theta) \tag{1}$$

$$Y = Y_0 + q_2 \cdot i_{Ty} + (l_1 + l_2 cos(\theta)) sin\psi \tag{2}$$

$$Z = Z_0 + q_3 \cdot i_{Tz} - (l_1 + l_2 cos(\theta)) cos\psi \tag{3}$$

$$\psi = \psi_0 + q_4 \cdot i_{T\psi} \tag{4}$$

$$\theta = \theta_0 + q_5 \cdot i_{T\theta} \tag{5}$$

$$\varphi = \varphi_0 + q_6 \cdot i_{T\varphi} \tag{6}$$

where i_{Tx}, i_{Ty}, i_{Tz}, $i_{T\psi}$, $i_{T\theta}$, $i_{T\varphi}$ represent the transfer functions of the transforming mechanisms which generate the given movements, and X_0, Y_0, Z_0, ψ_0, θ_0, φ_0 represent the initial values obtained for q_i=0, i=1,...,6.

The architecture of the robot in Figure 3a is shown in Figure 3b. A portion of the robot's workspace (Figure 3b), which has the shape of a parallelepiped with base $P_1P_2P_3P_4$ is being considered. In the workspace there is an obstacle ABCD$A'B'C'D'$ which has to be avoided during the operation. It is assumed that the robot's effector has to move between points P_1 P_3, but on a trajectory that should avoid the obstacle ABCD$A'B'C'D'$.

The plane of points P_1, P_2, P_3, P_4 is called work plane (represented as Wp), namely a plane in which the robot has to complete a number of operations (retrieve or place objects, feed equipment etc.).

We are looking for a simple and comfortable method of the robot control, which would provide as input data the coordinates of some points on the direct trajectory Td and make the robot move

on the trajectory Ta which avoids the obstacle. This is achieved by a "deliberately erroneous" training of the neural network which models the robot's behaviour.

The set of training data, under the form of input-output matrix pairs, is obtained by the association of the effector's coordinates (X, Y, Z, ψ, θ, φ) with the joints coordinates (q_i, i=1, ..., 6) resulted by means of mathematical model or by measuring on the physical model. The generation of the training data is made according to table of Figure 4.

No.	Points of training	Wished input						Wished output	Origin q_i
1.	P_1	X_1	Y_1	Z_1	ψ_1	θ_1	φ_1	$q_{i,1}$	P_1
2.	P_2	X_2	Y_2	Z_2	ψ_2	θ_2	φ_2	$q_{i,2}$	P_2
3.	P_3	X_3	Y_3	Z_3	ψ_3	θ_3	φ_3	$q_{i,3}$	P_3
4.	P_4	X_4	Y_4	Z_4	ψ_4	θ_4	φ_4	$q_{i,4}$	P_4
5.	A	X_A	Y_A	Z_A	ψ_A	θ_A	φ_A	$q_{i,A'}$	A'
6.	B	X_B	Y_B	Z_B	ψ_B	θ_B	φ_B	$q_{i,B'}$	B'
7.	C	X_C	Y_C	Z_C	ψ_C	θ_C	φ_C	$q_{i,C'}$	C'
8.	D	X_D	Y_D	Z_D	ψ_D	θ_D	φ_D	$q_{i,D'}$	D'
9.	O_1	X_{O1}	Y_{O1}	Z_{O1}	ψ_{O1}	θ_{O1}	φ_{O1}	$q_{i,O'1}$	O'_1

Figure 4: The way for obtaining the training data

In order to avoid the obstacle, the robot has to move on a deviating trajectory Ta. For the robot to move on the deviating trajectory Ta, its control equipment has to receive information regarding the shape of the trajectory as some coordinates of some points on the trajectory. A possible description of the trajectory is given by its mathematical expression, which has the disadvantage of having to determine it. The next step is to determine the coordinates of the joints for a set of points which belong to the deviating trajectory Ta.

According to the approach of this paper the avoiding trajectory is approximated by a set of few points, without knowing the mathematical expression of the trajectory, and the determination of the joints is done on basis of a neural model.

In order to complete the data in table of Figure 4, in the case of the robot in Figure 3b, there have been chosen the points P_1(200,200,200), P_2(800,200,200), P_3(800,800,200) and P_4(200,800,200). The obstacle is considered to be a parallelepiped defined by the points A(400,400,200), B(600,400,200), C(600,600,200) and D(400,600,200), situated in the work plane and the points A'(400,400,400), B'(600,400,400), C'(600,600,400) and D'(400,600,400), situated in a plane parallel to the work plane, 200 mm away.

For the validation of the method, there has been considered the move of the effector on the diagonal P_1P_3 (figure 3b). In order to establish the set of the training and testing examples for the move on the segments P_1A and CP_3, rule R1 is applied. As for the movement on the segment AC, there is applied rule R2. The set of the training data is shown in table of Figure 5.

There has been applied a 4-fold cross-validation technique of the method. Thus, the set of the training examples in table of Figure 5 has been divided into four subsets. The inclusion rule regarding the training examples in the four subsets is described by means of the indexes of the lines in table of Figure 5, grouped in the subsets M_k, k=1, ...,4, according to the algorithm below:

$$M_k = \{j_{k,n} \mid j_{k,n} = k + 4n, k \in [1, 4], n \in [0, 75]\} \tag{7}$$

For the validation of the method, out of the four subsets, there has been successively retained a subset for validation, while the other three subsets have been used, merged, for training. Thus, there have been completed four rounds of training and validation of the method, using the neural networks. Within each round, there have been developed several neural models of the three-layer perceptron type, having the architecture 6-m-6, where m represents the number of

Training points	Effector coordinates							Joint coordinates ($q_{i,j}$, i=1,...,6; j=1,...,301)					
j	X_j	Y_j	Z_j	Z'_j	ψ_j	θ_j	φ_j	$q_{1,j}$	$q_{2,j}$	$q_{3,j}$	$q_{4,j}$	$q_{5,j}$	$q_{6,j}$
0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	200	200	200	200	0	0	0	20	20	20	20	20	20
2	202	202	200	200	0	0	0	20.2	20.2	20	20	20	20
3	204	204	200	200	0	0	0	20.4	20.4	20	20	20	20
4	206	206	200	200	0	0	0	20.6	20.6	20	20	20	20
...													
100	398	398	200	200	0	0	0	39.8	39.8	20	20	20	20
101	400	400	200	400	0	0	0	40	40	40	20	20	20
102	402	402	200	400	0	0	0	40.2	40.2	40	20	20	20
103	404	404	200	400	0	0	0	40.4	40.4	40	20	20	20
...					0	0	0	0	0	0	20	20	20
199	596	596	200	400	0	0	0	59.6	59.6	40	20	20	20
200	598	598	200	400	0	0	0	59.8	59.8	40	20	20	20
201	600	600	200	400	0	0	0	60	60	40	20	20	20
202	602	602	200	200	0	0	0	60.2	60.2	20	20	20	20
203	604	604	200	200	0	0	0	60.4	60.4	20	20	20	20
204	606	606	200	200	0	0	0	60.6	60.6	20	20	20	20
...													
299	796	796	200	200	0	0	0	79.6	79.6	20	20	20	20
300	798	798	200	200	0	0	0	79.8	79.8	20	20	20	20
301	800	800	200	200	0	0	0	80	80	20	20	20	20

Figure 5: The set of training examples for cross-validation

the neurons in the hidden layer. The activation functions chosen have been the log-sigmoid for the neuron for the hidden layers, and the purelin function, respectively, for the neuron on the output layer. The instruction has been completed by using the Levenberg-Marquardt method, in the case of the Matlab application. Following the criterion of the minimization of the mean square error throughout the repeated trainings, there have been retained models for which m=31. The training parameters have default values, namely maximum epochs (1000), performance goal (0), minimum gradient (10^{-05}), maximum validation checks (6), multiplication factor (0.001), multiplication factor decrease ratio (0.1), multiplication factor increase ratio (10), maximum value of multiplication factor (10^{10}).

Table presented in Figure 6 shows the most effective (minimum mean square error, marked as MSE) obtained when training the networks in the case of each of the four rounds.

Round	Training MSE
1.	$6.0xe^{-3}$
2.	$2.8xe^{-2}$
3.	$3.0xe^{-2}$
4.	$7.9xe^{-4}$

Figure 6: The best MSE

It has been noticed that for each of the four neural models obtained by means of the combination of three subsets M_k, there has been obtained, through testing on the fourth test subset, very good results for the coordinates X, Y, ψ, θ, φ. The results of the four simulations are briefly shown in Figure 7 and in table of Figure 8.

The analysis of the results obtained shows that there appear problems when simulating the coordinate Z at the intersection of the direct trajectory with the obstacle, close to the latter. Outside the area close to the points AA' and CC', all the four simulations grant good results for coordinate Z as well.

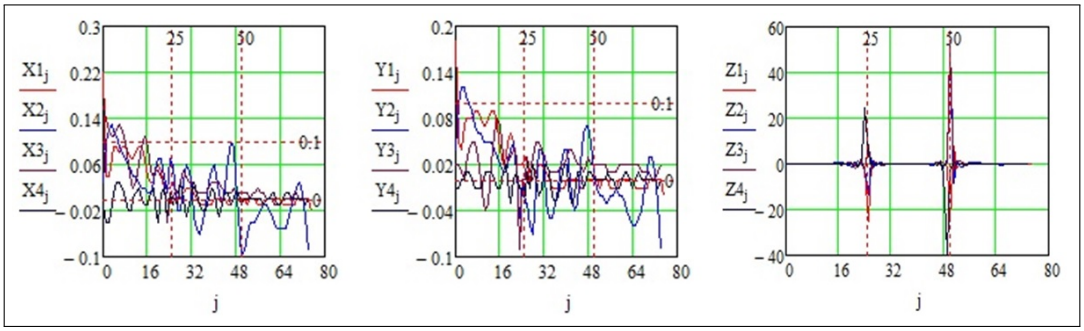

Figure 7: The errors of coordinates X, Y, Z

No. testing points	Cross validation											
j	M_1			M_2			M_3			M_4		
0	$\%X1_j$	$\%Y1_j$	$\%Z1_j$	$\%X2_j$	$\%X2_j$	$\%Z2_j$	$\%X3_j$	$\%Y3_j$	$\%Z3_j$	$\%X4_j$	$\%Y4_j$	$\%Z4_j$
1	0.22	0.18	-0.03	0.03	0.04	0.05	0.16	0.02	-0.08	-0.01	-0.01	0.02
2	0.04	0.05	0.05	0.10	0.10	0.07	0.14	0.02	-0.13	-0.04	-0.01	0.05
3	0.04	0.04	0.06	0.12	0.12	0.06	0.12	0.01	-0.17	-0.04	0.00	0.04
...												
21	0.06	0.06	0.01	0.07	0.04	-1.08	0.05	0.03	0.34	-0.04	-0.03	0.27
22	0.05	0.05	0.10	0.07	0.05	-0.72	0.04	0.04	-1.78	0.01	0.00	0.38
23	0.02	0.02	-0.47	0.04	0.01	1.58	0.03	0.03	-1.52	0.02	0.01	-1.08
24	0.02	0.02	0.83	0.01	-0.05	0.13	-0.04	-0.09	2.17	-0.03	-0.01	2.92
25	0.02	0.02	-2.49	0.07	0.00	-2.73	0.00	0.00	-3.40	0.01	-0.01	24.70
26	0.03	0.03	-24.97	0.06	-0.03	-13.56	0.00	0.00	8.70	-0.01	-0.02	3.85
27	0.02	0.02	1.26	0.03	-0.03	2.29	0.01	0.03	-3.00	0.00	0.01	-0.09
28	0.00	0.00	-0.64	-0.02	-0.06	-1.07	0.01	0.01	1.38	-0.01	-0.01	0.12
29	0.00	0.00	0.22	-0.02	-0.07	-0.15	0.03	0.03	0.12	0.03	0.02	-0.15
30	0.01	0.01	0.07	0.03	-0.01	0.40	0.05	0.05	-0.81	0.00	0.00	0.07
...												
46	-0.01	-0.01	-0.09	0.07	0.02	-0.57	0.01	0.02	0.21	0.01	0.01	-0.07
47	-0.01	-0.01	-0.33	0.10	0.06	0.39	0.01	0.01	0.33	-0.01	-0.01	-0.44
48	0.00	0.00	0.17	0.09	0.07	0.83	0.02	0.02	-0.41	0.00	0.00	1.05
49	0.00	0.00	0.78	0.04	0.07	-2.02	0.01	0.03	-0.41	0.01	0.01	-3.93
50	0.00	0.00	-1.49	-0.03	0.05	2.75	0.01	0.02	0.73	0.01	0.01	-39.31
51	0.00	0.00	-24.99	-0.10	0.01	47.89	0.01	0.02	51.64	-0.01	0.00	-3.01
52	0.00	0.00	7.55	-0.09	-0.01	-8.56	0.01	0.02	-3.20	0.00	0.00	0.64
53	-0.01	-0.01	-1.74	-0.06	-0.02	1.53	0.01	0.02	1.05	0.00	0.00	0.21
54	0.00	0.00	0.29	-0.04	-0.02	-1.41	0.00	0.02	1.52	0.00	0.00	-0.21
55	0.00	0.00	0.42	-0.04	-0.03	-1.50	0.01	0.02	-0.50	0.00	0.00	0.02
...												
74	0.00	0.00	0.01	-0.03	-0.03	0.15	0.00	0.01	0.00	-0.01	-0.01	0.04
75	0.00	0.00	-0.02	-0.09	-0.09	0.15	0.00	0.02	-0.02	0.00	0.00	-0.07
76	-0.02	-0.02	-0.05									
	RMSE for Z:		16.35203	RMSE for Z:		13.06999	RMSE for Z:		12.7729	RMSE for Z:		19.23435

Figure 8: Cross-validation results

In a subsequent stage, efforts have been made to improve the solution by the development of some simplified neural models. Thus, there have been considered the networks of the three-layer perceptron type, with 6 neurons in the input layer, corresponding to position (X, Y, Z, ψ, θ,φ) of the effector, and a single neuron in the output layer, corresponding to the coordinate of the joint q_3, which determines the Z coordinate. There has been applied the validation technique of the models, the 4-fold cross-validation. The training has been completed using the set of the examples shown in table 2; a remark that should be mentioned in the case being that only the values of the coordinate q_{3j} have been taken into account as output. This set has been divided into four subsets according to the rules described by sets M_k, k=1, ..., 4. For each of the four combinations of the sets M_k, k=1, ...,4, there have been realized four trainings of the network. Thus, during the training, there have been obtained mean square errors that belong to the interval $[10^{-10}, 10^{-1}]$. For each of the four rounds of training-validation corresponding to the four combinations of the sets M_k, k=1, ...,4, there has been determined the root mean square error (RMSE) of approximation of coordinate Z. This has been calculated as an overall mean of all the individual errors $Z_{j,l}^{(k)}$ for a given value k data, where l=1, ..., 4 corresponds to the four neural models obtained within each round. These values are shown in table of Figure 9.

Number of testing points j	Cross-validation															
	M_1				M_2				M_3				M_4			
0	$\Delta Z_{j,1}^{(1)}$	$\Delta Z_{j,2}^{(1)}$	$\Delta Z_{j,3}^{(1)}$	$\Delta Z_{j,4}^{(1)}$	$\Delta Z_{j,1}^{(2)}$	$\Delta Z_{j,2}^{(2)}$	$\Delta Z_{j,3}^{(2)}$	$\Delta Z_{j,4}^{(2)}$	$\Delta Z_{j,1}^{(3)}$	$\Delta Z_{j,2}^{(3)}$	$\Delta Z_{j,3}^{(3)}$	$\Delta Z_{j,4}^{(3)}$	$\Delta Z_{j,1}^{(4)}$	$\Delta Z_{j,2}^{(4)}$	$\Delta Z_{j,3}^{(4)}$	$\Delta Z_{j,4}^{(4)}$
1	-0,001	-0,041	-0,234	0,079	0,000	-0,119	0,461	-0,059	2,256	-1,048	-0,001	-1,409	1,060	0,013	0,064	-0,032
2	-0,001	-0,031	-0,216	0,077	0,000	-0,118	0,428	-0,058	2,166	-0,970	-0,001	-1,375	1,654	0,013	0,068	-0,030
3	-0,001	-0,022	-0,198	0,074	0,000	-0,116	0,402	-0,057	2,091	-0,899	-0,001	-1,343	2,211	0,013	0,071	-0,028
…																
22	0,000	-0,026	0,034	-0,267	0,000	0,044	-3,460	-0,039	1,740	-0,927	0,002	-0,369	-9,481	0,008	-0,029	0,001
23	-0,001	-0,030	-0,032	-0,461	0,000	0,572	-12,696	-0,038	1,555	-3,104	0,000	2,796	10,090	0,010	0,271	0,002
24	-0,002	-0,034	-0,531	-0,024	0,001	1,719	-1,811	-0,037	0,055	-9,585	-0,035	1,152	48,128	0,016	0,578	0,000
25	0,000	-0,038	-1,434	1,673	-0,102	-2,746	59,249	-0,093	59,838	19,992	-2,327	5,756	101,070	15,183	94,398	4,899
26	-14,981	-99,853	-99,944	-46,647	0,102	-111,156	-51,178	12,837	3,938	-20,809	7,291	22,060	-45,740	0,144	-0,949	0,229
27	-0,002	0,040	1,127	-1,091	0,000	8,894	3,049	-0,057	2,046	6,082	0,017	-5,243	-7,538	0,002	0,365	-0,004
28	0,000	0,036	0,562	0,356	0,000	-2,918	12,609	-0,056	1,783	1,850	0,000	-2,480	11,334	0,006	0,367	-0,004
29	0,000	0,032	0,213	0,367	0,000	-1,146	5,186	-0,055	1,587	0,354	-0,002	1,655	15,798	0,007	0,166	-0,003
30	0,000	0,029	0,013	0,185	0,000	-0,177	-2,233	-0,054	1,406	0,024	-0,002	2,654	12,724	0,006	0,090	-0,002
…																
48	0,000	0,017	-0,141	-0,089	0,000	-1,251	1,780	-0,027	0,324	0,283	0,011	-7,329	3,404	-0,003	-0,057	-0,004
49	0,000	0,021	0,127	-0,301	0,000	-1,382	13,452	0,041	0,418	8,121	-0,083	-13,537	-12,827	-0,008	0,566	-0,004
50	-0,001	0,024	1,079	-1,669	0,000	-4,614	-11,118	-0,904	2,237	-12,871	1,757	23,662	-44,820	-4,583	-100,176	0,705
51	-12,486	-178,302	-100,256	-116,688	170,280	103,936	99,415	156,207	112,027	39,528	23,061	-6,690	107,787	-0,221	-0,524	-0,688
52	0,005	-0,036	-1,026	2,379	0,001	10,627	10,384	1,315	-4,863	-5,776	-0,240	10,641	58,543	0,004	0,097	0,007
53	0,000	-0,031	-0,099	-0,055	0,001	-0,721	-16,056	-0,188	-0,233	2,797	-0,010	7,917	22,864	-0,003	0,104	0,007
54	0,000	-0,026	0,160	-0,262	0,001	-0,424	-6,344	-0,084	-0,005	4,455	-0,005	2,124	3,487	-0,006	0,100	0,006
…																
73	0,000	0,037	0,174	-0,320	0,000	-0,256	-0,582	-0,056	-0,489	1,381	-0,001	-0,517	3,082	-0,020	0,077	-0,006
74	0,000	0,029	0,170	-0,323	0,000	-0,259	-0,588	-0,057	-0,556	1,766	-0,001	-0,465	3,433	-0,021	0,078	-0,008
75	0,000	0,019	0,164	-0,325	0,000	-0,262	-0,593	-0,058	-0,643	2,236	-0,001	-0,404	3,745	-0,022	0,081	-0,009
76	-8,3E-05	0,0062	0,155032	-0,32785												
	RMSE=16,01783				RMSE=17,73955				RMSE=8,455625				RMSE=13,45884			

Figure 9: The overall RMSE

The analysis of the data in tables of Figures 8 and 9 shows that there is no significant difference between the errors of approximation of coordinate Z in the case of the two modelling approaches. As a conclusion of this cross-validation, it has been remarked that the method can be applied, but in order to avoid the obstacle it is necessary to envelope it (to cover the obstacle with a smoother surface). The following case studies play the role of evaluating the avoidance method by using the neural models in which the obstacle is enveloped.

3 Case study for an enveloped obstacle

In order to analyse the avoidance precision, based on the same number of training points, the obstacle with three types of envelopes (Figure 10) has been considered.

The point O_1 has the coordinates (500,500,200), while the point O_1' has the coordinates (500,500,450), the distance O_1O_1' being greater than the segment AA'. In order to reduce the number of the training examples, it is considered that ψ=θ=φ=0.

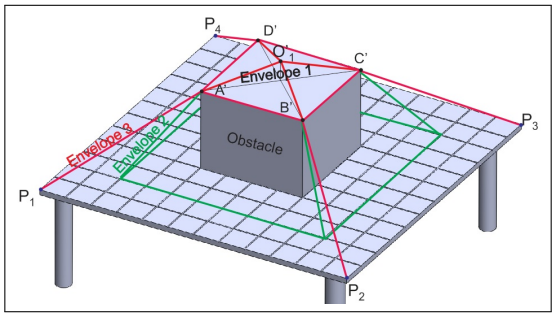

Figure 10: The obstacle envelopes

Training points	Effector coordinates							Joint coordinates ($q_{i,j}$, i=1,...,6; j=1,...,169)					
j	X_j	Y_j	Z_j	Z'_j	ψ_j	θ_j	φ_j	$q_{1,j}$	$q_{2,j}$	$q_{3,j}$	$q_{4,j}$	$q_{5,j}$	$q_{6,j}$
0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	200	200	200	200	0	0	0	20	20	20	20	20	20
2	250	200	200	200	0	0	0	25	20	20	20	20	20
3	300	200	200	200	0	0	0	30	20	20	20	20	20
4	350	200	200	200	0	0	0	35	20	20	20	20	20
5	400	200	200	200	0	0	0	40	20	20	20	20	20
6	450	200	200	200	0	0	0	45	20	20	20	20	20
7	500	200	200	200	0	0	0	50	20	20	20	20	20
8	550	200	200	200	0	0	0	55	20	20	20	20	20
9	600	200	200	200	0	0	0	60	20	20	20	20	20
10	650	200	200	200	0	0	0	65	20	20	20	20	20
11	700	200	200	200	0	0	0	70	20	20	20	20	20
12	750	200	200	200	0	0	0	75	20	20	20	20	20
13	800	200	200	200	0	0	0	80	20	20	20	20	20
14	200	250	200	200	0	0	0	20	25	20	20	20	20
...													
78	800	450	200	200	0	0	0	80	45	20	20	20	20
79	200	500	200	200	0	0	0	20	50	20	20	20	20
80	250	500	200	200	0	0	0	25	50	20	20	20	20
81	300	500	200	200	0	0	0	30	50	20	20	20	20
82	350	500	200	200	0	0	0	35	50	20	20	20	20
83	400	500	200	400	0	0	0	40	50	40	20	20	20
84	450	500	200	425	0	0	0	45	50	42.5	20	20	20
85	500	500	200	450	0	0	0	50	50	45	20	20	20
86	550	500	200	425	0	0	0	55	50	42.5	20	20	20
87	600	500	200	400	0	0	0	60	50	40	20	20	20
88	650	500	200	200	0	0	0	65	50	20	20	20	20
89	700	500	200	200	0	0	0	70	50	20	20	20	20
90	750	500	200	200	0	0	0	75	50	20	20	20	20
91	800	500	200	200	0	0	0	80	50	20	20	20	20
92	200	550	200	200	0	0	0	20	55	20	20	20	20
...													
168	750	800	200	200	0	0	0	75	80	20	20	20	20
169	800	800	200	200	0	0	0	80	80	20	20	20	20

Figure 11: Training data for Envelope 1

For the points in the work plane that do not belong to the obstacle proper or to its envelope, the coordinates of the joints are calculated based on the coordinates (X_j, Y_j, Z_j, ψ_j, θ_j, φ_j)

which define these points.

For the other points in the work plane which belong to the obstacle, or to its envelope, namely those which are at the intersection between the work plane and the envelope of the obstacle, the coordinates of the joints are calculated based on the coordinates of some corresponding points situated on the envelope.

For each of the three envelopes of the obstacle, there has been created and trained a neural network. Table of Figure 11 shows how the training data for Envelope 1 has been achieved.

The training of the neural network has been completed having as input signals coordinates X_j, Y_j, Z_j, ψ_j, θ_j, φ_j of the points (columns 1-3 and 5-7 in table of Figure 11), and as output signals, the coordinates of the joints $q_{i,j}$ (columns 8-13 in table of Figure 8) corresponding to points X_j,Y_j and Z'_j. The same has been applied in the case of Envelopes 2 and 3.

Testing points j	End-effector coordinates									Erors [%]								
	X_j	Y_j	Z_j	Envelope 1 Z'_j	Envelope 2 Z'_j	Envelope 3 Z'_j	ψ_j	θ_j	φ_j	Envelope 1			Envelope 2			Envelope 3		
										$\%Xr_j$	$\%Yr_j$	$\%Zr_j$	$\%Xr_j$	$\%Yr_j$	$\%Zr_j$	$\%Xr_j$	$\%Yr_j$	$\%Zr_j$
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	200	200	200	200	200	200	0	0	0	0.82	1.225	0.73	-0.53	-0.58	-0.87	0.635	0.71	0.24
2	250	250	200	200	200	250	0	0	0	0.24	-0.44	3.06	1.78	1.51	1.72	-0.40	-0.57	-1.09
3	275	275	200	200	200	275	0	0	0	1.54	0.61	2.88	1.18	0.98	3.43	-0.22	-0.33	-1.42
4	300	300	200	200	200	300	0	0	0	1.76	0.76	1.48	0.76	2.25	0.69	-0.05	-0.09	-1.99
5	325	325	200	200	200	325	0	0	0	0.98	0.10	1.61	-1.11	-1.11	6.18	-0.10	-0.15	-2.13
6	350	350	200	200	200	350	0	0	0	0.26	-0.48	2.36	-0.25	0.39	7.94	-0.09	-0.15	-0.60
7	375	375	200	200	300	375	0	0	0	0.06	-0.64	40.21	0.79	-0.59	5.00	0.04	0.06	-0.98
8	400	400	200	400	400	400	0	0	0	0.49	-0.28	-3.94	1.62	-0.39	-2.44	-0.23	-0.23	-0.89
9	425	425	200	412.5	412.5	412.5	0	0	0	1.03	0.14	-2.11	1.70	-0.04	-3.12	-0.22	-0.24	-1.13
10	450	450	200	425	425	425	0	0	0	0.98	0.15	2.17	1.28	0.59	-1.02	-0.02	0.02	-0.81
11	475	475	200	437.5	437.5	437.5	0	0	0	0.41	-0.16	3.75	0.72	1.30	0.43	-0.01	-0.02	-1.70
12	500	500	200	450	450	450	0	0	0	0.38	0.04	-0.71	0.48	1.31	-0.33	0.03	-0.01	-2.82
13	525	525	200	437.5	437.5	437.5	0	0	0	0.61	0.36	-1.22	0.23	0.88	1.10	0.01	0.05	-1.62
14	550	550	200	425	425	425	0	0	0	0.47	0.15	-0.49	0.23	0.33	0.29	-0.19	-0.14	-0.19
15	575	575	200	412.5	412.5	412.5	0	0	0	-0.06	-0.32	-1.54	0.43	0.29	-2.98	-0.04	0.00	-1.04
16	600	600	200	400	400	400	0	0	0	-0.04	-0.28	-5.27	0.55	0.50	-3.98	-0.08	-0.08	-2.81
17	625	625	200	200	300	375	0	0	0	0.38	0.04	24.69	0.47	0.36	3.32	-0.12	-0.13	-2.80
18	650	650	200	200	200	350	0	0	0	0.43	0.03	0.82	0.17	0.35	6.38	-0.26	-0.30	-2.48
19	675	675	200	200	200	325	0	0	0	0.23	-0.18	0.93	-0.41	0.22	3.40	-0.09	-0.13	-2.06
20	700	700	200	200	200	300	0	0	0	0.20	-0.28	0.96	-0.23	0.60	2.62	0.10	0.10	-0.37
21	725	725	200	200	200	275	0	0	0	0.20	-0.38	1.44	-0.46	0.60	-0.05	-0.09	-0.13	2.45
22	750	750	200	200	200	250	0	0	0	0.18	-0.44	0.63	0.14	1.18	4.58	-0.14	-0.19	-0.26
23	800	800	200	200	200	200	0	0	0	0.18	-0.09	0.44	-0.13	-0.92	-0.58	-0.06	-0.07	0.62

Figure 12: Errors for Envelope 1, 2, 3

In order to validate the method and the neural models, there have been considered as input data the coordinates of points (X_j, Y_j, Z_j, ψ_j, θ_j, φ_j) corresponding to the move of the effector on the direct trajectory P_1P_3 (columns 1-3 and 7-9 in table of Figure 12). Based on the coordinates of the joints $q_{i,j}$ simulated by the neural models corresponding to the envelopes, by means of relations (1)-(6), there have been calculated the effector coordinates and they have been graphically represented in Figure 13. The error obtained through the simulation on the neural model corresponding to each envelope is shown in table of Figure 12 (columns 10 - 18).

The results in table of Figure 12 show that Envelope 1 does not solve the problem at the borders of the intersection of the direct trajectory with the obstacle, as it is possible for the latter to be hit. This problem is solved in the case of Envelopes 2 and 3. The precision obtained by simulation for the models Envelope 2 and Envelope 3 can be accepted only in the case of some handling applications that do not require a high level of precision. This precision can be improved by increasing the number of training examples and by increasing their density in the workspace. In this research, there have been used only 169 training examples for the entire

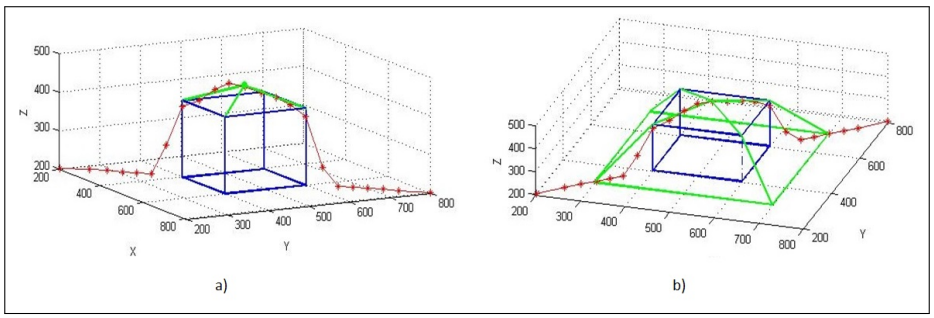

Figure 13: Effector coordinates for a. Envelope 1; b. Envelope 2

workspace, the distance between two successive points being 50 mm.

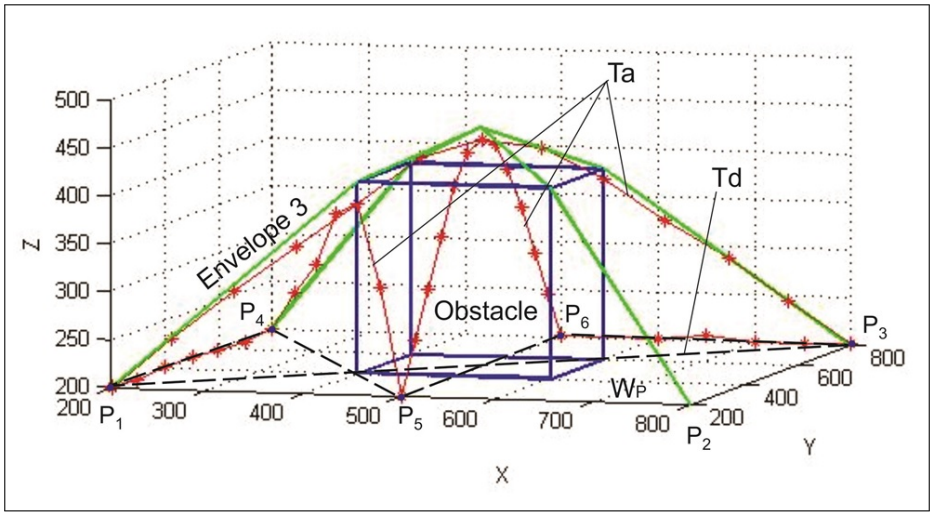

Figure 14: Direct and simulated trajectory

In order to test the obstacle avoidance in the case of programming several trajectories in the work plane (Envelope 3), there has been considered the robot's move on direct trajectories under the form of straight segments between points P_1-P_4-P_5-P_6-P_3-P_1 (Figure 14). The points P_1, P_4, P_5, P_6, P_3 are points in which the robot has to complete operations and where the effector has to position itself with an accuracy corresponding to the application. All the points P_1, ..., P_6 are situated in the work plane Wp. For the validation, there has been used the neural network for Envelope 3 and the results are shown in Figure 14.

The analysis of the results in Figure 14 reveals that, although the input data consisted of coordinates of some points on the direct trajectories Td, the neural model offers the joint coordinates that make the robot avoid obstacle when direct trajectory Td intersects the envelope of this obstacle. When the direct trajectory does not intersect the obstacle, the neural model NM provides data that leads the robot's effector on the direct trajectory Td.

4 Conclusions and future work

In order to validate the method by numerical research, there has been considered a robot with six degrees of freedom that has to move between two points in the workspace Wp. In plane Wp, there has been considered a parallelepiped-shaped object which has to be avoided. The research aimed to obtain several neural models of the robots kinematics based on a certain

method of creation of the set of training examples. Within the research, it has been noticed that the model obtained by the training of the robot so that it move on a trajectory that avoids, to the limit, the straight trajectory (the completed trajectory follows the obstacles outline) does not approximate well the move along the axis OZ at the frontier of the obstacle. There have been recorded relative errors of approximately 50% in such points.

In order to obtain an improved model, there have been considered three envelopes that dress the obstacle. For each envelope, there has been trained a neural network, each having the same number of training examples. The set of training examples has been generated by inverse kinematics analysis, considering a cloud of equally distanced points in the robot's work plane. The joint coordinates have been generated depending on the obstacle's envelope, in accordance with rule R described in the paper.

It has been noted that in all the three cases the neural model provides the joint coordinates that lead the end-effector on a bypass trajectory. In all the three cases studied, the bypass trajectory intersects the obstacle near the points that limit its superior base ($A'B'C'D'$). The positive deviation from the coordinate Z does not affect the obstacle avoidance, only the negative ones.

The analysis of the errors in the case of the coordinates (Xr_j, Yr_j, Zr_j) simulated by the neural network relating to the programmed coordinates (X_j, Y_j, Z_j) shows that the results are influenced by the choice of the obstacle envelope as follows:

- for Envelope 1:
  - the coordinates Xr_j and Yr_j are approximated with errors less than 2%
  - there are important errors in the case of coordinates Zr_j simulated by the neural model in the area close to the obstacle (40%, table of Figure 12, row 7)
  - close to points A' and C', the deviant trajectory intersects the obstacle ($\%Zr_j$ = -5.27%, table of Figure 12, row 16)
- for Envelope 2:
  - the coordinates Xr_j and Yr_j are approximated with errors less than 2.5%
  - the coordinates Zr_j simulated by the neural model are quite well approximated ($\%Zr_j$<8%, table of Figure 12, row 6)
  - close to points A' and C', the deviant trajectory intersects the obstacle ($\%Zr_j$= -3.98%, table of Figure 12, row 16)
- for Envelope 3:
  - the coordinates Xr_j and Yr_j are approximated with errors less than 1%
  - the coordinates Zr_j simulated by the neural network are well approximated ($\%Zr_j$<3%)
  - close to points A' and C', the deviant trajectory intersects the obstacle ($\%Zr_j$=-2.81%, table of Figure 12, row 16).

The problem of the intersection between the bypass trajectory and the obstacle can be solved by the choice of an envelope that exceeds the obstacle in all its points. Thus, for the case in point, if one chooses the parallelepiped $ABCDA'B'C'D'$ with sides 10% larger that the dimensions of the obstacle, the problem of the collision is going to be avoided. Another option is to increase the training examples number.

Future research aims to optimize the avoidance trajectories and to improve the position accuracy in the working points. The trajectory optimization will account for accuracy and

energy consumption refinements. The positioning accuracy in the working points will be made by choosing a denser cloud of points around them. Further research aims to study the method presented in this paper using other types of neural networks and make a comparative analysis of the results.

Bibliography

[1] Ciupan E.; Bojan I.(2009); Own Software Used in the Modelling of an Economic Supplying Activity, *Proceeding of the 5th International Conference on the Management of Technological Changes*, ISSN 1726-9679, (1): 209-212.

[2] Ciupan E. (2007); Software Designed for Modelling and Simulating Using Three-layer Neural Networks, *Annals of DAAAM 2008 & Proc. of the 19th International DAAAM Symposium*, ISBN 978-960-89832-7-4, (1): 275-276.

[3] Ciupan E. (2010); A Model for the Kinematical Analysis of a Six Degrees of Freedom Parallel Robot, *Proceedings of the 2010 IEEE International Conference on Automation, Quality and Testing, Robotics AQTR 2010*, ISBN 978-1-4244-6724-2, (1): 261-264.

[4] Ciupan E. (2012); Modelling a Handling Robot to Avoid an Obstacle, *Proc. of the 2012 IEEE International Conference on Automation, Quality and Testing, Robotics AQTR 2012*, ISBN 978-1-4673-0702-4, (1):391-395.

[5] Cong S.; Liang Y. (2009); PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems, *IEEE Trans. Ind. Electron*, ISSN 0278-0046, 56, (10): 3872-3879.

[6] Debbache G.; Bennia A.; Gola N. (2007); Neural Networks-Based Adaptive State Feedback Control of Robot Manipulators, *International Journal of Computers Communications & Control*, ISSN 1841-9836, 2(4):328-339.

[7] Feng Y.; Wanh Y.; Yang Y. (2012); Inverse Kinematics Solution for Robot Manipulator Based on Neural Network under Joint Subspace, *International Journal of Computers, Communications & Control*, ISSN 1841-9836, 7(3): 459-472

[8] Karray F.; Gueaieb W.; Al-Sharhan S. (2005); The Hierarchical Expert Tuning of PID Controllers Using Tools of Soft Computing, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B , 35(6):1283-1294

[9] Lee S. H. et al. (2007); Patent KR100752098, Robot System Based on Neural Network.

[10] Lewis F. L.; Jagannathan S.; Yesildirak A. (1998); *Neural Network Control for Robots Manipulators and Nonlinear Systems*, ISBN0748405968, Tylor & Francisc Inc. Bristol, 1998.

[11] Mann G. K. I.; Hu B-G.; Gosine R.G. (2001); Two-Level Tuning of Fuzzy PID Controllers, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, 31(2): 263-269.

[12] Min T. et al. (2012); Patent CN102346489, Pulse Neural Network Based Method for Controlling Object Tracking of Robot.

[13] Park K. T. et al. (2005); Patent KR20070072314, Method for Controlling Pose of Robot With Using Neural Network, Recording Medium Thereof, Apparatus for Controlling Pose of Robot With Using Neuron-Network and Robot Therewith.

[14] Pisla D. L.; Itul T. P.; Pisla A. (2007); Considerations Regarding the Geometrical Modelling of Parallel Mini-Manipulators With Triangle Platform, *Proc. of 11th Int. Research/Expert Conference Trends in the Development of Machinery and Associated Technology TMT 2007*, ISBN 9958-617-30-7: 607-610.

[15] Rigatos G. (2009); Model-based and model-free control of flexible-link robots: A comparison between representative methods, *Applied Mathematical Modelling*, 33(10): 3906-3925.

[16] Siciliano B.; Katib O. (2008); Springer Handbook of robotics, Springer-Verlag, Berlin Heidelberg, ISBN 978-3-540-23957-4.

[17] Uang H. J.; Lien C.C. (2006); Mixed H2/H & PID tracking control design for uncertain spacecraft systems using a cerebellar model articulation controller, *IEEE Proc.- Control Theory and Applications*, 3(1): 1-13.

[18] Yu D. L.; Chang T. K.; Yu D.W. (2005); Fault Tolerant Control of Multivariable Processes Using Auto-Tuning PID Controller, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B , 35 (1): 32-43.

[19] Yu W.; Rosen J. (2013); Neural PID Control of Robot Manipulators application to an Upper Limb Exoskeleton, *IEEE Transactions on Cybernetics*, 43(2): 673-684.

[20] Zhang P.Y.; Lu T.S.; Song L.B. (2004); RBF networks-based inverse kinematics of 6R manipulator, *International Journal of advanced manufacturing technology*, 26(1): 144-147.

[21] Zilouchian A.; Jamshidi M. (2001); Intelligent Control Systems using Soft Computing Methodologies, CRC Press LLC, ISBN 0-8493-1875-0.